



redhat.<sup>®</sup>

[www.europe.redhat.com](http://www.europe.redhat.com)



# Red Hat

**RHS333**

**Red Hat Enterprise Security: Network  
Services**

**RHS333-RHEL5-en-1-EMEA-20070604**

**Red Hat Europe, 10 Alan Turing Road,  
Guildford, Surrey. GU2 7YF.  
United Kingdom  
Tel: +(44)-1483-300169  
FAX: +(44)-1483-574944**

**RHS333**

**Red Hat Enterprise Security: Network  
Services**

**RHS333-RHEL5-en-1-EMEA-20070604**

## **Table of Contents**

### **RHS333 - Red Hat Enterprise Security: Network Services**

#### **RHS333: Red Hat Enterprise Security: Network Services**

Copyright	xi
Welcome	xii
Participant Introductions	xiii
Red Hat Enterprise Linux	xiv
Red Hat Enterprise Linux Variants	xv
Red Hat Network	xvi
Other Red Hat Supported Software	xvii
The Fedora Project	xviii
Classroom Network	xix
Notes on Internationalization	xx
Objectives	xxi
Audience and Prerequisites	xxii

#### **Unit 1 - The Threat Model and Protection Methods**

Objectives	2
Internet Threat Model	3
The Attacker's Plan	4
System Security	5
Service Availability	6
Information Leakage	7
Authentication and Trust	8
Input Validation Attack	9
Protection Mechanism	10
Packet Filters	11
Network Proxies	12
Local Security	13
TCP Wrappers and xinetd	14
PAM and SELinux	15
Service Security	16
ExecShield	17
End of Unit 1	18

#### **Unit 2 - Basic Service Security**

Objectives	20
Basic Host Security	21
SELinux	22
SELinux, continued	23
SELinux: Targeted Policy	25
SELinux: Management	27
SELinux Troubleshooting	29

Host-based Access Control	30
Netfilter/iptables	31
Review of TCP Wrappers	32
Access Control	33
Logging	34
Running Other Commands	35
banners	36
xinetd Configuration Review	37
/etc/xinetd.d Issues	38
Location Limits	39
Time Limits	40
Interface Limits	41
Usage Limits	42
flags and deny_time	43
Interaction Between Filters	44
End of Unit 2	45
<b>Lab 2: Basic Service Security</b>	<b>46</b>
Sequence 1: Controlling the default host-based firewall	47
Sequence 2: TCP wrappers	48
Sequence 3: xinetd SENSOR traps	49

## Unit 3 - Cryptography

Objectives	54
Network Security	55
Switches and Security	56
Cryptography	58
Cryptography	59
OpenSSL	60
Symmetric Encryption	61
Cryptographic Hashes	63
Message Authentication Codes	64
User Authentication	65
Asymmetric Encryption	66
Digital Signatures	67
Key Distribution	68
Digital Certificates	69
TLS/SSL Handshake	70
Generating a RSA Private Key	72
Generating a Certificate Signing request (CSR)	73
Creating a CA-signed Certificate	74
Certificate Expiration	75
Certificate Revocation	76
Managing a CRL	77
Operational Issues	78
GnuPG	79
End of Unit 3	80
<b>Lab 3: Cryptography</b>	<b>81</b>
Sequence 1: Setting up a private Certificate Authority	82

Sequence 2: Self-signed certificates	84
Sequence 3: Signing certificates	86
Sequence 4: Installing the public CA certificate in Mutt	87
Sequence 5: Revocation of a certificate	88
Sequence 6: Using gpg to exchange encrypted email	89
Sequence 7: Exercises	90

## Unit 4 - Logging and NTP

Objectives	101
Time Synchronization	102
Service Profile: NTP	103
Basic Design of NTP	104
Simple Client Configuration	105
Server Configuration	106
NTPv4 Authentication	107
restrict and Access Control	108
Service Profile: syslog	109
Weaknesses of syslog	110
Protecting Log Servers	111
End of Unit 4	112
<b>Lab 4: Time Synchronization and syslog</b>	<b>113</b>
Sequence 1: Configuring a basic NTP server	114
Sequence 2: Monitoring NTP servers	115
Sequence 3: Logging to a central log host	117

## Unit 5 - BIND and DNS Security

Objectives	121
Vulnerabilities	122
Resolutions	123
Types of Name Servers	124
Service Profile: BIND	125
DNS Security	126
Attacks on DNS	127
Address Match Lists and acl	128
TSIG: Transaction Signatures	129
Installing TSIG Keys	130
Using TSIG Security	132
Limitations of TSIG	133
Restricting Zone Transfers	134
Restricting Recursive Queries	135
Single Server Topology	136
Split Server Topology	137
Blocking All Queries	138
Bogus Servers and Blackholes	139
Views	140
Defining Views	141

view Example	142
version.bind	143
bind-chroot Package	144
Monitoring with logging	145
channel	146
category	147
logging Example	148
Dynamic Update Security	149
Setting up DDNS	150
End of Unit 5	151
<b>Lab 5: BIND and DNS Security</b>	<b>152</b>
Sequence 1: Improving named service security	153
Sequence 2: Securing zone transfers with TSIG	156
Sequence 3: Setting up views	158
Sequence 4: Challenge Projects	159

## Unit 6 - Network Authentication: RPC,NIS and Kerberos

Objectives	162
Vulnerabilities	163
Resolutions	164
Network-managed users	165
Account Management	166
What is Sun RPC	167
RPC Security Issues	168
NIS Overview	169
Service Profile: NIS	170
Weakness of RPC	171
Improving NIS Security	172
Improving NIS Security	173
Kerberos	174
Principals	175
Initial Authentication	176
Ticket Authentication	177
Service Profile: Kerberos	178
/etc krb5.conf	179
Installing a Master KDC	180
Kerberos and DNS	181
kdc.conf	182
kadm5.acl	183
Using kadmin	184
Installing Application Servers	185
Kerberos Clients	186
Debugging Kerberized Services	187
Kerberos Security	188
Preauthentication	189
Ticket Validation	190
Cross-Realm Trust	191
Kerberos Encryption	192

<b>End of Unit 6</b>	193
<b>Lab 6: Network Authentication: RPC, NIS, and Kerberos</b>	194
Sequence 1: Configuring the NIS server	195
Sequence 2: Improving NIS security	196
Sequence 3: Configuring the Kerberos KDC	198
Sequence 4: Configuring the Kerberos application server	200

## Unit 7 - Network File System

<b>Objectives</b>	208
Traditional NFS	209
The Traditional NFS Protocol	210
Service Profile: NFS	211
Vulnerabilities of NFSv2/3	212
Improving Security with Static Ports	213
Protecting Data Confidentiality in NFSv2/3	214
Controlling Name Mappings	216
Protecting Data Integrity in NFSv2/3	218
Improving Security with NFS4	219
The NFS4 Pseudo Filesystem	220
The NFS4 Pseudo Filesystem	221
RPC Security using GSSAPI	222
NFS4 Security Modes	223
Current Limitations	225
Example: Configuring an NFS4 Server	226
Example: Configuring an NFS4 Client	227
Troubleshooting NFS4	228
General NFS Client Security	229
Client Mount Options	230
Client Mount Options	231
Client Mount Options	232
End of Unit 7	233
<b>Lab 7: Network File System</b>	234
Sequence 1: Traditional NFSv3 export	235
Sequence 2: Prepare security and single-sign-on	236
Sequence 3: Add NFS4 exports	237
Sequence 4: Switch to Encrypted NFS4	238
Sequence 5: Bypassing Portmap	239
Sequence 6: Questions	240

## Unit 8 - OpenSSH

<b>Objectives</b>	256
Vulnerabilities	257
Resolutions	258
Service Profile: sshd	259
Server Configuration	260
SSH Protocols	261

Server Authentication	262
User Authentication	263
User Access Control	264
Login Messages	265
Logging Activity	266
Client Configuration	267
Client-side Server Authentication	268
Client-side User Authentication	269
Protecting Private Keys	270
authorized_keys Options	271
stunnel	272
Port Forwarding	273
X11 Forwarding	274
End of Unit 8	275
<b>Lab 8: OpenSSH Configuration</b>	<b>276</b>
Sequence 1: Public Key Authentication	277
Sequence 2: Port Forwarding	278

## **Unit 9 - Electronic Mail with Sendmail**

Objectives	283
Vulnerabilities	284
Resolutions	285
Server Topologies	286
User Mail Access	287
User Mail Privacy and Security	288
GnuPG or S/MIME?	289
Other Security Concerns	290
Service Profile: Sendmail	291
Server Security	292
Server Security: DoS	293
Server Security: File Permissions	294
smrsh	295
STARTTLS	296
Anti-Spam Mechanisms	297
/etc/mail/access	298
Authenticated Relay	299
DNS Blackhole Lists	301
End of Unit 9	302
<b>Lab 9: Sendmail</b>	<b>303</b>
Sequence 1: Initial Setup	304
Sequence 2: Set up and use a DNS blackhole list	305
Sequence 3: The access map and relaying	307
Sequence 4: Challenge Project	308

## **Unit 10 - Postfix**

Objectives	311
------------	-----

<b>Vulnerabilities</b>	<b>312</b>
<b>Service Profile: Postfix</b>	<b>313</b>
<b>Postfix Security Principles</b>	<b>314</b>
<b>Postfix Design</b>	<b>315</b>
<b>/etc/postfix/master.cf</b>	<b>317</b>
<b>Receiving Mail</b>	<b>318</b>
<b>Delivering Mail</b>	<b>319</b>
<b>Queues</b>	<b>320</b>
<b>Queues</b>	<b>321</b>
<b>postconf</b>	<b>322</b>
<b>Basic Configuration Review</b>	<b>324</b>
<b>Service Security</b>	<b>325</b>
<b>Postfix Security: DoS</b>	<b>326</b>
<b>Restricting Relaying</b>	<b>327</b>
<b>/etc/postfix/access</b>	<b>328</b>
<b>DNS Blackhole Lists</b>	<b>329</b>
<b>Procmail and SpamAssassin</b>	<b>330</b>
<b>Content Filtering</b>	<b>331</b>
<b>Postfix with SASL/TLS</b>	<b>332</b>
<b>Configuring SASL/TLS</b>	<b>333</b>
<b>End of Unit 10</b>	<b>334</b>
<b>Lab 10: Postfix</b>	<b>335</b>
<b>Sequence 1: Inspecting and tuning the default postfix configuration</b>	<b>336</b>
<b>Sequence 2: Tuning the default configuration</b>	<b>339</b>
<b>Sequence 3: Restricting Incoming Email</b>	<b>340</b>
<b>Sequence 4: Restricting relaying</b>	<b>341</b>
<b>Sequence 5: Setting up TLS and SASL authentication</b>	<b>342</b>

## **Unit 11 - FTP**

<b>Objectives</b>	<b>351</b>
<b>Vulnerabilities</b>	<b>352</b>
<b>Resolutions</b>	<b>353</b>
<b>The FTP Protocol</b>	<b>354</b>
<b>FTP Servers</b>	<b>355</b>
<b>Service Profile: vsftpd</b>	<b>356</b>
<b>Login Banners</b>	<b>357</b>
<b>Informational Capabilities</b>	<b>358</b>
<b>Logging Capabilities</b>	<b>359</b>
<b>Local Users</b>	<b>360</b>
<b>User/Group Access Control</b>	<b>361</b>
<b>Anonymous FTP</b>	<b>362</b>
<b>Anonymous FTP Uploading</b>	<b>363</b>
<b>Connection Restrictions</b>	<b>364</b>
<b>Host Access Restrictions</b>	<b>365</b>
<b>Other Useful Options</b>	<b>366</b>
<b>End of Unit 11</b>	<b>367</b>
<b>Lab 11: FTP</b>	<b>368</b>
<b>Sequence 1: Active vs. Passive FTP</b>	<b>369</b>

Sequence 2: FTP-only users	371
Sequence 3: Anonymous uploads	372

## **Unit 12 - Apache Security**

Objectives	377
Vulnerabilities	378
Resolutions	379
Service Profile: Apache	380
<Directory>	381
Apache Access Configuration	382
Flat File Authentication	383
Flat File Authentication	384
Managing Passwords	385
Kerberos Authentication	386
Options	387
Common Misconfigurations	388
Options FollowSymLinks	389
Options Indexes	390
Installing mod_ssl	391
SSL Virtual Hosts	392
CGI - Common Gateway Interface	393
Options ExecCGI	394
CGI Secure Programming	395
CGI With Unix Shell Script	396
SSI: Server Side Includes	397
SSI Security	398
suEXEC	399
End of Unit 12	400
<b>Lab 12: Apache Security</b>	<b>401</b>
Sequence 1: Install Apache and configure .htaccess files	402
Sequence 2: CGI Setup and suEXEC	404
Sequence 3: Configuring mod_SSL	406
Sequence 4: Decrypting Private Key	407
Challenge Sequence 5: Secure .htaccess	408

## **Unit 13 - Intrusion Detection and Recovery**

Objectives	415
Intrusion Risks	416
Security Policy	417
Detecting Possible Intrusions	418
Detecting Possible Intrusions	419
Monitoring Network Traffic	420
Monitoring Open Ports	421
Detecting Modified Files	422
Detecting Modified Files	424
Investigating and Verifying Detected Intrusions	426

Creating a Disk Image	427
Detecting and Defeating Backdoors	428
Detecting and Defeating Root Kits	429
Detecting and Defeating Root Kits	430
Recovering from an Intrusion	431
Reporting and Documenting the Intrusion	432
End of Unit 13	433
<b>Lab 13: Intrusion Detection and Recovery</b>	<b>434</b>
Sequence 1: Generate a md5 checksum database	435
Sequence 2: Install a trojaned software on your system	436
Sequence 3: Create a snapshot	437
Sequence 4: Use the rpm database to determine changes	438

# Introduction

## RHS333: Red Hat Enterprise Security: Network Services

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2894 or +1 (919) 754 3700.

## Copyright

- The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2007 Red Hat, Inc.
- No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.
- This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.
- If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please email [training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 866 626 2994 or +1 919 754 3700.

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Welcome

Please let us know if you have any special needs while at our training facility.

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <ttraining@redhat.com> or phone toll-free (USA) +1 (866) 826 2894 or +1 (919) 754 3700.

### Phone and network availability

Please only make calls during breaks. Your instructor will show you which phone to use. Network access and analog phone lines may be available; your instructor will provide information about these facilities. Please turn pagers to silent and cell phones off during class.

### Restrooms

Your instructor will notify you of the location of these facilities.

### Lunch and breaks

Your instructor will notify you of the areas to which you have access for lunch and for breaks

### In case of Emergency

Please let us know if anything comes up that will prevent you from attending.

### Access

Each facility has its own opening and closing times. Your instructor will provide you with this information.

## Participant Introductions

Please introduce yourself to the rest of the class!

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (819) 754 3700.

## **Red Hat Enterprise Linux**

- Enterprise-targeted operating system
- Focused on mature open source technology
- 18-24 month release cycle
  - Certified with leading OEM and ISV products
- Purchased with one year Red Hat Network subscription and support contract
  - Support available for seven years after release
  - Up to 24x7 coverage plans available

**5**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ttraining@redhat.com](mailto:ttraining@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 764 3700.

The Red Hat Enterprise Linux product family is designed specifically for organizations planning to use Linux in production settings. All products in the Red Hat Enterprise Linux family are built on the same software foundation, and maintain the highest level of ABI/API compatibility across releases and errata. Extensive support services are available: a one year support contract and Update Module entitlement to Red Hat Network are included with purchase. Various Service Level Agreements are available that may provide up to 24x7 coverage with guaranteed one hour response time. Support will be available for up to seven years after a particular release.

Red Hat Enterprise Linux is released on an eighteen to twenty-four month cycle. It is based on code developed by the open source community and adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors such as Dell, IBM, Fujitsu, BEA, and Oracle. Red Hat Enterprise Linux provides a high degree of standardization through its support for five processor architectures (Intel x86-compatible, AMD AMD64/Intel 64, Intel Itanium 2, IBM POWER, and IBM mainframe on System z).

## Red Hat Enterprise Linux Variants

- Two Install Sets available
- Server
  - Red Hat Enterprise Linux Server
  - Red Hat Enterprise Linux Advanced Platform
- Desktop
  - Red Hat Enterprise Linux Desktop
  - Workstation Option
  - Multi-OS Option

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ttraining@redhat.com](mailto:ttraining@redhat.com)> or phone toll-free (USA) +1 (800) 826 2994 or +1 (918) 754 3700.

Currently, on the x86-compatible architecture, the product family includes:

*Red Hat Enterprise Linux Advanced Platform:* the most cost- effective server solution, this product includes support for the largest x86-compatible servers, unlimited virtualized guest operating systems, storage virtualization, high-availability application and guest fail-over clusters, and the highest levels of technical support.

*Red Hat Enterprise Linux:* the basic server solution, supporting servers with up to two CPU sockets and up to four virtualized guest operating systems.

*Red Hat Enterprise Linux Desktop:* a general purpose client solution, offering desktop applications such as the OpenOffice.org office suite and Evolution mail client. Add-on options provide support for high-end development workstations and virtualization.

## Red Hat Network

- A comprehensive software delivery, system management, and monitoring framework
  - *Update Module*: Provides software updates
    - Included with all Red Hat Enterprise Linux subscriptions
  - *Management Module*: Extended capabilities for large deployments
  - *Provisioning Module*: Bare-metal installation, configuration management, and multi-state configuration rollback capabilities
  - *Monitoring Module* provides infrastructure health monitoring of networks, systems, applications, etc.

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

Red Hat Network is a complete systems management platform. It is a framework of modules for easy software updates, systems management, and monitoring, built on open standards. There are currently four modules in Red Hat Network; the Update Module, the Management Module, the Provisioning Module, and the Monitoring Module.

The Update Module is included with all subscriptions to Red Hat Enterprise Linux. It allows for easy software updates to all your Red Hat Enterprise Linux systems.

The Management Module is an enhanced version of the Update Module, which adds additional features tailored for large organizations. These enhancements include system grouping and set management, multiple organizational administrators, and package profile comparison among others. In addition, with RHN Proxy Server or Satellite Server, local package caching and management capabilities become available.

The Provisioning Module provides mechanisms to provision and manage the configuration of Red Hat Enterprise Linux systems throughout their entire life cycle. It supports bare metal and existing state provisioning, storage and editing of kickstart files in RHN, configuration file management and deployment, multi-state rollback and snapshot-based recovery, and RPM-based application provisioning. If used with RHN Satellite Server, support is added for PXE bare-metal provisioning, an integrated network tree, and configuration management profiles.

## Other Red Hat Supported Software

- Global Filesystem
- Directory Server
- Certificate Server
- Red Hat Application Stack
- JBoss Middleware Application Suite

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

Red Hat offers a number of additional open source application products and operating system enhancements which may be added to the standard Red Hat Enterprise Linux operating system. As with Red Hat Enterprise Linux, Red Hat provides a range of maintenance and support services for these add-on products. Installation media and software updates are provided through the same Red Hat Network interface used to manage Red Hat Enterprise Linux systems.

*Global Filesystem:* an open source cluster file system appropriate for enterprise deployments, allowing servers to share a common pool of storage.

*Directory Server:* an LDAP-based server that centralizes directory storage and data distribution, such as user and group data.

*Certificate Server:* identity management software, using the Red Hat Directory Server as its back-end LDAP data repository.

*Red Hat Application Stack:* the first fully integrated open source stack, supplying everything needed to run standards based web applications, including Red Hat Enterprise Linux, JBoss Application Server with Tomcat, JBoss Hibernate, and a choice of open source databases: MySQL or PostgreSQL, and Apache Web Server.

*JBoss Middleware Application Suite:* a suite of applications that provide a complete middleware solution.

For additional information, see the following web pages:

- *Global Filesystem:* <https://www.redhat.com/solutions/gfs/>
- *Directory Server:* <https://www.redhat.com/solutions/directoryserver/>
- *Red Hat Application Stack:*  
<https://www.redhat.com/solutions/rhappstack/>
- *JBoss Middleware Application Suite:* <https://www.redhat.com/jboss/>

## The Fedora Project

- Red Hat sponsored open source project
- Focused on latest open source technology
  - Rapid four to six month release cycle
  - Available as free download from the Internet
- An open, community-supported proving ground for technologies which may be used in upcoming enterprise products
- Red Hat does not provide formal support

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2694 or +1 (919) 754 3700.

The Fedora Project is a community supported open source project sponsored by Red Hat intended to provide a rapidly evolving, technology-driven Linux Distribution with an open, highly scalable development and distribution model. It is designed to be an incubator and test bed for new technologies that may be used in later Red Hat enterprise products. The basic Fedora Core distribution will be available for free download from the Internet.

The Fedora Project will produce releases on a short four to six month release cycle, to bring the latest innovations of open source technology to the community. This may make it attractive for power users and developers who want access to cutting-edge technology and can handle the risks of adopting rapidly changing new technology. Red Hat does not provide formal support for the Fedora Project.

For more information, visit <http://www.fedoraproject.org>.

## Classroom Network

	Names	IP Addresses
Our Network	example.com	192.168.0.0/24
Our Server	server1.example.com	192.168.0.254
Our Stations	stationX.example.com	192.168.0.X
Hostile Network	cracker.org	192.168.1.0/24
Hostile Server	server1.cracker.org	192.168.1.254
Hostile Stations	stationX.cracker.org	192.168.1.X
Trusted Station	trusted.cracker.org	192.168.1.21

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 626 2994 or +1 (919) 754 3700.

## Notes on Internationalization

- Red Hat Enterprise Linux supports nineteen languages
- Default language can be selected:
  - During installation
  - With **system-config-language**
    - System->Administration->Language
- Alternate languages can be used on a per-command basis:

```
$ LANG=en_US.UTF8 date
```
- Language settings are stored in **/etc/sysconfig/i18n**

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

Red Hat Enterprise Linux 5 supports nineteen languages: English, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Korean, Malayalam, Marathi, Oriya, Portuguese (Brazilian), Punjabi, Russian, Spanish and Tamil. Support for Assamese, Kannada, Sinhalese and Telugu are provided as technology previews.

A system's language can be selected during installation, but the default is US English. To use other languages, you may need to install extra packages to provide the appropriate fonts, translations and so forth. These can be selected during system installation or with **system-config-packages** (Applications->Add/Remove Software).

The currently selected language is set with the **LANG** shell variable. Programs read this variable to determine what language to use for output:

```
[student@stationX ~]$ echo $LANG  
ru_RU.UTF8
```

A system's default language can be changed with **system-config-language** (System->Administration->Language), which affects the **/etc/sysconfig/i18n** file.

Languages with non-ASCII characters may have problems displaying in some environments. Kanji characters, for example, may not display as expected on a virtual console. Individual commands can be made to use another language by setting **LANG** on the command-line:

```
[student@stationX ~]$ LANG=en_US.UTF8 date  
Thu Feb 22 13:54:34 EST 2007
```

Subsequent commands will revert to using the system's default language for output.

SCIM (Smart Common Input Method) can be used to input text in various languages under X if the appropriate language support packages are installed. Type **Ctrl-Space** to switch input methods.

## Objectives

- Learn how to improve and maintain the security of typical network services provided on a Red Hat Enterprise Linux system
- Focus on mechanisms at the Transport layer or higher in the OSI network model
  - Securing services
  - Cryptography
  - Intrusion Detection and prevention

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2904 or +1 (919) 754 3700.

### *Scope of RHS333*

In this class focus will be on improving the security of network services through mechanisms at the service level. This course will not focus on using firewalls or VPNs to protect the network. These approaches will be covered in another course, RHS342 - Developing Red Hat Firewall Solutions.

This course is not designed to cover it all. There are many areas within security that are interesting and would merit further investigation. Without going too deep into these topics, a few of them will be covered lightly. While learning about firewalls and VPN is important, if those mechanisms are broken, services are vulnerable to attacks. A very real, but often neglected problem is how we deal with services. The emphasis of this course is therefore how to strengthen the security of these.

Another topic that will be covered is encryption. This will not be an in depth look at encryption, but rather give system administrators an insight when to use it and what algorithms to use.

While trying to prevent attackers from breaking into our machines is important, it is impossible to make a totally secure system. At the end of this manual, some tips on how to detect intrusion will be discussed. A discussion on how to make it difficult for attackers will also be featured.

## Audience and Prerequisites

- Audience: System administrators, consultants, and other IT professionals
- Prerequisites: RH253, RH300, RHCE certification or comparable work experience; knowledge of basic configuration of the network services covered in the course

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

# **Unit 1**

## **The Threat Model and Protection Methods**

**1-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Objectives

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

- Basic attack on the system
- Simple measures for attacks on host security
- Protection methods

1-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc., or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Internet Threat Model

- End systems are assumed to be secure
- The network is assumed to be insecure
  - An attacker may read any packet sent
  - An attacker may alter or block any packet sent, and may insert forged packets
- There is a desire to
  - Avoid compromise of the end systems
  - Protect the communication channel

1-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cctraining@redhat.com](mailto:cctraining@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (619) 754 3700.

### *Assumptions about the threat*

In this course, an attempt to improve the security of network services is done. In order to do this, it is often a good idea to make an assumption of the threat model as a starting point.

The modern Internet threat model usually makes the assumption that end systems participating in a communication exchange are not compromised, but that the network itself is insecure. The attacker does not control one of the end systems, but can potentially read, change, or block any network packet sent, and may insert forged packets into the network containing arbitrary headers and data payload.

The goal will be to maintain the security of the end systems, and to authenticate and protect the integrity and confidentiality of communications between systems, on a network assumed to be hostile.

Some protocols that are necessary to work violate these assumptions. For example, older versions of NFS makes the assumption that the network is secure, which makes that service much harder to protect from the point of view of this threat model.

## The Attacker's Plan

- Gather intelligence on the target system
- Identify vulnerabilities of the target
- Gain any access to the target required
- Use a vulnerability to successfully compromise the system
- Avoid retaliation for the attack

1-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 764 3700.

### *Outline of a compromise*

A successful attack proceeds through several stages. In the first stage, the attacker identifies potential targets and begins to gather as much information as possible about them and their environment. Once this information is collected, the attacker analyzes it and identifies any potential security vulnerabilities the target might have. These initial stages may involve passive data collection, or the attacker may actively probe the target.

Once the attacker has identified a weakness in the target, any special access that is needed to exploit the weakness must be gained. In a man-in-the-middle attack, this may involve causing network traffic to pass through a machine under the attackers control. For a system protected by a firewall, this may involve bypassing the firewall. When trying to gain access to the root account on a remote system, this may involve gaining access to a non-privileged user account. Therefore, an actual attack may actually consist of a number of small individual attacks to gain consecutively greater access.

When the required access has been gained, the attacker uses the vulnerability to successfully compromise the target. The communication channel may be hijacked, a bug in a set-uid root program may be used to gain superuser access, and so on.

Finally, the attacker hopes that the intrusion will not be detected. The damage is done, logs may be erased, root kits may be installed to hide evidence of the intrusion, back doors may be installed to get in again later, before the attacker leaves.

## System Security

- End system compromises include
  - Attacks on system availability
  - Unauthorized access
  - Inappropriate use by authorized users
- These compromises may involve
  - Exploiting flawed or misconfigured software
  - Exploiting protocol weaknesses
  - Exploiting trust relationships

1-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Service Availability

- Applications must remain available
- Denial of Service (DoS) attacks
  - May simply be used to disrupt service
  - May be used so attacker can spoof clients
- Monitoring is critical
  - Service status and network activity

1-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

### Service Availability

A service must be available when it is needed or it is not secure. The simplest form of attack on a service is to make it unavailable. This type of attack is referred to as a *denial of service* (or “DoS”) attack. This sort of attack may exploit a bug in the service that causes it to crash, or it may crash a supporting service or the operating system of the server on which it is running. A very common form of denial of service attack against network services is a *network flow attack*, in which the attacker transmits so much network traffic at the target server that all available bandwidth is consumed. The effective result of the denial of service attack is that the service is unable to communicate with legitimate users.

Denial of service attacks may simply be mounted in spite, to take a particular server off-line. In some cases, a denial of service attack may be intended to disable the server so that a rogue server controlled by the attacker may pose as that machine as part of another attack.

One particularly difficult form a denial of service attack is the *distributed denial of service* (or “DDoS”) attack. In this case, the attacker targets the victim from a large number of machines scattered around the Internet. Frequently, the hostile machines have themselves been compromised by the attacker in order to use them in the attack.

From a service-based point of view, defenses against denial of service attacks include keeping software up to date and monitoring service availability. The network administrator should also monitor the network for unusual activity or traffic. The Linux kernel provides protection against some forms of DOS attacks.

## Information Leakage

- Services leak information
  - Banner messages (software and version)
  - Hosts or accounts that may use the service
  - Resources that are available
- This information may be used by an attacker to prepare an intrusion attempt

1-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[trainings@redhat.com](mailto:trainings@redhat.com)> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 784 3700.

### *Information Leakage*

Many services leak information about their configuration and parameters to remote users, sometimes intentionally and sometimes not. This information can provide useful intelligence to an attacker prior to an attack. There are many different ways in which network services can leak information:

Service banner messages that are printed out prior to authentication can expose details of the software package being used to provide the service and its current revision number. This could be useful if a particular version of that software package is vulnerable to an attack. For example, the contents of `/etc/issue.net` are displayed by `in.telnetd` prior to authentication. In the default configuration, this indicates the machine is running Red Hat Enterprise Linux, including version, kernel, and which processor architecture!

Information about hosts or accounts authorized to use the service may also be leaked. For example, the `showmount` command can be used to ask a remote NFS server which exports it is providing and which hosts may access them.

In some cases, once a service has been accessed, details of its configuration can be dumped by the remote system. This was the case with the FTP protocol's `STAT` command. This may also suggest vulnerabilities to an attacker.

There are many subtle ways in which services may leak information about themselves and the server. In general, minor information leakage does not in itself compromise the security of a machine. However, it may make it easier for an attacker to plan a successful attack.

## Authentication and Trust

- Secure services need to authenticate client identities and authorize access
- Weak authentication
  - Credential capture attacks
  - Leads to mis authentication, compromise
- Inappropriate trust
  - Stolen data or authentication credentials
  - Compromise of one trusted host can lead to compromise of many hosts

1-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ctraining@redhat.com](mailto:ctraining@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *Authentication and Authorization*

Secure services need to control access and only grant it to legitimate users. *Authentication* is how the host determines that a user or client is who they claim to be. *Authorization* is what that user or client is permitted to do. So if the user joe tries to log into a system and gives the right password, he is authenticated. But if it is 13:00 and PAM has been configured to only let joe log on during business hours, he will not be authorized to access the system and the login will fail. Authorization policies indicate what access an authenticated user is trusted with.

One way an attacker may get access to a system is by exploiting weak authentication or inappropriate trust relationships. A weak authentication method can be spoofed easily. One type of attack is a credential capture attack, in which the attacker somehow obtains valid credentials that allow him to pose as a legitimate user. This often involves using network sniffers to monitor the network for clear-text user names and passwords, cookies, or tickets that may be replayed to the service to gain access. One example of a protocol vulnerable to credential capture is telnet.

Attackers may also exploit inappropriate trust relationships to gain access. An attacker with unprivileged user-level access may find a way to exploit poor or misconfigured file permissions to steal tickets or credentials of other users from the local file system. If a service on one host trusts all users on another host, then if the other host is attacked both may be compromised. If many hosts trust the compromised host, they all may be compromised.

## Input Validation Attack

- Responsible for many remote exploits
- Program fails to properly check user input for validity
- Buffer overflows
  - More data placed in buffer than expected
  - Crashes program or executes arbitrary code
- Format string attacks
  - User data used in an insecure way

1-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

### *Trusting the Malicious User*

Input validation attacks are one of the most common ways that network services are compromised. This class of attacks is generally caused by bugs in the network services themselves. Network services are often very complex programs, written by human beings. If there is one lesson that has been learned about secure software development, it is that complex systems are much harder to debug and secure than simple ones. Therefore, bugs exist and can be hard to find.

A program vulnerable to an input validation attack does insufficient checking of user input before processing it. The attacker takes advantage of this bug by sending carefully crafted code to the program that will cause the program to do something the programmer never intended.

One common form of input validation attack is the *buffer overflow attack*. In this attack, the attacker sends data to the program as part of a normal operation. However, this data is crafted to be larger in size than normal data would be. If the program does not check the size of the data first, it may attempt to place it in a buffer or fixed-size array too small to hold the data. This causes part of the program memory to get overwritten by the data. Normally, a mistake of this kind would crash the program. A more clever or better prepared attacker may be able to overwrite the execution stack, and cause the program to run arbitrary code.

Another form of input validation attack is the *format string attack*. In this attack, the program is again fed unexpected data. The user data is then used as the input of a function call or other command without checking it for safety first. Specifically, a format string attack is possible if unchecked user input is used directly in a printf() function. The general problem of using unverified user data in a command is demonstrated by the large number of attacks on insecure CGI scripts.

The impact of service bugs can be limited by auditing service code, running services as unprivileged users, limiting the number of services running on a host, and limiting which clients can access those services.

## Protection Mechanism

- Firewalls
  - Netfilter
  - Proxy
  - Local packet filters
- TCP Wrappers
- Xinetd
- Pluggable Authentication Modules
- Security Enhanced Linux
- Service specific
- Application hardening

1-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Protection mechanisms*

When a user is trying to access a service a whole range of security mechanisms will have to be accessed in order to determine whether the request should be granted or not.

1. Check the packet filter firewall or application proxy
2. If system has a local packet filter, check this
3. If service is linked against libwrap.so, check TCP Wrappers
4. If service is controlled by xinetd, run appropriate checks
5. Let Pluggable Authentication Modules make a decision on whether access is granted
6. Verify permissions using Security Enhanced Linux, if it is enforcing.
7. If service is not linked against libwrap.so, service specific configuration must now be consulted. *This is the topic of this manual.*
8. ExecShield can protect against Buffer-Overflows.

## Packet Filters

- Low level security
- Used as firewalls

1-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2804 or +1 (919) 734 3700.

### *Packet filtering firewalls*

Firewalls are designed to catch packets on the way in and out of a network. There are two kinds of firewalls, packet filters and proxy servers. These should be used together for best protection.

Packet filters are low level filters that looks at the headers of a packet going through a machines kernel. It looks at things like if the packet is malformed, where the packet comes from, where it is going, protocols, interfaces and so on. It matches these specifics against a set of rules. Based on this, it can determine whether the packet should be allowed through.

There should never be more than one route out of a network, which makes packet filters ideal. It also makes it easier to control network traffic, since only one node needs to be monitored. As mentioned above, we only expect the header of a packet, so it will be efficient performance-wise. On the other hand, configuration of firewalls may be complex and syntax can be difficult to learn. They can also be difficult to test, they either work or they don't, but why is often a mystery. Packet filters have often only rudimentary capabilities, so logging, NAT, connection tracking may not be available (current version of Red Hat Enterprise Linux firewall, netfilter, has all of these capabilities).

Packet filters have no way of inspecting payload. This can cause problems, because we may actually allow malicious or private information through, because the packet filter only checks the header of the packet. To combat this problem, we can let the packets run through a proxy server. This will only look at the content of the packet, so if the packet filter let it through, the proxy can now look at content, and filter out things like SPAM, viruses, worms and the like.

## Network Proxies

- Often used in DMZ
- Good at filtering content

1-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (800) 626 2994 or +1 (919) 754 3700.

### *Proxy filters*

While packet filters are excellent for firewalling, their biggest problem is that they cannot inspect payload. To do this, we need an application level firewall, a proxy server that acts as a man in the middle. The client on one side send a message through the proxy firewall to another machine. The firewall will inspect the payload to determine whether the packet is allowed through.

Proxy servers can inspect sender/recipient (DNS or maybe ip-address), user authentication, valid protocol usage, content and so on. There are many advantages to this, the rule-set can be complex (who is accessing what service, from where, at what time, is the content legal etc.) The content can be cached and there are advanced logging options. There is also no ip-forwarding, so packets cannot accidentally be passed through.

The disadvantage of a proxy firewall is that it needs to understand the protocol used. Only a few protocols are supported. Clients must often be reconfigured to send traffic through the proxy, and with all the things going on, a high demand is put on hardware.

## Local Security

- Why do local security?
  - Firewalls does not stop intruders from within
  - Firewalls cannot offer perfect protection
  - Firewalls cannot filter content like the application can

1-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 784 3700.

*Single host security = Bastion host?*

While firewalling of a network is extremely important, it is important to understand that many of the attacks against machines comes from the inside. Such attacks might not be malicious per se, but they open up holes in your network. It may be a user wanting to get through the firewall, installs a modem and opens up the entire network through this line. Or a user might have scribbled their password on a postit note and hung that on the monitor. Anyone walking by, even outsiders, can see this and later use it. The list of possibilities are endless.

Firewalls will protect your systems, provided they are set up correctly. And even then, there are no guarantees that it will not let an attacker through. Firewalls cannot be trusted as an only mean of protection.

Proxy firewalls does not forward packets in the kernels but instead hand it through a filter from one network card to another. It can inspect payload, hence give a more detailed filter tool. However, it is very costly doing it here, so data is usually not inspected here. That is left to the application itself to filter.

External firewalls can do just so much. By letting a single machine do part of the job, filtering can be more advanced. It is common to use a combination, however this course will not deal much with the external security.

## TCP Wrappers and xinetd

- TCP Wrappers
  - Test services linked against libwrap.so
  - Powerful for logging and access control
- xinetd
  - Access control for services under xinetd's control

1-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2064 or +1 (919) 754 3700.

### *Control through wrappers*

After setting up firewalls, it may be an idea to protect individual machines on the inside. There are two mechanisms for doing this, use of TCP Wrappers and xinetd.

TCP Wrappers are generic configuration files, used by services linked against libwrap.so. When a user attempts to access a service, TCP Wrappers will now check whether certain criteria has been met. If so, the user is connected with the service. If not, it will be rejected, possibly writing information to log files.

Xinetd controls service that are not frequently used. If the TCP Wrappers allow access to xinetd (xinetd is linked against libwrap.so), the service specific configuration files will decide whether traffic is actually let through or not.

## PAM and SELinux

- Pluggable Authentication Modules
  - Control access to individual services
    - Time
    - Usage limits
    - Location
- Security Enhanced Linux
  - Mandatory Access Control

1-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3780.

### *Access Control*

Pluggable Authentication Modules (PAM) are used to determine whether a user should have access to a service or not. It calls a set of libraries that must be passed in order to get access. PAM set up rules on how to interpret the results. It can determine whether a user should get access to a service based on whether the user managed to type in correct password, where the user attempts to log in from, time of day, resources already used and more.

Security Enhanced Linux set up control of services to protect the system from being broken into. In traditional Linux, there are a set of permissions that is assigned to files and programs. While these give a certain security, they give an intruder a great deal of potential power. SELinux on the other hand, demands that an attacker must hold special credentials in order to use services or access files. If the attacker does not have these credentials, then access will be denied. It can be compared to running services in chrooted jails, the service can only access a set of defined files, and an intruder with access to that service, will not have access to anything else.

## Service Security

- Control mechanisms not available to the entire system
- Specialized security

1-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *Securing applications*

While setting up firewalls and general protection "it being a packet filter or an application proxy, tcpwrappers, xinetd, SELinux or PAM", there is a need for protection of services that such generic tools cannot provide. It is therefore a special need to look at the specific services to determine what needs to be done to grant access to authorized users, while protecting the service from intrusion.

This manual is focused on single host integrity and security of communication between hosts. It will look at host based packet filters and transport layer encryption. Network firewalls, filtering proxies and network layer encryption is taught in RHS342 - Developing Red Hat Firewall and VPN solutions.

## ExecShield

- Segment Limits
- NX/XD-Technology
- `sysctl kernel.exec-shield`

1-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[certifying@redhat.com](mailto:certifying@redhat.com)> or phone toll-free (USA) +1 (866) 826 2984 or +1 (812) 754 3700.

### *ExecShield*

The ExecShield feature in Red Hat Enterprise Linux supports two technologies that protect applications from being compromised by most buffer exploits. The goal of these features is to prevent code that is maliciously deposited in data areas of an application from being executed. These features -*NX/XD* and *Segmentation*- use different techniques to achieve the same result.

#### *The Segment Limit Approach*

The effect of applying segment limits is that the first N megabytes of the virtual memory of a process are executable while the remaining virtual memory is not. The operating system kernel selects the value of N. With such a segment limit in place, the operating system must make sure that all program code is located below this limit while data, especially the stack, should be located in the higher virtual memory addresses. When a violation of the execution permission happens, the program triggers a segmentation fault and terminates, this behavior is identical to when a program tries to violate read or write memory permissions.

#### *NX/XD Technology*

Intel and AMD extended their processors architecture by adding a No eXecute (AMD) or eXecute Disable (Intel) permission to the set of existing memory permissions. As with the existing read and write memory permissions, the kernel can control No eXecute/eXecute Disable permissions of the programs' virtual memory. Using NX technology is a more fine-grained approach than the previously mentioned segment limits approximation. There is one caveat concerning NX/XD technology. As more permission bookkeeping information is required, this only works in the PAE 64 bit pagetable format (the PAE technology allows 32-bit x86 processors to use more than 4 GB of physical memory). PAE pagetables are not supported by all x86 processors.

You can enable ExecShield for marked programs by calling `sysctl -w kernel.exec-shield=1` and `sysctl -w kernel.exec-shield=2` for all programs. `execstack` is a program which sets, clears, or queries executable stack flag of libraries and binaries. `kernel.exec-shield=1` is set by default.

## **End of Unit 1**

- Questions and Answers
- Summary
  - Basic types of attacks
  - Simple steps to improve host security
  - Protection mechanisms

**1-18**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

## **Unit 2**

### **Basic Service Security**

**2-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (888) 626 2994 or +1 (919) 754 3700.

## Objectives

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

- SELinux
- Kernel packet filter
- TCP Wrapper
- xinetd

2-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 828 2994 or +1 (919) 754 3700.

## Basic Host Security

- Run only necessary network services
  - Every extra service on a host is another possible vulnerability
  - Separate servers by function
- Keep software up to date
  - Use yum or Automatic Updates from Red Hat Network to stay current
- Limit access to critical servers

2-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (619) 754 3708.

### *Basic Principles of Host Security*

Only run services that are necessary and in use on the servers. Every network-accessible service is potentially another way for the attacker to penetrate the system. Furthermore, these services should only be accessible by hosts that need to access them. On a public web server, that may be the entire Internet. On a private file server, that may only be selected workstations.

To keep the number of different services running on a single server to a minimum, it may be a good idea to separate servers by function. That means, if possible, the mail server should not also be the web server. If the mail server is a separate machine from the web server, that may mean a Sendmail exploit that compromises the mail server does not also result in the web server being compromised.

The software packages for those services should be kept up to date. Older versions of software may be vulnerable to security bugs or buffer overflows that have been repaired in newer releases. One easy way to do this is to take advantage of Red Hat Network. Every Red Hat Enterprise Linux host comes with a Red Hat Network entitlement. For a workstation, Automatic Updates initiated by rhnsd may help to keep the host up to date with low administrative overhead. For a critical server, it may be more desirable to use the **yum** (RHEL5) or **up2date** (RHEL4 and earlier) utility to manually update the server during a scheduled downtime, to avoid service disruptions due to unexpected changes in how the software behaves. A test lab can be used to test software updates before they are applied to production servers.

Limit user-level access to the servers. Once an attacker can log onto a system, there are many more ways that system can be attacked in an effort to gain root access. An unprivileged account may be used as a stepping stone to further control.

## SELinux

- Mandatory Access Control (MAC) -vs- Discretionary Access Control (DAC)
- A rule set called the *policy* determines how strict the control
- Processes are either restricted or unconfined
- The policy defines which resources a restricted process is allowed to access
- Any action that is not explicitly allowed is, by default, denied

2-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

In an effort to deal with ever increasing threats to their data systems, the US government assigned the National Security Agency (NSA) the task of developing a single set of rules that all other agencies would follow in handling confidential information. By evaluating previous breaches, the NSA determined that a major hurdle to securing data was internal users bypassing local security. In some cases, users would inadvertently open access to a system. A classic example would be a user executing the command `chmod 777 ~`. To the user, this may seem an easy way to allow co-workers to share files. They may not realize that this gives everyone in the world access to potentially confidential information.

In more extreme cases, users would intentionally disable security for more insidious reasons. Both of these situations are examples of a user having the discretion to control the access to their system. The NSA felt the solution was to have systems implement *Mandatory Access Control (MAC)* over the users. In MAC, a set of rules, known as the *policy*, identify what a process is allowed to do. Anything that is not explicitly permitted is, by default, denied. Ideally, different policies could be implemented depending upon how strict the security needs.

The NSA's first implementation of MAC was a system called Mach, which introduced the concept of *Type Enforcement (TE)*. Objects (files, directories, resources) were assigned a type value. Users and processes were also assigned a type value. The policy contains rules that allow a user or process to access objects. It was possible for a policy to be so *strict*, it was difficult to manage, so occasionally users and processes were allowed to be *unconfined*. This meant the policy did not apply to that user or process.

The NSA decided Mach made a better security framework than operating system. To get this framework deployed on production systems, the NSA needed access to the OS source code. They took advantage of the open source Linux kernel, and developed a set of patches to enable MAC. These patches became known as Security Enhanced Linux, or SELinux for short. They were later refined and incorporated into the kernel source by Red Hat.

## SELinux, continued

- All files and processes have a *security context*
- The context has several elements, depending on the security needs
  - user:role:type:sensitivity:category
  - user\_u:object\_r:tmp\_t:s0:c0
  - Not all systems will display s0:c0
- **ls -Z**
- **ps -Z**
  - Usually paired with other options, such as **-e**

2-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email [training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

There is an old UNIX saying that “everything is a file”. Traditional file access is controlled by user, group, and permission settings. To SELinux, everything is an object and access is controlled by security elements stored in the inode’s extended attribute fields. Collectively, the elements are called the security context. Presently, there are five supported elements, although all five may not be present on all systems:

user	Indicates the type of user that is logged into the system. If a user logs in as root, they will have the user value of <code>root</code> . Other users will have a value of <code>user_u</code> . If they escalate their privileges with <code>su</code> , they will still have the user value of <code>user_u</code> . Processes have a value of <code>system_u</code> .
role	Defines the purpose of the particular file, process, or user. Files have the role of <code>object_r</code> . Processes get the role of <code>system_r</code> . Users also have the role of <code>system_r</code> , because (to Linux) users are similar to processes.
type	Used by Type Enforcement to specify the nature of the data in a file or processes. Rules within the policy say what process types can access which file types.
sensitivity	A security classification sometimes used by government agencies.
category	Similar to group, but can block root’s access to confidential data.

To view the security context of a file, use the `ls` command’s `-Z` option:

```
[root@shadex4 ~]# ls -Z /root/anaconda-ks.cfg /var/log/messages
-rw-r--r--  root root  user_u:object_r:user_home_t  anaconda-ks.cfg
-rw-----  root root  system_u:object_r:var_log_t  /var/log/messages
```

Generally speaking, files inherit a directory’s security context:

```
[root@stationX ~]# ls -Zd /etc /etc/hosts
drwxr-xr-x  root root  system_u:object_r:etc_t   /etc
-rw-r--r--  root root  system_u:object_r:etc_t   /etc/hosts
```

Some files, however, get a unique security context, for added security:

```
[root@stationX ~]# ls -Z /etc/shadow /etc/aliases
```

```
-r-----  root root  system_u:object_r:shadow_t      /etc/shadow  
rw-r--r--  root root  system_u:object_r:etc_aliases_t  /etc/aliases
```

If a system were running under the most secure configuration, everything would be restricted by SELinux. This is not practical in most cases. For this reason, SELinux is being deployed in phases. In Red Hat Enterprise Linux 4, SELinux was protecting 13 processes. In the initial release of Red Hat Enterprise Linux 5, this count had increased to 88.

To determine if a process is protected, use the **ps** command's **-Z** option:

```
[root@stationX ~]# ps -ZC syslogd,bash  
LABEL          PID  TTY    TIME     CMD  
system_u:system_r:syslogd_t  1888 ?  00:00:00  syslogd  
user_u:system_r:unconfined_t 2583 pts/0  00:00:00  bash
```

Any process whose type is *unconfined\_t*, is not yet restricted by SELinux. To view the entire process stack, use either **ps -eZ** or **ps Zax**.

*Note:* A restricted process is sometimes called *protected*, though it is the data that is protected, not the process.

## SELinux: Targeted Policy

- The targeted policy is loaded at install time
- Most local processes are *unconfined*
- Principally uses the type element for *type enforcement*
- The security context can be changed with **chcon**
  - **chcon -t tmp\_t /etc/hosts**
- Safer to use **restorecon**
  - **restorecon /etc/hosts**

2-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The source code that allows SELinux to protect a process is integrated into the kernel, but the rules that define how SELinux enforces security is defined in the policy. The NSA's most secure policy is called the *strict* policy. By default, Red Hat uses the *targeted* policy. The targeted policy "targets" certain processes to be restricted, and then enforces their access to files and resources. Other policies also exist, such as the *Multi Level Security* (MLS). All policies could be thought of as a subset of the Strict policy.

The policy defines the elements that can be used, and whether users are allowed to manipulate the elements. Since the targeted policy uses Type Enforcement as its principal security mechanism, it pays the most attention to the type element of the security context. The policy allows the **chcon** command to change the security context.

```
[root@stationX ~]# ls -Z install.log
-rw-r--r--  root root  root:object_r:user_home_t  install.log
[root@stationX ~]# chcon -t etc_t install.log
[root@stationX ~]# ls -Z install.log
-rw-r--r--  root root  root:object_r:etc_t        install.log
```

The **chcon** command can only use types that are defined in the policy. Rather than memorizing all the possible types that can be used, **chcon --reference** can take the security context from one object, and apply it to another.

```
[root@stationX ~]# ls -Z anaconda-ks.cfg
-rw-----  root root  system_u:object_r:user_home_t  anaconda-ks.cfg
[root@stationX ~]# chcon --reference /etc/shadow anaconda-ks.cfg
[root@stationX ~]# ls -Z anaconda-ks.cfg
-rw-----  root root  system_u:object_r:shadow_t     anaconda-ks.cfg
```

A safer alternative is the **restorecon** command. With **restorecon**, the policy determines and applies the object's default context.

```
[root@stationX ~]# restorecon /root/*
[root@stationX ~]# ls -Z /root
```

```
-rw-----  root root  root:object_r:user_home_t    anaconda-ks.cfg
-rw-r--r--  root root  root:object_r:user_home_t  install.log
```

## SELinux: Management

- Modes: Enforcing, Permissive, Disabled
  - Changing enforcement is allowed in the Targeted policy
  - **getenforce**
  - **setenforce 0 | 1**
  - Disable from GRUB with **selinux=0**
  - **/etc/sysconfig/selinux**
- **system-config-securitylevel**
  - Change mode, Disabling requires reboot
- **system-config-selinux**
  - Booleans
- **setroubleshootd**
  - Advises on how to avoid errors, not ensure security!

2-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[trainings@redhat.com](mailto:trainings@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The policy for a system is defined in the `/etc/sysconfig/selinux` file as is the mode of operation. The policy can be Disabled, Enforcing, or Permissive. Disabled means the policy is ignored. Permissive is a mode for troubleshooting or development that logs policy violations, but does not prevent programs from running. Enforcing is the default mode.

The **getenforce** command can be used determine the system's current mode. The targeted policy allows the use of the **setenforce** command to toggle between the Enforcing (1) and Permissive (0) mode:

```
[root@stationX ~]# getenforce
Enforcing
[root@stationX ~]# setenforce 0
[root@stationX ~]# getenforce
Permissive
```

The only way to disable SELinux is to change `/etc/sysconfig/selinux`, and reboot, or use the `selinux=0` Grub kernel option.

The GUI tool, **system-config-selinux**, allows for changes of a few other SELinux options. The targeted policy allows certain features to be controlled through a set of *booleans*. These are predefined functions that can selectively turn off enforcement of certain daemons. It would be preferable to ensure objects have the correct security context rather than shutting off security features.

When an application attempts something that is not authorized by the policy, SELinux blocks access and an error is logged to `/var/log/audit/audit.log` in case `auditd` is running (this is the default). When `auditd` is not running, SELinux logs to `/var/log/messages`. The application is often unaware

of why it failed. This can make troubleshooting difficult. To help in the troubleshooting process, the **`setroubleshootd`** daemon will alert you of the error by placing a warning icon on the alert panel. Clicking on the icon will display a possible fix for the error. It is important to realize that the proposed solution may not be the best solution for the problem.

## SELinux Troubleshooting

- What is the error?
  - Check `/var/log/audit/audit.log` for AVC denials
- Is the process doing something it should not?
- Does the target have the right context?
- Does a boolean setting need adjustment?

2-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

SELinux policy violations are logged to `/var/log/audit/audit.log` unless auditd is not running, in which case they are logged to `/var/log/messages`. An error might read like the following:

```
Feb 25 01:38:23 station15 kernel: audit(1109313503.808:0): avc: denied  
{ read } for pid=4346 exe=/usr/sbin/httpd name=joe dev=hda2 ino=311297  
scontext=root:system_r:httpd_t tcontext=system_u:object_r:user_home_dir_t ↵  
tclass=dir
```

This translates as: PID 4346, a `/usr/sbin/httpd` process, with the context `root:system_r:httpd_t` was denied read access to a directory, named `joe`, which is inode 311297 on `/dev/hda2`, which has the context `system_u:object_r:user_home_dir_t`.

At this point several questions must be asked. Is the process being blocked for legitimate reasons -- is it doing something inappropriate? If not, then is the target's context wrong? If so, the correct context needs to be determined and set with chcon. If the policy is being too strict, perhaps a "boolean" setting can be adjusted with `system-config-securitylevel` or `setsebool`. In the worst case, perhaps SELinux can be disabled for just the affected service, or entirely.

Resources that can help troubleshoot SELinux problems include the Red Hat SELinux Guide on [www.redhat.com/docs](http://www.redhat.com/docs). Advanced SELinux policy writing techniques are covered in depth in the course "RHS429 - SELinux Policy Administration".

## Host-based Access Control

- Kernel packet filters
  - Netfilter: `Iptables` and related tools
- TCP wrappers
  - Supported by most network services
- `xinetd`-based access controls
  - Only usable by `xinetd`-based services

2-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Access Control By Hostname or IP address*

For the remainder of this unit, a look at three generic mechanisms through which access control and logging may be applied on the basis of hostname or IP address will be taken. First, there will be a brief glance at the kernel's packet filtering mechanism, Netfilter. Second, there are the advanced uses of the TCP wrappers facility, which is supported by most network services. Finally, there will be a discussion on access controls and advanced options available to services managed by `xinetd`.

## Netfilter/iptables

- Netfilter is the first line of defense
  - **system-config-securitylevel**
  - **Iptables** can be used for fine-tuning
- Local firewall is enabled by default
  - Accept ICMP, IPSec, CUPS, multicast DNS, responses to requests initiated by the host and traffic arriving on the loopback interface.
  - **RH-Firewall-1-INPUT** custom chain

2-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828-2964 or +1 (919) 754-3700.

### Netfilter: iptables and related tools

The system's first line of defense is the kernel packet filter, Netfilter. By default, the installer enables a host-based firewall using **system-config-securitylevel**. These simple firewall rules are appropriate for a workstation. By default, outbound network traffic is not restricted, and connection tracking is used to accept only inbound:

- Traffic arriving on the loopback interface (lo)
- Traffic related to connections initiated by the localhost
- ICMP packets
- Traffic using IPSec protocols (esp and ah)
- Traffic using multicast DNS (port 5353, used with Zeroconf/Avahi/Bonjour)
- Traffic using the CUPS print daemon (port 631)

The installer also permits other optional firewall rules to be set to allow network traffic to particular well-known ports. Use **system-config-securitylevel** to enable or disable this basic firewall or adjust the optional rules. You may also use the **iptables** command to adjust the firewall rules. The custom chain **RH-Firewall-1-INPUT** contains these rules by default, and all traffic processed by the INPUT chain is sent to that custom chain when the firewall is enabled.

Basic iptables techniques are covered in the core RHCE curriculum and will not be repeated here. Advanced packet filtering techniques are covered in depth in RHS342 - Developing Red Hat Firewall Solutions. The site <http://www.netfilter.org/> also has additional information and tutorials on how to work effectively with Netfilter and iptables.

## Review of TCP Wrappers

- daemon list : client list : [option]
- Options are listed in hosts\_options(5)
  - There may be zero, one, or more options
  - Option %<letter> substitutions work as specified in hosts\_access(5)
  - Options behave the same in either hosts.allow or hosts.deny files

2-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (656) 826 2994 or +1 (919) 754 3700.

The concept of matching connections is central to the operation of TCP Wrappers. A match occurs when an incoming connection matches a rule in which the service is in the daemon list and the connecting host is in the client list. If a match occurs the options listed will be executed.

The possible options available are described in the hosts\_options(5) man page. The extensions allow for logging, access control, command execution, modification of network behavior, username lookup and other options. Some of these options will be explored in the following pages.

If there is more than one option, each option should be separated by colons. Colons that are part of an option must be preceded by a backslash. The options are executed in the order specified.

Since Red Hat Enterprise Linux uses a version of TCP Wrappers compiled with -DPROCESS\_OPTIONS, an option may not be a bare shell command (as shown in hosts\_access(5) in the “Shell Commands” section). Instead, shell commands must be executed using spawn or twist.

## Access Control

- daemon list : client list : **ALLOW/DENY**
- Changes how rule will be interpreted no matter which file it is in
- Allows consolidation of rules into one file
  - May simplify rule management
- Must be the last option in the rule

2-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 866 826 2994 or +1 (519) 784 3700.

The use of ALLOW and DENY as options enable you to specify whether or not a match allows or denies a connection request regardless of which file the match is found in. This makes it possible to consolidate rules in the hosts.allow and hosts.deny files into a single file.

*Example:* (in /etc/hosts.deny)

```
in.telnetd : .example.com : ALLOW
ALL:ALL
```

This lets users from hosts in the example.com domain telnet to this machine even though the match appears in /etc/hosts.deny.

## Logging

- daemon list:client list:*severity* [*fac.*]*pri*
- Logs the connection using the syslog facility and priority specified
  - Facility is optional
- Default is authpriv.info

2-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The severity option allows you to control the facility and priority of messages sent to syslog. The default is authpriv.info. Specification of a facility is optional. This can be used to emphasize the importance of certain connections.

*Examples:* (in either hosts.allow or hosts.deny)

`sshd : ALL : severity local7.warn`

would cause any sshd connections to be logged using the local7 facility with priority warn.

`sshd : ALL : severity notice`

would cause any sshd connections to be logged using the authpriv facility with priority notice.

## Running Other Commands

- **daemon list : client list : *spawn* command**
  - Executes command in a child process
  - Default I/O is connected to */dev/null*
- **daemon list : client list : *twist* command**
  - Service is replaced by command
  - Default I/O is connected to client
- Both support %<letter> substitution

2-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2804 or +1 (919) 754 3700.

The spawn and twist options both cause a command to be run when a connection matches the rule. With spawn, the connection is allowed or denied and the shell command specified is run as a background child process on the server. Standard input, output, and error are sent to */dev/null* unless redirected. With twist, if the rule matches the shell command will be run instead of the wrapped server. Standard input, output, and error for the command will be connected to the connecting client.

The spawn command is useful for running some alert program on the server when a rule matches. The twist option is useful to redirect connections from particular hosts to a differently configured server process, or to output some error message to the connecting client.

*Examples:*

`ALL : 192.168.0. : spawn /bin/echo `date` %c %d >> /var/log/tcpwrap.log`

will cause any attempts for connections from the 192.168.0/24 network to be logged. Where the connection is allowed or not depends on whether or not the line appears in */etc/hosts.allow* or */etc/hosts.deny*.

`vsftpd : 192.168.0. : twist /bin/echo "421 Connection prohibited."`

will cause clients connecting to vsftpd from the 192.168.0/24 network to be sent a 421 error with the text "Connection prohibited."

## banners

- **daemon list:client list:banners directory**
  - The file "directory/daemon name" is sent to the client before connected to service
  - %<letter> substitution works in the banner
  - Only works with TCP-based services

2-15

For use only by a student enrolled in a Red Hat Training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

The banners option will cause a file to be sent to a client that connects to the service. banners requires a single argument, the name of the directory that contains the banner files. Each service using banners will look in that directory for a file with the same name as that of the service.

*Example:*

Suppose you have a directory named /var/banners, and the following line in hosts.allow:

```
in.telnetd,vsftpd : ALL : banners /var/banners
```

This would cause the file /var/banners/vsftpd to be sent to anyone who connects to the ftp server on the machine, and /var/banners/in.telnetd to be sent to anyone who telnets to the machine. The vsftpd file might contain the lines:

```
220-Hello to %c from the server on ftp.example.com!
220-All connections to this server are logged.
220-
```

This message would appear on the connecting ftp client before authentication. The %c would be expanded with the client's host name as specified in hosts\_access(5).

## xinetd Configuration Review

- Services configured in /etc/xinetd.d
- Each configuration file is named for the service it configures

2-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

Individual services controlled by xinetd each have a configuration file in /etc/xinetd.d. The names of the configuration files in the /etc/xinetd.d directory are not important as long as they do not contain a “.” or a “~”. By convention, the files are normally named for the services they configure.

Here is an example from /etc/xinetd.d/telnet:

```
service telnet
{
    disable     = no
    flags       = REUSE
    socket_type = stream
    wait        = no
    user        = root
    server      = /usr/sbin/in.telnetd
    log_on_failure += USERID
}
```

The service configuration begins with a service statement followed by the service name from /etc/services. The actual configuration arguments are contained in a block defined by braces (“{}”) and each line consists of an attribute with an optional operator (“=”, “+=”, “-=”) and values. For example, the actual server started up by xinetd is /usr/sbin/in.telnetd, specified by the server attribute.

For more information on basic xinetd configuration, see **xinetd.conf** (5).

## /etc/xinetd.d Issues

- Files in /etc/xinetd.d are read in alphabetical order
- First *enabled* service block which applies to the interface is used
  - All others are ignored
- Can accidentally get wrong program if several programs using the port exist
  - krb5-telnet or telnet

2-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The /etc/xinetd.conf file uses an includedir directive to add all the files in /etc/xinetd.d to the xinetd configuration. These files are added in alphabetical order. There can be more than one service block that might apply to a particular network port in the configuration. The first service block that matches for an interface and has disable = no set will be used, and all service blocks that might apply after that are ignored.

This can be useful. In conjunction with the bind directive, a multi-homed host can have different access restrictions or server programs answering requests from clients contacting different local IP addresses, allowing virtual hosting and other tricks.

On the other hand, this can be confusing. If several services exist that use the same service port, and two or more of them are turned on at the same time by mistake, then the one with the /etc/xinetd.d file that is read first will be the one that is used. For example, there are two telnet servers that can be managed through xinetd. These are controlled by the /etc/xinetd.d files krb5-telnet (for /usr/kerberos/sbin/telnetd) , and telnet (for /usr/sbin/in.telnetd) . If both services were enabled, the krb5-telnet file would be used and the telnet file ignored.

## Location Limits

- **Syntax**

- Allow with `only_from = host_pattern`
- Deny with `no_access = host_pattern`
- The most exact specification is authoritative

- **Example**

- `only_from = 192.168.0.0/24`
- `no_access = 192.168.0.1`

2-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email [cstraining@redhat.com](mailto:cstraining@redhat.com) or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 764 3700.

If neither `only_from` nor `no_access` are specified for a service, the default behavior is to permit the connection. If both are specified for a service, the “most exact” specification is used. In some cases, what the “most exact” match is may be ambiguous, so it may be a good idea to use only one or the other of these specifications when possible.

A `host_pattern` may use one or more:

- IP addresses (`192.168.0.1`). Rightmost zeros are treated as wild cards. `0.0.0.0` matches all hosts on the Internet.
- IP addresses with netmask (`192.168.0.0/255.255.255.0` or `192.168.0.0/24`).
- host names (`server1.example.com`). This may cause a DNS lookup on every connection.
- domain names (`.example.com`). This may cause a DNS lookup on every connection.
- network names from `/etc/networks` if that file exists (`@mynetwork`).

*Example:*

```
only_from      =      192.168.0.0/24
no_access      =      192.168.0.{1,3,5,9}
```

will only allow access to hosts on the `192.168.0.0/24` network, with the exception of four machines on that network.

## Time Limits

- **access\_times** - specifies the time of day the service will be available
  - `access_times = <time range> ...`
  - Hours range from 0-23, minutes from 0-59

2-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 734 3700.

Time of day limits can be imposed through the `access_times` attribute. The `<time_range>` specifies the start and end of the time window during which the service may be accessed, using 24-hour time. There may be multiple time ranges on a single line separated by spaces.

*Example:*

```
access_times = 08:00-17:00
```

## Interface Limits

- bind - associates the service with a particular interface
  - bind = <ip address>
  - interface is a synonym for bind
- Can use the same port with different services on different IP addresses
- Can limit access to a service to clients connecting to a certain IP address

2-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2984 or +1 (919) 754 3700.

Should there happen to be multiple IP addresses assigned to a host, the bind attribute allows an association to a particular service's configuration with a particular IP address. This allows the service one configuration when it is accessed through one address and a different configuration when it is accessed through another address.

*Example:*

```
service telnet
{
    disable      = no
    flags        = REUSE
    socket_type = stream
    wait         = no
    user         = root
    server       = /usr/sbin/in.telnetd
    bind         = 192.168.0.7
}

service telnet
{
    disable      = no
    flags        = REUSE
    socket_type = stream
    wait         = no
    user         = root
    server       = /usr/sbin/in.telnetd
    access_times = 8:00-17:00
    bind         = 192.168.1.7
}
```

will cause connection requests to 192.168.1.7 to be restricted to business hours, but requests to 192.168.0.7 will be allowed all the time. Notice that both service statements in the example above are in the *same* /etc/xinetd.d/telnet file.

## Usage Limits

- **cps - limit rate of incoming connections**
  - `cps = <connectionspersec> <waitperiod>`
- **per\_source - set maximum number of connections from a particular source IP**
  - `per_source = <#connections/UNLIMITED>`
  - **instances** can be used instead to specify a maximum number of connections from all hosts

2-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 784 3700.

The number of connections allowed per second can be limited with the `cps` attribute. When the rate of incoming connections exceeds `<connectionspersec>`, `xinetd` will refuse to accept any further connections for `<waitperiod>` seconds.

*Example:*

```
cps = 25 30
```

will cause the service to stop accepting connections for 30 seconds if the rate of incoming connections exceeds 25 per second.

The maximum number of connections established from a particular source IP address can be limited with the `per_source` attribute. The maximum number of connections from all hosts can be limited with the `instances` attribute. Both attributes expect as an argument either an integer number of connections or the keyword `UNLIMITED`.

*Example:*

```
per_source = 15
```

limits the number of connections from any particular IP address to fifteen. This can not be used to override the `instances` attribute.

Other attributes exist that are useful to control service usage. For example, the `max_load` attribute sets the one minute load average at which the service will stop accepting connections.

## **flags and deny\_time**

- **flags modify the behavior of xinetd**

- INTERCEPT - check each packet for authenticity rather than just first one
- SENSOR - establishes a "trap" service
  - Used in association with deny\_time attribute
  - deny\_time = <FOREVER|NEVER|#minutes>

2-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 734 3700.

The flags attribute modifies the behavior of xinetd. The attribute may be followed by a number of flags. Some useful ones are discussed here.

*Example:*

```
flags = SENSOR INTERCEPT
```

sets the SENSOR and INTERCEPT flags (discussed below) for the service.

INTERCEPT causes each packet sent to the service to be tested for acceptability. If a service has `wait=yes` and `socket_type=dgram` set, only the first packet of the connection will be tested if the INTERCEPT flag is not set. If a service has `wait=yes` and `socket_type=stream` set, access control is not honored if the INTERCEPT flag is not set. There are some other performance and behavioral considerations discussed in `xinetd.conf(5)`.

SENSOR is used in association with the `deny_time` attribute to provide a "trap" service. When a connection is established to a service that has the SENSOR flag set, the IP address for the connecting machine is placed in a global `no_access` list for the period of time specified by `deny_time`. The global `no_access` applies to all xinetd managed services. The `deny_time` can be set to FOREVER (until xinetd is restarted), NEVER (log only), or some number of minutes. This can be used to block hosts scanning a machine for open ports. However, an attacker that knows that a machine is running a SENSOR can use it in a denial of service attack to block other hosts from connecting to that server, by connecting to the sensor port with forged source IP addresses.

## Interaction Between Filters

- First restriction takes effect
  - Checks against *iptables* first
  - For *xinetd* services: TCP wrappers that affect *xinetd*, and *xinetd* configuration
  - TCP wrappers that affect the service itself
- The service may have its own internal restrictions (NFS, Apache, etc.)

2-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

There is a well defined order in which limits are applied to incoming connections. Initially, the rules set in the kernel by *iptables* are applied. Secondly, if the service is linked against *libwrap*, the TCP wrappers rules are applied. For *xinetd*-based services, TCP wrappers rules for both *xinetd* and for the service itself are examined. In addition, any configuration settings in the *xinetd* configuration files are examined. If the incoming message passes all these checks, then it can reach the service. However, the requests sent to the service may still be subject to service-managed access controls.

*Example:*

```
$ iptables -A INPUT -s ! 192.168.0.0/24 -p tcp --dport 80 -j DROP
```

would block any packets destined for port 80 using the TCP protocol coming from anywhere except the 192.168.0.0/24 network, *prior* to any consideration by TCP wrappers or *xinetd*.

## End of Unit 2

- Questions and Answers
- Summary
  - Simple steps to improve host security
  - TCP wrapper options
  - Advanced xinetd configuration

2-24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

## Lab 2

# Basic Service Security

---

**Goal:** In this lab, you will explore some host-based security techniques using **iptables**, TCP wrappers, and **xinetd**. You should work with a partner to implement and test various access restrictions on your workstations.

A test machine on a "non-local" network, cracker.org, has been made available by the instructor. You may ssh into this machine from your workstation for testing purposes. The root password on that machine is also redhat.

**Estimated Duration:** 60 minutes

**System Setup:** Create three user accounts: `peter`, `paul`, and `mary`. Set a password for each account.

Unless otherwise directed, all attempts to connect to services should be made using one of these users. All other actions should be performed as `root`. The default `root` password is `redhat`.

In this class, you will occasionally need to install additional software packages which are part of the standard distribution or which are provided by the instructor. Several yum repositories containing these packages have been set up on `server1.example.com` to simplify the software installation process.

To set up access to these repositories, log in to your station as `root` and download the yum configuration file from the classroom FTP server:

**lftpget ftp://server1.example.com/pub/gls/server1.repo**

Then, copy `server1.repo` to the `/etc/yum.repos.d/` directory. Yum should now be ready to use. Since your machine is not registered to use Red Hat Network, when yum is run you will see messages indicating that RHN will not be used as one of your sources of software packages.

A review of some basic yum commands:

<code>yum list</code>	List all available packages
<code>yum install package</code>	Download and install package
<code>yum info package</code>	Get information about package

See the `yum(8)` man page for more information.

## Sequence 1: Controlling the default host-based firewall

### Scenario:

By default, the installer sets up some basic connection-based packet filtering rules on inbound network traffic. The intention of this is to block access to network services which a system administrator may turn on inadvertently. In this sequence, we will ensure that the firewall is enabled. Over the course of subsequent labs, we will open access to your local services only as needed.

As a quick review, remember that you can delete rules with **iptables -D CHAIN rule-number** if you make a mistake. The **iptables -L --line-numbers** command can help you determine the correct line number of your broken rule. Do not forget to use **service iptables save** to save your work, or you will lose your changes on the next reboot!

### Instructions:

1. Log in as **root**, and run **system-config-securitylevel**.
2. In that interface, set Firewall to Enabled. Mark **SSH** as a trusted service.
3. Run the **iptables -L -v -n** command and examine your firewall rules. You should see three standard chains and a custom chain, **RH-Firewall-1-INPUT**, on the **filter** table. Rules are set in the INPUT and FORWARD chains directing all traffic to the custom chain. The custom chain permits any traffic from the **lo** network interface (localhost), any ICMP traffic, traffic using the AH/ESP IPsec protocols, and traffic part of or related to communications initiated by your host. It should also accept inbound traffic to port 22/tcp, for **sshd**. Any other traffic will be rejected by the firewall with an ICMP Host Prohibited message.
4. Set up a new custom chain for our rules in this class, **RHS333**.
5. Now make sure that all incoming traffic is processed by the **RHS333** chain before it is processed by **RH-Firewall-1-INPUT**.

We want the **RHS333** chain to match first on the **INPUT** chain. If there are no matches on the **RHS333** chain, then the firewall will return to the **INPUT** chain, which will cause the **RH-Firewall-1-INPUT** rules to be processed.
6. Save your work.

## Sequence 2: TCP wrappers

**Scenario:** Configure TCP wrappers to control connections to your insecure **telnet** server. Connections will only be accepted from local hosts. A banner message will be sent to hosts not on the local network.

**Instructions:**

1. Make sure the **xinetd** and **telnet-server** RPMs are installed. **yum install telnet-server** should get both packages. Enable both services on your workstation.
2. In the firewall, open the telnet server port (23/tcp) so that anyone on the network can connect to it.

Verify the setting is correct with the **iptables -L**.

3. Make the directory **/var/mesg/** and create an executable file called **/var/mesg/deny** that has the following contents:

```
#!/bin/bash
echo "External connection refused at $(/bin/date)"
```

This will be used to print a banner message out to telnet clients with the current date. In practice you probably wouldn't do this (it's an information leak from your server), but it will suffice to show that the **twist** rule that you are about to set is actually running **/var/mesg/deny**.

4. Modify the **/etc/hosts.deny** file so the **/var/mesg/deny** script you just created is run instead of **in.telnetd** for all telnet connections from outside the local network.
5. Modify the **/etc/hosts.allow** file to allow telnet connections from the example.com network (192.168.0/24) and the localhost network (127/8).
6. Working with another student as your partner, have them attempt to telnet to your machine from their machine. This should work. Then ssh into the cracker.org workstation and try to telnet to your machine from there. That should display the output of the **twist** script.
7. Remove the changes you made to your **/etc/hosts.allow** and **/etc/hosts.deny** files.

## Sequence 3: xinetd SENSOR traps

### Scenario:

**xinetd** can be used to set a "trap" on particular service ports. If an attacker tries to connect to a booby-trapped port, you can log the information or use it to add the scanning host to **xinetd**'s global `no_access` rule for some length of time. In this sequence, you will set up a trap that will disable connections from machines that attempt to connect to the **rlogind** port on your machine.

### Instructions:

1. Verify you are able to **telnet** to your neighbor's machine (stationY.example.com) and vice-versa. (This shows that **xinetd** is currently allowing connections.)
2. Open the **rlogin** port (513/tcp) on your workstation.
3. Create or edit your `/etc/xinetd.d/rlogin` file so that it has the following contents (activate the **SENSOR** with a denial time of two minutes):

```
service login
{
    flags = SENSOR
    deny_time = 2
    socket_type = stream
    wait = no
    user = root
    server = /bin/false
}
```
4. Reload the **xinetd** service.
5. Ask your neighbor to **telnet** to your machine . Once your neighbor's ability to **telnet** to your machine is verified, you should ask your neighbor to attempt to **rlogin** to your machine. This will trigger the trap.
6. Verify that the **rlogin** attempt failed and that for the next two minutes no **xinetd** based services are available to your neighbor. Look in `/var/log/messages` on your machine for any relevant log messages from **xinetd**. In practice, you might want to set `deny_time = NEVER` so you log the probe attempts, but don't deny further connections. Why? How could an attacker use a sensor trap against you?
7. Cleanup: Remove the firewall rules that permit access to your **telnet** and **rlogin** ports.  
Also, be sure to remove any changes you have made to your `/etc/hosts.allow` and `/etc/hosts.deny` files. This will avoid unexpected problems in future labs.

## **Sequence 1 Solutions**

4. Set up a new custom chain for our rules in this class, RHS333.

```
# iptables -N RHS333
```

5. Now make sure that all incoming traffic is processed by the RHS333 chain before it is processed by RH-Firewall-1-INPUT.

We want the RHS333 chain to match first on the INPUT chain. If there are no matches on the RHS333 chain, then the firewall will return to the INPUT chain, which will cause the RH-Firewall-1-INPUT rules to be processed.

```
# iptables -I INPUT 1 -j RHS333
```

6. Save your work.

```
# service iptables save
```

## Sequence 2 Solutions

2. In the firewall, open the telnet server port (23/tcp) so that anyone on the network can connect to it.

```
# iptables -A RHS333 -p tcp --dport 23 -j ACCEPT
```

4. Modify the /etc/hosts.deny file so the /var/mesg/deny script you just created is run instead of in.telnetd for all telnet connections from outside the local network.

```
# echo "in.telnetd:ALL:twist      /var/mesg/deny" >> /etc/hosts.deny
```

5. Modify the /etc/hosts.allow file to allow telnet connections from the example.com network (192.168.0/24) and the localhost network (127/8).

```
# echo "in.telnetd:192.168.0. 127." >> /etc/hosts.allow
```

## Sequence 3 Solutions

2. Open the rlogin port (513/tcp) on your workstation.

```
# iptables -A RHS333 -p tcp --dport 513 -j ACCEPT
```

4. Reload the **xinetd** service.

```
# service xinetd reload
```

6. In practice, you might want to set `deny_time = NEVER` so you log the probe attempts, but don't deny further connections. Why? How could an attacker use a sensor trap against you?

- Denial of service attacks

7. Cleanup: Remove the firewall rules that permit access to your **telnet** and **rlogin** ports, and disable the telnet, xinetd, and rlogin services.

Also, be sure to remove any changes you have made to your `/etc/hosts.allow` and `/etc/hosts.deny` files. This will avoid unexpected problems in future labs.

```
# iptables -D RHS333 -p tcp --dport 23 -j ACCEPT
# iptables -D RHS333 -p tcp --dport 513 -j ACCEPT
# chkconfig telnet off; chkconfig rlogin off
# service xinetd stop; chkconfig xinetd off
```

## **Unit 3**

# **Cryptography**

**3-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ct-training@redhat.com](mailto:ct-training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 784 3700.

## Objectives

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

- Network communication security
- Introduction into cryptography
- OpenSSL

3-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Network Security

- Communication channels are often vulnerable to a “man in the middle” attack
  - Eavesdropping
  - Data injection
  - Session hijacking
- Authentication and encryption of messages can protect data integrity and confidentiality

3-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

### *Authentication and Encryption*

Network communication channels are usually insecure. This is particularly true for wireless networking, but is also true for standard network communication. In general, it is possible for an attacker to intercept messages sent over the network. The most common form of attack is simple eavesdropping. Many traditional network protocols, such as SMTP or Telnet, use cleartext messages to communicate. These messages may expose authentication credentials; such as usernames and passwords, or sensitive data.

Another form of attack is data injection. The attacker attempts to insert false messages into an established, legitimate communication taking place between two hosts. Data injection attacks may be used to send bogus log messages to a host, or to cause a network service to misbehave in some way.

Session hijacking is when the attacker tries to break or take over the existing connection. This may be used as a simple denial of service attack, but usually the attacker tries to take over communication in both directions and allows the legitimate users to continue to communicate. This is known as a *man in the middle attack*. This sort of attack may be used to eavesdrop on the communication, or to subtly inject altered data so that both machines still think the communication is secure while allowing the attacker to suppress critical messages.

Cryptographic techniques can provide authentication and encryption of the communication channel, protecting data integrity and providing confidentiality.

## Switches and Security

- A network switch does not provide security
  - Provides improved bandwidth management
- Attacks on network switches include
  - MAC Flooding
  - MAC Duplication
  - ARP Redirection
- Port security, static ARP entries, and **arptables** may provide some defense

3-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *Network switches and insecurity*

Network hubs broadcast all network traffic to every machine connected to the hub. Traffic meant for a particular machine is labeled with its link-layer MAC (Media Access Control) address. Network switches reduce bandwidth consumption by keeping track of which MAC addresses are associated with which of its ports and sending traffic intended for a particular machine only to a particular port. It is a common misconception that network switches improve network security. Unfortunately, this is not true. A switch can be vulnerable to sniffing or session attacks. Some common attacks include:

- **MAC flooding.** Switches may be plugged into switches or hubs, so a switch port may have many MAC addresses assigned to it. The attacker attempts to overwhelm the switch's memory by flooding it with a huge number of MAC addresses for a particular port. Some switches "fail open" when they run out of memory and begin to act like a hub.
- **MAC duplication.** Some network cards have multiple connections to the same switch to increase bandwidth through trunking. In some trunking modes, this may cause the same MAC address to appear on multiple switch ports. If the attacker forges his MAC address to match a target machine on the switch, the switch may attempt to send its traffic to both hosts!
- **ARP redirection.** Sometimes, an IP address changes port and network card. For instance, this is used by high-availability clusters to fail over a network service from one server to another. An attacker broadcasts forged ARP redirects, telling the network that the MAC address associated with a particular IP address has changed. This causes the target machine to redirect traffic meant for that IP address to a host on the network under the attacker's control. The attacker may then forward that traffic on to the legitimate destination, allowing communication between the legitimate hosts to continue while being intercepted by the attacker.

Some switches can defend against MAC flooding and MAC duplication through a feature called "port security", in which a particular switch port is associated with one particular MAC address. Another approach is to set static ARP entries between critical systems (e.g., arp -s 192.168.0.254 00:01:02:ab:cd:ef), but this is hard to manage. Another useful utility is arptables. This tool can set up

more flexible ARP-based packet filters in the kernel, in the same way that iptables works to set up IP address-based packet filters.

# Cryptography

- Key building block of secure protocols
- Authentication and confidentiality of data
  - Encryption does not automatically imply authentication, and vice versa
  - Generally does not hide the fact that communication is taking place, or with whom
- Cryptography does not solve every problem
  - May be easier to break host security than break an encrypted communication

3-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2004 or +1 (919) 784 3700.

## *Introduction to Cryptography*

Cryptography is a key building block for secure network communication protocols. Cryptographic techniques can provide authentication and confidentiality to network connections. One major misconception is that encryption automatically implies secure authentication. This is not true, but a well-designed protocol will provide both.

Another issue many people neglect is that cryptography generally does not provide secret communication. The data transmitted is confidential; that is, an observer does not know what was said. However, the observer can tell that the communication took place, who was talking to whom, and how much communication was occurring. This sort of *traffic analysis* in itself can sometimes provide useful information to an attacker.

Cryptography is not a cure-all. In many cases, an attacker may find it easier to compromise one of the communicating hosts than to break the encryption in use. However, encryption is important. Interception of network messages can be a much more subtle and hard to detect attack than a successful host compromise.

# Cryptography

- Kerckhoffs' Principle
  - Security must depend on secrecy of the key, not on secrecy of the encryption method
  - Harder to replace algorithms than secret keys
- Secure authentication and integrity can be more important than secure encryption
  - Cryptography can help with both
  - Authentication may not automatically handle blocked or repeated messages

3-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## *Basic principles*

One of the key axioms of cryptography is Kerckhoffs' Principle. This states that the security of an encryption method must depend on the secrecy of the key, and not on the secrecy of the encryption method. One reason for this is that secret encryption protocols have a very poor track record. They are often found to have serious design flaws when they are analyzed. Another reason is that a truly secure encryption algorithm should work even if it is being used against one of its designers. Finally, a practical reason is that it is much harder to replace broken algorithms than it is to deploy new secret keys.

Another somewhat surprising principle is that often authentication and data integrity are more important than secure encryption. If a message cannot be authenticated, then an attacker can change the data being communicated. This can lead to more damaging attacks than simple interception of messages. In addition, authentication in itself does not do anything to protect against the attacker replaying captured messages or to notice if messages have been lost. Usually extra steps must be taken by the communication protocol to detect these conditions.

## OpenSSL

- **libcrypto:** Generic encryption support
  - Provides cryptographic functions
- **libssl:** Support for TLS/SSL protocols
  - Connection-based security for authentication, confidential data transfer, session integrity
- **openssl:** Multi-purpose encryption utility
  - Generate encryption keys, digital certificates
  - Encrypt/decrypt data manually

3-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2964 or +1 (919) 754 3700.

The OpenSSL package includes:

- *libcrypto*, a library of generic encryption-related functions that can be used by other protocols and programs. Some packages that use libcrypto include BIND, OpenSSH, and Cyrus SASL.
- *libssl*, a library providing functions for an open source implementation of the Transport Layer Security (TLSv1) protocol standardized in RFC 2246, as well as the older Secure Socket Layer (SSL) protocols developed by Netscape. TLS cannot be used with UDP services, as it requires a reliable connection-based transport layer protocol to function. Programs that may use libssl include Sendmail, OpenLDAP, Apache, and IMAP, among others.
- *openssl* is a multi-purpose command-line utility that can be used to generate keys or encrypt data using a number of different cipher algorithms. It can also create and manage X.509 digital certificates and calculate message digests.

The encryption capabilities provided by OpenSSL are used by a large number of secure services. Further information on OpenSSL is available at the project website, [www.openssl.org](http://www.openssl.org).

Other libraries and toolsets which support TLS/SSL and encryption are also included in Red Hat Enterprise Linux. These include the Mozilla Project's Network Security Services software (in the **nss** package) and GnuTLS (in the **gnutls** package)

## Symmetric Encryption

- Secret key used to both encrypt and decrypt
- Block ciphers encrypt fixed-size blocks of plaintext as fixed-size blocks of ciphertext
  - Block cipher modes are used to encrypt data larger than a single block (ECB, CBC, etc.)
- Algorithms: DES, 3DES, Blowfish, AES, etc.
- Utilities: **gpg**, **openssl enc**

3-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

### Secret Key Encryption

Symmetric, or secret key, encryption uses a shared secret key to encrypt from plaintext to ciphertext and to decrypt from ciphertext back to plaintext. One problem with symmetric encryption is how to get two hosts that may never have communicated before to agree on a shared secret key without exposing that key as cleartext to an eavesdropper.

The most common kind of symmetric encryption algorithm is the *block cipher*. A block cipher encrypts a fixed-size block of plaintext as a fixed-size block of ciphertext. DES is an obsolete block cipher with a small 56-bit key and 64-bit block size that makes it vulnerable to attack by modern computers. AES is a block cipher standardized by the US government and intended to replace DES and Triple DES. A modern block cipher like AES typically has a block size of 128 bits. Other block ciphers include Blowfish, Twofish, RC6, Serpent, IDEA, and CAST5.

Normally, we do not have exactly one block of data to communicate. A *block cipher mode* determines how the block cipher encrypts data larger than one block in size. Generally, the plaintext is padded to an exact multiple of the block size first. The naive block cipher mode is ECB, or electronic codebook, mode. In this mode each block is encrypted as a separate message using the same key. However, this means that if there are two identical plaintext blocks, there will be two identical ciphertext blocks, simplifying cryptanalysis. The most widely used block cipher mode is CBC, cipher block chaining. In this mode, each plaintext block is XORed with the preceding ciphertext block before encryption. The first block is XORed with an *initialization vector* generated from a *nonce*, a random number that should only be used once with a particular encryption key.

The **openssl enc** command supports a large number of encryption algorithms and modes. See **enc(1)** for details. As an example, to encrypt a file named **plaintext** with a password, using triple DES in CBC mode, stored base64 encoded:

```
$ openssl enc -des3 -salt -a -in plaintext -out ciphertext.des3
```

To decrypt the resulting **ciphertext.des3** file:

```
$ openssl enc -d -des3 -salt -a -in ciphertext.des3 -out plaintext
```

The `gpg -c plaintext` command can also be used for symmetric encryption. Use `--cipher-algo` to select the symmetric cipher; a list is available from `gpg --version`. To decrypt, use `gpg -d ciphertext.gpg`.

## Cryptographic Hashes

- Converts an input string of any length to an output string of fixed length
  - One-way: not feasible to get plaintext from hash
  - Collision-free: not feasible to find two strings that hash to the same output
- Algorithms: CRC-32, MD5, SHA-1, etc.
  - CRC-32 is *not* cryptographically secure!
- Utilities: **cksum**, **md5sum**, **openssl dgst**

3-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *Message Digests*

A *cryptographic hash*, or message digest, is commonly used to store UNIX passwords and to provide integrity checks for network communications. A cryptographic hash converts an input string of any length to an output string of fixed length. A cryptographically secure hash must be one-way and collision-free. For a one-way hash, it is not feasible for a computer to recover the original message from the hash. A hash is weakly collision-free if, given a hash, it is not feasible to find another string that hashes to the same value. A hash is strongly collision-free if it is not feasible to find any two strings that hash to the same value.

The CRC-32 algorithm (and the cksum utility) is *not* cryptographically secure. It should not be used as a cryptographic hash. Some older protocols do, for backward-compatibility purposes. The most widely used cryptographic hash today is probably MD5, but most new applications are using SHA-1 which is slower but considered to be slightly more secure.

The md5sum utility can be used to take the MD5 hash of a file. It can be useful to detect whether a file has changed or not. A more flexible utility is openssl dgst, which can compute a message digest using a number of algorithms, including MD5 and SHA-1. See dgst(1) for details. For example, to take a SHA-1 digest of /etc/grub.conf :

```
$ openssl dgst -sha1 /etc/grub.conf
```

SHA-1 was broken in January 2005. It is not safe as a hash function for digital signatures, although it does not affect applications such as HMAC where collisions are not important.

## Message Authentication Codes

- Secret key maps plaintext to random output
- Used for preventing message tampering
  - Attacker needs secret key to forge MAC value
- CBC-MAC: use block cipher to construct
  - Encrypt the message in CBC mode and throw away all but the last block of output
- HMAC: use keyed cryptographic hash

3-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ttraining@redhat.com](mailto:ttraining@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

### *Hashes and MACs*

A message authentication code, or MAC, is used to maintain the integrity of a network communication. The basic principle is that the MAC function uses a shared secret key to map plaintext to a fixed-length random output. When the server sends a message to the client, it also sends the value computed by the MAC function. Since an attacker does not know the secret key, the client can verify that this message came from the server and not an attacker.

One way to generate a MAC is to encrypt the message with a block cipher in CBC mode, and then throw away all but the last block of ciphertext. There are some important implementation details that must be followed to avoid some simple attacks. One very dangerous mistake is to use the same key for both CBC mode encryption and CBC-MAC authentication.

A somewhat simpler approach is to use a cryptographic hash function in conjunction with a secret key as a MAC. The basic HMAC technique is detailed in RFC 2104. HMAC is widely used in conjunction with both MD5 and SHA-1. This can be used to check the integrity of information stored or sent on insecure media.

## User Authentication

- Cryptographic hash of account password is stored in on-line database
  - Made more difficult to break by adding random “salt” to password
  - MD5-based hash by default, old modified DES version also available
- System hashes password given to login
- If passwords match, user is authenticated
- Utilities: **passwd, openssl passwd**

3-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 820 2994 or +1 (919) 754 3700.

### *Applications of Hashes*

One well-known application of cryptographic hashes is for storage of user passwords. When a user's password is set, it is hashed and the hash is stored in `/etc/shadow`. To make brute force attacks more difficult, a random public “salt” is added to the password, so that two users with the same password will have different password hashes. Originally, UNIX systems used a DES-based hash, but this is insecure by modern standards. By default, Red Hat Enterprise Linux stores passwords using a MD5-based password hashing algorithm originally developed for FreeBSD. By hashing the password, we ensure that the user's plaintext password is not stored on the system.

When a user attempts to log in, their entry in `/etc/shadow` is retrieved and the password entered is hashed using the same salt as was specified in their password entry. If that hash is the same as the one in `/etc/shadow`, then the user has successfully authenticated.

`openssl passwd` can generate valid password hashes without setting passwords. For example, to generate a standard MD5-based password hash:

```
[root@localhost ~]$ openssl passwd -1  
Password:  
Verifying'- Password:  
$1$ZZz.YxjI$PpTsm.4VCgRdeouevjy.H1.  
[root@localhost ~]$
```

In the output hash, `$1$` indicates this is a MD5 hash, and the value up to the next dollar sign (`ZZz.YxjI`) is the salt. See `ss1passwd(1)` for more information.

## Asymmetric Encryption

- Public key to encrypt, private key to decrypt
  - Based on trapdoor one-way functions
- Partial solution to key distribution problem
  - Can give the public key to everybody
- Algorithms: RSA, ElGamal
- Utilities: **gpg**, **openssl rsautl**

3-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

### *Public Key Encryption*

Asymmetric, or public key, encryption relies on two complementary keys; a public key and a private key. What one key encrypts, only the other key can decrypt. The public key can be made widely available. The private key must be kept absolutely secret. Anyone who has the public key may encrypt messages that can only be decrypted by the holder of the private key. This partially solves the key distribution problem of symmetric key encryption.

A number of public key encryption algorithms have been developed, the most widely-used being RSA. In general, RSA is not used as the bulk data cipher for a message. The reason for this is that RSA is limited in the size of the message it can encrypt, and available asymmetric algorithms are much slower than available symmetric algorithms. Instead, it is common to use RSA to transmit a secret symmetric session key securely, and switch to the faster key once the key exchange is complete.

The **openssl rsautl** utility can be used to encrypt and decrypt messages using RSA.

## Digital Signatures

- Private key to encrypt, public key to decrypt
  - Any holder of the public key can decrypt the message, but only a holder of the private key could have encrypted it
- For speed and improved security, hash the plaintext and sign that, then encrypt
- Algorithms: RSA, ElGamal, DSA
  - DSA is intended for digital signatures only

3-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (888) 626 2994 or +1 (919) 784 3700.

### *Digital signatures*

Digital signatures are the public-key analogue to message authentication codes. With a digital signature, the private key is used to encrypt a message or the hash of a message. Any user that has a copy of the public key can decrypt the signature and compare it to the message. If the resulting message or hash matches, then it was encrypted by the holder of the private key. Typically, the plaintext of a message is signed, and the entire resulting message is encrypted. There are arguments for encrypting and signing in either order.

DSA, the Digital Signature Algorithm (or DSS, the Digital Signature Standard) is a public key method intended only for producing digital signatures, developed by the NSA for NIST. It was developed while the RSA patent was still in force. Other public key algorithms such as RSA or ElGamal may also be used for digital signatures.

## Key Distribution

- Diffie-Hellman key exchange
  - Client and server agree on shared secret without transmitting it over the network
  - Use digital signatures to authenticate messages
- Public key encryption key exchange
  - Client generates symmetric session keys, sends to server encrypted with server's public key
- Still need to verify that the public key belongs to the other host

3-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Diffie-Hellman and Key Exchange*

There are a number of ways in which public key encryption can help solve the problem of symmetric key distribution. The first public key algorithm developed was the Diffie-Hellman key exchange method, which enables two hosts to agree on a shared secret without transmitting it over the network. Basically, both hosts publicly select a large prime,  $p$ , and a smaller generator,  $g$  (which is typically 2). Each host selects a random number, one  $x$  and the other  $y$ . The hosts then generate and exchange public keys, ( $gx \bmod p$ ) and ( $gy \bmod p$ ). The shared secret  $k = (gx)y \bmod p = (gy)x \bmod p$ . The attacker sees  $p$ ,  $g$ ,  $gx$ , and  $gy$ , but not  $x$  or  $y$ , or  $k$ . Without  $x$  or  $y$ , there is no efficient algorithm known that the attacker can use to compute  $k$ . This shared secret can then be used to generate symmetric session keys shared by both hosts. The use of "mod" here refers to "Modulus", an operator that reports the remainder. Hence  $a \bmod b$  is the remainder when dividing  $a$  by  $b$ .

The attacker can try to interfere with the key exchange process, so Diffie-Hellman is normally used in conjunction with DSA or RSA signatures to authenticate the messages.

Another way to distribute keys is to use normal RSA encryption and signatures. In this method, the client generates a secret symmetric session key and encrypts it with the server's public key. The server is the only host that can decrypt the message and recover the secret session key.

The main remaining problem here is that both key exchange methods rely on the client being able to get the server's legitimate public key. Many protocols ask the server itself or a central key server to provide the public key on request. But how do we know that an attacker did not spoof the public key? There needs to be a way that a client can verify a server's public key to make this work.

## Digital Certificates

- Trusted third party digitally signs public key
  - Certificate Authority (CA) has a public key that is known by everyone involved
- The resulting digital certificate contains
  - Server's public key and expiration date
  - Information about the owner of the key
  - Information about the CA and the CA's signature
  - Information on how the certificate may be used

3-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Digital Certificates and the CA*

Digital certificates are used to authenticate connections and distribute encryption keys. In a digital certificate, a third party trusted by both the client and the server vouches for the authenticity of the public key. This third party is called a certificate authority, or CA. A certificate contains the following information:

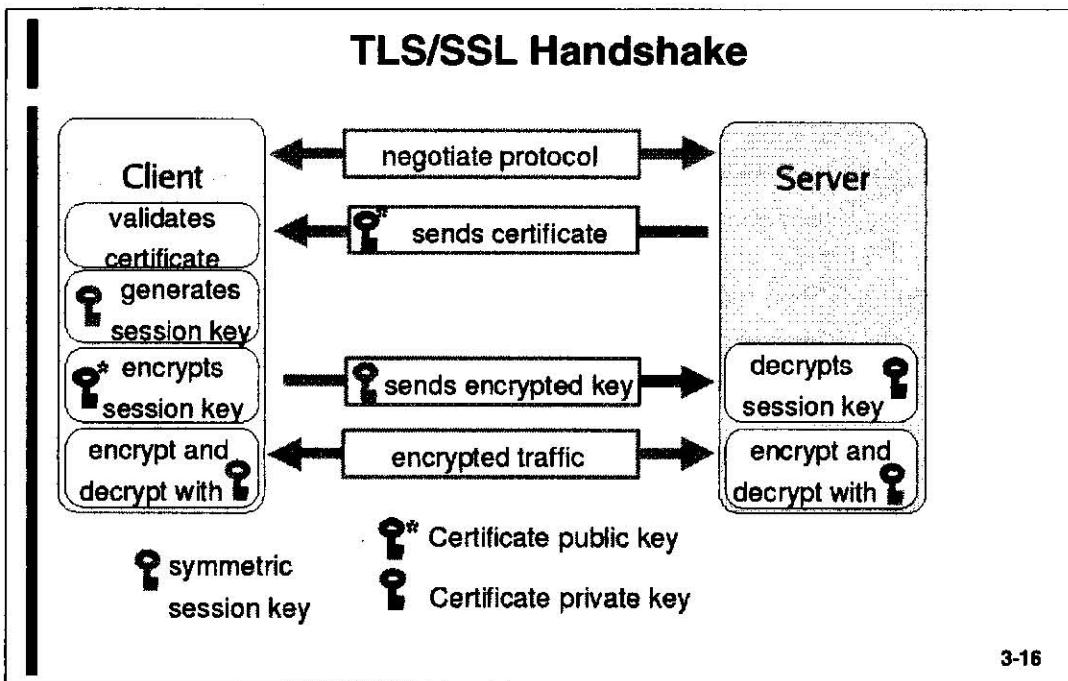
- The public key for the client or server and its expiration date
- Information identifying the legitimate owner of the key
- Information about how the certificate may be used
- Information about the CA (the issuer)
- A hash of the entire certificate, digitally signed with the CA's private key

Anyone with the CA's own certificate (and therefore its public key) can verify that the certificate for the server is signed and considered legitimate by the trusted CA.

Who vouches for the CA? A root CA has a self-signed certificate, in a sense vouching for itself. The administrator must determine what root CAs to trust. Many client programs have the CA certificates for widely trusted commercial certificate authorities installed in advance. Many sites run their own private CA so that they may control their own certificates.

TLS certificates use a standard format, X.509. The identities of the issuer and the owner of a certificate are recorded as a X.500 Distinguished Name.

An alternative public key infrastructure model, used by OpenPGP, uses a "web of trust" in place of a central CA. In this model, anyone may sign anyone else's certificate. Each participant must then decide which known signatures (or combination of known signatures) can be trusted to introduce unknown certificates.



For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ctraining@redhat.com](mailto:ctraining@redhat.com)> or phone toll-free (USA) +1 (800) 826 2994 or +1 (918) 754 3700.

TLS/SSL is a hybrid symmetric key and public key system. Slow public key cryptography is used to securely transmit a symmetric encryption/decryption key, known as a session key. The session key is small, and thus can be encrypted quickly using asymmetric RSA encryption. Once the session key is exchanged, subsequent communications are encrypted with fast and efficient symmetric key encryption and the slow public key is not used.

#### *TLS Handshake:*

1. Client requests a TLS connection from server. Then client and server negotiate key exchange, encryption, and message integrity protocols.
2. Server transmits digital certificate (and public key).
3. Client verifies authenticity of server certificate.
4. Client generates new symmetric session key using negotiated protocol.
5. Client transmits session key to server, encrypted with certificate's public key.
6. Server decrypts session key with its private key.
7. Client and server both switch to session key for further communication.

This handshake authenticates the server to the client, but not the client to the server. Commonly, the client provides other authentication to the server using the protected connection (a credit card number, a

username/password combination, and so on). However, client-side TLS authentication is also available. The server may request the *client* to send it a digital certificate to verify its identity before the client sends it the symmetric session key. The server may then refuse to accept the connection if the client certificate is not authorized or cannot be authenticated.

## Generating a RSA Private Key

- Generates private key file, includes information needed to compute public key
  - **make \*.key** in /etc/pki/tls/certs
  - **openssl genrsa** command
- Can encrypt the key for theft protection
  - Service will prompt for password when started
  - Attacker may be able to recover the password from main memory in any case

3-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 764 3700.

To make a TLS/SSL digital certificate, we first need a private RSA key. A Makefile that can be used to generate a 3DES-encrypted private key exists in /etc/pki/tls/certs:

```
$ make somename.key
```

This will prompt for a pass phrase for the private key. Any time a service using this key is started, users will be prompted for the pass phrase. The idea is that if an attacker manages to steal the key, they cannot use it without the pass phrase. There are two problems with this idea. First, if the service starts at boot, the server cannot be rebooted unattended or it will hang waiting for the password on boot. Second, if the service is running a clever attacker can probably recover the pass phrase from memory with a debugger or similar tool, however, this means the system is already compromised.

The unencrypted private key can be generated manually:

```
$ (umask 77; openssl genrsa 1024 > somename.key)
```

The umask 77 is to ensure that somename.key is readable only by root. Alternatively, an existing encrypted private key can be adjusted so that it is unencrypted or vice versa:

```
$ openssl rsa -in encrypted.key -out unencrypted.key      [decrypt key]  
$ openssl rsa -in unencrypted.key -des3 -out encrypted.key   [encrypt key]
```

Removing the passphrase eliminates the problem of pass phrase prompting at service startup, but at the expense of key security.

## Generating a Certificate Signing request (CSR)

- CA creates the final certificate from the Certificate Signing Request
  - Does not contain the private key!
  - CA must verify the identity of the CSR owner before signing
- Need a key pair for the CSR
  - make \*.csr creates key if necessary
  - openssl req command

3-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (819) 754 3700.

### Preparing the CSR

Whether using a commercial certificate authority or running a private CA, when creating a new CA-signed certificate, it is necessary to prepare a certificate signing request (CSR) first. This contains the basic information that will go in the final digital certificate, but is not yet signed by a CA. The file /etc/pki/tls/certs/Makefile can be used to generate a CSR, or it can be manually generated. Assuming *somefile.key* exists, run

```
$ (umask 77; openssl req -new -key somefile.key -out somefile.csr)
```

from within /etc/pki/tls/certs/. This will now prompt for identification information in X.500 format that will be associated with the public key in the final certificate. This is typically the country abbreviation, state or province name, city or locality name, the name of the organization, a person's name or the server's hostname, and a contact e-mail address.

Do not send the private key to the CA. All the CA needs is the certificate request.

To output the contents of a CSR in human-readable format, run

```
$ openssl req -noout -in somefile.csr -text
```

## Creating a CA-signed Certificate

- `$ openssl ca -in my.csr -out my.crt`
  - Verify information, then approve
- Every certificate issued should have a unique serial number (stored in `index.txt`)
- Keep a copy of the final CRT file so the certificate can be revoked if necessary

3-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Being a Certificate Authority*

`openssl ca` can be used to manage a small CA. In order to use it effectively, the CA build directory needs to be set up properly. The structure of this directory is specified in the [CA\_default] section of the `/etc/pki/tls/openssl.cnf` file. In addition, this section specifies what default settings should be used when generating a new server or user certificate. The database directive points to a file, `index.txt` by default, that stores information about all certificates issued, their unique serial numbers, and when they expire or whether they have been revoked due to compromise or loss. A copy of every certificate generated should be kept on the system, so that the certificate may be revoked later if necessary.

By default, the CA private key is stored in `./demoCA/private/cakey.pem`. If the private key to your CA is ever stolen, the attacker can issue any certificate they want for your organization. It is a good practice never to connect the machine used to manage the certificate authority to any network. Some organizations only load the private key on the server when it is in use, and store it off-line under lock and key the rest of the time.

## Certificate Expiration

- Expiration is a safety measure to help reduce the risk of a private key compromise
  - Short expiration periods are inconvenient, but speed recovery if the key is compromised
  - CA certificates typically have a long lifetime
  - Individual certificates have shorter lifetime
  - Be sure to deploy new certificates well before the old ones expire!

3-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

### *Expiration*

All digital certificates have a lifetime. The longer a key is in use, the more likely it is that it will be compromised. If a key expires after a certain date, then even if it is compromised and nobody knows it, it will eventually expire and become invalid. Therefore, it is a good practice to use fairly short lifetimes for certificates. However, if the lifetime is too short, certificate management may become difficult.

Typically, CA certificates have a much longer lifetime than regular certificates. Combine that with the fact that they can certify new certificates, and this is one more reason why CA certificates need to be very closely protected.

It is important to regularly replace certificates, and to have a procedure in place to do so smoothly. It is a good idea to start using new certificates well before the old ones expire. That way, if the new certificate does not work properly, you have time to replace it. This goes double for CA certificates. If your CA certificate is about to expire, generate a new CA certificate well in advance and start using it to sign all new requests and renewals. Most clients are capable of using multiple CA certificates at the same time.

## Certificate Revocation

- Revocation of a compromised key is hard
- One solution is a certificate revocation list (CRL)
  - Requires periodic downloads of the latest CRL
  - Many users or clients do not bother!
- Some key distribution systems use very short expiration times and simply allow certificates to expire naturally

3-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (650) 826 2904 or +1 (919) 754 3700.

### *Revocation*

A certificate authority may revoke a certificate at any time. This capability is used in order to ensure that compromised certificates are marked as invalid as soon as possible. It turns out that revocation is a very hard problem from a practical point of view.

One solution, used by TLS/SSL, is the certificate revocation list. The CA generates a list of all certificates that are no longer valid but that have not expired, and makes it available to the public. Users or clients must periodically download the latest CRL, or they may accept a known to be compromised certificate as valid.

In practice, many users or clients do not even try to use CRLs, so a compromised certificate continues to be valid until it expires naturally.

Other key distribution systems, such as Kerberos, dispense with CRLs. Instead, the certificates that are distributed have a very short lifetime, on the order of hours. If a certificate is compromised, it can be allowed to expire naturally and the attacker's window of opportunity is quickly closed.

## Managing a CRL

- To revoke a certificate:
  - `$ openssl ca -revoke stolen.crt`
- To build an up to date CRL from *index.txt*:
  - `$ openssl ca -gencrl -out crl.pem`
- Some programs can automatically download and import the latest CRL from a web URL
  - Mozilla, expects the CRL in DER format

3-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

### CRL Management

To revoke a certificate, run:

```
$ openssl ca -revoke badcert.crt
```

and to generate a new CRL run the command:

```
$ openssl ca -gencrl -out revoked.crl
```

Some clients, like Mozilla, expect the CRL to be in DER (Distinguished Encoding Rules - an encoding that guarantees bit quality) format, not PEM (Privacy Enhanced Mail) format. To convert the CRL, run:

```
$ openssl crl -in revoked.crl -outform DER -out revoked.der.crl
```

In addition, Mozilla honors the application/x-x509-ca-cert and application/x-pkcs7-crl MIME types for CA certificates and CRLs respectively. This can simplify certificate management.

To examine the contents of a CRL, you can use:

```
$ openssl crl -in revoked.crl -noout -text
```

## Operational Issues

- Programs and users fail to check identities
- Programs and users fail to use CRLs
- A third-party CA can issue bogus certificates
  - If standard CA is not to be trusted, may need to be disabled manually
- A private CA must make sure all clients have its CA certificate installed

3-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Problems with PKI*

In practice, TLS and public key infrastructures in general are not perfect. Some issues:

Names are hard. Programs and users often fail to check the identity associated with a certificate. Instead, they just check to see that it is signed by a trusted CA. This changes the model from “I, the CA, certify that this key belongs to the server” to “I, the CA, certify that this key belongs to somebody”.

Too many programs that do TLS/SSL do not support adequate certificate management, particularly for private CAs and certificate revocation lists. If they do support CRLs, most users do not know how to use them.

If a trusted third party is used as a CA, it can issue bogus certificates claiming to be from any site. Those bogus certificates will be, as far as client programs and users can tell, equally as valid as the correct ones. Assuming the CA holds the private keys for those bogus certificates, they can mount a man in the middle attack on communication. If they mistakenly issue a certificate for one site to someone else, the lack of adequate man in the middle attack on communication. to attacks using the bogus certificate until it expires.

Just because a CA signed a certificate, does that imply anything about what the holder of that certificate is permitted to do?

This can be difficult to configure. Some restrictions can theoretically be placed in the certificate by the CA, but not all clients actually honor them.

When running a private CA, make sure that all clients have the matching public CA certificate installed. If not, then users may get error messages or have other problems.

## GnuPG

- GnuPG is a powerful encryption tool
- To generate a key: `gpg --gen-key`
- Exchange public keys with others to encrypt email and verify signatures

3-24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email [ctraining@redhat.com](mailto:ctraining@redhat.com) or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Using gpg*

The Gnu Privacy Guard (GnuPG) allows you to generate keypairs for use in asymmetric cryptography. In other words, GnuPG and a Mail client can be used together to send digitally signed and/or encrypted email. First, you will need to create a keypair. This is done with the command: `gpg --gen-key`. You will be prompted for information about yourself, which will be associated with the key. This information will include the email address to be associated with the keys and a passphrase that will be required in order to sign, encrypt or decrypt anything using them. While the passphrase may be left blank, it is highly recommended that you set one. Otherwise, someone who compromised your account would also be able to view encrypted documents and send signed emails as you. Once configured to do so Evolution can use GnuPG to encrypt, decrypt and sign emails transparently, but the `gpg` command can be used to perform these operations manually on any file.

Some examples:

`gpg -a -o me@example.com.pubkey --export` exports your public key to a file

`gpg --import joe@example.com.pubkey` imports joe's key from a file into your collection

`gpg -a -r joe@example.com --sign --encrypt filename` encrypts and signs a file for joe

`gpg --decrypt filename` decrypts and verifies the signature on a file

## **End of Unit 3**

- Questions and Answers
- Summary
  - Network communication vulnerabilities
  - Introduction to cryptography
  - Working with digital certificates for TLS/SSL
  - Operational issues with TLS

**3-25**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

## **Lab 3**

### **Cryptography**

---

<b>Goal:</b>	Set up a private SSL certificate authority for managing digital certificates. Import private CA certificates and CRLs into client applications. Learn about OpenSSL troubleshooting tools.
<b>Estimated Duration:</b>	90 minutes
<b>System Setup:</b>	Standard Red Hat Enterprise Linux installation from the previous lab.

## **Sequence 1: Setting up a private Certificate Authority**

**Scenario:** In this sequence, you will set up your own CA for managing SSL digital certificates. We will use this CA in later sequences.

**Instructions:**

1. We want to set up /etc/pki/CA as your certificate authority's working directory. Open /etc/pki/tls/openssl.cnf in a text editor. Find the section labeled [ CA\_default ], and edit the following lines in the section to read:

```
dir = /etc/pki/CA
certificate = $dir/my-ca.crt
crl = $dir/my-ca.crl
private_key = $dir/private/my-ca.key
```

2. The [ req\_distinguished\_name ] section lists several default options you may want to change. In particular, you may want to set new defaults for C, ST, L, and O to appropriate values for your organization, such as

```
countryName_default = US
stateOrProvinceName_default = North Carolina
localityName_default = Raleigh
0.organizationName_default = Example, Inc.
```

Save the file and exit.

3. The /etc/pki/CA directory should be owned as root.root and have permissions 0700. It should contain a private subdirectory with the same permissions. Create some supporting directories for certificates and CRLs:

```
# mkdir /etc/pki/CA/{certs,crl,newcerts}
```

4. Create an empty certificate index:

```
# touch /etc/pki/CA/index.txt
```

In addition, create a file to indicate the next certificate serial number to be issued:

```
# echo 01 > /etc/pki/CA/serial
```

5. Next, while in /etc/pki/CA, you need to generate a private key and a self-signed CA certificate. You will be prompted for a passphrase, which will be needed later: use one you will remember:

```
# (umask 077; openssl genrsa -out private/my-ca.key -des3 2048)
```

For your CA certificate, take the defaults for CountryName, StateOrProvinceName, LocalityName, and Organization, and for CommonName use "Station X Certificate Authority". Set the other fields as you see fit:

```
# openssl req -new -x509 -key private/my-ca.key -days 365 > ✓  
my-ca.crt
```

The /etc/pki/CA/private/my-ca.key file is the private key for your CA. This file must be very carefully protected. The my-ca.crt file is the public CA certificate that will eventually be distributed to your users.

6. The CA should make its public certificate easily downloadable by clients. Install the **httpd** and **mod\_ssl** packages on your station and start the web server. Allow traffic through your firewall:
7. Create a directory /var/www/html/certs and copy /etc/pki/CA/my-ca.crt there. Check its permissions to make sure it is world-readable and labeled with the SELinux type *httpd\_sys\_content\_t*.

## Sequence 2: Self-signed certificates

**Scenario:** We will experiment with SSL using the secure IMAP service, and we will also use the web server later in this lab. Make sure that you have the dovecot and mutt packages installed.

**Instructions:**

1. Edit /etc/dovecot.conf and disable the insecure IMAP protocol.
2. Enable and start the secure IMAP service, and allow traffic through your firewall.
3. Copy the /usr/share/doc/dovecot-version/examples/mkcert.sh to your home directory.
4. Edit the mkcert.sh file and change the following lines:

```
OPENSSL=${OPENSSL-openssl}
SSLDIR=${SSLDIR-/etc/pki/dovecot}
OPENSSLCONFIG=${OPENSSLCONFIG-dovecot-openssl.cnf}
```

they should read:

```
OPENSSL=/usr/bin/openssl
SSLDIR=/etc/pki/dovecot
OPENSSLCONFIG=/etc/pki/tls/openssl.cnf
```
5. Remove the following files:  
`/etc/pki/dovecot/certs/dovecot.pem  
/etc/pki/dovecot/private/dovecot.pem`
6. Execute the mkcert.sh and create the new keys. Use your machine's fully-qualified hostname (*stationX.example.com*) as the Common Name in the SSL certificate.
7. Restart dovecot and verify it is running.
8. Configure mutt to use your IMAPS server. Log in as mary. Create a ~/ .mutt directory if it does not already exist. In that directory, create a muttrc file containing the following lines, where *stationX.example.com* should be your machine's hostname:

```
set folder=imaps://stationX.example.com/
set spoolfile=imaps://stationX.example.com/
set imap_force_ssl=yes
```

9. As mary, run mutt. You should get a warning about the self-signed certificate that dovecot is currently using. The details of the certificate are displayed, and you are asked whether you want to reject the certificate and the connection, or if you wish to accept it for this session.

Type **r** to reject the certificate, then **q** to quit mutt.

Why might it be a bad idea to accept this certificate?

## Sequence 3: Signing certificates

**Scenario:** The Dovecot IMAPS server keeps its TLS certificate in PEM format in the /etc/pki/dovecot/certs/dovecot.pem file. It keeps its private key, also in PEM format, in the file /etc/pki/dovecot/private/dovecot.pem. In this sequence, you will generate a new TLS certificate signed by your private Certificate Authority.

### Instructions:

1. As root, from your home directory, generate a new private key.
2. Then, create a new signing request. Make sure that the CountryName, StateOrProvinceName, LocalityName, and Organization are the same as those for your CA certificate. For CommonName, use the name of your workstation (*stationX.example.com*).

To see the contents of your CSR in human-readable form, run:

```
# openssl req -in dovecot.csr -noout -text
```

3. As the CA, sign the request.

You will be prompted for the password to the CA's private key. Review the request, sign it, and commit it. You now have a CA-signed certificate! In /etc/pki/CA/index.txt, you should see a line like this:

```
v 080131030651Z 01 unknown /C=US/ST=North  
Carolina/O=Example, Inc./CN=stationX.example.com
```

This indicates that the certificate with serial number 01 is Valid, expires on 2008-01-31 at 3:06:51 UTC, and gives the subject (fully distinguished name) of the certificate. In newcerts, you will also find another copy of your new certificate, named 01.pem. Keep this certificate on the CA; you may need it if the certificate is ever compromised and needs to be revoked.

4. Copy your signed certificate and your private key for dovecot to the correct location in /etc/pki:
5. As mary, run mutt to check her mail again. You should see the certificate warning again, but this time the issuer should read "*Station X Certificate Authority*". The warning is still displayed because mutt still does not trust that issuer as a CA, which you will fix in the next sequence.

Type **r** to reject the certificate and **q** to quit mutt.

## **Sequence 4: Installing the public CA certificate in Mutt**

Instructions:

1. 1.#As mary, download the CA certificate from your Certificate Authority.
2. Copy my-ca.crt to the ~mary/.mutt directory.
3. In ~mary/.mutt/muttrc, add the following line to configure the certificates in my-ca.crt as belonging to trusted Certificate Authorities:  
`set certificate_file=~/mutt/my-ca.crt`
4. Send an e-mail to mary so there is something in her mail spool.
5. As mary, run mutt to check her e-mail one more time. This time you should not see the certificate warning displayed, but should be prompted for the IMAP username and password for mary. Enter this information and check mary's mail.

Type **q** to quit mutt.

## Sequence 5: Revocation of a certificate

### Instructions:

1. First, configure Firefox to trust your CA certificate. Start the Firefox web browser. Select the menu **Edit, Preferences, Advanced**, and on the Security tab, click the "View Certificates" button. In the window that appears, on the "Authorities" tab, click the "Import" button, and import the `my-ca.crt` file.

2. It has come to your attention that the `dovecot.pem` has been compromised. To make sure we revoke the correct certificate, look up its subject and serial number:

```
# openssl x509 -in /etc/pki/dovecot/certs/dovecot.pem -noout <
    -serial -subject
```

3. Verify this information in `/etc/pki/CA/index.txt`.

4. Assuming that the certificate has serial number 01, revoke it using one of the certificates you have saved in `/etc/pki/CA/newcerts/`.

Look again in `index.txt`. The certificate should now be marked as R for revoked, and there should be a second timestamp indicating the time at which it was revoked.

5. Create a file to indicate the next certificate revoke list number:

```
# echo 00 > /etc/pki/CA/crlnumber
```

6. Next, build an up-to-date certificate revocation list.

Copy the crl to the correct location.

You can view the contents of the CRL with

```
# openssl crl -in my-ca.crl -noout -text
```

7. Firefox expects a CRL to be in a special binary format called *DER*. We need to convert the CRL from the Base64-encoded PEM format to the raw DER format.

8. Copy `my-ca-der.crl` to `/var/www/html/certs`. In Firefox, reload `http://stationX.example.com/certs/` and click on the `my-ca-der.crl` file. The server should send another MIME type to Firefox to let it know the `*.crl` file is a Certificate Revocation List. It should import automatically. You will be asked if you want to enable **Automatic Updates**. Go ahead.

## Sequence 6: Using gpg to exchange encrypted email

**Scenario:** *alice* and *bob* are users who would like to exchange information securely, using *GNU Privacy Guard* (*gpg*) to provide encryption services. You are to create the two users, and generate public/private key pairs for each. Next, *alice* will obtain *bob*'s public key, and encrypt information for him. *bob* will then decrypt the information.

**Instructions:**

1. Create the users *alice* and *bob*. Give each of them a password.
2. Generate a public/private key pair for each user.

You will be prompted for several key parameters. Choose default options. When asked for owner details, set the Real Name to *Alice*. Other information can be specified as you wish. You will also be asked for a passphrase. Use any passphrase you wish (and can remember!), or just press *Enter* twice for a null passphrase.

Repeat the same steps for the user *bob*, setting his Real Name to *Bobby* (*gpg* complains that *bob* is too short.)

3. Examine *alice*'s public and private keys, which are now stored in *pubring.gpg* and *secring.gpg*.
4. Have *bob* export his public key into an ASCII file that can be easily transferred to *alice*. Then have *alice* import *bob*'s public key into her public key ring.
5. Now that *alice* has acquired a copy of *bob*'s public key, and added it to her public keyring, she can encrypt messages to *bob*. Use the following sequence of commands to have *alice* send *bob* an encrypted copy of the */var/log/dmesg* file.

You might be warned that it is not certain that the key really belongs to the owner. Use the key anyway.

6. Now have *bob* check his mail using the simple command-line utility *mail*, and save *alice* message to a file.

You will be prompted for the filename of the plaintext result. Use the default *message.txt*. Note that *gpg* automatically performs the expected behavior, namely, decrypting the message with the appropriate private key. This default behavior can be modified with command-line arguments.

## **Sequence 7: Exercises**

### **Instructions:**

1. Using **gpg**, have **alice** send **bob** a message encrypted with a symmetric encryption scheme.
2. Have **alice** both sign and encrypt a message to **bob**. What additional piece of information will **bob** need before he is able to verify the signature?
3. Have **alice** create a detached signature for her message, and send **bob** both parts. Have **bob** verify the detached signature.
4. Have **bob** sign **alice**'s public key, thereby helping **alice** convince other people that she is the appropriate owner for the public key.

## Sequence 1 Solutions

6. The CA should make its public certificate easily downloadable by clients. Install the **httpd** and **mod\_ssl** packages on your station and start the web server. Allow traffic through your firewall:

```
# yum install httpd mod_ssl  
  
# chkconfig httpd on; service httpd start  
  
# iptables -A RHS333 -m multiport -p tcp --dport 80,443 -j ACCEPT  
  
# service iptables save
```

7. Create a directory **/var/www/html/certs** and copy **/etc/pki/CA/my-ca.crt** there. Check its permissions to make sure it is world-readable and labeled with the SELinux type **httpd\_sys\_content\_t**:

```
# cp /etc/pki/CA/my-ca.crt /var/www/html/certs/  
  
# chcon -t httpd_sys_content_t /var/www/html/certs/
```

## Sequence 2 Solutions

1. Edit `/etc/dovecot.conf` and disable the insecure IMAP protocol.

```
#protocols = imap imaps
```

should read

```
protocols = imaps
```
2. Enable and start the secure IMAP service, and allow traffic through your firewall.

```
# service dovecot start
```

```
# chkconfig dovecot on
```

```
# iptables -A RHS333 -p tcp --dport 993 -j ACCEPT
```
3. Copy the `/usr/share/doc/dovecot-version/examples/mkcert.sh` to your home working directory:

```
# cd /root
```

```
# cp /usr/share/doc/dovecot-version/examples/mkcert.sh .
```
5. Remove the following files:

```
/etc/pki/dovecot/certs/dovecot.pem
```

```
/etc/pki/dovecot/private/dovecot.pem
```

```
# rm -f /etc/pki/dovecot/certs/dovecot.pem
```

```
# rm -f /etc/pki/dovecot/private/dovecot.pem
```
6. Execute the `mkcert.sh` and create the new keys. Use your machine's fully-qualified hostname (`stationX.example.com`) as the Common Name in the SSL certificate.

```
# sh mkcert.sh
```
7. Restart dovecot and verify it is running.

```
# service dovecot restart
```

```
# service dovecot status
```

```
# tail /var/log/messages
```

```
# tail /var/log/maillog
```
9. Why might it be a bad idea to accept this certificate?
  - Someone may have forged the certificate and signed it themselves.

## Sequence 3 Solutions

1. As root, from your home directory, generate a new private key:

```
# ( umask 77; openssl genrsa 1024 > dovecot.key )
```

2. Then, create a new signing request. Make sure that the CountryName, StateOrProvinceName, LocalityName, and Organization are the same as those for your CA certificate. For CommonName, use the name of your workstation (*stationX.example.com*):

```
# ( umask 77; openssl req -new -key dovecot.key -out dovecot.csr )
```

3. As the CA, sign the request:

```
# openssl ca -in dovecot.csr -out dovecot.crt
```

4. Copy your signed certificate and your private key for dovecot to the correct location in /etc/pki:

```
# cp ~/dovecot.key /etc/pki/dovecot/certs/dovecot.pem  
# cat dovecot.crt >> /etc/pki/dovecot/certs/dovecot.pem  
# cp /etc/pki/dovecot/certs/dovecot.pem /etc/pki/dovecot/private/dovecot.pem  
# service dovecot restart
```

## **Sequence 4 Solutions**

1. As mary, download the CA certificate from your Certificate Authority:

```
# lftpget -v http://stationX.example.com/certs/my-ca.crt
```

Copy my-ca.crt to the ~mary/.mutt directory.

```
# cp /etc/pki/CA/my-ca.crt ~mary/.mutt/
```

Send an e-mail to mary so there is something in her mail spool:

```
# echo Hello | mail -s "Test" mary
```

## Sequence 5 Solutions

4. Assuming that the certificate has serial number 01, revoke it using one of the certificates you have saved in /etc/pki/CA/newcerts/:

```
# openssl ca -revoke /etc/pki/CA/newcerts/01.pem
```

6. Next, build an up-to-date certificate revocation list:

```
# openssl ca -gencrl -out my-ca.crl
```

Copy the crl to the correct location:

```
# cp my-ca.crl /etc/pki/CA/
```

7. Firefox expects a CRL to be in a special binary format called *DER*. We need to convert the CRL from the Base64-encoded PEM format to the raw DER format:

```
# openssl crl -in my-ca.crl -outform DER -out my-ca-der.crl
```

8. Copy my-ca-der.crl to /var/www/html/certs. In Firefox, reload <http://stationX.example.com/certs/> and click on the my-ca-der.crl file. The server should send another MIME type to Firefox to let it know the \*.crl file is a Certificate Revocation List. It should import automatically. You will be asked if you want to enable **Automatic Updates**. Go ahead.

```
# cp /etc/pki/CA/my-ca-der.crl /var/www/html/certs/
```

## Sequence 6 Solutions

1. Create the users alice and bob. Give each of them a password.

```
# useradd alice  
# passwd alice  
# useradd bob  
# passwd bob
```

2. Generate a public/private key pair for each user.

You will be prompted for several key parameters. Choose default options. When asked for owner details, set the Real Name to Alice. Other information can be specified as you wish. You will also be asked for a passphrase. Use any passphrase you wish (and can remember!), or just press *Enter* twice for a null passphrase.

```
[alice@stationX] $ gpg --gen-key
```

3. Examine alice's public and private keys, which are now stored in pubring.gpg and secring.gpg.

```
[alice@stationX] $ ls ~/.gnupg/  
[alice@stationX] $ gpg --list-keys  
/home/alice/.gnupg/pubring.gpg  
-----  
pub 1024D/B6B4B635 2010-09-23 Alice (demo key) <alice@stationX.c ↵  
om>  
sub 1024g/27B5C1BF 2010-09-23  
  
[alice@stationX] $ gpg --list-secret-keys  
/home/alice/.gnupg/secring.gpg  
-----  
sec 1024D/B6B4B635 2010-09-23 Alice (demo key) <alice@stationX.c ↵  
om>  
ssb 1024g/27B5C1BF 2010-09-23
```

4. Have bob export his public key into an ASCII file that can be easily transferred to alice. Then have alice import bob's public key into her public key ring.

```
[bob@stationX] $ gpg --export --armor Bobby > /tmp/bob.key  
[bob@stationX] $ cat /tmp/bob.key  
  
[alice@stationX] $ gpg --import /tmp/bob.key  
[alice@stationX] $ gpg --list-keys  
/home/alice/.gnupg/pubring.gpg
```

```

-----
pub 1024D/6B51AC3D 2010-09-25 Alice (test key)
<alice@stationX.example.com>
sub 1024g/2E5A824A 2010-09-25

pub 1024D/E4BF6FDC 2010-09-25 Bobby (test key)
<bob@stationX.example.com>
sub 1024g/D17F52CB 2010-09-25

```

- Now that `alice` has acquired a copy of `bob`'s public key, and added it to her public keyring, she can encrypt messages to `bob`. Use the following sequence of commands to have `alice` send `bob` an encrypted copy of the `/var/log/dmesg` file.

```
[alice@stationX]$ cp /var/log/dmesg message.txt
[alice@stationX]$ gpg --encrypt --armor --recipient Bobby \
message.txt
```

You might be warned that it is not certain that the key really belongs to the owner. Use the key anyway.

```
[alice@stationX]$ head message.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.2 (GNU/Linux)
Comment: For info see http://www.gnupg.org
```

```
hQEAOypkljvRf1LLEAP/Xh1eZKV6Zh4na5WnjT6FJLAKoaECmo/EJ3mIB45gDInV
oVhqX2zf5ndw+OP98hN1WRuXro+7tVeX3FYWSzOleQKPIDzJhfTlbCFtoEo7+PA8
9vXVkPeRdzEp2NMrQe+XP09RSiqbDGETWNQE1RC9hrG8yeYY1WYqvTeCoY9u2E4D
/2+OmxxXIdXXj+3S1UZtGL7J0+8MnyT1Kb54BmXdbA28ASMsEJI5zWkJxNkcuGYF
KEKczYtYFP0gengONhwzsfi/oXuL2cURtnmIjXAAhm62gVf9L8NokUqg4eB/ejtA
```

```
[alice@stationX]$ mail -s "here it is" bob < message.txt.asc
```

- Now have `bob` check his mail using the simple command-line utility `mail`, and save `alice` message to a file.

```
[bob@stationX]$ mail
Mail version 8.1 6/6/93. Type ? for help.
"/var/spool/mail/bob": 1 message 1 new
>N 1 alice@stationX.example Sun Sep 24 23:00 71/3947 "here it is"
& w message_from_alice
"message_from_alice" [New file]
& q
```

```
[bob@stationX] $ less message_from_alice  
[bob@stationX] $ gpg message_from_alice
```

You will be prompted for the filename of the plaintext result. Use the default `message.txt`. Note that `gpg` automatically performs the expected behavior, namely, decrypting the message with the appropriate private key. This default behavior can be modified with command-line arguments.

## Sequence 7 Solutions

1. Using **gpg**, have **alice** send **bob** a message encrypted with a symmetric encryption scheme.
  - a. When encrypting and decrypting, use **gpg -c** or **gpg --symmetric test-file**. This asks for **alice**'s signature.
2. Have **alice** both sign and encrypt a message to **bob**. What additional piece of information will **bob** need before he is able to verify the signature?
  - a. Bob will need Alice's public key to verify her GPG signature, as well as the signed, encrypted message.
  - b. Alice needs to export her public key: **gpg --export --armor Alice > /tmp/alice.key**
  - c. Then she needs to sign and encrypt the message: **gpg -sea message.txt**
  - d. She will be prompted for her passphrase, then when prompted will add **Bobby** as a recipient; when prompted again, hit return to end the recipient list and create the encrypted file. Copy **message.txt.asc** to **/tmp**.
  - e. Then Bob needs to import her public key: **gpg --import /tmp/alice.key**
  - f. Finally, Bob needs to decrypt **/tmp/message.txt.asc** and verify that it came from Alice: **gpg -d /tmp/message.txt.asc**
3. Have **alice** create a detached signature for her message, and send **bob** both parts. Have **bob** verify the detached signature.
  - a. **alice** will create the detached signature using **gpg -b** or **gpg --detach-sign**. Bob will verify the detached signature with **gpg --verify**.
4. Have **bob** sign **alice**'s public key, thereby helping **alice** convince other people that she is the appropriate owner for the public key.
  - a. **bob** will use the following command: **gpg --sign-key Alice**

## **Unit 4**

### **Logging and NTP**

**4-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Objectives

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

- Configure NTP for time synchronization
- Configure syslog for central logging

4-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Time Synchronization

- Some security protocols require well-synchronized clocks
  - Protection from "replay attacks"
  - Kerberos, DNSSEC, etc.
- Time synchronization makes system logs easier to analyze
- Workstation hardware clocks tend to drift over time without correction

4-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <ctraining@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Service Profile: NTP

- Type:`#System V-launched daemon`
- Packages:`#http`
- Daemons:`#ntpd`
- Script:`#ntpd`
- Ports:`#123 / udp (ntp)`
- Configuration:`#/etc/ntp.conf, /etc/ntp/*`
- Related:`#system-config-date`

4-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2904 or +1 (919) 754 3700.

Service	Permissions	Normally runs as
<code>/usr/sbin/ntpd</code>	755, root.root	user ntp, group ntp
File	Permissions	Purpose
<code>/etc/ntp.conf</code>	644, root.root	Main configuration file, readable by ntp but only writeable by root
<code>/etc/ntp/</code>	755, root.root	Config file directory; should not be generally writeable!
<code>/etc/ntp/data/</code>	755, ntp.ntp	Config file directory; writeable by ntp
<code>/etc/ntp/keys</code>	600, ntp.ntp	<i>Contains encryption keys!</i> Readable only by ntp
<code>/var/lib/ntp/drift</code>	644, ntp.ntp	Hardware clock drift statistics. Read-write by ntp
<code>/etc/ntp/step-tickers</code>	644, root.root	Optional servers to use for initial clock synchronization at boot
<code>/etc/sysconfig/ntpd</code>	644, root.root	Passes command-line options to ntpd. Sets user to ntp.

The distribution ships with `/etc/ntp/keys` owned by root with permissions 600; unfortunately, the ntpd service will not be able to read the encryption keys with the file in that mode.

Supplemental Utility	Permissions	Purpose
<code>/usr/sbin/ntpdate</code>	755, root.root	Set the system clock once from an NTP server.
<code>/usr/sbin/ntp-genkeys</code>	755, root.root	Generate encryption keys for security.
<code>/usr/sbin/ntpq</code>	755, root.root	Standard NTP query tool.
<code>/usr/sbin/ntpdc</code>	755, root.root	Special NTP query tool.
<code>/usr/sbin/ntptrace</code>	755, root.root	Trace chain of NTP servers back to initial time source.

Documentation on NTP is available at <http://www.ntp.org>.

## Basic Design of NTP

- Any NTP client is a potential server
  - "stratum-1" has radio clock or atomic clock
  - "stratum-2" server
    - Is a client of a stratum-1 server
    - May serve time data to stratum-2 or higher (stratum-3, etc.) clients/servers
- Work group usually has 3 low-stratum servers most clients use for time

4-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (519) 784 3700.

The lower the stratum of a NTP server, the closer it is to a highly accurate source of time. A stratum-1 server has direct access to a good time source, such as an atomic clock or a WWV, or GPS radio receiver. The highest stratum allowed is stratum-15; a "stratum-16" server is not synchronized. It is rare for an organization to use more than two or three strata internally.

NTP clients will send a synchronization request to their server(s) every fifteen minutes or so – more frequently if the service has started recently or there have been communication problems with the server(s). The clients will attempt to determine the most accurate response based on a number of criteria and use that response to synchronize their system clocks.

An organization usually arranges to have three low-stratum servers as primary time sources. The majority of client workstations will then be told to synchronize from those three servers. So if an organization has three central time servers with radio clocks at stratum-1, then the rest of the client workstations synchronizing from them are at stratum-2. The central servers may not have radio clocks, but may themselves synchronize from a public stratum-1 or stratum-2 time server. For example, the central servers might synchronize from public stratum-2 servers (placing themselves at stratum-3) and the organization's client workstations at stratum-4.

Having three central time servers allows clients to reject bogus synchronization messages if one of the servers' NTP daemons or clocks malfunctions. It is possible for a client to synchronize with fewer time servers if necessary but it is less secure.

## Simple Client Configuration

- **ntpdate:** used to initially set clock
- Clients should have three NTP servers
  - Can get by with one
  - With three, clients can tell if a server is malfunctioning and ignore it
- In /etc/ntp.conf specify server
  - server 192.168.10.3

4-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 920 2994 or +1 (919) 754 3700.

Before starting an NTP server, the workstation's clock should already be set close to the correct time; within fifteen minutes or so is acceptable. ntpdate can be run as root to get an initial clock setting from a synchronized NTP server: ntpdate *Hostname\_or\_IP\_Address*. The client clock will immediately begin to drift from the correct time so ntpd should be and running soon after setting the time in order.

Configure ntpd by setting up a valid /etc/ntp.conf . On a client, a minimal file should contain at least a server line for each server the client will synchronize with and a driftfile line indicating a file that will be used to store information about the inaccuracy of the local hardware clock. The drift file should be readable and writable by ntpd.

```
server 192.168.0.1
server 192.168.0.2
server clock.example.com
driftfile /var/lib/ntp/drift
```

Once ntpd has started, the client will need to poll each of the servers several times so it can learn how accurate they are and establish synchronization. This normally takes about five minutes.

This configuration is sufficient to establish basic time synchronization, but no authentication or session security is really established yet.

## Server Configuration

- Similar to client with three time sources
- Peer with the other two workgroup servers
- Third: an external *server*
- Radio clock or GPS (uses clock driver)
- Public NTP server
- Last resort: local system clock (LCL)
- Inaccurate source of time; use fudge to advertise at a very high stratum (stratum-10 or higher)

4-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 625 2994 or +1 (919) 754 3700.

Basic configuration of an organizational server is very similar to that of a end-user client. Again, if possible, you want to have at least three NTP servers as time sources. Two of these are normally the other two organizational servers; these are configured as *peers*, which allows them to synchronize with each other or provide time data to each other. This is done by using the keyword peer instead of the keyword server.

The third server should be an external time source and should be different for each time server. These are identified with the server keyword and can either be public NTP servers with access to a good time source or device drivers to access a hardware clock attached to the local server. These drivers can usually be distinguished as they are given a fake IP address on the 127/8 subnet reserved for the localhost.

LCL is a special driver for the local hardware clock. If there is a system that cannot get time from the Internet and an affordable and accurate clock is not available, the server could use the motherboard hardware clock as an inaccurate time source. Since motherboard clocks tend to be very low quality and gain or lose time quickly, if one is set up, use the fudge command so that the server advertises it at a very high (and therefore unreliable) stratum, at least 10.

```
peer 192.168.0.2          # a server peer
server 10.15.0.4          # a public NTP server on 10.15.0.4
server 127.127.29.0        # a Trimble GPS (on a serial port)
server 127.127.1.0        # the motherboard hardware clock
fudge 127.127.1.0 stratum 10 # ...which is advertised as stratum-10
```

## NTPv4 Authentication

- Uses keys for authentication and session security
  - Helps prevent NTP clock skew attacks
- **ntp-keygen** to generate key file
  - File should only be readable by user ntp
- Enable in /etc/ntp.conf
  - Each server or peer needs key option

4-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <ctraining@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The problem with the traditional NTP protocol is that time synchronization messages do not have any session security. A *time skew attack* could be launched in which an attacker sends fake NTP responses to a host and cause it to gradually shift its time. Since some protocols authenticate using tickets that expire at a certain time, this could be used in conjunction with a *replay attack* where the attacker replays a captured but expired ticket. By changing the time on the server the attacker may convince it that the ticket is still valid.

NTP version 4 added an authentication feature that uses keys to authenticate NTP messages. A key file can be generated with the command:

```
$ ntp-keygen -T
```

The -T option indicates that we will be using a trusted key. This will generate a set of files in the current directory with a name like `ntpkey_RSAkey_stationX.example.com.3218471490` and one called `ntpkey_RSA-MD5cert_stationX.example.com.3218471490`, where the number is a timestamp in NTP format. This also creates a set of symbolic links: `ntpkey_cert_stationX.example.com` and `ntpkey_host_stationX.example.com`. On the other machines in the network, create keys with similar names with the following command:

```
$ ntp-keygen
```

In NTP version 4, a new mechanism called autokey was introduced. On each client or server, add a key option to `ntp.conf`:

```
server      192.168.0.2 autokey
keys        /etc/ntp/keys
```

After generating the keys mentioned above, we restart the other machines ntp server first. Then the main server, where generation of the key was done with the -T option, is started. After a while, the servers will start communicating using these keys.

It is recommended to change the keys once a month.

## restrict and Access Control

- **restrict lines in ntp.conf used for IP-based access control**
  - Supplement for NTPv4 Authentication
  - Line with most specific netmask used
- **Flags determine restrictions in effect**
  - If no flags specified, then no restrictions
  - ignore, noquery, noserve, etc.

4-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

restrict lines in /etc/ntp.conf provide an IP-address method of access control. The restrict option is not intended as a replacement for NTPv3 Authentication, but as a supplement. The syntax of the command is:

`restrict IP-address [flags...]` - Applies to a single host.

`restrict network mask netmask [flags...]` - Applies to an entire network.

`restrict default [flags...]` - Default; applies to any host.

The restrict line with the most specific netmask determines the restrictions in effect. If the special keyword default is used instead of an IP address, then the netmask is assumed to be 0.0.0.0 (least specific). Otherwise, if no mask statement is used, then the netmask is assumed to be 255.255.255.255 (most specific).

The address specification is followed by zero or more flags that determine which restrictions should be in effect for hosts that use the rule. If no flags are specified, then there are no restrictions in force. Some flags that can be specified include:

ignore	ignore all packets from these hosts
noquery	ignore all status queries and configuration requests (with ntpq and ntpdc)
nomodify	ignore remote reconfiguration requests, allow queries and use for time service
noserve	ignore requests for time service, but allow queries and remote reconfiguration
notrust	do not use these hosts as a time server (useful if using server lines with host names)

Some sample restrict lines:

```
restrict default ignore          # Refuse access by default.  
restrict 127.0.0.1              # No restrictions!  
restrict 192.168.1.0 mask 255.255.255.0 nomodify noquery
```

## Service Profile: syslog

- Type:**#**System V-launched daemon
- Packages:**#**sysklogd
- Daemons:**#**syslogd, klogd
- Script:**#**syslog
- Ports:**#**514/udp (syslog)
- Configuration: /etc/syslog.conf, /etc/sysconfig/syslog

4-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 426 2984 or +1 (919) 754 3700.

Service	Permissions	Normally runs as
/sbin/syslogd	755, root.root	user <i>root</i> , group <i>root</i>
/sbin/klogd	755, root.root	user <i>root</i> , group <i>root</i>
File	Permissions	Purpose
/etc/syslog.conf	644, root.root	Main configuration file for syslogd and klogd.
/etc/sysconfig/syslog	644, root.root	Pass command-line options to syslogd and klogd. Includes listening for messages from other hosts.

## Weaknesses of syslog

- syslog is a connectionless UDP protocol
  - Packets may be lost or arrive out of order
- No authentication or session security
  - Message injection and replay attacks
  - Denial-of-service attacks
- Data is not encrypted on the network
  - Sensitive information in log messages may be sniffed in transit

4-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

## Protecting Log Servers

- Block log messages from external hosts
- Keep /var/log on a separate partition
- Emerging standards to watch:
  - RFC 3195 "syslog-reliable"
    - New secure log protocol using 601/tcp
  - "syslog-sign" (in Internet-Draft)
    - Adds authentication/session security to existing syslog protocol

4-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2984 or +1 (919) 754 3700.

Protecting a standard centralized syslog server is difficult due to the flaws in the protocol itself. There are a number of different measures that can limit this exposure:

syslogd should not be allowed to accept messages on port 514/udp from hosts that should not be sending log messages to your server. This can be accomplished with appropriate iptables rules, for example. Remember that this does not prevent a program or user on an authorized machine from sending the server a bogus log message.

One denial of service attack on a log server is to send it so many log messages that its hard drive fills up. It is a good idea to keep /var/log on a separate partition so that even if that partition fills up, other programs can still operate normally. Keep in mind however, that since syslogd will not have any free space on its partition, log messages might be lost.

There are some emerging standards being worked on by the IETF to address the flaws in the syslog protocol:

The first of these to be released is RFC 3195 ("Syslog-Reliable"). This is a totally new secure logging protocol that communicates on port 601/tcp. Syslog-Reliable encrypts all log data transmitted across the network, as well as guaranteeing authentication and session security. This standard was finalized in November 2001.

The second of these standards is still in Internet-Draft form, and is known as "syslog-sign". This protocol takes the existing syslog standard and adds authentication and session security features to it. It still uses port 514/udp, but can detect if log messages have been lost or are received out of order. Data transmitted across the network is not encrypted, however.

A BSD-licensed syslog daemon for Linux capable of using both the standard syslog protocol as well as the new Syslog-Reliable protocol was recently released, SDSC Secure Syslog. While too new to be included in the distribution at this time, it may be worth investigating as a more secure alternative to *sysklogd*. Check out <http://security.sdsc.edu/software/sdsc-syslog/> for more information.

## **End of Unit 4**

- Questions and Answers
- Summary
  - Time synchronization with NTP
  - Centralized logging with syslogd

**4-13**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <tctraining@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Lab 4

### Time Synchronization and syslog

---

<b>Goal:</b>	To install and securely configure a NTP server and to practice simple configuration of syslogd
<b>Estimated Duration:</b>	60 minutes
<b>System Setup:</b>	The <code>ntp</code> package is required. Use <code>rpm -q</code> to determine whether this package is installed. If not, you may install it with:  <code>yum install ntp</code>  The <b>system-config-securitylevel</b> firewall should still be enabled by default.
<b>Situation:</b>	The instructor should provide you with information on how to access to a workstation in the cracker.org domain, so that you can test some of your configuration settings.

## **Sequence 1: Configuring a basic NTP server**

**Scenario:** In this sequence you will set up a host to be a top-stratum NTP server for a workgroup, rather than simply an NTP client.

**Instructions:**

1. Replace the existing `/etc/ntp.conf` file with one that contains only the following lines:

```
peer 192.168.0.Y      # your partner's workstation
server 192.168.0.254   # a server outside your organization
server 127.127.1.0     # a local hardware clock
fudge 127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
```

The first line defines your partner's workstation as a peer of yours. In other words, you are both running organizational servers which are sharing time data and are operating at approximately the same stratum. (If you are partnered with more than one person, include peerlines for each of your partners.) Replace the `Y` in the IP address with your partner's station number.

The second line makes your workstation a client of a server at 192.168.0.254, which presumably has access to a high quality source or sources of time data which your peer server does not.

The third line simulates a configuration line for a WWV radio clock or other local hardware time source. Since we do not have such a source, we will use the low quality clock on the motherboard as a stand-in, and mark it as low quality with the following fudgeline. It is not good practice to use the low quality motherboard clock as a time source unless there is no other option.

The last line specifies a file in which to store information about how your hardware clock tends to drift over time.

2. Now you need to set your hardware clock so that it's roughly correct. The time server on `server1.example.com` should be synchronized already.
3. Permit other hosts to contact your NTP server through the firewall.
4. Start up the NTP server, and configure it to start after reboot.

## Sequence 2: Monitoring NTP servers

**Scenario:** Now we will use a tool called **ntpq** to monitor the status of the NTP servers.

**Instructions:**

1. Run the **ntpq** command at the shell prompt. At the **ntpq>** prompt, type the following command:

```
ntpq> peer
```

You should see output that looks something like the following:

```
remote refid          st t when poll reach   delay offset  ↵
jitter ↵
=====
server1 navobs1.wustl.e 4 u 10    64   1      2.702  -17.182 ↵
 0.008
stationY .INIT.        16 u -64    0    64     0.000   0.000 ↵
 4000.00
LOCAL(0) .LOCL.        10 l  8    64   1      0.000   0.000 ↵
 0.008
```

**remote** indicates one of your time sources; **refid** specifies what that time source is currently synchronized to, and **st** is the current stratum of the remote server. The **ntpq** command **hostnames no** will cause **peer** to display IP addresses rather than (truncated) host names in the first two fields. **poll** indicates how often, in seconds, the remote server is queried, and **when** lists how many seconds have passed since the last poll. **reach** is an octal register indicating how reliably the host can be reached (377 is the best value possible). The last three fields are the remote clock's reported **delay/offset/jitter** in milliseconds.

In the example above, the NTP server has not been in reliable touch with its remote time servers for long enough to synchronize to any of them. The stationY.example.com server is itself unsynchronized (stratum 16, refid 0.0.0.0) right now. It normally takes about five minutes for a new NTP server to synchronize to a running server.

2. Still in **ntpq**, connect to your partner's NTP server and see how it is doing. Type:

```
ntpq> host stationY.example.com
```

Now any query commands you type will be sent to the remote NTP server, rather than your own. Try typing a **peer** command, then use **host** to switch back to your own NTP server and compare to your own server. In the next section, we will limit the ability of remote **ntpq** clients to access your NTP server in this way.

3. After about five minutes, your NTP server should synchronize with one or more of its servers. In **ntpq** on your own station, run the **peer** command again:

```
remote      refid          st t when poll reach delay offset ✓
jitter ✓
=====
===
*server1  navobs1.wustl.e  4 u 55   64   377  2.210 -18.092 ✓
  0.075
+stationY server1.example 5 u 33   64    17   0.196 -10.663 ✓
  0.819
LOCAL(0) LOCAL(0)        10 l 53   64   377  0.000  0.000 ✓
  0.008
```

The character in front of the **remote** field shows your synchronization status with that host. A space indicates the peer isn't providing useful time (in the example above, due to being at a very low stratum relative to the other servers); \* is the "system peer", providing the best data; + indicates servers with times close to the system peer, and - indicates servers with less accurate time data. The \* and + times are combined to estimate the "true" time.

## Sequence 3: Logging to a central log host

Instructions:

1. First, edit `/etc/sysconfig/syslog` to allow remote `syslog` messages.
2. Assuming that you still have the `system-config-securitylevel` firewall enabled, we need to set up a packet filtering rule to accept log messages sent by authorized machines to the local syslog port (514/udp). Other messages should continue to be blocked.

Save your work.

3. Restart `syslogd` on stationX.example.com so your changes to `syslogd` take effect.
4. Set up `syslogd` on your second machine to send some log messages to stationX. In `/etc/syslog.conf`, append the following line:

```
user.* @stationX.example.com
```

We also want to save all mail to stationX. Find the line with `mail` and add this line below:

```
mail.* @stationX.example.com
```

5. Restart `syslogd` on stationY. Now the second machine will send log messages for the `USER` facility to your new central log server on stationX.example.com.
6. Test the new setup from stationY by using `logger` to generate a syslog message:

```
# logger -i -p user.info -t yourname "This is a test"
```

Does the message appear in the `/var/log/messages` file on stationX?

While we are at it, try:

```
# logger -i -p mail.info -t yourname "This is a test"
```

Does the message appear in the `/var/log/maillog` file on stationX?

7. Implement steps 4 through 6 on the cracker.org workstation. (Cooperate with other students in class who may be editing the file at the same time.) Does the message from that station appear in your `/var/log/messages` file?

## **Sequence 1 Solutions**

2. Now you need to set your hardware clock so that it's roughly correct. The time server on server1.example.com should be synchronized already.

```
# ntpdate -b server1.example.com
```

3. Permit other hosts to contact your NTP server through the firewall:

```
# iptables -A RHS333 -p udp --dport 123 -j ACCEPT
```

```
# service iptables save
```

4. Start up the NTP server, and configure it to start after reboot:

```
# service ntpd start
```

```
# chkconfig ntpd on
```

## Sequence 3 Solutions

1. First, edit /etc/sysconfig/syslog to allow remote syslog messages.

```
SYSLOGD_OPTIONS="-r -m 0"
```

2. Assuming that you still have the **system-config-securitylevel** firewall enabled, we need to set up a packet filtering rule to accept log messages sent by authorized machines to the local syslog port (514/udp). Other messages should continue to be blocked.

```
# iptables -A RHS333 -s 192.168.0.7 -p udp --dport 514 -j ACCEPT
```

Save your work.

```
# service iptables save
```

3. Restart **syslogd** on stationX.example.com so your changes to **syslogd** take effect.

```
# service syslog restart
```

## **Unit 5**

### **BIND and DNS Security**

**5-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Objectives

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

- DNS Review
- Configuration and use of TSIG
- Restrictions of zone transfers and queries
- "Split namespaces" and views
- Configuration of a chroot() name server
- Logging: categories and channels

5-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2904 or +1 (919) 754 3700.

## Vulnerabilities

- Use of your DNS information to launch other attacks
- Spoofing of legitimate host information by a third party
- Compromise of *named* daemon

5-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Resolutions

- Use TSIG keys with access control
- Restrict zone transfers, recursive queries, or all queries
- Run BIND in a chroot() jail
- Configure logging

5-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (656) 820 2894 or +1 (818) 754 3700.

## Types of Name Servers

- Master and slave name servers provide *authoritative* information about a zone
  - Must answer iterative queries from servers; may answer recursive queries for clients
  - Slaves get updated database from master through zone transfers
- Caching-only name servers are not authoritative for any zone
  - Answer recursive queries for clients

5-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

A quick review of DNS terminology:

- A *zone* is the part of the DNS namespace for which a name server is responsible. It usually consists of a domain and may include one or more subdomains of that domain.
- A *master* name server contains the actual database files for a zone.
- A *slave* name server gets up to date zone data from a master server with a *zone transfer*.
- Both master and slave servers provide *authoritative* zone information to clients that query them. If those clients are other name servers those servers may cache that data locally and provide it in response to future queries as *non-authoritative* information.
- A query may be *recursive* or *iterative* (non-recursive). If a server responds to an iterative query, it provides the answer if it already knows it, otherwise it tells the client which server to ask for more information. If a server responds to a recursive query, it provides the client with the final answer even if the server has to look it up on other servers by issuing several iterative queries first.
- Master and slave servers normally must answer iterative queries from other servers; they may also answer recursive queries, although this is usually not desirable.
- A *caching-only* name server does not serve authoritative data for any zone. It is normally used to answer recursive queries from client workstations.

## Service Profile: BIND

- Type:**#System V-launched daemon**
- Packages:**bind, bind-utils, bind-chroot**
- Daemons:**named**
- Script:**#named**
- Ports:**#53/udp, 53/tcp**
- Configuration:**/etc/named.conf, /var/named/\***
- Related:**caching-nameserver, openssl**

5-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cctraining@redhat.com](mailto:cctraining@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 784 3700.

Service	Permissions	Normally runs as
<code>/usr/sbin/named</code>	755, root.root	user <i>named</i> , group <i>named</i>

named normally sends UDP queries using a random unprivileged port (for example, 32768/udp). This can be changed to an arbitrary port (for example, 53/udp) in the options block in named.conf:  
`>query-source address * port 53`

File	Permissions	Purpose
<code>/etc/named.conf</code>	644, root.root	Main configuration file, readable by named but only writable by root.
<code>/etc/rndc.key</code>	640, root.named	Contains symmetric TSIG key for use with rndc.
<code>/var/named/</code>	755, named.named	Readable only by group named. Config file directory; should not be world-writable!
<code>/var/named/named.ca</code>	644, named.named	Root name server hints file.

Zone files are normally stored somewhere in `/var/named` and should have permissions 644, named.named.

Supplemental Utility	Permissions	Purpose
<code>/usr/bin/{dig,host}</code>	755, root.root	DNS lookup utilities
<code>/usr/bin/nslookup</code>	755, root.root	Old DNS lookup utility
<code>/usr/bin/nsupdate</code>	755, root.root	DNS dynamic update utility
<code>/usr/sbin/rndc</code>	755, root.root	Remote name server control program; named may listen for this on port 953/tcp.
<code>/usr/sbin/dnssec-keygen</code>	755, root.root	Generates TSIG and DNSSEC keys.
<code>/usr/sbin/dnssec-makekeyset</code>	755, root.root	Package a DNSSEC keyset for parent signature.
<code>/usr/sbin/dnssec-signkey</code>	755, root.root	Tool to sign a DNSSEC keyset for child zone.
<code>/usr/sbin/dnssec-signzone</code>	755, root.root	Tool to sign a zone for DNSSEC.

## DNS Security

- DNS is a critical piece of infrastructure
  - Almost every site must run a nameserver
- Information stored in DNS is required by many network services
  - Spoofed DNS responses can lead to subversion of hostname-based security
  - An attack on DNS may be the first stage of an attack on something else

5-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2964 or +1 (919) 754 3700.

## Attacks on DNS

- Host enumeration
  - Determine existing hosts, possible targets
- Cache poisoning (name spoofing)
  - Bogus information fed to server cache through packet interception, client flooding, or control of a legitimate name server
- Attacking the parent zone
  - Cause NS delegation to be changed to hijack control of your zone

5-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trivialog@redhat.com> or phone toll-free (USA) +1 (800) 828 2994 or +1 (919) 754 3700.

An attacker may use DNS to perform host enumeration as preparation for a planned attack. They can query a name server to gather information about the zones it is responsible for; what hosts it has, what IP addresses are and are not in use on the network, and so on. An attacker may request zone transfers, or may attempt to walk through an IP address space one PTR record at a time. Restriction of zone transfers and the use of a split namespace are two defenses against host enumeration.

A much more serious type of attack is *cache poisoning*, or *name spoofing*. With cache poisoning, an attacker attempts to get the DNS server to load bogus information (preferably with large TTLs) into the name server's cache. There are several methods to accomplish this. With packet interception and client flooding, the attacker attempts to forge a response to one of the DNS queries and get it to the resolving name server before the real authoritative name server being queried responds. Another approach is to cause the target to query a legitimate but rogue name server controlled by the attacker. The attacker then injects bogus information in the response. These attacks may be used to attempt to bypass name-based access controls, or to redirect traffic to a rogue server posing as a trusted host. Defenses against these attacks include limiting acceptance of recursive queries and deployment of DNSSEC.

A third form of attack may actually target the zone's parent name server. If attackers can gain control of one of the ancestor zones, they can modify or replace the NS records delegating control of the zone and hijack it. Vulnerability to this attack depends on the security of the ancestor zones.

## Address Match Lists and *acl*

- Address match lists for access control
  - A semicolon-separated list of items
  - IP address, network prefix, cryptographic key, or named address match list
- *acl* names an address match list
  - acl "mylist" { 127/8; 192.168.0/24; };
  - Four defaults *acl*'s

5-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

Some access control directives take an address match list as an argument. An address match list is a semicolon separated list of items that may be:

- an IP address 192.168.0.254;
- a network prefix in CIDR notation (trailing zeros may be omitted) 172.16/12;
- a named TSIG cryptographic key key "server1-slave1.";
- a named address match list (*acl*) "internal";

Items may be negated with a leading ! as well. The list is checked in order, and the first match applies. Therefore, subsets should be listed first:

```
allow-query { ! 10.0.0.14; 10/8; };
```

A named address match list or *acl*, can be defined with the *acl* directive:

```
acl "internal" { 192.168.0/24; 192.168.1/24; };
```

It is a good idea to use *acl* to declare named address match lists for use in access control, as it allows changes to be done in one place, rather than in many directives throughout the named.conf file.

Four *acl*'s are defined by default:

any	matches all hosts
none	matches no hosts
localhost	matches the IP addresses of all interfaces on the system
localnets	matches any host on a network for which the host has an IP address

## TSIG: Transaction Signatures

- Authenticates DNS messages by signing them with a shared symmetric key
  - Secure access control mechanism
  - Guarantees message not altered in transit
- Requires clocks to be synchronized between communicating machines

5-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

TSIG (Transaction Signatures) is a method to secure access control and authentication of DNS messages between two machines by using symmetric key encryption. A shared secret key is generated, then installed on both stations that need to authenticate messages. One machine will use the key to "sign" DNS messages sent to the other machine. The other machine may use the key in appropriate address match lists to determine whether the DNS request sent will be honored.

Symmetric encryption is used instead of public key encryption for speed reasons. Therefore, the "signature" is not really a digital signature in a formal sense, because either the sending or the receiving machine could have signed the message. Each pair of machines that needs to communicate should use a different symmetric key for security reasons. If either machine is compromised, the key must be replaced with a new one.

TSIG messages have a time stamp associated with them that has an inception time and expiration time to determine for how long the messages are valid. This is intended to help prevent replay attacks. The clocks on both machines must be kept synchronized, preferably with a securely-configured NTP service.

The TSIG mechanism is defined in RFC2845.

## Installing TSIG Keys

- Generate a key with **dnssec-keygen**
- Use key directive to install and name the key on both machines
  - The name must be the same both places
- Only *named* on the two servers should know the key!
  - Configure key in a file with secure permissions and use include

5-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The dnssec-keygen command can be used to generate TSIG symmetric keys. The last argument to the command is the name of the key. By convention, it contains the host names of the communicating machines:

```
$ dnssec-keygen -a HMAC-MD5 -b 128 -n HOST server1-slave1.example.com.  
Kserver1-slave1.example.com.+157+39088
```

The key can now be extracted from the file

Kserver1-slave1.example.com.+157+39088.private that was just created in the current directory. The files content looks like this:

```
Private-key-format: v1.2  
Algorithm: 157 (HMAC_MD5)  
Key: 24XuKLUSdkQYjbDXw7Z48g==
```

Extract the key line. It needs to be configured in /etc/named.conf on both servers with a key statement. The key must have the same name on both servers:

```
key "server1-slave1.example.com." {  
algorithm hmac-md5;  
secret "24XuKLUSdkQYjbDXw7Z48g==";  
};
```

This key must be kept secret. Only user or group named should be able to read it. Since /etc/named.conf is set world-readable by default, one approach is to put the key statement in a separate key file instead of placing it directly in /etc/named.conf. Permissions of the key file are then set to 640, root.named, and is read into /etc/named.conf with an include statement in the latter file.

```
include "/etc/server1-slave1.example.com.key";
```

The configuration set up by the *caching-nameserver* RPM uses this approach to secure the TSIG key it randomly generates for the rndc control facility.

## Using TSIG Security

- Key to authenticate communication with server is declared in keys statement in a server block
- On server, access is controlled by using the named key in an appropriate directive's address match list
- Response from server is also signed with TSIG key

5-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

On a master and slave server communicates, authentication is necessary. This is defined in /etc/named.conf, in a keys statement in a server block for the machine being accessed:

```
server 192.168.0.254 {  
    keys { server1-slave1.example.com.; };  
};
```

On the server being accessed, named keys can be used as part of a directive's address match list to control access to the server. This is very commonly used to limit zone transfers from a master server to its slaves only. When feasible to use, it can be more secure than using IP-based access controls, as it is possible to spoof IP addresses.

```
allow-transfer { key "server1-slave1.example.com."; };
```

The response from the server is also signed with the shared TSIG key, so that the machine accessing the server can verify the origin of the response.

## Limitations of TSIG

- Symmetric key management can be difficult and does not scale
- Only provides next-hop security, not end-to-end security
- Does not provide data confidentiality
- Currently, cannot use to secure communication between stub resolver (workstation) and name server

5-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2894 or +1 (919) 754 3700.

TSIG is very useful to authenticate access between two DNS servers securely. However, it has some shortcomings, and there are some things that it does not secure.

One of the biggest difficulties with TSIG is that it uses shared symmetric keys for security. This means that if the key on either end station is compromised, it must be replaced. This also makes key distribution difficult as the key must not be compromised in transit. Typically, distribution is managed by manual methods, which may involve secure external utilities such as scp. In addition, each pair of machines that need to communicate must exchange a key in advance. This means that TSIG does not scale to the entire Internet DNS database. It is mainly useful for enterprise-level access control.

TSIG only provides “next-hop” security. It secures a client’s communication with its immediate server. It does not guarantee anything about the security of that server’s communication with other servers. This means that TSIG cannot guarantee the authenticity of data returned from a recursive query; it *can* guarantee authenticity of the results of iterative queries or zone transfers from the zone master. For end-to-end data integrity, DNSSEC must be implemented for the zone as well.

A design decision was made that DNS data is “public”, and therefore no effort is made to keep the contents of queries or responses confidential. They are sent clear text.

Another design goal of TSIG was that it should be lightweight enough to be built into “*stub resolvers*”, the code built into the C library on every Linux workstation that initiates DNS queries. This would allow the connection between a client workstation and its name server to be authenticated. However, as of this writing, the stub resolver for Linux does not support verification of queries with TSIG keys (nor does the stub resolver for any other OS the author knows of.) This means client/name server communication cannot be secured without running an appropriately configured caching name server on each workstation, which is not really reasonable. Hopefully, this situation will improve in the near future.

## Restricting Zone Transfers

- Zone transfers should be strictly limited
  - Master server should only allow slaves
  - Slaves should not allow zone transfers
- **allow-transfer**
  - Takes an address match list
  - Most secure to restrict by key, not IP
  - Can restrict globally or for a zone

5-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 784 3700.

Zone transfers are often used as a way for an attacker to quickly identify all the hosts in a particular zone. This can be used to gather information about the organization, select targets, or to pick out unused IP addresses so the attacker can appear to be coming from one and potentially bypass network prefix based access controls.

There is no reason why normal hosts need to perform zone transfers. In practice, it is a good idea to strictly limit which hosts are allowed to perform zone transfers. A master server should only allow its slaves to perform zone transfers. Slaves should not allow any machine to perform zone transfers.

Zone transfers are controlled by the **allow-transfer** directive, which takes as an argument an address match list and can be used in either the global options statement or in a zone statement.

```
allow-transfer { 192.168.1.1; 172.20.12.253; };
```

Many sites configure these restrictions using IP addresses, but as we have discussed, IP address based access control may be bypassed by a determined attacker. It is best to use TSIG key relationships to restrict zone transfers instead. This is one of the most common uses of TSIG.

```
allow-transfer { key "server1-slave1"; key "server1-slave2"; };
```

## Restricting Recursive Queries

- You should not allow recursive queries on authoritative name servers
  - Use separate caching name servers for client DNS lookups
  - Helps prevent cache poisoning and other types of name service attacks
  - recursion no;

5-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2894 or +1 (919) 754 3700.

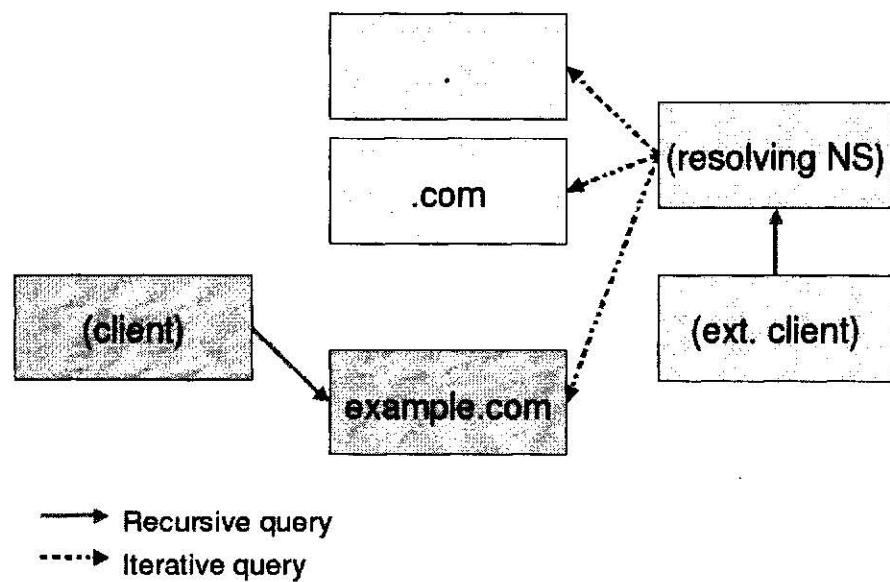
It is a good practice to separate the master and slave name servers that provide authoritative data for a zone; from resolving name servers that client workstations use to perform DNS queries. This is good from a performance point of view; handling a single recursive query for a client is often more work than answering several iterative queries for other servers. In addition, many cache poisoning attacks start with the attacker sending the targeted name server a recursive query for a resource record on a name server controlled by the attacker. By disabling recursive queries, this route of attack is eliminated.

Recursive queries are allowed by default. In the global options section of `/etc/named.conf`, the server can be blocked from performing recursive queries with the `recursion` statement:

```
options {  
    recursion no;  
};
```

With `recursion no` set, recursive queries received will be treated as if they were iterative queries, possibly redirecting the querying client to the root name servers for further information.

## Single Server Topology



5-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

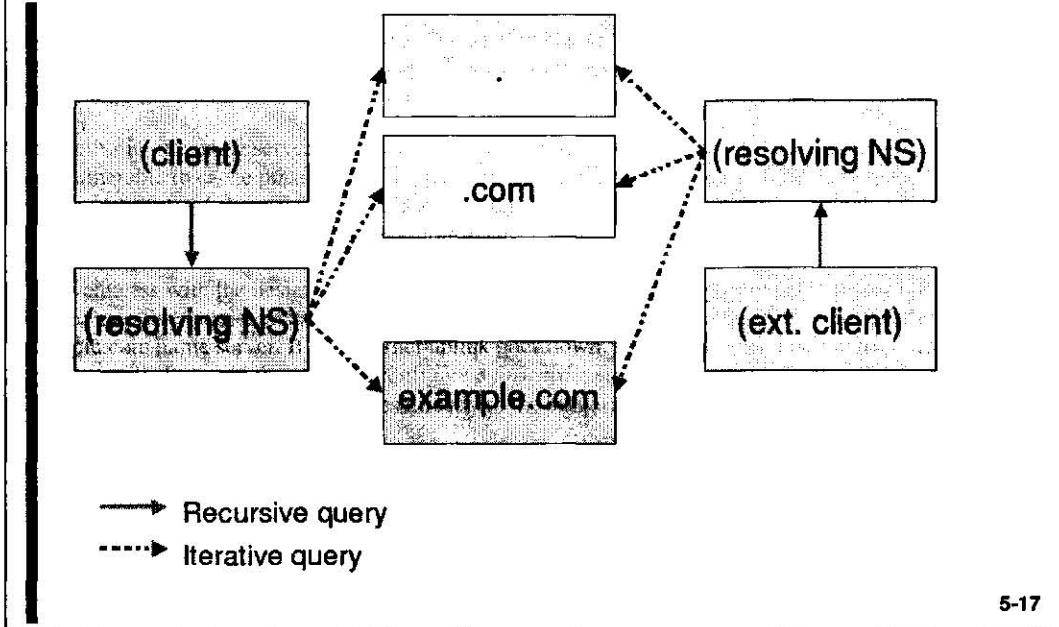
Here is an example of a site that is using the same name server to serve authoritative zone information and to handle recursive name lookups for its client workstations. Both (client) and (ext. client) are trying to resolve a host name in example.com; (client) is managed by example.com and (ext. client) is some external host on the Internet.

The master and slave name servers for example.com accept recursive queries from hosts in example.com, but only answer iterative queries from external clients. This could be configured using view statements in named.conf similar to those on the previous page. The example.com name servers may also use view statements so example.com hosts can see the full namespace but external hosts can only see a censored shadow namespace.

While we are calling this a “single server” topology, in reality we should have at least one slave for example.com as well, and it will authenticate its zone transfers from the master with a TSIG key. The master will not allow any other hosts to perform zone transfers. The slaves will not allow any hosts to perform zone transfers.

This configuration requires a minimal number of servers. However, allowing recursive queries from example.com hosts on the authoritative servers still leaves open the possibility of cache poisoning caused by a query issued by an example.com host. Externally-available name servers are more vulnerable. Should there be an intrusion, the consequences can be severe.

## Split Server Topology



5-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

Here is a more secure configuration than the one on the previous page. There is a separate caching-only name server the local clients use to resolve DNS queries. It only accepts queries (recursive or otherwise) from example.com.

The authoritative name servers for the example.com zone now only accept iterative queries, making them less vulnerable to cache poisoning. They may still provide different views of the namespace to external clients and to clients in example.com, however. Otherwise, the configuration of the name servers in this topology is similar to the one on the previous page.

One weakness with the design on the diagram above is that the caching-only name server initially needs to issue iterative queries to external name servers (the root and .com name servers) in order to find the example.com name server. If the Internet uplink goes down, this could cause internal name resolution to break as well. To avoid this, we can configure a *forward zone* on the example.com caching-only name server to send all example.com queries directly to the authoritative example.com name servers:

```
zone "example.com" {  
    type forward;  
    forwarders { 192.168.0.254; 192.168.1.253; };  
};
```

## Blocking All Queries

- Useful for internal-only name servers
  - Internal recursive server for client lookups
  - Private DNS server for "split namespace"
- **allow-query**
  - Similar to **allow-transfer**
  - Usually cannot block by key due to lack of client support; block by IP instead

5-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Sometimes, it is necessary to block all queries to a particular name server from certain hosts. For instance, it is a good idea to block external hosts from using the caching-only name server your internal client workstations use to resolve DNS queries. Alternatively, you may have a private "stealth server" containing different authoritative information about your zone than the public authoritative servers, which must not be accessed by external hosts.

In both cases, use **allow-query**, either in the global options section or on a zone-by-zone basis, to restrict access to the name server. Like **allow-transfer**, **allow-query** takes an address match list as an argument.

```
allow-query { 192.168.0/24; };
```

Unlike **allow-transfer**, **allow-query** requires use of IP-based access controls.

## Bogus Servers and Blackholes

- Some name servers respond with bad information
  - May be misconfigured or deliberately hostile
  - Use `bogus` in an appropriate `server` directive to prevent queries to a bad server
- Systems on the `blackhole` list will not be responded to or queried
  - If rejected by `allow-query`, server transmits refusal message

5-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

Name servers that provide bad information are encountered. It may be due to an inadvertent misconfiguration, or it may be intentional. A server block can be defined for that server and marked with `bogus` to prevent queries to a rogue name server in the future.

```
server 192.168.1.253 { bogus yes; };
```

The `blackhole` option prevents a server from querying or responding to a rogue name server or servers. `blackhole` takes an address match list as an argument:

```
options {  
blackhole { 192.168.1.253; };  
};
```

The difference between `blackhole` and `allow-query` with `bogus` is that a server matching the `blackhole` option is never responded to, while a host that does not match an `allow-query` option will be notified that the query was rejected.

## Views

- Name server normally provides one view of the DNS namespace
- Different hosts can be shown different views of a zone by the server
  - Most hosts see public DNS information
  - Some hosts see private DNS information; those hosts may be behind a firewall
  - A "split namespace"

5-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2904 or +1 (919) 754 3700.

Normally, a name server provides a single consistent view of its part of the DNS namespace. However, sometimes it is desirable that some hosts get different data for a zone than other hosts. For example, you may want to advertise information about hosts behind a firewall to internal hosts only, not the entire Internet. This is called a *split namespace*: internal hosts see the complete namespace while external hosts see a censored version commonly called the "shadow namespace". The shadow namespace typically contains only information for hosts that are accessible to the general Internet, such as web servers and mail servers.

Historically, the way split namespaces were set up involved various clever configurations involving two sets of authoritative name servers, one for internal use only and one for external use only. However, BIND 9 provides an easy way to configure multiple views of the same zone to different hosts with the `view` directive.

## Defining Views

- view directive defines a view
- match-clients defines which clients see which view
  - Order is important; first match applies
- Most things can be declared in a view
  - ACLs can be used but not defined in a view
  - If even one view is defined, all zones must be defined inside a view

5-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The view directive defines a new view of the name server. Each view directive is labeled with a name, which can be anything, but should describe the purpose of the view. The first view directive must appear sometime after the global options directive. Each view has a match-clients statement, which takes an address match list that defines for which hosts this view will be visible. Order of view directives is important. The view with the first match-clients statement that matches the querying host will be used. Therefore, the most specific views should come first, and a default view statement, last.

Inside the view directive, are zone directives relevant to that view. Different views may use different zone database files for each zone. This allows each view to provide different information for each zone. If any view directives is used, then every zone directive must be declared inside a view directive.

Most directives and options can be defined for a particular view. One exception is acl; which cannot be declared within a view, although a globally declared ACL can be referenced within a view. Many of the statements that normally are enclosed in the global options directive can be declared (without an options directive) inside a view as well. Options set in a view that conflict with a global option override the global option.

The options currently supported in view statement are documented in  
`/usr/share/doc/bind-version/misc/options`

## view Example

```
view "private" {
    match-clients { 192.168.0/24 };

    zone "example.com" {
        type master;
        file "example.com-private";
    };
}
```

5-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (819) 754 3700.

```
options {
directory "/var/named";
recursion no;
};

acl "internal" { 127/8; 192.168.0/24; };
view "internal;"{
    match-clients { "internal"; };
recursion yes;

zone "example.com" {
    type master;
    file "example.com-internal";
};
};

view "external" {
    match-clients { any; };

zone "example.com" {
    type master;
    file "example.com-external";
};
};
```

## version.bind

- BIND leaks version information through built-in CH TXT resource records
  - `version.bind` (BIND 8.2 and later)
  - `authors.bind` (BIND 9.1.0 and later)
- Best way to block is to use a custom view to suppress all CH records

5-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

BIND name servers, by default, advertise their version number to the world in a special resource record of class CHAOS and type TXT called `version.bind`. The host command may be used to look up this record:

```
$ host -c chaos -t txt version.bind nameserver
```

An attacker might make use of this information in order to take advantage of a version-specific bug or remote exploit affecting the name server. Many DNS administrators consider it good practice to suppress this information. This can be done with a special version declaration in the global options statement in `/etc/named.conf`. The version number can be changed to any arbitrary text string. This will hide your current version number, but any response may suggest to the attacker that you are running BIND 8.2 or later.

Best to block all queries to `version.bind` and `authors.bind`. This can be done by using views and by configuring a custom view to catch all requests for CHAOS records, for example:

```
view "kill-chaos" chaos {
    match-clients { any; };
    recursion no;

zone "." {
    type hint;
    file "/dev/null";
};
};
```

## **bind-chroot Package**

- Installs a chroot environment under /var/named/chroot
  - Moves existing config files into the chroot environment, replacing the original files with symlinks
  - Updates /etc/sysconfig/named with a **named** option:  
`ROOTDIR=/var/named/chroot`
  - Tips
    - Inspect /etc/sysconfig/named after installing bind-chroot
    - Run `ps -ef | grep named` after starting named to verify startup options

5-24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[crain@redhat.com](mailto:crain@redhat.com)> or phone toll-free (866) 426-2204 or +1 (951) 754-3700.

If you select the DNS Server package group, either during initial installation or from the system-config-packages GUI, the bind-*chroot* package will be installed. This package contains a tree of files which can be used as a chroot jail for the **named** program from the bind package.

When starting, **named** is chrooted to the directory specified in `ROOTDIR` prior to reading any configuration files. **named** cannot access any files above the `ROOTDIR`, so a file referenced in the configuration as `/some/path` must actually be located in `/var/named/chroot/some/path`.

**named** sees the file as /some/path. The system administrator sees the file as /var/named/chroot/some/path.

If you remove or comment out the `ROOTDIR` setting, then `named` runs without chrooting first, so all files and directories are relative to the root (`/`) directory.

#### Verify startup options:

```
[root@station1 ~]# ps -ef | grep named
```

```
named      13185      1  0 12:38 ?          00:00:00 /usr/sbin/named -u named
-t /var/named/chroot
```

Additional startup options can be passed to **named** via the **OPTIONS** variable in `/etc/sysconfig/named`. The **named(8)** man page contains information on other startup options.

## Monitoring with logging

- BIND has a very flexible and configurable logging system
- channel defines where log information should go
  - Can use custom channel or use one of four predefined channels
- category defines what should be logged
  - Many different categories exist

5-25

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

BIND comes with a very flexible logging system that can be readily customized in many ways. It does this through a mechanism of log channels and categories. A logging directive in `/etc/named.conf` will be used to configure this logging system.

A `channel` defines a target to which log messages may be directed. This might be a file, `syslog()` facility, or standard error, or the messages may be discarded.

All log messages are divided into one of fifteen `categories`. A category directive will be used to determine to which channels log messages should be directed. Messages in one category may be directed to multiple channels.

## channel

- channel defines target for logs
  - Can *syslog* to any facility or use a file
- Need to specify a severity as well
  - Similar to *syslog* severity; default is **info**
- Additional options for verbose output
  - Four channels predefined

5-26

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

All log output will be directed to one or more channels. Four channels are predefined, but it is possible to define more channels with the channel directive. Specify logging to a file can be done with the file statement followed by a file name, or to a syslog() facility with the syslog statement followed by a facility name.

Each channel must also have a severity statement; the channel will only record log messages of the given severity level or higher. The severity levels are identical to the ones used for syslog() messages, with the exception of severity dynamic, which accepts messages based on the named service's current debug level.

Three additional options may be specified. The print-severity option will also log the severity level of messages. The print-category option will also log the category of messages. The print-time option will also log the date and time of the message; since syslog() already records this information, it is mostly useful with file channels.

The four predefined channels are::

- channel "default\_syslog" { syslog daemon; severity info; };
- channel "default\_debug" { file "named.run"; severity dynamic; };
- channel "default\_stderr" { stderr; severity info; };
- channel "null" { null; };

The default\_debug channel is special, because it only produces output if the server is running at a non-zero debug level. The default\_stderr channel logs messages to the name server's standard error. If you specify the null channel for a category, all messages in that category will be discarded.

## category

- category statement associates a category with a channel for logging
- Fifteen categories to choose from
  - *default, general, client, config, dispatch, dnssec, lame-servers, network, notify, queries, resolver, security, update, xfer-in, xfer-out*
  - *default* has a category statement predefined

5-27

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The category statement takes one of several category names as a label, and the names of one or more channels as the target to log messages of that category to. If there are not any specific category statements, the default is:

To discard all messages in a particular category, you should specify the null channel.

Category	Purpose
default	Defines default channel for categories.
general	Catch-all category for unclassified messages.
client	Client request problems.
config	Configuration file problems.
dispatch	Dispatch of inbound packets to internal server modules.
dnssec	DNSSEC and TSIG.
lame-servers	Problems due to remote server misconfiguration.
network	Related to network operations.
notify	NOTIFY announcements (new protocol where master may notify slaves of zone changes).
queries	Query processing.
resolver	Recursive query processing.
security	Accepted or denied requests.
update	Dynamic updates.
xfer-in	Zone transfers received by the server.
xfer-out	Zone transfers sent by the server.

## logging Example

```
logging { channel locallog { syslog local4; severity info; }; category default { locallog; };
```

5-28

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ctraining@redhat.com](mailto:ctraining@redhat.com)> or phone toll-free (USA) +1 (800) 826 2994 or +1 (910) 754 3700.

The channel and category statements all must be included in a single logging directive for the name server. A longer example of a logging directive than the one above:

```
logging {
    channel locallog { syslog local4; severity warn; };
    channel xferfile { file "xfers.log"; severity info; };
    channel updates { file "update.log"; severity info; };
    category xfer-in { xferfile; };
    category xfer-out { xferfile; };
    category update { updates; };
    category default { locallog; default_debug; };
};
```

## Dynamic Update Security

- Dynamic updates are dangerous
  - Authorized hosts can delete all zone data
- Best: TSIG and update-policy
  - Windows 2000 uses GSS-TSIG, not yet supported by BIND
- IP-based security and allow-update
- Separate zone for dynamic hosts (i.e., \*.dyn.example.com)

5-29

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

DNS dynamic updates, originally defined in RFC 2136, allow client machines to make changes to zone data. The operating system that most commonly uses dynamic updates is Microsoft Windows 2000, XP and 2003; clients may update their A and PTR records, and Domain Controllers may also update SRV records. Dynamic updates are a major change from the traditional way DNS is managed, where only the system administrator in control of the master zone database can add or delete records from DNS. Dynamic updates raise some important security issues surrounding control of the DNS namespace.

Potentially, hosts that are authorized to dynamically update may be able to delete or change data for all of the hosts in the zone. For this reason, BIND is configured to prohibit dynamic updates of zones by default. The safest way to configure the system is to use TSIG authentication and the BIND 9 update-policy directive. This is a fine-grained mechanism that can limit which keys can update which records in the zone. The best option is to use the DHCP server that comes with Red Hat Enterprise Linux, which supports TSIG updates -- look in the `dhcpcd.conf(5)` man page under "Dynamic DNS Update Security". Windows DHCP servers use a proprietary GSS-TSIG mechanism recently published in RFC 3645, but that is not yet supported by BIND.

As an alternative, one solution is to set up a separate zone for all hosts using dynamic updates. Then those hosts are allowed to update the zone based on their IP addresses by using the allow-update directive. This is less secure, and cannot limit which records particular hosts can update. Critical systems and those accessible to the public are placed in a different zone that only contains static DNS entries and does not allow dynamic updates, so their information cannot be overwritten by rogue hosts.

```
zone "dyn.example.com" {
    type master;
    file "dyn.example.com";
    allow-update { "dynhosts"; };
};
```

## Setting up DDNS

- Use **dnssec-keygen** to create keys
- Add this key to BIND and DHCP configuration file
- Make sure DHCP sends name, otherwise BIND will not update DNS

5-30

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <ctraining@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

In `/etc/dhcpd.conf` identification of server and ddns-method is done:

```
ddns-update-style interim;
```

This key must also be used in `/etc/named.conf`, these lines are identical:

```
key newkey {  
    algorithm hmac-md5;  
    secret "secret_md5_hash";  
};
```

Add a description of what key to use in what zone, remember reverse zone too!

```
zone example.com. {  
    primary 192.168.0.X;  
    key mykey;  
}
```

Finally specify the information for the machines which are allowed to update the server:

```
subnet 192.168.0.0 netmask 255.255.255.0 {  
  
    host station2 {  
        hardware ethernet 00:A0:CC:39:AF:B0;  
        fixed-address 192.168.0.2;  
    }  
}
```

Add similar setup for your DNS (see previous page)

## **End of Unit 5**

- Questions and Answers
- Summary
  - TSIG key configuration and use
  - Restriction of zone transfers and queries
  - Split namespaces and view
  - Running named in a chroot() jail
  - Logging with categories and channels

**5-31**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Lab 5

# BIND and DNS Security

---

<b>Goal:</b>	Set up a chrooted DNS name server and secure zone transfers with TSIG
<b>Estimated Duration:</b>	90 minutes
<b>Situation:</b>	In this lab you will partner with another student. One of you will set up a DNS master server for domainX.example.com, where the X is the station number of your master server. The other will configure a slave server for domainX.example.com, and the two of you will use template files as a guide to help you improve the security of that domain and its servers. (If you have more than two people in a working group, one of you should set up the master server and the rest should set up slave servers.)
<p>Throughout this lab, the host and domain names that you use will be based upon the IP address of your and your partner's machines. Any time the lab refers to a name that contains X, you should replace X with the station number of your master server (the last segment of its IP address). Likewise, any time the lab refers to a name that contains Y, you should replace it with the station number of your slave server. For example, if your master server's IP address is 192.168.0.3, you would replace stationX.domainX.example.com with station3.domain3.example.com.</p>	

## **Sequence 1: Improving named service security**

Instructions:

1. *Obtaining necessary files*

The bind, bind-utils and caching-nameserver packages from the distribution are required. Use **rpm -q** to determine whether or not these packages are installed. If not, install them as root by running:

```
# yum install bind bind-utils caching-nameserver
```

The instructor should provide you with information on how to access to a workstation in the cracker.org domain, so that you can test some of your configuration settings later in the lab.

2. *Master Server Setup*

On the master server, you will also need to install the rhs333-master-server RPM using yum as root:

```
# yum install rhs333-master-server
```

In `/etc/named.conf`, in the options block, add:

```
allow-transfer { 192.168.0.Y; };
```

where 192.168.0.Y is the IP address of the slave server.

In `/etc/named.conf`, replace all X with the address of the master server. In `/var/named/`, modify both the `db.192.168.0.XX` and the `domainXX.zone` files to replace X with the address of the master server. Finally, rename the two files such that X is replaced with the address of the master server.

### **3. Slave Server Setup**

On the slave server, you will need to install the rhs333-slave-server RPM using yum as root:

```
# yum install rhs333-slave-server
```

In /etc/named.conf, specify your master server for your slave zones. For each of your slave zones, find the lines:

```
zone "domainX.example.com" {
    type slave;
    file "slaves/domainX.zone";
    masters { 192.168.0.X; };
};

zone "X.0.168.192.IN-ADDR.ARPA" {
    type slave;
    file "slaves/db.192.168.0.X";
    masters { 192.168.0.X; };
};
```

and replace the X with the station number of your master server. Make sure to go into /var/named/ and change the database files, too!

### **4. Testing the configuration**

Both servers will need to open ports 53/udp and 53/tcp to allow access to BIND.

At this point, you should have working master and slave name servers for domainX.example.com. Start the name servers and test the configuration with a couple of dig commands:

```
[root@stationY]# service named restart
[root@stationY]# dig @localhost stationX.domainX.example.com
[root@stationY]# dig @localhost domainX.example.com axfr

[root@stationX]# dig @localhost domainX.example.com xfr
```

### **5. Changing configuration to be chrooted.**

In this section, we're going to improve the security of our master and slave name server by obscuring their version numbers and chrooting them into /var/named/chroot/.

Install the bind-chroot rpm on your master and slave server. This will set up your DNS to be chrooted. By default it will set the root directory for **named** to /var/named/chroot. The RPM should copy your files into appropriate places in /var/named/chroot and restart named automatically when installed with yum:

You are going to do a zone transfer on your slave. Make sure **named** has permission to write to this directory.

Make sure that `/var/named/chroot/var/named/slaves/` uses the correct SELinux type, which is `named_cache_t`.

On both, the master server and the slave server, open `/var/named/chroot/etc/named.conf` in a text editor. Edit the first few lines of the options block to appear as follows (leave any forwarders lines alone):

```
options {  
    version "No version for you!" ;
```

Save `named.conf` and restart **named**.

Make sure your chrooted name server is still working. While you are at it, check your name server's advertised version.

```
[root@stationY]# dig version.bind chaos txt @stationX.example.co  
m
```

## Sequence 2: Securing zone transfers with TSIG

**Scenario:** Now we need to start securing the communication between the master server and the slave server. In this sequence we are going to secure zone transfers on the servers using a shared TSIG key. In order for this sequence to function properly, your master and slave servers should have their clocks synchronized.

**Instructions:**

1. First, either you or your partner needs to generate the encryption key. (It is probably more convenient to generate it on the slave.)

```
[root@stationY]# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST \
stationY-stationY
KstationX-stationY.+157+50029
```

The output tells us the name of the file containing our new key: in this case `KstationX-stationY+157+50029.private`. The numbers are the DNSSEC algorithm (157 = HMACMD5), and the key's fingerprint (50029) which will probably be different on your machine.

2. From `KstationX-stationY+157+50029.private` we will extract the shared key:

```
[root@stationX]# cat KstationX-stationY+157+50029.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: cfepTknNE5C4cS0upOe6pQ==
```

The key here is `cfepTknNE5C4cS0upOe6pQ==`; this will be different in your file. Keep track of this file; we will need it for testing our configuration later.

3. On both the master and the slave name server, create a new file called `/var/named/chroot/etc/transfer.key` with the following contents:

```
key "stationX-stationY." {
    algorithm hmac-md5;
    secret "cfepTknNE5C4cS0upOe6pQ==";
};
```

4. Protect the contents of the `transfer.key` so only the name server can read it and only root can write to it:

```
# chown root.named /var/named/chroot/etc/transfer.key
# chmod 640 /var/named/chroot/etc/transfer.key
# ln -s /var/named/chroot/etc/transfer.key /etc/transfer.key
```

5. In `/var/named/chroot/etc/named.conf` on both the master and the slave server, add the following lines at the very top of the file.

```
include "/etc/transfer.key";
```

6. On the slave server, prohibit all zone transfers from anywhere. Add the line in bold to your options block so it appears as follows:

```
options {  
    allow-transfer { none; };  
    version "No version for you!";  
    forwarders { 192.168.0.254; };  
    forward only;  
};
```

7. Set up your slave server to authenticate itself to your master server using the key in /etc/transfer.key. Add the following lines to /var/named/chroot/etc/named.conf after the include line, where 192.168.0.X is the IP address of your master server:

```
server 192.168.0.X {  
    keys { stationX-stationY.; };  
};
```

8. On the master server, change the allow-transfer line to your options block so it appears as follows:

```
options {  
    allow-transfer { key stationX-stationY.; };  
    version "No version for you!";  
    forwarders { 192.168.0.254; };  
    forward only;  
};
```

9. Now restart both servers.

10. It's time to test our configuration. First, try to execute a zone transfer of domainX.example.com without using TSIG authentication by running the following command on the slave server:

```
[root@stationY] # dig @stationX.example.com domainX.example.com axfr
```

This should result in "Transfer failed." Now try it with our TSIG key, by typing (all on one line):

```
[root@stationX] # dig -y stationX-stationY.:cfepTknNE5C4cS0up0e6p Q=@stationX.example.com domainX.example.com axfr
```

where stationX-stationY. is your key name and your secret key is cfepTknNE5C4cS0up0e6pQ==. This should result in a successful zone transfer, including an server-generated TSIG record authenticating the zone transfer.

## Sequence 3: Setting up views

Instructions:

- Finally, we will set up different views for users in example.com and cracker.org on the master server. Move your /var/named/chroot/etc/named.conf to a backup file and create a new file (see below). Replace domainX.zone with the following:

```
acl "internal" { 127/8; 192.168.0/24; };
acl "cracker" { 192.168.1/24; };

options {
    directory "/var/named";
    recursion no;
};

view "internal" {
    match-clients { "internal"; };
    recursion yes;
    zone "domainX.example.com" {
        type master;
        file "domainX-internal";
    };
};

view "cracker" {
    match-clients { any; };
    zone "domainX.example.com" {
        type master;
        file "domainX-cracker";
    };
};
```

- Copy /var/named/chroot/var/named/domainX.zone to /var/named/chroot/var/named/domainX-internal and /var/named/chroot/var/named/domainX-cracker. Make sure both files have permissions 640, are owned by user root and group named, and have the SELinux type *named\_zone\_t*.

- Add an entry for a web server inside domainX-internal:

```
viewtestX IN A 192.168.0.X
```

Do not forget to increment the serial number in the SOA record! Save the file.

- Restart your servers.

- Test it from stationY as well as from cracker.org:

```
# host viewtestX.domainX.example.com stationX.example.com
```

## **Sequence 4: Challenge Projects**

Instructions:

1. Reconfigure your name server to reject connections from the cracker.org machine. First use `allow-query`, then use `blackhole`. Use a network sniffer like `wireshark` or `tcpdump` to observe the difference in how the rejections are handled.
2. Experiment with setting up your own logging channel and log some category of information to it. Try both syslog-based and file-based solutions.
3. Set up a zone on your server that allows dynamic updates, and try adding and removing resource records with the `nsupdate` tool. (Look at the `nsupdate(8)` man page for help.) How could this be a potential security problem?
4. Experiment with using keys to authenticate users for using views.

## Sequence 1 Solutions

4. Both servers will need to open ports *53/udp* and *53/tcp* to allow access to BIND:

```
# iptables -A RHS333 -p udp --dport 53 -j ACCEPT  
# iptables -A RHS333 -p tcp --dport 53 -j ACCEPT  
# service iptables save
```

5. Install the *bind-chroot* rpm on your master and slave server. This will set up your DNS to be chrooted. By default it will set the root directory for **named** to */var/named/chroot*. The RPM should copy your files into appropriate places in */var/named/chroot* and restart named automatically when installed with yum:

```
[root@stationX]# yum install bind-chroot
```

You are going to do a zone transfer on your slave. Make sure **named** has permission to write to this directory.

```
[root@stationY]# chown root.named *  
/var/named/chroot/var/named/slaves  
[root@stationY]# chmod 775 /var/named/chroot/var/named/slaves
```

Make sure that */var/named/chroot/var/named/slaves/* uses the correct SELinux type, which is *named\_cache\_t*.

```
[root@stationY]# chcon -t named_cache_t *  
/var/named/chroot/var/named/slaves/
```

Save *named.conf* and restart *named*.

```
[root@stationY]# service named restart
```

## **Unit 6**

### **Network Authentication: RPC,NIS and Kerberos**

**6-1**

For use only by a student enrolled in a Red Hat Training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Objectives

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

- The RPC protocol and portmap
- Protecting portmap
- NIS weaknesses
- Improving NIS security
- Secure authentication with Kerberos

6-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

## Vulnerabilities

- portmap exposes open ports to remote users
  - Dynamic port assignment complicates protection of services
- NIS transmits account data unencrypted on the network
- NIS servers have easily spoofed client authentication mechanisms!

6-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Resolutions

- Use TCP wrappers and packet filtering to protect *portmap* and RPC services
- Restrict hosts that have access to NIS
- Do not pass RPC data on external networks
- Use Kerberos with NIS
  - NIS provides account information
  - Kerberos provides secure authentication!

6-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

## Network-managed users

- Information about accounts may be stored on a remote server
- Two types of information must be provided
  - Authentication: a way to prove the identity of a user
  - Account Information: UID number, primary group, default shell, and so on

6-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[trainings@redhat.com](mailto:trainings@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (619) 754 3700.

Information about user and group accounts can be stored in local files like `/etc/password` on each workstation or server. However, it may be easier to keep account information synchronized among many computers by storing it centrally in a remote network server.

Two types of information must be provided for each account. Authentication information is used to prove the identity of a user. Account information provides basic information about an account to the NSS component of glibc: UID, group memberships, default shell, and so on..

## Account Management

- Local accounts use files in /etc to store both account information and authentication information
- Network accounts store account information in directory services (NIS, LDAP, DNS/Hesiod)
- Network accounts may store authentication information in directory services, or in a separate authentication-only service (Kerberos)

6-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2904 or +1 (919) 784 3700.

Local files contain two types of information. Authentication information is used to prove the identity of a user. In local files, this is the password hash associated with each account. Account information provides basic information about the account; the UID, group memberships, default shell, and so on.

Network services store account information in a directory service of some sort. Common services include NIS, LDAP, and DNS (for Hesiod). They may use the directory service for authentication as well, either by storing password hashes in the directory or by using some special method to test authentication.

Alternatively, the directory service may only be used to store account information. Authentication is then tested by using a separate authentication service like Kerberos.

## What is Sun RPC

- Remote Procedure Call protocol
- Session protocol used by NIS and NFS
  - Protocol transported by TCP or UDP
- RPC services have program numbers
  - Identifies particular RPC services
  - Matched to ports through portmap

6-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Sun RPC, also known as ONC RPC, is a remote procedure call protocol used to implement client/server communications mechanisms that allow applications to share data across a network.

RPC operates independently from the underlying transport protocol, typically TCP or UDP. RPC does not provide any mechanism for handling timeouts, retransmission requests, or management of duplicate transmissions. RPC depends on the application or transport protocol to handle these details. RPC message data is encoded into a data format called XDR, which is operating system independent.

Each RPC service has a program number that is used by remote hosts to identify and locate it, much like standard TCP and UDP services have well-known port numbers. The `/etc/rpc` file lists standard RPC program numbers for network services. When a RPC program starts, it opens random unused TCP or UDP ports and registers those ports and its RPC program number with the local portmap service. Clients that want to talk to a RPC program contact the server's portmap service first to find out what port or ports that RPC program is currently using.

The two most important and common Sun RPC services are NFS (the Network File System) and NIS (the Network Information Service).

## RPC Security Issues

- portmap exposes the list of available RPC services
  - \$ `rpcinfo -p [servername]`
- Even if blocked from using portmap, an attacker may directly scan for the RPC service ports with nmap
- Dynamic port assignment makes use of firewalls hard!

6-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

By design, portmap exposes the list of all available RPC services to remote systems. To ask portmap which RPC services are running on which ports on a particular server, run the command:

```
$ rpcinfo -p [servername]
```

Portmap can be protected with TCP wrappers, or firewall rules restricting access to ports 111/tcp and 111/udp. However, this does not block an attacker from directly scanning for the RPC service port and connecting to it:

```
$ nmap -sR -p 1-65535 [servername] (RPC scan on all TCP ports)
```

```
$ nmap -sR -sU -p 1-65535 [servername] (RPC scan on all UDP ports)
```

Non-RPC services may also be reported. The UDP scan may take a long time to run; targeting a typical Linux server, no more than 20 ports can be scanned per second.

It can be hard to configure a firewall to pass traffic for a RPC service because port numbers are assigned dynamically. The two main approaches are to modify the firewall as RPC services register with portmap, or to force the RPC services to always use a fixed port number.

## NIS Overview

- Simple directory service for system and account information
- All NIS servers and clients are members of a named NIS domain
  - Single master server, multiple slave servers
- Minimal network security
- Support for NIS version 1 and 2

6-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

The Network Information Service, or NIS, is one popular network service that can be used to manage system and account information on multiple systems from a central server. NIS uses a single master server and optionally one or more slave servers, each running `ypserv`, to share information with NIS clients running `ypbind`. The NIS protocol is based on Sun RPC, and therefore clients and servers must also run a local service called `portmap`, which helps remote systems contact the local `ypserv` or `ypbind` program. Clients and servers that are bound to each other are normally members of the same NIS domain, identified by an arbitrary string selected by the system administrator.

NIS servers are typically used to synchronize account information. They can share the contents of

`/etc/passwd`, `/etc/shadow`, and `/etc/group` files by converting them into NIS maps. Each NIS map consists of a set of key/value pairs. For instance, one typical NIS map used is `passwdbyname`, where the key is an account name and the value is the matching line of user information for that account in `/etc/passwd` format.

The network connection between servers and clients is not encrypted, and there are no integrity or secrecy guarantees on NIS information. Passwords are sent as hashes, but nevertheless security of NIS information is minimal. NIS is still widely used in low-security environments because setup and maintenance is relatively simple.

Red Hat Enterprise Linux supports version 1 and 2 of the NIS protocol on both clients and servers. The client software also may work with NIS+ (NIS version 3) servers, but this functionality is minimally maintained.

## Service Profile: NIS

- Type:**#System V-managed services**
- Packages:**#ypserv**
- Daemons:**#ypserv, rpc.yppasswdd, rpc.ypxfrd**
- Scripts:**#ypserv, yppasswdd, ypxfrd**
- Ports:**#Dynamically assigned by portmap**
- Configuration:**#var/yp/\*, /etc/ypserv.conf (/etc/yp.conf for ypbind)**
- Related:**#portmap, ypbind, yp-tools**

6-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Weakness of RPC

- Server messages not authenticated to client
  - No message integrity checks
  - Easy to pose as server or forge data
- Client messages not authenticated to server
  - Easy to pose as valid client and capture database
- No secrecy of data transmitted
  - Sniffing can capture individual password hashes and other data

6-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

The biggest weakness of NIS is the lack of message authentication and integrity checking in the protocol. It is easy for an attacker to act as a man-in-the-middle and forge server responses to client systems. If an attacker can forge account information, they can:

- Change a the UID of an account to 0
- Change default shell from /etc/nologin to something useful
- Add or remove group memberships
- Add or remove entire users

and if the password hashes are stored in NIS for authentication, they can be changed to known values to allow login as any account with any password desired.

Anyone that knows the NIS domain name can bind to the server as a valid client and dump the passwd (or shadow) map by default. If password hashes are stored in the map, then off-line cracking of the password hashes can be attempted.

Even without dumping the NIS passwd map, all NIS messages are sent as cleartext. An attacker can capture individual lookups of passwd map entries and get valid hashes with a sniffer, then crack the hashes off-line. An attacker can also capture the NIS domain name and useful information about valid accounts in this way.

## Improving NIS Security

- Only allow NIS traffic on internal networks
- Block access to portmap and NIS services from external or untrusted hosts
  - TCP wrappers and iptables
  - Static ports
  - /var/yp/securenets

6-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email [cctraining@redhat.com](mailto:cctraining@redhat.com) or phone toll-free (USA) +1 (866) 826 2964 or +1 (916) 754 3700.

NIS traffic should be restricted to trusted internal networks to limit the exposure to attack. Do not allow it to cross insecure wireless or external networks! NIS traffic uses UDP, so it can not be protected using TLS, although IPsec associations would work.

Access to portmap and ypserv should be blocked for external hosts. IP-based access controls are available through TCP wrappers, and iptables can be used with some work. Static ports may be set in /etc/sysconfig/network on stationX:

```
YPSERV_ARGS="-p 841"
YPPASSWDD_ARGS="-p 842"
YPXFRD_ARGS="-p 843"
```

In addition, ypserv can be protected with a /var/yp/securenets file that contains lines listing pairs of permitted netmasks and networks. For example,

```
255.255.255.0 192.168.0.0
```

allows all hosts on 192.168.0.0/24 to bind if they know the NIS domain name. However, these controls are still vulnerable to IP address spoofing.

## Improving NIS Security

- Do not allow server discovery by broadcast
  - Simplifies ypserv impersonation
- Do not store password hashes in NIS
  - Use NIS only for account information
  - Use Kerberos for authentication

6-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

NIS clients can locate a server in a number of ways. The best way is to establish a static binding in `/etc/yp.conf` like this:

```
domain NISDOMAIN server hostname
```

where `NISDOMAIN` is the NIS domain name and `hostname` is a NIS server for that domain. An alternative to this method is to broadcast on the local net for a server and bind to the first one that answers. This is a bad idea. An attacker can then simply set up a rogue server on the local net and mount a denial of service attack on the real server. When the clients try to rebind, the only server that responds is the attacker's host, and now the attacker controls all NIS data.

Since NIS messages are not authenticated and have no integrity protection, an attacker can forge them. One attack is to replace the password hash in a NIS response for a valid account with some value chosen by the attacker. Now the attacker can log into that account with any password desired. To defend against this attack, do not store password hashes in NIS. Instead, use a more secure mechanism like Kerberos for authentication.

This does not stop the attacker from altering other information for valid accounts, but it may make it more difficult for the attacker to gain access to a valid account if they do not have that access already.

## Kerberos

- Secure network authentication system
  - Based around credentials called *tickets*
  - Tickets secured by secret key encryption
- Three participants
  - Key Distribution Center (KDC)
  - Application Servers (network services)
  - Client users

6-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Kerberos: the three-headed guardian*

Kerberos is a secure network authentication system based on shared secrets and symmetric encryption. Instead of transmitting passwords across the network, passwords are used as symmetric keys to encrypt and decrypt *tickets*, which may be used to verify the identity of a user or network service.

Tickets are issued by a central key server called a Key Distribution Center, or KDC. The local *realm* consists of all hosts that use the same KDC to get tickets, and is analogous to a NIS domain or LDAP suffix. By convention, the DNS domain of the site in all-caps is usually the Kerberos realm name. For example, the Kerberos realm for hosts in example.com might be EXAMPLE.COM.

Each host that runs a network service that supports Kerberos authentication is called an *application server*.

Each application server and each Kerberos user has an identity and a password associated with it. The KDC knows the passwords of all users and services in the realm.

Users initially log in using a password, and are issued a special ticket by the KDC. That ticket can be used to authenticate the user to additional Kerberos-aware hosts and services without forcing the user to type the account password a second time, enabling “single sign-on” capabilities.

# Principals

- A principal identifies each participant in a Kerberos authentication
  - Users and network services
  - Identified by *primary*, *instance*, and *realm*
- Each principal has a password
  - Passwords are used as encryption keys
  - Users memorize passwords
  - Services store passwords in a keytab file
  - KDC knows all passwords

6-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## *Primary, instance, and realm*

A Kerberos *principal* identifies the participants in a Kerberos authentication. Both client users and network services are represented by principals. Every principal has an associated password, and all principals and passwords are stored on the central KDC. Principal names have the form

`primary/instance@REALM`

The instance is optional, and if the realm is not provided it defaults to the one defined in `/etc/krb5.conf`.

A Kerberos client user has their username as the primary, and the instance is usually not defined. Instances are used to provide special roles, such as administrative access.

`elvis@EXAMPLE.COM`  
`elvis/admin@EXAMPLE.COM`

These two principals may be assigned to the same user, but from the point of view of Kerberos have separate passwords. The `elvis` principal may be useful for normal authentication, but authentication as the `elvis/admin` principal might be required to perform system maintenance on the Kerberos database. Users normally memorize their passwords.

A Kerberos application server must also have a principal in the database. The primary usually indicates the type of service, and the instance identifies which host the principal belongs to. For example

`ftp/station1.example.com@EXAMPLE.COM`

might belong to the FTP server on `station1.example.com`. Principals with a primary of `host` are quite common; `host` is used for services that provide general shell access (like `sshd` and `rlogind`). Services store their passwords on the application server in a local keytab file so that they may start up without human intervention.

## Initial Authentication

- User enters username and password
- Login program sends request for a TGT for that principal to the KDC
- KDC sends the login program a TGT encrypted using the user's password
- If the login program can decrypt the TGT with the password provided by the user, the user is authenticated

6-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

### *Initial Kerberos authentication*

On initial login, the user provides his or her username and Kerberos password to the login program. The program first converts the username to a principal name. It then sends a request to the KDC's "authentication service" for a ticket-granting ticket (TGT) for that principal.

The KDC generates a secret session key that will be used as the ticket-granting ticket. It keeps one copy, and encrypts a second copy using the password that belongs to the user's principal as the encryption key. The KDC sends the encrypted copy back to the login program.

The login program receives the response, and attempts to decrypt it using the password entered by the user as the decryption key. If this is successful, the user is successfully authenticated. The decrypted TGT is also saved in a local credentials cache for later use.

## Ticket Authentication

- Client sends request for a service ticket to the KDC's ticket granting service
- KDC sends client two identical copies
- One encrypted with the TGT
- One encrypted with the service password
- Client sends the network service
- Ticket encrypted with service's password
- A timestamp encrypted with the ticket

6-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### A Kerberos transaction

Once a Kerberos user has a current TGT, it can be used to authenticate that user to a network service that supports full Kerberos authentication without re-entering a password, by using ticket-based authentication.

The user's client first sends a request to the KDC's ticket granting service for a ticket that will authenticate it to a particular network service principal.

The KDC generates another secret session key that will be used as the ticket. One copy of the ticket is encrypted using the user's current TGT as the encryption key. A second copy is encrypted using the password of the network service's principal as the encryption key. Both copies are sent to the user's client.

The user's client uses its current TGT to decrypt the first copy of the ticket. It then creates an *authenticator* by encrypting a current timestamp with the decrypted ticket. The client then sends the second copy of the ticket (still encrypted with the network service's password) and the authenticator to the network service.

The network service uses the password stored in its keytab file to decrypt the second copy of the ticket. Since only the service and the KDC know the password, if this works the service knows the ticket came from the KDC. The service then uses the decrypted ticket to decrypt the authenticator. If this works, then the service knows the user's client decrypted its ticket, and therefore knows its own password on the same KDC. Since the user knows its own password, the user has successfully authenticated.

The timestamp in the authenticator is one defense against ticket replay attacks. Therefore, hosts participating in Kerberos need to keep their clocks synchronized, typically by using a service like NTP.

## Service Profile: Kerberos

- Type:**#System V-launched daemon**
- Packages:**krb5-server, krb5-libs**
- Services:**#krb5kdc, kadmind, kpropd**
- Scripts:**#krb5kdc, kadmin, kprop**
- Ports:**#88/udp, 464/udp, 749/tcp, 754/tcp**
- Configuration:**#/etc/krb5.conf, /var/kerberos/krb5kdc/\***
- Related:**ntp, krb5-workstation, pam\_krb5**

6-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Services	Permissions	Purposes
/usr/kerberos/sbin/krb5kdc	755, root.root	key distribution service
/usr/kerberos/sbin/kadmind	755, root.root	remote database admin service
/usr/kerberos/sbin/kpropd	755, root.root	receives database on slave servers
Utilities	Permissions	Purposes
/usr/kerberos/sbin/kdb5_util	755, root.root	database creation/destruction/backup
/usr/kerberos/sbin/kadmin.local	755, root.root	local database admin client
/usr/kerberos/sbin/kadmin	755, root.root	remote database admin client
/usr/kerberos/sbin/kprop	755, root.root	pushes database to slave servers
Configuration Files	Permissions	Purposes
/etc/krb5.conf	644, root.root	primary client and server configuration
/var/kerberos/krb5kdc/kdc.conf	644, root.root	server specific configuration
/var/kerberos/krb5kdc/kadm5.acl	644, root.root	database administration access control
/var/kerberos/krb5kdc/kpropd.acl	644, root.root	principals permitted to update slave
Databases	Permissions	Purposes
/var/kerberos/krb5kdc/principal	600, root.root	the Kerberos database

*Note:* Some configurations will also listen for connections on port 88/tcp

## /etc/krb5.conf

- Configuration settings for the realm
  - [libdefaults], [domain\_realm], and [realm] sections
  - Map DNS domain to realm name
- Client use file's settings to find KDC
  - Can encode in file or use DNS SRV
- Kerberized apps use [appdefaults]

6-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 426 2904 or +1 (919) 754 3700.

### *Basic realm configuration*

The /etc/krb5.conf file contains settings for basic configuration of the Kerberos realm. The most important sections of the file are [libdefaults], which controls defaults used by the Kerberos library; [realms], which describes realm-specific information; and [domain\_realm], which determines the Kerberos realm to use when authenticating to a particular remote host.

Some important options in [libdefaults]:

- default\_realm (The default Kerberos realm for the client.)
- dns\_lookup\_kdc (Whether to use DNS SRV records to find realm servers.)

Some important options in [realms]:

- admin\_server (The hostname:port of the kadmin server.)
- kdc#(The hostname:port of the KDC.)

If dns\_lookup\_kdc is set, the host finds the KDC for a particular realm by looking up DNS SRV records in the DNS domain matching the Kerberos realm name. For example:

```
_kerberos._udp          IN SRV 0 0 88 kdc-master.example.com.  
_kerberos._udp          IN SRV 0 0 88 kdc-slave.example.com.  
_kerberos-master._udp   IN SRV 0 0 88 kdc-master.example.com.  
_kerberos-adm._tcp      IN SRV 0 0 749 kdc-master.example.com.  
_kpasswd._udp           IN SRV 0 0 464 kdc-master.example.com.  
kdc-master.example.com. IN A 192.168.0.200  
kdc-slave.example.com.  IN A 192.168.0.201
```

## Installing a Master KDC

- Set up `/etc/krb5.conf`
  - Define basic settings for realm
- Set up `kdc.conf` and `kadm5.acl`
- Create database with `kdb5_util`
- Run `kadmin.local`
  - Set up principals for users and services
  - Create keytab file for `kadmind`
- Start `krb5kdc` and `kadmind`

6-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 784 3700.

### *KDC Configuration*

To set up the KDC for a Kerberos realm, first the basic `/etc/krb5.conf` file defining the realm must be configured. In addition, `/var/kerberos/krb5kdc/kdc.conf` controls some KDC-specific settings and `/var/kerberos/krb5kdc/kadm5.acl` controls `kadmin` access. These files will be discussed in more detail over the next few slides.

Then, the initial Kerberos database needs to be created with

```
$ kdb5_util create -r REALM -s
```

The service will prompt for a master password for the realm, which should be saved. This password is needed by all KDCs to decrypt the contents of the realm password database. The `-s` option causes the master server to cache this password in a *stash file* on disk so that it does not need to be entered when the `krb5kdc` service is started up at boot.

Then `kadmin.local` can be run on the KDC host and set up other principals for users and network services. One thing that is needed in `kadmin.local` is to extract passwords for the principals `kadmin/admin@REALM` and `kadmin/changepw@REALM` to a keytab file for `kadmind`:

```
kadmin.local: ktadd -k kadm5.keytab kadmin/admin kadmin/changepw
```

The default `kdc.conf` file requires that this `kadm5.keytab` file be copied to `/var/kerberos/krb5kdc` on the KDC. At this point the KDC services can be started:

```
$ service krb5kdc start; chkconfig krb5kdc on  
$ service kadmin start; chkconfig kadmin on
```

## Kerberos and DNS

- In addition to `dns_lookup_kdc`, can replace `[default_realms]` with `dns_lookup_realm`
  - Uses TXT record to determine realm
- DNS is normally not secure from spoofing
  - Either option may make attacks simpler
  - Use `/etc/krb5.conf` options instead

6-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Dangers of DNS service location*

In addition to using `dns_lookup_kdc`, the `[libdefaults]` option `dns_lookup_realm` may be turned on to replace the `[default_realms]` section. In this case, a lookup of TXT records for a DNS domain or host to determine its Kerberos realm:

```
_kerberos.example.com. IN TXT "EXAMPLE.COM"
```

The advantage of these options is that they simplify configuration management of the realm. The disadvantage is that DNS lookups may usually be spoofed. Therefore, using them may make attacks easier to implement.

It may be more secure to manually specify the information in `/etc/krb5.conf` on each station and distribute updates securely as needed:

```
[realms]
EXAMPLE.COM = {
    kdc = 192.168.0.200:88
    admin_server = 192.168.0.200:749
}

[default_realms]
.example.com = EXAMPLE.COM
```

## kdc.conf

- Configuration settings for the KDC
  - Stored in /var/kerberos/krb5kdc
- Default settings in [kdcdefaults]
- [realms] contains KDC parameters
  - Divided into realm-specific subsections
  - Supported principal encryption types
  - ACL and keytab file locations

6-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 828 2994 or +1 (919) 754 3700.

### *Configuration of the KDC*

In /etc/krb5.conf, the profile relation in the [kdc] section specifies the location of the KDC configuration file:

```
[kdc]
profile = /var/kerberos/krb5kdc/kdc.conf
```

This file has two main sections, [kdcdefaults] for default KDC settings, and [realms] for realm-specific KDC settings. In the latest versions of MIT Kerberos shipped with Red Hat Enterprise Linux 5, setting kdc\_tcp\_ports = 88 in the [kdcdefaults] section allows krb5kdc to respond to requests using the TCP protocol as well as the standard UDP protocol.

The [realms] section is divided up into per-realm sub-relations. In each sub-relation, master\_key\_type specifies the encryption algorithm used for the Kerberos database master encryption key. The supported\_enctypes option specifies the encryption-salt methods used to encrypt tickets.

In the original version of Kerberos shipped in Red Hat Enterprise Linux 3, only *des-cbc-crc* and *des3-hmac-sha1* were supported. The latest updates of the Kerberos packages in Red Hat Enterprise Linux 3 and 4 also support arcfour-hmac, which is used by Microsoft KDCs. Avoid using *des-cbc-crc* when possible; while mandated by the Kerberos standard, DES is prone to brute-force attacks on modern hardware, and *CRC-32* is not a cryptographically secure hash algorithm. The most common salt method needed is normal, although others are implemented as well.

## kadm5.acl

- Specifies what access a principal has to the database through kadmin
  - Stored in /var/kerberos/krb5kdc
- File format
  - Principal, permissions, target principal
  - Name, instance or realm may be wildcarded
  - Lower-case permissions allow, upper-case permissions deny

6-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (819) 754 3700.

### *Remote database administration*

The kadmind service provides remote administration access to the Kerberos database. The `kadm5.acl` file controls which administrative privileges are available to which principals. The file contains a series of entries, one per line, consisting of the principal being used, the access permissions allowed, and optionally a principal that is the recipient of the operation. The standard access permissions are:

- a/A#Allow/deny addition of principals or policies
- d/D#Allow/deny deletion of principals or policies
- m/M#Allow/deny modification of principals or policies
- c/C#Allow/deny password changes for principals
- i/I#Allow/deny database inquiries /para>
- l/L#Allow/deny listing all principals or policies
- \*#Equivalent to admcil

For example:

```
biff/admin@EXAMPLE.COM      ADMCIL      */admin@EXAMPLE.COM
*/admin@EXAMPLE.COM          *          *
```

The first line allows any principal with an instance of admin full access to the database. The second line restricts the biff/admin from operating on principals with instance of admin.

## Using kadmin

- Can manage Kerberos principals
  - addprinc, delprinc, modprinc, cpw
- Can set and manage password policies
  - Password aging, minimum length, history
  - addpol, delpol, modpol, getpol, listpols
- Can extract principals into keytab file
  - ktadd, randomizes password

6-24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (808) 626 2904 or +1 (919) 754 3700.

### *Managing Kerberos principals*

The kadmin and kadmin.local tools are the main ways Kerberos principals are managed. Users authenticate with a principal to kadmin; kadmin.local is run on the master KDC and does not use Kerberos authentication. Administrative users may add, delete, or modify principals, or change the stored password, depending on the settings in the ACL file. Principals can be set to expire on a particular day, and a flag may be set to force password changes.

Standard password expiration policies can also be set to control maximum and minimum lifetimes of passwords and password reuse. Each policy has a name; if no policy is specified when a principal is created and a policy named "default" exists, it is used. To create a policy in kadmin:

```
$ kadmin: addpol -maxlife 90days -minlife 3days -history 5 default
```

To set a principal named biff/admin to use the default policy:

```
$ kadmin: modprinc -policy default biff/admin
```

The **ktadd** command in kadmin is used to extract keys stored in the Kerberos database into a service keytab file, which may be specified with the -k option. When a key is extracted by ktadd, the password is first randomized as a security precaution.

## Installing Application Servers

- Set up `/etc/krb5.conf`
- Run `kadmin`
  - Create service principal
    - `addprinc host/www.x.org@X.ORG`
  - Extract service principal to keytab file
    - `ktadd -k krb5.keytab host/www.x.org`
- Start Kerberized network service

6-25

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### Kerberos-enabled network services

Once the KDC is configured, setting up Kerberos application servers is relatively straightforward. The network service needs to be installed on the system, and the `/etc/krb5.conf` file for the realm needs to be set up. An appropriate new principal for the network service needs to be added to the Kerberos database with `kadmin`. The password for that principal needs to be extracted into a keytab file and securely copied to the application server. The default keytab file location is usually `/etc/krb5.keytab`.

The `krb5-workstation` package includes a number of Kerberos-enabled services executable by `xinetd`:

daemon	port	principal	client
config			
eklogin	klogind tcp	host/* tcp	/usr/kerberos/bin/ rlogin
kshell	kshd	544/tcp	/usr/kerberos/bin/ rsh
gssftp	ftpd	21/tcp	/usr/kerberos/bin/ ftp
krb5-telnet	telnetd	23/tcp	/usr/kerberos/bin/ telnet

These services provide Kerberos authentication, and can provide encryption with the shared session key. Other services in the distribution may also support authentication with Kerberos tickets. These services include `sshd`, `slapd`, and `httpd`, among others.

## Kerberos Clients

- Set up `/etc/krb5.conf`
- Configure `pam_krb5.so` for Kerberos authentication, gets initial TGT
  - `kinit` to get a new TGT
  - `klist` lists available credentials
  - `kdestroy` deletes all credentials
- Tickets stored in `/tmp/krb5cc_UID`

6-26

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2894 or +1 (919) 784 3700.

### *Kerberos clients and initial Kerberos login*

To set up a Kerberos client, first the `/etc/krb5.conf` file must be configured for the realm. If the client is an application server (for instance, for `kshd`), the keytab file must be configured. For network services that do not support authentication by Kerberos ticket, the `pam_krb5.so` module can be set up in appropriate

`/etc/pam.d` files to allow authentication by username and Kerberos password.

The `pam_krb5.so` module can get some configuration settings from a `pam = {}` block in the `[appdefaults]` section of the `/etc/krb5.conf` file. Some useful settings include:

- `ticket_lifetime` (Default ticket lifetime in seconds)
- `renew_lifetime` (How long time to renew a ticket after it is issued, in seconds)
- `validate` (If true, uses local host/hostname principal from `/etc/krb5.keytab` to verify TGT came from the KDC)
- `forwardable` (true by default, gets TGT that can be forwarded to other hosts)
- `minimum_uid` (If a user with a UID lower than the one specified attempts authentication, the request will be ignored by the PAM module)

If Kerberos authentication is set up with the `authconfig` tool, `/etc/krb5.conf` and `/etc/pam.d/system-auth` are modified appropriately.

## Debugging Kerberized Services

- The password for the principal stored in the keytab file must match the one on the KDC
- **ktutil** can be used to view keytab files
  - Can show the versions of the principals stored in a keytab file
- **kvno** requests a ticket from the KDC
  - Can show which version of the principal requested is stored on the KDC

6-27

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

### Troubleshooting Kerberos authentication

In order for Kerberos authentication to work, the copy of a network service principal's password stored in the local keytab file and the copy stored on the KDC must match. Every time a principal's password is changed, its version number also changes. Tools can be used to check the version number of a password stored for a particular principal in the KDC and in the keytab file to verify that they match.

The **ktutil** utility can display which versions of which principals are stored in a keytab file. When run, **ktutil** opens up an interactive command-line prompt. At the prompt, the **rkt** command takes the name of a keytab file to read as its argument. Then the **list** command will list the principals stored in the file. Multiple principals with identical names may appear to support multiple encryption algorithms:

```
$ ktutil
ktutil: rkt /etc/openldap/ldap.keytab

ktutil: list

slot KVNO Principal
-----
1    3 ldap/station.example.com@EXAMPLE.COM
2    3 ldap/station.example.com@EXAMPLE.COM
ktutil: q
```

The **kvno** utility may be used by any user holding a valid TGT. It requests a ticket for a particular principal from the KDC, and displays the KDC's version number of the principal on standard output:

```
$ kvno ldap/station.example.com
ldap/station.example.com@EXAMPLE.COM: kvno = 3
```

Remember that every time a principal and its password are extracted to a keytab file by using **ktadd**, it randomizes the password for the service first!

## Kerberos Security

- Assumes secure hosts, insecure net
- KDC compromise = Realm compromise
  - Security of the KDC is *critical*
- If application server is compromised, service keys are compromised
- If client is compromised
  - Stolen password compromises user
  - Stolen TGT may be used to pose as user until it expires

6-28

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

### *Kerberos security issues*

The basic Kerberos security model assumes secure hosts and an insecure network. If host security is compromised, Kerberos security will be compromised to some extent. The most critical host is the KDC, which knows all passwords in the Kerberos realm. If the KDC is compromised, all passwords in the realm are compromised. Therefore, it is critical that the KDC is kept secure. The KDC should not perform any other tasks or provide any other network services for this reason.

If a Kerberos application server is compromised, then its service keytab files are also compromised. All service principals used by that server will need to be changed. If a client is compromised, then user credential caches are compromised. The attacker may pose as those users using issued tickets until they expire, which usually occurs after several hours. If the attacker can capture or guess a user password, then the account is compromised and the password must be changed.

Kerberos is dependent on other services, such as NIS or LDAP, to map user principals to user account information. Successful attacks on those supporting services may also violate host security.

## Preauthentication

- Ticket is encrypted with user password
- Normally can ask KDC for any user's TGT
  - Can not use without user's password
  - But can try to crack password used to encrypt captured ticket
- Require preauthentication if possible

6-29

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Kerberos password attacks and pre-authentication*

As part of normal initial authentication, a client can ask the KDC for any user principal's TGT. Without the user's password, the TGT can not be used. However, an attacker can try to guess the password used to encrypt the TGT. Even if the TGT has expired by the time the attacker guesses correctly and decrypts it, the attacker now knows the user's password and can use it to authenticate as that user.

To protect against this, the KDC can require pre-authentication. Before the KDC will give a client a TGT, the client must send it a request containing a current timestamp encrypted with the user's password. If the request decrypts, the KDC has authenticated the user on the client, and issues the TGT so that the client can authenticate the user as well. Otherwise the KDC returns an error message.

Preauthentication is enabled on a principal-by-principal basis. For example, to require pre-authentication for the biff/admin principal, in kadmin run

```
kadmin: modprinc +require_pcreauth biff/admin
```

To require it by default on all principals as they are created, in the block for the local realm in the [realms] section of the /var/kerberos/krb5kdc/kdc.conf on the KDC, add the directive

```
default_principal_flags = +pcreauth
```

This defense is not perfect. An attacker may still collect valid tickets by sniffing normal Kerberos network traffic.

## Ticket Validation

- Attacker can try posing as the KDC and a user to trick a client system into authenticating
  - Client requests TGT for principal
  - Fake KDC sends TGT for principal encrypted with password known to the attacker
- To defeat, login program must validate the KDC

6-30

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (819) 754 3700.

### *Mutual authentication*

An attacker can pose as the KDC and as a user at the same time to trick a Kerberos client to inappropriately authenticate them as that user. The attacker attempts initial Kerberos authentication as a principal using some password. The client sends a request for the TGT to the KDC. The attacker uses a machine posing as the TGT to respond with a forged ticket for that principal encrypted with the password the attacker used. The client blindly trusts this ticket came from the real KDC, and the attacker is allowed to falsely authenticate.

To defeat this, the client program must verify the TGT came from the real KDC. Before completing authentication, the client may use the TGT to ask for a ticket to its own host/hostname principal. Since only the client and the legitimate KDC know the password to that principal, the fake KDC is not able to forge this response and the false authentication can be detected and blocked.

Not all programs perform this validation of initial authentications. It is turned off for pam\_krb5 by default. To turn it on, set validate=true in the pam block in the [appdefaults] section of

/etc/krb5.conf and make sure that /etc/krb5.keytab contains the host principal and password for the system.

## Cross-Realm Trust

- Two Kerberos realms may trust each other
  - Users in one realm may authenticate to services in another
- Direct trusts share krbtgt principals
  - krbtgt/OTHER.COM@EXAMPLE.COM allows hosts in EXAMPLE.COM get tickets in OTHER.COM
- Indirect trusts: hierarchical, capaths

6-31

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Trusting other realms*

One Kerberos realm may trust users in another Kerberos realm and vice versa through cross-realm trusts.  
Users

in one realm can request tickets to services in another realm from their own KDC.

Direct trusts are implemented by having both realms share a common principal and password. Two principals are used, one for each direction of the trust relationship:

krbtgt/OTHER.COM@EXAMPLE.COM  
krbtgt/EXAMPLE.COM@OTHER.COM

The disadvantage of direct trusts is that every pair of realms using them must share principals and passwords.

Hierarchical trusts are based on subrealms. If two realms, EAST.EXAMPLE.COM and WEST.EXAMPLE.COM both have a direct trust relationship with EXAMPLE.COM, then they implicitly trust each other as well, using EXAMPLE.COM as an intermediary. Realms that are not related hierarchically may also set up indirect trusts, but the certificationpath for tickets must be explicitly specified in the [capaths] section of /etc/krb5.conf.

## Kerberos Encryption

- Kerberos session key can be used to encrypt data
  - Often is not used by default
- No “Perfect Forward Security”
  - If TGT is eventually cracked, can recover session key and decrypt data
- Some algorithms are weak (DES) or not widely supported yet (AES)

6-32

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (518) 754 3706.

### *Kerberos encryption issues*

The secret session key sent in a ticket used for authentication to a service may also be used to encrypt the connection after it has been made. However, this is often not enabled by default. For example, the Kerberos telnet client needs to specify that encryption should be turned on by adding the -x option.

Kerberos encryption has some disadvantages compared to TLS or SSH. The session key is sent over the network, encrypted by the TGT or service principal's password. An attacker can use a sniffer to capture and save the ticket containing the session key and the encrypted data stream. If they ever crack the encryption on the ticket, they can use the recovered session key to decrypt the captured encrypted data stream. TLS and SSH use an authenticated Diffie-Hellman protocol to agree on a secret key without transmitting it over the network, even in encrypted form, and permanently delete the secret key as soon as the communication ends. This is meant to provide “perfect forward security” of the encrypted data stream.

Another issue with Kerberos encryption is that some of the encryption modes available are not secure. The standards require des-cbc-crc be supported; but DES can be brute-force cracked, and CRC-32 is not a cryptographically secure MAC algorithm. Also, some newer, secure encryption modes optionally supported by MIT Kerberos (like AES) are not yet widely supported by other Kerberos implementations. This can cause interoperability problems if the newer modes are turned on.

## **End of Unit 6**

- Questions and Answers
- Summary
  - RPC and portmap
  - NIS security issues
  - Kerberos

**6-33**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

## Lab 6

# Network Authentication: RPC, NIS, and Kerberos

---

<b>Goal:</b>	Improve security of centralized network authentication
<b>Estimated Duration:</b>	90 minutes
<b>System Setup:</b>	Access to a cracker.org workstation for testing.
<b>Situation:</b>	In this lab you will work with a partner to set up a NIS domain and Kerberos realm. Your NIS server will provide centralized information about user accounts, and the Kerberos KDC will provide centralized information about account information.  Only one partner will run the NIS server, and only one partner will run the KDC. The instructions for this lab assume that both services are running on the same machine, stationX.example.com. The other partner's machine, stationY.example.com, will act as a test client.

## Sequence 1: Configuring the NIS server

Instructions:

1. Install the **ypserv** rpm on your server, stationX.example.com.
2. The **portmap** service should be on by default. Verify this, and turn it on if necessary.
3. One issue with **portmap** is that it advertises information about which Sun RPC services have ports open on your system. Run **rpcinfo -p localhost** and you should see the ports in use by the status RPC service (from **rpc.statd**).
4. Open the local firewall to allow **portmap** traffic through.

The problem with this setting is that now any host on the Internet can use **rpcinfo** to quickly see which Sun RPC services are running. From the cracker.org host, run

```
# rpcinfo -p stationX.example.com
```

and see what the results are. We will deal with this later in the lab. For now, leave this setting alone.

5. First, we need to allow traffic through the local firewall to the NIS server ports. By default, **ypserv** starts up on a random privileged port. We need to change **ypserv** so that it always uses the same privileged port. Add the following line to **/etc/sysconfig/network**:

```
YPSERV_ARGS="-p 808"
```

(We will not be running **ypxfrd**, but if we were there would also need to be a similar line for **YPXFRD\_ARGS**.) In addition, set your NIS domain up now. Some administrators believe that a random-looking string should be used for the NIS domain name, for added obscurity. We will use the hostname of your station instead:

```
NISDOMAIN=STATIONX.EXAMPLE.COM
```

6. Set your NIS domain name manually, as root, with a shell command.
7. Allow traffic to port 808 to pass through your firewall.
8. Start the NIS server.
9. As root, initialize the NIS maps:
10. From the cracker.org system, run **rpcinfo -p stationX.example.com** to confirm that several new services are running on your NIS server.
11. Now it is time for cracker.org to do some spying. Log in on the cracker.org machine as root. Dump the **passwd** map from your NIS domain:  
.

## Sequence 2: Improving NIS security

### Instructions:

1. Restrict **ypserv** so that it only provides NIS information to localhost and the example.com subnet. On the NIS server, create a **/var/yp/securenets** file containing the lines:

```
255.0.0.0 127.0.0.0  
255.255.255.0 192.168.0.0
```

2. We could use TCP wrappers to protect **portmap**. Instead, we will adjust the local firewall:

```
# iptables -D RHS333 -p tcp --dport 111 -j ACCEPT  
# iptables -D RHS333 -p udp --dport 111 -j ACCEPT  
# iptables -A RHS333 -p tcp --dport 111 -s 192.168.0.0/24 -j ✓  
    ACCEPT  
# iptables -A RHS333 -p udp --dport 111 -s 192.168.0.0/24 -j ✓  
    ACCEPT  
# service iptables save
```

3. Restart **ypserv**.

4. From cracker.org, try **rpcinfo -p stationX.example.com** again. This should not work.

5. From cracker.org, try dumping the **passwd** map again.

6. Set up your machine and your partner's machine to be clients of the NIS domain. Run **system-config-authentication**, select the **Enable NIS Support** check box, select **Configure NIS**, then fill in the NIS configuration screen:

NIS Domain: STATIONX.EXAMPLE.COM  
NIS Server: stationX.example.com

7. Restart **sshd** on your partner's workstation to make sure it checks the NIS maps.

8. On your NIS server, use **useradd** to create a new user and **passwd** to set a password for that user.

Then rebuild your NIS maps to make sure that the new user is added to NIS.

9. On your partner's station, create and populate a home directory for that user:

```
[root@stationY]# mkdir /home/userX  
[root@stationY]# cp -a /etc/skel/.[!.]* /home/userX  
[root@stationY]# chown -R userX:userX /home/userX/
```

10. Attempt to log into your partner's station as **userX**. Look in **/etc/passwd** on your partner's station. Now run **getent passwd userX**. **userX** should not be in the **passwd** file, but **getent** should return information.

Now it is time to look at NIS traffic with a sniffer. Make sure that you have **wireshark** and **wireshark-gnome** packages installed on your NIS server, then run the **wireshark** command.

This will open up the graphical interface for a network sniffer. The top pane will display a packet trace, the bottom pane the raw packets, and the middle pane is a dissector that will try to parse the raw packets.

Click on the icon farthest to the left on the icon bar to set up a capture. For the any device, click **Start**. This should open a new Capture window. Minimize the Capture window and click **Close** on the Capture Interface window.

On the stationY.example.com machine, log in as userX again. Wait a moment, then return to the sniffer's Capture window, and click **Stop**.

In the packet trace window on top, look in the Protocol column for YPSERV messages between the IP addresses of stationX and stationY. You should see a sequence of four packets; a "V2 DOMAIN Call" from stationY to stationX, a "V2 DOMAIN Reply" from stationX to stationY, a "V2 MATCH Call" from stationY to stationX, and finally "V2 MATCH Reply" from stationX to stationY. Select that last packet by clicking on it in the packet trace. Look at the raw packet in the bottom pane. You should see userX's account information in cleartext there.

11. In preparation for the next sequence, as **root** on the NIS server lock the userX account:

```
# usermod -p '!!' userX  
# cd /var/yp; make
```

## Sequence 3: Configuring the Kerberos KDC

**Scenario:** Now, we will stop using NIS to provide authentication information. Instead, NIS will provide account information and authentication will be handled by Kerberos. You will need to work with a partner in this section; one of you will be the KDC and the other a Kerberos application server. Choose one of your hosts to be the KDC (stationX.example.com below) and use STATIONX.EXAMPLE.COM as your Kerberos realm.

**Instructions:**

1. Make sure that you have the `krb5-server` and `krb5-workstation` packages installed on your KDC. Open the necessary firewall ports.
2. On the KDC, open `/etc/krb5.conf` in a text editor. Make the following changes: In `[libdefaults]`:

```
default_realm = STATIONX.EXAMPLE.COM
```

`[realms]` should read in its entirety:

```
[realms]
STATIONX.EXAMPLE.COM = {
    kdc = 192.168.0.X:88
    admin_server = 192.168.0.X:749
}
```

`[domain_realm]` should read:

```
[domain_realm]
stationX.example.com = STATIONX.EXAMPLE.COM
stationY.example.com = STATIONX.EXAMPLE.COM
```

This `[domain_realm]` section tells the machine that both stations are members of the STATIONX.EXAMPLE.COM Kerberos realm.

In `[appdefaults]`, add the following line inside the curly braces for the PAM block:

```
validate = true
```

This causes PAM to verify that it is talking to the true KDC by asking it for a ticket to the system's host principal. Save your work.

3. Initialize the Kerberos database on the KDC with a stash file so it can start up automatically. Run as root:

4. In `/var/kerberos/krb5kdc/kdc.conf`, edit the `[realms]` block to start:

```
[realms]
  STATIONX.EXAMPLE.COM = {
    master_key_type = des3-hmac-sha1
    default_principal_flags = +preauth
```

... and so on.

5. Open `/var/kerberos/krb5kdc/kadm5.acl` in an editor. Change the file so it reads:

```
* /admin@STATIONX.EXAMPLE.COM *
```

6. On the KDC, use **kadmin.local** to add some user and admin principals to the database.

7. Still in **kadmin.local**, list the principals in the database. Note that some were automatically created when you created the database. Also, examine the details of a single principal.

8. Create the `kadmind` keytab file:

9. A number of files we just created may have incorrect SELinux types. Fix them.

10. Start the KDC daemons.

## **Sequence 4: Configuring the Kerberos application server**

**Scenario:** In this sequence, you will set up both of your computers to perform user authentication using the new Kerberos KDC in place of the NIS service.

**Instructions:**

1. As **root** on your new KDC, stationX.example.com, run **kadmin.local** again. Create a host principal for your KDC.

2. Still in **kadmin.local** on the KDC, extract the new principal into a local keytab file readable only by root:

Ensure that /etc/krb5.keytab has the correct SELinux type.

Now both the KDC and services running as **root** on stationX.example.com have a copy of the password for that computer's host principal. Quit **kadmin.local**.

3. Log in as **root** to the computer which is *not* being used as the KDC, stationY.example.com. Make sure that the **krb5-workstation** package is installed. Copy /etc/krb5.conf from the KDC on stationX to stationY.

Ensure that /etc/krb5.conf has the SELinux type.

4. Still logged in as **root** on stationY.example.com, run **kadmin**, authenticating as the **root/admin** principal.

5. In **kadmin** on stationY, create a host principal for stationY.example.com and extract it to that host's /etc/krb5.keytab file.

Ensure that /etc/krb5.keytab has the correct SELinux type.

Now both the KDC and the services running as **root** on stationY.example.com have a copy of the password for the stationY.example.com host principal. Exit **kadmin**.

6. On both computers, run **system-config-authentication**. On the Authentication tab, select the **Enable Kerberos Support** check box, and keep the current Kerberos configuration. Leave the NIS settings enabled as well.

7. Run **getent passwd** to verify that the NIS user information is still visible for **userX**.

8. On stationY.example.com, switch to a text login prompt. Try logging in as **userX**, using the password your partner set earlier for the **userX** Kerberos principal. This should work. As that user, run **klist**. You should see that you have a Kerberos TGT.

9. As that user, **ssh** to stationX.example.com. You should be able to login without being prompted for a password! The **sshd** service has Kerberos ticket authentication turned on by default.

Now try to **ssh** back to stationY again. You should be prompted for a password. Why? Run **klist**. You should not have any tickets, since you did not enter a password or run **kinit** when you logged in, and **ssh** did not forward your credentials.

10. Log out of stationX. It is possible to configure **ssh** to forward your credentials. As **root**, edit **/etc/ssh/sshd\_config**, on both hosts, and add the following line to the bottom of the Host \* section:

```
GSSAPIDelegateCredentials yes
```

11. As **userX** on stationY, run **klist** to verify you still have a TGT. Try to **ssh** to stationX again. Run **klist**. You should have a TGT this time. Try to **ssh** back to stationY from stationX. This time it should work without prompting you for your Kerberos password.
12. The reason **GSSAPIDelegateCredentials** is off by default is so you do not accidentally forward a TGT to an insecure system. You can explicitly ask not to forward credentials by specifying the **-k** option. Try using **ssh** to log into stationX one more time, but this time add **-k**. Run **klist** to verify that your credentials were not forwarded.

13. DEBUG hints:

If you are experiencing problems, check that time on the machines is synchronized. For a quick test do the following: On stationY:

```
[root@stationY] # chkconfig krb5-telnet on
```

On stationX:

```
[root@stationX] # telnet -Fx1 userX stationY.example.com
```

## Sequence 1 Solutions

4. Open the local firewall to allow **portmap** traffic through:

```
# iptables -A RHS333 -p tcp --dport 111 -j ACCEPT  
# iptables -A RHS333 -p udp --dport 111 -j ACCEPT  
# service iptables save
```

6. Set your NIS domain name manually, as root, with a shell command:

```
# nisdomainname STATIONX.EXAMPLE.COM
```

7. Allow traffic to port 808 to pass through your firewall:

```
# iptables -A RHS333 -p udp --dport 808 -j ACCEPT  
# iptables -A RHS333 -p tcp --dport 808 -j ACCEPT  
# service iptables save
```

8. Start the NIS server:

```
# chkconfig ypserv on; service ypserv start
```

9. As root, initialize the NIS maps:

```
# /usr/lib/yp/ypinit -m
```

This command will ask you to list your NIS servers. Since you have no slave servers, and the program will enter the name of your master server for you, simply type *Ctrl-d* and then confirm that the list is correct.

11. Now it is time for cracker.org to do some spying. Log in on the cracker.org machine as root. Dump the **passwd** map from your NIS domain:

```
# ypcat -d STATIONX.EXAMPLE.COM -h stationX.example.com passwd
```

## Sequence 2 Solutions

3.     `# service ypserv restart`
5.     From cracker.org, try dumping the `passwd` map again:  
  
`# ypcat -d STATIONX.EXAMPLE.COM -h stationX.example.com passwd`  
This should not work.
7.     `{root@stationY}# service sshd restart`
8.     On your NIS server, use `useradd` to create a new user and `passwd` to set a password for that user.  
  
`# useradd userX; passwd userX`  
Then rebuild your NIS maps to make sure that the new user is added to NIS:  
  
`# cd /var/yp; make (or: make -C /var/yp)`

## Sequence 3 Solutions

1. Make sure that you have the `krb5-server` and `krb5-workstation` packages installed on your KDC. Open the necessary firewall ports.

```
[root@stationX]# iptables -A RHS333 -p udp --dport 88 -j ACCEPT
[root@stationX]# iptables -A RHS333 -p udp --dport 464 -j ACCEPT
[root@stationX]# iptables -A RHS333 -p tcp --dport 749 -j ACCEPT
[root@stationX]# service iptables save
```

The latest versions of the Kerberos KDC may also use port 88/tcp.

3. Initialize the Kerberos database on the KDC with a stash file so it can start up automatically. Run as root:

```
[root@stationX]# kdb5_util create -r STATIONX.EXAMPLE.COM -s
```

You will be prompted for the master database password.

6. On the KDC, use `kadmin.local` to add some user and admin principals to the database.

```
[root@stationX]# kadmin.local
Authenticating as principal root/admin@STATIONX.EXAMPLE.COM with
password.
kadmin.local: addprinc root/admin
WARNING: no policy specified for root/admin@STATIONX.EXAMPLE.COM ✓
;
defaulting to no policy
Enter password for principal "root/admin@STATIONX.EXAMPLE.COM":
Re-enter password for principal "root/admin@STATIONX.EXAMPLE.COM" ✓
:
Principal "root/admin@STATIONX.EXAMPLE.COM" created.
kadmin.local: addprinc userX
[...]
```

Remember the passwords you set on these principals.

7. Still in `kadmin.local`, list the principals in the database. Note that some were automatically created when you created the database. Also, examine the details of a single principal.

```
kadmin.local: listprincs
K/M@STATIONX.EXAMPLE.COM
kadmin/admin@STATIONX.EXAMPLE.COM
kadmin/changepw@STATIONX.EXAMPLE.COM
kadmin/history@STATIONX.EXAMPLE.COM
krbtgt/STATIONX.EXAMPLE.COM@STATIONX.EXAMPLE.COM
root/admin@STATIONX.EXAMPLE.COM
userX@STATIONX.EXAMPLE.COM
kadmin.local: getprinc userX
Principal: userX@STATIONX.EXAMPLE.COM
Expiration date: [never]
```

```
Last password change: Fri Jan 21 23:47:27 EST 2005
[...]
```

8. Create the kadmin.keytab file:

```
kadmin.local: ktadd -k /var/kerberos/krb5kdc/kadm5.keytab
    kadmin/admin
[...]
kadmin.local: ktadd -k /var/kerberos/krb5kdc/kadm5.keytab
    kadmin/changepw
[...]
```

Quit **kadmin.local**.

9. A number of files we just created may have incorrect SELinux types. Fix them.

```
[root@stationX]# restorecon -R -v /var/kerberos/krb5kdc/
[root@stationX]# restorecon -R -v /var/log/
```

10. Start the KDC daemons:

```
[root@stationX]# service krb5kdc start; chkconfig krb5kdc on
[root@stationX]# service kadmin start; chkconfig kadmin on
```

## Sequence 4 Solutions

1. As root on your new KDC, stationX.example.com, run **kadmin.local** again. Create a host principal for your KDC:

```
kadmin.local: addprinc -randkey host/stationX.example.com
```

2. Still in **kadmin.local** on the KDC, extract the new principal into a local keytab file readable only by root:

```
kadmin.local: ktadd -k /etc/krb5.keytab host/stationX.example.com
```

Ensure that /etc/krb5.keytab has the SELinux type `krb5_keytab_t`:

```
[root@stationX]# restorecon /etc/krb5.keytab
```

3. Log in as root to the computer which is *not* being used as the KDC, stationY.example.com. Make sure that the `krb5-workstation` package is installed. Copy `/etc/krb5.conf` from the KDC on stationX to stationY:

```
[root@stationY]# scp root@stationX.example.com:/etc/krb5.conf /etc/krb5.conf
```

Ensure that `/etc/krb5.conf` has the correct SELinux type:

```
[root@stationY]# chcon -t krb5_conf_t /etc/krb5.conf
```

4. Still logged in as root on stationY.example.com, run **kadmin**, authenticating as the root/admin principal.

```
[root@stationY]# kadmin -p root/admin
```

You will be prompted for the Kerberos password that you set for this principal when you created it in the previous sequence.

5. In **kadmin** on stationY, create a host principal for stationY.example.com and extract it to that host's `/etc/krb5.keytab` file.

```
kadmin: addprinc -randkey host/stationY.example.com
```

```
kadmin: ktadd -k /etc/krb5.keytab host/stationY.example.com
```

Ensure that `/etc/krb5.keytab` has the correct SELinux type.

```
[root@stationY]# restorecon /etc/krb5.keytab
```

## **Unit 7**

### **Network File System**

**7-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (918) 754 3700.

## **Objectives**

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

- Discuss weaknesses of traditional NFS protocols and services
- Improve security and robustness of NFS servers and clients
- Deploy NFS4 in production for improved security

**7-2**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Traditional NFS

- NFSv1 - Sun internal-use-only
- NFSv2
  - Synchronous communications only
  - Stateless
  - UDP only
- NFSv3 added:
  - Asynchronous communications
  - TCP support for stateful connections
  - Larger network reads and writes
  - Support for 64-bit file sizes

7-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2964 or +1 (819) 754 3700.

One of the original RPC services, NFS allows a client computer to mount a directory from a server and use the file system on that server as if were on a local disk.

The version 1 NFS server was an internal Sun release.

Version 2 is documented in RFC1094 and is still widely used.

Version 3 (RFC 1813) added some features, like support for 64-bit file sizes and offsets, TCP as an alternative transport protocol, and larger maximum sizes of read and write operations. From a security standpoint, the public versions are virtually identical.

Traditionally, NFS runs over UDP. NFSv3 introduced NFS over TCP. The Red Hat Enterprise Linux NFS client communicates using TCP by default. Use of UDP is discouraged but can be forced by the client with the “udp” mount option

## The Traditional NFS Protocol

- Transparent file access based on Remote Procedure Calls
  - Client connects to server's portmap for ports, versions, and transports
  - Client connects to server's rpc.mountd
  - Client is issued an initial file handle
  - Additional RPC commands sent to server's rpc.nfsd to read or change remote file system
  - File locking handled by rpc.lockd and rpc.statd
- Mounts are stateless when using udp

7-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

When an NFSv2 or NFSv3 client first mounts an export, it contacts rpc.mountd on the server, which authenticates the client based on host-level access controls. If authenticated, an initial file handle for the export is returned to the client. At this point the client has mounted the export. *Note:* this is not true authentication since there is no shared secret or other means of verifying the client is who it claims to be.

All further accesses are managed through RPC commands sent to the server's rpc.nfsd service, including requests to read or change files, or make changes to the file system (adding or deleting files or directories). File locking services are managed by a pair of additional services, rpc.lockd and rpc.statd.

The communication between client and server is stateless. A mounted file system remains mounted even if the server goes down; there is no sense of a connection being broken just because the server is unavailable. Of course, file activity cannot occur with a shut down server. But communication will continue when the server comes back up.

## Service Profile: NFS

- **Type:** System V-launched service
- **Packages:** nfs-utils
- **Daemons:** rpc.nfsd, rpc.mountd, rpc.statd, rpc.lockd
- **Scripts:** nfs, nfslock
- **Ports:** 2049/tcp, 2049/udp (nfs) others by portmap (111/{tcp,udp})
- **Configuration:** /etc/exports
- **Related:** portmap, quota

7-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Vulnerabilities of NFSv2/3

- Complex service runs as *root*
- Data transmitted as clear text
  - No data integrity or secrecy guarantees
- Host-based access control only; easily spoofed
- Dynamic port assignment complicates firewall setup
- Name mapping issues
  - The same UID/GID on server and client may map to different real users
  - Server trusts client for user identity
- Dependence on portmap
  - portmap exposes open ports to remote users

7-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

NFS suffers from a number of weaknesses in its design. It is a complex service that must run as root on the server. This means that there is a higher likelihood of bugs that may be exploited by a remote attacker to gain control of the server.

Authentication is weak, dependent primarily on host-based mechanisms. There is no concept of user-based authentication to a share. Data is transmitted as clear text, with no data integrity or secrecy guarantees. This may expose arbitrary data on the export to sniffing. It also makes it easy for an attacker to act as a “man-in-the-middle” and inject NFS commands to read or change critical files on the exported file system.

Internal ownership of files on the file system are stored as UID and GID numbers, not strings. The server trusts whatever UID and GID the client claims to be, so users may get access to privileged information if mappings are inconsistent between the server and client. Worse, a rogue client could claim to be a UID or GID without any way for the server to confirm the identity.

The various components of NFS are RPC services dependent on portmap, and except for nfsd have their network ports assigned randomly at startup. This makes it difficult to protect the services with packet filtering rules, as it is hard to predict on which port they will start up.

## Improving Security with Static Ports

- Enforce static ports at startup
  - Simplifies firewall configuration
  - Avoids collisions with other services
- Edit /etc/sysconfig/nfs:

```
LOCKD_TCP_PORT=888  
LOCKD_UDP_PORT=888  
MOUNTD_PORT=889  
STATD_PORT=12001
```

- nfsd uses TCP/UDP 2049 by default
- View active ports with **rpcinfo -p 192.168.0.X**

7-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2964 or +1 (919) 754 3700.

Typically, Sun RPC services select a random unused port on startup and register it with the portmapper. However, since the port is random, it can vary slightly from boot to boot, complicating efforts to protect the host with port-based firewalls. Also, it is possible (if rare) for the RPC service to grab a port that belongs to a well known service that has not yet started, causing problems later in the boot sequence. Therefore, it can be useful to force RPC services to start up and run on particular network ports.

The main nfsd service listens on two static ports, 2049/udp and 2049/tcp, by default.

The Red Hat Enterprise Linux scripts /etc/init.d/nfs\* read several configuration files before starting services. You can review the init scripts to determine which variables are available and which config files should hold your variable assignments.

*Which variables are available?* You can use less to review the files, or you can use the following grep command:

```
grep -oh '\$\{[^:upper:][^:digit:]\}\+\>' /etc/init.d/nfs* | sort -u
```

*Which file should contain the variable assignments?* In general, you should use the file that is sourced *last* in the init script. For nfs and nfslock, this means that you should define the variables in /etc/sysconfig/nfs, which does not exist by default.

*Why choose the ports in the examples above?* The choice is somewhat arbitrary as long as you avoid collisions with other services. Also, rpc.statd must bind to an ephemeral port (> 1024). A quick review of /etc/services implies that these ports are not used for services on our system. To be sure, run the following command to list all listening ports on your server:

```
netstat -tulpn
```

## Protecting Data Confidentiality in NFSv2/3

- Restrict access to trusted hosts
  - Use TCP wrappers to protect daemons
  - Do not allow NFS traffic to traverse insecure networks
  - Limit NFS exports by IP
- Do not export to localhost
  - Old versions of *portmap* forwarded mount requests
  - Attacker could appear to be mounting the share from *localhost*
  - Current *portmap* does not have this issue
- Do not export directories containing server configuration files

7-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

The following BASH script will identify the NFS daemons that are protected by TCP Wrappers, either statically by including code or dynamically by linking to libwrap.

```
#!/bin/bash

daemon_list="portmap rpc.mountd rpc.lockd rpc.statd"
for x in $daemon_list
do
    if ldd `which $x` | grep -q libwrap
    then
        echo "$x is libwrapped"
    elif strings `which $x` | grep -q hosts.allow
    then
        echo "$x is statically compiled with support for TCP Wrappers"
    else
        echo "$x is not protected"
    fi
done
```

It is considered bad practice for a NFS server to export its shares to itself. This was reinforced by a bug in old versions of *portmap*, which allowed *portmap* to proxy mounts for external hosts. When *portmap* forwarded the mount request, it was treated as if it came from *localhost* rather than from the host originating the request, bypassing access controls.

The current implementation of *portmap* that ships with Red Hat Enterprise Linux is no longer vulnerable to this attack. Nevertheless, it is still good practice not to export shares back to the server.

Due to the sensitivity of NFS network traffic and its lack of protection, do not allow NFS exports or mounts to occur that causes that traffic to cross untrusted networks. Likewise, it is a bad practice to export

system disks or directories containing sensitive configuration files used by the NFS server. Those files should be kept on local partitions private to the server. If the files must be distributed to other servers, it is better to use a secure copy protocol such as scp or rsync with SSH.

## Controlling Name Mappings

- Use centralized authentication when possible
  - Appropriate combinations of NIS, LDAP, and Kerberos
  - Prevent users from getting shell access to server
    - pam\_listfile
    - /etc/security/limits.conf
  - Consider options in /etc(exports
    - root\_squash maps uid/gid 0 to nfsnobody
    - all\_squash maps all uids and gids to nfsnobody

7-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### Centralized Authentication

Use centralized authentication to ensure consistency between the server and clients for UID-to-user and GID-to-group mappings. Traditionally this mechanism has been NIS, but LDAP using Red Hat Directory Server or OpenLDAP is also feasible. If these mappings are wrong, the wrong users may have permission to read or change files.

pam\_listfile may be the easiest way to prevent users from getting shell access to the server when using centralized authentication. One approach would involve the following steps:

- Create a NIS or LDAP group called *server-admins* with membership consisting of only those usernames that need shell access.
- Create /etc/logingroups.allow:  

```
echo "server-admins" > /etc/logingroups.allow
```
- Add a line to /etc/pam.d/system-auth:  

```
auth required pam_listfile.so onerr=fail item=group sense=allow file=/etc/logingroups.allow
```

Another approach would be to set the default maxlogins to zero in /etc/security/limits.conf. This approach would provide granularity, but might require more work in establishing allowed maxlogins for root and *server-admins*.

### Export Options

*root\_squash* is a default option when exporting filesystems via NFS and prevents a user on an untrusted or trojaned client from gaining root access on the NFS server. Filesystems with user data should always be exported with *root\_squash* enabled. This causes the root user on the client to be treated by NFS as UID 2^32, nfsnobody. This protects files owned by root on the exported directory from tampering.

Using *no\_root\_squash* may be necessary to support diskless workstations such as netboot. In that scenario, care should be taken to ensure that netboot filesystems are on a different filesystem than user or confidential system data. Preferably, the netboot filesystem should be placed on a separate server altogether since the client root user will also be treated as root on the NFS server.

*all\_squash* is *not* a default option because it maps *all* UIDs and GIDs to nfsnobody. This can be useful on publicly-accessible shares, such as filesystems that are exported read-only via both ftp and nfs. The default option is *no\_all\_squash*.

## Protecting Data Integrity in NFSv2/3

- NFS server A should not be a client of NFS server B
  - Avoid possible deadlock scenario at boot-time
- Consider options in /etc(exports
  - **ro** ensures data is read-only despite filesystem permissions
  - **sync** ensures data is written to disk with each nfs operation
    - Required for NFSv2
    - May have slower performance than **async**
  - **secure** ensures that mount requests originate from a privileged port
  - **fsid=n**
    - n is a 32-bit number
    - Allows failover of file handles and locks in clustered configurations
    - **fsid=0** has special meaning for NFS4

7-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <mailto:ceas@us.redhat.com> or phone toll-free (USA) +1 (866) 826 2864 or +1 (919) 754 3700.

When possible, it is good practice for the NFS server not to be a client of another NFS server. “Cross-mounting” can lead to problems after a power outage, when multiple NFS servers are attempting to come up at the same time. You can end up with several servers deadlocked partway through the boot process, each waiting for a hard-mounted export from another server to become available before they try to start their own NFS servers.

Whenever possible, NFS directories should be exported read-only. Never export read-write to the Internet; only export read-write to carefully selected and explicitly specified hosts under your control.

To limit disk corruption due to network problems or server crashes, you should export your NFS file systems with the sync option set. The NFSv2 protocol was designed assuming this option is set, but until recently under Linux the default was async, which is not safe. NFSv3 can handle asynchronous writes better, but it is still safer (if somewhat slower) to select sync.

The default secure option mandates that clients accessing the export must be communicating from a privileged port. This is intended to provide some assurance that an NFS export is being mounted by a superuser-controlled process, and not a user issuing random RPC commands. While this does not provide much additional security, it is still better than using the insecure export option, which turns this feature off.

An NFS server normally uses its own major/minor device numbers when creating file handles and locks for clients. This can cause problems with stale handles and locks in a failover environment. The server will create file handles and locks based on the fsid when specified in /etc(exports, thus enabling handles and locks to be consistent across physical machines and thereby alleviating the problem. Any 32-bit number is acceptable; however, **fsid=0** has special meaning under NFS4 and will be discussed later in this unit.

## Improving Security with NFS4

- **NFS4 is Kerberized NFS!**
  - Requires strong identity mapping via GSSAPI to Kerberos
  - Provides mutual authentication of hosts and services
  - Optionally provides Kerberized user authentication
  - Optionally provides integrity checking and data encryption
  - Safely supports special permissions and ACL's
- **NFS4 protocol is integrated within the kernel**
  - rpc.{mountd, lockd, statd} functionality is integrated into NFS4 protocol
  - Removes server and client dependence on portmap
  - Uses a single port: TCP 2049
  - Connections are always stateful
  - autofs4 kernel module supports automounter
- **Seamless exports: all exports appear as one psuedo-filesystem**

7-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 734 3700.

NFS4 is documented in RFC 3530 and is designed for the Internet. It addresses many of the concerns with security and robustness that are associated with previous versions of NFS. Kerberos authentication and encryption features are supported through the RPCSEC\_GSS mechanism. Mounting and locking services are consolidated into the core protocol.

The functionality of rpc.mountd, rpc.lockd and rpc.statd are built into the kernel for NFS4, so these are not started manually. Dependence on portmap is also removed, though portmap may still be necessary for other services, such as *ypserv*.

When starting the NFS server, it has ability to use v4, v3 and v2. When a client wants to connect using NFS4, it is necessary to use the file system nfs4, if file system is not mentioned, nfs is assumed (v2/v3). It uses tcp as default, provided client speaks v3. NFS over udp can be enforced with the option *udp*.

NFS4 has built in support for ACL; however, it is not the POSIX model, but based on Microsoft Windows NT.

These sites provide information on the development of NFS4:

<http://www.citi.umich.edu/projects/nfsv4/>

<http://www.nfsv4.org/>

## The NFS4 Pseudo Filesystem Client Access

- NFS4 access is implemented as a virtual filesystem
  - Must specify `-t nfs4` when mounting
  - Must mount `stationX:/` instead of `stationX:/exports/path`
    - Server's NFS4 exports are mounted as a single network filesystem
    - Enables client to browse server exports
    - Only a single entry in client's `/etc/fstab` is needed
    - Only one NFS4 mount per server
    - Shortest exported filesystem is usually the "root" export

```
mount -t nfs4 stationX:/ /stationX
```

7-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 784 3700.

The NFS filesystem modules are located in `/lib/modules/$(uname -r)/kernel/fs/{autofs4,nfs*}`.

Since all NFS4 exports for a particular server will be accessed under a single mountpoint, it is recommended to use a mountpoint name that reflects the server's name. The following example shows stationY mounting an NFS4 filesystem from stationX.

```
[root@stationY ~]# showmount -e stationX
Export list for stationX:
/exports          192.168.0.0/24
/exports/child_fs 192.168.0.0/24
[root@stationY ~]# mkdir /stationX
[root@stationY ~]# mount -t nfs4 stationX:/ /stationX
[root@stationY ~]# tree /stationX/
/stationX/
|-- child_fs
|   |-- test_file2
`-- test_file

1 directory, 2 files
```

Client browsing and access to the NFS4 pseudo filesystem is described in RFC 3530.

## The NFS4 Pseudo Filesystem Server Presentation

- All NFS4 exports *must* be presented as a single tree
  1. Choose one filesystem to be the NFS4 *root*
  2. Bind each additional exported filesystem below the NFS4 root

```
mount --bind /child_fs /exports/child_fs
```

- Define in /etc/fstab for persistence

### 3. /etc(exports options

- **fsid=0** indicates which filesystem is the root export
- **crossmnt** on the parent allows child filesystems to be exported
- **no\_subtree\_check** on each child export allows access to the child filesystem

7-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (818) 754 3700.

Quoting from RFC 3530:

It is the server's responsibility to present the pseudo filesystem that is complete to the client.

As an alternative to specifying **crossmnt** on the root export, you can add **nohide** to every child export.

**Hint:** when exporting multiple filesystems as an NFS4 pseudo filesystem, each export should appear in both /etc(exports and /etc/fstab.

## RPC Security using GSSAPI

- RPCSEC\_GSS provides an RPC wrapper for GSSAPI
- Enable `rpcsec_gss` in `/etc/sysconfig/nfs`

`SECURE_NFS=yes`

- Supporting daemons encapsulate security context within RPC's
  - `rpcidmapd` runs on server and client to provide identity mapping
    - `/etc/idmapd.conf` must be consistent on client and server
  - `rpcsvcgssd` runs on server to *receive* authentication requests
  - `rpcgssd` runs on client to *send* authentication requests

7-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[tca@us.redhat.com](mailto:tca@us.redhat.com)> or phone toll-free (USA) +1 (866) 626 2864 or +1 (819) 784 3700.

RFC 2203 defines the `rpcsec_gss` protocol and is used by NFS4 to wrap the strong security that Kerberos provides. Kerberos must be installed and configured on the server and clients in order for NFS4 to work. The following GSSAPI daemons incorporate this functionality for NFS4:

- `rpc.idmapd` transports account *information* and maps user and group identities between the client and the server. It is normally started with `/etc/init.d/rpcidmapd` but can also be started in the foreground for troubleshooting. See **man rpc.idmapd** for more information.  
Its configuration file, `/etc/idmapd.conf`, includes usable default settings. The default configuration squashes unknown users to *nobody* as specified in RFC 3530. See **man 5 idmapd.conf** for configuration help.
- `rpc.svcgssd` runs on the RPC server, receives account *authentication*, and establishes the security context with Kerberos via GSSAPI. It is normally started with `/etc/init.d/rpcsvcgssd` but can also be started in the foreground for troubleshooting. See **man rpc.svcgssd** for more information.
- `rpc.gssd` runs on the RPC client and sends account *authentication* to the server. It is normally started with `/etc/init.d/rpcgssd` but can also be started in the foreground for troubleshooting. See **man rpc.gssd** for more information.

You *must* enable `SECURE_NFS` before attempting to use NFS4. Additionally, the word *yes* must appear in lower-case, as indicated in this snippet from `/etc/init.d/rpc*gssd`:

```
# Check for and source configuration file otherwise set defaults
[ -f /etc/sysconfig/nfs ] && . /etc/sysconfig/nfs
[ "${SECURE_NFS}" != "yes" ] && exit 0
```

The `rpcsec_gss` daemons depend on kernel modules in

```
/lib/modules/$(uname -r)/kernel/net/sunrpc/
```

## NFS4 Security Modes

- Export restrictions in /etc(exports
  - Traditional IP/CIDR specifies mutual host and service authentication
    - Client must specify **sec=sys**
  - **gss/krb5** adds user authentication
    - Client must specify **sec=krb5**
  - **gss/krb5i** adds integrity checking via secure checksums
    - Client must specify **sec=krb5i**
  - **gss/krb5p** adds privacy via data encryption
    - Client must specify **sec=krb5p**

7-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2904 or +1 (919) 754 3700.

Security modes are cumulative. For example, **gss/krb5i** mandates Kerberos authentication and performs secure integrity checks on every NFS operation. The integrity check is an encrypted checksum. Specify **gss/krb5p** to use Kerberos for authentication, integrity, and privacy (i.e., data encryption).

### *Using traditional client restrictions*

This security mode appears to be the same as traditional NFS, but it actually uses Kerberos to authenticate the hosts. The server's /etc(exports uses traditional IP-based client restrictions:

```
/filesys 192.168.0.0/24 (ro,fsid=0)
```

When a client wants to connect to the server, it mounts by specifying nfs4 as the filesystem type:

```
mount -t nfs4 stationX:/ /stationX
```

The above command is the equivalent of running:

```
mount -t nfs4 -o sec=sys stationX:/ /stationX
```

### *Using advanced security:*

Specify the security mode in /etc(exports:

```
/filesys gss/krb5p(ro,fsid=0)
```

The client can view the required security mode using:

```
[root@stationY ~]# showmount -e stationX  
/filesys gss/krb5p
```

The server is requiring full data privacy, so the client must also specify krb5p as the security mode when mounting:

```
mount -t nfs4 -o sec=krb5p stationX:/ /stationX
```

## Current Limitations

- When using gss/krb5\* security modes:
  - gss/krb5\* prevents cross-filesystem binds
  - Pseudo filesystem must be fully within one filesystem
- Only one security mode per root export

7-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

## Example: Configuring an NFS4 Server

- Add Kerberos host and service principles to server keytab
  - host/stationX.example.com
  - nfs/stationX.example.com
- Configure /etc(exports)

```
# export /fs to nfs4 clients; v2 and v3 not allowed
/fs  gss/krb5p(rw,fsid=0)
```
- Configure rpcsec\_gss
  1. Edit /etc/sysconfig/nfs: SECURE\_NFS=yes
  2. Restart nfs to automatically start rpcidmapd and rpcsvcgssd

7-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

In this example, stationX is the NFS4 server.

The server's /etc/krb5.keytab must contain credentials for the host and service Kerberos principles. We are dealing with NFS in this instance, so the service principle is in the form of nfs/fully-qualified-domain-name.top-level-domain@REALM.

If this keytab or the required credentials are not present on the NFS4 server, then NFS4 exports will fail. One symptom would be authentication failures in the KDC's logfile.

In the example above, we are exporting the filesystem that is currently mounted on the server at /fs. For client restrictions, we are specifying that only properly-configured kerberized clients are able to access the filesystem. Once authentication has succeeded, all NFS operations are protected with encrypted checksums. Additionally, all data is encrypted to preserve privacy.

## Example: Configuring an NFS4 Client

- Add Kerberos host and service principles to client keytab
  - host/stationY.example.com
  - nfs/stationY.example.com
- Configure *rpcsec\_gss*
  1. Edit /etc/sysconfig/nfs: **SECURE\_NFS=yes**
  2. Enable & restart daemons: /etc/init.d/{**rpcidmapd, rpcgssd**}
- Mount options
  - Must specify filesystem type **nfs4**
  - Specify *rpcsec\_gss* security mode

```
mount -t nfs4 -o rw,sec=krb5p 192.168.0.X:/ /stationX
```

7-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

In this example, stationY is the NFS4 client.

The client's /etc/krb5.keytab must contain credentials for the host and service Kerberos principles. We are dealing with NFS in this instance, so the service principle is in the form of **nfs/fully-qualified-domain-name.top-level-domain@REALM**.

If this keytab or the required credentials are not present on the NFS4 server, then NFS4 mounts will fail. One symptom would be authentication failures in the KDC's logfile.

You must enable the **rpc{idmapd, gssd}** daemons in order for the correct kernel modules and supported programs to be run:

```
for DAEMON in rpcidmapd rpcgssd; do
    service $DAEMON restart
    chkconfig $DAEMON on
done
```

In the example above, we are mounting the filesystem that is currently exported from the server using **fsid=0**. Once authentication has succeeded, all NFS operations are protected with encrypted checksums. Additionally, all data is encrypted to preserve privacy.

Note especially that the filesystem type **nfs4** *must* be specified.

## Troubleshooting NFS4

- Troubleshoot all related hosts and services:
  - KDC and kadmin
  - /etc(exports and /etc/fstab
  - /var/log/{kadmind,krb5kdc}.log
  - /var/log/{secure,messages}
  - View name mappings on server: `cat /proc/net/rpc/nfs4.idtoname/content`
  - Flush server name mappings:  
`echo $(date '+%s') > /proc/net/rpc/nfs4.idtoname/flush`
  - Flush client name mappings: reload all nfs4-related modules (see notes)
- Restart nfssd on server
  - Unmount clients to avoid stale file handles: `umount -at nfs4`

7-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2894 or +1 (919) 754 3700.

To view name mappings on the server, run:

```
[root@stationX ~]# cat /proc/net/rpc/nfs4.idtoname/content
#domain type id [name]
192.168.0.0/24 user 500 nfs4user1@localdomain
192.168.0.0/24 user 0 root@localdomain
192.168.0.0/24 group 0 root@localdomain
192.168.0.0/24 group 501 nfs4user1@localdomain
```

Note that localdomain is listed since that is the default *Domain* setting in /etc/idmapd.conf.

Name mappings cannot currently be flushed on the client via the proc filesystem. To completely reload nfs4 modules on the client, you can:

1. unmount all nfs4 mounts: `umount -at nfs4`
2. stop the rpc{gssd,idmapd}, autofs daemons
3. unload kernel modules:

```
modprobe -r nfs lockd fscache nfs_acl rpcsec_gss_krb5 auth_rpcgss autofs4
```

4. start the rpc{gssd,idmapd}, autofs daemons
5. mount nfs4 filesystems

## General NFS Client Security

- Make sure UIDs/GIDs are consistent
  - NIS or LDAP can help
- If traffic will cross insecure or external networks outside your organization's control
  - Do not use NFSv2/3
  - Use NFS4, but carefully consider which security mode to use
- Avoid stale file handles
  - Unmount before restarting nfs server
  - Use the automounter to minimize number of NFS mounts
  - Export from server using `fslid=n`

7-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

For file permissions on your NFS exports to be meaningful, especially in the context of special permissions and ACL's, it is very important that all workstations and servers use centralized user identities. Otherwise, the wrong user may have access to sensitive files. Traditionally, NIS was designed for this role, but LDAP authentication is a modern alternative.

Again, since NFSv2/3 traffic is not encrypted, do not establish a mount with a server that will cause traffic to cross untrusted networks. Otherwise, there is a risk of exposing the data in every file that is accessed.

Use NFS4 wherever it is possible. Since it has the ability to authenticate users as well as to encrypt data, it is preferred. Even when using `sec=sys`, the hosts perform mutual authentication through Kerberos.

For the sake of performance, it is best to minimize the number of active NFS mounts on a workstation, particularly if they are coming from multiple NFS servers. If a lot of exports are hard mounted and their server goes down, the workstation can have problems even if the mounts are not being used. It is a good idea to use an automounter so that only the exports that are used are mounted at any one time.

Also try to avoid mounting exports from servers outside the organization's control. If the options `dev`, `suid`, or `exec` access are permitted, the administrator of the export can craft a device or program that a user on the system could use to gain inappropriate access. Even if while being cautious, the administrator could take the server off-line without warning, causing problems on the station until the stale NFS mounts are cleaned up.

## Client Mount Options

- For all NFS versions:
  - `rw / ro, async / sync`
  - `dev / nodev, uid / nosuid, exec / noexec`
  - `nouser / user`
  - `hard / soft, nointr / intr`
  - `lock / nolock`
  - `sec=sys|krb5|krb5i|krb5p` for NFS4 mounts
- See also the `nfs(5)` man page

7-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2894 or +1 (919) 754 3700.

## Client Mount Options

- Use `nodev` whenever possible
  - Honoring device files is rarely necessary
- Mount with `nosuid`, `noexec`, and `ro` whenever possible
- Use `sync` to improve data reliability
  - Always use with NFSv2 servers
- Use TCP unless UDP is your only option

7-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2904 or +1 (919) 754 3700.

NFS exports rarely have a legitimate need to include device files; you should almost always mount the export with `nodev`. Whenever feasible, it is a good idea to mount the share `nosuid` (disabling set-uid programs), `noexec` (disabling all programs), and `read-only`. In many cases, this will not be an possible due to the intended purpose of the export.

As with the server, it is possible to mount with the `sync` option to improve the reliability of data writes, especially if communicating with an NFSv2 server.

If it can be avoided, do not use the option `udp` with NFSv3, as it is connectionless.

## Client Mount Options

- Use hard for best data reliability
  - hard is the default behavior
- intr allows timed-out operations on a mount to be interrupted
  - nointr is the default, slightly more secure
- Do not use nolock
- port=N and mountport=N

7-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

In most cases, mount an NFS share with the default hard option. If at some point the NFS server fails to respond, the process will hang and continue retrying the command until the server responds. The alternative option, soft, will report an I/O error if the NFS server times out on a communication. This is less secure, as simple network congestion can lead to lost data.

It may be useful to give users and processes the option of reporting an I/O error rather than hanging on a stale hard mount forever. The intr option will allow an interrupt to cause a NFS operation timing out on a hard NFS mount to report an I/O error and continue like a soft mount. The default, nointr, is slightly more secure since it does not permit users this choice.

Whenever possible, do not use the nolock option. This disables file locking; if two programs on different clients try to write to a file at the same time and file locking is disabled, data corruption can be the result. It is sometimes necessary when mounting a share from an NFS server that does not correctly implement locking.

Two options, port=N and mountport=N, tell mount to communicate on particular ports to access rpc.nfsd and rpc.mountd if the portmap service has been blocked. For instance, if a workstation was blocked from accessing port 111 on server, but the user knew nfsd was running on port 2049, mountd was running on port 32770, and /share was exported, then the command:

```
$ mount -o port=2049,mountport=32770,nolock,server:/share /mnt/pnt
```

would still succeed in mounting that export on /mnt/pnt.

*tcp*

## **End of Unit 7**

- Questions and Answers
- Summary
  - NFS weakness
  - Static port settings
  - Server and export security
  - Client and mount security
  - Kerberized NFS
  - The NFS4 pseudo-filesystem

7-24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 826 2994 or +1 (919) 754 3700.

## Lab 7

# Network File System

---

<b>Goal:</b>	Improve security for traditional NFS servers and workstations. Deploy NFS4 in production for data privacy. Utilize single-sign-on via Kerberos and NIS for centralizing UID and GID name mappings.
<b>Estimated Duration:</b>	90 minutes
<b>System Setup:</b>	You and a partner should already have two installations of Red Hat Enterprise Linux using Kerberos and NIS for centralized authentication. Throughout this lab, stationX refers to the machine running the KDC and ypserv; stationY refers to your other installation.  Access to a cracker.org installation for testing.
	This lab requires that you have <code>portmap</code> and <code>nfs-utils</code> installed. Use <code>rpm -q</code> and <code>yum</code> to make sure the packages are installed.
<b>Situation:</b>	Management has asked you to place some kickstart configuration files in a 10 MiB <code>/kickstart</code> filesystem and make them available via NFSv3 to the classroom network 192.168.0.0/24 as well as cracker network at 192.168.1.0/24.  Additionally, management has heard of NFS4 and insists on using it to provide encrypted, read-write access to the following filesystems on stationX: <ul style="list-style-type: none"><li>• <code>/kickstart</code></li><li>• <code>/home</code></li><li>• <code>/projects</code> for nfs4 group collaboration</li></ul> You have been instructed to create new users <code>nfs4user1</code> and <code>nfs4user2</code> , both members of the group <code>nfs4group</code> , for access to the exported filesystems. The nfs4 users should never get shell access to the KDC or NFS server.

## **Sequence 1: Traditional NFSv3 export**

**Scenario:** Divide this complex lab scenario into manageable sequences, beginning with provisioning of the /kickstart filesystem via NFSv3.

**Deliverable:** An NFSv3 export at stationX:/kickstart with adequate security.

**Instructions:**

1. Configure NFS daemons to start on static ports.
2. Add firewall rules to allow the 192.168.0/24 and 192.168.1/24 networks access to the NFS-related ports.
3. Create and mount a 10 MiB ext3 filesystem at /kickstart. Label the filesystem "kickstart". Remember to make the mount persistent. Create a test file.
4. Export the /kickstart filesystem.
5. Test your export from stationY.
6. If root\_squash is the default export option, then why can root display the contents of the test file?

## **Sequence 2: Prepare security and single-sign-on**

**Scenario:** In this sequence, you will configure both the client and the server to use Kerberized NFS.

**Deliverable:** NFS4 exports secured with Kerberos principals.

**Instructions:**

1. On stationX, confirm that the /etc/krb5.keytab file contains the host/stationX.example.com principal.  
Create the principal nfs/stationX.example.com and add it to /etc/krb5.keytab.
2. On stationY, your partner's machine, confirm that the /etc/krb5.keytab file contains the host/stationY.example.com principal. Create the principal nfs/stationX.example.com and add it to /etc/krb5.keytab.
3. On stationX, enable *rpcsec\_gss* for NFS. Restart the nfs and rpcidmapd services.
4. On stationY, enable *rpcsec\_gss* for NFS. Restart the rpcidmapd and rpcgssd services. Ensure that rpcgssd will start on boot.
5. On stationX, create the users nfs4user1 and nfs4user2 and the group *nfs4group*. Both users should be members of that group and should not have valid NIS passwords.
6. Add Kerberos principals for nfs4user1 and nfs4user2. Set the passwords to redhat.
7. Test your configuration by logging in to stationX and stationY as nfs4user1 and nfs4user2 using ssh.
8. Prevent the *nfs4group* users from getting shell access on stationX.
9. Re-test ssh access to stationX as the nfs4 users.

## **Sequence 3: Add NFS4 exports**

**Deliverable:** NFS4 exports using `sec=sys` security to authenticate hosts and services

**Instructions:**

1. Prepare pseudo-filesystem on stationX. Create the directory `/projects`, which is setgid and writable by group `nfs4group`.
2. Export filesystems.
3. Mount from stationY.
4. Fix the home directories for the nfs4 users now that you have a working mount.
5. Test on stationY.

## **Sequence 4: Switch to Encrypted NFS4**

**Deliverable:** NFS4 exports using `sec=krb5p` to authenticate hosts, services, and users as well as to provide data integrity and encryption

**Instructions:**

1. On stationY, umount NFS filesystems and change `/etc/fstab` to specify `sec=krb5p` instead of `sec=sys`.
2. On stationX, change exports to require encryption.
3. On stationY, mount NFS filesystems.
4. On stationY, umount NFS filesystems.
5. On stationX, re-create the pseudo filesystem as one filesystem.
6. Export the new unified filesystem.
7. On stationY, retest!

## Sequence 5: Bypassing Portmap

**Scenario:** Use the cracker.org workstation to read files on your /kickstart NFSv2/3 export.

**Deliverable:** A mount point on the cracker.org workstation that bypasses portmap protections

**Instructions:**

1. On cracker.org, create a directory called /mnt/stationX. Attempt to mount stationX:/kickstart on cracker.org's /mnt/stationX. If you completed previous labs, this should fail because cracker.org is blocked from using portmap.
2. However, the NFS service ports themselves are still exposed. Can cracker.org find these ports and connect to them? On cracker.org, use the **nmap** port scanner to scan the privileged ports for **rpc.mountd**:

```
nmap -sR -p 1-1024 stationX.example.com
```

After a delay (this can take a long time, if you are scanning udp ports as well), one of the lines of output should read something like this:

```
892/tcp open (mountd V1-3)
```

This should provide enough information for us to establish the mounts from cracker.org without talking to **portmap** on stationX:

```
mount -o mountport=892,proto=tcp,port=2049,nolock stationX:/usr/ ↴  
exports-1 /mnt/stationX-1
```

3. On cracker.org, start editing a file in the /mnt/stationX directory. Confirm that you can write to the file. Do not exit from the editor.
4. On stationX.example.com, comment out the export for /kickstart in /etc/exports and reexport the directory (**exportfs -r**).
5. On cracker.org, continue editing the open file and attempt to resave it (but do not exit from the editor). Why did this fail?
6. Remove the comment sign from the /kickstart line in /etc/exports. Re-run the **exportfs -r** command. Then, on cracker.org, try to save your changes. Note that the change succeeded this time.

## **Sequence 6: Questions**

Instructions:

1. Assuming we implement NFS security measures similar to the ones in this lab, are there still ways an attacker can target the NFS service? What additional measures might we take to make attacks more difficult?
2. In this lab, we did not use the firewall to block cracker.org's access to the NFS server. Try to change the firewall rules from Sequence 1 to implement this method of access control. Are these services able to use TCP wrappers? Examine the manual pages. If so, try implementing the access controls that way. What advantages does one method have over the other?

## Sequence 1 Solutions

1. To configure static ports, add the following lines to `/etc/sysconfig/nfs`:

```
LOCKD_TCPPORT=890  
LOCKD_UDPPORT=890  
STATD_PORT=12001  
MOUNTD_PORT=892
```

Restart NFS and confirm that your settings are effective.

```
[root@stationX]# service nfs restart  
[root@stationX]# service nfslock restart  
[root@stationX]# rpcinfo -p | grep --color -e stat -e lock -e *  
mount -e nfs
```

```
100024      1    udp  12001  status  
100024      1    tcp   12001  status  
100003      2    udp   2049   nfs  
100003      3    udp   2049   nfs  
100003      4    udp   2049   nfs  
100003      2    tcp   2049   nfs  
100003      3    tcp   2049   nfs  
100003      4    tcp   2049   nfs  
100021      1    udp   890    nlockmgr  
100021      3    udp   890    nlockmgr  
100021      4    udp   890    nlockmgr  
100021      1    tcp   890    nlockmgr  
100021      3    tcp   890    nlockmgr  
100021      4    tcp   890    nlockmgr  
100005      1    udp   892    mountd  
100005      1    tcp   892    mountd  
100005      2    udp   .892   mountd  
100005      2    tcp   892    mountd  
100005      3    udp   892    mountd  
100005      3    tcp   892    mountd
```

2. Add firewall rules to allow access to the NFS-related ports.

First, we will remove the portmap rules from the NIS lab.

```
[root@stationX]# iptables -D RHS333 -s 192.168.0.0/24 -p udp <  
--dport 111 -j ACCEPT  
[root@stationX]# iptables -D RHS333 -s 192.168.0.0/24 -p tcp <  
--dport 111 -j ACCEPT
```

Then, give the world access to portmap and the NFS ports:

```
[root@stationX]# for PORT in 111 890 892 2049 12001; do  
for PROTO in tcp udp; do
```

```
iptables -A RHS333 -p $PROTO --dport $PORT -j ACCEPT
done
done
```

Review your iptables rules, then save:

```
[root@stationX]# iptables -nvL RHS333
...output omitted...
[root@stationX]# service iptables save
```

3. Create and mount a 10 MiB ext3 filesystem at /kickstart. Label the filesystem "kickstart". Remember to make the mount persistent. Create a test file.

```
[root@stationX]# mkdir /kickstart
[root@stationX]# lvcreate -L10M vol0 -n kickstart
[root@stationX]# mke2fs -j -L kickstart /dev/vol0/kickstart
```

Add to /etc/fstab:

```
LABEL=kickstart /kickstart ext3 defaults 0 0
```

Mount and create a test file:

```
[root@stationX]# mount /kickstart
[root@stationX]# echo "Traditional NFS ro test" > /
/kickstart/test
```

4. Export the /kickstart filesystem:

```
[root@stationX]# cat > /etc/exports << EOF
/kickstart 192.168.0.0/24(ro,sync,all_squash) \
192.168.1.0/24(ro,sync,all_squash)
EOF
```

Re-export all filesystems:

```
[root@stationX]# exportfs -r
```

5. Test your export from stationY.

Create a mount point:

```
[root@stationY]# mkdir /ks
```

Edit /etc/fstab to include the following line:

```
192.168.0.X:/kickstart /ks nfs soft,nosuid,nodev 0 0
```

Automatically mount all filesystems that can be automatically mounted:

```
[root@stationY]# mount -a
```

Confirm that you can read files from the export:

```
[root@stationY]# cat /ks/test  
Traditional NFS ro test
```

6. root can display the test file because default permissions allow the directory and file to be world-readable. root is mapped to the nfsnobody account.

## Sequence 2 Solutions

1. On stationX, confirm that the /etc/krb5.keytab file contains the host/stationX.example.com principal.

Create the principal nfs/stationX.example.com and add it to /etc/krb5.keytab.

Use ktutil and kadmin to confirm or create your principals:

```
[root@stationX ~]# kadmin
Authenticating as principal root/admin@STATIONX.EXAMPLE.COM with password.
Password for root/admin@STATIONX.EXAMPLE.COM:
kadmin: listprincs
...output truncated...
kadmin: addprinc -randkey nfs/stationX.example.com
WARNING: no policy specified for nfs/stationX.example.com@STATIONX.EXAMPLE.COM;
defaulting to no policy
Principal "nfs/stationX.example.com@STATIONX.EXAMPLE.COM" created.
→ -e des-cbc-crc :normal
kadmin: ktadd[nfs/stationX.example.com
Entry for principal nfs/stationX.example.com with kvno 3, encryption type
Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/krb5.keytab.
Entry for principal nfs/stationX.example.com with kvno 3, encryption type
ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal nfs/stationX.example.com with kvno 3, encryption type
DES with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal nfs/stationX.example.com with kvno 3, encryption type
DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: q
```

When finished, your keytab should resemble:

```
[root@stationX ~]# ktutil
ktutil: rkt /etc/krb5.keytab
ktutil: list
slot KVNO Principal
----- ↴
-----
1      3      host/stationX.example.com@STATIONX.EXAMPLE.COM
```

- ```

2      3      host/stationX.example.com@STATIONX.EXAMPLE.COM
3      3      host/stationX.example.com@STATIONX.EXAMPLE.COM
4      3      host/stationX.example.com@STATIONX.EXAMPLE.COM
5      3      nfs/stationX.example.com@STATIONX.EXAMPLE.COM
6      3      nfs/stationX.example.com@STATIONX.EXAMPLE.COM
7      3      nfs/stationX.example.com@STATIONX.EXAMPLE.COM
8      3      nfs/stationX.example.com@STATIONX.EXAMPLE.COM
...other possible entries truncated

2. On stationY, your partner's machine, confirm that the /etc/krb5.keytab file contains the host/stationY.example.com principal. Create the principal nfs/stationX.example.com and add it to /etc/krb5.keytab.

```

Use **ktutil** and **kadmin** to confirm or create your principals:

```

[root@stationY ~]# kadmin
Authenticating as principal root/admin@STATIONX.EXAMPLE.COM with
password.
Password for root/admin@STATIONX.EXAMPLE.COM:
kadmin: listprincs
...output truncated...
kadmin: addprinc -randkey nfs/stationY.example.com
WARNING: no policy specified for ↵
          nfs/stationY.example.com@STATIONX.EXAMPLE.COM;
defaulting to no policy
Principal "nfs/stationY.example.com@STATIONX.EXAMPLE.COM" ↵
  created. → e des-cbc-crc:normal
kadmin: ktadd[nfs/stationY.example.com
Entry for principal nfs/stationY.example.com with kvno 3, ↵
  encryption type
Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/krb5.keytab.
Entry for principal nfs/stationY.example.com with kvno 3, ↵
  encryption type
ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal nfs/stationY.example.com with kvno 3, ↵
  encryption type
DES with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal nfs/stationY.example.com with kvno 3, ↵
  encryption type
DES cbc mode with RSA-MD5 added to keytab ↵
  WRFILE:/etc/krb5.keytab.
kadmin: q

```

When finished, your keytab should resemble:

```

[root@stationY ~]# ktutil
ktutil: rkt /etc/krb5.keytab
ktutil: list

```

```

slot KVNO Principal
----- ✓
-----
1 3 host/stationY.example.com@STATIONX.EXAMPLE.COM
2 3 host/stationY.example.com@STATIONX.EXAMPLE.COM
3 3 host/stationY.example.com@STATIONX.EXAMPLE.COM
4 3 host/stationY.example.com@STATIONX.EXAMPLE.COM
5 3 nfs/stationY.example.com@STATIONX.EXAMPLE.COM
6 3 nfs/stationY.example.com@STATIONX.EXAMPLE.COM
7 3 nfs/stationY.example.com@STATIONX.EXAMPLE.COM
8 3 nfs/stationY.example.com@STATIONX.EXAMPLE.COM
...other possible entries truncated

```

- On stationX, enable *rpcsec\_gss* for NFS. Restart the nfs and rpcidmapd services.

Edit */etc/sysconfig/nfs* and add the following line:

**SECURE\_NFS=yes**

Restart NFS and rpcidmapd. Notice that rpcsvcgssd will fail to stop since it was not running initially. However, it starts successfully.

```

[root@stationX ~]# service nfs restart
Shutting down RPC svcgssd:                                [FAILED]
Shutting down NFS mountd:                                 [ OK ]
Shutting down NFS daemon:                                [ OK ]
Shutting down NFS quotas:                               [ OK ]
Shutting down NFS services:                             [ OK ]
Starting RPC svcgssd:                                  [ OK ]
Starting NFS services:                                 [ OK ]
Starting NFS quotas:                                  [ OK ]
Starting NFS daemon:                                   [ OK ]
Starting NFS mountd:                                    [ OK ]
[root@stationX ~]# service rpcidmapd restart
Shutting down RPC idmapd:                                [ OK ]
Starting RPC idmapd:                                   [ OK ]

```

- On stationY, enable *rpcsec\_gss* for NFS. Restart the rpcidmapd and rpcgssd services. Ensure that rpcgssd will start on boot.

Edit */etc/sysconfig/nfs* and add the following line:

**SECURE\_NFS=yes**

Start or restart the appropriate daemons:

```

[root@stationY ~]# service rpcgssd status
rpc.gssd is stopped

```

```
[root@stationY ~]# service rpcgssd start
Starting RPC gssd: [ OK ]
[root@stationY ~]# service rpcidmapd restart
Shutting down RPC idmapd: [ OK ]
Starting RPC idmapd: [ OK ]
[root@stationY ~]# chkconfig rpcgssd on
```

5. On stationX, create the users nfs4user1 and nfs4user2 and the group *nfs4group*. Both users should be members of that group and should not have valid NIS passwords.

Add users, rebuild maps, and confirm:

```
[root@stationX ~]# groupadd nfs4group
[root@stationX ~]# useradd -G nfs4group nfs4user1
[root@stationX ~]# useradd -G nfs4group nfs4user2
[root@stationX ~]# make -C /var/yp
make: Entering directory `/var/yp'
gmake[1]: Entering directory `/var/yp/stationX'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating netid.byname...
gmake[1]: Leaving directory `/var/yp/stationX'
make: Leaving directory `/var/yp'
[root@stationX ~]# ypcat passwd | grep nfs4
nfs4user2:x:501:502::/home/nfs4user2:/bin/bash
nfs4user1:x:500:501::/home/nfs4user1:/bin/bash
[root@stationX ~]# ypcat group | grep nfs4
nfs4group:x:500:nfs4user1,nfs4user2
nfs4user2:x:502:
nfs4user1:x:501:
```

6. Add Kerberos principals for nfs4user1 and nfs4user2. Set the passwords to redhat.

```
[root@stationX ~]# kadmin
Authenticating as principal root/admin@STATIONX.EXAMPLE.COM with password.
Password for root/admin@STATIONX.EXAMPLE.COM:
kadmin: addprinc nfs4user1
WARNING: no policy specified for nfs4user1@STATIONX.EXAMPLE.COM; defaulting to no policy
Enter password for principal "nfs4user1@STATIONX.EXAMPLE.COM": r
edhat
Re-enter password for principal "nfs4user1@STATIONX.EXAMPLE.COM": redhat
Principal "nfs4user1@STATIONX.EXAMPLE.COM" created.
kadmin: addprinc nfs4user2
```

```
WARNING: no policy specified for nfs4user2@STATIONX.EXAMPLE.COM; ↵
         defaulting to no
policy
Enter password for principal "nfs4user2@STATIONX.EXAMPLE.COM": ↵ ↵
edhat
Re-enter password for principal ↵
  "nfs4user2@STATIONX.EXAMPLE.COM": redhat
Principal "nfs4user2@STATIONX.EXAMPLE.COM" created.
```

7. Test your configuration by logging in to stationX and stationY as nfs4user1 and nfs4user2 using ssh.

```
[root@stationY]# ssh nfs4user1@stationX
[root@stationX]# ssh nfs4user1@stationY
```

Both of the above commands should work, but nfs4user1 will not have a home directory on stationY yet. If you cannot ssh to both hosts, review your steps above to make centralized authentication work.

8. Prevent the *nfs4group* users from getting shell access on stationX.

Begin by reading /usr/share/doc/pam-\*/\*txts/README.pam\_listfile for tips.

Create /etc/logingroups.deny:

```
[root@stationX]# echo 'nfs4group' > /etc/logingroups.deny
```

Edit /etc/pam.d/system-auth and add the following to the top:

```
auth required pam_listfile.so item=group sense=deny ↵
      file=/etc/logingroups.deny onerr=fail
```

9. Re-test ssh access to stationX as the nfs4 users. This time, you should be denied access to a shell on stationX despite entering the correct password.

```
[root@stationY]# ssh nfs4user1@stationX
```

/var/log/secure on stationX should have one or more messages similar to the following:

```
pam_listfile(sshd:auth): Refused user nfs4user1 for service sshd
```

## Sequence 3 Solutions

1. Prepare pseudo-filesystem on stationX. Create the directory /projects, which is setgid and writable by group nfs4group.

Create /projects as a group collaboration directory.

```
[root@stationX]# mkdir /projects  
[root@stationX]# chmod 3770 /projects  
[root@stationX]# chgrp nfs4group /projects
```

Create a directory tree for binding other filesystems. Very little space is needed, so use the root filesystem for now.

```
[root@stationX]# mkdir -p /exports/{kickstart,home,projects}
```

Edit /etc/fstab to add persistent bindings for the mountpoints you created above. Note that bindings should appear *after* the particular device has been mounted. For example, /home must be mounted before it can be bound to /exports/home.

```
/kickstart  /exports/kickstart  none    bind    0 0  
/home      /exports/home     none    bind    0 0  
/projects   /exports/projects  none    bind    0 0
```

Mount the directory tree:

```
[root@stationX]# mount -a
```

Confirm the mounts.

```
[root@stationX]# mount  
...output truncated...  
/home on /exports/home type none (rw,bind)  
/projects on /exports/projects type none (rw,bind)  
/kickstart on /exports/kickstart type none (rw,bind)
```

2. Export filesystems.

Edit /etc(exports:

```
/kickstart      192.168.0.0/24(ro,sync,all_squash) ↵  
                 192.168.1.0/24(ro,sync,all_squash)  
/exports        192.168.0.0/24(rw,sync,fsid=0,crossmnt)  
/exports/home   192.168.0.0/24(rw,sync,no_subtree_check)  
/exports/projects 192.168.0.0/24(rw,sync,no_subtree_check)  
/exports/kickstart 192.168.0.0/24(rw,sync,no_subtree_check)
```

Reload the exports table:

```
[root@stationX]# exportfs -r
```

Mount from stationY.

3. On stationY, view the export list for stationX.

```
[root@stationY ~]# showmount -e stationX
Export list for stationX:
/exports          192.168.0.0/24
/kickstart        192.168.0.0/24
/exports/home    192.168.0.0/24
/exports/projects 192.168.0.0/24
/exports/kickstart 192.168.0.0/24
```

Create a mountpoint on stationY:

```
[root@stationY]# mkdir /stationX
```

Edit /etc/fstab:

```
192.168.0.X:/   /stationX  nfs4  rw,soft,intr,nodev,sec=sys  0 0
```

To observe that sec=sys mounts use Kerberos for mutual host and service authentication, open a root shell on stationX and leave it open in order to follow the krb5 logs:

```
[root@stationX ~]# tail -f /var/log/k*
```

Mount the NFS4 root export on stationY while following the krb5 logs on stationX:

```
[root@stationY]# mount -a
```

On stationX, you should see log messages reporting the Kerberos ticket exchange.

4. Fix the home directories for the nfs4 users now that you have a working mount.

On stationX:

```
[root@stationX]# usermod -d /stationX/home/nfs4user1 nfs4user1
[root@stationX]# usermod -d /stationX/home/nfs4user2 nfs4user2
[root@stationX]# make -C /var/yp
```

5. Test on stationY.

```
[root@stationY]# ssh nfs4user1@stationY
[nfs4user1@stationY ~]$ tree /stationX
...output truncated...
|   |-- zshenv
|   '-- zshrc
|-- home
|   '-- lost+found [error opening dir]
|   '-- nfs4user1
|       '-- nfs4user2 [error opening dir]
|-- kickstart
```

```

|   '-- lost+found [error opening dir]
`-- projects
    '-- testroot

[nfs4user1@stationY ~]$ ls -ld /stationX/projects/
drwxrws--T 2 root nfs4group 4096 Mar 12 20:24 /
    /stationX/projects/
[nfs4user1@stationY ~]$ echo 'this is a test' > /
    /stationX/projects/test1
[nfs4user1@stationY ~]$ ls -l !$
ls -l /stationX/projects/test1
-rw-rw-r-- 1 nfs4user1 nfs4group 15 Mar 12 2007 /
    /stationX/projects/test1

```

*Tip:* If you do not see the proper group ownership as expected, it's possible that the client has outdated mappings and needs to be flushed. Either reboot the client or try the following steps:

```

[root@stationY]# umount -at nfs4
[root@stationY]# umount -at nfs
[root@stationY]# service rpcgssd stop
[root@stationY]# service rpcidmapd stop
[root@stationY]# service ypbnd stop
[root@stationY]# modprobe -r nfs lockd fscache nfs_acl \
    rpcsec_gss_krb5 auth_rpcgss autofs4
[root@stationY]# service ypbnd start
[root@stationY]# service rpcidmapd start
[root@stationY]# service rpcgssd start
[root@stationY]# mount -a

```

You have already seen that hosts are authenticated when mounting an NFS4 share. Now test that sec=sys mounts do not use Kerberos for user authentication.

```

[nfs4user1@stationY ~]$ kdestroy
[nfs4user1@stationY ~]$ klist
klist: No credentials cache found (ticket cache
FILE:/tmp/krb5cc_500_htlkHM)

Kerberos 4 ticket cache: /tmp/tkt500
klist: You have no tickets cached
[nfs4user1@stationY ~]$ cat /stationX/projects/test1
this is a test
[nfs4user1@stationY ~]$ vi !$
vi /stationX/projects/test1
[nfs4user1@stationY ~]$ !cat
cat /stationX/projects/test1
this is a test
another test

```

## Sequence 4 Solutions

1. On stationY, unmount NFS filesystems.

```
[root@stationY]# umount -a -t nfs4  
[root@stationX]# umount -a -t nfs
```

Change /etc/fstab specifications from sec=sys to sec=krb5p.

```
[root@stationY]# sed -i 's/sec=sys/sec=krb5p/' /etc/fstab
```

2. On stationX, change exports to require encryption.

```
[root@stationX]# sed -i <  
'/exports/{s/192.168.0.0\//24/gss\//krb5p/}' /etc(exports  
exportfs -r
```

3. On stationY, mount NFS filesystems.

```
[root@stationY]# mount -a
```

Connect to stationY as nfs4user1 and attempt to access a child filesystem on stationX.

```
[root@stationX]# ssh nfs4user1@stationY  
[nfs4user1@stationY ~]$ tree /stationX  
/stationX/  
|-- home  
|-- kickstart  
`-- projects
```

Note that the directories are empty. Bound filesystems do not work when using user authentication (gss/krb5), integrity checking (gss/krb5i), or data encryption (gss/krb5p).

4. On stationY, unmount NFS filesystems.

```
[root@stationY]# umount -at nfs4  
[root@stationY]# umount -at nfs
```

5. On stationX, re-create the pseudo filesystem as one filesystem. This can be done within the root filesystem or on another partition. It does not matter as long as the entire export tree is within a single filesystem. The sample solution given here assumes that space is available with the root filesystem.

Taking NFS off-line ensures that no active changes are made to files within the directories during while copying.

```
[root@stationX]# service nfs stop  
[root@stationX]# mkdir /krb5exports  
[root@stationX]# cp -a /kickstart !$  
[root@stationX]# cp -a /home !$  
[root@stationX]# cp -a /projects !$
```

If you created `/krb5exports` as a separate filesystem, remember to add an entry to `/etc/fstab`.

6. Edit `/etc(exports` to export the new unified filesystem. Comment out the existing NFS4 exports and add a new line:

```
/krb5exports gss/krb5p(rw, sync, fsid=0)
```

Reload the exports table:

```
[root@stationX]# exportfs -r
```

7. On stationY, mount the nfs4 filesystem.

```
[root@stationY ~]# mount -a
```

As nfs4user1 on stationY, test the mounts.

```
[root@stationY]# ssh nfs4user1@stationY
[nfs4user1@stationY ~]$ tree /stationX
...output truncated...
|   |-- zshenv
|   '-- zshrc
|-- home
|   |-- lost+found [error opening dir]
|   '-- nfs4user1
|       '-- nfs4user2 [error opening dir]
|-- kickstart
|   '-- lost+found [error opening dir]
 '-- projects
     '-- testroot

[nfs4user1@stationY ~]$ ls -ld /stationX/projects/
drwxrws--T 2 root nfs4group 4096 Mar 12 20:24 /
    /stationX/projects/
[nfs4user1@stationY ~]$ echo 'this is a test' > /
    /stationX/projects/test1
[nfs4user1@stationY ~]$ ls -l !$
ls -l /stationX/projects/test1
-rw-rw-r-- 1 nfs4user1 nfs4group 15 Mar 12 2007 /
    /stationX/projects/test1
```

As an additional test, use `vi` to edit some test files. Note, however, that `vi` will appear to hang if you have not correctly set up the NFS4 mounts.

*Tip:* If you do not see the proper group ownership as expected, it's possible that the client has outdated mappings and needs to be flushed. Either reboot the client or try the following steps:

```
[root@stationY]# umount -at nfs4
```

```
[root@stationY]# umount -at nfs
[root@stationY]# service rpcgssd stop
[root@stationY]# service rpcidmapd stop
[root@stationY]# service ypbond stop
[root@stationY]# modprobe -r nfs lockd fscache nfs_acl \
    rpcsec_gss_krb5 auth_rpcgss autofs4
[root@stationY]# service ypbond start
[root@stationY]# service rpcidmapd start
[root@stationY]# service rpcgssd start
[root@stationY]# mount -a
```

## **Unit 8**

### **OpenSSH**

**8-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## **Objectives**

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

- SSH server configuration
- SSH client configuration
- TCP port and X11 forwarding

**8-2**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2964 or +1 (919) 754 3700.

## Vulnerabilities

- Unencrypted network connections prone to sniffing, spoofing, and hijacking
- Need to protect authentication
  - Defend against credential capture attacks
  - Permit alternative authentication methods
- Need to protect data
  - Provide interactive login security
  - Provide security for other data channels

8-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Resolutions

- Use SSH for interactive login sessions
- Configure public key user authentication between trusted hosts
- Protect unencrypted data channels with TCP port forwarding

8-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Service Profile: sshd

- Type:**#**System V-launched service
- Packages:**#**openssh, openssh-server
- Daemons:**#**sshd
- Scripts:**#**sshd
- Port:**#**22/tcp (ssh)
- Configuration:**#**/etc/ssh/sshd\_config
- Related:**#**openssh-{askpass, askpass-gnome}, openssh-clients, openssl

8-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

## Server Configuration

- `/etc/ssh/sshd_config`
  - Protocols
  - User authentication methods
  - User access controls
  - Login messages and logging
- **Host key files**
  - `ssh_host*_key`
  - `ssh_host*_key.pub`
  - `moduli`

8-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cctrainlog@redhat.com](mailto:cctrainlog@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

### *Server-side configuration*

The main configuration file for sshd is `/etc/ssh/sshd_config`. It controls which versions of the SSH protocol, encryption ciphers, and MACs are offered to the client. It determines which user authentication methods may be used, which users may login, and the settings for messages displayed to the user.

Every host has its own unique public key pair that is used for authenticating the server to the client. In fact, the OpenSSH server generates three host keys the first time it starts up, stored in `/etc/ssh`. Two files are used for RSA authentication and key negotiation when using version 1 of the SSH protocol. The `moduli` file contains parameters for the Diffie-Hellman key agreement used by version 2 of the protocol. The DH negotiation may be authenticated with either RSA or DSA host keys, both of which are stored in a private key file and a matching public key file.

## SSH Protocols

- Two major versions of the protocol
  - Use protocol version 2 whenever possible
  - SSH protocol version 1 is subject to message integrity attacks
- Protocol directive
  - Lists protocols offered to the client; client gets to pick which one will be used

8-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 784 3700.

### *SSH protocol issues*

There are two major versions of the SSH protocol, version 1 and version 2. Whenever possible, avoid using the version 1 protocol. It used a CRC-32 checksum as a weak message authentication code. Since CRC-32 is not cryptographically strong, message injection attacks are possible. The second version of the protocol uses client-server negotiations to select among stronger hashes for its message authentication code, and is not vulnerable to this attack.

The Protocol directive in `sshd_config` may be used to control which versions of the protocol will be supported by the server. The server does not get to request a preference -- if both are offered, the client may decide which version to use. By default, both version 2 and version 1 are offered. If you set `Protocol 2`, your server will only accept version 2 of the protocol, which will increase session security but may lock out older clients.

## Server Authentication

- Transport layer authentication
  - Encryption cipher and MAC is negotiated
  - Diffie-Hellman key exchange, server is authenticated by DSA or RSA public key
  - Client compares server public key against file containing public keys of known hosts
- If server key is not in client's known hosts file, client can not tell if key is legitimate or not

8-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 828 2994 or +1 (919) 754 3700.

### *SSH transport protocol*

When a SSHv2 connection is initiated, the client and server negotiate which key exchange method, encryption cipher, and MAC will be used. The client picks from the options offered by the server. Normally, both sides generate a shared secret with a Diffie-Hellman key exchange, authenticated with the server's public host key. The client has a local file containing public host keys of known servers. If the key offered by the server matches the key in the local file, the server (and the key exchange) is authenticated. At this point the connection is secure, and user authentication begins. If the key does not match, the client aborts the connection.

If the server does not have an entry in the known hosts file on the client, there is no way for the client to tell if the key really belongs to that server. The key might belong to an attacker who is attempting to insert themselves as a "man in the middle" of the connection. This can be avoided if good host keys are distributed in advance by the system administrator. In some cases this is not feasible, and by default users are shown the hash of the public host key received and asked whether they want to risk accepting it. (The hash is short enough to be communicated verbally.)

## User Authentication

- Once secure connection is established, user authentication method is negotiated
  - PasswordAuthentication
  - PubkeyAuthentication
    - RSAAuthentication for SSHv1
  - HostbasedAuthentication
    - Insecure, off by default
- Other methods are available but not as well supported

8-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 764 3700.

### *User authentication*

There are three standard methods of user authentication that are well supported by the SSH protocol. The server offers which modes are available, and the client may try each in turn.

Password authentication allows the client to send the password for an account on the server. This method is enabled by default.

Public key authentication allows a user to store a public key on the server in advance. If someone attempts to ssh into that account, their client will be sent a challenge encrypted with that public key. If the client possess the matching private key, it can decrypt the challenge and send it back to the server. Anyone who can use the private key is treated as the user, and therefore it must be carefully protected. If so, this provides a secure method of authentication without a password. There are two different modes, for version 2 and 1 of the SSH protocol, and both are on by default.

Host-based authentication is not secure, and should not be used. This method emulates the old /etc/hosts.equiv and ~/.rhosts functionality of rsh. Some additional security is provided by the requirement that public key authentication of the host must also succeed. Standard public key authentication is usually a better solution. Host-based authentication is off by default.

Other authentication methods are also available, but are not as well supported or are less secure.

## User Access Control

- **User access**
  - AllowUsers/AllowGroups
  - DenyUsers/DenyGroups
  - PermitRootLogin
- **StrictModes**
  - If a user's SSH configuration files or home directory are world-writable, deny access

8-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. if you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 826 2964 or +1 (919) 754 3700.

### *User access control*

Several options are available to better manage user access through OpenSSH. You may configure your server to permit only a few users or groups of users, or conversely, disallow a select few. If you explicitly permit users or groups, then only those included are permitted access.

With a default install, root is permitted access. PermitRootLogin can be set to no, to without-password (disabling PasswordAuthentication for root), or to forced-commands-only (disabling all authentication except PubKeyAuthentication when there is a forced command set for the key in the `~/.ssh/authorized_keys` file).

Another user control option, StrictModes, checks that the access modes of the user directory and configuration files on the server. If the directory is group or other writable, user public key authentication fails. Users on Red Hat Enterprise Linux have a default umask of 002, which causes new files and directories to have group write access: ensure that access modes for user files are writable for the user only!

## Login Messages

- **Banner**
  - Specifies a file containing a message to print out prior to authentication
  - Useful for acceptable use warning
- **PrintMotd**
  - Print out /etc/motd on interactive login

8-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

### *Informational messages*

Some sites need to display a “Terms of Use” warning notice prior to authentication on a server. The SSH version 2 protocol supports the **Banner** option, which displays a specified file to the client before authentication. Both SSH protocols support the display of /etc/motd after successful authentication.

## Logging Activity

- **SyslogFacility**
  - Red Hat uses **AUTHPRIV** by default
- **LogLevel**
  - Not the same as *syslog* log levels
  - **DEBUG** log levels not recommended for normal operation

8-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ttraining@redhat.com](mailto:ttraining@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 5700.

### *Logging account activity*

Successes and failures to establish a session or authenticate user access is logged through syslog. The default configuration shipped with Red Hat Enterprise Linux uses the AUTHPRIV facility. The SyslogFacility option may be used to change this setting.

LogLevel sets the verbosity of sshd logging, not the syslog priority it uses. The possible levels are QUIET, FATAL, ERROR, INFO, DEBUG, DEBUG2, and DEBUG3. The default level is INFO; avoid DEBUG levels because sensitive information about user connections is exposed. If an alternate facility is used, ensure that access modes to the log file are properly restricted. Remember that syslog messages forwarded to another host over the network are sent unencrypted.

## Client Configuration

- `/etc/ssh/ssh_config`
  - Users may use command line options or their `~/.ssh/config` to override
- Divided into Host sections
- Protocol directive
  - Which versions of the SSH protocol to try, in which order

8-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 5700.

### *Client-side configuration*

A system wide configuration file, `/etc/ssh/ssh_config`, sets the default configuration options for the ssh client. These options may be overridden by the user through a personal copy of the file in `~/.ssh/config`, and both these files may be overridden by command line options.

The client configuration file is divided up into Host sections. The Host directive is followed by a matching expression indicating which hosts the following settings apply to. A Host section is ended by the next Host directive. Host \* matches all hosts and may be used for global defaults. Different Host sections may have different settings for configuration file directives. For example, the Protocol directive may be set differently in different Host blocks, changing which versions of the SSH protocol will be attempted with which hosts. (Protocol 2,1 will cause the client to try protocol version 2 first, and if that is not available will fall back to version 1.)

## Client-side Server Authentication

- `/etc/ssh/ssh_known_hosts`
  - Users may have `~/.ssh/known_hosts`
  - `UserKnownHostsFile`
- `StrictHostKeyChecking`
  - Default `ask` lets users decide whether to accept an unknown server key

8-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2894 or +1 (919) 754 3700.

### *Client-side authentication of the server*

In order to authentication the server in the initial key exchange, the client keeps files containing public keys for all known ssh servers. There is a system-wide file, `/etc/ssh/ssh_known_hosts`, and each user may have their own file in `~/.ssh/known_hosts`. When the client connects to a server, it sends its public key. The client looks for an appropriate entry in its key files, and if it finds one and it matches, the server is authenticated. If it finds one, but it does not match, the connection is aborted. (An unexpected host key change may indicate that an attacker is attempting to hijack the connection and is sending their own bogus server key.)

If it does not find one, there is a problem. The client can not verify that the key actually belongs to the server and not an attacker. The `StrictHostKeyChecking` option controls what happens next. If set to yes, then the connection is aborted. This requires that the known hosts files are kept up to date manually, but is the most secure. If set to no, the client automatically accepts the unknown key without question. This is the least secure. The default setting is ask, which notifies the user of the problem and presents them with a hash of the server's public key. This hash is short enough to provide by telephone or through some other trusted channel. The user decides if they trust the unknown key is legitimate, and accept or reject it and the connection.

## Client-side User Authentication

- User authentication methods to request
  - PasswordAuthentication
  - PubkeyAuthentication
  - Key files and ~/.ssh/authorized\_keys
  - HostbasedAuthentication
  - Insecure, do not use
- Methods actually available are under the control of the remote server

8-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *Client-side user authentication*

The user can specify which user authentication methods they want the client to attempt, if available. For example, the server can offer password authentication, but the client may be configured not to attempt it.

Public key authentication requires some user configuration in advance. First, the user must generate a public key pair with ssh-keygen appropriate for the version of the protocol in use.

To generate a RSA key pair for user authentication with SSH protocol version 2:

```
$ ssh-keygen -t rsa
```

This will create two files, `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`. The first file is the private key, and must be kept secure. When generated, it may be protected with a password so that it can not easily be used by an attacker if it is stolen. The second file is the public key. This file is copied to the server that the user wants to log into using public key authentication. The contents of the file are then appended to `~/.ssh/authorized_keys`.

When a user attempts public key authentication, a random challenge is generated and messages are sent encrypted with each of the authorized keys in turn. If one of these messages can be successfully decrypted and sent back to the server, that proves the client controls one of the matching private keys.

Other key files are `id_dsa` and `id_dsa.pub` (generated with `ssh-keygen -t dsa` for use with the version 2 protocol) and `identity` and `identity.pub` (generated with `ssh-keygen -t rsa` for use with the version 1 protocol).

## Protecting Private Keys

- Can password protect private keys
- **ssh-agent** can hold key passwords
  - Enter passwords once at start of session
  - Manage passwords with **ssh-add**
- Makes key theft harder, not impossible

8-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ctraining@redhat.com](mailto:ctraining@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *Password protection on private authentication keys*

One of the main reasons public key authentication is used is to provide password-free SSH authentication. However, if the private key is not protected by a password and the host it is stored on is compromised, it may be stolen by the attacker and used to gain access to other systems.

Password-free SSH authentication can still be provided by using ssh-agent to register private key passwords. This program is run automatically when X is started. The **ssh-add** command is then used to store or delete passwords from the authentication agent. Once a password is stored in the agent, it is automatically provided to ssh when needed for public key authentication. It is common to have ssh-add run automatically when X starts up. The utility `gnome-session-properties` can be used in the default GNOME desktop environment to configure this. ssh-add can also be run from the command line. With the `-l` or `-L` options, it lists all keys for which passwords are stored. Passwords can be deleted from the agent with `-D`, or individually with `-d key-file`.

While this makes private key theft harder, if the client host is compromised it is still possible. For example, the attacker could replace ssh-add with a version that collects passwords, or could use a debugger to examine memory used by ssh-agent to capture the passwords. Remember that the SSH security model assumes that the end hosts are secure and the network is insecure.

## authorized\_keys Options

- Limits how a particular public key is used for authentication
- Comma separated options precede the key on its line in authorized\_keys
  - from="hostpattern"
  - command="command"
  - no-port-forwarding

8-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 826 2994 or +1 (919) 754 3700.

### *Limiting acceptable uses of authorized keys*

In the authorized\_keys file, each key is on a single line. The key lines specify the key information in several space-delimited fields:

SSHv1 key: *options bit-size exponent modulus comment*

SSHv2 key: *options key-type base64-key comment*

In both cases, the comment field is used to identify the key to human beings, but is not used by ssh or sshd.

The options field is an optional comma-delimited list of directives that can be used to limit how the key pair may be used for authentication. Some useful options include:

- from="hostpattern" only allows the key to be used if the connection is originating from a host in the comma-delimited *hostpattern*. Wildcards are \* and ?, and ! can be used to remove hosts from the list. (Example: from="\*.example.com").
- command="command" forces the command listed to run instead of the user's interactive shell or any command they may specify on their ssh client's command line. This limits how this key pair can be used. (Root can be restricted to forced-commands-only mode.)
- no-port-forwarding and no-X11-forwarding turns off those options. permitopen="host:port" can be used to limit which host and port can be used with port forwarding.
- no-pty prohibits the user from being assigned a pty; this may be useful if they need an 8-bit clean channel with the command option.

*Example:*

```
command="/usr/bin/top",no-port-forwarding ssh-dss  
AK4LjaHjaOneAtR[...more key stuff...] user@example.com
```

## stunnel

- Provides secure access to insecure services
- Uses SSL, no built in cryptography
  - need a certificate
- Protect against interception of data
- Prevents data manipulation

8-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

### *stunnel*

stunnel provides security to insecure services such as POP, IMAP, telnet, rlogin etc. This can be done without making changes to the code of the service. For example, xinetd can invoke stunnel as a wrapper for the services it controls, or it can be run from standalone services. It sends insecure data to stunnel, which tunnels this securely over the net. It can also receive encrypted data, sending it to another machine, or launch a service on localhost to talk to a remote server through the tunnel.

stunnel performs authentication using the SSL handshake. Using SSL ensures that both authentication and data transfer is secure.

Using stunnel is very simple. Here is an example:

```
$ stunnel -d imaps -r localhost:imap
```

This basically says this: listen to the imaps port 993. Encrypt the traffic and forward it to the imap port 143 on localhost.

For more information, see <http://www.stunnel.org>.

## Port Forwarding

- *ssh* and *sshd* can forward TCP traffic
- Obtuse syntax can be confusing
  - -L clientport:host:hostport
  - -R serverport:host:hostport
- Can be used to bypass access controls
  - Requires successful authentication to remote *sshd* by client
  - AllowTcpForwarding

8-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

### TCP port forwarding

The *ssh* client can initiate connections in which either it or the remote server forward arbitrary TCP traffic over the secure channel to another network port. The syntax to use local port forwarding or remote port forwarding is confusing and can be difficult to understand.

The -L clientport:host:port option sets up local port forwarding. For example:

```
$ ssh -L 3025:mail.example.com:25 -N station1.example.com
```

Tells *sshd* on *station1.example.com*:

“I, the *ssh* client, will listen for traffic on my host's port 3025 and send it to you, *sshd* on *station1*. You will decrypt it and forward that traffic to port 25 on *mail.example.com* as if it came from you.”

The -R serverport:host:port option sets up remote port forwarding. For example

```
$ ssh -R 3025:mail.example.com:25 -N station1.example.com
```

Tells *sshd* on *station1.example.com*:

“You, the *sshd* on *station1*, will listen for traffic on your port 3025 and send it to me. I, *ssh*, will decrypt it and forward that traffic to port 25 on *mail.example.com* as if it came from me.”

This is useful to secure arbitrary traffic, although often the host specified is localhost because the forwarded traffic is not encrypted. However, this capability can also be used to bypass firewalls. Remember, the remote *sshd* must have *AllowTcpForwarding* yes set (the default) and the client must be able to initiate the connection and successfully authenticate for SSH port forwarding to work. In addition, by default the forwarded port (3025 in the examples) is bound to localhost. *GatewayPorts* yes must be set in the configuration file for *ssh* (with -L) or *sshd* (with -R) to allow remote hosts to use the port. (The *ssh* command may instead use the -g option with -L.)

## X11 Forwarding

- Special case of port forwarding
  - `sshd` forwards a port on the server over the SSH channel to client's local X server
  - `$DISPLAY` and `xauth` keys are set on the server automatically
- Attacker that controls the remote server can eavesdrop on your client's X display
  - `ForwardX11` and `-X` option

8-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (619) 754 3700.

### *X11 forwarding security*

Red Hat Enterprise Linux configures X11 forwarding on by default. This is a special case of TCP port forwarding. On interactive login, traffic to port 6010 is typically forwarded from the remote server over the secure channel to ssh, which decrypts it and forwards it to the local X server. At the same time, `$DISPLAY` is set on the server to send X protocol traffic to that port, and `xauth` keys are set to give the remote user access to the local client's X display. The result from a user perspective is that they can use ssh to log into a remote system, run an X client, and have its interface appear automatically and securely on their local display.

The danger of X11 forwarding is that if the remote server is compromised, an attacker can capture the `xauth` keys and gain access to port 6010, and use it connect to the local X display. At that point, the remote attacker can run hostile X clients on the display to monitor keystrokes, or otherwise eavesdrop on or interfere with the display. In the client configuration file, `ForwardX11 no` is set to disable X11 forwarding. To turn it on, change this to `ForwardX11 yes` or use the `-X` option to the ssh command.

## **End of Unit 8**

- **Questions and Answers**
- **Summary**
  - Server authentication
  - User authentication
  - Port forwarding security issues

**8-21**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

# **Lab 8**

## **OpenSSH Configuration**

---

**Goal:** Learn about SSH port forwarding and public key authentication.

**Estimated Duration:** 45 minutes

**System Setup:** Standard installation after preceding labs.

Use **rpm -q** to verify that the **openssh**, **openssh-clients**, and **openssh-server** packages are installed.

Make sure that you can resolve your system hostname, both fully-qualified and not, and the your IP address also reverse resolves to your hostname.

Verify that the user accounts **peter**, **paul**, and **mary** from Lab 1 still work. Also verify that your firewall is still permitting new connections to reach port 22/tcp (**sshd**).

## **Sequence 1: Public Key Authentication**

### **Instructions:**

1. As paul, generate a public key for authentication:  
Save the `id_rsa` and `id_rsa.pub` files and set a valid passphrase for your private key.
2. Transfer a copy of the public key (`id_rsa.pub`) to your partner's machine. Save it in paul's account as `~paul/.ssh/authorized_keys` and make sure that it is readable and writable only by paul.
3. Use `ssh` (as paul) to log into your partner's machine as paul. You should be prompted your private key's passphrase, but you should need no other password to authenticate successfully.  
Log out.
4. If paul started the X11 session (through the graphical login screen or `startx`) on your host, you may use `ssh-add` to allow `ssh-agent` (you must start this manually if it did not start automatically) to provide your passphrase when needed.
5. Try to `ssh` to your partner's machine again. This time no passphrase should be needed. The passphrase will be forgotten when `ssh-agent` exits on logout, or you can use `ssh-add -D` to make `ssh-agent` forget all passphrases at any time.

## Sequence 2: Port Forwarding

**Scenario:** The SSH protocol allows arbitrary TCP traffic to be forwarded over the secure connection in a sub-channel. In this sequence, we will investigate how this works and how it can be used to bypass host access controls. You will **telnet** to port 25/tcp on a partner's workstation and, using SMTP protocol commands, send mail to user **mary** on that station. The tricky part is that your partner's workstation will only be allowing connections to that port from the localhost interface.

In order for this sequence to work, your partner must be running **sshd**, it must permit port forwarding, you must be able to connect to it, and you must be able to authenticate as a user on his or her system.

**Instructions:**

1. Your partner's machine will be referred to below as *remote*. First, try to connect directly to TCP port 25 on stationY with the **telnet** command.  
The connection should be refused.
2. Next, use **ssh** to open a tunnel connecting a new listening port on your workstation to the remote port 25/tcp. The command below will cause connections to port 3025/tcp on your workstation to be encrypted by **ssh**, forwarded to the **sshd** on stationY, decrypted, and forwarded as cleartext by stationY to localhost:25. The **-N** option stops **ssh** from opening an interactive shell on stationY. You will need an account on stationY for authentication; the example uses user **peter**.
3. The previous command should appear to hang. Open a new terminal window. In this window, connect to your local listening port on the **ssh** client.
4. In this terminal, send an e-mail to user **mary**. You should type the lines in bold in response to the prompts in the SMTP exchange. The exact responses you see may vary slightly from the text below.

```
HELO localhost
250 stationY Hello stationY [127.0.0.1], pleased to meet you
MAIL FROM: root@stationY
250 2.1.0 root@stationY... Sender ok
RCPT TO: mary@stationY
250 2.1.5 mary@stationY... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
This is a test message for RHS333 Lab 7.
.
250 2.0.0 hB7L2xNN003257 Message accepted for delivery
QUIT
```

5. Check **mary**'s e-mail on *remote*. The message should be in her mailbox.

6. On your machine, type **Control-c** in the **ssh** terminal window, or otherwise kill the process. This should close the **ssh** tunnel.

7. Next, we will disable port forwarding on station Y. On station Y, add the following line to **/etc/ssh/sshd\_config**:

```
AllowTcpForwarding no
```

Restart **sshd** to apply your changes:

```
service sshd restart
```

8. Repeat steps 2 and 3 again. The connection should fail, and in the **ssh** client window you should see an error message like:

```
channel 1: connection failed: administratively prohibited: open failed
```

You should also look in **/var/log/secure** on station Y to see the error message logged there.

## **Sequence 1 Solutions**

1. As paul, generate a public key for authentication:

```
[paul@stationX] $ ssh-keygen -t rsa
```

4. If paul started the X11 session (through the graphical login screen or startx) on your host, you may use **ssh-add** to allow **ssh-agent** (you must start this manually if it did not start automatically) to provide your passphrase when needed:

```
[paul@stationX] $ ssh-add  
Enter passphrase for /home/paul/.ssh/id_rsa: {your passphrase}  
Identity added: /home/paul/.ssh/id_rsa (/home/paul/.ssh/id_rsa)
```

## Sequence 2 Solutions

1. Your partner's machine will be referred to below as `remote`. First, try to connect directly to TCP port 25 on stationY with the `telnet` command:

```
[root@stationX]# telnet stationY 25
```

The connection should be refused.

2. Next, use `ssh` to open a tunnel connecting a new listening port on your workstation to the remote port 25/tcp. The command below will cause connections to port 3025/tcp on your workstation to be encrypted by `ssh`, forwarded to the `sshd` on stationY, decrypted, and forwarded as cleartext by stationY to localhost:25. The `-N` option stops `ssh` from opening an interactive shell on stationY. You will need an account on stationY for authentication; the example uses user `peter`.

```
[root@stationX]# ssh -L 3025:localhost:25 -N peter@stationY
```

3. The previous command should appear to hang. Open a new terminal window. In this window, connect to your local listening port on the `ssh` client.

```
[root@stationX]# telnet localhost 3025
```

You should see output similar to the following:

```
Trying 127.0.0.1...
Connected to remote (127.0.0.1).
Escape character is '^].
220 remote ESMTP Sendmail 8.12.10/8.12.10; Sun, 7 Dec 2003
16:02:59 -0500
```

Notice that even though you have connected to port 3025 on your host, you are communicating with port 25 on your partner's host, stationY.

7. Next, we will disable port forwarding on stationY. On stationY, add the following line to `/etc/ssh/sshd_config`:

```
AllowTcpForwarding no
```

Restart `sshd` to apply your changes:

```
service sshd restart
```

## **Unit 9**

### **Electronic Mail with Sendmail**

**9-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## **Objectives**

**Upon completion of this unit, you should be able to:**

- Server topologies ad service security
- Encryption of email messages
- Securing user access to the mail spool
- Connection filtering

**9-2**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2504 or +1 (919) 784 2700.

## Vulnerabilities

- Denial of service attacks
- Users need access to the mail spool
  - Users have local account on mail server or need to access the server externally
- E-mail is sent as clear text
- Need to control transmission of "spam"
  - May still need to let legitimate hosts relay

9-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (819) 754 3700.

## Resolutions

- Protect and tune the Sendmail service
- Require users to access the mail server through a secure IMAP or POP service
- Encrypt e-mail messages
- Filter e-mail with a blacklist, blackhole, and/or procmail configuration

9-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Server Topologies

- In-only
- Out-only
- Relay-only
  - Firewalls may separate the MTAs
- Mail system components
  - MUA (Mail User Agent)
  - MTA (Message Transfer Agent)
  - MDA (Mail Delivery Agent)

9-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2984 or +1 (919) 754 3700.

One of the most useful, and most troublesome, capabilities of a mail transfer agent is its ability to relay messages on the behalf of other systems. This relay capability is necessary for many types of common and useful MTA configurations.

Many organizations delegate the responsibility for handling mail to and from the “outside world” to a single system or, at most, a few dedicated systems. This central mail server may be the only host permitted to transmit mail through the firewall. Relaying is necessary for these configurations, as internal clients must be able to relay through the mail server in order to send mail. In some situations it may also be necessary for mail servers outside the organization to relay through a mail server to reach systems on the organization’s internal network.

- An *in-only* mail server will only accept messages destined for accounts on the local system on which the server is running.
- An *out-only* mail server will not accept messages from outside the domain but will relay mail from systems on the inside domain to systems on the outside.
- A *relay-only* server does not accept any mail for local clients. A relay-only server exists strictly to pass mail between different domains, such as on a firewall.

To send a message, a user invokes a mail user agent (MUA), such as mail or pine. The MUA hands the message off to a mail transfer agent (MTA), usually Sendmail or Postfix, which contacts the MTA on the destination server and hands the message off to the mail delivery agent (MDA) that deposits the message in the recipient’s mailbox. The recipient may read that mailbox directly, or download it through a mail access agent (MAA) providing IMAP or POP service.

## User Mail Access

- Mail-only accounts
  - On server vs. off server
- NFS exported mail spool is bad idea
  - Potential for userX on client to authenticate as userY on server
- Use cyrus-imapd or dovecot (pop or imap)
  - Unencrypted imap or pop not recommended

9-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 626 2894 or +1 (919) 754 3700.

The simplest approach to setting up a mail server is to create interactive login accounts for users on the mail server. Users can then send and read mail using any one of numerous MUAs that can directly read their file in `/var/spool/mail`. However, this forces you to allow users on the same host as the mail server.

A more secure approach is to disallow interactive login access to the mail server itself and require users to access their e-mail from a separate system. The user accounts still exist on the mail server, but interactive login access is prohibited by setting the user's shell to `/bin/false`. There are a number of different methods to provide the mail-only users with access to their e-mail.

One popular method is to NFS share the `/var/spool/mail` directory on the server and then have each client mount that share on its `/var/spool/mail` directory. Although convenient, this is not recommended. In addition to possible locking problems that can exist between different implementations of the NFS protocol this opens up other serious security holes. One of the worst is caused by the weakness of NFS user authentication. If a user controls the root account on any remote host mounting the share, they can claim to be running as any non-privileged UID that owns files on the share. This would allow them to read and delete any user's mail stored in `/var/spool/mail`.

A more secure method is to require client systems to use a mail access agent such as POP3 or IMAP to transfer mail from the server. If this approach is chosen, use more secure versions of these services, POP3S and IMAPS. The insecure versions of these protocols transmit all data as clear text, including authentication information.

Control of which ones are to be enabled at startup is handled by editing `/etc/cyrus.conf` and commenting out the appropriate lines then running service `cyrus-imapd <start|stop|restart>`:

# add or remove based on preferences

```
imap      cmd="imapd" listen="imap" prefork=5
imaps     cmd="imapd -s" listen="imaps" prefork=1
pop3      cmd="pop3d" listen="pop3" prefork=3
pop3s     cmd="pop3d -s" listen="pop3s" prefork=1
```

## User Mail Privacy and Security

- Encryption of message before submission to mail system most secure
- Two popular solutions
  - GNU Privacy Guard (GnuPG)
  - Secure Multipurpose Internet Mail Extensions (S/MIME)

9-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[trainlog@redhat.com](mailto:trainlog@redhat.com)> or phone toll-free (USA) +1 (800) 828 2904 or +1 (919) 754 3700.

The best way to ensure privacy of an e-mail message is to encrypt the message from the time it is sent until the time it is read by the recipient. Both GNU Privacy Guard (OpenPGP) and Secure MIME (S/MIME) offer this capability.

OpenPGP and S/MIME can be used for both message authentication and privacy. In message authentication, the sending application generates a hash of the message and encrypts the hash using the sending user's private key. The recipient of the message can decrypt this hash using the sender's public key and compare the decrypted value with a value calculated for the received message. If the two hashes do not match then the message has been altered in some way.

For privacy, OpenPGP and S/MIME can be used to encrypt a message. For efficiency, the message is encrypted using a randomly generated symmetric key, called the session key. This session key is then encrypted using the recipient's public key. To decrypt the message, the recipient first decrypts the session key with his or her private key and then uses the session key to decrypt the message.

Another use for these applications stems from the concept of "non-repudiation" or the incapacity of the sender being able to deny having sent a message. Legally binding repudiation is hard to implement electronically. Nevertheless, sign or encrypt messages that is sent to guarantee that the sender, or someone with access to the sender's private key, did send the message and the message has not been altered in transit. Conversely, someone who receives a message from someone else that has been signed or encrypted can take that as a guarantee that the message was sent by the right person or someone with access to their private key.

## GnuPG or S/MIME?

- **GnuPG**
  - Sign and encrypt
  - Widely available in Linux MUAs
  - Based on a "Web of Trust" PKI
- **S/MIME**
  - Sign and encrypt
  - Integrates easily in mixed environment
  - Hierarchical Public Key Infrastructure PKI

9-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 636 2884 or +1 (919) 754 3700.

The OpenPGP standard (implemented by GnuPG) and the S/MIME standard are functionally equivalent, but not compatible. Both applications use MIME and are based around public key exchange and have many similar features.

GnuPG can be used with many MUAs in Red Hat Enterprise Linux. It is based on a "web of trust" model. In the web of trust, a user can designate a key from a person known to them as "trusted". That trusted person's key can then be used to validate keys from people the user does not know but whom the trusted person knows. The gpg command can be used to encrypt, decrypt, and sign messages and to manage keys.

S/MIME is integrated into many common MUAs such as Netscape and Outlook. It is based around a hierarchical structure of digital certificates signed by trusted certificate authorities. If this reminds you of OpenSSL, it should; the openssl smime command can manipulate S/MIME messages, and the certificates can be normal X.509 certificates in PEM format. Some authorities argue that S/MIME is better suited for large user bases in corporate environments.

## Other Security Concerns

- Dangers of HTML mail and automatic MIME execution
  - Web bugging
  - Viruses

9-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cstraining@redhat.com](mailto:cstraining@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

Most modern graphical MUAs allow users to send messages with HTML formatting. This allows for more visually appealing messages but can lead to security and privacy problems. One vulnerability opened up by allowing the use of HTML in mail messages is *web bugging*.

Web bugs are used by some web site operators to track users. A web bug is typically a link to a small image, typically a 1 pixel by 1 pixel image, stored on a remote server. The image is designed so that it does not actually display in the user's browser (or in this case, e-mail agent) but when the receiving client retrieves the image, the HTTP request sent to the image's server identifies where the request came from. This can provide a back door for someone to gain "inside information" about the configuration of your internal network. Web bugs in email messages have also been used by spammers to verify active email addresses, or as the first step in a DNS cache poisoning attack.

MIME itself can present problems for the recipient of a mail message. The intent behind MIME types is to allow the browsing application to start the appropriate program to interpret the MIME encoded data contained in a mail message. Most MUAs can be configured to automatically start the appropriate application to display data as it is encountered. If the encoded data is some form of executable program or shell script, this can lead to disaster if the encoded application contains malicious code.

## Service Profile: Sendmail

- **Type:** System V-launched service
- **Packages:** sendmail, sendmail-cf
- **Daemons:** sendmail
- **Script:** sendmail
- **Ports:** 25/tcp
- **Configuration:** #/etc/mail/sendmail.cf,  
/etc/mail/submit.cf, /etc/aliases, /etc/mail/
- **Related:** procmail, imap
- **Alternative:** postfix

9-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2004 or +1 (919) 784 3700.

The default MTA on Red Hat Enterprise Linux is Sendmail. Sendmail is perhaps the most widely used MTA on UNIX systems, and therefore an understanding of secure Sendmail configuration is important.

| Service                     | Permissions | Normally runs as                         |
|-----------------------------|-------------|------------------------------------------|
| /usr/sbin/sendmail.sendmail | 2755        | root.smmsp, depends on mode of operation |

Two sendmail processes are run at service startup. One acts as a MTA, uses sendmail.cf as its configuration file, and runs as user root, group smmsp. The other processes the client submission queue, uses submit.cf as its configuration file, and runs as user smmsp, group smmsp.

/usr/sbin/sendmail is a symbolic link to /etc/alternatives/mta , which is a symbolic link to the /usr/sbin/sendmail.sendmail binary by default. On a system using Postfix as its MTA, this link will point to /usr/sbin/sendmail.postfix instead.

| File                    | Permissions         | Purpose                                            |
|-------------------------|---------------------|----------------------------------------------------|
| /etc/mail/sendmail.cf   | 644, root.root      | Configuration file for the MTA                     |
| /etc/mail/submit.cf     | 444, root.root      | Configuration file for the mail submission program |
| /etc/aliases            | 644, root.root      | For forwarding, source of /etc/aliases.db          |
| /etc/mail/*             | 644, root.root      | Supplementary configuration files                  |
| /etc/mail/sendmail.mc   | 644, root.root      | Macro control file to build sendmail.cf            |
| /var/spool/mail         | 775, root.mail      | Spool directory for locally delivered e-mail       |
| /var/spool/mqueue       | 700, root.mail      | Queue of messages being processed by MTA           |
| /var/spool/clientmqueue | 770,<br>smmsp.smmsp | Queue of messages being processed by MSP           |

## Server Security

- `confSMTP_LOGIN_MSG`
  - Change version banner
- `confPRIVACY_FLAGS`
  - Suppress information leakage

9-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

Some people say that the less information provided to hostile parties the better off they are. One simple step that can be taken is to obscure the version information that Sendmail automatically provides when someone connects to port 25. This banner message can be modified in `/etc/mail/sendmail.mc`:

```
define(`confSMTP_LOGIN_MSG', `Your message here')
```

The `confPRIVACY_FLAGS` options can be used to suppress information leaked through SMTP commands or status queries. The option `goaway` can be used to turn off almost all SMTP status queries, and includes all options except `nobodyreturn`, `noreceipts`, `noetrn`, `restrictqrun`, `restrictmailq`, and `restrictexpand`. It may break communication with programs that do not strictly adhere to the SMTP or ESMTP protocols.

## Server Security: DoS

- **sendmail.mc options to control DoS attacks**
  - confMAX\_MESSAGE\_SIZE
  - confMAX\_DAEMON\_CHILDREN
  - confCONNECTION\_RATE\_THROTTLE
  - confMIN\_FREE\_BLOCKS

9-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2904 or +1 (919) 754 3700.

Denial of service attacks can be remarkably easy to carry out against Sendmail. One of the most effective attacks is to repeatedly send large messages or messages with large attachments to an account on the targeted system thereby filling up the mail spool directory and effectively preventing mail messages from reaching the other users of that system. Complicating matters is the fact that it can sometimes be difficult to distinguish between a DDoS attack in progress and a high volume of email activity.

There are several parameters that can be set in /etc/mail/sendmail.mc to reduce the effectiveness of DoS attacks:

- confMAX\_MESSAGE\_SIZE - maximum size of message to accept, default is unlimited,
- confMAX\_DAEMON\_CHILDREN - refuse connections when this number of child processes has been reached, the default is unlimited,
- confCONNECTION\_RATE\_THROTTLE - maximum connections per second,
- confMIN\_FREE\_BLOCKS - number of blocks that must be available on mail spool device to accept mail, default is 100 blocks.

There are a few other parameters you may wish to modify that can help stop or delay DoS attacks:

- confMAX\_HEADERS\_LENGTH - maximum length of all message headers in bytes,
- confMAX\_HOP - number of times messages can pass through server before being considered in a loop,
- confMAX\_RCPTS\_PER\_MESSAGE - limit maximum number of recipients for a message,
- confREFUSE\_LA - incoming mail connections will be refused if the system load average increases above this value.

## Server Security: File Permissions

- Restrict write access to root only
  - Configuration files
  - Database files
- `confSAFE_FILE_ENV`
  - Only deliver to regular files
- `confDONT_BLAME_SENDMAIL`
  - Turns off safety checks, avoid using

9-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Another simple but important step to secure Sendmail is to make sure that file ownerships and permissions are set correctly for the various configuration files and directories. All configuration files, map databases and directories should be owned by root. In addition, make sure that only root has write access to these locations. Under the default Sendmail installation this includes:

- `/etc`
- `/etc/mail`
- `/etc/mail/sendmail.cf`
- `/etc/mail/*`

The sendmail configuration directories should contain only regular files, not symbolic links. The mail queue directory, `/var/spool/mqueue`, should be owned by root. The mail spool directory, `/var/spool/mail`, should be owned by user root and group mail and have permissions 0775 set.

The `SafeFileEnvironment` variable can be used to specify a directory that sendmail will chroot to before delivering messages. This also prevents sendmail from delivering messages to symbolic links. To set in `/etc/mail/sendmail.mc`:

```
define(`confSAFE_FILE_ENV', `some_directory')dn1
```

One configuration option that should be avoided is `confDONT_BLAME_SENDMAIL`. This option is used to tell sendmail to ignore various file permission safety checks.

## smrsh

- Restricted shell for local mail delivery
- Limits which commands .forward files or /etc/aliases can pipe mail into
  - Programs must be installed in /etc/smrsh
  - Built-ins: echo, exec, exit
- Strips off initial pathnames on programs
- Restricted special characters
  - No I/O redirection or subshells

9-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (650) 626 2994 or +1 (918) 754 3700.

The Sendmail restricted shell, smrsh, provides a secure environment for the local mail delivery program. After Sendmail receives the message from the remote mail transfer agent (MTA) it hands the message to the local delivery program. The local delivery program can go ahead and deposit the message in the recipient's mailbox or it can pass the message off to another program for further processing.

The message can be handed off to another program by using the syntax “*program\_name*” in either the recipient's .forward file or by setting up an appropriate entry in /etc/aliases . For example, orders might be an entry in /etc/aliases that points all mail messages directed to orders to an automatic order entry system:

```
orders:#| /usr/local/bin/process_orders
```

Although useful, such programs, especially poorly written ones, can serve as pathways for attackers to exploit. If the program somehow allows an attacker to pass his or her own commands through, those commands will be executed as if that person were logged in interactively. smrsh provides a restricted shell environment for the local delivery program and allows the system administrator to specifically control which commands are available to the local delivery program.

Using smrsh, the only commands available to the local delivery program are the built-in commands “echo”, “exec” and “exit”. Any other program that the local delivery program is allowed to use must exist in the /etc/smrsh directory. Never consider placing certain programs, such as the Perl interpreter, in /etc/smrsh . In addition to restricting which programs can run, smrsh also restricts most special characters and strips leading path information off of command names before trying to execute them. The smrsh feature is enabled by default.

The Postfix local daemon can also be configured to use the Sendmail smrsh restricted shell by running the command

```
$ postconf -e "local_command_shell = /usr/sbin/smrsh -c"
```

## STARTTLS

- TLS protection for SMTP communication
- If offered by the remote server, Sendmail will always attempt to use TLS
- Very useful, but there are limitations
  - Only protects the next hop; no guarantee of end-to-end protection if relayed
  - Messages not protected in queues or spool

9-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email [cttrain@redhat.com](mailto:cttrain@redhat.com) or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

STARTTLS is an extension to the SMTP protocol that provides support for TLS encryption and authentication of SMTP messages sent between two mail servers. If STARTTLS is offered by a server, Sendmail will always attempt to use it by default. To configure a server to offer TLS encryption, obtain a CA-signed certificate (and private key) for the mail server and the CA's public certificate. Set the following defines in /etc/mail/sendmail.mc:

```
define(`confSERVER_CERT', `/etc/mail/certs/sendmail.pem')
define(`confSERVER_KEY', `/etc/mail/certs/sendmail.pem')
```

where confSERVER\_CERT defines the location of the PEM encoded certificate, and confSERVER\_KEY defines the location of the PEM encoded private key. They may be in the same file or different files. Also set:

```
define(`confCACERT_PATH', `/etc/mail/certs')
define(`confCACERT', `/etc/mail/certs/my-ca.crt')
```

where confCACERT is a file containing the CA's public certificate, and confCACERT\_PATH is a directory that may contain other CA certificates. For faster lookup, set up special symlinks in that directory for each CA certificate file:

```
$ for i in *.crt; do ln -s $i $(openssl x509 -noout -hash < $i).0; done
```

STARTTLS is very useful, but it only protects the network connection between two mail servers. If a message is being relayed, there is no guarantee that the next hop will also be encrypted. Furthermore, the message itself will be (perhaps temporarily) stored unencrypted on disk in any queues and mail spools it may happen to pass through. If a guaranteed end-to-end protection for an e-mail message is wanted, encrypt it with S/MIME or GnuPG.

## **Anti-Spam Mechanisms**

- Local mechanisms
  - /etc/mail/access and blacklisting
- Remote mechanisms
  - dnsbl: MAPS and other DNS blackholes
- Local delivery filtering
  - procmail and SpamAssassin

**9-16**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## /etc/mail/access

- Used to accept or deny incoming email

|                 |                     |
|-----------------|---------------------|
| spammer@aol.com | REJECT              |
| spamRus.net     | REJECT              |
| 10.3            | OK                  |
| virtualdom.com  | RELAY               |
| user@dom9.com   | ERROR:500 disabled  |
| nobody@         | ERROR: 550 bad name |

- By default, can not send or receive messages for entries resulting in ERROR or REJECT
  - FEATURE(`blacklist\_recipients')

9-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

The /etc/mail/access database file can reject email from individual users (the first line above), entire domains (the second line), or an entire IP subnet (third line).

As shown on the fourth and fifth lines, the access database can also direct sendmail to accept or relay email from subnets and domains. The second column of this file can be one of a few different values:

- REJECT - rejects the sender with a general purpose message
- OK - accepts mail (for receipt, *not relay*) even if other rules, such as failed DNS lookup, might reject it.
- RELAY - accept mail for relaying
- DISCARD - claim to accept delivery, but discard the message completely. You should normally use REJECT instead.
- ERROR : 550 "Some text" - like REJECT but rejects with an arbitrary text message.

If not using the default error code (550), should specify both RFC1893 and RFC821 error codes:ERROR:4.2.2:450 quota exceeded

The blacklist\_recipients feature, that is turned on by default, automatically disables the sending of mail to any addresses listed in /etc/mail/access.

## Authenticated Relay

- Clients may be authorized to relay using other methods besides an address
  - Useful for laptop users and other stations with variable IP addresses or hostnames
- STARTTLS client certificates
- SMTP AUTH and SASL mechanisms

9-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (888) 626 2984 or +1 (919) 754 3700.

In addition to allowing relaying based on hostname or IP address, relaying may be allowed if a client successfully authenticates itself to the mail server. This is useful for mobile users that may not have a fixed IP address or may be operating on networks that include clients that should not be allowed to relay through the organization's mail server automatically.

STARTTLS may be used to provide authentication through client-side TLS certificates. On the Sendmail-based *client* system, two additional defines must be set, confCLIENT\_CERT and confCLIENT\_KEY, which point to the PEM formatted client certificate and private key respectively. On the Sendmail *server* that will authenticate relaying, special rules need to be added to the /etc/mail/access map. A CERTISSUER rule specifies the distinguished name of a CA that issued a certificate, and a CERTSUBJECT rule specifies the distinguished name of the certificate itself. For example:

CERTISSUER: \ ↵

```
/C=US/ST=North+20Carolina/L=Raleigh/O=Example,+20Inc./CN=Certificate+20Author  
y/emailAddress=root@example.com \  
SUBJECT
```

CERTSUBJECT: \ ↵

```
/C=US/ST=North+20Carolina/L=Raleigh/O=Example,+20Inc./CN=stationX.example.com  
mailAddress=root@example.com \  
RELAY
```

would indicate the certificates issued by the DN specified on the CERTISSUER line will next have their subject DN checked, and that certificates that also have the CERTSUBJECT DN specified will be permitted to relay.

The SMTP AUTH protocol uses SASL, the Simple Authentication and Security Layer, to provide authentication mechanisms. The client authenticates itself by providing an authorization ID, an authentication ID (often the same), and a password. Several different mechanisms are provided by SASL, including PLAIN (simple, but not secure by itself), shared secret mechanisms like DIGEST-MD5 and CRAM-MD5, and Kerberos-based mechanisms like GSSAPI. STARTTLS is needed to protect some

**SMTP AUTH** modes, but **SMTP AUTH** allows more traditional mechanisms such as usernames and passwords to be used in place of TLS client certificates to authenticate clients and authorize relaying.

More information on setting up **STARTTLS** and **SMTP AUTH** is available in  
`/usr/share/sendmail-cf/README`.

## DNS Blackhole Lists

- DNS can be used as a database to store information about sources of "spam"
  - FEATURE(`dnsbl','nospam.example.com')
- DNS lookup using reversed octets of sender IP address, with the domain of the blackhole list appended
- Can implement locally or use public or subscription-based blackhole services

9-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2004 or +1 (919) 754 3700.

DNS-based blackhole lists allow Sendmail to combat unsolicited e-mail by checking the address of a connecting MTA to see if it has been listed on the blackhole DNS server as a known source of "spam". Any mail originating from a site listed in the blackhole list is rejected, unless overridden by the /etc/mail/access map.

Use the FEATURE(`dnsbl') in /etc/mail/sendmail.mc to enable DNS blackhole lookup mechanism. By default, this will use a subscription-only service, MAPS, but other blackhole lists can be specified.

The operation of the DNS blackhole mechanism is simple. When a remote MTA request a connection, sendmail reverses the octets in the address of the connecting MTA and prepends it to the domain name *blackholes.mail-abuse.org*. If the nameserver at MAPS returns a resource record, then the mail transfer is rejected.

As an example, if sendmail receives a request from a MTA at 192.168.1.254 it will do a name lookup on the address: 254.1.168.192.blackholes.mailabuse.org.

On the MAPS DNS server, resource records for known sources of SPAM take the form:

254.1.168.192.blackholes.mail-abuse.org#IN#A#127.0.0.2

The address of 127.0.0.2 is arbitrary.

The default action of the dnsbl feature if the server can not be found is to allow mail to be forwarded. Large sites may wish to configure a name server as a slave of the authoritative blackhole server.

A site specific implementation of dnsbl list can be done by adding the appropriate resource records to the nameserver and modifying the dnsbl configuration line in /etc/mail/sendmail.mc to contain the name of the nameserver:

FEATURE(`dnsbl', `server1.example.com')dn1

Try the following example: host 228.146.182.207.dnsbl.njabl.org

## **End of Unit 9**

- Questions and Answers
- Summary
  - Server security
  - Data security
  - Securing user email
  - Blocking spam

**9-20**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (800) 626 2984 or +1 (919) 794 3700.

## **Lab 9**

### **Sendmail**

---

- Goal:** Use the a DNS blackhole list to block mail from particular hosts. Use the /etc/mail/access map to exempt hosts from the blackhole list and to selectively allow relaying.
- Estimated Duration:** 60 minutes
- System Setup:** Select a partner for this lab. In the sections below, "your machine" refers to stationX.example.com, and "your partner's machine" refers to stationY.example.com.

## **Sequence 1: Initial Setup**

### **Instructions:**

1. Verify that you have `sendmail-cf.*.rpm` installed on both machines. It is needed to use the M4 macro language.
2. On your machine, edit the `/etc/mail/sendmail.mc` file to allow connections from remote systems. Change the line that reads:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

to read:

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

3. Restart Sendmail to rebuild the `/etc/mail/sendmail.cf` file:
4. Make sure that remote systems can access your mail server through your firewall.
5. Have your partner send a message to `mary@stationX.example.com`. Did you receive it?

## **Sequence 2: Set up and use a DNS blackhole list**

**Scenario:** In this sequence you will set up a DNS server which will be used not to provide name service, but to store a central database of hosts which should not be permitted to send e-mail to your mail server.

**Instructions:**

1. Make sure that your name server is still working. Replace each X below with your station number:

```
[root@stationY]# host stationX.domainX.example.com
```

2. From stationY.example.com, telnet to the SMTP port and send a mail message:

```
[user@stationY]$ telnet stationX.example.com 25
Trying 192.168.0.X...
Connected to stationX.example.com (192.168.0.X).
Escape character is '^].
220 stationX.example.com ESMTP Sendmail 8.12.10/8.12.10; Sun,
7 Dec 2003 16:02:59 -0500
HELO stationY.example.com
250 stationX.example.com Hello stationY.example.com [192.168.0.Y]
|,
pleased to meet you
MAIL FROM: root@stationY.example.com
250 2.1.0 root@stationY.example.com... Sender ok
RCPT TO: mary@stationX.example.com
250 2.1.5 mary@stationX.example.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hello, Mary!
.
250 2.0.0 hB7L2xNN003257 Message accepted for delivery
QUIT
```

3. Now we will add an entry to the DNS blackhole list to block e-mail from stationY.example.com. Edit the /var/named/chroot/var/named/domain-internalX.zone configuration file on stationX.example.com to include the following resource record:

```
Y.0.168.192.domainX.example.com. IN A 127.0.0.2
```

Update the SOA serial number, save the file, and reload the nameserver.

4. Add the following line (it should all be on one line) to /etc/mail/sendmail.mc (before the MAILER directives):

```
FEATURE(`dnsbl', `domainX.example.com', `550 Message from
"`${client_addr}" blocked.' )dnl
```

5. Restart **sendmail** to rebuild the **sendmail.cf** file.
6. Try to repeat step 3. This time, the connection should be prohibited. Consider using a packet sniffer such as **tcpdump** or **wireshark** to monitor the communication; do you see the DNS query sent to the DNS blackhole server?

## **Sequence 3: The access map and relaying**

### **Instructions:**

1. Edit `/etc/mail/access` on stationX.example.com so that it contains the following line, and restart `sendmail` again:

```
stationY.example.com OK
```

2. Have your partner send stationX.example.com another message. The rule in the access map should override the information in the DNS blackhole list, and the message should go through.

3. Change the entry in `/etc/mail/access` to allow relaying messages:

```
stationY.example.com RELAY
```

4. Restart `sendmail` on both stationX and stationY.

5. This time, from stationY, connect to the mail server on stationX but use the mail server to relay the message back to stationY:

```
[user@stationY] $ telnet stationX.example.com 25
Trying 192.168.0.X...
Connected to stationX.example.com (192.168.0.X).
Escape character is '^'.
220 stationX.example.com ESMTP Sendmail 8.12.10/8.12.10;
Thu, 11 Dec 2003 23:15:52 -0500
HELO stationY.example.com
250 stationX.example.com Hello stationY.example.com [192.168.0.Y]
|,
pleased to meet you
MAIL FROM: root@stationY.example.com
250 2.1.0 root@stationY.example.com... Sender ok
RCPT TO: mary@stationY.example.com
250 2.1.5 mary@stationY.example.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hello again, Mary!
.
250 2.0.0 hB7L2xNN003257 Message accepted for delivery
QUIT
```

Check mary's e-mail on stationY.example.com. Was your message relayed properly?

## Sequence 4: Challenge Project

### Instructions:

1. Sendmail support for STARTTLS can use SSL certificates to authenticate relaying through the server. STARTTLS also provides data integrity and confidentiality for a connection between the client and next-hop server. You can generate a CA-signed SSL certificate using the configuration from Lab 3 and set up STARTTLS using the following `sendmail.mc` file directives:

```
define(`confCACERT_PATH', `/etc/pki/CA')
define(`confCACERT', `/etc/pki/CA/my-ca.crt')
define(`confSERVER_CERT', `/etc/pki/tls/certs/sendmail.crt')
define(`confSERVER_KEY', `/etc/pki/tls/certs/sendmail.key')
define(`confCLIENT_CERT', `/etc/pki/tls/certs/sendmail.crt')
define(`confCLIENT_KEY', `/etc/pki/tls/certs/sendmail.key')
```

`confCACERT` points to your CA certificate for client certificate authentication. `confCACERT_PATH` is optional and points to other CA certificates (and symlinks named with their hashes pointing to them). `confSERVER_CERT` and `confSERVER_KEY` point to a certificate and private key in PEM format used to authenticate Sendmail as a server receiving mail; `confCLIENT_CERT` and `confCLIENT_KEY` point to a certificate and key used to authenticate Sendmail when sending mail to another server. The server certificate and client certificate may be identical.

Then you can use rules in the `access` map to permit relaying. `CERTISSUER` rules match the issuer name of the certificate, and `CERTSUBJECT` matches certificate subject names. For example:

```
CERTISSUER:/C=US/ST=North+20Carolina/L=Raleigh/O=Example,
+20Inc./CN=Station+20X+20Certificate+20Authority/emailAddress=
root@example.com SUBJECT
```

```
CERTSUBJECT:/C=US/ST=North+20Carolina/L=Raleigh/O=Example,
+20Inc./CN=stationY.example.com/emailAddress=root@example.com ✓
RELAY
```

(Seven characters in the actual issuer name need to be replaced in the `access` map rule with a + followed by the ASCII hex code: < > ( ) " + and **Space**. For example, spaces are replaced with +20. The `ascii(7)` man page can provide the appropriate codes.) In the example above, if a certificate signed by the `CERTISSUER` issuer name listed is presented to Sendmail, it will check the `access` map for a `CERTSUBJECT` rule that specifies the certificate's subject name. If that matches the `CERTSUBJECT` rule above, then relaying will be allowed for that host.

If you finish the other sequences early, generate an SSL certificate for Sendmail and attempt to configure STARTTLS relay authentication using the information above. You may configure relaying for any certificate signed by the issuer, or only for particular client certificates signed by the issuer.

## **Sequence 1 Solutions**

3.     Restart Sendmail to rebuild the /etc/mail/sendmail.cf file:

```
[root@stationX]# service sendmail restart
```

4.     Make sure that remote systems can access your mail server through your firewall:

```
[root@stationX]# iptables -A RHS333 -p tcp --dport 25 -j ACCEPT  
[root@stationX]# service iptables start
```

## **Unit 10**

### **Postfix**

**10-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Objectives

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

- Understand Postfix design principles and architecture
- Configure Postfix using postconf
- Configure basic Postfix security
- Use Connection filtering and access control
- Use Content filtering
- Secure email transmissions using TLS

10-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 784 3700.

## Vulnerabilities

- Monolithic design of Sendmail has led to security issues in the past
- Denial of Service attacks
- Need to control transmission of “spam”
- Virus protection

10-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Service Profile: Postfix

- Type:#System V-launched service
- Packages:postfix
- Daemons:master and sub-daemons
- Script:postfix
- Ports:25/tcp (smtp), 465/tcp (tls)
- Configuration:/etc/postfix/main.cf,  
/etc/postfix/master.cf /etc/postfix/\*
- Alternatives:sendmail

10-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2094 or +1 (919) 754 3700.

Postfix is an alternative MTA provided with Red Hat Enterprise Linux. Postfix has a reputation for being faster, easier to configure, less resource intensive and more secure than Sendmail.

| Service                     | Permissions     | Normally runs as                        |
|-----------------------------|-----------------|-----------------------------------------|
| /usr/libexec/postfix/master | 0755, root.root | user root, controls various sub-daemons |

The master daemon runs at service startup. It is the principal controller for the Postfix MTA, and calls the associated sub-daemons. It uses /etc/postfix/master.cf to control which sub-daemons it starts, and /etc/postfix/main.cf contains the options that control Postfix settings.

To enable Postfix, run system-switch-mail and select Postfix. /usr/sbin/sendmail is a symbolic link to /etc/alternatives/mta, which is a symbolic link to the /usr/sbin/sendmail.sendmail binary by default. On a system using Postfix as its MTA, this link will point to /usr/sbin/sendmail.postfix instead.

| File                   | Permissions       | Purpose                                              |
|------------------------|-------------------|------------------------------------------------------|
| /etc/postfix/main.cf   | 644, root.root    | Configuration file for the MTA                       |
| /etc/postfix/master.cf | 644, root.root    | Configuration file for the daemons (master and subs) |
| /etc/aliases           | 644, root.root    | For forwarding, source of /etc/aliases.db            |
| /etc/postfix/*         | 644, root.root    | Supplementary configuration files                    |
| /var/spool/postfix/*   | 700, postfix.root | Postfix queue directories                            |
| /var/spool/mail/       | 775, root.mail    | Spool directory for locally delivered e-mail         |

## Postfix Security Principles

- Exposed programs run at fixed low privilege
- Separation of processes provides insulation
- External processes are single-threaded
- No SUID processes
- Dynamic memory allocation to prevent buffer overruns
- Memory objects limited to prevent “wedging” under high load

10-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

A great deal of care was taken in the design and coding of Postfix to avoid the introduction of serious security flaws. This is what the designer and maintainer of Postfix, Wietse Venema, says about it:

“Postfix attempts to be fast, easy to administer, and secure, while at the same time being Sendmail compatible enough to not upset existing users. Thus, the outside has a Sendmail-ish flavour, but the inside is completely different.”

The ability to run at a fixed low privilege in a chrooted environment makes Postfix a very secure option for administrators looking for a fast, secure, and easily maintained MTA.

## Postfix Design

- master daemon controls the processes
  - 21 semi-resident cooperating sub-daemons
  - Runs sub-daemons on demand
- Processes communicate through UNIX domain sockets or FIFOs
  - Only *smtpd* and *smtp* are exposed to the net

10-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2894 or +1 (819) 754 3700.

Postfix runs as a *master* daemon. The master daemon controls twenty one cooperating sub-daemons that are described in more detail on the following slides. When the Postfix server is started, there are normally three daemons that run. The master daemon starts first running as root, followed by pickup and nqmgr that both run as user postfix.

```
$ ps aux|grep postfix
```

```
root      30362  0.0  0.3 25128 1596 ?          Ss    15:47 0:00 ↗
/usr/libexec/postfix/master
postfix   30367  0.0  0.3 25192 1544 ?          S     15:47 0:00 pickup -l -t fif
-u
postfix   30368  0.0  0.3 25244 1600 ?          S     15:47 0:00 qmgr -l -t fifo
```

Sub-daemon operations are configured in */etc/postfix/master.cf* and may use the transport types "inet" for Internet sockets, "unix" for UNIX-domain sockets, and "fifo" for named pipes. A typical *master.cf* follows:

| service type | private unpriv | chroot | wakeup | maxproc | command + args |                 |
|--------------|----------------|--------|--------|---------|----------------|-----------------|
| smtp         | inet           | n      | -      | -       | smtpd          |                 |
| pickup       | fifo           | n      | -      | 60      | pickup         |                 |
| cleanup      | unix           | n      | -      | -       | cleanup        |                 |
| qmgr         | fifo           | n      | -      | 300     | 1              | nqmgr           |
| rewrite      | unix           | -      | -      | -       | -              | trivial-rewrite |
| bounce       | unix           | -      | -      | -       | 0              | bounce          |
| defer        | unix           | -      | -      | -       | 0              | bounce          |
| flush        | unix           | n      | -      | 1000?   | 0              | flush           |
| proxymap     | unix           | -      | -      | -       | -              | proxymap        |
| smtp         | unix           | -      | -      | -       | -              | smtp            |
| relay        | unix           | -      | -      | -       | -              | smtp            |
| showq        | unix           | n      | -      | -       | -              | showq           |
| error        | unix           | -      | -      | -       | -              | error           |
| local        | unix           | -      | n      | -       | -              | local           |
| virtual      | unix           | -      | n      | -       | -              | virtual         |
| lmtp         | unix           | -      | -      | n       | -              | lmtp            |

**maildrop unix** -n -n - pipe

## /etc/postfix/master.cf

- The master.cf file configures which Postfix sub-daemons master will run
- Each line has 8 fields that represent settings for individual services
  - smtp inet n - n - - smptd
- The master.cf file only needs to be edited when you want to add sub-daemons or change the configuration of an existing one

10-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (816) 784 3700.

Each service defined in the master.cf file has an eight field definition. These fields (from left to right) are:

*Service:* Any name that is valid for the specified transport type (the next field). With INET transports, a service is specified as host:port. The host part (and colon) may be omitted. Either host or port may be given in symbolic form or in numeric form. Examples for the SMTP server: localhost:smtp receives mail via the loopback interface only; 10025 receives mail on port 10025.

*Transport type:* inet for Internet sockets, unix for UNIX-domain sockets, fifo for named pipes.

*Private:* Is access restricted to the mail system? Default is private service. Internet (inet) sockets can not be private.

*Unprivileged:* Whether the service runs with root privileges or as the owner of the Postfix system. Only the pipe, virtual and local delivery daemons require privileges.

*Chroot:* Whether or not the service runs chrooted to the mail queue directory.

*Wakeup time:* Automatically wake up the named service after the specified number of seconds. A ? at the end of the wakeup time field requests that wake up events be sent only to services that are actually being used. Specify 0 for no wakeup. Presently, only the pickup, nqmgr and flush daemons need a wakeup timer.

*Max procs:* The maximum number of processes that may execute this service simultaneously. Default is to use a globally configurable limit (the default\_process\_limit configuration parameter in main.cf). Specify 0 for no process count limit.

*Command + args:* The command to be executed. The command name is relative to the Postfix program directory (pathname is controlled by the daemon\_directory configuration variable). Adding one or more -v options turns on verbose logging for that service; adding a -D option enables symbolic debugging (see the debugger\_command variable in the main.cf configuration file). See individual command man pages for specific command-line options, if any.

## Receiving Mail

- Mail initially arrives in different ways
  - Local mail is posted by sendmail to the maildrop queue, and is received by pickup
  - Mail to port 25/tcp is received by smtpd
- Both pickup and smtpd pass mail to the cleanup daemon for processing
  - Basic checks, content filtering, header cleanup
  - Puts mail in *incoming* queue, notifies qmgr

10-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

## Delivering Mail

- qmgr moves messages being delivered from the *incoming* to the *active* queue
  - Messages that temporarily can not be delivered are moved to the *deferred* queue
- Mail may be delivered in different ways
  - local delivers messages to local files
  - smtp sends or relays messages to other hosts

10-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (888) 626 2904 or +1 (919) 784 3700.

## Queues

- Queues are in `/var/spool/postfix`
- Five primary mail queues:
  - *incoming* (for arriving mail)
  - *active* (mail being processed for delivery)
  - *deferred* (temporarily undeliverable mail)
  - *corrupt* (unreadable/damaged mail)
  - *hold* (mail kept on hold until taken care of)

10-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ccs.training@redhat.com](mailto:ccs.training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

There are five primary queues used by Postfix. All five are located in `/var/spool/postfix`. The `maildrop` queue is used for locally posted mail, the `incoming` queue is used for arriving mail, the `active` queue is used to hold mail that has been opened by qmgr for delivery, and the `deferred` queue is for mail that can not presently be delivered.

High performance mail servers require some additional thought to their initial construction. One thing the mail administrator should consider carefully is disk partitioning. For example, on a dedicated mail server that will be under a heavy load, you should separate the operating system from the queues and the logs. Optimum performance dictates the use of a fast SCSI subsystem and separate spindles for `/`, `/var/log`, `/var/spool/mail`, and `/var/spool/postfix`.

## Queues

- Messages placed in the *hold* queue are held indefinitely and delivery is not attempted
  - Access control or content filtering rules can move messages to *hold*
  - The **postsuper** command can release messages from *hold*
- Unreadable or damaged messages are placed in the *corrupt* queue

10-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cstraining@redhat.com](mailto:cstraining@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (819) 754 3708.

Besides the four primary mail queues, Postfix has additional auxiliary mail queues.

The *hold* queue is used for mail that is being held indefinitely by the mail server. Mail in the *hold* queue will not be delivered, but will not expire either. The **postsuper** command must be used to release mail being held, and the **postcat** command may be used to examine held mail. Mail can be moved to *hold* manually, or it can end up there due to access control or content filtering rules.

The *corrupt* queue is used to retain unreadable or damaged messages that Postfix can not process. These files must be inspected by a human system administrator.

The **postsuper** command (aka. the Postfix superintendent) is used by the superuser to manage Postfix queues.

The **postsuper** command is very powerful and should be used with care. The following command would delete all mail in the queues to or from user@example.com:

```
$ mailq | tail +2 | awk ~BEGIN { RS = ""} / user@example\.com$/ \  
{ print $1 } ~ | tr -d ^\#*\!` | postsuper -d -
```

For more detailed information on the **postsuper** command examine **man postsuper**.

## postconf

- **postconf** is the default configuration tool
  - Displays and edits /etc/postfix/main.cf
- Uses “plain English” configuration directives
- By default, displays all configuration settings
  - The -n option displays only non-defaults
  - The -d option displays default settings
- The -e option edits parameters in main.cf:
  - \$ postconf -e "myorigin = example.com"

10-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

**postconf** can be used to check and alter the values of postfix parameters in a similar way to the **sysctl** command.

For example:

To change the myorigin parameter:

```
$ postconf -e "myorigin = example.com"
```

To inspect the current value of the myorigin parameter:

```
$ postconf myorigin
```

To see the default value:

```
$ postconf -d myorigin
```

The /etc/postfix/main.cf file can also be edited manually. This file consists of simple directives, setting the values of postfix configuration parameters. If many configuration settings are to be changed, this may be an easier method than using **postconf -e** repeatedly.

Entries in /etc/postfix/main.cf can become very long, particularly when giving lists of arguments. To improve readability, postfix allows you to break up these commands over several lines. Any line that begins with whitespace is considered to be a continuation of the preceding line. An example:

```
smtpd_recipient_restrictions = permit_mynetworks,  
reject_unauth_destination, reject_rbl_client sbl.spamhaus.com
```

Can also be expressed as:

```
smtpd_recipient_restrictions = permit_mynetworks,
```

```
reject_unauth_destination,  
reject_rbl_client sbl.spamhaus.com
```

The commas are not required, but can improve readability.

## Basic Configuration Review

- Some parameters should be configured
  - `myhostname` is the host's fully qualified name
  - `myorigin` is the domain used on outgoing mail
  - `mydestination` lists domains for which the host will receive mail
  - `inet_interfaces` specifies the network interfaces on which Postfix listens for mail
  - `mynetworks` lists networks for which the host will relay mail

10-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

A few basic parameters should be set in `main.cf` when first setting up Postfix.

`myhostname` should normally be set to the server's fully qualified domain name. This is the default, and is used by a number of other basic configuration options.

`myorigin` should be set to the domain used in outgoing mail. It is set to `$myhostname` by default. If you are configuring `station2.example.com` and you want mail from `joe@station2.example.com` to appear to be coming from `joe@example.com`, you would set `myorigin = example.com`.

`mydestination` should be a space delimited list of domains for which the server will accept delivery. By default, it is set to `$myhostname` and `localhost`.

`inet_interfaces` specifies the network interfaces that Postfix will bind to. Like Sendmail, by default it will only accept connections to port 25 from `localhost`. Setting `inet_interfaces = all` will cause Postfix to listen to port 25 on all network interfaces. This will expose some slightly permissive default relaying permissions, which will need to be restricted with the `mynetworks_style` or `mynetworks` directives.

When the `inet_interfaces` value is altered, `postfix` must be restarted.

## Service Security

- E-mail to *root* is forwarded to user *postfix*
  - Modify */etc/aliases* to forward *root* mail to a system administrator
- Suppress *VRFY* information leakage
  - *disable\_vrfy\_command = yes*
- Force clients to start connection with *HELO*
  - *smtpd\_helo\_required = yes*

10-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 628 2984 or +1 (619) 784 3709.

One common security setting that new Postfix administrators run into trouble with is the handling of e-mail sent to the root account. Under Sendmail, it is not recommended that mail is read while logged in as root, but Sendmail *DOES* deliver mail to the root user. Under Postfix mail is *NOT EVER* delivered to *root* and there *MUST* be an alias in effect for the root user (*/etc/aliases*). Do not forget to run the *postalias* command after adding a new alias:

```
$ postalias /etc/aliases
```

Postfix also provides a **newaliases** which will accomplish the same task.

Note that under Red Hat Enterprise Linux 5 */etc/aliases* is provided in the setup rpm. This is a change as the aliases files were each provided separately by sendmail and postfix previously.

Postfix will respond to SMTP VRFY requests by default. This is sometimes used by spammers and attackers to collect information about valid e-mail addresses on the system. This setting can be suppressed:

```
$ postconf -e "disable_vrfy_command = yes"
```

It may also be sensible to require that clients communicating with the mail server send a SMTP HELO or EHLO to identify themselves first:

```
$ postconf -e "smtpd_helo_required = yes"
```

Another set of defaults that postfix administrators will typically use are as follows:

```
smtpd_recipient_restrictions = permit_mynetworks, reject
```

This lists the access controls applied when a host issues an SMTP RCPT TO: instruction and is most often used to control relaying through your postfix server. Note that some form of reject *MUST* be included or postfix will complain loudly!

## Postfix Security: DoS

- Delay messages if delivery rate is slower than arrival rate
  - `in_flow_delay`
- Limit local deliveries in parallel to one user
  - `local_destination_concurrency_limit`

10-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 625 2994 or +1 (919) 754 3700.

Denial of service attacks can be remarkably easy to carry out against MTAs. One of the most effective attacks is to repeatedly send large messages or messages with large attachments to an account on the targeted system. This fills up the mail spool directory and effectively prevents mail messages from reaching the other users of that system. It can sometimes be difficult to distinguish between a DDoS attack in progress and a normal but high volume of e-mail activity.

There are several parameters that can be set in `/etc/postfix/main.cf` to reduce the effectiveness of DoS attacks. These include:

`in_flow_delay = <number of seconds to delay before accepting messages if arriving faster than delivering>`

`local_destination_concurrency_limit = <number of messages delivered simultaneously to the same local recipient>`

## Restricting Relaying

- Relaying allowed for local subnets by default
  - Exposed if you set `inet_interfaces = all`
  - Default is `mynetworks_style = subnet`
  - Set `mynetworks_style = host`
  - Alternatively, set a restrictive `mynetworks`
- “Percent hack” relaying allowed by default
  - Set `allow_percent_hack = no`

10-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2964 or +1 (819) 754 3700.

As shipped by Red Hat, Postfix has `inet_interfaces = localhost` set, which only allows local processes to talk to the mail server. If `inet_interfaces = all` is set, then remote clients can contact the Postfix server. However, this also exposes some other default settings that will allow certain types of relaying to take place.

By default, Postfix has `mynetworks_style = subnet` set. This allows all hosts on subnets directly connected to your host to relay mail. Either set `mynetworks_style = host`, or set up a `mynetworks` directive restricting this to a specific list of hosts or networks:

```
mynetworks = 192.0.2.0/24 127.0.0.0/8
```

Postfix also honors the “percent hack” relaying syntax by default. This was once used in order to forward mail through a mail gateway to a host the mail server can resolve or reach that is unreachable from the client. The basic syntax is:

```
user%kremvax.bitnet@mail.example.com
```

This message would be sent to `mail.example.com`, which would relay it to `user@kremvax.bitnet`. This setting can also be abused by spammers, so set `allow_percent_hack = no` in your `main.cf` file.

## /etc/postfix/access

- Similar to the Sendmail access file

|                 |        |
|-----------------|--------|
| spammer@aol.com | REJECT |
| spamRus.net     | REJECT |
| 10.3            | OK     |

- May be applied multiple times (main.cf)

- `check_helo_access`, `check_sender_access`, `check_client_access`, `check_recipient_access`

10-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 784 3700.

The /etc/postfix/access file has a very similar syntax to the /etc/mail/access file used by Sendmail. It can reject mail matching a user@domain e-mail address, a domain, or an entire IP subnet. In the second column, values that may be used include

|        |                                                                   |
|--------|-------------------------------------------------------------------|
| REJECT | Rejects the message with a general purpose message.               |
| OK     | Accepts the message for receipt.                                  |
| [45]xx | Sends an arbitrary 4xx (temporary) or 5xx (permanent) error code. |
| HOLD   | Sends the message to the holdqueue.                               |

After adding entries to /etc/postfix/access, you must use the postmap command to rebuild the access map:

```
$ postmap /etc/postfix/access
```

The map can be used by several different categories of checks if configured properly:

```
smtpd_recipient_restrictions = permit_mynetworks,
                                reject_unauth_destination,
                                check_sender_access hash:/etc/postfix/access,
                                check_client_access hash:/etc/postfix/access,
                                check_helo_access hash:/etc/postfix/access,
                                check_recipient_access hash:/etc/postfix/access,
                                reject_unauth_pipelining
```

## DNS Blackhole Lists

- DNS can be used as a database to store information about sources of “spam”
  - reject\_rbl\_client sbl-xbl.spamhaus.org
- DNS lookup using reversed octets of sender IP address, with the domain of the blackhole list appended
- Can implement locally or use public or subscription-based blackhole services

10-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[tca@us.redhat.com](mailto:tca@us.redhat.com)> or phone toll-free (USA) +1 (908) 629 2994 or +1 (919) 754 2700.

Other mechanisms for controlling spam include the use of the Postfix `reject_rbl_client` configuration as well as procmail and spamassassin. To configure Postfix to use `reject_rbl_client` add entries like the following to `/etc/postfix/main.cf`:

```
smtpd_recipient_restrictions = reject_rbl_client cn-kr.blackholes.us,  
                                reject_rbl_client singapore.blackholes.us,  
                                reject_rbl_client malaysia.blackholes.us,  
                                reject_rbl_client nigeria.blackholes.us,  
                                reject_rbl_client bl.spamcop.net,  
                                reject_rbl_client relays.ordb.org,  
                                reject_rbl_client proxies.relays.monkeys.com,  
                                reject_rbl_client sbl.spamhaus.org
```

## Procmail and SpamAssassin

- Postfix does not use Procmail by default
  - `mailbox_command = /usr/bin/procmail`
- Procmail may start SpamAssassin
- An alternative is to have `master` call `spamc` for mail received on external interfaces
  - Mail sent from the localhost is not filtered
  - Then `spamc` can reinject the e-mail into Postfix by calling local `sendmail.postfix`

10-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

By default, Postfix does not use Procmail as a MDA. However, it can be configured to use Procmail as a local mailbox delivery agent by setting the `mailbox_command` parameter. Then user `~/.procmailrc` files will be honored by default, as will the system-wide `/etc/procmailrc` file.

Procmail can be used to call SpamAssassin as with Sendmail. As an alternative, `spamd` can be started at boot and Postfix can call `spamc` automatically for all mail received on external interfaces. In this configuration, Postfix does not process mail sent by localhost through `spamc`, so once incoming mail is processed, `spamc` resends the e-mail by calling the local Postfix `sendmail` program. This requires some simple changes to `main.cf` and `master.cf`, and a basic tutorial on the subject is available at:

<http://aaron.boim.com/unix/postfix+spamassassin.html>

## Content Filtering

- Incoming messages can be filtered before receipt is acknowledged
  - header\_checks
  - mime\_header\_checks
  - body\_checks
  - body\_checks\_max\_size

10-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2884 or +1 (919) 754 3700.

Postfix provides some unique capabilities to filter messages *prior to receipt*. These include the ability to discriminate based upon content in mail headers, bodies, and MIME types. This reduces the waste of processing power and storage space caused by filtering out mail after it has been received.

Here is a sample header\_checks configuration:

```
header_checks = regexp:/etc/postfix/header_checks
```

In the /etc/postfix/header\_checks file there might be:

```
/^X-Mailer: SuperMail-2/ REJECT Bad mailer!
```

In this case a match on this X-Mailer: mail header will cause the mail to be returned.

The body\_checks setting works in the same way as header\_checks:

```
body_checks = regexp:/etc/postfix/body_checks
```

In the /etc/postfix/body\_checks file there might be:

```
/Viagra Is Now Available/ REJECT Go away!
```

With this configuration, Postfix will block the receipt of any e-mail that contains the words “*Viagra Is Now Available*” in the message body. The body\_checks\_max\_size parameter will specify the maximum amount of body text that body\_checks will process in a given e-mail message.

Attachments are checked in mime\_header\_checks. For example:

```
mime_header_checks = regexp:/etc/postfix/mime_header_checks
```

In /etc/postfix/mime\_header\_checks there might be:

```
/name=[^>]*.\(ade|adp|asd|bas|bat|chm|cmd|com|cpl|crt|dbx|dll|exe|hlp|hta|\\inf|ins|isp|lnk|mde|mdt|vb|vbe|vbs)/ REJECT attachment prohibited
```

## Postfix with SASL/TLS

- Simple Authentication and Security Layer
- Transport Layer Security
- Configuring these two services requires changes to /etc/postfix/main.cf and
- /etc/postfix/master.cf

10-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

SASL, the Simple Authentication and Security Layer, is a method for adding authentication support to connection-based protocols. Red Hat Enterprise Linux 5 uses SASL2 that includes saslauthd (a stand alone authentication daemon).

The TLS protocol is defined in RFC 2246 where it says:

*"The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery."*

Postfix may be configured to use SASL/TLS by adding some additional configuration directives to /etc/postfix/main.cf and /etc/postfix/master.cf. In addition a small change must be made to /usr/lib/sasl2/smtpd.conf.

## Configuring SASL/TLS

- create ssl CA cert and local certificate
- edit /etc/postfix/main.cf
- edit /etc/postfix/master.cf
- edit /usr/lib/sasl2/smtpd.conf
- touch /etc/sasldb2
- chown postfix /etc/sasldb2

10-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3760.

SASL/TLS requires the creation of a CA cert and a local certificate. Instructions on how to do this properly are included in the lab that follows this chapter as well as in the solution script included in this course. Once you have SASL/TLS working you may test it like this:

```
$ telnet server1.moongroup.com 25

Trying 204.157.7.157...
Connected to server1.moongroup.com (204.157.7.157).
Escape character is '^]'.
220 server1.moongroup.com ESMTP (postfix rel. 2.2/RHEL4) MOONGROUP.COM uses SPF!
SPF!

ehlo stealth

250-server1.moongroup.com
250-PIPELINING
250-SIZE 10240000
250-ETRN

250-STARTTLS <---- shows TLS active

250-AUTH GSSAPI PLAIN LOGIN DIGEST-MD5 CRAM-MD5
250-AUTH=GSSAPI PLAIN LOGIN DIGEST-MD5 CRAM-MD5
250 8BITMIME
```

## End of Unit 10

- Questions and Answers
- Summary
  - Postfix design principles
  - Service security
  - Relaying and access restrictions
  - Content filtering
  - TLS

10-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2694 or +1 (919) 754 3700.

## Lab 10

### Postfix

---

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Goal:</b>               | Configure Postfix to block selected content and restrict relaying.                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Estimated Duration:</b> | 60 minutes                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>System Setup:</b>       | <p>Ensure that the <code>postfix</code>, <code>system-switch-mail</code>, and <code>system-switch-mail-gnome</code> packages are installed.</p> <p>If you need to install them, use <code>yum</code> as described in the previous labs.</p> <p>Run the <code>system-switch-mail</code> and select postfix as your default mta.</p> <p>You should also ensure that the <code>smtp</code> port on your external interface is not blocked by your system firewall.</p> |
| <b>Situation:</b>          | You have been asked to implement a Postfix SMTP server which will send and receive mail for your domain. You will need to apply restrictions to ensure your postfix server is not available as a relay to unknown hosts and to reject email which meets certain criteria.                                                                                                                                                                                           |

## Sequence 1: Inspecting and tuning the default postfix configuration

**Scenario:** Postfix is a very secure and fast MTA. In this lab we will enable postfix as our system's default MTA and tweak its default configuration to improve its security.

**Instructions:**

1. First, inspect the Red Hat default postfix configuration, using the **postconf** tool.

```
[root@stationX ~]# postconf -n
```

This should generate output similar to the following:

```
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases
command_directory = /usr/sbin
config_directory = /etc/postfix
daemon_directory = /usr/libexec/postfix
debug_peer_level = 2
html_directory = no
inet_interfaces = localhost
mail_owner = postfix
mailq_path = /usr/bin/mailq.postfix
manpage_directory = /usr/share/man
mydestination = $myhostname, localhost.$mydomain, localhost
newaliases_path = /usr/bin/newaliases.postfix
queue_directory = /var/spool/postfix
readme_directory = /usr/share/doc/postfix-2.3.3/README_FILES
sample_directory = /usr/share/doc/postfix-2.3.3/samples
sendmail_path = /usr/sbin/sendmail.postfix
setgid_group = postdrop
unknown_local_recipient_reject_code = 550
```

This listing represents the current *non-default* settings for your postfix MTA. There are, however, many default settings that are not displayed here. Rather than have every single configuration option written into its configuration file, the majority of settings for postfix are given default values and the configuration file is used to override these defaults.

Compare this output to the content of the main postfix configuration file:  
`/etc/postfix/main.cf`. Note that this file is very well commented and contains many examples of common parameters, including the above output. There are several unused parameters in here which can be easily enabled by uncommenting a few lines and some minor editing.

2. To see *all* the current configuration parameters for postfix, use the **postconf** tool without specifying any options:

```
[root@stationX ~]# postconf
```

This will produce over 500 lines of output, some of which looks like this:

```
2bounce_notice_recipient = postmaster
access_map_reject_code = 554
address_verify_default_transport = $default_transport
address_verify_local_transport = $local_transport
address_verify_map =
address_verify_negative_cache = yes
address_verify_negative_expire_time = 3d
address_verify_negative_refresh_time = 3h
address_verify_poll_count = 3
address_verify_poll_delay = 3s
...other output omitted...
virtual_gid_maps =
virtual_mailbox_base =
virtual_mailbox_domains = $virtual_mailbox_maps
virtual_mailbox_limit = 51200000
virtual_mailbox_lock = fcntl
virtual_mailbox_maps =
virtual_minimum_uid = 100
virtual_transport = virtual
virtual_uid_maps =
```

The **postconf** command works in a similar fashion to the linux kernel's **sysctl** command - it can be used to directly edit the configuration of a running postfix server, or simply display the current values of some or all parameters. Most alterations to **main.cf** do not require a restart of the postfix service.

A few examples:

Retrieve the current content of the **mynetworks** parameter:

```
[root@stationX ~]# postconf mynetworks
```

alter the **mynetworks** parameter on the fly:

```
[root@stationX ~]# postconf -e "mynetworks = 127.0.0.1/32"
```

Retrieve the default content of the **mynetworks** parameter:

```
[root@stationX ~]# postconf -d mynetworks
```

For more information about the many uses of **postconf**, see

```
[root@stationX ~]# man postconf
```

For very detailed documentation on the majority of postfix configuration parameters, see:

```
[root@stationX ~]# man 5 postconf
```

### 3. A note on configuration file syntax

Many of postfix's configuration parameters can contain lists of definitions. These can be

Whitespace-delimited:

```
mynetworks = 127.0.0.0/8 192.168.0.0/24
```

Comma-delimited:

```
mynetworks = 127.0.0.0/8, 192.168.0.0/24
```

Or spread across multiple lines as follows.

```
mynetworks = 127.0.0.0/8  
           192.168.0.0/24
```

The only restriction on multiple-line entries is that any continuation lines must be indented to be considered part of the same definition. This can be very helpful in keeping your configuration file readable - some postfix paramters can result in very long lines.

You can also refer to previous paramaters in a very similar way to bash shell variables, for example:

```
myhostname = station4.example.com
```

```
myorigin = $myhostname
```

which can simplify your configuration considerably.

## **Sequence 2: Tuning the default configuration**

**Scenario:**

In this lab sequence we will be making quite a few changes to our existing postfix configuration in order to improve its security on several fronts. We will implement access controls, restrict relaying and set up some basic content filters.

Changing the parameters involved in this lab can be achieved using one of two methods:

1. Editing the `/etc/postfix/main.cf` file directly
2. using the `postconf -e` command

To complete the lab, you may be required to refer back to the postfix unit (or possibly other units on the course).

Remember that you will need to reload or restart postfix after any changes to `/etc/postfix/main.cf`

**Deliverable:**

An externally accessible postfix MTA with improved security.

**Instructions:**

1. Configure Postfix to accept incoming mail on all local interfaces.
2. Enable the use of `procmail` for local email processing
3. Reduce information leakage by disabling the `vrify` command.
4. Change the default SMTP banner to reveal less information.

By default, postfix betrays a lot of information about itself to connecting clients, including which MTA we are running and in which version. To be RFC compliant we are only required to provide a hostname and the ESMTP prompt. Customise your banner to contain only that information.

5. Test your new configuration

Connect to your `smtp` port using `telnet` and issue a `vrify` command. Check that the greeting has changed and that `vrify` commands no longer work.

## **Sequence 3: Restricting Incoming Email**

**Scenario:** Now that postfix is accessible from the outside and we have reduced the amount of information it volunteers to connecting clients, we wish to control which email we are willing to accept.

**Deliverable:** A postfix server with restrictions on the email it accepts

**Instructions:**

1. Configure your postfix server to send and receive#mail for domainX.example.com, as well as your hostname and localhost.
2. Configure your postfix server to rewrite outgoing email to appear to come from domainX.example.com.
3. Enable header and body checking in your postfix configuration.
4. Configure postfix such that mail with a 'Subject:' containing the string "Email lottery" is rejected with the response "bye, bye spam!"
5. Mail attachments with the extensions .doc ,xls and .ppt are rejected with the message "open formats only, please"
6. Mail whose body text contains the phrase "Bad Credit Can Cost You" should be rejected with the response "no credit spam permitted."
7. Test your configuration using a mail client of your choice.

## **Sequence 4: Restricting relaying**

**Scenario:** Currently any host on your local networks can use your postfix server as a relay. We are going to place restrictions on who can do this and close a few more security loopholes.

**Deliverable:** A postfix mailserver that allows only selected hosts to use it as a relay

**Instructions:**

1. Restrict relaying on your postfix server to localhost, your hostname and one other host in the local network.
2. Force connecting hosts to issue a correct HELO or EHLO before sending any other commands:
3. As other external SMTP servers may also have this restriction in place, ensure that the postfix `smtp` daemon always sends a correct EHLO greeting.

## **Sequence 5: Setting up TLS and SASL authentication**

**Scenario:** You have decided that users who relay through your postfix server will only be permitted to do so if they provide a valid username and password. To protect this data you will need to configure TLS encryption for your mailserver.

**Deliverable:** A postfix server that offers STARTTLS on client connection and uses SASL for user authentication

**Instructions:**

1. Using the CA you configured in lab3, create a private key and certificate for use with postfix.
2. Call your private key `postfix.key` and your server certificate `postfix.crt`.
3. Create the directory `/etc/postfix/certs` Copy your server certificate, private key and CA certificate into this new directory.
4. Add the following entries to your `/etc/postfix/main.cf` file.

```
smtpd_use_tls = yes  
smtpd_tls_key_file = /etc/postfix/certs/postfix.key  
smtpd_tls_cert_file = /etc/postfix/certs/postfix.crt  
smtpd_tls_CAfile = /etc/postfix/certs/my-ca.crt  
smtpd_tls_loglevel = 1
```

5. finally, restart postfix

```
[root@stationX ~]# service postfix restart
```

6. Now test that postfix offers STARTTLS to connecting clients.

This is easily accomplished using telnet. If TLS is correctly configured, you should see the STARTTLS line in the output below:

```
[root@stationX ~]# telnet stationX.example.com 25
Trying 192.168.22.220...
Connected to stationX.example.com (192.168.22.220).
Escape character is '^].
220 stationX.example.com ESMTP
ehlo stationX
250-stationX.example.com
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

7. Now that we have TLS working, we should configure authentication using **saslauthd**

This will use the **saslauthd** service to check users and passwords. This is already configured to use pam on our system, so all local users will be able to use their login passwords to send email. Should you wish to alter this, to use a different authentication mechanism, you should edit the /etc/sysconfig/saslauthd file.

As this is already configured, all we need do is ensure that the necessary service is running and will start at next boot.

```
[root@stationX ~]# service saslauthd start
[root@stationX ~]# chkconfig saslauthd on
```

8. Now we need to add the configuration options to postfix that will enable the use of saslauthd.

Add the following lines to /etc/postfix/main.cf

```
smtpd_sasl_auth_enable = yes
broken_sasl_auth_clients = yes
smtpd_sasl_security_options = noanonymous
```

- 9.** Finally, we need to ensure that postfix allows SASL authenticated users to send email.

Relaying restrictions are controlled by the `smtpd_recipient_restrictions` parameter, which lists the checks that should be made when clients issue an SMTP RCPT TO : command.

Add this to your main.cf file:

```
smtpd_recipient_restrictions = permit_mynetworks,  
                                permit_sasl_authenticated,  
                                reject_unauth_destination
```

- 10.** Test your email server.

Send email to and from your system with attachments and suspicious text.

## Sequence 2 Solutions

1. Configure Postfix to accept incoming mail on all local interfaces.

By default, postfix (as with many other network services) will only accept incoming connections on the local loopback address. You can confirm this using netstat:

```
[root@stationX ~]# netstat -tlpn | grep ':25'
```

This is controlled by the `inet_interfaces` parameter in `/etc/postfix/main.cf`. Alter this (using either `postconf` or the editor of your choice) as follows:

```
inet_interfaces = all
```

2. Enable the use of `procmail` for local email processing

For this we need to set the `mailbox_command` parameter:

```
mailbox_command = /usr/bin/procmail
```

3. Reduce information leakage by disabling the `vrfy` command.

The SMTP `vrfy` command allows connecting hosts to check whether a username is a valid local recipient for email. Clearly this exposes information that we would prefer to keep private, so let's turn it off:

```
disable_vrfy_command = yes
```

4. Change the default SMTP banner to reveal less information.

By default, postfix betrays a lot of information about itself to connecting clients, including which MTA we are running and in which version. To be RFC compliant we are only required to provide a hostname and the ESMTP prompt. Customise your banner to contain only that information.

Add the following to `/etc/postfix/main.cf`

```
smtpd_banner = $myhostname ESMTP
```

5. Test your new configuration

Connect to your smtp port using telnet and issue a `vrfy` command. Check that the greeting has changed and that `vrfy` commands no longer work.

```
[root@stationX ~]# telnet stationX 25
```

You should be greeted with your customised prompt:

```
[root@localhost ~]# telnet stationX 25
Trying 192.168.0.X...
Connected to stationX.example.com (192.168.0.X).
Escape character is '^].
220 stationX.example.com ESMTP
```

**Attempt to verify an existing username:**

```
Connected to stationX.example.com
(192.168.0.X).
Escape character is '^]'.
220 stationX.example.com ESMTP

vrfy username
502 5.5.1 VRFY command is disabled
```

## Sequence 3 Solutions

1. Configure your postfix server to send and receive#mail for domainX.example.com, as well as your hostname and localhost.

```
mydestination = $myhostname, localhost.$mydomain,  
localhost.localdomain, localhost
```

2. Configure your postfix server to rewrite outgoing email to appear to come from domainX.example.com.

```
myorigin = domainX.example.com
```

3. Enable header and body checking in your postfix configuration.

- a. Add these settings to /etc/postfix/main.cf

```
header_checks = regexp:/etc/postfix/header_checks  
body_checks = regexp:/etc/postfix/body_checks  
mime_header_checks = regexp:/etc/postfix/mime_header_checks
```

- b. You will need to create these files before reloading postfix:

```
[root@stationX ~]# touch ✓  
/etc/postfix/{header,body,mime_header}_checks
```

4. Configure postfix such that mail with a 'Subject:' containing the string "Email lottery" is rejected with the response "bye, bye spam!"

Add the following line to /etc/postfix/header\_checks:

```
/^Subject.*Email lottery/ REJECT bye, bye spam!
```

5. Mail attachments with the extensions .doc .xls and .ppt are rejected with the message "open formats only, please"

Add this line to /etc/postfix/mime\_header\_checks:

```
/^name[^>] (.doc|.xls|.ppt)$/ REJECT open formats only, ✓  
please.
```

6. Mail whose body text contains the phrase "Bad Credit Can Cost You" should be rejected with the response "no credit spam permitted."

Add this line to /etc/postfix/body\_checks

```
/Bad Credit Can Cost You/ REJECT no credit spam permitted
```

7. Test your configuration using a mail client of your choice.

Send emails matching the patterns in your header and body\_checks files to a valid user on your system.

to test attachments, create an empty file that matches a banned pattern, thus:

```
[root@stationX ~]# touch /tmp/myfile.doc
```

Now send that file as an attachment to a valid user on your system

```
[root@stationX ~]# echo test email | mutt username@stationX.example.com -s "attachment test" -a /tmp/myfile.doc
```

## Sequence 4 Solutions

1. Restrict relaying on your postfix server to localhost, your hostname and one other host in the local network.

By default the hosts that can use your postfix server as a relay are defined using either `mynetworks_style` or `mynetworks`. As `mynetworks` takes precedence we should define this quite strictly.

```
mynetworks = 127.0.0.1/32, 192.168.0.X, 192.168.0.Y
```

2. Force connecting hosts to issue a correct HELO or EHLO before sending any other commands:

Add this line to your configuration file:

```
smtpd_helo_required = yes
```

3. As other external SMTP servers may also have this restriction in place, ensure that the postfix `smtp` daemon always sends a correct EHLO greeting.

Add this line to your configuration file:

```
smtp_always_send_ehlo = yes
```

# **Unit 11**

## **FTP**

**11-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2004 or +1 (919) 754 3700.

## Objectives

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

- Available FTP server
- Limiting exposed server information
- Configuration of logging
- Types of users and user classes
- Restricting access by host or user

11-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[crailling@redhat.com](mailto:crailling@redhat.com)> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

## Vulnerabilities

- Weak authentication
- Easily hijacked
- Plain text passwords and data transfers

11-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 764 3700.

## Resolutions

- Only use *ftp* when other options are not feasible. Consider *sftp* or *scp*.
- Restrict uploads for anonymous users.
- Restrict local users to home directories.
- Hide the server's version and other sensitive service information.

11-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2804 or +1 (919) 754 3700.

## The FTP Protocol

- Two channels, *command* and *data*
- Client contacts server on port 21 to open command channel
- PORT: Server opens data channel to client
  - Server connects from port 20 to client's port
- PASV: Client opens data channel to server
  - Server specifies which port
  - Works better through some firewalls

11-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The FTP file transfer protocol was defined in RFC959. Unlike most protocols, FTP uses two separate communication channels, a command channel and a data channel. This makes the service hard to protect with firewalls or encrypted tunnels. The client contacts the server on port 21 to open the command channel. The commands sent to port 21 are clear text commands; telnet server 21 can be used to open a command channel to an FTP server for troubleshooting.

After authentication, either the PORT or PASV command is sent by the client each time it needs to open the data channel. Both modes specify the connection port through a comma delimited, six-element string, where each element is a sixteen bit number ranging from 0 to 255. The string 192,168,0,5,156,3 indicates a connection to 192.168.0.5 port 39939 ((256 \* 156) + 3).

In an active connection, the client will issue a PORT command that specifies the port the server should connect to to deliver requested information. If the client sent the protocol commands:

```
PORT 192,168,0,1,207,185
200 PORT command successful.
LIST
```

the server would attempt to connect to 192.168.0.1 port 53177 (normally, from port 20 on the server) and send the directory listing of the current directory.

Alternatively, a client may issue a PASV command to request a data channel:

```
PASV
227 Entering Passive Mode (192,168,0,7,107,20)
```

The server's response tells the client that it should connect to 192.168.0.7 port 27412 to open the data channel before it sends the LIST command.

## FTP Servers

- vsftpd
  - “Very Secure” and fast FTP server
- Some alternatives are also shipped
  - gssftp (in `krb5-workstation`)
  - tux

11-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 764 3700.

In the following sections, several topics on how to configure the standard FTP server in Red Hat Enterprise Linux will be looked at. For each topic, configuration will be discussed with security related settings for vsftpd in mind. Unless stated otherwise, configuration settings for vsftpd should be made in `/etc/vsftpd/vsftpd.conf`.

There are also other ftp servers available in the distribution. The `krb5-workstation` package includes a Kerberized ftp server, and the `tux` kernel-based web server has support for anonymous ftp service as well.

## Service Profile: vsftpd

- **Type:**#System V-launched daemon
- **Packages:**#vsftpd
- **Daemons:**#vsftpd
- **Ports:**#21/tcp (ftp), 20/tcp (ftp-data)
- **Configuration:**#/etc/vsftpd/vsftpd.conf#/etc/vsftpd/ftpusers /etc/pam.d/vsftpd
- **Conflicts:**#Kerberos ftpd (gsftpd)

11-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc., or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

The “vs” in vsftpd stands for “very secure”. This is meant to indicate that security was kept firmly in mind throughout the design of this FTP server. Furthermore, testing has shown that vsftpd is also very fast and scales to large numbers of connections well. It is currently used by a number of large, high-profile FTP sites, and is recommended by SANS.

| Service                 | Permissions        | Normally runs as                     |
|-------------------------|--------------------|--------------------------------------|
| /usr/sbin/vsftpd        | 755, root.root     | user <i>root</i> , group <i>root</i> |
| <b>File</b>             | <b>Permissions</b> | <b>Purpose</b>                       |
| /etc/vsftpd/vsftpd.conf | 600, root.root     | Main configuration file              |
| /etc/vsftpd/ftpusers    | 600, root.root     | List of users denied access          |
| /etc/pam.d/vsftpd       | 644, root.root     | PAM service configuration file       |
| /var/ftp                | 755, root.root     | Anonymous FTP directory              |

## Login Banners

- Banner provides information before login
- Set security warning banners
  - `banner_file=filename`
- Suppress server and version information
  - `ftpd_banner=FTP server ready.`
  - `banner_file` overrides this option

11-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (916) 754 3700.

An attacker may attempt to determine the version of the operating system or the service daemons running on the system. The default greeting may provide the attacker with potentially useful information about the FTP server.

`220 (vsFTPD 1.2.0)`

This one-line greeting can be suppressed by setting `ftpd_banner=Some Text` in `/etc/vsftpd/vsftpd.conf`.

`220 FTP server ready.`

If `banner_file=filename` is set, the `ftpd_banner` option is ignored and the contents of the file `filename` are displayed when someone connects to the server.

## Informational Capabilities

- Display file when client enters directory
  - `message_file=.message`
  - `dirmessage_enable=YES`

11-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

Sometimes, it is nice to be able to automatically display a message to a FTP client when it changes to a particular directory on the server. vsftpd can be set up to display a file called `.message` (if it exists) when a user changes into the directory containing the file.

For vsftpd, `dirmessage_enable=YES` turns on message display on cd, and `message_file=.message` sets the name of the file to display automatically to `.message` (the default).

## Logging Capabilities

- To log all uploads and downloads:
  - `xferlog_enable=YES`
- To log all FTP commands:
  - `log_ftp_protocol=YES`
  - `xferlog_std_format=NO`

11-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2864 or +1 (919) 784 3700.

Most FTP sites keep a log of all file transfers to and from the server. By default, vsftpd logs file transfers to `/var/log/vsftpd.log`. The lines that enable this are

`xferlog_enable=YES` - enable logging

`xferlog_std_format=YES` - optional; use wu-ftpd's log format

Sometimes sites keep a log of all FTP protocol commands issued. For vsftpd, all commands can be logged to `/var/log/vsftpd.log` with the option

`log_ftp_protocol=YES`

but then also set `xferlog_std_format=NO`, vsftpd can also log file transfers to both `/var/log/xferlog` and `/var/log/vsftpd.log` with the option `dual_log_enable=YES`.

## Local Users

- Users with local accounts may log in using their username and password
- Local users start in their home directory
  - By default, does not *chroot* users
  - `chroot_local_user=YES`
- Have read-write access by default

11-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2094 or +1 (918) 754 3700.

Users who have a regular interactive account can use FTP to connect to their own home directory. By default, vsftpd does not chroot them. To make vsftpd chroot users into their home directory, set the option `chroot_local_user=YES` in `/etc/vsftpd/vsftpd.conf`.

The ability of local users to download and upload files to the server's filesystem is otherwise restricted by normal UNIX file permissions; they have read-write access by default. To turn off write access:

```
write_enable=NO
```

Local user access can also be turned off entirely (useful if you are setting up an anonymous FTP only server):

```
local_enable=NO
```

## User/Group Access Control

- Can control access by user with a file
  - /etc/vsftpd/ftpusers
- Can also set up access control by group
  - Use pam\_listfile.so with item=group

11-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2694 or +1 (819) 754 3700.

vsftpd supports files that contain lists of users, one username per line, that should not be allowed access to the FTP server. For security reasons, root is one of these users by default. This functionality is actually implemented by PAM, and can be used to create a file that contains a list of groups that should not be allowed access to the FTP server.

The PAM configuration file for vsftpd is `/etc/pam.d/vsftpd` by default. If `userlist_enable=YES`, vsftpd references the file defined by `userlist_file` (`/etc/vsftpd/ftpusers` by default). By default, users listed in this file are denied login before being prompted for a password. If `userlist_deny=NO` is set, then all users except those listed in this file are denied login. To enable a prohibited groups file, add the following text as the first auth line in `/etc/pam.d/vsftpd`.

```
auth required /lib/security/pam_listfile.so item=group sense=deny \
file=/etc/ftpgroups onerr=succeed
```

## Anonymous FTP

- Anonymous user can login by default
  - `vsftpd` sets up the `/var/ftp` directory
  - Can login as user anonymous or as `ftp`
- Chroots to `/var/ftp`
- Has read-only access by default

11-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ttraining@redhat.com](mailto:ttraining@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The `vsftpd` server allows the anonymous FTP user to login by default if `/var/ftp` exists. The anonymous user is chrooted into `/var/ftp` by default and cannot see the rest of the files on the server.

The `vsftpd` RPM will set up `/var/ftp` with reasonable permissions for you. None of the files or directories under `/var/ftp` are owned or writable by user or group `ftp` by default, so the anonymous user has read-only access.

If the anonymous FTP user is allowed to upload files, it is very important to carefully control the upload directories. Insecure FTP sites are commonly used to exchange stolen software or data and otherwise abused. Any data uploaded to the site should be carefully examined by a trusted administrator before being made available for download.

Anonymous access to the FTP server can be explicitly prohibited:

```
anonymous_enable=NO
```

## Anonymous FTP Uploading

- Uploading of files by anonymous user should be carefully controlled
- Set permissions and **umask** to only allow uploads to the upload directory
- Disable use of sensitive commands
- Examine files before allowing others to download

11-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 784 3700.

In this example, `/var/ftp/incoming` will be our upload directory. It should be set up with the permissions:

```
drwx-wx---    1 root      ftp        4096 Jun  6 13:41 /var/ftp/incoming
```

This allows the anonymous user to write files into the directory, but not list the contents of the directory. When the files are uploaded, make sure that they are not readable by user ftp; change owner to root and make sure file permissions are set to 0600. Neither user ftp nor group ftp should own any other directories or files on the FTP site!

```
anon_upload_enable=YES  
anon_umask=077  
chown_uploads=YES  
chown_username=root
```

## Connection Restrictions

- `max_clients` limits the number of clients that may be connected
- `max_per_ip` limits the number of clients that may be connected from one IP address

11-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2864 or +1 (919) 754 3760.

One denial-of-service attack on an FTP server can be for a single host to open many simultaneous connections to the service at the same time. Some FTP clients also can use this in an attempt to bypass bandwidth limits you may set up.

Some useful directives:

`max_clients=250` - 250 clients can be active simultaneously  
`max_per_ip=10` - One IP address can have 10 concurrent connections

## Host Access Restrictions

- Block access from certain IP addresses
- Using TCP Wrappers
  - vsftpd uses vsftpd in first field
  - Use twist option to display a message file

11-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2804 or +1 (919) 754 3700.

A good way to block access to the FTP server based on IP address is with TCP Wrappers. It is possible to output a message file by using the twist statement. In that case, every line of the file that is output should start with 530- so it is displayed by the FTP client. To enable TCP wrappers support, set `tcp_wrappers=YES`.

*Example (in /etc/hosts.deny):*

```
vsftpd:192.168.1.0/255.255.255.0:twist /bin/cat /var/ftp/.denied
```

denies access to hosts on the 192.168.1.0/24 network, displaying to those hosts the file `/var/ftp/.denied`.

## Other Useful Options

- `hide_ids` tells the client that all files are owned by user `ftp`, group `ftp`
- Limits can be placed on transfer rate
  - `anon_max_rate`
  - `local_max_rate`
- `user_config_dir` allows per-user configuration settings

11-17

For use only by a student enrolled in a Red Hat Training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2904 or +1 (919) 754 3700.

If you set `hide_ids=YES`, the client will be told that all files are owned by user `ftp` and group `ftp`, even if they are not.

Limits can be placed on the transfer rate for anonymous and local users. Use `anon_max_rate` to set the maximum rate in bytes per second for anonymous FTP connections; use `local_max_rate` for locally authenticated users.

It is possible to set configuration options on a per-user basis. If `user_config_dir` is defined, when a user logs in `vsftpd` will look for a configuration file with the same name as the user in that directory. For example, if

```
user_config_dir=/etc/vsftpd_user_conf
```

then when the user Bob tries to log in, `vsftpd` will use the settings in `/etc/vsftpd_user_conf/bob` for that session.

## **End of Unit 11**

- Questions and Answers
- Summary
  - Overview of operation
  - Restricting information
  - Restricting users
  - Restricting access

**11-18**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

## **Lab 11**

### **FTP**

---

**Goal:** This lab will explore some of the features of the **vsftpd** server. In particular, you will set up a **guest** user.

**Estimated Duration:** 45 minutes

**System Setup:** This lab requires that the **vsftpd**, **lftp** and **nc** packages are installed.

## Sequence 1: Active vs. Passive FTP

**Scenario:** Clients may converse with an FTP server using either active or passive mode. In active mode, the client uses the **PORT** command to tell the server the port on which the client will be listening. In passive mode, the client uses the **PASV** command to request a port to which the client will connect to receive information.

**Instructions:**

1. Make sure **vsftpd** is running and will restart on reboot.
2. The two-connection model used by FTP makes configuring the firewall rules to allow access a little more difficult than with other services. First, accept all traffic to the FTP command port:

```
# iptables -A RHS333 -p tcp --dport 21 -j ACCEPT  
# service iptables save
```

This works for active mode. For passive mode, the clients will be connecting to a second, random port on the server. Edit **/etc/sysconfig/iptables-config** to include the line

```
IPTABLES_MODULES="ip_conntrack_ftp"
```

Save the file, and **modprobe ip\_conntrack\_ftp**. This kernel module will watch for **PASV** commands and allow appropriate RELATED traffic.

3. Use the **telnet** command to pretend that you are an FTP client. Connect to the **ftp** port on your partner's station (**stationY.example.com**) with **telnet** and log in as user **ftp** and with password **root@example.com**.

```
[root@stationX]# telnet stationY.example.com 21  
Trying 192.168.0.Y...  
Connected to stationY.example.com (192.168.0.Y)  
Escape character is '^]'.  
220 (vsFTPD 1.2.0)  
USER ftp  
331 Please specify the password.  
PASS root@example.com  
230 Login successful.
```

4. Issue the **PASV** command and write down the reply.

```
PASV  
227 Entering Passive Mode (192,168,0,Y,20,196)
```

5. Calculate the server port where your next request's data will be available. Multiply the fifth number in parentheses by 256 and add the sixth number.

6. In another window, **telnet** to the calculated port on your partner's machine.
7. In the first window, type **LIST**. You should see a directory listing appear in the second window, then the second window's **telnet** connection should be closed.
8. Now try an active connection. In the first window, issue the following **PORT** command, assuming 192.168.0.Y is the IP address of your machine:

```
PORT 192,168,0,Y,122,105
200 PORT command successful. Consider using PASV.
```

(If you're connecting to an FTP server running on localhost, you must use 127.0.0.1 as the source address in your **PORT** command or it will be rejected for security reasons by the server.)

9. What port on our machine did we tell the server to use? In the second window, use the **nc** command to listen on that port for our data connection.
10. In the first window, type **LIST**. You should see a directory listing appear in the second window, then the second window's **nc** program should exit.
11. What's the difference between **PORT** and **PASV**? In the first window, type **QUIT** to disconnect your session.

## Sequence 2: FTP-only users

**Scenario:** You support a number of online "communities" each of which has its own web page. Configure **vsftpd** so that user **ftpguest** with password **ftpguest** is chrooted into **/var/www/html/community/** and starts in **/var/www/html/community/ftpguest/**. Do not allow **ftpguest** to have interactive shell access.

Make sure SELinux does not block ftp access to home directories:

```
# setsebool ftp_home_dir on
```

**Instructions:**

1. Create the user and assign a password.
2. Create the user's home area and assign ownership to the guest user.
3. Modify the guest user account's home directory and shell. Note that **vsftpd** can be configured to allow a user to chroot into a parent of their home directory.
4. Edit **/etc/vsftpd/vsftpd.conf** to uncomment the options

```
chroot_list_enable=YES  
chroot_list_file=/etc/vsftpd/chroot_list
```

and also add the line

```
passwd_chroot_enable=YES
```

Restart the **vsftpd** service.

5. Edit **/etc/vsftpd/chroot\_list** so that it contains a single line:  
**ftpguest**
6. Use **ftp** or **lftp** to connect to your ftp server as **ftpguest**. Can you change directory out of the **/var/www/html/community/** directory hierarchy? Can you login on a virtual console as **ftpguest**?

## **Sequence 3: Anonymous uploads**

**Scenario:** There is often a need to have an area where anonymous users can upload files. Without careful configuration, this can end up being used by attackers as a "warez" site. This lab configures a safe upload directory with **vsftpd**.

**Instructions:**

1. Create a directory (`/var/ftp/incoming/`) to accept the uploads. This directory should have `rwx` permissions for owner `root` and group `daemon`, `-wx` permissions for others (773).
2. Modify `/etc/vsftpd/vsftpd.conf` to allow uploads to the `/var/ftp/incoming` directory. Files created in the `/var/ftp/incoming` directory should have `0600` permissions, and belong to `root`. (In practice, it would be more secure to set up a special non-privileged user to own the files instead of `root`.) Clients should not be allowed to create new directories.

```
anon_upload_enable=YES  
write_enable=YES  
anon_mkdir_write_enable=NO  
chown_uploads=YES  
chown_username=root  
anon_umask=077
```

Reload the **vsftpd** service.

3. Test your configuration by logging in to your machine as the `anonymous` user. Upload your `.bash_profile` to the server. You may wish to save it as a non-hidden file. Verify that you are unable to list the `incoming` directory and can not retrieve the file you just uploaded.

## **Sequence 1 Solutions**

1. Make sure **vsftpd** is running and will restart on reboot.

```
# service vsftpd start; chkconfig vsftpd on
```

## **Sequence 2 Solutions**

1. Create the user and assign a password.

```
# useradd -M ftpguest; passwd ftpguest
```

2. Create the user's home area and assign ownership to the guest user.

```
# mkdir -p /var/www/html/community/ftpguest  
# chown ftpguest.ftpguest /var/www/html/community/ftpguest/
```

3. Modify the guest user account's home directory and shell. Note that vsftpd can be configured to allow a user to chroot into a parent of their home directory.

```
# usermod -d /var/www/html/community./ftpguest ftpguest  
# usermod -s /sbin/nologin ftpguest
```

### **Sequence 3 Solutions**

```
1.      # mkdir /var/ftp/incoming/  
      # chown root.daemon  
      # chmod 773 /var/ftp/incoming/
```

## **Unit 12**

### **Apache Security**

**12-1**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 828 2994 or +1 (919) 754 3700.

## Objectives

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

- Using the Directory directive
- Configuring mod\_ssl
- Secure CGI/SSI programming
- Using suEXEC

12-2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2904 or +1 (919) 754 3700.

## Vulnerabilities

- Default Apache installation is usually more open than desired.
- Careless use of CGI scripts can cause major security failures.
- HTTP messages are sent unencrypted by default.

12-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## Resolutions

- Avoid common misconfigurations.
- Establish policy to deny all access that is not specifically allowed.
- Use techniques to enhance security when programming CGI scripts or using SSI.
- Use SSL when possible.

12-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (819) 754 3700.

## Service Profile: Apache

- Type:#System V-launched service
- Packages:#`httpd`
- Daemons:#`httpd`
- Script:#`httpd`
- Ports:#80/tcp (http), 443/tcp (https)
- Configuration:#`/etc/httpd/*`, `/var/www/*`
- Related:#`cux`, `mod_ssl`, many others

12-5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[ttraining@redhat.com](mailto:ttraining@redhat.com)> or phone toll-free (USA) +1 (888) 826 2994 or +1 (919) 754 3700.

## <Directory>

- `<Directory /path/to/directory>`  
  Directives  
  `</Directory>`
- **Controls the configuration for a directory**
- **Absolute paths must be used**

12-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The Directory directive allows security parameters to be tuned for each directory on the server. The default configuration is to allow any file on the filesystem that can be reached through a URL to be served. A decision can also be made so that it is not allowed to let users override the servers configuration or allow symbolic links to be followed:

```
<Directory />
Options FollowSymlinks
AllowOverride None
</Directory>
```

More options will be discussed later.

## Apache Access Configuration

- Apache provides directory- and file-level host-based access control
- Host specifications may include dot notation numerics, network/netmask, and dot notation hostnames and domains
- The Order statement provides control over “order”, but not always in the way one might expect

12-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <trainings@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Directories and even files can have host-based access restrictions applied to them. Access is controlled through the use, either alone or in combination, of allow, deny, and order statements. The following lines, for example, access to /var/www/html/internal would be granted to only those hosts whose IPs reverse to a name in the example.com domain:

```
<Directory /var/www/html/internal>
order allow,deny
allow from .example.com
</Directory>
```

The order statement can take two arguments: order allow,deny and order deny,allow.

These arguments suggest that one can control the order in which Apache evaluates its access lists. It would be more accurate to understand these as statements of default behavior for hosts that are not included in the access control lists or when hosts that are included are specified in a contradictory fashion. If a host is included in a deny or allow list, but not both, then Apache will apply the access control. If a host is not in a deny or allow list, then the host will be handled by the default mechanism for the given order policy. If a host is included in both allow and deny lists, then it is handled by the default mechanism. It breaks down as follows:

order allow,deny -#allows explicitly allowed clients, denies everyone else; clients matched by both allow and deny are denied

order deny,allow -#denies explicitly denied clients, allows everyone else, clients matched by both allow and deny are allowed

For more information, see [http://httpd.apache.org/docs-2.0/mod/mod\\_access.html](http://httpd.apache.org/docs-2.0/mod/mod_access.html)

## Flat File Authentication

```
• <Directory /var/www/html/secret>
  AuthName "realm"
  AuthType basic
  AuthUserFile /path/to/passwords
  Require valid-user
</Directory>
```

12-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 828 2964 or +1 (919) 734 3700.

Multiple mechanisms are available to provide password-based security for all or part of a web site. These include flat files; LDAP servers; SQL databases; and others. Flat file authentication is the easiest to set up. It is not the most powerful method, but for many web sites it is good enough.

### *AuthName*

This statement specifies what “realm” is being logged into (in case the user has different usernames and passwords for different parts of the web site; realm is a concept roughly analogous to the NIS concept of domains). The *realm* must either be a single word or a string enclosed in double quotes.

### *AuthType*

Specifies that Basic authentication should be used; other types, such as Digest, are possible and are more secure, but most clients currently in use don’t support anything besides Basic authentication.

### *AuthUserFile*

Specifies the path to the password file that was created using `htpasswd`. This file should not be within `DocumentRoot`.

### *AuthGroupFile*

Declares the group file to be used for the authentication.

## Flat File Authentication

- Require user user1 user2, ...
- Require group group1 group2 ...
- Require valid-user
- Multiple Requires possible

12-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

Require statements define who can access the realm you are defining. It is sufficient that any of the requirements is fulfilled. It is possible to specify more than one user or group on the same line or to specify multiple Require lines.

### *Require user*

Only given users can access the realm.

### *Require group*

Only members of the given group can access the realm.

### *Require valid-user*

Anyone in the user file can access the realm.

## Managing Passwords

- **htpasswd** administers password files
  - -c creates a new file
  - -m uses Apache MD5 encryption

```
$ htpasswd -c -m /etc/httpd/htpasswd bob
New password: secret
Re-type new password: secret
Adding password for user bob
```

12-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 820 2994 or +1 (813) 754 3700.

Users can be added to a password file with the htpasswd command. This command is also used to modify an existing user's password.

*Important switches:*

- b#Batch mode. The password is given as an extra argument on the command line.
- c#Create the htpasswd file. If htpasswd file already exists, it is rewritten and truncated.
- D#Delete user. If the username exists in the htpasswd file it will be deleted.

*Encryption method:*

- m#hash passwords using Apache's modified MD5 algorithm (Default on Microsoft Windows.)
- d#hash passwords using the standard modified DES crypt() algorithm. (Default on all other platforms.)
- s#Use SHA encryption for passwords.

Warning: the - c switch will *recreate* the password file. If the file already exists all the contents will be *lost!*

## Kerberos Authentication

- **mod\_auth\_kerb** provides Kerberos authentication to Apache
- Based on Kerberos ticket exchange
- Could also be used for Basic Auth mechanism
- Use of SSL is strongly recommended

12-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <ctrlcrllog@redhat.com> or phone toll-free (USA) +1 (800) 626 2904 or +1 (979) 754 3700.

The **mod\_auth\_kerb** RPM is an Apache module which provides Kerberos authentication to the Apache Webserver. It is based on the typically Kerberos ticket exchange, which is called Negotiate method. To get this working you need a browser which supports the negotiateauth extension. The version of Firefox we ship with Red Hat Enterprise Linux 5 supports this mechanism.

The module also offers the Basic Auth mechanism which retrieves a username and a password from the browser and checks them against a Kerberos database. The module does not do any special encryption. That means in case of the Basic Auth mechanism, simply base64 encoding is used. This can easily be converted to plain text. The use of **mod\_ssl** is strongly recommended.

In order to get Kerberos Authentication working, the following requirements must be met:

- Webserver host must have a Kerberos principal (*addprinc -randkey HTTP/www.example.com*)
- The webserver principal has to be exported to a keytab file on the webserver host (*ktadd -k /etc/httpd/conf/keytab HTTP/www.example.com*)
- The keytab file must be readable only by Apache
- *network.negotiate-auth.trusted-uris* option in Firefox has to be set to the Kerberos REALM

Here is an example */etc/httpd/conf.d/auth\_kerb.conf* config file for the Apache webserver:

```
LoadModule auth_kerb_module modules/mod_auth_kerb.so
```

```
<Location /private>
SSLRequireSSL
AuthType Kerberos
AuthName "Kerberos Login"
KrbMethodNegotiate On
KrbMethodK5Passwd Off
KrbAuthRealms EXAMPLE.COM
Krb5KeyTab /etc/httpd/conf/keytab
require valid-user
</Location>
```

## Options

- Indexes
- ExecCGI
- Includes
- IncludesNoEXEC

12-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 426 2904 or +1 (919) 754 3700.

Options specify which Apache features are available for a Directory.

Example:

```
Options IncludesNoExec Indexes
```

Possible Options include:

Indexes	Creates a directory listing if no index file is present
ExecCGI	Allows the execution of CGI scripts
Includes	Enables Server Side Includes (SSI)
IncludesNoExec	Enables SSI without executing any commands
FollowSymLinks	Symbolic links are followed
SymLinksIfOwnerMatch	Only if the owner of the symlink is the same as the target file
MultiViews	If a document is available in multiple languages it is displayed according to the language settings for the browser.
All	All options are turned on
None	All options are disabled

## Common Misconfigurations

- Some Options can lead to problems
  - FollowSymLinks
  - ExecCGI
  - Indexes
- UserDir
- Serving directories that should not be shared

12-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2964 or +1 (919) 754 3700.

When Apache follows a symbolic link out of a directory tree, it does not change the directory name being matched against the <Directory> directive. Therefore, allow symbolic links with care.

The Indexes option will cause a formatted directory listing to be generated if the URL resolves to a directory instead of a file. This may not be desired. If the Indexes option is not turned on for a directory, a message denying access to the directory will be returned.

While popular among users, the UserDir directive, when set to public\_html, forces users to allow world access into their home directory. A better alternative is to set the UserDir directive to a pathname:

```
UserDir      /var/www/html
```

As previously discussed, general access to all directories should be disabled and appropriate access should be granted explicitly.

## Options FollowSymLinks

- FollowSymLinks is default for both the / (root) and /var/www/html directories
- Any web content that is a soft link will be followed - beware of users:
  - \$ ln -s / banner.jpg
  - \$ wget http://server/banner.jpg/etc/passwd
- FollowSymLinks should be disabled if not needed.

12-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2964 or +1 (919) 754 3700.

The slide demonstrates an example of the danger of allowing symbolic links. If symbolic links are a requirement, consider the SymLinksIfOwnerMatch option. This option causes the server to only follow the symbolic link the files pointed to by the link are owned by the same user who owns the link.

## Options Indexes

- The Indexes option allow directory browsing, if no DirectoryIndex file (like `index.html`) exists.
- Disable Indexes if automatic directory indexing is not intended.

12-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cctraining@redhat.com](mailto:cctraining@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

If the Indexes option is available for a directory, when the URL requested points to a directory rather than a file, and there is no file in the directory that is listed under the `DirectoryIndex` directive, the server will return a formatted directory listing. If the Indexes option is disabled for the directory, the client will receive a message to the effect of "You do not have permission to access this directory".

## Installing mod\_ssl

- Stop *httpd* before installing *mod\_ssl*
- \$ rpm -Uvh mod\_ssl\*
- /etc/httpd/conf/ssl.\* directories
- Generate SSL key and certificate
  - Manually with *openssl*
  - Using /etc/httpd/conf/Makefile
- Edit /etc/httpd/conf.d/ssl.conf

12-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2964 or +1 (919) 754 3700.

In the /etc/httpd/conf directory, the *mod\_ssl* RPM creates the following directories:

- ssl.crl - Directory for Certificate Revocation Lists
- ssl.crt - Directory for signed SSL certificates
- ssl.csr - Directory for Certificate Signing Requests
- ssl.key - Directory for Private keys
- ssl.prm - Directory for public DSA Parameter Files

It also creates a configuration file, /etc/httpd/conf.d/ssl.conf, that is an Include of httpd.conf.

A password-protected key can be generated with a convenient Makefile, or an unencrypted key can be generated by hand:

```
$ make genkey  
OR  
$ (umask 077; openssl genrsa 1024 >/etc/httpd/conf/ssl.key/server.key)
```

With the key, generate a test certificate or a CSR with the same Makefile:

```
$ make certreq  
OR  
$ make testcert
```

## SSL Virtual Hosts

- SSL virtual hosting requires a unique IP address per site
  - Limitation of SSL, not *mod\_ssl* or Apache
- Generate or obtain key and certificate, add three lines to *VirtualHost* block
  - *SSLEngine On*
  - *SSLCertificateFile /etc/httpd/conf/ssl.crt/name.crt*
  - *SSLCertificateKeyFile /etc/httpd/conf/ssl.key/name.key*

12-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (619) 784 3700.

Due to the nature of the TLS protocol, virtual hosts requiring SSL may not be name-based; you must use IP-based virtual hosting. The standard */etc/httpd/conf.d/ssl.conf* file includes a *\_default\_*:443 virtual host that will capture HTTPS requests for all IP addresses that do not have an associated *VirtualHost* block. The *VirtualHost* blocks listening on port 443 must precede the default one in the *ssl.conf* file, but follow the *Listen 0.0.0.0:443* directive.

The following example configuration would allow Apache to service HTTPS traffic to [www.example.com](http://www.example.com), at the IP address 10.1.2.3. It might be a good idea to set up a virtual host for 10.1.2.3:80 without the SSL directives for HTTP traffic.

```
<VirtualHost 10.1.2.3:443>
    ServerName www.example.com
    DocumentRoot /var/www/www.example.com/html
    ServerAdmin webmaster@example.com
    ErrorLog /var/log/httpd/www.example.com-error_log
    TransferLog /var/log/httpd/www.example.com-access_log
    SSLEngine on
    SSLCertificateFile /etc/httpd/conf/ssl.crt/www.example.com.crt
    SSLCertificateKeyFile /etc/httpd/conf/ssl.key/www.example.com.key
    SSLCipherSuite ALL:!ADH:!EXPORT56:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
</VirtualHost>
```

The *SSLCipherSuite* directive specifies which ciphers are offered by the server, in what preferred order. The example specifies that we want all ciphers (except those with no encryption), but then to exclude the anonymous Diffie-Hellman and 56-bit export ciphers, and then re-sort the cipher list, by “high-strength”, “medium-strength”, “low-strength”, those used by SSLv2, and finally any weak export ciphers remaining. Another directive worth investigating is *SSLRequire*, which can be used to restrict access based on various characteristics of the negotiated TLS connection.

## CGI - Common Gateway Interface

- Programs run on server
- CGI programs interact with server
  - Environment variables
  - CGI variables (forms)
  - Read and write files
- CGI programs can be written in any language

12-18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

The CGI specification defines a set of “rules” that allow programs to interact with web servers, such as Apache.

CGI programs allow web developers to provide dynamic content: a web page is generated “on-the-fly” when a user clicks on a link that points to a CGI program. CGI programs can also solicit input from users by accepting form data. From a user’s perspective, the fact that they are accessing a CGI program is usually completely transparent.

CGI programs normally run as user apache, the identity that Apache web server child processes are using. Apache passes various types of information to CGI programs through environment variables. These environment variables can contain information about the connection or may contain information supplied by the user of a form. The CGI programs can also read and write files on the web server. It is this capability that can make them somewhat dangerous from a security standpoint.

CGI programs can be written in almost any programming language, even as a shell script. Early versions of CGI programs were written in C and C++ but interpreted scripting languages such as Perl, Python and Tcl have become more popular because of their rapid application development capabilities. Securing CGI programs will require familiarity with whatever programming languages are being used on the web server to provide CGI content.

## Options ExecCGI

- CGI scripts (and ExecCGI) should be limited to particular directories
- Users with write access to those directories must be trusted to audit the CGI programs before deploying them

12-19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <ctraining@redhat.com> or phone toll-free (USA) +1 (866) 826 2004 or +1 (918) 754 3700.

CGI programs and scripts should be limited to special directories that only the web site administrator and trusted assistants have write access to. All CGI programs must be carefully audited for bugs and potential security holes before they are copied into a directory that has Options ExecCGI set.

As the Apache documentation notes, by allowing the users to execute CGI scripts that are not audited, the server is at their mercy. The administrator is trusting that they will not write scripts that will accidentally or deliberately expose the system to attack. Such dangerous CGI scripts can be quite simple; a web form that executes the input text as a command on the web server, for example.

## CGI Secure Programming

- As with every language, there are good coding practices, and things to avoid.
- General Rules to follow:
  - Do not trust user input; check it!
  - Do not directly execute user input
  - Do not assume input is in the expected format

12-20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[trainings@redhat.com](mailto:trainings@redhat.com)> or phone toll-free (USA) +1 (866) 626 2964 or +1 (919) 754 3700.

A very common mistake that is made with CGI programs is to use the information given by a user directly on the command line of the CGI script. This can allow the user to execute arbitrary commands.

User input should always be checked no matter what the scripting language used. If the user is limited to the character set [A-Za-z\_0-9.-] it should be fairly safe. In particular, if the user needs to enter characters other than those, pay particular attention to the input and make sure it can not be used to pass arbitrary commands.

## CGI With Unix Shell Script

- Poor security!
  - Do not use in a production environment
- Works for quick and dirty CGI scripts
- Slower than other languages
- The shell *is* the program
  - Easy to work with environment variables
  - Trivial to pass bad input to the shell

12-21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (619) 754 3700.

While a Unix shell is convenient and easy to use for cranking out quick CGI scripts, security experts are quick to point out that Unix shells have poor input checking methods compared to other languages.

It is recommended Unix shell scripts be limited to prototyping simple CGI scripts only. Once you have a proof of concept script, rewrite your Unix shell CGI script in a more robust (and speedy) language.

## SSI: Server Side Includes

- Instructs Apache server to include other files/data within HTML
  - Enabled by Options Includes

12-22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (888) 626 2904 or +1 (919) 754 3700.

Server-side includes provide an easy way to create simple dynamic documents without the use of CGI. While less flexible than CGI scripts, SSI are typically faster and more efficient.

*Examples:*

```
<!--#include virtual="header.ssi" -->
```

includes common header/footer HTML code.

```
Document last modified on <!--#flastmod file="index.html" -->
```

includes the modification date of the specified file.

Server-side include functionality is most often used to include common HTML fragments throughout a website. For example, copyright information at the bottom of each webpage can be included in a single file and included by all other files. This makes management of the included information much more convenient because the web author only needs to make any changes to one file and not every file in the entire website.

```
<!--#include virtual="/disclaimer.ssi" -->
```

More information about SSI can be found at:

<http://httpd.apache.org/docs-2.0/howto/ssi.html>

## SSI Security

- **include - Includes file or URI**
  - At worst, allows referencing other CGI programs in defined ScriptAlias locations:
  - <!--#include virtual="/head.cgi" -->
- **exec - Includes output of program**
  - Runs arbitrary commands or CGI programs
  - <!--#exec cmd="/usr/bin/who" -->
  - Use `IncludesNoExec` to disable exec

12-23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 836 2894 or +1 (919) 754 3700.

SSI is fairly safe but the `exec` directive is potentially dangerous.

Example:

```
<!--#exec cmd="cat /etc/passwd" -->
```

Use the `IncludesNoExec` option to prevent inappropriate use of `exec` but allow all other SSI operations:

```
Options IncludesNoExec
```

## **SuEXEC**

- CGIs and SSIs run as user apache
- suEXEC forces both CGI and SSI programs run as a specified user/group
- Will not execute any program unless:
  - It is owned and only writable by the user
  - It is in a directory only writable by the user

12-24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

Normally the web server performs all operations, including execution of CGI scripts, as a user with very few privileges. Apache defaults to running as the user apache. This is a safe and effective strategy for mostly static websites, but many dynamic websites have CGI scripts that need more privileges. For example, a script may need to write to a file, or access a database as a user other than apache. Apache includes the suexec program with its distribution which can be useful in this situation. Apache was built with the --enable-suexec configuration option in Red Hat Enterprise Linux, and includes the suexec program in /usr/sbin.

With suexec installed on the server, and the mod\_suexec module loaded, Apache will use it to run CGI scripts as a specific user in the following circumstances:

- 1.#When a CGI script is in a user's public\_html directory and is accessed via the ~user/ URI.
- 2.#When a website configured via the <VirtualHost> directive includes a SuexecUserGroup directive inside the <VirtualHost> section.

```
<VirtualHost 192.168.1.2:80>
    ServerName www.somedomain.org
    ServerAdmin webmaster@somedomain.org
    ScriptAlias /cgi-bin/ /www/somedomain/cgi-bin
    DocumentRoot /www/somedomain/docs
    SuexecUserGroup someuser somegroup
</VirtualHost>
```

- 3.#The suexec binary is uid root. This is enabled by default.

The suexec program will *not* run programs as the root user, nor can it use mod\_perl or mod\_php due to the embedded UID nature inherent to these CGI replacements.

## **End of Unit 12**

- Questions and Answers
- Summary
  - Per-directory access controls
  - Common misconfigurations
  - Configuring mod\_ssl
  - CGI/SSI security

**12-25**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (800) 626 2994 or +1 (919) 754 3700.

## **Lab 12**

### **Apache Security**

---

**Goal:** Configure the web server for secure operation and learn of the effects of SuexecUserGroup specifications with suEXEC.

**Estimated Duration:** 90 minutes

## **Sequence 1: Install Apache and configure .htaccess files**

**Scenario:** In this sequence we will install the Apache webserver and configure your machine to authenticate users for some content.

**Instructions:**

1. As root, install Apache, and some related Apache module packages. We will also install the text-based elinks web browser for testing our server.

2. Edit the /etc/hosts file so it appears as follows:

```
127.0.0.1 localhost localhost.localdomain  
192.168.0.X stationX.example.com stationX
```

3. As root, open /etc/httpd/conf/httpd.conf for editing. Find the <Directory /var/www/html> block and remove Indexes from the Options line to turn off automatic index creation for the default web site's document tree. Can a content administrator re-enable filesystem browsing if they can't edit httpd.conf, but can create content in the DocumentRoot of a virtual host or in a user ~/public\_html directory?

4. Create a new directory tree and some content for authorized users to access:

```
# mkdir /var/www/html/my_secrets  
# echo "This is my file that only authorized people can see." > <  
/var/www/html/my_secrets/index.html  
# mkdir /var/www/html/my_secrets/my_mp3s  
# echo "This simulates retrieving an actual file." > <  
/var/www/html/my_secrets/my_mp3s/index.html
```

5. Set up your protected directory by changing the value of the AllowOverride directive in your httpd.conf, it should be in the <Directory /var/www/html> section:

```
AllowOverride AuthConfig
```

6. Start your webserver and configure it to run in runlevels 2, 3, 4 and 5.

7. In an earlier lab, you allowed access to the web server ports through the firewall. Make sure that these rules are still in place (with iptables -L) or add them again.

8. If you try to access `http://stationX.example.com/my_secrets/index.html` you will see that you still have access to the content as you have not yet set up a `.htaccess` file. Let's remedy that. Create the file `/var/www/html/my_secrets/.htaccess`. It should have the following content:

```
AuthName "Access to my_secrets"
AuthType Basic
AuthUserFile /var/www/my_passwords
require valid-user
```

The settings above will cause a dialog box to be generated by the client's web browser. The title of the dialog box will contain the phrase "Access to my\_secrets". The dialog will also have a box for the user to input a username and password. The username and password will be checked against the data in `/var/www/my_passwords` and a valid username and password must be provided.

9. Configure the password file `/var/www/my_passwords` with the `htpasswd` command.
10. Test the setup. Are you prompted by your web browser for the username and password? Are you able to authenticate? If you are having problems authenticating, check to make sure that all the files, including `/var/www/my_passwords`, are readable by the apache user account. Once you authenticate to use the content, the browser will cache the information. Close your browser and open a new one. Try accessing `http://stationX.example.com/my_secrets/my_mp3s/index.html`. Notice that your `.htaccess` file protects not only the directory that holds it, but all content in the directory tree.

## **Sequence 2: CGI Setup and suEXEC**

**Scenario:** In this sequence we will modify the virtual host settings we created by adding CGI and suEXEC capabilities. We will also create a CGI program to use for testing.

**Instructions:**

1. Create a new `cgiuser` user on your system.
2. Now create a `VirtualHost` section for `stationX.example.com`:

```
NameVirtualHost 192.168.0.X

<VirtualHost 192.168.0.X>
    ServerName stationX.example.com
    ServerAdmin webmaster@stationX.example.com
    DocumentRoot /var/www/html
    ErrorLog logs/error_log
    CustomLog logs/access_log combined
    ScriptAlias /cgi-bin/ /var/www/virtual/cgi-bin/
</VirtualHost>
```

Create the `/var/www/virtual/cgi-bin` directory, and make sure it's owned by the `cgiuser` user and group, with access permissions 0755. Restart `httpd`.

3. Create a `script.sh` file in this directory with the following contents:

```
#!/bin/bash

echo Content-type: text/html
echo

whoami
echo '<br><br>'
id || echo "Will not work with SELinux."
echo '<br>'
```

Make sure you make this script executable and owned by user and group `cgiuser`.

4. Test this script to be sure that it is working properly. Try the command

What user is running the script?

Make sure SELinux permits the execution of the script.

5. Next add `SuexecUserGroup` lines to the `stationX.example.com` `VirtualHost` section of `httpd.conf`:

```
SuexecUserGroup cgiuser cgiuser
```

6. Now restart the Apache webserver, and execute the script again. Compare the output.

Now, what user is running the script? If the output is the same, consult `error_log` for possible errors. Make sure the script and the `cgi-bin` directory are owned by user and group `cgiuser`.
7. It is time to protect the server against displaying directory information and disallowing symbolic links. Create a directory for the content:

```
# mkdir /var/www/html/options
```
8. Inside your `VirtualHost` entry in `/etc/httpd/conf/httpd.conf`, add an entry for the new directory you just created:

```
<Directory /var/www/html/options>
    Options -FollowSymLinks
</Directory>
```
9. Create a symbolic link that points to `/`:

```
# ln -s / /var/www/html/options/link.jpg
```
10. Open a browser and try to follow the link.

This should work.
11. Restart your web server and test that it works.

This should give you a blank page. If it doesn't, go back and look at error messages for clues.
12. Open a browser and try to follow the link.

This should not work. If it does, check your logs again.

## **Sequence 3: Configuring mod\_SSL**

### **Instructions:**

1. Make sure that there isn't a SELinux related problem, which prohibits Apache to ask for a certificate passphrase:
2. Create an RSA private key for your Apache server. For the passphrase, use `redhat`.
3. To see details of your RSA private key, you can run the following:

```
[root@stationX]# openssl rsa -noout -text -in server.key
read RSA key
Enter pass phrase:
Private-Key: (1024 bit)
modulus:
    00:dd:a2:9c:cb:16:99:65:b0:1c:ce:5d:ef:0d:48:
    f8:b5:37:2b:13:a5:7b:73:7f:cb:b8:aa:e5:6d:fb:
...extensive output omitted ...
```

Compare the decrypted output displayed by the above command, to the encrypted contents of the `server.key` file.

4. Create a Certificate Signing Request (CSR) with the server RSA private key. Be sure to replace placeholders below your information.
5. Finally, use your CA from Lab 3 to sign your server's certificate.
6. Now that you have successfully created and signed a Certificate, move the certificate into place, then modify your Apache server configuration to enable SSL.
7. To test your SSL configuration, begin by restarting your Apache service.  
  
If the server doesn't start, remove the virtual host created earlier from `/etc/httpd/conf/httpd.conf`.
8. Now use Firefox to connect to your main site. Note the use of `https` designating an SSL connection:  
  
You should be given several warnings, and informational screens, but eventually connected.

## **Sequence 4: Decrypting Private Key**

**Scenario:** While you were on vacation, the web server was stopped for maintenance. When your boss, tried to restart the web server, it prompted for the pass phrase (which he did not know). Your boss has now ordered you to change your SSL configuration so that your private key is stored in clear text.

**Instructions:**

1. Start by making a backup of your encrypted private key:
2. Now use the **openssl** utility to decrypt your key and store it using the original filename.
3. Use **cat** to compare the contents of the two keys. Then use **chmod** to change permissions on your clear-text key (**server.key**) to 700 (only accessible by **root**).
4. Test your configuration changes by restarting the Apache server, and verifying that it no longer prompts for a decryption key.

## **Challenge Sequence 5: Secure .htaccess**

**Deliverable:**

**Instructions:**

1. In Sequence 1, you set up basic username/password authentication to your "secure" directory. However, the authentication credentials are passed from the client's browser to your server as cleartext over the network, making them subject to capture. Set up SSL to protect the authentication. Generate an appropriate certificate, make any necessary changes to the default SSL virtual host in /etc/httpd/conf.d/ssl.conf, and add the directive SSLRequireSSL to your .htaccess file. You can assure yourself that the exchange is protected by examining the network messages before and after your changes with a network sniffer such as **wireshark** or **tcpdump**.

## Sequence 1 Solutions

1. As root, install Apache, and some related Apache module packages. We will also install the text-based elinks web browser for testing our server.

```
# yum install httpd mod_ssl elinks
```

6. Start your webserver and configure it to run in runlevels 2, 3, 4 and 5.

```
# service httpd restart  
# chkconfig httpd on
```

7. In an earlier lab, you allowed access to the web server ports through the firewall. Make sure that these rules are still in place (with iptables -L) or add them again.

```
# iptables -A RHS333 -m multiport -p tcp --dport 80,443 -j ACCEPT  
# service iptables save
```

9. Configure the password file /var/www/my\_passwords with the htpasswd command.

```
# htpasswd -c /var/www/my_passwords username
```

Note that we are using the -c option to *create* this file. If we were adding users to an already existing file we would not use the -c option, as it would overwrite any entries that already existed. Also realize that the username provided does not have to be an existing account on the machine, but is rather the username that the server is expecting from the client's web browser dialog.

## Sequence 2 Solutions

1. Create a new cgiuser user on your system.

```
# useradd cgiuser
```

2. # mkdir /var/www/virtual/cgi-bin/  
# chown cgiuser.cgiuser  
# chmod 755 /var/www/virtual/cgi-bin/  
# service httpd restart

4. Test this script to be sure that it is working properly. Try the command

```
# links -dump http://stationX.example.com/cgi-bin/script.sh
```

Make sure SELinux permits the execution of the script.

```
# setsebool -a httpd_enable_cgi=1
```

6. Now restart the Apache webserver, and execute the script again. Compare the output.

```
# links -dump http://stationX.example.com/cgi-bin/script.sh
```

10. Open a browser and try to follow the link:

```
# links http://stationX/options/link.jpg/etc/passwd
```

11. Restart your web server and test that it works:

```
# service httpd reload  
# links http://stationX.example.com/options
```

12. Open a browser and try to follow the link:

```
# links http://stationX/options/link.jpg/etc/passwd
```

## Sequence 3 Solutions

1. Make sure that there isn't a SELinux related problem, which prohibits Apache to ask for a certificate passphrase:

```
# setsebool httpd_tty_comm=1
```

2. Create an RSA private key for your Apache server. For the passphrase, use redhat.

```
# cd /etc/httpd/conf
# (umask 077; openssl genrsa -des3 -out server.key 1024)
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase:
Verifying password - Enter pass phrase:
```

4. Create a Certificate Signing Request (CSR) with the server RSA private key. Be sure to replace placeholders below your information.

```
# openssl req -new -key server.key -out server.csr
Enter PEM pass phrase: redhat
Country Name (2 letter code) [GB]: US
State or Province Name (full name) [Berkshire]: North Carolina
Locality Name (eg, city) [Newbury]: Raleigh
Organization Name (eg, company) [My Company Ltd]: Red Hat
Organizational Unit Name (eg, section) []:
Common Name (eg, your server's hostname) []: stationX.example.co m
... accept defaults for remainder ...
```

5. Finally, use your CA from Lab 3 to sign your server's certificate.

```
# openssl ca -in server.csr -out server.crt
```

6. Now that you have successfully created and signed a Certificate, move the certificate into place, then modify your Apache server configuration to enable SSL.

```
# mv server.key /etc/httpd/conf/
# mv server.crt /etc/httpd/conf/
```

In the <VirtualHost \_default\_:443> section of /etc/httpd/conf.d/ssl.conf, locate and modify the following two SSL directives to read:

```
SSLCertificateFile /etc/httpd/conf/server.crt
SSLCertificateKeyFile /etc/httpd/conf/server.key
```

7. To test your SSL configuration, begin by restarting your Apache service.

```
# service httpd restart
```

```
Shutting down http: [ OK ]
Starting httpd:
Apache/1.3.14 mod_ssl/2.7.1 (Pass Phrase Dialog)
Some of your private key files are encrypted for security ✓
reasons.
In order to read them you have to provide us with the ✓
passphrases.
Server stationX.example.com:443 (RSA)
Enter pass phrase: redhat [ OK ]
```

8. Now use Firefox to connect to your main site. Note the use of `https` designating an SSL connection:

```
# firefox https://stationX.example.com
```

## **Sequence 4 Solutions**

1. Start by making a backup of your encrypted private key:

```
# cd /etc/httpd/conf  
# cp server.key server.key.encrypted
```

2. Now use the **openssl** utility to decrypt your key and store it using the original filename:

```
# openssl rsa -in server.key.encrypted -out server.key  
read RSA key  
Enter PEM pass phrase: redhat  
writing RSA key
```

## **Unit 13**

### **Intrusion Detection and Recovery**

**13-1**

For use only by a student enrolled in a Red Hat Training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (608) 626 2994 or +1 (919) 754 3700.

## **Objectives**

**Upon completion of this unit, you should be able to:**

- Intrusion risks
- Developing a security policy
- Investigating and documenting intrusions
- Recovering from intrusions

**13-2**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2894 or +1 (919) 754 3700.

## Intrusion Risks

- System downtime
- Theft of data
- Modification or destruction of data
- Installation of hostile software
- Bad publicity and financial impacts

13-3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2994 or +1 (919) 754 3700.

The risks of a compromised system depend in many ways upon the goals of the attacker. Some people break into systems just to see if they can do it, often with no particular goal. More often, the attacker will have a specific purpose in mind for the attack or the compromised system. These purposes can include:

- Disruption of activity or denial of service
- Defacement of websites
- Use of the system as a base of operations to attack other machines
- Use of the system as a “zombie” host participating in a distributed denial-of-service attack
- Corrupting or holding hostage sensitive data
- Industrial espionage or silent theft of sensitive data

An attacker who breaks in, copies data, and then leaves without returning or interrupting the system can be very difficult to detect and track. The best defense against this sort of attacker is to keep sensitive data that must be on the network on dedicated systems with tightly controlled authentication for only users that must have access.

## Security Policy

- Your organization should have a policy on
  - Detection of possible intrusions
  - Verification and investigation of intrusions
  - Recovery from intrusions
  - Reporting of intrusions
  - Documentation of the policy's execution
- The policy should be developed with the support of management and legal counsel

13-4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

There is no such thing as a completely secure system. Your goal in securing your systems should be two-fold:

1. Put up enough barriers so that it is not worth it to an attacker to break in, while still maintaining the usability of your systems.
2. Be prepared with a plan of action for what you will do when an attacker breaks in anyway.

Every organization should have a incident response plan and policies in place to handle an intrusion. These policies and plans should be formulated with the help of management and legal counsel, supported widely in the organization, and tested periodically.

Mark Cox (head of Red Hat's security response team) and Michael Tiemann (VP, Open Source Affairs) developed a paper on security best practices that is available at [http://www.redhat.com/advice/best\\_practice\\_security.html](http://www.redhat.com/advice/best_practice_security.html). Regarding security policy they had this to say:

*"The first and foremost best security practice is to have a documented security policy. Regardless of how complete or incomplete this policy may be, the policy is the objective reference against which one can measure "what did we say we would do, what did we do, and what do we need to do in the future to do better?" If the policy is found to have had a hole, the policy can be amended and security improved. However, if there is no policy, then every incident becomes a fire drill: can it ever happen again? Will it ever happen again? What if it does happen again? Trying to make those decisions in the absence of a documented security policy leads most often to either spending a lot of money on a solution that does not really fix the root problem, or spending nothing, learning nothing, and hoping it does not happen again."*

# Detecting Possible Intrusions

- Regular monitoring of...
    - Log files
    - Network traffic
    - Open ports
    - Modified files

13-5

**For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent from Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[trainings@redhat.com](mailto:trainings@redhat.com)> or phone toll-free (USA) +1 (866) 628 2904 or +1 (919) 784 3700.**

After everything has been done to prevent an intrusion, there is still a need to monitor the systems for signs of attempted or successful break-ins. Host-based and network-based intrusion detection systems, manual monitoring, and judicious use of standard tools can help catch an intrusion early.

## Detecting Possible Intrusions

- Monitoring log files
    - Systems should log to local files and to a dedicated remote host
    - Logs should be analyzed on at least a daily basis using *logwatch*
    - Logwatch results should be delivered to a separate system
    - Look for signs of subverted services

13-6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, exploited, or distributed please email [abuse@redhat.com](mailto:abuse@redhat.com), or phone 1-866-338-3904 or 1-800-784-2700.

In many cases, it does an attacker no good if they compromise a system only to be immediately detected and locked out by an administrator. Therefore, the first action of most attackers is to cover their tracks by altering the system logs. Local system logs on a compromised system should be treated sceptically. One good defense against altered logs is to send log messages both to local files and to a remote, central log host. Differences between the logs should immediately get your attention.

The logs should be analyzed on a frequent basis, at least daily, by automated tools such as logwatch. The logwatch tool provides an easy to read summary of the activity reported in a system's logs. No configuration is necessary to make logwatch work once it is installed. It is always a good idea to have logwatch send its reports to a mailbox that is offsite or at least on another machine. This can be set by altering the MailTo directive in `/etc/log.d/logwatch.conf`.

While logwatch reports are very useful, when investigating an intrusion the logfiles should always be examined manually for unusual entries. For example, an entry like:

might be indicative of a buffer overflow used to subvert the NFS mounted daemon.

## Monitoring Network Traffic

- Intrusion Detection Systems
- Use iptables rules to log suspicious access attempts
- Use tools like tcpdump and wireshark to capture and store suspicious traffic

13-7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 828 2904 or +1 (919) 764 3700.

An **Intrusion Detection System (IDS)** is any software that aids an administrator in identifying attempts to circumvent security measures on his or her systems. There are host-based IDSes like Tripwire that analyze the local station, and network-based IDSes such as Snort that monitors the network for unusual activity.

Snort monitors network traffic in real time and compares captured packets against a set of “rules” that describe hostile or otherwise suspicious traffic. Snort can log to the system log files or to a database and can deploy “agents” across a network, all reporting to a central logging system. Snort is third-party software that is not included in Red Hat Enterprise Linux. See <http://www.snort.org> for more information.

The iptables firewall can be a valuable tool for detecting network scans and unauthorized access attempts. Any describable packet can be logged using the LOG target. For example, the following rules:

```
$ iptables -I INPUT -p tcp --syn --dport 80 -j ACCEPT
$ iptables -A INPUT -p tcp --syn -j LOG --log-prefix "Connection Attempt: "
$ iptables -A INPUT -p tcp --syn -j DROP
```

would allow attempts to initiate a connection with port 80 (http) but log and then drop any attempts to access other TCP ports. Note that these examples apply only to TCP (as opposed to UDP) traffic. Use the iptables connection tracking facilities to identify new connections of any type.

If the iptables logging reveals repeated unusual access attempts, it may be wise to do a network capture for later analysis. The command:

```
$ tcpdump 'dst port not 80 and tcp-syn' -w access.log
```

would capture all new TCP connection attempts to ports other than 80 and store the packets in a file called access.log. The log file could be analyzed later using tcpdump or wireshark to get a better idea of the nature of these connection attempts.

## Monitoring Open Ports

- **netstat (from the local system)**
  - Show listening daemons: `$ netstat -tulpn`
  - Show active connections: `$ netstat -tupn`
- **nmap (from a remote system)**
  - Make sure you are authorized to scan first!
  - Scan a host: `$ nmap -P0 server1.example.com`
  - Scan a subnet: `$ nmap -sV 192.168.0.0/24`
  - GUI front-end: `nmapfe`

13-8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

It is a common tactic of intruders to install hostile daemons on a compromised system. Such daemons may provide “back door” remote access to the system, or may be “zombie” daemons awaiting a command to begin contributing to a distributed denial-of-service attack. As such it is important to monitor which services are listening for connections on your systems. The command:

`$ netstat -tulpn`

will list all TCP (-t) or UDP (-u) services currently in “listen” (-l) mode. The -p switch causes the executable name and process ID of each daemon to be listed as well and the -n switch suppresses DNS lookups.

Since netstat is a local utility resident on the suspect machine, it is often a target for modification or replacement. Trusting the local netstat when investigating a possible compromise is therefore not a good idea. Port scanners like nmap should be run regularly from external hosts to see what services appear to be listening on the machines. Be sure to note which systems where the scans come from so that it is possible to tell the difference between evidence of one of the test scans and evidence of someone else's. The command:

`$ nmap -P0 mybox.example.com`

would list all listening services on mybox.example.com that are not blocked by a firewall rule or some other host-based access mechanism. The -p0 is important because otherwise nmap will initially try to ping the target system to see if it is up. If the firewall blocks ping packets, nmap will report no services found on the machine. The nmapfe tool provides a helpful graphical interface for controlling nmap's many options. nmap can be used to establish a baseline record of which ports are normally open on a system. Daily scans via cron can be compared to the baseline, highlighting any changes. If a scan is done from or to a system that is not under the administrators control, make certain that permission has been given from the relevant authorities before using nmap. Unauthorized scans will almost always be a violation of a network's usage policy.

## Detecting Modified Files

- **md5sum**
  - Compare to listings at [rhn.redhat.com](http://rhn.redhat.com)
  - Fingerprints can be stored in a file for later comparison
- **cmp**
  - Byte-by-byte comparison of two files
  - Slower than **md5sum**, but harder to fool
  - Must have known-good version of file available
- **prelink** modifies ELF shared libraries and ELF dynamically linked binaries

13-9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (800) 826 2994 or +1 (919) 754 3700.

It is important to keep a record of the original attributes of files on the system. Then comparisons can be made to see if there have been any changes. Important files to monitor in this manner include binaries, anything run from cron, init scripts and kernel modules.

The **md5sum** tool can be used for this purpose. It cryptographically hashes the contents of any file, using a variation of the MD5 algorithm. The result of this calculation is a string of characters that can serve as a “fingerprint” for the file. It is “statistically infeasible” for two different files to hash to the same value. Thus, if **md5sum** is run in a trusted environment, fingerprints can be taken of important system binaries and compare them to a “baseline” record of their original values. The following command, for example, would record an **md5sum** for every executable on the system in a file called **execs.md5**:

```
$ find / -type f -perm +ugo+x -exec /usr/bin/md5sum {} \; > /root/execs.md5
```

The fingerprints file should be stored in a secure location, preferably on another system. The next command, when run from a trusted system on a mounted image of the suspect system, would print an alert for any entries in **execs.md5** whose **md5sums** had subsequently changed (the **sed** command here simply prepends the directory where the suspect system's image has been mounted to each file's path in **execs.md5**):

```
$ sed 's/ \// \/\path\/\to\image\//' executables.md5 | md5sum --check
```

It is possible, however difficult or unlikely, for a file to be modified in such a way that its **md5sum** remains the same. A slower but much harder to fool utility is **cmp**. It compares two files byte-by-byte and reports any differences. To accomplish this, an unmodified copy of each binary of files to be tested is needed.

There might be a problem when using **md5sum** on ELF shared libraries and dynamically linked binaries because of **prelink**. **prelink** is a program which modifies ELF shared libraries and ELF dynamically linked binaries, so that the time which dynamic linker needs for their relocation at startup significantly decreases and also due to fewer relocations the runtime memory consumption decreases too.

`rpm -V` compensates for prelink.

## Detecting Modified Files

- **tripwire**
  - Third-party product
  - Commercial and free versions are available
  - Highly configurable catalog of file properties
- **rpm**
  - Keep a backup database as a baseline
    - `$ rpm -V package`
    - `$ rpm -Vf /path/to/file`
    - `$ rpm -Va`

13-10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[cstraining@redhat.com](mailto:cstraining@redhat.com)> or phone toll-free (USA) +1 (956) 626 2904 or +1 (919) 754 3700.

Tripwire is a third-party utility that does not come with Red Hat Enterprise Linux. Free and commercial versions of Tripwire exist. The free version is stable but unmaintained. Tripwire works by storing an encrypted database of file attributes including size, ownership, permissions and md5sum for all specified files on a system. Tripwire then regularly performs comparisons of the current state of files to their information as recorded in the database. Discrepancies are noted and emailed to an administrator in daily reports. The free version of Tripwire is available at <http://www.tripwire.org/>.

Files installed as part of an RPM package have information including size, ownership, permissions and md5sum stored in the RPM database (`/var/lib/rpm/*`). These values can be compared against the current properties of files on the system with the **rpm -V** command. For every file rpm finds that has changed, it prints out a string of characters representing whichever of the following changes are applicable to that file (from the rpm man page):

- S - file Size differs
- M - Mode differs (includes permissions and file type)
- S - MD5 checksum differs
- L - readLink(2) path mis-match
- U - User ownership differs
- G - Group ownership differs
- T - mTime differs
- C - SELinux Context differs

The RPM database has few protections against modification by an attacker. It is a good practice to keep a backup of the database on CD or other media. The backup database can be used instead of the one in

`/var/lib/rpm` by setting the `%_dbpath` RPM macro. This can be useful in conjunction with the rescue environment:

```
$ rpm --root /mnt/sysimage --define '_dbpath /mnt/backup/rpm' -vav
```

## Investigating and Verifying Detected Intrusions

- Work from a secure environment
  - Rescue mode
  - Using a trusted system
  - Linux-on-CD custom distributions
- Image suspect block devices for analysis
- Perform file integrity checks

13-11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2984 or +1 (919) 754 3700.

The first rule of investigating a suspect system is that any element of that system should be treated as trustworthy. For example, the md5sum command can be used to find modified files... unless the md5sum binary itself has been modified to ignore files the intruder does not want the administrator to know about. Even if a trusted binary is copied over from another system, it is possible that a hostile kernel module could be intercepting system calls and again hiding information an intruder does not want the administrator to see. So, how can the administrator investigate a system without using its kernel or any of its binaries?

The first CD in the Red Hat Enterprise Linux installation set or a CD made from the included boot.iso image can be used to enter "rescue mode". Rescue mode is a self-contained Linux system loaded entirely from the CD. It uses its own kernel and its own binaries. To enter rescue mode, boot from an appropriate CD and type linux rescue at the boot prompt. The system's drives will be mounted in /mnt/sysimage. There are also a number of Linux distributions designed to provide a more complete feature set than rescue mode while still working from a single bootable CD. The most popular of these is probably the Knoppix distribution, which includes a derivative distribution geared toward intrusion detection and investigation. See <http://www.knoppix-std.org> for more information.

Another useful investigative technique is to create an exact duplicate of the suspect system's hard disk for analysis. This has a number of advantages. First, it eliminates the risk of altering the system yourself in the course of the investigation, which can invalidate the findings as evidence. Second, it allows the system to be re-installed, patched and restored from a recent backup so as to get it back into production even as the investigation continues. The main disadvantage to using a disk image is that, as with a rescue environment, it only allows the administrator to review data stored on the system's hard disks, not what is in the system's memory.

Having employed a secure method of accessing the data on a suspect system, the next step should be to look for signs of an intrusion. Modifications to important files, the addition of new user accounts and strange filenames like "HIDDEN" (which may indicate a LKM root kit) are all signs to look for. Tools exist to automatically look for the signs of common root kits. Visit <http://www.chkrootkit.org> for more information. Even without such tools there are a number of investigative options available. Try to think of some telltale sign that the attacker might not have thought to cover up. For example, the command find -ctime 1 will return a list of all files in the current directory, including its subdirectories, that have been created or had their properties modified in the last day. Is there anything in that list that looks like it should not be there?

## Creating a Disk Image

- Create a copy to study, do not use the original evidence
  - Do this from a trusted rescue environment!
- Partition images can be mounted for analysis
  - `$ mount -o loop victim.hda1.img /mnt/victim/`
- Whole disk images can be used to duplicate a system for study

13-12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Creating a disk image is often the first step of an investigation. Two copies are recommended; one to be archived in a secure location with the original media, and one to use as a working copy for the investigation. Assuming the hard drive has been attached as a slave IDE device to a trusted system, you could image it with the dd command:

```
$ dd if=/dev/hdd of=/evidence/image-hdd bs=1k conv=noerror,sync
```

It is important not to use the original hard drive for any part of the analysis but the initial imaging. If there is any modification of the drive, its value as evidence in any potential legal proceedings is likely to be damaged. Records should be kept of who has access to the original hard drive at all times, and access should be strictly limited.

Images could also be taken of individual partitions, which could then be loop mounted for further examination.

## Detecting and Defeating Backdoors

- Strict inbound and outbound firewall rules
- Regular checks of listening ports and files compared to a known baseline
- Network traffic monitoring and recording with tools like tcpdump or Snort
- Monitoring open files with lsof and fuser

13-13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2904 or +1 (919) 754 3700.

A common security error is to pay close attention to inbound firewall rules while not restricting outbound traffic at all. Restricting outbound connections can be tricky on workstations, where clients may be using arbitrary high ports connecting to virtually anything, but on servers this should prove less of a problem. A web server, for example, can probably be expected to only transmit data from ports 80 (http) 22 (ssh), and maybe 20/21 (ftp) to arbitrary high ports. It can also be expected to connect from its own high ports to port 53 (dns) on your local nameserver, and maybe to port 123 (ntp) on your network's time servers. Any other outbound packets should be logged and dropped. Even a hidden backdoor daemon is of no use to an attacker if it cannot get out. While it is possible for an attacker to modify the firewall rules to allow his or her backdoor to have access and even to cover up such modifications, it is one more obstacle for him or her to anticipate and overcome, thus raising the bar for who can successfully compromise the system and remain undetected.

## Detecting and Defeating Root Kits

- Regular checks of log files and network traffic for suspicious activity
- Regular checks for promiscuous network interfaces
- Test system binaries on a trusted system
- Break automated root kit installs
- Root kit detectors

13-14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2694 or +1 (919) 754 3700.

There are two components to defeating an attacker. The first component is detecting the attacker's presence; monitoring log files, network activity and so forth. Several techniques for this type of monitoring have been discussed in previous slides. The second component, which can assist with the first, is interfering with the normal operation of an attacker's tools. The next several slides will discuss methods for doing each.

Exploits, root kits and other intrusion tools can be extremely clever and require a nontrivial understanding of C programming, networking and the inner workings of the Linux operating system to write. However, once a tool is written it is often packaged and distributed with the less technical user in mind. Many, probably the majority, of the people who break into computer systems are what the security community calls "script kiddies". A script kiddy is a person who uses exploits but does not write or even necessarily understand them. They simply run automated tools that look for known vulnerabilities, then run one script to exploit the vulnerability and another to install a root kit. The objective for script kiddies is often a feeling of power or daring, rather than any deep interest in the technology behind what they are doing. This can prove to be a significant advantage to the system administrator. If something is done to a system that the root kit or backdoor's installation script does not expect, then it may fail to install and the script kiddies may be thwarted. Even something as simple as restricting access to gcc, doing chattr +i on important files, chattr +a on logs or mounting /usr read-only can be powerful weapons if the opponent does not know Linux well enough to undo their effects!

There also exist tools for detecting root kits by looking for telltale strings and fragments within a compromised binary. These methods can detect both traditional root kits and more advanced LKM (loadable kernel module) based root kits. Root kit detectors evolve quickly to keep up with the latest intrusion tools and so are not practical to include in a distribution. A popular root kit detector, called **chkrootkit**, can be downloaded from <http://www.chkrootkit.org>.

## Detecting and Defeating Root Kits

- Special considerations for loadable kernel module (“LKM”) root kits
  - Can subvert processes without altering files
  - Can hide processes, services, and files from local tools
- Defenses against LKM root kits
  - Investigate from a trusted environment
  - Use root kit detection tools
  - Use a static kernel (not recommended)

13-15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email [ctraining@redhat.com](mailto:ctraining@redhat.com), or phone toll-free (USA) +1 (866) 626 2604 or +1 (919) 754 3700.

Traditional root kits have one very serious drawback: once suspected, they are relatively easy to detect. Because traditional root kits rely on modified system binaries, all a suspicious administrator has to do is copy the binary to a trusted system and use `cmp` or `md5sum` to detect any modifications.

A newer generation of root kits that rely on loadable kernel modules (LKMs) are much more difficult to detect. Because modules have direct and unlimited access to the kernel’s memory space, a hostile module can easily subvert the operating system’s functionality at its lowest level. A common tactic is for an LKM root kit to replace kernel system calls like `open()` and `stat()` with its own versions that omit or modify important data. Thus, it does not matter if a wary administrator uses only trusted binaries to check the running system, since the data is being altered at a lower level. LKM root kits avoid detection by tools like `md5sum`, `cmp` and `tripwire` for the same reason. Many LKM root kits can, however, be detected by special root kit detection tools as discussed previously.

It is possible to defeat many LKM-based root kits by compiling a static kernel with no LKM support. However, it is still possible for a root kit to subvert a static kernel by overwriting elements of the kernel in memory through `/dev/kmem`. (This is sometimes called “runtime kernel patching”.) Since Red Hat does not support custom kernels, and due to the inconveniences of working with static kernels, this defense is not recommended. If protection is still required, the Linux Intrusion Detection System (LIDS), might be a way to go. LIDS implements Mandatory Access Control, much like SELinux does. The web page for LIDS is <http://www.lids.org>. Another option could be to run User Mode Linux at <http://user-mode-linux.sourceforge.net>. UML is based on running virtual machines for each service which means that if one service is broken it will not affect any other service. SELinux, which is included in Red Hat Enterprise Linux 5, does a similar thing.

In short, if an attacker can compromise the system, cover his or her tracks, install an LKM root kit or employ runtime kernel patching as described above and be careful with how the system is subsequently used, then there is a very good chance that the intrusion will never be noticed. The best defense is, as always, to put up as many barriers as possible to prevent attackers from getting to that point in the first place.

## Recovering from an Intrusion

- Restore from a known-good backup
  - Do *not* attempt to repair the compromised system!
- Monitor the system for further attacks
  - The attacker may attempt to regain access
  - The attacker may succeed in regaining access

13-16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 628 2994 or +1 (919) 754 3700.

Once a compromise has been detected and verified, the best course of action to wipe the system immediately and reinstall it from a trusted backup. If you can determine the date of the attack, then you may be able to conclude that backups made prior to that are safe. However, also be aware that any backup from before the intrusion will include whatever security hole was used to compromise the machine in the first place. This is why determination of the cause of an intrusion is so important. Make sure that whatever hole was exploited, as well as any other outstanding security issues, have been fixed before exposing your system to the network again.

Once the system has been compromised, it is likely that the attacker will attempt to connect to it again. The restored system should be carefully watched to make sure that subsequent intrusion attempts by the attacker are unsuccessful.

## Reporting and Documenting the Intrusion

- How and when the intrusion was detected
- What actions were taken by whom
- What the nature of the intrusion was
- What was done to recover from the intrusion
  - Were policy changes needed?
  - Can you detect similar future intrusions like this?
- Notification of law enforcement if necessary
  - What evidence was gathered?

13-17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <[training@redhat.com](mailto:training@redhat.com)> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Documentation is the key to improving the team's performance in the face of future attacks. Has the team dealt with this type of attack before? If so, what did they do? If not, are there any symptoms or signs that are similar to a previous attack? Who has had the most first-hand experience with investigating this sort of potential intrusion?

After recovering from an intrusion it is important to note what parts of the policy worked and what did not. If the intrusion was detected by one of the secondary lines of defense (an internal firewall or IDS, for example), how did the intruder circumvent the first line (the firewall or IDS on your DMZ gateway)? What kind of changes can and should be made to the defenses to prevent it happening again?

Under certain circumstances it may be *required* to notify law enforcement of a compromise or other unauthorized activity that may have been uncovered lest the company become an accessory to crimes in which the systems are involved. Under other circumstances, such as when the intrusion is severe and beyond the team's ability to investigate properly, it may simply be a very good idea to bring in law enforcement. In the United States, the FBI may be able to image a machine and perform their investigation from those images without removing any computers from the premises. Consult with legal counsel or staff when developing a security policy and when investigating intrusions to determine appropriate actions.

## **End of Unit 13**

- Questions and Answers
- Summary
  - Intrusion risks
  - Developing a security policy
  - Investigating and documenting intrusions
  - Recovering from intrusions

**13-18**

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Training Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 826 2994 or +1 (919) 754 3700.

## **Lab 13**

### **Intrusion Detection and Recovery**

---

**Goal:** Install a trojaned version of /bin/ps and learn how to identify this intrusion.

**Estimated Duration:** 45 minutes

## **Sequence 1: Generate a md5 checksum database**

### **Instructions:**

1. Generata a list of md5 checksums from all your files under /bin.

For security reasons, this should normally be done direct after a clean install of your system. You should also select other directories which you wish to monitor (for example, /usr/sbin/).

Now you have a list of all checksums from all your files under /bin and /usr/sbin. You can use this list to identify possible changes or file replacements made by an intruder.

## **Sequence 2: Install a trojaned software on your system**

Instructions:

1. Install the `rhs333-ts` rpm from server1.

Congratulations! You have just installed a trojaned version of your `ps` command.

This version of `/bin/ps` will hide different processes which could be started by an intruder after he gained root access on your system. For example, a back-doored `sshd` or `login` program would not appear in the output from your `ps` command.

2. Run the `ps` command. Everything works as expected. So, if you don't have an IDS, it is difficult to notice that there have been such changes made to your system.

## Sequence 3: Create a snapshot

Instructions:

1. Run the **md5sum** command again against the directories you selected before and compare the output with your database file.

You should see, that there is an md5 checksum mismatch on the `/bin/ps` program!

This could be due to an update rpm you installed, but could also due to an installed trojan by an intruder. If you install new packages on your system you have to build a new md5checksum database as well. Otherwise, you may get a lot of false-positive alerts.

2. Remove the `rhs333-ts` rpm package and run the two commands above again. This time you should not be warned by **diff** that there are any differences between your original database file and the newly created snapshot. Removing the `rhs333-ts` package put the original **ps** command in place again.
3. You can write a small script which creates daily or hourly snapshots from your system and automatically compares these snapshots with the original database you created in Sequence 1. Let this script run daily or hourly, as you like, by cron. Every time **diff** notices a difference between the database and the snapshot, it should sent you an email alert.
4. Voila. You have build a simple host-based Intrusion Detection System.

## **Sequence 4: Use the rpm database to determine changes**

Instructions:

1. Install the `rhs333-ts` rpm package again as described previously in Sequence 2.

2. Run the following command:

```
[root@stationX]# rpm -V -f /bin/ps
```

3. You should see something like:

```
SM5....T. /bin/ps
```

4. Read the `rpm` man page and search for the option `-V` or `--verify`. What does the output above mean?

5. It is usually a good idea to make a backup of your compromised system. Work with a colleague to take a backup of your system. On stationX, use the `cat` command over `ssh` to take a backup of your boot partition. Make a backup partition (make sure it is big enough!) and mount it under `/backup`. Then back it up!

```
[root@stationX]# ssh stationY "cat /dev/bootpart" > /backup/stationY-bootpart.img
```

6. Mount the new image using the loopback device and investigate the image:

```
[root@stationY]# mkdir /mnt/backup  
[root@stationY]# mount -o loop /backup/stationY-bootpart.img /mnt/backup
```

Don't forget to remove the `rhs333-ts` rpm again!

## **Sequence 1 Solutions**

1. Generate a list of md5 checksums from all your files under /bin:

```
# md5sum /bin/* > /root/files_md5sum 2> /dev/null
```

For security reasons, this should normally be done direct after a clean install of your system. You should also select other directories which you wish to monitor (for example, /usr/sbin/).

```
# md5sum /usr/sbin/* >> /root/files_md5sum 2> /dev/null
```

## **Sequence 2 Solutions**

1. Install the rhs333-ts rpm from server1.

```
# yum install rhs333-ts
```

## **Sequence 3 Solutions**

1. Run the **md5sum** command again against the directories you selected before and compare the output with your database file.

```
# md5sum /bin/* /usr/sbin/* > "/root/files_md5sum.$(date -v +%Y%m%d)" 2> /dev/null  
# diff files_md5sum "files_md5sum.$(date +%Y%m%d)"
```