Red Hat System Administration III
Student Workbook
Red Hat Enterprise Linux 6
Release 2-20110124

RED HAT SYSTEM ADMINISTRATION III

Red Hat Enterprise Linux 6 RH255
Red Hat System Administration III
Edition 2

Author	Forrest Taylor
Author	George Hacker
Author	David Duffey
Author	Joshua Hoffman
Author	Robert Locke
Editor	Steven Bonneville
Editor	Mark Howson

Contributors: Andrew Dingman

Document Conventions

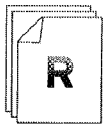
Notes and Warnings

Note

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

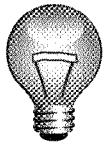
Comparison

"Comparisons" look at similarities and differences between the technology or topic being discussed and similar technologies or topics in other operating systems or environments.



References

"References" describe where to find external documentation relevant to a subject.



Important

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.

Warning

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.

Introduction

Welcome to class!

Thank you for attending this Red Hat training class. Please let us know if you have any special needs while at our training facility.

Please ask the instructor if you have any questions about the facility, such as operating hours of the facility and when you will have access to the classroom, locations of restrooms and break rooms, availability of telephones and network connectivity, and information about the local area.

As a courtesy to other students, please place your pager or cell phone's ringer on vibrate or mute, or turn off your devices during class. We ask that you only make calls during break periods.

If you have a personal emergency and are unable to attend or complete the class, please let us know. Thank you!

About Red Hat Enterprise Linux

This course is taught using Red Hat Enterprise Linux, an enterprise-targeted Linux distribution focused on mature open source software designed specifically for organizations using Linux in production settings.

Red Hat Enterprise Linux is sold on a subscription basis, where the subscription gives you continues access to all supported versions of the operating system in binary and source form, not just the latest one, including all updates and bug fixes. Extensive support services are included: a support contract and Update Module entitlement to Red Hat Network are included for the subscription period. Various Service Level Agreements are available that may provide up to 24x7 coverage with a guaranteed one hour response time for Severity 1 issues. Support will be available for up to seven years after a particular major release (ten years with the optional "Extended Update Support" Add-On).

Red Hat Enterprise Linux is released on a multi-year cycle between major releases. Minor updates to major releases are released roughly every six months during the lifecycle of the product. Systems certified on one minor update of a major release continue to be certified for future minor updates of the major release. A core set of shared libraries have APIs and ABIs which will be preserved between major releases. Many other shared libraries are provided, which have APIs and ABIs which are guaranteed within a major release (for all minor updates) but which are not guaranteed to be stable across major releases.

Red Hat Enterprise Linux is based on code developed by the open source community, which is often first packaged through the Red Hat sponsored, freely-available Fedora distribution (`http://fedoraproject.org/`). Red Hat then adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors. Red Hat Enterprise Linux provides a high degree of standardization through its support for four processor architectures (32-bit Intel x86-compatible, AMD64/Intel 64 (x86-64), IBM POWER, and IBM mainframe on System z). Furthermore, we support the 4000+ ISV certifications on Red Hat Enterprise Linux whether the RHEL operating system those applications are using

is running on "bare metal", in a virtual machine, as a software appliance, or in the cloud using technologies such as Amazon EC2.

Currently, the Red Hat Enterprise Linux product family includes:

- *Red Hat Enterprise Linux for Servers*: the datacenter platform for mission-critical servers running Red Hat Enterprise Linux. This product includes support for the largest x86-64 and x86-compatible servers and the highest levels of technical support, deployable on bare metal, as a guest on the major hypervisors, or in the cloud. Subscriptions are available with flexible guest entitlements of one, four, or unlimited guests per physical host. Pricing is based on the basis of the number of socket-pairs populated on the system motherboard, the number of guests supported, the level of support desired, and the length of subscription desired.

  Red Hat Enterprise Linux for IBM POWER and *Red Hat Enterprise Linux for IBM System z* are similar variants intended for those system architectures.

- *Red Hat Enterprise Linux Desktop*: built for the administrator and end-user, Red Hat Enterprise Linux Desktop provides an attractive and highly productive environment for knowledge workers on desktops and laptops. Client installations can be finely tailored and locked down for simplicity and security for any workstation task.

  The basic *Desktop* variant is designed for task workers who have a limited amount of administrative control over the system, who primarily use productivity applications like Firefox Evolution/Thunderbird, OpenOffice.org, and Planner/TaskJuggler. The more sophisticated *Workstation* variant is designed for advanced Linux users who need a stand-alone development environment, and who are expected to have local super-user privileges or selected super-user privileges.

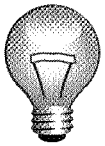
In addition, other variants exist such as *Red Hat Enterprise Linux for HPC Head Node* and *Red Hat Enterprise Linux for HPC Compute Node* (targeted at high-performance computing clusters), and *Red Hat Enterprise Linux for SAP Business Applications*. For more information please visit `http://www.redhat.com/`.

Additional Red Hat Enterprise Linux Software

Two additional software update channels are provided with Red Hat Enterprise Linux beyond the core software packages shipped:

- *Supplementary*: the "Supplementary" channel provides selected closed source packages, built for Red Hat Enterprise Linux as a convenience to the customer. These include things like Adobe Flash or proprietary Java JVMs.

- *Optional*: the "Optional" channel provides selected open source packages, as a convenience only. They are generally included in another Red Hat Enterprise Linux variant as a fully-supported package, or are a build requirement for the distribution. These packages are only available through a Red Hat Network child channel.



Important

Supplementary and *Optional* packages are provided with limited support, as a customer convenience only.

Red Hat also offers a portfolio of fully-supported *Add-Ons for Red Hat Enterprise Linux* which extend the features of your Red Hat Enterprise Linux subscription. These add-ons allow you to add capabilities and tailor your computing environment to your particular needs. These Add-Ons include support for high availability application clustering, cluster file systems and very large file systems, enhanced system management with Red Hat Network, extended update support, and more.

Note

Please visit `http://www.redhat.com/rhel/add-ons/` for more information about available *Add-Ons for Red Hat Enterprise Linux*.

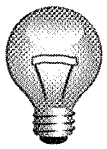
For information about other products which are provided by Red Hat, such as Red Hat Enterprise Virtualization, JBoss Enterprise Middleware, Red Hat Enterprise MRG, and various custom consulting and engineering services, `http://www.redhat.com/products/` also has useful information.

The Fedora Project also provides additional packages for Red Hat Enterprise Linux through *EPEL (Extra Packages for Enterprise Linux)*. EPEL is a volunteer-based community effort to create a repository of high-quality add-on packages which can be used with Red Hat Enterprise Linux and compatible derivatives. It accepts legally-unencumbered free and open source software which does not conflict with packages in Red Hat Enterprise Linux or Red Hat add-on products. EPEL packages are built for a particular major release of Red Hat Enterprise Linux and will be updated by EPEL for the standard support lifetime of that major release.

Red Hat does not provide commercial support or service level agreements for EPEL packages. While not supported officially by Red Hat, EPEL provides a useful way to reduce support costs for unsupported packages which your enterprise wishes to use with Red Hat Enterprise Linux. EPEL allows you to distribute support work you would need to do by yourself across other organizations which share your desire to use this open source software in RHEL. The software packages themselves go through the same review process as Fedora packages, meaning that experienced Linux developers have examined the packages for issues. As EPEL does not replace or conflict with software packages shipped in RHEL, you can use EPEL with confidence that it will not cause problems with your normal software packages.

For developers who wish to see their open source software become part of Red Hat Enterprise Linux, often a first stage is to sponsor it in EPEL so that RHEL users have the opportunity to use it, and so experience is gained with managing the package for a Red Hat distribution.

Visit `http://fedoraproject.org/wiki/EPEL/` for more information about EPEL.



Important

EPEL is supported by the community-managed Fedora Project and not by Red Hat Support.

Contacting Red Hat Technical Support

One of the benefits of your subscription to Red Hat Enterprise Linux is access to technical support through Red Hat's customer portal at `http://access.redhat.com/`. If you do not have a Red Hat account on the customer portal or are not able to log in, you can go to `https://access.redhat.com/support/faq/LoginAssistance.html` or contact Customer Service for assistance.

You may be able to resolve your problem without formal technical support by searching Knowledgebase (`https://access.redhat.com/kb/knowledgebase/`). Otherwise, Red Hat Support may be contacted through a web form or by phone depending on your support level. Phone numbers and business hours for different regions vary; see `https://access.redhat.com/support/contact/technicalSupport.html` for current information. Information about the support process is available at `https://access.redhat.com/support/policy/support_process.html`.

Some tips on preparing your bug report to most effectively engage Red Hat Support:

- *Define the problem.* Make certain that you can articulate the problem and its symptoms before you contact Red Hat. Be as specific as possible, and detail the steps you can use (if any) to reproduce the problem.
- *Gather background information.* What version of our software are you running? Are you using the latest update? What steps led to the failure? Can the problem be recreated and what steps are required? Have any recent changes been made that could have triggered the issue? Were messages or other diagnostic messages issued? What *exactly* were they (exact wording may be critical)?
- *Gather relevant diagnostic information.* Be ready to provide as much relevant information as possible; logs, core dumps, traces, the output of **`sosreport`**, etc. Technical Support can assist you in determining what is relevant.
- *Determine the Severity Level of your issue.* Red Hat uses a four-level scale to indicate the criticality of issues; criteria may be found at `https://access.redhat.com/support/policy/GSS_severity.html`.

Warning

Bugzilla is not a support tool! For support issues affecting Red Hat Enterprise Linux, customers should file their bugs through the support channels discussed above in order to ensure that Red Hat is fully aware of your issue and can respond under the terms of your Service Level Agreement. Customers should *not* file bugs directly in the `http://bugzilla.redhat.com/` web interface.

For Red Hat Enterprise Linux, Bugzilla is used by engineering to track issues and changes, and to communicate on a technical level with Engineering partners and other external parties. Anyone, even non-customers, can file issues against Bugzilla, and Red Hat does monitor them and review them for inclusion in errata.

However, Red Hat does not guarantee any SLA for bugs filed directly in Bugzilla (bypassing normal support channels). A review might happen immediately, or after a time span of any length. Issues coming through Support are always prioritized above issues of similar impact and severity filed against Bugzilla. Also, work arounds and hotfixes if possible and appropriate may be provided to customers by Support even before a permanent fix is issued through Red Hat Network.

Red Hat considers issues directly entered into Bugzilla important feedback, and it allows us to provide efficient interaction with the open source development community and as much transparency as possible to customers as issues are processed. Nevertheless, for customers encountering production issues in Red Hat Enterprise Linux, Bugzilla is not the right channel.

About This Course

Red Hat System Administration III

Red Hat System Administration III (RH255) is designed for experienced Linux system administrators with Red Hat Certified System Administrator (RHCSA) certification or equivalent skills who want to broaden their skill set. This course is focused on enhancing the student's automation skills while securely deploying and managing network services such as BIND DNS, Apache HTTP, Postfix SMTP, and network file sharing. This course places an emphasis on security, including monitoring, packet filtering, access controls, and SELinux. Students already familiar with RHCSA-level administration tasks will leave this course with exposure to all competencies tested by the RHCE exam.

Objectives

- To train students in the skills needed to be a senior Linux system administrator responsible for managing network services and server security
- To prepare students to validate those skills in the RHCE exam

Audience and Prerequisites

- Students who are veteran Linux administrators wishing to learn additional automation techniques, how to deploy and secure key network services, and how to manage other key security features of Red Hat Enterprise Linux
- Students who have taken *Red Hat System Administration I* and *Red Hat System Administration II*; or who have taken *RHCSA Rapid Track Course*; or who have either the RHCT or RHCSA certification; or who have equivalent skills

Structure of the Course

Red Hat training courses are interactive, hands-on, performance-based, real world classes meant to engage your mind and give you an opportunity to use real systems to develop real skills. We encourage students to participate in class and ask questions in order to get the most out of their training sessions.

This course is divided up into a number of *Units* organized around a particular topic area. Each Unit is divided up into multiple *Sections* which focus on a specific skill or task. The unit will start with an introduction to the material, then move on to the first section.

In each section, there will be a *presentation* led by the instructor. During the presentation, it may be a good idea to take notes in your student workbook (this book), and the instructor may remind you to do so. The presentation is followed by a short activity or *assessment* to give you the opportunity to practice with the material or review procedures. After a review of the assessment, the instructor will move on to the next section. At the end of the unit, there will normally be a hands-on lab exercise of some sort (a "*criterion test*") which will give you an opportunity to learn by doing and review your understanding of the unit's content. Please feel free ask questions in

class, or asking the instructor for advice and help during the end-of-unit exercise. We want the classroom environment to be a "low risk" place where you feel comfortable asking questions and learning from things that work and things that do not at first.

Orientation to the Classroom Network

Two subnets may be used in this course. The primary classroom network is 192.168.0.0/24, and belongs to hosts in the DNS domain "example.com". This network will be used for most classroom activities. Some courses use a second subnet, 192.168.1.0/24, belonging to hosts in the DNS domain "remote.test". This network can be reached from hosts in example.com, and is used in lab exercises which require testing services or security settings from machines (theoretically) outside your administrative control.

Students are each assigned a physical machine (desktop*X*.example.com on 192.168.0.*X*) which may host two or more virtual machines for lab activities, server*X*.example.com and host*X*.example.com.

In some courses, students may also use a non-root account on a test machine in the remote.test domain, remote*X*.example.com (192.168.1.*X*) to test access to network services on their example.com machines in lab activities.

The instructor controls a number of machines which students may see as well. The machine instructor.example.com (also known as instructor.remote.test) is the classroom utility server, providing default routing services, DHCP, DNS name service, one or more YUM repositories of software used by the class, and other network services. It is also connected to the classroom video projector to allow the instructor to display slides and demonstrations. It provides a virtual machine for the instructor, demo.example.com, which the instructor will use for in-class demonstrations.

Machine name	IP addresses	Role
desktop*X*.example.com	192.168.0.*X*	Physical student workstation
server*X*.example.com	192.168.0.*(X+100)*	Main student virtual machine
host*X*.example.com	192.168.0.*(X+200)*	Secondary student virtual machine
remote*X*.remote.test	192.168.1.*X*	Student test machine in remote.test domain (shared)
instructor.example.com	192.168.0.254	Physical instructor machine and utility server
instructor.remote.test	192.168.1.254	Identity of instructor.example.com on remote.test network
demo.example.com	192.168.0.250	Instructor virtual demonstration machine

Table 1. Classroom Machines

Internationalization

Language Support

Red Hat Enterprise Linux 6 officially supports twenty-two languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Oriya, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu. Support for Maithili, Nepalese, and Sinhala are provided as Technology Previews.

System-wide Default Language

The operating system's default language is normally set to US English (en_US.UTF-8), but this can be changed during or after installation.

To use other languages, you may need to install additional package groups to provide the appropriate fonts, translations, dictionaries, and so forth. By convention, these package groups are always named ***language*-support**. These package groups can be selected during installation, or after installation with PackageKit (**System** → **Administration** → **Add/Remove Software**) or **yum**.

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

Per-user Language Selection

Users may prefer to use a different language for their own desktop environment or interactive shells than is set as the system default. This is indicated to the system through the **LANG** environment variable.

This may be set automatically for the GNOME desktop environment by selecting a language from the graphical login screen by clicking on the **Language** item at the bottom left corner of the graphical login screen immediately prior to login. The user will be prompted about whether the language selected should be used just for this one login session or as a default for the user from now on. The setting is saved in the user's **~/.dmrc** file by GDM.

If a user wants to make their shell environment use the same **LANG** setting as their graphical environment even when they login through a text console or over **ssh**, they can set code similar to the following in their **~/.bashrc** file. This code will set their preferred language if one is saved in **~/.dmrc** or will use the system default if one is not:

```
i=$(grep 'Language=' ${HOME}/.dmrc | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

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:

```
[user@host ~]$ LANG=fr_FR.UTF-8 date
lun. oct. 24 10:37:53 CDT 2011
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to check the current value of **LANG** and other related environment variables.

Input Methods

IBus (Intelligent Input Bus) can be used to input text in various languages under X if the appropriate language support packages are installed. You can enable IBus with the **im-chooser** command (**System** → **Preferences** → **Input Method**).

Language Codes Reference

Language	$LANG value	Language package group
English (US)	en_US.UTF-8	*(default)*
Assamese	as_IN.UTF-8	assamese-support
Bengali	bn_IN.UTF-8	bengali-support
Chinese (Simplified)	zh_CN.UTF-8	chinese-support
Chinese (Traditional)	zh_TW.UTF-8	chinese-support
French	fr_FR.UTF-8	french-support
German	de_DE.UTF-8	german-support
Gujarati	gu_IN.UTF-8	gujarati-support
Hindi	hi_IN.UTF-8	hindi-support
Italian	it_IT.UTF-8	italian-support
Japanese	ja_JP.UTF-8	japanese-support
Kannada	kn_IN.UTF-8	kannada-support
Korean	ko_KR.UTF-8	korean-support
Malayalam	ml_IN.UTF-8	malayalam-support
Marathi	mr_IN.UTF-8	marathi-support
Oriya	or_IN.UTF-8	oriya-support
Portuguese (Brazilian)	pt_BR.UTF-8	brazilian-support
Punjabi	pa_IN.UTF-8	punjabi-support
Russian	ru_RU.UTF-8	russian-support

Language	$LANG value	Language package group
Spanish	es_ES.UTF-8	spanish-support
Tamil	ta_IN.UTF-8	tamil-support
Telugu	te_IN.UTF-8	telugu-support
Technology Previews		
Maithili	mai_IN.UTF-8	maithili-support
Nepali	ne_NP.UTF-8	nepali-support
Sinhala	si_LK.UTF-8	sinhala-support

Table 2. Language Codes

UNIT ONE

GETTING STARTED WITH THE CLASSROOM ENVIRONMENT

Introduction

Topics covered in this unit:

- Virtualization tools
- Review of select System Administration I and System Administration II topics

Virtualization Tools & Review

In this unit, we will review prerequisite knowledge from the System Administration I and System Administration II courses. We will also introduce virtualization tools we will use throughout the course. Ask questions and take notes here as there will be a quiz on these topics after the discussion.

Virtual Machine Tools

- **Virtual Machine Manager** is the graphical tool used to manage virtual machines. You can run the tool from **Applications** → **System Tools** → **Virtual Machine Manager**, or by running the **`virt-manager`** command from the command-line.

  This interface allows you to power on or power off virtual machines, assign memory and CPU resources, monitor performance, and connect to the console of the virtual machines.

- The **`virsh`** command-line tool also allows you to manage your virtual machines. It provides the same fuctionality as **`virt-manager`**.

  Running **`virsh`** by itself will place you in a **`virsh`** command shell where you can run commands like **`start`**, **`shutdown`**, **`destroy`**, etc. Run **`help`** for more information on any of these subcommands.

  Examples:

```
[root@desktopX ~]# virsh list
 Id Name                 State
----------------------------------
1   vserver              running

[root@desktopX ~]# virsh destroy vserver
[root@desktopX ~]# virsh list --all
 Id Name                 State
----------------------------------
-   vserver              shut off

[root@desktopX ~]# virsh start vserver
[root@desktopX ~]# virsh list
 Id Name                 State
----------------------------------
1   vserver              running
```

System Boot Process and Runlevels

- Interrupting GRUB

  To interrupt GRUB, reboot the machine and watch for the GRUB screen just after the BIOS screen. Press the **Esc** key to interrupt the countdown. Use the up and down arrow keys to select a kernel. Press **a** to append kernel arguments or press **e** to edit any of the lines in the stanza. If you changed any lines, press **Enter** to accept those changes (**Esc** will remove your changes and go back to the original settings), and press **b** to boot.

- Specifying single-user mode

To specify single-user mode, interrupt GRUB, choose the kernel you would like to use, then press **a**. Append a **Space** followed by **s** and press **Enter**. Press **b** to boot.

LDAP User Authentication

- **system-config-authentication**

Use **system-config-authentication** to configure authentication, including LDAP. To configure an LDAP client for this course, you will need the following items:

1. LDAP URL: **ldap://instructor.example.com**
2. LDAP Search Base DN: **dc=example,dc=com**
3. Use TLS to encrypt connections and the Certificate Authority (CA) Certificate from: **ftp://instructor.example.com/pub/example-ca.crt-CA-CERT**
4. Authentication Method. Choose either **Kerberos password** and complete the Kerberos information, or choose **LDAP password** to use the LDAP server specified above to provide passwords.

Automount Home Directories

- **/etc/auto.master**

/etc/auto.master provides the master configuration for **autofs**. The first column provides the mount point, the second column provides the map (a file name in this course) and the third column provides the options. If the third column is blank, **autofs** will use the default mount options.

Example:

```
/home/guests /etc/auto.guests
```

- **/etc/auto.guests**

The **/etc/auto.guests** file was specified in **/etc/auto.master**, so this file must be created. This file can also has three columns. The first column provides the local mount point (relative to the top-level directory listed in **/etc/auto.master**), the second column provides the options, and the third column provides the remote filesystem to mount. If the second column is blank, **autofs** will use the default mount options.

Example using LDAP user home directories:

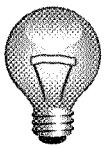
```
ldapuser1 -rw instructor.example.com:/home/guests/ldapuser1
ldapuser2 -rw instructor.example.com:/home/guests/ldapuser2
ldapuser3 -rw instructor.example.com:/home/guests/ldapuser3
ldapuser4 -rw instructor.example.com:/home/guests/ldapuser4
...
```

We can use wildcards and metacharacters to shorten this file. Notice the pattern: only the first column and the last entry of the last column change. We can use * to match any entry in the

first column, and **&** to reuse whatever we matched in the first column. Since **rw** is one of the default mount options, we can actually remove that as well and we get a single entry:

```
*   instructor.example.com:/home/guests/&
```

- **autofs** service



Important

At the time of writing, there is a bug in the **autofs** init script (bugzilla #624444) that causes **service autofs restart** to sometimes fail. Until this is fixed, use

service autofs reload

or

service autofs stop && service autofs start.

SSH User Authentication and Control

- SSH key pairs

The Secure Shell, **ssh**, allows you to authenticate using a private-public key scheme. This means that you generate two keys, called your private key and your public key. The private key should, as the name implies, be kept private. The public key can be given to anyone. An SSH server that has your public key can issue a challenge that can only be answered by a system holding your private key. As a result, you can authenticate using the presence of your key. This allows you to access systems in a way that does not require typing a password every time but is still secure.

- **ssh-keygen**

Key generation is done using the **ssh-keygen** command. You can use a key type of DSA or RSA with SSH version 2. SSH protocol version 1 is known to have a security flaw, and therefore its use is not recommended unless you need to connect to legacy SSH servers.

During key generation, you will be given the option to specify a passphrase, which must be provided in order to access your private key. This way, even if the key is stolen, it is very difficult for someone other than you to use it. This gives you time to make a new key pair and remove all references to the old ones, before the private key can be used by an attacker who has cracked it.

It is always wise to passphrase-protect your private key since the key allows you to access other machines. However, this means that you must type your passphrase whenever the key is used, making the authentication process no longer password-less. This can be avoided using **ssh-agent**, which can be given your passphrase once at the start of your session (using **ssh-add**) so it can provide it as necessary while you stay logged in.

Once your SSH keys have been generated, they are stored by default in the **.ssh/** directory of your home directory. Default modes should be 600 on your private key and 644 on your public key.

- **ssh-copy-id**

Before you can use key-based authentication, you will need to copy your public key to the destination system. This can be done with **ssh-copy-id**.

```
[student@desktopX ~]$ ssh-copy-id -i .ssh/id_rsa.pub root@desktopY
```

When you copy your key to another system via **ssh-copy-id**, it uses the **~/.ssh/id_rsa.pub** file by default. If you use a different key, or give your key a different name, it will have to be specified with the **-i** option when using **ssh-copy-id**.

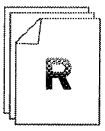
- **/etc/ssh/sshd_config**

The **/etc/ssh/sshd_config** file is used to manage the SSH service. For example, you can prevent **root** SSH logins or prevent password SSH logins (only allowing SSH keys) in this file.

ISET

ISET was a memorization aid we used in earlier classes to help you remember the basic steps needed to successfully deploy system and network services. The four steps of ISET are:

1. I_____
2. S_____
3. E_____
4. T_____



References

Red Hat Enterprise Linux Virtualization
- Chapter 31: Managing guests with the Virtual Machine Manager (virt-manager)

Red Hat Enterprise Linux Storage Administration Guide
- Section 11.3: **autofs**

autofs(5), **auto.master**(5) man pages

Red Hat Enterprise Linux Deployment Guide
- Section 8.2.4: Using a Key-Based Authentication

Practice Quiz

System Administration Review

1. The _____ command will run the Virtual Machine Manager GUI, and _____ is the command line equivalent.

2. Press _____ to release the pointer in the virtual machine graphical console.

3. Passing the argument _____ to the kernel at boot executes the boot process up to the end of /etc/rc._____ and then opens a root shell without asking for a password.

4. In Red Hat Enterprise Linux 6 the System Security Services Daemon _____ manages access to remote directories and authentication mechanisms for clients.

5. The _____ command provides a GUI to configure user authentication.

6. In order to trust an LDAP server we use a _____ file to verify the TLS / SSL connection.

7. Run "service _____ reload" to have the automounter reload its main configuration file /etc_____.

8. An _____ map is specified in the /etc/auto.master file by giving a directory which will contain mount points and the name of a second file containing information about the mount points managed in that directory.

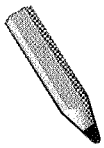
9. An automounter map file can use the _____ wildcard character for the key and use the match in the source path with the _____ character.

10. The command to generate a new ssh key pair is _____.

11. The command _____ will copy the user's public key into a ~/.ssh_____ file.

12. To allow only certain usernames to login via ssh, edit /etc_____ and add the _____ directive, or to restrict root add _____ no.

13. Run "yum _____ httpd" to install Apache, "_____ httpd _____" to start it, and "_____ httpd _____" to enable it to persist across reboots.



Test

Criterion Test

Case Study

Getting Started with the Classroom Environment

Before you begin...

Run **`lab-setup-server`** on desktopX to prepare serverX for the exercise.

Duffey Dynamics Institute, a research firm investigating deep water human habitation, recently dismissed their System Administrator. Several new Red Hat Enterprise Linux servers are installed but not configured for use.

One of the new systems, your serverX, is needed to provide services to the new Underwater Farming Lab.

The following services are needed:

- Configure serverX to authenticate users via the main company LDAP server using TLS.
  - LDAP Server: instructor.example.com
  - LDAP Search Base DN: **`dc=example,dc=com`**
  - LDAP Certificate: **`ftp://instructor.example.com/pub/example-ca.crt`**
  - Use LDAP Authentication
- Configure serverX to automount user home directories on demand via NFS.
  - NFS Server: instructor.example.com
  - NFS Export: **`/home/guests/`*`username`***
- The lab also needs a web server, so install Apache and set it to start at boot.
- Lastly, lock down the SSH server so that root cannot login.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Personal Notes

Unit Summary

Virtualization Tools & Review

Since you've completed this unit, you are able to:

- Manage virtual systems with **virsh**
- Recover the root password using single user mode
- Use LDAP authentication with **system-config-authentication** and **sssd**
- Automount NFS-based home directories
- Use **ssh-keygen** and **ssh-copy-id**
- Limit SSH users via **/etc/ssh/sshd_config**
- Install, start, enable and test (ISET) Apache

UNIT TWO

ENHANCE USER SECURITY

Introduction

Topics covered in this unit:

- Escalation via **sudo**
- Kerberos authentication
- Troubleshooting with SSSD

Configuring sudo

The **sudo** command allows a user to be permitted to run a command as root or as another user, based on settings in the **/etc/sudoers** file. This file is updated with the **visudo** command, which ensures that the file is protected against simultaneous edits, meets various sanity checks, and that there are no basic parsing errors.

Search & Learn: Configuring sudo

Consult the reference documents below to answer the following questions. The instructor will review the solutions (as found in the appendix) with the class when the activity is complete.

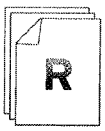
1. What is the basic syntax of a sudoers rule (a *user specification* that controls which commands a user may run)?

2. What sudoers entry would grant members of the *wheel* group access to any command?

3. How do you prevent a user from being prompted by **sudo** for their own password?

4. How would you grant a list of users (who do not share a common group) **sudo** access to any command?

5. How could you set up sudoers entries to grant a list of users **sudo** access to a specific list of commands (rather than to all commands)?



References

sudoers(5), **sudo**(8), and **visudo**(8) man pages

/etc/sudoers contains sample **sudoers** recipes

Practice Resequencing Exercise

sudo Rules Syntax

For each of the five requirements below, write down the number of the matching rule on the line in front of the definition.

___ Grants members of *wheel* group access to any command

___ Associates users *bob*, *mary*, *david*, and *george* with the name **ADMINS**

___ Grants user *joseph* access to commands in the **NETWORKING** group

___ Grants user *natalie* access to **SOFTWARE** commands without needing her password

___ Grants the *dba* group access to the **service** and **chkconfig** commands for the **mysql** service

1. `%wheel  ALL=(ALL) NOPASSWD:ALL`
2. `joseph  ALL=  NETWORKING`
3. `User_Alias ADMINS = bob, mary, david, george`
4. `%joseph  ALL= /sbin/ip, /bin/ping, /sbin/ifconfig, /sbin/route`
5. `%wheel  ALL=  ALL`
6. `natalie  ALL=(ALL) SOFTWARE`
7. `%dba  /sbin/service mysql, /sbin/chkconfig mysql ALL`
8. `%dba  ALL= /sbin/service mysql, /sbin/chkconfig mysql`
9. `natalie  ALL=  NOPASSWD: SOFTWARE`

Kerberos Configuration

Kerberos is a method of secure authentication over an insecure network which was originally developed at MIT. Kerberos authenticates users without passing passwords over the network, protecting the integrity of the password and preventing its capture by network sniffers. Instead, it passes *tickets* which are encrypted using the user's password as an encryption key. When a user performs initial authentication to a system using a Kerberos password, the login program askes the Kerberos authentication server, or *key distribution center*, for a ticket for that user. The KDC sends the login program a ticket for that user, and if they type in the password that decrypts the ticket, the user has successfully authenticated.

We will examine the steps and key configuration elements of pointing a system to an existing Kerberos realm.

Instead of using **system-config-authentication** to make changes, we will focus on the command-line tool, **authconfig**. Take notes on needed options with this tool as they are presented.

- Kerberos Realm - The set of machines that all use the same KDCs (Kerberos authentication servers) for authentication

- Key Distribution Center (KDC) - Central servers that store information about Kerberos passwords and issue Kerberos tickets (authentication credentials)

- Kerberos Admin Server - Servers that allow remote administration (for example, these servers are used to update passwords). Normally the master KDC is the admin server for the realm.

Kerberos Configuration Demonstration

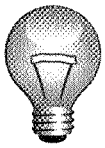
```
[root@serverX ~]# yum groupinstall -y directory-client
[root@serverX ~]# yum install -y openldap-clients
[root@serverX ~]# yum install -y krb5-workstation
[root@serverX ~]# authconfig --enableldap --ldapserver=instructor.example.com
 --enableldaptls --ldaploadcacert=ftp://instructor.example.com/pub/example-
ca.crt --ldapbasedn="dc=example,dc=com" --disableldapauth --enablekrb5 --
krb5kdc=instructor.example.com --krb5adminserver=instructor.example.com --
krb5realm=EXAMPLE.COM --enablesssd --enablesssdauth --update
```

Note

You must explicitly disable LDAP authentication when using the command line. Also, we will explicitly enable SSSD (explained later), to mirror what the graphical tool, **system-config-authentication** does.

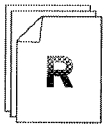
Test the configuration:

```
[root@serverX ~]# getent passwd ldapuserX
ldapuserX:*:17XX:17XX:LDAP Test User X:/home/guests/ldapuserX:/bin/bash
[root@desktopX ~]# ssh ldapuserX@serverX
ldapuserX@serverX's password: kerberos
```



Important

In the classroom, when using Kerberos, the password for ldapuserX is **kerberos**. When using LDAP authentication, the password for ldapuserX is **password**.



References

Red Hat Enterprise Linux Deployment Guide

- Section 8.1: The Authentication Configuration Tool

system-config-authentication(8), **kerberos**(1), and **sssd**(8) man pages

Practice Performance Checklist

Kerberos Configuration Exercise

You will modify your previous LDAP-based configuration to now use only Kerberos for authentication. LDAP will still be used to provide account information.

Perform the following steps on serverX unless directed otherwise.

- ☐ Verify the necessary packages are installed for Kerberos authentication.

- ☐ Configure system to use the following LDAP and Kerberos settings:
  - LDAP Server: instructor.example.com (uses TLS)
  - LDAP Certificate: `ftp://instructor.example.com/pub/example-ca.crt`
  - LDAP Base DN: dc=example,dc=com
  - Kerberos Realm: EXAMPLE.COM
  - Kerberos KDC: instructor.example.com
  - Kerberos Admin Server: instructor.example.com
  - Be sure the **sssd** service is enabled
- ☐ Test the change by logging in to serverX with ssh:
  - Username: **ldapuserX** (where X is your station number)
  - Password: **kerberos**

Troubleshooting System Security Services Daemon (SSSD)

Authentication Troubleshooting

As the instructor answers each of these questions, take notes in the space provided:

1. What is the drawback of using LDAP or Kerberos to authenticate desktop or laptop users versus locally-defined user accounts?

   When the system is "offline", credentials cannot be verified with the online services (LDAP or Kerberos), thus preventing login.

2. What can be implemented to resolve this problem?

3. How does one configure SSSD?

4. What if I want to configure SSSD from the command line?

5. How will this affect troubleshooting the authentication process?

6. Where can we see what the SSSD service is doing?

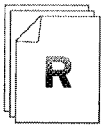
   Troubleshooting sssd:

   - *Look in* **`/var/log/sssd/`** *for logs.*

   - *Modify* **`/etc/sssd/sssd.conf`** *to increase information logged:*

   ```
   debug_level=10
   ```

7. What if I cannot log in to view the log files or correct an authentication misconfiguration?



References

Red Hat Enterprise Linux Deployment Guide

- Section 8.2: The System Security Services Daemon (SSSD)

sssd.conf(5), **sssd-krb5**(5) **sssd-ldap**(5) man pages

Practice Quiz

Troubleshooting Authentication Quiz

1. How does one normally configure SSSD?

   _____

2. Which directory holds log messages from sssd?

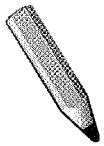
   _____

3. How can we increase the logging detail that is generated?

   _____

4. When you cannot log in to correct an authentication misconfiguration, what approaches are available to you?

   _____



Test

Criterion Test

Case Study

Enhance User Security

Before you begin...

Run **lab-setup-taylorlocke** on desktopX to prepare serverX for the exercise.

Taylor and Locke, a prestigious law firm, recently hired a security consultant to advise them regarding their servers. As the law firm's servers hold sensitive client information, security is a priority!

The security consultant recommended that all servers use LDAP for centralized accounts and Kerberos for authentication. Overall, the LDAP/Kerberos deployment went well. However, one of the servers that you manage appears to be mis-configured.

Correct the configuration on serverX so that LDAP users are able to login with Kerberos authentication (details below).

- LDAP Server: **instructor.example.com** (uses **TLS**)
- LDAP Certificate: **ftp://instructor.example.com/pub/EXAMPLE-CA-CERT**
- LDAP Base DN: **dc=example,dc=com**
- Kerberos Realm: **EXAMPLE.COM**
- Kerberos KDC: **instructor.example.com**
- Kerberos Admin Server: **instructor.example.com**

Test the change by logging in to serverX with ssh:

- Username: **ldapuserX** (where *X* is your station number)
- Password: **kerberos**

Once LDAP users can login, configure autofs to provide automounted home directories. The home directories are shared from instructor.example.com.

The security consultant suggested that administrative privileges on their servers be limited as much as possible.

On serverX, set up **sudo** to allow the following:

- Users **morris**, **borris**, and **horace** may use sudo to run only the **service** and **chkconfig** commands without supplying their password.

  Note: You will need to add accounts on serverX for **morris**, **borris**, and **horace**. Do not add **morris**, **borris**, and **horace** to any supplemental groups.

When you are ready to check your work, reboot serverX and run **lab-grade-taylorlocke** on serverX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Personal Notes

Unit Summary

Kerberos Configuration

In this section you learned how to:

- Use a Kerberos server to check user passwords

Troubleshooting System Security Services Daemon (SSSD)

In this section you learned how to:

- Correct problems with the **sssd** service and network authentication

UNIT THREE

BASH SCRIPTING AND TOOLS

Introduction

Topics covered in this unit:

- Bash programming constructs
- Shell programming tools
- Password aging defaults

Bash Programming

The default shell in Red Hat Enterprise Linux is**bash**. The **bash** shell can be used to interactively enter commands at a prompt, or it can read commands from a special text file, a *shell script*. In this section, you will learn **bash** programming skills and syntax that can help you automate many tasks that are normally done manually. The syntax you will learn will be useful both in scripts and directly on the command line.

Getting Started with Scripts

How do you create a new shell script?

1. Create a text file containing bash commands. The first line of the file should be:

```
#!/bin/bash
```

2. Make the file executable (using **chmod**)
3. Put it in a directory in the user's **$PATH**
   - **~/bin** - for a user's private programs
   - **/usr/local/bin** - locally developed scripts used by others on the system
   - **/usr/local/sbin** - locally developed scripts used by **root**

Shell Variables

Shell variables are used to assign a value to a name for later use in the script, and are local to the shell command-line or script in which they are declared.

- To define or assign a value:

```
FRUIT=apple
```

- To reference or use a variable:

```
$FRUIT
${FRUIT}
```

Note the **${FRUIT}** syntax clearly marks the exact name of the variable within the curly braces. This can matter if the variable is used in the middle of a text string where part of the string could be confused as part of a variable name: **${FIRST}.${LAST}**, for example.

Note

Environment variables are shell variables which have been *exported* so that they can be seen by programs or subshells started by the command-line or script in which they are declared. These can be used to affect the behavior of programs. We will not discuss environment variables further here, although you can think of them as a special class of shell variables.

Command Substitution

Command substitution executes a given command in a subshell and replaces the commmand substitution in the script with the output of the command.

- Syntax: **`$( shell command )`**
- Example:

```
touch datafile.$(id -un)
TODAY=$(date +%Y-%m-%d)
```

Quoting and Escaping

Quoting and escaping is used to remove the special meaning of special characters or reserved words in a string when the shell parses it. This causes the string to be handled literally, rather than expanding variables or otherwise treating parts of it as having special meaning.

There are three kinds of quoting:

- Weak Quoting

  Placing a string in *double quotes* preserves the literal value of all characters in the string *except* the **`$ ` \`** and **`!`** characters. In other words, variable expansion and command expansion still work inside double quotes:

```
echo "can I have a $FRUIT"
echo "The current time is $(date +%r)."
```

- Strong Quoting

  Placing a string in *single quotes* preserves the literal value of all characters in the string, disabling *all* expansion:

```
echo 'Make $$$ Fast!'
rm 'untitled folder'
```

- Escaping

  A non-quoted **`\`** is the *escape character*. It preserves the literal value of the next character that follows. (For example, **`\$PATH`** would be the exact string **`$PATH`** not the contents of the **`PATH`** variable.)

```
echo Make \$\$\$ Fast\!
ls untitled\ folder
```

Repetition

A **bash** **for**-loop is used for repetition of the same commands over a list of values. In some languages, it would be referred to as a foreach-loop. The variable ***var*** below would take the value of the first item after **in** and the commands between **do** and **done** would be run; then ***var*** takes the value of the next item in the list after **in** and the process is repeated until we run out of items in the list.

- Syntax: **for *var* in *item1 item2 item3*; do *command1; command2*; done**
- Example:

```
for i in *.conf; do
    ls $i
    cp $i $i.bak
done
```

Conditional Branching

Most Linux commands return an *exit status* when they complete. The value of this exit status can be used to make decisions about what code to run. This is called *conditional branching.*

An exit status of **0** is returned when programs complete successfully. This is treated by **bash** as a logical *True* value. A non-zero exit status usually indicates an error occurred, and is treated by **bash** as a logical *False* value.

For example, the following list show what the exit status of **grep** means:

- 0 - pattern found in the files specified
- 1 - pattern not found in the files specified
- >1 - some other error (unable to open files, bad search expression, etc.)

The **test** command can be used to evaluate expressions in a **bash** script. It evaluates the expression specified by its arguments and returns a zero exit status when the expression is true, and a non-zero exit status when the expression is false. **test** has an alternate syntax which uses square brackets to enclose the expression, which can be easier to read.

- Syntax: **test *EXPRESSION*** or **[*EXPRESSION*]**
- Non-zero or zero length string operators: **test -{n|z} *STRING***
- String comparison operators: **==**, **!=**, **<**, **>**, **<=**, **>=**
- Numeric comparison operators: **-eq**, **-ne**, **-lt**, **-gt**, **-le**, **-ge**
- File status operations: **test -{e|f|d|r|w|x|s} *FILE***
- Logical operators: **-o**, **-a**, and **!**

The **if** command checks the exit value of the command or list that follows **if**. If the first command evaluates to true/zero, then it runs the list of commands after **then** up to any **else**. If the first command evaluates to false/non-zero, then **if** runs the list of commands between **else** and **fi** (**if** spelled backwards), which marks the end of the **if** block.

- Syntax: **if *command*; then *command1; command2*; else *command3*; fi**
- Examples:

```
if test "$USER" != 'root' ; then
    echo you are not logged in as root
fi


if [ $(id -u) -lt 9 ]; then
    echo "The number $(id -u) is less than 9!"
fi


if grep "^${user}:" /etc/passwd &> /dev/null ; then
    echo "${user} is a local user on the system."
else
    echo "${user} is not a local user."
fi
```

Use this space for notes

Reading Input and Positional Parameters

There are two easy ways to read user input into a variable in **bash**. The first is to use **read** to prompt the user for input (with the **-p** option) and store it directly into one or more variables:

- Interactive input

```
read -p 'Enter your first and last name:' FIRST LAST
```

The other is to use *positional parameters* to read input from the command-line arguments or options passed to the script. Various special variables store the number of options passed, the individual parameters parsed by **bash**, or the entire raw command-line.

- Number of positional parameters specified: **$#**
- Positional parameters themselves: **$1**, **$2**, **$3**, ...
- All positional parameters: **"$@"**

Running Commands on Remote Machines

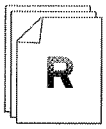
The **ssh** command can be used to open a shell on a remote machine and run commands there. Sometimes, you may want to save the output of the commands run on the remote machine, either on the remote machine or on your local machine. Depending on how you quote the I/O redirection and commands you run with **ssh**, either result is possible.

- Output is redirected to a local file:

```
ssh user@host 'command1; command2' > log.local
```

- Output is redirected to a remote file:

```
ssh user@host 'command1; command2 > log.remote'
```



References

Bash Reference Manual
/usr/share/doc/bash-*/doc/bashref.pdf
(provided by the **bash-doc** package)

bash(1) man page.

Practice Case Study

Writing a Bash Shell Script

Perform the following steps on serverX unless directed otherwise.

Sam has asked you to write a shell script to generate a report called "$am's Report". The report should take a list of directories on the command line and will prompt for the name of the report requester.

If enough arguments are not given, an error message is displayed:

```
usage: ./samsreport.sh directories...
```

For each command line argument the script should print out the current argument, a colon, a space and then one of the following:

- is empty
- is not empty
- is not a directory

For example, when executing the following:

```
$ ./samsreport.sh Desktop Documents .bashrc
```

The script will prompt the user for a name:

```
Who is this report for?
```

Then, if "Jim" was entered, the script should output:

```
$am's Report
 Desktop: is not empty
 Documents: is empty
 .bashrc: is not a directory
___generated for Jim___
```

Sam is very particular and has asked that the output match exactly as above including the dollar sign for the letter S in Sam. For example, the footer should have exactly three underscores, the footer message, and three more underscores without any added whitespace.

To make development easier, it is recommended to write and test your script in this order:

- Print out the usage if there aren't any arguments
- Prompt "Who is this report for? " and read a NAME
- Print the header "$am's Report"
- Print the footer "___generated for NAME___"

- Insert code between the header and footer to print out the body of the report. Ask your instructor for help if you need some hints.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Text Processing Tools

The following commands are often used when writing scripts to automate tasks. Many scripts need to process the output of other commands to change it into a form they can manipulate or make decisions on. This course introduces you to the most useful commands and the options used most frequently in scripts. Take time to research these commands after the course to learn about their more extensive capabilities.

The diff Command

- The **diff** command is used to compare the contents of two files for differences. It can also be used to create patch files used to make changes to similar files across multiple computers in an enterprise environment.

Option	*Description*
-c	display surrounding lines for context
-u	use unified output format (useful for generating patch files)
-r	perform a recursive comparison of files, starting with the named directories

Table 3.1. Useful diff Options

Example:

- Suppose a service is malfunctioning on server1 but the same service works on server20. Thanks to **diff** and the use of simple, text-based configuration files, the working and non-working configurations are easily compared:

```
[user@host ~]$ diff file.conf-server1 file.conf-server20
1c1
< Hostname = server1
---
> Hostname = server20
2c2
< Setting1 = a
---
> Setting1 = A
3a4
> Setting2 = B
5d5
< Setting4 = D
```

diff reports there are four places where the configurations differ:

1. On line 1, the Hostname setting is different in each file. Given the files are from different systems, this makes sense and is probably not the problem.
2. On line 2, the Setting1 variable is set to a on server1 and A on server20. This could cause problems if the service is case-sensitive.
3. Setting2 is set on server20, but not on server1. In other words, if you were to convert the server1 version of the file to the server20 version of the file you would have to add Setting2 after line 3, which would become line 4 of the file.

4. Setting4 is set on server1, but not on server20. In other words, you would need to delete line 5 in the server1 file to make it like the server20 file.

- The following command creates a relatively small patch file that can synchronize changes to many files across multiple servers in an organization:

```
[user@host ~]$ diff -Naur original-dir dir-with-changes > patchfile
```

The patch Command

- **patch** takes a patch file patchfile containing a difference listing produced by **diff** and applies those differences to one or more original files producing patched versions. Normally the patched versions replace the originals, but backups can be made when the **-b** option is specified. Original files will be renamed with a **.orig** filename suffix.

  Example:

  - **patch** can be used to apply a simple patch file to a single file using the following syntax:

```
[root@host etc]# patch issue patchfile
patching file issue
```

  - The following command shows how to use a patch file created with **diff -Naur**. **patch** is executed once the user has changed into the comparable directory similar to the original directory the patch file was made from:

```
[user@host orig-dir]$ patch -b < /tmp/patchfile
patching file hosts
patching file network
```

The grep Command

- **grep** displays the lines in a file that match a pattern. It can also process standard input.

  The pattern may contain regular expression metacharacters and so it is considered good practice to always quote your regular expressions. Basic regular expressions will be covered later in this unit.

Option	*Description*
-i	perform a case-insensitive search
-n	precede returned lines with line numbers
-r	perform a recursive search of files, starting with the named directory
-c	display a count of lines with the matching pattern
-v	return lines that do not contain the pattern
-l	list the names of files that have at least one line containing the pattern

Table 3.2. Useful grep Options

Examples:

- To list lines containing either "cat" from the **pets** file:

```
[user@host ~]$ grep 'cat' pets
```

- To list only lines of output from the **ps** command, which lists running processes, that contain the string "init":

```
[user@host ~]$ ps ax | grep 'init'
```

The cut Command

- **cut** is used to "cut" fields or columns of text from a file and display it to standard output.

Option	*Description*
-d	specify a delimiter for extracting fields (**Tab** is the default)
-f	specify fields to extract from each line
-c	specify columns of text to extract from each line

Table 3.3. Useful cut Options

Examples:

- To display a list of UIDs from **/etc/passwd** (UIDs are stored in the third colon-delimited field):

```
[user@host ~]$ cut -f3 -d: /etc/passwd
```

- A slightly more complicated example shows how **cut** can be used with other tools to extract a single piece of information, in this case the system's IP addresses, from a larger block of output. After examining the command and output below, try running **/sbin/ip addr** by itself and see if you can piece together what each step of the pipeline does:

```
[user@host ~]$ /sbin/ip addr | grep 'inet ' | cut -d ' ' -f6 | cut -d / -f1
192.168.0.254
127.0.0.1
```

The head Command

- The **head** command is used to display just the first few lines of a file.

Examples:

- The default behavior displays 10 lines:

```
[user@host ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

```
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
```

- The **-n** option specifies the number of lines to display:

```
[user@host ~]$ head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
```

The tail Command

- **tail** displays the last few lines of a file. This utility is often used by system administrators to read the most recent entries in log files. The **-f** option causes tail to continue to display the file in "real time", showing additions to the end of the file as they occur.

  Examples:

  - To display the last three lines from **/var/log/cron**:

```
[root@host ~]# tail -n 3 /var/log/cron
root (10/13-18:01:00-658) CMD (run-parts /etc/cron.hourly)
CRON (10/15-13:45:56-422) STARTUP (fork ok)
root (10/15-13:50:00-781) CMD (/sbin/rmmod-as)
```

  - System administrators keep an eye on the system log using the following command:

```
[root@host ~]# tail -f /var/log/messages
```

  tail -f will continue to show updates to the file until **Ctrl+c** is pressed.

The wc Command

- **wc** counts the number of lines, words, bytes and/or characters in a file. On traditional Unix systems every character in a text file took up exactly 1 byte, so the file size could be assumed to be equal to its size in bytes. However with the advent of internationalization and larger character sets like Unicode, some characters can take up to four bytes. Therefore if a document uses any non-ASCII characters, the **-m** option is required to get an accurate character count.

  When more than one file is specified, **wc** reports totals for whatever values it has been told to count. **wc** can also accept data on the standard input. This can be useful for counting the number of lines in a command's output.

Option	*Description*
-l	display the number of lines
-w	display the number of words
-c	display the number of bytes
-m	display the number of characters

Table 3.4. Useful wc Options

Examples:

- Processing multiple files with **wc**:

```
[user@host ~]$ wc .bash*
 1000  2203 23417 .bash_history
    3     3    33 .bash_logout
   16    38   327 .bash_profile
   11    34   289 .bashrc
 1030  2278 24066 total
```

- **ls** can be used with **wc** to display the number of files in a directory. The following command displays the number of files found in **/tmp**:

```
[user@host ~]$ ls /tmp | wc -l
32
```

The sort Command

- **sort** is used to sort text data. This data can be in a file or the output of another command. **sort** is often used with pipes.

Option	*Description*
-n	sort numerically instead of by character
-k	sets the sort field
-t	specify a different field separator (default is whitespace)

Table 3.5. Useful sort Options

Examples:

- The example below displays an alphabetical list of users whose login shell is set to **bash**:

```
[user@host ~]$ grep bash /etc/passwd | cut -d: -f1 | sort
alex
gdm
joshua
root
star
```

The uniq Command

- **uniq** "removes" duplicate adjacent lines from a file. To print only unique line occurrences in a file ("removing" all duplicate lines), input to **uniq** must first be sorted. Since **uniq** can be given fields or columns on which to base its decisions, these are the fields or columns upon which its input must be sorted. Used without options, **uniq** removes duplicate lines in its input, using the entire record as a decision key.

Option	*Description*
-u	display unique lines only
-d	display lines that have duplicates

Option	Description
-c	display each line once with a count of occurrences

Table 3.6. Useful uniq Options

Examples:

- The following example uses **sort** and **uniq** to list the shells used in **/etc/passwd**:

```
[user@host ~]$ cut -d: -f7 /etc/passwd | sort | uniq
/bin/bash
/bin/false
/bin/sync
/dev/null
/sbin/halt
/sbin/nologin
/sbin/shutdown
```

The tr Command

- **tr** tr is used to translate characters; that is, given two ranges of characters, any time a character in the first range is found, it is translated into the equivalent character in the second range. This command is commonly used in shell scripts to convert data into an expected case.

  Example:

  - In this example the user is queried for data. If the user responds in lower case, the **tr** command does nothing. When the user responds in upper case the characters will be changed to lower case:

```
echo -n 'Enter yes or no: '
read answer
answer="$(echo $answer | tr 'A-Z' 'a-z')"
```

The sed Command

- The **sed** command is the stream editor used to perform edits on a stream of textual data. Given a file name to process, **sed** will perform a search and replace on all lines in the file, sending the modified data to standard output; that is, it does not actually modify the existing file. As with **grep**, **sed** is often used in pipelines.

  Since **sed** commands often contain characters which can be interpreted as shell metacharacters, quote **sed** commands as demonstrated in the examples below.

  By default, **sed** operates on all lines in a file. It is possible to provide **sed** with addresses limiting commands to just those lines.

Command	Description
s/*old*/*new*/	perform string substitution, replace occurrences of *old* with *new*
d	delete matched lines

Table 3.7. Useful sed Commands

Examples:

- The **pets** file will be displayed to standard output with the string "cat" being replaced by the string "dog":

```
[user@host ~]$ sed 's/cat/dog/' pets
```

- By default, **sed** makes a maximum of one change per line. To instruct **sed** to make multiple changes per line, append the **g** command, standing for global, to the end of the substitute command:

```
[user@host ~]$ sed 's/cat/dog/g' pets
```

- **sed** searches are case-sensitive by default. To change this, append an **i** to the **sed** substitute command:

```
[user@host ~]$ sed 's/cat/dog/gi' pets
```

- In this example, the entire **pets** file will be sent to standard output, but the replacement of "cat" for "dog" will only be performed on lines 10 through 35, inclusive:

```
[user@host ~]$ sed '10,35s/cat/dog/' pets
```

- The following **sed** command "deletes" lines 1 through 25. All lines are displayed except the first 25 lines:

```
[user@host ~]$ sed '1,25d' pets
```

- The **-e** option allows multiple **sed** commands to be specified at once:

```
[user@host ~]$ sed -e 's/cat/dog/g' -e 's/cow/goat/g' pets
```

Regular Expressions

- The term "regular expression" (often shortened to "regex") refers to a search pattern that uses special characters, called "metacharacters", to represent something other than the literal meaning of those characters. For example, the character ^, when used in a regex-aware program's search pattern means, "the current line starts with", rather than an actual "^" symbol. As in **bash**, you can tell the program that you wish to use a metacharacter literally by pre-pending a \, so \^ would match a literal "^".

Metacharacter	*Meaning*
^	line begins
$	line ends
.	any single character (does not match a line break)
[xyz]	a single character that is x, y, or z
[^xyz]	a single character that is not x, y, or z

Table 3.8. Essential Regular Expression Metacharacters

So a regular expression like ^[Cc]hapter [Oo]ne$ is like saying "The line begins with 'C' or 'c', which is followed by 'hapter ', then 'O' or 'o' and ends with 'ne'". This is very useful for situations where you need to specify not just what you are looking for, but where you are looking for it.

Examples:

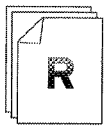
- Suppose you want to print the **/etc/passwd** line for user **brad** using **grep**. The following command will match the line for **brad**, but it will also match **bradley**, **bradford**, and any other line that contains "brad":

```
[user@host ~]$ grep brad /etc/passwd
```

- A regular expression can be used for more precise results:

```
[user@host ~]$ grep '^brad:' /etc/passwd
```

This course covers only the subset of regular expressions most often used by system administrators. The full set of regular expressions is more commonly used by developers writing applications that interpret text data.



References

diff(1), **patch**(1), **grep**(1), **cut**(1), **head**(1), **tail**(1), **wc**(1), **sort**(1), **uniq**(1), **tr**(1), **sed**(1), and **regex**(7) man pages

Practice Quiz

Text Processing Commands

Fill in the appropriate command for solving each problem.

1. To update an Apache directive in **`httpd.conf`** across multiple machines with semi-unique configurations, you could use the _____ utility to create a patchfile

2. To search and replace or delete strings in a file, regardless of context, use _____

3. The _____ command will take either a copied context (_____ **`-c`**) or unified context (_____ **`-u`**) file as input

4. To list all users in **`/etc/passwd`** that use the bash shell, run _____ **`'/bin/bash$' /etc/passwd`**

5. To remove all comments from **`/etc/rc.sysinit`** you could use _____ **`-v '^#'`** or _____ **`'s/^#.*$//'`**

6. To sort all lines in **`/etc/passwd`** by their UID, run _____ **`-n -t: -k3 /etc/passwd`**

7. To list all of the shells from **`/etc/passwd`**, run _____ **`-d: -f7 /etc/passwd`**

8. The tool used to strip out and count duplicate lines of text is _____ **`-c`**

9. To list the number of times a shell is used in **`/etc/passwd`**, run _____ **`-d: -f7 /etc/passwd |`** _____ **`|`** _____ **`-c`**

10. To change all upper case letters to lower case you could use the _____ command

11. To get just the first two lines of **`ls -tl`**, pipe the command to _____ **`-2`**

12. To grab the last line of **du -h**, pipe the command to _____ **-1**

13. To remove the first line of **ls -al**, pipe the command to _____ **-n+2**

14. To remove the footer from **du -h**, pipe the command to _____ **-n-1**

Password Aging

Skim the references below to learn how to make changes to **/etc/login.defs** and **/etc/default/useradd** to set password aging defaults when new accounts are created. Complete the tables below since we will review by playing a game after the reading exercise.

Key Defaults Defined in /etc/login.defs

Fill in the following values:

Variable	*Definition*	*Default Value*
PASS_MIN_DAYS	Minimum number of days that must pass between password changes	0 - No restriction
PASS_MAX_DAYS		
PASS_WARN_AGE		

Table 3.9. **/etc/login.defs** Definitions

Key Defaults Defined in /etc/default/useradd

Fill in the following values:

Variable	*Definition*	*Default Value*
INACTIVE		
EXPIRE		

Table 3.10. **/etc/default/useradd** Definitions

The chage Command

- The **chage** command adjusts password and account expiration values for existing accounts.
- Examples:
  - Display the password and account information for a user called **rochelle**:

```
[root@serverX ~]# chage -l rochelle
Last password change      : Dec 17, 2010
```

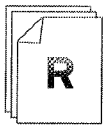
```
Password expires     : never
Password inactive     : never
Account expires      : never
Minimum number of days between password change  : 0
Maximum number of days between password change  : 99999
Number of days of warning before password expires : 7
```

- Set the minimum password age to 10 days, the maximum to 90 days, and the password warning period to 14 days for a user called **alex**:

```
[root@serverX ~]# chage -m 10 -M 90 -W 14 alex
```

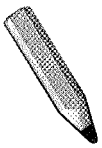
- Set the password inactivity period to 30 days and the account expiration date to April 1, 2012 for a user called **sue**:

```
[root@serverX ~]# chage -I 30 -E 2012-04-01 sue
[root@serverX ~]# grep sue /etc/shadow
sue:!!:14994:0:99999:7:30:15431:
```



References

login.defs(5), **useradd**(8), **chage**(1), and **shadow**(5) man pages



Test

Criterion Test

Case Study

Scripting User Management Tasks

Perform the following steps on serverX unless directed otherwise.

Write a bash script that creates user accounts using data from a text file. Each line of the file contains four colon-separated fields that specify the information for a single user. The four fields represent the username, supplemental group membership, password, and maximum password age in days. Lines that start with "#" are ignored. Download a sample file containing the user list from **`ftp://instructor.example.com/materials/user_list.txt`**

Create any necessary supplemental groups before testing your script.

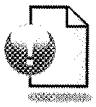
If your script does not work properly, then use text processing tools to extract the usernames from **`user_list.txt`** into a different file called **`usernames.txt`**. The following bash code will remove the user accounts so that you can test your program again:

```
#!/bin/bash
for u in $(cat usernames.txt)
do
    echo "Deleting $u"
    userdel -r $u
done
```

When you have completed the lab, copy your scripts to desktopX for future reference.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Personal Notes



Unit Summary

Bash Programming

Since you've completed this unit, you are able to:

- Program in Bash

Text Processing Tools

Since you've completed this unit, you are able to:

- Use shell tools common in bash scripts

Password Aging

Since you've completed this unit, you are able to:

- Configure password aging defaults

UNIT FOUR

FILE SECURITY WITH GNUPG

Introduction

Topics covered in this unit:

- File encryption with GPG

Encrypting Files with GnuPG

Encryption is a technique which protects data stored on the system or transmitted over the network from compromise. It can be used to *verify* that data has come from a particular person without modification, and also to *maintain confidentiality* of the data.

You may already be familiar with some uses of encryption in Linux, including Kerberos (which uses symmetric encryption, based on the password as a shared secret key, to authenticate users), TLS/SSL to protect network services, or LUKS to protect the contents of file systems on block devices. In this unit, you will look at ways to validate and/or encrypt individual files using a tool called **GNU Privacy Guard**, also known as **GnuPG** or **GPG**.

Definitions

- *Public-key encryption*: Encryption using a matched pair of encryption keys, a public key and a private key. Also called *asymmetric encryption*.
- *Public key*: The encryption key given to anyone who wants to communicate confidentially with the owner of the private key (the public half of the matched key pair). What is encrypted by the public key can only be decrypted by the private key.
- *Private key*: The encryption key kept secret by the owner (the private half of the matched key pair). What is encrypted by the private key can be decrypted by anyone with the matching public key. Messages encrypted with the private key are not secret, but holders of the matching public key can verify that they came from the holder of the private key. This is effectively a *digital signature*.
- *GNU Privacy Guard*: "GnuPG" or "GPG" is an open source implementation of the OpenPGP public key encryption system. You will learn how to use GnuPG 2.0 in this class.

gpg Options

Take some additional notes as you discuss these options:

1. Generate a key pair: **`gpg --gen-key`**

2. List public keys: **`gpg --list-keys`**

3. Export a public key: **`gpg --export --armor key-id -o file.key`**

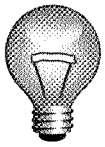
4. Import a public key: **`gpg --import file.key`**

5. Encrypt a file: **`gpg --encrypt --armor -r key-id file`**

6. Decrypt a file: **gpg --decrypt *file***

Note

For *key-id* in the commands above, the most common ways to specify a key are to use either the user's e-mail address, the **pub** key's full GPG fingerprint with no spaces (from **gpg --fingerprint**), or the last eight hexidecimal digits of the **pub** key's fingerprint (the actual "key ID"). See "HOW TO SPECIFY A USER ID" in the **gpg**(1) man page for all the variants possible.

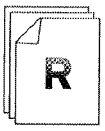


Important

Importing a GnuPG public key will allow you to use it, but any validation checks will indicate it is not "valid". If you have verified through a secure channel that a key with a particular fingerprint does belong to its claimed owner, you can sign it with your private key to indicate that you trust that it is valid. This is done with the command:

```
gpg --sign-key key-id
```

GnuPG even allows you to indicate that you trust other people who have given you keys to validate keys for you. This is where GnuPG's "web of trust" comes in. Signing keys and building a "web of trust" is beyond the scope of this course.



References

Red Hat Enterprise Linux Security Guide

- Section 3.9: Using GNU Privacy Guard (GnuPG)

gpg(1) man page

/usr/share/doc/gnupg-*/FAQ)

The GnuPG website at `http://www.gnupg.org/`

Practice Quiz

Secure Files with GnuPG

1. _____ encryption uses a pair of keys to encrypt/decrypt information. These keys are known as a _____ key (to be shared) and a _____ key (protected).

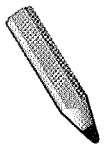
2. The **gpg** _____ command will create a key pair.

3. When creating a key pair you will specify a key type, the length of key in bits, an expiration, your name/e-mail information and a _____ used to protect the private key.

4. The **gpg** _____ command displays the public keys that have been imported into your keyring.

5. **`gpg --encrypt --armor myfile`** will create an encrypted file named _____, after prompting for the intended _____.

6. Using the sender's private key to encrypt a message is called a _____.



Test

Criterion Test

Performance Checklist

File Security with GnuPG

Partner with another learner to create an environment where you can encrypt files, share them, then decrypt them.

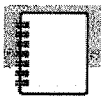
Perform these steps as **student** on your serverX machine.

- ☐ Create your own public/private key pair.

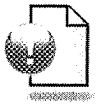
Note

You must have a graphical session available to successfully generate a GPG key. **gpg** now uses a graphical application to enter and validate the key.

- ☐ Export your public key to share with your partner.
- ☐ Copy your exported public key to your partner's serverX machine (perhaps use **scp**).
- ☐ Import your partner's public key.
- ☐ Create a text file with a message for your partner to read.
- ☐ Encrypt the file with your partner's public key.
- ☐ Exchange encrypted files with your partner (again, use **scp**).
- ☐ Decrypt the file encrypted by your partner and verify you can read the message they sent.



Personal Notes



Unit Summary

Encrypting Files with GnuPG

In this section you learned how to:

- Generate and use GPG keys to encrypt and decrypt files
- Store or transmit confidential information securely
- Protect sensitive information from others than the intended recipient

UNIT FIVE

PACKAGE MANAGEMENT

Introduction

Topics covered in this unit:

- Yum plugins
- Package security considerations
- RPM package specifications
- RPM building
- Publish RPM packages

Yum Plugins

The **yum** package manager can have its capabilities extended through the use of *plugins*. A number of plugins are provided as *yum-plugin-** packages in Red Hat Enterprise Linux 6. These can be used to do such things as download only security errata, prioritize packages from different **yum** repositories, and download but not install packages, for example.

In this section we will investigate just two of these plugins, **yum-plugin-verify** and **yum-versionlock**.

Search & Learn: Yum Plugins

Consult the reference documents below to answer the following questions:

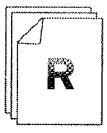
1. What is the purpose of the **yum-plugin-verify** package? Install it on serverX.

2. What new **yum** sub-commands does **yum-plugin-verify** introduce and what do the commands do?

3. What is the purpose of the **yum-plugin-versionlock** package? Install it on serverX.

4. Which configuration file is used to lock down specific packages?

5. What is the format of that configuration file?



References

Red Hat Enterprise Linux Deployment Guide
- Section 1.4: Yum Plugins

yum-verify(1) and **yum-versionlock**(1) man pages

Practice Resequencing Exercise

Yum Plugins

For each of the four definitions below, write down the number of the matching file or command on the line in front of the definition.

___ This command lists all file changes from the original state of a package, including some that **rpm** would ignore.

___ This file contains packages that **yum** will refrain from updating (one package name per line).

___ This command is 100% compatible with **rpm -V** functionality.

___ This command verifies a package and ignores changes to configuration files.

1. **/etc/yum/pluginconf.d/versionlock.conf**
2. **yum verify *package***
3. **yum verify-all *package***
4. **/etc/yum/pluginconf.d/versionlock.list**
5. **yum verify-rpm *package***

RPM Package Design

Managing software in the form of RPM packages is much simpler than working with software which has simply been extracted into a file system from an archive. It lets you track which files were installed by the software package, which ones need to be removed if it is uninstalled, and check to ensure supporting packages are present when it is installed.

Therefore, it is useful to know how to create RPM packages for your own software. For the remainder of this unit, we will look at how to create a basic RPM package and point you to resources which will help you learn how to create more complex packages as your skills grow.

Design/Structure of an RPM

Each RPM package is made up of three basic components:

- *metadata* - Data about the package: the package name, version, release, builder, date, dependencies, etc.
- *files* - archive of files provided by the package (including file attributes)
- *scripts* - these execute when the package is installed, updated, and/or removed

When building an RPM package, the metadata about the package needs to be specified, the files in the archive need to be provided, and the scripts that should be run when the package is installed or uninstalled need to be embedded.

Note

Internally, files are stored as a **cpio** archive inside the package file. The **rpm2cpio** command can be used to extract them to the current working directory without installing the package: **rpm2cpio *package-1.2.3-4.el6.x86_64.rpm* | cpio -id**

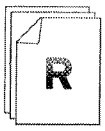
The following **rpm** queries are useful for investigating the structure of an RPM package:

- **rpm -qd** - list documentation files (**%doc**)
- **rpm -qc** - list configuration files (**%config**)
- **rpm -q --scripts** - list **%pre**, **%post**, **%preun**, and **%postun** scripts

Security Implications

When packages are installed, they are executed as **root**:

- internal scripts can harm the system
- files can be installed that have a negative impact on the system
- files can be created with **setuid** and **setgid** permissions



References

Red Hat Enterprise Linux Deployment Guide

- Section 3.2.6: Querying RPM

rpm(8), **rpm2cpio**(8), and **cpio**(1) man pages

RPM Package Specifications

To construct an RPM package, you will need a build specification file or *spec file*. A spec file is simply a text file that contains information on how to build the installable RPM package. You can think of it as being roughly divided into five parts:

- The *introduction* or *preamble*, listing metadata about the package (name, version, license, etc.)
- The *build* instructions, which specify how to compile and prepare the software
- The *scriptlets*, which specify commands to run on install, uninstall, or upgrade
- The *manifest*, a list of files to package and their permissions on package installation
- The *changelog*, which tracks changes made to this RPM package

Important Preamble Directives:

- **Name** - The name of the package, usually chosen by the developers. For detailed guidance, look at the Fedora Naming Guidelines, at *http://fedoraproject.org/wiki/Packaging:NamingGuidelines*
- **Version** - The version of the package (usually numeric), usually chosen by the developers.
- **Release** - The release of the package, chosen by the packager. This should increase each time you release a new package for distribution if you still use the same Version of the software.
- **Group** - The group to which the package belongs. See **/usr/share/doc/rpm-*/GROUPS** for the default set of groups, or use one of your own. This field is semi-obsolete, and is not related to **yum** package groups.
- **License** - The "Short License identifier" of the license used for the software. Detailed guidance on how to set this in a standard way can be found at *http://fedoraproject.org/wiki/Packaging/LicensingGuidelines*
- **Summary** - A short one-line description of the software. (Keep to about 50 characters or less.)
- **Source** - The file to be used as the source code. If there are more than one file used as source, add a number. E.g., Source0, Source1, Source2, etc.
- **BuildArch** - The architecture to use when building the package. Defaults to the system architecture. A common argument is **noarch**, which means that the package is architecture independent (often these packages consist of scripts or data files).
- **Requires** - A list of explicit requirements this package depends on. This could be a list of files or other packages. **rpmbuild** can generally autodetect most library dependencies, but there are some cases where you may need to list an explicit dependency. See *http://fedoraproject.org/wiki/Packaging/Guidelines#Requires* for additional guidance on **Requires**.
- **BuildRequires** - A list of requirements that are needed to *build* this package. This is a list with similar syntax to that of **Requires**, for example **BuildRequires: /usr/bin/gcc, gimp-libs >= 2.6.11**. See the link in **Requires** above for information on how to tell if you need missing **BuildRequires**.

Required Spec File Sections:

- **%description** section - A long description of the software. No line should be more than 80 characters long, but you may have multiple lines.
- **%prep** section - extract sources to prepare for build
- **%build** section - shell code that compiles the project
- **%install** section - code that moves files to the $RPM_BUILD_ROOT chrooted environment
- **%clean** section - clean up the build tree
- **%files** section - list of files to include in the package
- **%changelog** section - list of packager changes and updates

rpmbuild Steps

When **rpmbuild** runs, the build process will work through the following sections in order:

1. %prep
2. %build
3. %install
4. Package the completed RPM
5. %clean

Creating a New Spec File

On Red Hat Enterprise Linux 6, **vim** has a macro that helps to create a specification file. Simply pass a file name that ends in **.spec**:

```
[student@serverX]$ vim foo.spec
```

vim will use the spec template to provide some common entries for RPM building.

Note

When an RPM package is built, a *source RPM* (SRPM) package is also created, with an architecture of **src**. Another way to get a spec file is to install a source package by running **rpm -ivh *package-1.2.3-4.src.rpm*** as a *non-root user*. The spec file for the package will be in **~/rpmbuild/SPECS**.

Example Spec File

An annotated example of a spec file follows.

```
%define debug_package %{nil}❶
%define product_family Red Hat Enterprise Linux
%define release_name Santiago
%define base_release_version 6
%define full_release_version 6.0
%define beta Beta

Name:           redhat-release ❷
Version:        %{base_release_version} ❸
Release:        6.0.0.24%{?dist} ❹
Summary:        %{product_family} release file ❺
Group:          System Environment/Base
License:        GPLv2
Obsoletes:      rawhide-release redhat-release-as redhat-release-es redhat-release-ws ❻
Source0:        redhat-release-6-4.tar.gz ❼

%description ❽
%{product_family} release files

%prep ❾
%setup -q

%build ❿
echo OK

%install ⓫
rm -rf $RPM_BUILD_ROOT

# create /etc
mkdir -p $RPM_BUILD_ROOT/etc

# create /etc/system-release and /etc/redhat/release
echo "%{product_family} release %{full_release_version}%{?beta: %{beta}}
 (%{release_name})" > $RPM_BUILD_ROOT/etc/redhat-release
ln -s redhat-release $RPM_BUILD_ROOT/etc/system-release

# write cpe to /etc/system/release-cpe
echo "cpe:/o:redhat:enterprise_linux:%{version}:%{?beta:%{beta}}%{!?beta:GA}" >
 $RPM_BUILD_ROOT/etc/system-release-cpe

# create /etc/issue and /etc/issue.net
cp $RPM_BUILD_ROOT/etc/redhat-release $RPM_BUILD_ROOT/etc/issue
echo "Kernel \r on an \m" >> $RPM_BUILD_ROOT/etc/issue
cp $RPM_BUILD_ROOT/etc/issue $RPM_BUILD_ROOT/etc/issue.net
echo >> $RPM_BUILD_ROOT/etc/issue

# copy yum repos to /etc/yum.repos.d
mkdir -p $RPM_BUILD_ROOT/etc/yum.repos.d
for file in *.repo; do
    install -m 644 $file $RPM_BUILD_ROOT/etc/yum.repos.d
done

# copy GPG keys
mkdir -p -m 755 $RPM_BUILD_ROOT/etc/pki/rpm-gpg
for file in RPM-GPG-KEY* ; do
    install -m 644 $file $RPM_BUILD_ROOT/etc/pki/rpm-gpg
done
```

```
# set up the dist tag macros
install -d -m 755 $RPM_BUILD_ROOT/etc/rpm
cat >> $RPM_BUILD_ROOT/etc/rpm/macros.dist << EOF
# dist macros.

%%rhel %{base_release_version}
%%dist .el%{base_release_version}
%%el%{base_release_version} 1
EOF

%clean ⓬
rm -rf $RPM_BUILD_ROOT

%files ⓭
%defattr(-,root,root)
%doc EULA GPL autorun-template
%attr(0644,root,root) /etc/redhat-release
/etc/system-release
%config %attr(0644,root,root) /etc/system-release-cpe
%config(noreplace) %attr(0644,root,root) /etc/issue
%config(noreplace) %attr(0644,root,root) /etc/issue.net
%config %attr(0644,root,root) /etc/yum.repos.d/*
%dir /etc/pki/rpm-gpg
/etc/pki/rpm-gpg/*
/etc/rpm/macros.dist

%changelog ⓮
* Mon Mar 29 2010 Dennis Gregorovic <dgregor@redhat.com> - 6-6.0.0.24
- Add beta debuginfo repos
- Resolves: rhbz#572308
```

❶ Macros (like variables) that can be used in the spec file

❷ The name of the package

❸ The version of the package. Notice it uses the **%{base_release_version}** macro defined above.

❹ The release of the package

❺ A short summary

❻ A list of package names that this package makes obsolete. If you had one of these packages installed on your machine, an update of this package would remove that package.

❼ A source file

❽ A long description

❾ The **%prep** section. Unfortunately, the RPM spec file uses % for sections as well as macros. **%prep** is a section, **%setup** is a macro.

❿ The **%build** section

⓫ The **%install** section. **$RPM_BUILD_ROOT** is a variable that expands to the "build root". Files are copied from the build directory to **$RPM_BUILD_ROOT**, as if **$RPM_BUILD_ROOT** was **/** on the file system the software will be installed in. Then the contents of **$RPM_BUILD_ROOT** listed in **%files** will be packaged into the final RPM file. You must create all necessary directories in **$RPM_BUILD_ROOT** before copying files to that location. Source files can be referenced using a relative path from the top-level **%{name}-%{version}** source directory. For example, if you wanted to have a file placed in **/root/bin/** (found in the **%{name}-%{version}/bin** directory) , you would need to do something like the following:

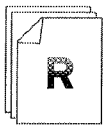
```
mkdir -p $RPM_BUILD_ROOT/root/bin
cp bin/my-script $RPM_BUILD_ROOT/root/bin
```

⓬ The **%clean** section. Normally clean only has the **rm** command above.

⓭ The list of files to be included in this package. Note that **%defattr** sets the default permissions the files will have, **%attr** can override that on a file-by-file basis. **%config** and **%doc** mark configuration files and documentation respectively. **%dir** marks a directory owned by the package. See *http://fedoraproject.org/wiki/Packaging:Guidelines#File_and_Directory_Ownership* and *http://fedoraproject.org/wiki/Packaging:Guidelines#Configuration_files* for more information.

⓮ The **%changelog** section is for the packager to list items that changed in this release. Newest entries to the changelog go at the start of the section. Each entry has the format seen in the example, and entries are separated by a blank line.

The example above does not use any scriptlets. For more information on scriptlets, see the draft Fedora RPM Guide referenced below.



References

Fedora RPM Guide -

http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/

Fedora Packaging Guidelines -

http://fedoraproject.org/wiki/Packaging:Guidelines

Practice Quiz

RPM Spec File

1. The package _____ is usually derived from the open source project while the package _____ is the packager's version.

2. The _____ directive categorizes the type of package being built.

3. The name of the tarball containing the files used to build the package is specified with the _____ directive.

4. The _____ directive specifies the target architecture the package is being built for. _____ will be its value when the package can be installed on any architecture.

5. The _____ directive specifies the 1-line description of a package while the _____ section provides a more thorough explanation of what that package is for.

6. The _____ section contains the code used to place files in the _____ chroot directory structure.

7. The _____ section defines which files and directories to package into the RPM.

8. The _____, _____, and _____ sections contain shell code used to assemble a package and clean up after it has been built.

Building and Signing an RPM Package

The five steps for building an RPM package:

1. *Tarball*

   Get the tar file containing the source. By default **rpmbuild** assumes the top-level directory of the archive is named **%{name}-%{version}**. Place this file in the **~/rpmbuild/SOURCES/** directory.

2. *Spec file*

   Create a spec file and populate the required fields. Place this file in the **~/rpmbuild/SPECS/** directory.

3. *rpmbuild*

   Use the **rpmbuild** command to build the package(s). For example,

   ```
   rpmbuild -ba demo.spec
   ```

4. *Sign*

   Use a GPG key to sign the RPM package. You can use **rpmbuild -ba --sign *demo*.spec** to build and sign the package in one step. If the package is already built, use **rpm --resign *demo-1.0-1.x86_64*.rpm** to add (or change) a GPG signature.

5. *Test*

   Test the package by installing it on a development system to ensure the correct payload, scripts, etc.

Preparing a GPG Signing Key

RPM packages are normally digitally signed so that users can verify that a package actually came from the preparer it claims to belong to. This helps to block forged packages from being installed if a yum repository is compromised in some way. The next few steps detail how to create your own signing key. Once you have a signing key you can use it for signing many packages.

If you do not have a GPG key yet, run the **gpg --gen-key** command to generate a new one.

Note

You must have a graphical session open to run **gpg --gen-key**. It uses a graphical box to accept your input for the passphrase.

```
[student@serverX ~]$ gpg --gen-key
gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc. This is free
 software: you are free to change and redistribute it. There is NO WARRANTY, to the extent
```

```
 permitted by law. Please select what kind of key you want: (1) RSA and RSA (default) (2)
 DSA and Elgamal (3) DSA (sign only) (4) RSA (sign only) Your selection?   Enter
RSA keys may be between 1024 and 4096 bits long. What keysize do you want? (2048)  Enter
Requested keysize is 2048 bits Please specify how long the key should be valid. 0 = key
 does not expire <n> = key expires in n days <n>w = key expires in n weeks <n>m = key
 expires in n months <n>y = key expires in n years Key is valid for? (0)  Enter
Key does not expire at all Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key. Real name: My Name
Email address: student@serverX.example.com
Comment:  Enter
You selected this USER-ID: "My Name <student@serverX.example.com>" Change (N)ame,
 (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key. Enter passphrase Passphrase: testing123
Please re-enter this passpassphrase. Passphrase: testing123
We need to generate a lot of random bytes. It is a good idea to perform some other action
 (type on the keyboard, move the mouse, utilize the disks) during the prime generation;
 this gives the random number generator a better chance to gain enough entropy.

gpg: /home/student/.gnupg/trustdb.gpg: trustdb created
gpg: key 54AF5285 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
pub 2048R/54AF5285 2010-12-09
   Key fingerprint = 315F E90B 1745 2288 EBAE 4E7B 4BC6 4568 54AF 5285
uid     My Name <student@serverX.example.com>
sub     2048R/D08B2951 2010-12-09
```

Find the public key ID from the output of **gpg --gen-key**, or run **gpg --fingerprint**

```
[student@serverX ~]$ gpg --fingerprint
/home/student/.gnupg/pubring.gpg
--------------------------------
pub 2048R/54AF5285 2010-12-09
   Key fingerprint = 315F E90B 1745 2288 EBAE 4E7B 4BC6 4568 54AF 5285
uid     My Name <student@serverX.example.com>
sub     2048R/D08B2951 2010-12-09
```

The public key ID is the string of eight hexadecimal characters after **pub 2048R/** (54AF5285 in the example above).

Export the public key (make sure you use your own key ID):

```
[student@serverX ~]$ gpg -a -o ~/RPM-GPG-KEY-student --export 54AF5285
```

Add the following to the **~/.rpmmacros** file (replacing the eight-character key ID with the key ID for your system) so RPM will sign packages with the key you created above.

```
[student@serverX ~]$ echo '%_gpg_name 54AF5285' > ~/.rpmmacros
```

Example RPM Package Build

The following shows an example of building an RPM package. The name of the package is **test**, version is **1.0** and release is **1**. It will provide a single file, **/usr/local/bin/myscript**, which simply runs the **date** command.

Create the directory, file and tarball:

```
[student@serverX ~]$ mkdir test-1.0
[student@serverX ~]$ cat << EOF > test-1.0/myscript
#!/bin/bash
date
EOF
[student@serverX ~]$ tar czvf test-1.0.tar.gz test-1.0
```

Create a spec file using **vim** in your home directory:

```
[student@serverX ~]$ vim test.spec
```

Note

In Red Hat Enterprise Linux 6, **vim** will automatically create a template spec file when you open a new file with a name that ends in **.spec**.

Fill in the fields as follows.

```
Name:           test
Version:        1.0
Release:        1%{?dist}
Summary:        A test package

Group:          Testing
License:        GPL
URL:            http://www.example.com/testing
Source0:        %{name}-%{version}.tar.gz❶
BuildRoot:      %(mktemp -ud %{_tmppath}/%{name}-%{version}-%{release}-XXXXXX)

BuildRequires:  /bin/rm, /bin/mkdir, /bin/cp❷
Requires:       /bin/bash, /bin/date

%description
A testing package meant to deploy a single file.

%prep
%setup -q

%build
#configure❸
#make %{?_smp_mflags}


%install
```

```
rm -rf $RPM_BUILD_ROOT
#make install DESTDIR=$RPM_BUILD_ROOT
mkdir -p $RPM_BUILD_ROOT/usr/local/bin
cp myscript $RPM_BUILD_ROOT/usr/local/bin

%clean
rm -rf $RPM_BUILD_ROOT


%files
%defattr(-,root,root,-)
#%doc
%attr(0755,root,root)/usr/local/bin/myscript❹

%changelog
* Thu Dec 09 2010 Forrest <forrest@redhat.com> 1.0-1
- Initial RPM
- Added /usr/local/bin/myscript
```

❶ The **%{name}** and **%{version}** macros are defined from the **Name:** and **Version:** lines above. Alternately, you could have used **test-1.0.tar.gz**.

❷ **rm**, **mkdir** and **cp** all come from the **coreutils** package, so you could have specified that packge instead of the commands. These are the commands that are used in the **%install** section.

❸ There are some macros that run even if they are commented, and **%configure** is one of them. If you comment **%configure** like: **#%configure**, it will complain about not finding **./configure**. Remove the **%configure** line entirely or remove the **%** from **configure**.

❹ The **%attr** was added to force the permission to 0755. Notice that the **%defattr** has a **-** in the permissions place. This means that the files will get the same permissions that they have inside the tarball. An alternate means to produce the same result would be to run **chmod 755 test-1.0/myscript** and rebuild the tarball.

Install the **rpm-build** package as root:

```
[root@serverX ~]# yum install -y rpm-build
```

Run **rpmbuild** as student. The first time you run it, you will get an error. You will fix the error shortly. Running the **rpmbuild** command will create the directory structure needed to build the RPM package.

```
[student@serverX ~]$ rpmbuild test.spec
error: File /home/student/rpmbuild/SOURCES/test-1.0.tar.gz: No such file or directory
```

Warning

You should always run **rpmbuild** to build packages as a *non-root* user. *Do not build packages as root.* The reason for this is that mistakes in the spec file, especially in the **%install** and **%clean** sections, are more likely to damage your build machine's installation if run as root.

Copy the files to the correct location:

```
[student@serverX ~]$ cp test-1.0.tar.gz rpmbuild/SOURCES/
[student@serverX ~]$ cp test.spec rpmbuild/SPECS/
[student@serverX ~]$ cd rpmbuild/SPECS/
```

Build and sign the package:

```
[student@serverX ~]$ rpmbuild --sign -ba test.spec
Enter pass phrase: testing123
Pass phrase is good.
...
```

Look for errors in the output of **rpmbuild** and fix any issues you find. If there are no errors, you should find:

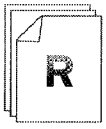
```
...
Wrote: /home/student/rpmbuild/SRPMS/test-1.0-1.el6.src.rpm
Wrote: /home/student/rpmbuild/RPMS/x86_64/test-1.0-1.el6.x86_64.rpm
...
```

Test the package by installing the key, installing the package and running the command:

```
[root@serverX ~]# rpm --import /home/student/RPM-GPG-KEY-student
[root@serverX ~]# cd /home/student/rpmbuild/RPMS/x86_64
[root@serverX ~]# yum localinstall test-1.0-1.el6.x86_64.rpm
[student@serverX ~]$ /usr/local/bin/myscript
Thu Dec 09 10:21:53 EST 2010
```

Note

When reviewing a completed package for release, you may find the formal Fedora Package Review Guidelines (in the References below) to be useful.



References

Fedora RPM Guide -
http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/

Fedora Packaging Guidelines -
http://fedoraproject.org/wiki/Packaging:Guidelines

Fedora Package Review Guidelines -
http://fedoraproject.org/wiki/Packaging:ReviewGuidelines

rpmbuild(8) man page

Publish RPM Packages

Once you have an RPM package, you need to have a way to distribute it to your Red Hat Enterprise Linux systems. Ideally, you have a Red Hat Network Satellite server and can use that to deploy and manage your custom packages. But if you do not, one easy way to make packages available to clients is to set up your own yum repository.

Create a yum Repository

```
[root@serverX ~]# yum install -y createrepo
[root@serverX ~]# mkdir -p /var/www/html/repo/Packages
[root@serverX ~]# cp test-1.0-1.el6.x86_64.rpm /var/www/html/repo/Packages
[root@serverX ~]# createrepo -v /var/www/html/repo/
[root@serverX ~]# cp /home/student/RPM-GPG-KEY-student /var/www/html/repo/
```

Sample yum Configuration File

```
[example]
name=example
description=Example Yum Repository
baseurl=http://serverX.example.com/repo
enabled=1
gpgcheck=1
gpgkey=http://serverX.example.com/repo/RPM-GPG-KEY-student
```

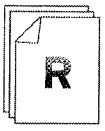
The **gpgkey** line could also look like the following, referencing an FTP server or a local file:

```
gpgkey=ftp://serverX/pub/RPM-GPG-KEY-student
```

-OR-

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-student
```

Whichever you choose, the GPG key file referenced is the public key matching the private GPG key that was used to sign the packages in the repository.



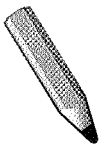
References

createrepo(8) man page

Practice Quiz

Create a Yum Repository

1. Install the _____ package if necessary.
2. Create a directory that can be _____.
3. Create a subdirectory called _____.
4. Copy _____ to be published into _____.
5. Execute _____ on the _____ directory.



Test

Criterion Test

Performance Checklist

Part 1: Query with Yum Plugins

- ☐ Identify and install, if necessary, the **yum** plugin that will allow you to use **yum** to view a package changelog for a package that is not installed.
- ☐ Use the plugin to view the changelog for the **zsh** package.
- ☐ Identify and install the **yum** plugin, if necessary, that will allow you to use **yum** to verify installed packages.
- ☐ Use the plugin to verify the **initscripts** package. Show all changes.

Performance Checklist

Part 2: Create an RPM

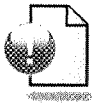
- ☐ Download the the file *ftp://instructor.example.com/pub/materials/hello.sh*.
- ☐ Create a simple RPM that installs **hello.sh** in **/root/bin**. Make sure that **hello.sh** is installed with a mode of 755.
- ☐ Create a GPG key and sign the package with the key. Export the public GPG key.

Note

You must have a graphical session available to successfully generate a GPG key. **gpg** now uses a graphical application to enter and validate the key.

- ☐ Deploy a web server and create a **yum** repository in **/var/www/html/Packages/**. Create a repository file that references *http://serverX/Packages*. Serve the GPG key from the web server and include the key in the repository file.
- ☐ Install your RPM package using the **yum** repository above and run **/root/bin/hello.sh**.

Personal Notes



Unit Summary

Yum Plugins

In this section you learned how to:

- Assess the state of installed packages and prevent **yum** from updating sensitive packages using **yum** plugins

RPM Package Design

In this section you learned how to:

- Use **rpm** to explore the structure of a package file
- Explain the security risks posed when installing packages from untrusted sources

RPM Package Specifications

In this section you learned how to:

- Write a "spec file" to build your own RPM software package

Building and Signing an RPM Package

In this section you learned how to:

- Use **rpmbuild** to build and sign a new RPM package file

Publish RPM Packages

In this section you learned how to:

- Create your own **yum** repository to deploy a small number of package files

UNIT SIX

NETWORK MONITORING

Introduction

Topics covered in this unit:

- Detecting open ports
- Packet capture and analysis

Detecting Open Ports

In this section, we will look at ways to determine which network ports are currently being listened on by network services for incoming connections. Open network services which have security vulnerabilities or misconfigurations are one of the more common ways computer systems are compromised. As each additional network service which is running adds another route for attackers, it is a good practice to only run network services which are actually needed and used. In order to determine this, it is important to be able to tell which network services are running on a machine.

Detect Local Services

The **netstat** utility displays information about open ports on the local system.

```
[root@serverX ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
 PID/Program name
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
 1286/rpcbind
tcp        0      0 0.0.0.0:21                  0.0.0.0:*                   LISTEN
 24876/vsftpd
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
 1607/sshd
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
 1446/cupsd
```

Each line of output displays a process (PID and Program name) that is **LISTEN**ing or part of an **ESTABLISHED** connection. A Local Address of **0.0.0.0** or **::** means that the process is listening on all interfaces.

Note

Be careful reading the output from **netstat**, as it tends to wrap to multiple lines (as shown above).

Fill in the below **netstat** options for each description:

- _____ - tcp protocol or open TCP ports
- _____ - udp protocol or open UDP ports
- _____ - listening ports only, not active, established connections
- _____ - display hosts and ports by number, not name
- _____ - display which local process controls a port

Detect Remote Services

The **nmap** utility is a *portscanner* which probes remote systems and displays information about which ports are open. This can help you determine if unexpected services are running or ports are available on systems you manage.

Warning

Portscanners are also used by attackers to gather intelligence about machines they would like to compromise. Therefore, many organizations have strict policies that restrict usage of portscanners to authorized security personnel. Likewise, using a portscanner to gather information about machines outside your control can be considered hostile activity.

Ensure that you are permitted to run portscanners under your organization's security policy, and do not run portscanners on a computer outside your control without your organization's and the owner's permission.

```
[root@desktopX ~]# nmap -A -sT server1

Starting Nmap 5.21 ( http://nmap.org ) at 2011-01-17 14:28 EST
Warning: Traceroute does not support idle or connect scan, disabling...
Nmap scan report for server1 (192.168.0.101)
Host is up (0.00055s latency).
rDNS record for 192.168.0.101: server1.example.com
Not shown: 996 filtered ports
PORT    STATE  SERVICE VERSION
21/tcp  open   ftp     vsftpd 2.2.2
22/tcp  open   ssh     OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey: 1024 f5:bc:70:0a:d9:aa:29:fc:80:a4:46:1a:cc:96:c8:1c (DSA)
|_2048 2e:fd:19:a2:e7:dd:67:b3:cc:46:b7:33:0c:b6:fa:06 (RSA)
80/tcp  open   http    Apache httpd 2.2.15 ((Red Hat))
|_html-title: Test Page for the Apache HTTP Server on Red Hat Enterprise Linux
443/tcp closed https
MAC Address: 52:54:00:00:00:01 (QEMU Virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.22 (Fedora Core 6)
Network Distance: 1 hop
Service Info: OS: Unix

OS and Service detection performed. Please report any incorrect results at http://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.18 seconds
```

Fill in the below **nmap** options for each description:

- _____ - performs a ping scan to identify hosts that respond to an ICMP ping request
- _____ - performs a TCP connect scan
- _____ - performs a UDP scan (which can take a very long time)
- _____ - limit to specific ports, which is particularly useful for UDP scans

- _____ - enables OS detection and Version detection, Script scanning and Traceroute
- _____ - verbose (use multiple **-v**'s to make it more verbose)

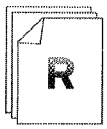
The Avahi Service

In the next exercise, you will be working with the **Avahi** service. The **Avahi** service is simply being introduced as a "typical" network service that many organizations might like to disable, and using or configuring **Avahi** is not an important part of this course.

Avahi is an implemention of Zeroconf "zero configuration networking" service discovery compatible with Apple Bonjour. Zeroconf permits machines on the same broadcast domain to communicate and discover each other's services without static networking, DHCP, or other explicit configuration.

One reason why **Avahi** may be interesting in the context of this section is that some services (such as **sshd**) may advertise the availability of open network ports through **Avahi**.

avahi-daemon must be running on systems that use **Avahi** to advertise or discover link-local services. **avahi-daemon** should *not* be running otherwise.



References

Red Hat Enterprise Linux Security Guide
- Section 2.2.8: Verifying Which Ports Are Listening

netstat(8) man page

Red Hat Enterprise Linux Security Guide
- Section 1.2.3.1: Scanning Hosts with Nmap

Nmap development site
http://www.insecure.org/

avahi-daemon(8) man page

Avahi development site
http://avahi.org/

Practice Performance Checklist

Closing Open Ports

Examine how the **Avahi** service presents itself on serverX.

- ☐ Log into serverX as root.
- ☐ Locally detect which ports the **Avahi** service is listening on. What command, with options, did you use to identify the open ports and which ports **Avahi** is listening on?
- ☐ Log into another window as root on desktopX.
- ☐ Scan the specific ports you identified in step 2 from desktopX using **nmap**. You should see the open **avahi-daemon** ports.
- ☐ Disable the **Avahi** service immediately and persistently on serverX.
- ☐ Redetect the open local ports on serverX. Does **avahi-daemon** have any open ports?
- ☐ Rescan the specific ports from desktopX and confirm the **avahi-daemon** ports are unavailable.

Capturing and Analyzing Network Traffic

Network sniffers are tools which allow the user to capture network traffic, which can simplify diagnosing and debugging network problems. Red Hat Enterprise Linux includes two network sniffers, the simple yet powerful **tcpdump** which simply captures packets, and the **Wireshark** utilities which can capture and parse network traffic for deeper analysis.

Warning

Since network sniffers can capture clear-text data transmitted over the network, some organizations restrict their use to authorized personnel only for security reasons. Also, organizations may have strict rules on when and how traffic may be captured and stored for a number of reasons. Ensure that you have permission to use such tools on your network and understand your organization's policies before working with them.

Capturing Packets with tcpdump

tcpdump can be used to capture network traffic to a file in a command-line only environment.

- List all available capture interfaces:

```
# tcpdump -D
```

- Capture all SSH network traffic:

```
# tcpdump -nn -l -s 2000 -w packets -i eth0 'port 22'
```

1. **-nn** = everything (including ports and protocols displays as numbers
2. **-l** = do line buffering to the file
3. **-s *snap_len*** = maximum number of bytes per packet to output
4. **-w *filename*** = file to write output to
5. **-i *interface*** = interface to capture
6. ***filter*** = keywords and logical operators used to filter packets (for example, **'host desktopX.example.com and port 25'**)

Use this space for notes

Analyzing Network Packets with wireshark

The **wireshark-gnome** package in the Optional repository is a graphical interface for the **Wireshark** network sniffer.

Note

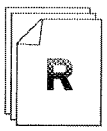
In addition to the functionality of Red Hat Enterprise Linux, Red Hat provides additional software, packaged with Red Hat's guidelines and security reviews, for the convenience of our customers. These packages cover both open source licensed software in an "Optional" Red Hat Network channel and proprietary licensed software in "Supplementary" Red Hat Network channel. These are not considered supported nor are the ABIs guaranteed.

1. Install the **wireshark-gnome** package on a system with X.
2. Copy the output file from **tcpdump** to the local machine.
3. Invoke **wireshark** (unprivileged) to examine the captured packets:

   ```
   $ wireshark filename
   ```

   The *filename* can also be opened through the **File** → **Open** pull-down menu item.
4. Select packets to view in the top frame (you can also filter the view)
5. Middle frame shows parsed packet header information
6. Bottom frame displays raw data

Use this space for notes



References

tcpdump(8) and **pcap-filter**(7) man pages

wireshark(1) man page

Wireshark Development Site
http://www.wireshark.org/

Practice Quiz

Packet Inspection

1. The _____ command is used on systems without X to capture network traffic to a file. The file can be _____ to a system with the _____ utility for further analysis.

2. _____ lists the available interfaces to capture network traffic from. The _____ is used to select the interface to use when capturing packets.

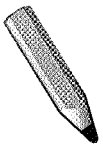
3. The _____ option to **tcpdump** specifies the maximum number of bytes per packet to output.

4. **tcpdump** creates a file with the captured packets when the _____ option is specified.

5. A _____ with _____ is the argument to **tcpdump** that more precisely specifies which network traffic to capture.

6. Files containing packet data can be analyzed graphically as an _____ using _____. This utility is provided by the _____ package.

7. The capture file to be analyzed can either be specified as an _____ or opened within _____ using the _____ pull-down menu item.



Test

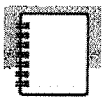
Criterion Test

Performance Checklist

Monitoring the HTTP Service

In this practice exercise, you will capture and analyze web traffic going between desktopX and serverX.

- ☐ Log in to serverX as root.
- ☐ If necessary, start the **httpd** service.
- ☐ Determine which ports are listening for connections. What is the status of port 80?
- ☐ Log into desktopX and externally scan the open ports of serverX. Is port 80 open?
- ☐ Use **tcpdump** on serverX to capture incoming HTTP connections to a capture file.
- ☐ While **tcpdump** is capturing packets on serverX, launch a web browser on desktopX and browse to http://serverX.example.com.
- ☐ Once you have browsed the site, stop **tcpdump** on serverX.
- ☐ Copy the capture file to desktopX.
- ☐ Use **wireshark** on desktopX to open the capture file for analysis.
- ☐ Find these in the captured traffic:
  - a packet from desktopX to serverX requesting web content
  - the packet(s) from serverX to desktopX that fulfill the HTTP request



Personal Notes

Unit Summary

Detecting Open Ports

In this section you learned how to:

- List "open ports" on a system
- List "remote open ports" on a system

Capturing and Analyzing Network Traffic

In this section you learned how to:

- Capture a sample of network traffic
- Distinguish frame, IP, and TCP/UDP layer information in captured traffic

UNIT SEVEN

ADVANCED NETWORK CONFIGURATION

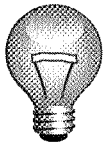
Introduction

Topics covered in this unit:

- IP alias configuration
- Configuring bonded network interfaces
- Tuning kernel parameters
- Static route configuration

Network Interface Configuration - IP Aliases

Assigning multiple IP addresses to a single interface is called IP aliasing. This is useful for situations such as web hosting where a single machine may run different services or sites on different IP addresses. DHCP does not support aliases.



Important

This course recommends disabling NetworkManager when configuring aliases and bonding. NetworkManager supports multiple IP addresses in a way that is not backward compatible with old-style network alias configuration as done in Red Hat Enterprise Linux 5. NetworkManager also currently doesn't work with NIC bonding. It is expected that both of these limitations will be addressed in the future.

For more information about the new method of persistently configuring multiple IP addresses, consult `http://live.gnome.org/NetworkManager/SystemSettings#ifcfg-rh`.

There are three basic steps to adding an IP alias:

1. Persistently disable NetworkManager

```
[root@demo ~]# service NetworkManager stop ; chkconfig NetworkManager off
Stopping NetworkManager daemon:                            [  OK  ]
```

2. Interactively add alias

```
[root@demo ~]# ip addr add 10.1.1.250/24 dev eth0 label eth0:0
[root@demo ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
 1000
    link/ether 52:54:00:00:00:fa brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.250/24 brd 192.168.0.255 scope global eth0
    inet 10.1.1.250/24 scope global eth0:0
    inet6 fe80::5054:ff:fe00:fa/64 scope link
       valid_lft forever preferred_lft forever
```

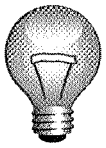
Persistently add alias by creating **`/etc/sysconfig/network-scripts/ifcfg-eth0:0`** with the following contents:

```
DEVICE=eth0:0
IPADDR=10.1.1.250
PREFIX=24
ONPARENT=yes
```

3. Restart the **`network`** service

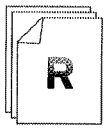
```
[root@demo ~]# service network restart
Shutting down interface eth0:                              [  OK  ]
Shutting down loopback interface:                          [  OK  ]
```

```
Bringing up loopback interface:                            [  OK  ]
Bringing up interface eth0:
Determining IP information for eth0... done.
                                                           [  OK  ]
```



Important

Avoid using the obsolete **ifconfig** command. If a system has new-style secondary IP addresses set on an interface that do not have a backward-compatibility IP alias label, **ifconfig** will not show the secondary address or addresses. Use the **ip addr** command instead.



References

Red Hat Enterprise Linux Deployment Guide

- Section 4.2.3: Alias and Clone Files

/usr/share/doc/initscripts-*/sysconfig.txt

Network Manager System Settings
http://live.gnome.org/NetworkManager/SystemSettings#ifcfg-rh

Network Interface Configuration - Bonding

Red Hat Enterprise Linux allows administrators to bind multiple network interfaces together into a single channel using the **bonding** kernel module and a special network interface called a channel bonding interface. Channel bonding enables two or more network interfaces to act as one, increasing bandwidth and/or providing redundancy, depending on the bonding mode chosen.

Physical Identification of a NIC

When working with multiple network cards, it is useful to be able to identify particular network cards physically. One method of physically identifying a NIC is to cause one or more of its LEDs to blink. To blink the LEDs on **eth0** for 30 seconds, run **ethtool -p eth0 30**.

Selected Linux Ethernet Bonding Modes

- Mode 0 (balance-rr) - Round robin policy, all interfaces are used. Packets are transmitted in a round-robin fashion through all slaves; any slave can receive.
- Mode 1 (active-backup) - Fault tolerant. Only one slave interface is in use at a time, but if it fails another slave takes over.
- Mode 3 (broadcast) - Fault tolerant. All packets are broadcast from all slave interfaces.

Other bonding modes are described in the kernel documentation **networking/bonding.txt** file.

Example Active-Backup Configuration

- **/etc/sysconfig/network-scripts/ifcfg-bond0**

  This file configures the network information for the bonded interfaces, as if it were a normal network interface file:

```
DEVICE=bond0
IPADDR=10.1.1.250
PREFIX=24
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
BONDING_OPTS="mode=1 miimon=50"
```

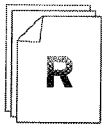
- **/etc/sysconfig/network-scripts/ifcfg-<name>**

  Each slave interface <*name*> needs a file containing the following configuration:

```
DEVICE=<name>
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
```

- **/etc/modprobe.d/bonding.conf**

```
alias bond0 bonding
```



References

Red Hat Enterprise Linux Deployment Guide
- Section 4.2.2: Channel Bonding Interfaces

Red Hat Enterprise Linux Deployment Guide
- Section 22.7.2: Using Channel Bonding

/usr/share/doc/kernel-*/Documentation/networking/bonding.txt

Practice Quiz

Network Bonding Configuration

1. Which mode of Linux Ethernet bonding primarily uses one slave interface and changes interface upon failure?

   (select one of the following...)
   a. Mode 0 (balance-rr)
   b. Mode 1 (active-backup)
   c. Mode 3 (broadcast)

2. Which mode of Linux Ethernet bonding uses all interfaces in a round robin fashion to achieve more throughput?

   (select one of the following...)
   a. Mode 0 (balance-rr)
   b. Mode 1 (active-backup)
   c. Mode 3 (broadcast)

3. When creating a bonded network interface, which configuration file contains the IP address and netmask definitions for the interface?

   (select one of the following...)
   a. **`/etc/sysconfig/network`**
   b. **`/etc/sysconfig/network-scripts/ifcfg-bond0`**
   c. **`/etc/sysconfig/network-scripts/ifcfg-`*`iface`***
   d. None of the above

4. When creating a bonded network interface, which configuration file defines the type of bonding?

   (select one of the following...)
   a. **`/etc/sysconfig/network`**
   b. **`/etc/sysconfig/network-scripts/ifcfg-bond0`**
   c. **`/etc/sysconfig/network-scripts/ifcfg-`*`iface`***
   d. None of the above

5. When creating a bonded network interface, which variable definitions must be specified in the **`/etc/sysconfig/network-scripts/ifcfg-`*`iface`*** configuration file?

   (select one of the following...)
   a. **`GATEWAY`**
   b. **`IPADDR`**
   c. **`MASTER`**
   d. None of the above

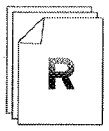
Tuning Kernel Network Parameters

Kernel parameters provide a mechanism to adjust the functioning of the Linux kernel. Generally speaking, whenever a kernel developer selects an arbitrary constant or implements functionality that may not be generally desired, a **sysctl** will be made available to adjust it. Commonly useful parameters will be documented online, in **kernel-doc** or in this and other Red Hat courses.

These parameters can be viewed or set via the **/proc/sys/** directory tree or the **sysctl** command.

Search & Learn: Kernel Tuning

1. Perform the following steps on serverX.
2. Install the **kernel-doc** RPM if it is not already installed.
3. How would you use **sysctl** to identify kernel parameters that control ping, or ICMP echo, behavior?
4. Which parameters look promising?
5. What command would you use to identify and/or examine kernel documentation that describes what those parameters are for?
6. How would you use **sysctl** to adjust kernel parameters to "hide" your system from ping requests?
7. How would you configure **sysctl** to persistently adjust kernel parameters to survive a reboot?



References

Red Hat Enterprise Linux Deployment Guide
- Section 19.3.9.4: /proc/sys/net/

Red Hat Enterprise Linux Deployment Guide
- Section 19.4: Using the sysctl Command

/usr/share/doc/kernel-doc-*/Documentation/sysctl/

/usr/share/doc/kernel-doc-*/Documentation/networking/ip-sysctl.txt

sysctl(8) man page

Practice Performance Checklist

Enable Ping Broadcast

The default configuration for Red Hat Enterprise Linux 6 configures the kernel to ignore ping broadcast requests. You will work with a partner to tune the kernel on serverX to respond to them instead.

- ☐ Find a partner to work with. If there is an odd number of students, a group of three will work.
- ☐ Send a broadcast ping to the 192.168.0.0/24 network. Note which hosts respond to the ping request.
- ☐ Tune serverX so that it will respond to ping broadcasts.
- ☐ Send another broadcast ping to the 192.168.0.0/24 network. Did your hosts respond?
- ☐ Persistently configure your serverX machines to respond to ping broadcasts and reboot.
- ☐ Send another ping broadcast. Did your configuration changes persist the reboot?

Static Route Configuration

The routing table for the system is displayed with the **ip route show** command.

```
[root@desktopX ~]# ip route show
192.168.0.0/24 dev br0  proto kernel  scope link  src 192.168.0.1
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1
default via 192.168.0.254 dev br0
```

The network of each local interface address is automatically placed in the above routing table. Note the **scope link src** which identifies that interface's address.

Enabling Kernel Routing

Routing must be enabled with a tunable kernel parameter, named **net.ipv4.ip_forward**.

net.ipv4.ip_forward tells the kernel what to do with foreign packets sent to it that have a destination address that does not match any addresses of the local system:

- **0** (default) - discard foreign packets
- **1** - forward foreign packets to the network interface determined by the routing table

Edit **/etc/sysctl.conf** to persistently change this setting, then run **sysctl -p** to reload.

Note

Although the focus of this course is IPv4 configuration, routing can also be enabled for IPv6. For more information, review kernel documentation for the **net.ipv6.conf.all.forwarding** kernel parameter.

Adding a Static Route to the Route Table

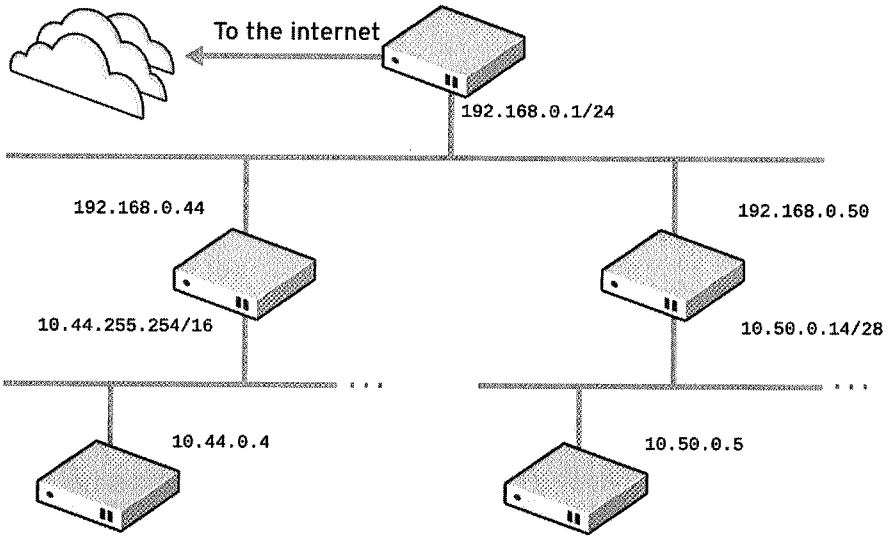
To add a route dynamically:

```
[root@serverX ~]# ip route add network/netmask via router_ip
```

To persistently add a route, edit the **/etc/sysconfig/network-scripts/route-*iface*** file, adding three lines (*X is an index for each route, starting at 0.*):

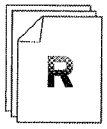
```
ADDRESSX=network
NETMASKX=netmask
GATEWAYX=router_ip
```

Routing Example Network Diagram



Network diagram assumptions:

- 192.168.0.1 is the default gateway for the 192.168.0.0/24 network and it is a NATing firewall.
- 10.44.255.254 is the default gateway for the 10.44.0.0/16 network.
- 10.50.0.14 is the default gateway for the 10.50.0.0/28 network.
- All hosts with 2 NICS have IP forwarding enabled.
- All hosts respond to ICMP requests and there is no filtering of ICMP traffic on the networks.
- There are no additional static routes.



References

Red Hat Enterprise Linux Deployment Guide

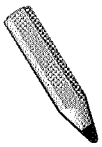
- Section 4.4: Configuring Static Routes

Practice Performance Checklist

Static Route Configuration

Get back together with your partner(s). One of you will work on your system called desktopX and the other will be called desktopY for this lab. The virtual machines running on them will be serverX and serverY respectively.

- ☐ Log in to desktopX and desktopY and run the **`lab-setup-routing`** script on both systems.
- ☐ When the virtual machines finish rebooting, determine the IP addresses of serverX and serverY.
- ☐ Tune desktopX and desktopY to support IP forwarding for IPv4 if it isn't configured already.
- ☐ Interactively configure a route on desktopX to reach the 10.Y.0.0/24 network through desktopY. Perform the following tests and note your results:
  - Can desktopX ping serverY?
  - Can serverX ping serverY?
  - Can desktopY ping serverX?
  - Can serverY ping desktopX?
- ☐ Interactively configure a route on desktopY to reach the 10.X.0.0/24 network through desktopX. Repeat the following tests and note your results:
  - Can desktopX ping serverY?
  - Can serverX ping serverY?
  - Can desktopY ping serverX?
  - Can serverY ping desktopX?
- ☐ Persistently configure desktopX and desktopY to implement the static routes needed for serverX and serverY to communicate with each other. Reboot desktopX and desktopY.
- ☐ Confirm that serverX and serverY can ping each other. If they cannot ping each other, then diagnose the routing issue and correct it.
- ☐ Execute the **`lab-cleanup-routing`** script on desktopX and desktopY to restore the virtual machines back onto the classroom network.



Test

Criterion Test

Case Study

Routing Network Traffic

Before you begin...

Run the **lab-setup-oshu** script on desktopX to prepare serverX for the exercise.

Operation Strategic Holistic Unusual (or OSHU), is an online chat system for fans of conspiracy fiction. In order to join the site they have two requirements, listed below.

1. To fulfill the first requirement, you must prove your ability to "disappear" a server. You will do this by modifying the configuration on serverX so that it does not respond to any ping requests. Make this change persistent so that it will still be in effect after a reboot.

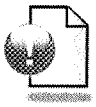
2. The second requirement is to join the "secret" OSHU network. To join the network, add an additional IP address to serverX, where X is your desktop/server number:

   10.42.10.X/24

When you have fulfilled the requirements, run **lab-grade-oshu** on desktopX to check your work.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Personal Notes



Unit Summary

Network Interface Configuration - IP Aliases

In this section you learned how to:

- Manually configure multiple IP addresses on one network card

Network Interface Configuration - Bonding

In this section you learned how to:

- Combine two network interfaces into one bonded interface

Tuning Kernel Network Parameters

In this section you learned how to:

- Make changes to kernel tuning parameters that affect networking settings

Static Route Configuration

In this section you learned how to:

- Configure static routes to different subnets

UNIT EIGHT

SECURE NETWORK TRAFFIC

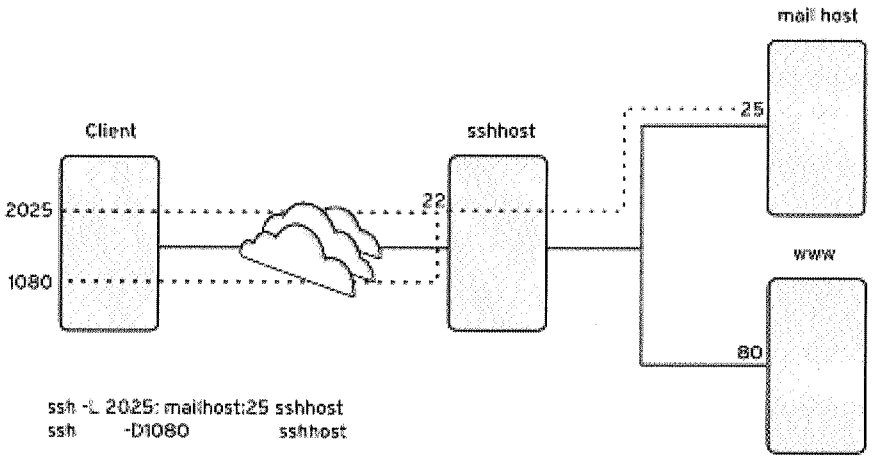
Introduction

Topics covered in this unit:

- SSH port forwarding
- Packet filtering
- Network address translation (NAT)

SSH Port Forwarding

The **ssh** utility can be used to tunnel or proxy network traffic through a firewall over an encrypted connection. It does this through a feature called *SSH port forwarding*, which can be configured in a number of different ways.



In the above network diagram, **sshhost** is a firewall bastion that sits between a client on the public network and a couple servers on a private network. The SSH port, tcp/22, is the only open port on the public interface. **sshhost** does not accept or route any other traffic from the public network.

The **ssh -L** *localport:remotehost:remoteport* **sshhost** command will forward connections from *localport* on the client over the secure channel to **sshhost**. When an application tries to connect to *localport* on the client, **ssh** forwards that connection over the secure channel to **sshhost**. The **sshd** daemon running on **sshhost** then connects to *remotehost:remoteport*, completing the tunnel.

Note: **sshhost** resolves "remotehost", not the client. For example **ssh -L 8080:localhost:80 sshhost** forwards port 8080 on the client to port 80 on **sshhost**.

The **ssh -D localport sshhost** command allows you to point your client's web browser to "localhost:localport" and all HTTP traffic would be forwarded over the secure channel to **sshhost**. It is **sshhost** that decides where to send this traffic. The **-D** option does this by having **sshhost** act as a SOCKS server and saves you from having to specify multiple **-L** options to forward to various remote hosts.

There will be certain situations when you still need to use **-L**, for example when the client application does not support SOCKS. Many commands you use every day, like X11 forwarding (**ssh -X**) and secure VNC (**vncviewer -listen**), are automatically setting up SSH tunnels for you. Now you know how to set up a secure tunnel for other services.

Two other options to **ssh** are useful with **ssh** tunnels:

- **-N**

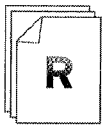
  This will not execute any remote commands (including **bash**). This will appear to hang the connection. Press **Ctrl+c** to drop the connection.

- **-f**

  This will place the **ssh** connection in the background just before running any remote commands.

The **-N** and **-f** options are often used together for a tunnel like the following:

```
[student@desktopX ~]$ ssh -Nf -L 2025:mailhost:25 sshhost
```



References

Red Hat Enterprise Linux Deployment Guide

- Section 9.4.2: Port Forwarding

Practice Exercise

Use SSH Port Forwarding

Before you begin...

Login as **student** on serverX and use **nc** to verify an SMTP service is accepting connections from localhost (i.e., **nc localhost 25** on serverX). You may have to install the **nc** package.

Carefully perform the following steps. Ask your instructor if you have problems or questions.

Your desktopX and serverX machines are running a local SMTP service that accepts connections only from localhost. In this exercise you will use **ssh** to connect to serverX's SMTP server from desktopX via a secure encrypted tunnel.

1. Login as **student** on desktopX.
2. Use **nc** to verify an SMTP service is running on desktopX.
3. Use **nc** to verify that the SMTP service does not allow connections from desktopX to serverX.
4. Run **ssh** to forward desktopX port 2025 to port 25 on serverX through a tunnel.
5. Open another window on desktopX and use **nc** to verify desktopX port 2025 connects to serverX's SMTP service. Note the name of the host displayed in the SMTP welcome banner.

Packet Filtering

The following is a list of key concepts you will need to know in order to setup a firewall. Pay attention as the class discusses each one and take notes in your books because you will have to use all of these keywords when creating your firewall in lab.

`iptables` Basics

- _____ - criteria determining which packets to match and a target, or action, determining what to do with those packets.
- _____ - a list of *rules* which will be checked in order, first match takes effect.
- _____ - the default action, **ACCEPT** or **DROP**, taken if no *rule* matches in a built-in *chain*.
- _____ - a set of *chains* used for a particular purpose: **`filter`** to block traffic, **`nat`** to modify the destination or apparent source of a packet.

Built-in Chains (`filter` table)

- _____ - packets addressed to the firewall
- _____ - packets originating from a service on the firewall (not forwarded)
- _____ - packets originating from another machine, that are not addressed to the firewall but are being forwarded (routed) elsewhere (when **`net.ipv4.ip_forward=1`**)

Targets

(Actions to take when packets match rules)

- _____ - the packet passes the chain
- _____ - the packet is dropped as if it was never seen
- _____ - the packet is rejected, and the firewall sends an error message (an ICMP port unreachable message by default)
- _____ - information about the packet is logged to syslog; we go on to the next rule in the chain

iptables Command

You may have used **`system-config-firewall`**, a graphical tool in Red Hat Enterprise Linux 6, to configure simple firewalls. Creating and managing more advanced configurations can be accomplished with the command line tool, **`iptables`**.

iptables is used to set or view rules in kernel memory.

iptables Options	*Definition*
-vnL --line-numbers	lists all rules, fully, in numeric mode
-A CHAIN <rule> -j <target>	adds a *rule* to the end of *CHAIN*
-I CHAIN # <rule> -j <target>	inserts a *rule* as rule # in *CHAIN*; if no #, then as the first rule
-D CHAIN #	deletes rule # from *CHAIN*
-F CHAIN	deletes all rules from *CHAIN*

Table 8.1. iptables Example Syntax

Rule (matching criteria) Syntax

An iptables rule includes matching criteria that can compares to header information found in the packet.

Concept	*Directive*
Source IP or network	**-s 192.0.2.0/24**
Destination IP or network	**-d 10.0.0.1**
UDP/TCP and ports	**-p udp --sport 68 --dport 67**
ICMP and types	**-p icmp --icmp-type echo-reply**
Inbound network interface	**-i eth0**
Outbound network interface	**-o eth0**
Connection tracking	**-m state --state ESTABLISHED,RELATED**

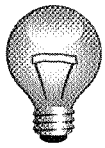
Table 8.2. iptables Matching Criteria

Connection tracking stores information about previously seen communications to make matching decisions. After a connection is allowed, information is placed in a connection tracking table until a timeout occurs, connections closes, or we see more matching traffic (reset timer). While this takes additional kernel memory, the benefit is to simplify rule design.

State	*Definition*
NEW	packet starts a new communication, adds a rule to the connection tracking table
ESTABLISHED	any packet that matches a rule in the connection tracking table
RELATED	traffic "related" in some way to ESTABLISHED traffic; protocols like FTP
INVALID	packet cannot be identified; normally these should be rejected or dropped

Table 8.3. Connection Tracking States

To help RELATED rules work, you may need to enable helper modules in **/etc/sysconfig/iptables-config**

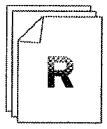


Important

Running the **iptables** command changes the netfilter kernel module rules in memory, but will NOT persist across a reboot.

Running **service iptables save** will take the current rules in memory and write them to **/etc/sysconfig/iptables** which is read during startup.

Alternatively, some administrators directly edit (or copy) the **/etc/sysconfig/iptables** file, then run **service iptables restart** to activate.



References

Red Hat Enterprise Linux Security Guide
- Section 2.5: Firewalls

Red Hat Enterprise Linux Security Guide
- Section 2.6: IPTables

iptables(8) man page

> Netfilter home page
> http://www.netfilter.org/

Practice Exercise

Implement a Firewall

Carefully perform the following steps. Ask your instructor if you have problems or questions.

In this exercise you will implement a firewall on serverX that rejects all packets, except that it will allow ICMP traffic for example.com and allow SSH for everyone.

1. Log into serverX as **root** using **virt-viewer** or **virt-manager**.
2. Create a simple deny all (except loop back) firewall by creating **/root/bin/resetfw.sh** that:
    1. sets the **INPUT** chain's default policy to **DROP**,
    2. flushes all rules in the filter table, and
    3. will **ACCEPT** all packets from the loopback interface
3. Run your script and record the results of the following:
    - **ping** and **ssh** serverX from desktopX and from remoteX.remote.test
4. What happens when you **ping** desktopX and 192.168.0.X from serverX now? Why?
5. Enable stateful firewalling by appending to your script a rule that will
    - **ACCEPT** all **ESTABLISHED, RELATED** packets
6. Run your script and record the results of the following:
    - **ping** desktopX and 192.168.0.X from serverX
7. Reject all packets from remote.test by appending to your script a rule that will
    - **REJECT** all packets from the 192.168.1.0/24 network
8. Run your script and record the results of the following:
    - **ping** and **ssh** serverX from desktopX and from remoteX.remote.test
9. Enable ICMP traffic for example.com by appending to your script a rule that will
    - **ACCEPT** all **icmp** traffic from 192.168.0.0/24
10. Run your script and record the results of the following:
    - **ping** and **ssh** serverX from desktopX
11. Enable SSH traffic for all hosts by modifying your script to
    - **ACCEPT** all **NEW** connections to **tcp** port **22**
12. Run your script and record the results of the following:

- **ssh** to serverX from desktopX and from remoteX.remote.test

13. Reject packets by default instead of dropping packets by appending to your script a rule that will:

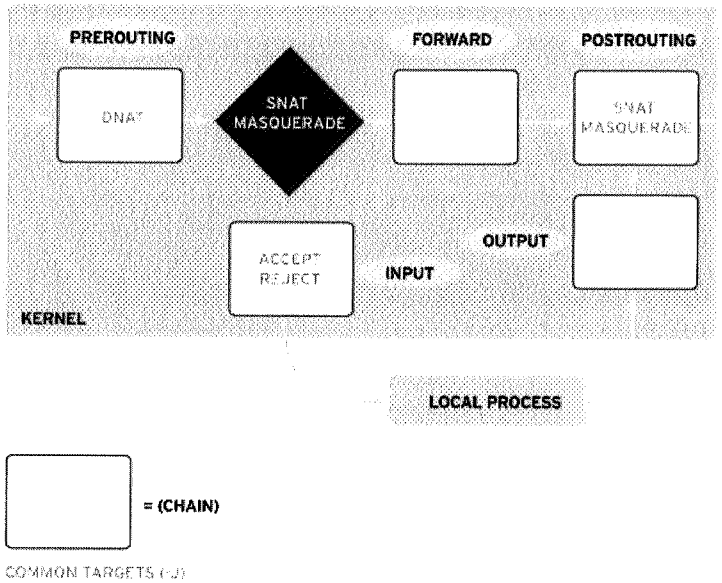
    - **REJECT** all other traffic

14. Run your script and record the results of the following:

    - **ping** and **ssh** serverX from desktopX and from remoteX.remote.test

Network Address Translation

Network Address Translation is used to manipulate the apparent source or desired destination address of network packets. The diagram below shows the order in which Netfilter chains on the **filter** and **nat** tables are processed:



Simple host-based firewalls may only have rules in the **INPUT** chain to **ACCEPT** or **REJECT** packets, but on a gateway or router for a private (non-routable) network, it is common to use the **PREROUTING** and **POSTROUTING** chain to modify packets.

The **nat** table uses three chains: **PREROUTING**, **OUTPUT**, and **POSTROUTING**. Network Address Translation is when a router modifies the source or destination IP address or port of network traffic passing through it. It is used for mapping a network of machines behind a single IP address, so that they may share a single public address and hide their internal network (**MASQUERADE** or **SNAT**). It is also used for redirecting traffic addressed to one IP address to another. This *destination NAT* is used for *port forwarding* (passing a port outside a firewall to a service inside it) and for transparent redirection to proxy services.

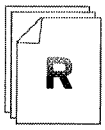
The **MASQUERADE** target causes the source IP address to be changed to match the IP of the interface it leaves the firewall on. Destination sends response back to the IP address of that interface. Connection tracking automatically translates the return traffic to its matching internal IP addresses and ports (tracks based on IP addresses and ports of both ends of the connection). The **SNAT** target causes the source IP address to change to a specified IP address with the option **--to-source**.

```
[root@demo ~]# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

The **DNAT** target causes the destination IP address to be changed to match the IP address specified be the **--to-destination** option. Router forwards packet to that address; this is why this chain is before the routing decision. Connection tracking automatically sends responses back to the original source with the original IP address on it, not the new one.

```
[root@demo ~]# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination
 192.168.0.254
```

Use this space for notes



References

Red Hat Enterprise Linux Security Guide
- Section 2.5: Firewalls

Red Hat Enterprise Linux Security Guide
- Section 2.6: IPTables

Practice Quiz

Network Address Translation

1. The chains available in the **filter** table are _____, _____, and _____

2. The chains available in the **nat** table are _____, _____, and _____

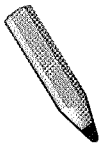
3. **iptables -t** _____ **-A** _____ **-o eth0 -j MASQUERADE**

4. **iptables -t** _____ **-A** _____ **-o eth0 -j SNAT** _____ **192.168.0.1**

5. **iptables -t** _____ **-A** _____ **-i eth0 -m tcp -p tcp --dport 80 -j DNAT** _____ **192.168.0.100:8080**

6. The **DNAT** target can only be used in the _____ chain and the _____ chain of the _____ table

7. To enable forwarding persistently across reboots add **net**_____**=1** to **/etc/**_____ and run _____ **-p**



Test

Criterion Test

Case Study

The Morris Worm and Fish Supply Company

Before you begin...

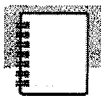
Run **`lab-setup-morrisworm`** on desktopX to prepare serverX for the exercise.

The Morris Worm and Fish Supply company is finally looking to modernize its business by opening a website. The web server will run on a private network behind a firewall. The firewall will forward all TCP port 80 traffic to the web server and will perform NAT so that the web server can reach external hosts.

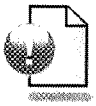
- desktopX.example.com will be the firewall, serverX.example.com will be the web server.
- Configure Apache to run on serverX.example.com. Put some custom content in **`/var/www/html/index.html`** that will uniquely identify the server.
- Configure the firewall on desktopX to perform NAT that will allow the web server to reach the outside network. You will be able to successfully **`ping`** instructor.example.com from serverX to confirm this works.
- Finally configure the firewall to forward all TCP port 80 traffic sent to it to the web server running on serverX. You will need to identify serverX's IP address to complete this step. Confirm this works by using a web browser from an external machine, NOT desktopX, to browse http://desktopX.example.com.

After you have successfully completed the lab, run **`lab-cleanup-morrisworm`** on desktopX to reset your network back to its original state.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Packet Filtering

In this section you learned how to:

- Set firewall rules with **iptables**
- Block or allow network traffic based on specific criteria
- Block or allow network traffic based on previous traffic seen

Network Address Translation

In this section you learned how to:

- Use **iptables** to set up IPv4 Network Address Translation
- Make packets passing through the Linux router appear to come from its outbound IP address
- Redirect packets passing through the Linux router to a different destination IP address

UNIT NINE

NTP SERVER CONFIGURATION

Introduction

Topics covered in this unit:

- Configuring time servers

Configure an NTP Server

NTP is the Network Time Protocol, a standard way for machines to provide and obtain correct time information on the Internet. A machine may get accurate time information from public NTP services on the Internet such as the NTP Pool Project, or from high-quality hardware clocks, which it then may itself serve to local clients.

Each server has a stratum classification. This defines how close it is to a reliable real-time hardware clock (which is stratum 0). So the most accurate NTP servers are stratum 1 servers. Stratum 2 servers do not have hardware clocks but they synchronize with stratum 1 servers and so on.

To configure an NTP server and client, you need to understand three main parameters in the **/etc/ntp.conf** file: **server**, **peer**, and **restrict**.

The first argument of the **server** line is the IP address or DNS name of the NTP server. Your IP address must be allowed access on the server with a **restrict** line discussed below. Following the server IP address or name, you can list a series of options for the server. The **ntp.conf**(5) man page recommends using the **iburst** option.

Like the **server** line, the **peer** line takes an NTP server and options arguments. The **server** is one stratum above your NTP server, and the **peer** is at the same stratum. You can specify more than one **server** and more than one **peer**, one per line.

The **restrict** line usually takes an IP address or DNS name as the first argument, however the first argument could be **default** to specify the restrictions apply as default settings. When there are multiple **restrict** directives that could apply to a given host, the directive with the most specific host/network specification matches. If the first argument is **-6**, the following restrictions (including **default**) apply to IPv6 addresses only. The **restrict** line has several flags that can be used to limit NTP functionality. When no flags are specified the matching host or network has no NTP restrictions. Some of the most common flags are explained below (taken from the **ntp_acc**(5) man page:

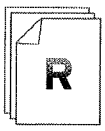
- **ignore**: Deny packets of all kinds, including **ntpq** and **ntpdc** queries.
- **kod**: Normally NTP silently ignores packets that violate access. When this flag is set a kiss-o'-death (KoD) packet is sent that notifies the client it hit an access violation. KoD packets are rate limited to no more than one per second. If another KoD packet occurs within one second after the last one, the packet is dropped.
- **nomodify**: Deny **ntpq** and **ntpdc** queries which attempt to modify the state of the server (i.e., run time reconfiguration). Queries which return information are permitted.
- **noquery**: Deny **ntpq** and **ntpdc** queries. Time service is not affected.
- **nopeer**: Deny packets which would result in mobilizing a new association. This includes broadcast, symmetric-active and manycast client packets when a configured association does not exist.
- **notrap**: Decline to provide mode 6 control message trap service to matching hosts. The trap service is a subsystem of the ntpdq control message protocol which is intended for use by remote event logging programs.

If the client time is off by more than just a few minutes, the NTP client will fail to sync with the server. You can use the **ntpdate -b *ntpserver*** command to roughly set the clock once so that it can synchronize.

If you want to include a hardware clock available to the machine as part of the NTP service, it uses a special IP address in the range **127.127.*type*.*unit***. This could be an accurate GPS or radio clock, or the inaccurate RTC built into the system (which has address 127.127.1.0).

Example: The real-time clock (RTC) is used to keep track of time on the motherboard. This clock is usually not very accurate. For this reason, if you use it you should force it to advertise at a lower stratum (normally 10). The section in **/etc/ntp.conf** would look like the following:

```
server          127.127.1.0
fudge           127.127.1.0       stratum 10
```



References

Red Hat Enterprise Linux Deployment Guide
- Section 13.2.2: Network Time Protocol Setup

ntp.conf(5) and **ntp_acc**(5) man pages

/usr/share/doc/ntp-*/html (from the **ntp-doc** package)

NTP Pool Project
http://www.pool.ntp.org/

Practice Quiz

Configuring NTP

Answer the questions below based upon the following NTP configuration file:

```
#/etc/ntp.conf

restrict default kod nomodify notrap nopeer noquery
restrict -6 default ignore

restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap nopeer
restrict 192.168.0.101 kod nomodify notrap
restrict 192.168.0.200

server 192.168.0.2
server 192.168.0.3
peer 192.168.0.101
```

1. The NTP client's time is off by 15 minutes, it will eventually sync with the servers.

   (select one of the following...)
   a. True
   b. False

2. The NTP client will use the computer's RTC (BIOS) as a time source.

   (select one of the following...)
   a. True
   b. False

3. 192.168.0.200 will be able to modify the time on this NTP server.

   (select one of the following...)
   a. True
   b. False

4. 192.168.0.4 will be able to query this NTP server.

   (select one of the following...)
   a. True
   b. False

5. 192.168.0.3 will be able to use this NTP server as a peer.

   (select one of the following...)
   a. True
   b. False

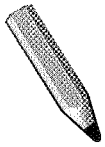
6. Anyone with an IPv4 address will be able use this NTP server as a time source.

   (select one of the following...)

a. True
b. False

7. Anyone with an IPv6 address will be able use this NTP server as a time source.

(select one of the following...)
a. True
b. False



Test

Criterion Test

Case Study

NTP Server Configuration

Before you begin...

Run **`lab-setup-howsonclock`** on desktopX.

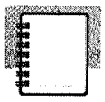
Howson Heavy Machine and Clock Manufacture, maker of clock tower parts and accessories, recently conducted an audit of all computer systems. The audit revealed several systems with out of sync clocks, including your serverX.example.com machine.

Set up NTP on your serverX to be a client of the NTP service running on instructor.example.com

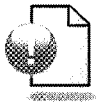
In order to have additional time sources, work with a few neighbors so that all of your serverX systems are set up to synchronize as NTP peers

When you have finished, run **`lab-grade-howsonclock`** on serverX to check your work.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Configure an NTP Server

In this section you learned how to:

- Setup a cluster of NTP servers as peers and use lower strata service
- Configure clients to use your local cluster of NTP servers

UNIT TEN

SYSTEM MONITORING AND LOGS

Introduction

Topics covered in this unit:

- System monitoring scripts
- Intrusion detection tools
- Log file management
- Centralized logging

Usage Reports

In this section we will begin by looking at a number of tools that are useful when writing scripts or automated tasks to collect reports on the usage of the system.

The class will be split into groups and given the commands below to research. For your group's command, what information does it provide? Write down how to generate a report with the command. Include useful options.

Once you have taken notes on your command, turn to the Case Study on the following page and follow the directions there.

At the end of this section, the other groups will share their findings and the instructor will review the commands. Be sure to take notes below on the commands that your group did not research.

Usage Reports Buzz Groups

1. Create a disk space usage report using **`df`**.

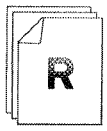
   Use this space for notes

2. Create a disk I/O usage report using **`iostat`**.

   Use this space for notes

3. Create a swap usage report using **`vmstat`**.

   Use this space for notes



References

df(1), **iostat**(1), and **vmstat**(8) man pages

Practice Case Study

Usage Reports

Use the tool you investigated to create a simple report that logs information to a file.

Once you have used the tool to generate a report, the instructor will have you share the command you used and explain the output with the rest of the class.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Monitor Systems with AIDE and sar

AIDE, the **Advanced Intrusion Detection Environment**, is a tool to check the integrity of files on the system. Essentially, when the system is in a known-good state, it is used to scan the system and collect information about installed files, their checksums, their permissions, and other characteristics. This information is then placed in a database file which can be stored off-line. Then, if you later suspect some compromise or other problem with file integrity, you can use AIDE to compare the state of the system against the stored database and check for any changes.

Steps to deploy AIDE

1. Install the **aide** package.
2. Customize **/etc/aide.conf** to your liking.
3. Run **/usr/sbin/aide --init** to build the initial database.
4. Store **/etc/aide.conf**, **/usr/sbin/aide** and **/var/lib/aide/aide.db.new.gz** in a secure location.
5. Copy **/var/lib/aide/aide.db.new.gz** to **/var/lib/aide/aide.db.gz** (the expected name).

Example /etc/aide.conf File

```
@@define DBDIR /var/lib/aide ❶
@@define LOGDIR /var/log/aide

database=file:@@{DBDIR}/aide.db.gz ❷
database_out=file:@@{DBDIR}/aide.db.new.gz ❸
gzip_dbout=yes
report_url=file:@@{LOGDIR}/aide.log ❹
report_url=stdout

# R is short for p+i+n+u+g+s+m+c+acl+selinux+xattrs+md5
NORMAL = R+rmd160+sha256 ❺
PERMS = p+i+u+g+acl+selinux

/ NORMAL ❻
!/etc/.*~
/root/..* PERMS
```

❶ Defining macros that can be used in **/etc/aide.conf**
❷ Configuration directive defining the location of the AIDE database. Note that this example uses a macro defined above.
❸ Configuration directive defining the location in which **aide --init** will save a newly created database file.
❹ Where the results of **aide --check** will be reported. Note that multiple locations are allowed.
❺ Group definition line. Files selected by AIDE in group **NORMAL** will store information about its regular permissions, inodes, number of links, user and group, size, mtime and ctime,

POSIX ACLs, SELinux context, extended attributes, MD5 checksum, RMD160 checksum, and SHA256 checksum.

6. Selection lines. The first one adds all files under **/** to be checked in group **NORMAL**; the second exempts all files in **/etc** that end in ~ from being checked; the third specifies that all files under **/root** that start with a **.** should be checked in group **PERMS** only. Note that this uses regular expression syntax.

Verifying system integrity with AIDE

- Run **/usr/sbin/aide --check** to check your system for inconsistencies.
- Results will be displayed on standard output and in **/var/log/aide/aide.log** by default.

Warning

If a system has had its root account or kernel compromised by an attacker, the installed version of AIDE or local copy of the database file may have been modified by the attacker or respond with false results. In this case, it is a good idea to boot the system with a known-good operating system environment with an off-line copy of AIDE and a copy of the backed-up AIDE database.

System Activity Reporter

Another useful system monitoring tool is **sar**, the **System Activity Reporter**, provided by the **sysstat** package. What **sar** does is collect information about system activity from the operating system at a particular point in time. It normally takes a sample of data over a selected time period, either once or on some repeating schedule. The information it collects can have to do with memory usage, disk I/O, network activity, and so on.

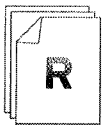
There are two modes in which **sar** operates. When **sysstat** is installed, a cron job is set up that takes a one second sample of system activity every ten minutes and saves it to a file. The **sar** command can be used to read this information. Otherwise, you can run **sar** from the command line to collect specific data, averaged over a certain period of time in seconds, a specified number of times.

Deploying the sar command

- Install the **sysstat** package. This package provides cron scripts that will gather data automatically.

Example sar commands

- Run **sar -A** to display all information collected today.
- Run **sar -u 2 5** to display five samples of system CPU usage spaced 2 seconds apart.



References

AIDE Quick Start

/usr/share/doc/aide-*/README.quickstart

AIDE Manual

/usr/share/doc/aide-*/manual.html

sysstat FAQ

/usr/share/doc/sysstat-*/FAQ

aide(1), **aide.conf**(5), and **sar**(1) man pages

Practice Resequencing Exercise

Resequencing Quiz: Implementing AIDE

__ Copy **/var/lib/aide/aide.db.new.gz** to **/var/lib/aide/aide.db.gz**

__ Run **/usr/sbin/aide --init** to build the initial database

__ Run **/usr/sbin/aide --check** to check your system.

__ Customize **/etc/aide.conf**

__ Store **/etc/aide.conf**, **/usr/sbin/aide** and **/var/lib/aide/aide.db.new.gz** in a secure location.

Tuning tmpwatch and logrotate

tmpwatch and logrotate Keywords

Disk storage is a limited resource. It can be wasted with stale scratch files that never get cleaned up. Also, unmanaged log files can grow until they consume all available disk space. A solution to these problems is to *watch* for stale scratch files and to *rotate* log files periodically.

Watch (**tmpwatch**)

The **tmpwatch** utility is used to purge stale temporary files from directories. The **/etc/cron.daily/tmpwatch** script runs nightly and removes files not accessed in the last ten days from **/tmp** and not accessed in the last thirty days from **/var/tmp**. You can adjust this script to change the schedule or add or remove directories from its management.

Use this space for notes

Rotate (**logrotate**)

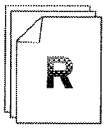
The **logrotate** utility also runs nightly in the **/etc/cron.daily/logrotate** script. It "rotates" logs so that they do not grow indefinitely, based on the configuration settings in its **/etc/logrotate.conf** and **/etc/logrotate.d/*** files. What this means is that the old log file **filename** is renamed **filename-*datestamp*** and a new log file **filename** is started. Eventually, after a few rotations, **logwatch** will throw away the oldest log file on the system to conserve space. At installation time, a program can add its log files to **logrotate** management simply by dropping an appropriate configuration file into the **/etc/logrotate.d/** directory.

Use this space for notes

Example **logrotate** configuration file:

```
/var/log/path-to-log-file {
   rotate 3
   size 2M
#  monthly   # another rotation alternative
   postrotate
      /sbin/service httpd reload
   endscript
}
```

In this example, three copies of **/var/log/path-to-log-file** will be kept before the oldest one is discarded. The log file will be rotated when **logwatch** runs if the log file exceeds 2 MB in size. After the old log file is moved, the **httpd** service will be reloaded, which will presumably close the old log file and start a new one.



References

Red Hat Enterprise Linux Deployment Guide

- Chapter 17: Log Files

Practice Exercise

Tuning tmpwatch and logrotate

Carefully perform the following steps. Ask your instructor if you have problems or questions.

Login as root and perform the following steps on serverX.

1. Adjust **/etc/cron.daily/tmpwatch** to remove files in **/tmp** that have not been accessed in seven (7) days.
2. Configure the global **logrotate** default setting to compress all log files when they are rotated.

Configure a Remote Logging Service

Remote Logging Search & Learn

Given the comments found in the **/etc/rsyslog.conf** file, complete the tasks listed below. Additional information relevant to these tasks can be found in **/usr/share/doc/rsyslog-*/rsyslog_conf_actions.html**.

1. Configure serverX to accept remote log messages using UDP.

   - Review the **/etc/rsyslog.conf** configuration file. What modifications would you have to make to accept remote log messages using UDP?

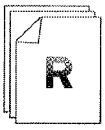
   - Make the necessary modifications and reload the **rsyslog** service on serverX.

2. Configure desktopX to send log messages via UDP to serverX.

   - What modification to **/etc/rsyslog.conf** on desktopX will cause all **info** priority and higher events to be sent to serverX using UDP?

   - Make the necessary modifications and reload the **rsyslog** service on desktopX.

3. Test your configuration by running **logger "Hello from desktopX"** on desktopX. Verify the message was received on both desktopX and serverX.

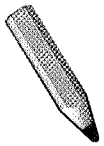


References

General **rsyslog** Documentation
/usr/share/doc/rsyslog-*/manual.html

rsyslog Configuration File Documentation
/usr/share/doc/rsyslog-*/rsyslog_conf_actions.html

rsyslog.conf(5) man page



Test

Criterion Test

Case Study

System Monitoring and Logs

Before you begin...

Run the **lab-setup-bloomin** script on desktopX to prepare serverX for the assessment.

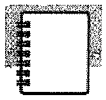
Every Bloomin' Thing is a nation wide cooperative of flower and plant growers. Among other things, the cooperative handles IT services for all members. The IT manager has decided to beef up security by requiring file integrity checking and remote logging on all servers, including your serverX.

Install **aide** on serverX and initialize the file integrity database. Do not wait for the database build to complete before continuing with the rest of the lab.

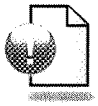
Configure rsyslog on desktopX to accept incoming log messages via UDP from serverX. Then configure **rsyslog** on serverX to send all ***.info** log messages to desktopX via UDP.

When you are ready to check your work, first run **lab-grade-bloomin** on serverX and then run it on desktopX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Usage Reports

In this section you learned how to:

- Verify and monitor filesystem integrity with tools like **df**, **iostat** and **vmstat**

Monitor Systems with **AIDE** and **sar**

In this section you learned how to:

- Configure intrusion detection and existing system monitoring tools to protect your system

Tuning **tmpwatch** and **logrotate**

In this section you learned how to:

- Manage system log files by monitoring their content and disk space utilization

Configure a Remote Logging Service

In this section you learned how to:

- Redirect system log messages to a centralized console server

UNIT ELEVEN

CENTRALIZED AND SECURE STORAGE

Introduction

Topics covered in this unit:

- Access iSCSI storage
- Block-level encryption

You will access centralized storage using iSCSI and secure a file system with block-level encryption.

Accessing iSCSI Storage

iSCSI (Internet SCSI) supports sending SCSI commands from clients (initiators) over IP to SCSI storage devices (targets) on remote servers. An iSCSI Qualified Name is used to identify initiators and targets and follows the format of: **`iqn.yyyy-mm.{reverse domain}:label`**. Network communication by default is cleartext to port 3260/tcp on the iSCSI target.

- iSCSI initiator: a client that needs access to raw SAN storage
- iSCSI target: a remote hard disk presented from an iSCSI server, or "target portal"
- iSCSI target portal: a server that provides targets over the network to an initiator
- IQN: "iSCSI Qualified Name". Each initiator and target needs a unique name to identify it; best practice is to use one likely to be unique on the Internet.

Warning

If you allow two initiators to log in to the same iSCSI target (remote hard disk) at the same time, it is important not to allow both initiators to mount the same file system from the same target at the same time. Unless a cluster file system such as GFS2 is in use, you risk file system corruption.

To access a new target with an iSCSI initiator:

- Install iSCSI initator software: *iscsi-initiator-utils*
- Set initiator's IQN in **`/etc/iscsi/initiatorname.iscsi`**

  (Usually a unique label in a namespace matching a DNS name controlled by the organization. Set randomly when *iscsi-initiator-utils* is installed.)
- Discover iSCSI targets provided by the iSCSI server (target portal)

```
iscsiadm -m discovery -t st -p 192.168.0.254
```

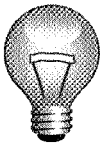
- Log in to one or more iSCSI targets on the server

```
iscsiadm -m node -T iqn.2010-09.com.example:rdisks.demo -p 192.168.0.254 -l
```

- Identify which device is the iSCSI target

  Look at the output of **`dmesg`** or **`tail /var/log/messages`**. Alternately, look at where the iscsi symlinks point with **`ls -l /dev/disk/by-path/*iscsi*`**, or check the status of **`iscsi`**: **`service iscsi status`**.
- At this point the iSCSI disk can be used as if it were a locally-attached hard drive.

Existing file systems can be mounted. If the disk is unformatted it can be partitioned with **fdisk**, and the partitions formatted with a file system or as an LVM physical volume, for example.



Important

When persistently mounting a file system on an iSCSI target in **/etc/fstab**:

1. Use **blkid** to determine the file system UUID and mount using UUID, not **/dev/sd*** device name. (The device name can come up differently from boot to boot depending on the order in which iSCSI devices respond over the network. This can cause the wrong device to be used if mounting by device name.)
2. Use **_netdev** as a mount option in **/etc/fstab**. (This ensures that the client will not attempt to mount the file system until networking is up. Otherwise the system will have errors at boot.)
3. Ensure the **iscsi** and **iscsid** services will start at boot time.

Note

The **iscsi** service logs into iSCSI targets found in **/var/lib/iscsi/nodes** which include **node.startup = automatic**.

The **iscsid** service provides the iSCSI daemon.

To list iSCSI targets:

- Run **service iscsi status**. The output provides the IQN of the iSCSI target, the IP address of the iSCSI target and the local device name:

```
[root@serverX ~]# service iscsi status
iSCSI Transport Class version 2.0-870
version 2.0-872
Target: iqn.2010-09.com.example:rdisks.serverX
        Current Portal: 192.168.0.254:3260,1
        Persistent Portal: 192.168.0.254:3260,1
                **********
                Interface:
                **********
                Iface Name: default
                Iface Transport: tcp
                Iface Initiatorname: iqn.1994-05.com.redhat:82f37447be11
                Iface IPaddress: 192.168.0.X+100
                Iface HWaddress: <empty>
                Iface Netdev: <empty>
                SID: 3
                iSCSI Connection State: LOGGED IN
                iSCSI Session State: LOGGED_IN
```

```
Internal iscsid Session State: NO CHANGE
************************
Negotiated iSCSI params:
************************
HeaderDigest: None
DataDigest: None
MaxRecvDataSegmentLength: 262144
MaxXmitDataSegmentLength: 8192
FirstBurstLength: 65536
MaxBurstLength: 262144
ImmediateData: Yes
InitialR2T: Yes
MaxOutstandingR2T: 1
************************
Attached SCSI devices:
************************
Host Number: 5  State: running
scsi5 Channel 00 Id 0 Lun: 0
scsi5 Channel 00 Id 0 Lun: 1
        Attached scsi disk sdb          State: running
```

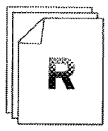
To discontinue use of an iSCSI target:

- Make sure none of the devices provided by the target are in use.
- Make sure all persistent references to use the target are removed from places like **/etc/fstab**.
- Log out of the iSCSI target to temporarily disconnect.

```
iscsiadm -m node -T iqn.2010-09.com.example:rdisks.demo -p 192.168.0.254 -u
```

- Delete the local record of the iSCSI target to persistently disconnect.

```
iscsiadm -m node -T iqn-2010-09.com.example:rdisks.demo -p 192.168.0.254 -o delete
```



References

Red Hat Enterprise Linux Storage Administration Guide

- Chapter 21: Online Storage Management

/usr/share/doc/iscsi-initiator-utils-*/README

Knowledgebase: "Can I put a swap device or file on iSCSI storage?"
https://access.redhat.com/kb/docs/DOC-4135

Practice Performance Checklist

Configuring iSCSI

Configure your serverX to us iSCSI storage existing on instructor.example.com.

Perform the following steps on serverX unless directed otherwise.

- ☐ Verify that the *iscsi-initiator-utils* package is installed, and install it if needed.
- ☐ Discover iSCSI targets on the iSCSI server on 192.168.0.254.
- ☐ Log into the iSCSI target **`iqn.2010-09.com.example:rdisks.serverX`** on 192.168.0.254.
- ☐ Identify the device file for your new iSCSI disk on your initiator.
- ☐ Set up a single partition on your new storage device, format the partition as ext4, and configure it to persistently mount on /mnt/iscsi at boot. *(Note: Do not forget to use* ***_netdev*** *as a mount option, or to mount by file system UUID and not by standard device name.)*
- ☐ Test your configuration.
- ☐ Unmount the new file system and remove or comment the line in **`/etc/fstab`**.
- ☐ Logout and delete the entry for the iSCSI target.

Encrypt Centralized Storage

By default, iSCSI communications between the initiator and target are sent in the clear for maximum performance. This has obvious security risks. iSCSI traffic can be isolated on a separate network or encapsulated in a IPsec or TLS encryption tunnel. This configuration is beyond the scope of this section.

LUKS (Linux Unified Key Setup) can be used to encrypt the contents of block devices. Another approach to improve the security of data on an iSCSI target is to use LUKS to encrypt the contents of the iSCSI target, its partitions, or its logical volumes, and only decrypt on-disk data once it has arrived at the iSCSI initiator.

To use LUKS with iSCSI, encryption must be set up prior to formatting the block device for use. For an existing encrypted block device, it must be opened (decrypted with its password) before its contents are used, and it should be closed when it is not in use. A file, **/etc/crypttab**, contains information needed to automatically prompt for LUKS passwords to open encrypted devices at boot time and to indicate the device name used by the decrypted device, which can then be used in **/etc/fstab**.

Your instructor will ask you to work with a small group in order to investigate and discuss the details of how to set up and use an encrypted file system. For each of the three general tasks, identify the specific subtasks and commands to accomplish them. Use the References and on-line documentation to assist you. The instructor will review the procedure in a few minutes.

Encrypt Centralized Storage Worksheet

Create an encrypted block device

1. Use **fdisk** to partition the disk:

2. For better security, fill the device with random data from **/dev/urandom**:

3. Encrypt the device with **cryptsetup luksFormat**:

Decrypt and persistently mount an encrypted block device

1. Decrypt the device using **cryptsetup luksOpen**:

2. Create a file system on the opened device:

3. Add an entry to **/etc/crypttab**:

4. Add an entry to **/etc/fstab**:

5. Mount the file system to ensure it is persistent:

Unmount and close an encrypted block device

1. Unmount the device:

2. Close the encrypted device using **cryptsetup luksClose**:

References

Red Hat Enterprise Linux Storage Administration Guide
- Chapter 12: Encrypted File System

Red Hat Enterprise Linux Installation Guide
- Appendix C.4: Creating Encrypted Block Devices on the Installed System After Installation

cryptsetup(8) and **crypttab**(5) man pages

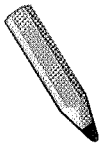
Practice Performance Checklist

Encrypting iSCSI

Modify the serverX iSCSI-attached storage to be encrypted.

Perform the following steps on serverX unless directed otherwise.

- ☐ Rediscover and log in to the iSCSI target **`iqn.2010-09.com.example:rdisks.serverX`** on 192.168.0.254.
- ☐ Create and mount an encrypted block device:
  - Reuse the partition you created in the last section. Encrypt the partition using a passphrase of **`redhat`**
  - Open the new partition with the label **`iscsi-secret`**
  - Format **`/dev/mapper/iscsi-secret`** with an ext4 file system
  - Manually mount **`/dev/mapper/iscsi-secret`** on **`/storage`**.



Test

Criterion Test

Case Study

Centralized Storage

Before you begin...

Run **`lab-setup-centralstore`** on desktopX to prepare serverX for the exercise.

Cold Storage, an appliances retailer, recently acquired new SAN storage that utilizes the iSCSI protocol.

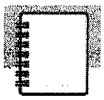
We will begin deployment with your serverX that will be configured to access its own dedicated iSCSI target:

- iSCSI Target IP Address: 192.168.0.254
- iSCSI Target: **`iqn.2010-09.com.example:rdisks.serverX`**

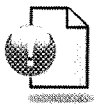
Partition (entire device), format, and persistently mount to **`/coldstorage`**.

When you are ready to check your work, run **`lab-grade-centralstore`** on serverX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Accessing iSCSI Storage

In this section you learned how to:

- Access, format, and mount an iSCSI storage device
- Permanently disconnect from an iSCSI storage device

Encrypt Centralized Storage

In this section you learned how to:

- Protect your company's data using block device encryption

UNIT TWELVE

SSL ENCAPSULATED WEB SERVICES

Introduction

Topics covered in this unit:

- Securing Apache with Encryption
- Customizing a Self-signed Certificate
- Generating a Certificate Signing Request

Securing Apache with Encryption

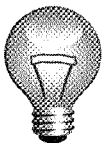
Apache HTTP Server Configuration Review

You should already know the basics of how to set up a simple **Apache HTTP Server**. In this unit, we will take a closer look at some more advanced but common configurations used with web server deployments, starting with how to set up support for TLS/SSL connections.

Apache HTTP Server is installed by the **web-server** yum group, the most critical package being *httpd*, which provides the core components of the web server. The **httpd** service script is used to start and start the server, and it listens for connections on TCP port 80 by default. It is confined by the SELinux policy for added security, which we have discussed and which is documented in the **httpd_selinux**(8) man page. Its main configuration file is **/etc/httpd/conf/httpd.conf**, which automatically includes all the files matching **/etc/httpd/conf.d/*.conf** as part of its file. By default, web content is served from subdirectories of **/var/www**, with the "**DocumentRoot**" of actual web pages in **/var/www/html**, although this can be changed in the configuration file.

Steps to Deploy TLS/SSL Encryption

Support for TLS/SSL web sites is provided by the *mod_ssl* RPM package. Its configuration file is **/etc/httpd/conf.d/ssl.conf**. Simply by installing it and restarting the web server, a SSL encrypted version of the default website on the server will be made available with a self-signed test certificate for localhost.



Important

To contact a TLS/SSL encrypted website on your server using a **https://** URL, you must make sure that clients can connect to TCP port 443 on your web server. Check your firewall settings.

If you use a web browser to connect to the secure web site, you will probably get a warning that the SSL certificate for the web site does not match the hostname of the site, and that the certificate is not signed by a trusted Certificate Authority. To fix this, you will need to obtain an SSL certificate for your web site's hostname signed by a public Certificate Authority, and you will need to make some configuration changes to **/etc/httpd/conf.d/ssl.conf**. The critical directives are **SSLCertificateFile**, which should point to a file in **/etc/pki/tls/certs/** containing your public SSL certificate, and **SSLCertificateKeyFile**, which should point to a file in **/etc/pki/tls/private/** containing your private SSL key. We will not go into further depth in this class on how to obtain a signed SSL certificate.

So, the basic steps are:

1. Make sure the **web-server** yum group is installed:

```
yum groupinstall web-server
```

2. Install the *mod_ssl* package:

```
yum install mod_ssl
```

3. If you are replacing the test certificate with a signed one:
   - Copy your certificate and private key to appropriate places in **/etc/pki/tls/**
   - Make sure both files have the SELinux type **cert_t** and that the private key is not world readable
   - Open **/etc/httpd/conf.d/ssl.conf** in an editor
   - Point **SSLCertificateFile** at your SSL certificate
   - Point **SSLCertificateKeyFile** at your SSL private key
   - Save and exit editing **/etc/httpd/conf.d/ssl.conf**
4. Restart the **httpd** service

X.509 Certificates

SSL certificates deliver a server's public encryption keys and identifying information to network clients. Clients use this information to initiate secure, authenticated communications with the server. The *X.509* protocol defines the following certificate components.

Subject

The identity of the server which presents the certificate, consisting of a Country (C), State (ST), City (L), Organizational Name (O), Organizational Unit (OU), and *Common Name* (CN), amongst others.

The *Common Name* is critically important, and should match the fully qualified hostname of the server presenting the certificate.

Public Key

The primary payload of the certificate. The public key is used by the client negotiate a secure session key with the server, which in turn is used to symmetrically encrypt the session.

Issuer

The identity of the *Certificate Authority* (CA), consisting of components similar to the *Subject*. By cryptographically signing the certificate, the *Issuer* asserts that the *Subject* is the proper owner of the *Public Key*.

In order to verify a certificate, the client must have previously obtained the public key of the *Issuer*, usually distributed as a self-signed certificate known as the *CA Certificate*.

Period of Validity

SSL certificates have an associated period of validity, limiting the window of vulnerability associated with any compromised credentials. In order to verify a certificate, clients generally ensure their local clock falls within the certificates period of validity.

Examining X.509 Certificates

While a variety of tools exist for manipulating X.509 certificates, Red Hat Enterprise Linux utilities commonly use the *openssl* library, accessible through its s front end command, **openssl**.

X.509 certificates are generally distributed as Base64 encoded binary blobs.

```
# cat serverX.crt
-----BEGIN CERTIFICATE-----
MIID7jCCAtagAwIBAgIBATANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMx
FzAVBgNVBAgMDk5vcnRoIENhcm9saW5hMRAwDgYDVQQHDAdSYWxlaWdoMRYwFAYD
...
8ukzyKmwL0HTUb3B/qt/9zvmg4TvPnSUk6CSPQ8KYs/IHKk1eJTGXCD+L+trjLQE
pmE=
-----END CERTIFICATE-----
```

The following **openssl** invocations can be used to extract useful information from X.509 certificates.

- Examine all information:

```
# openssl x509 -in serverX.crt -text
...
        Issuer: C=US, ST=North Carolina, L=Raleigh, O=Red Hat, Inc., OU=Training, CN=GLS
 Example CA Certificate
        Validity
            Not Before: Jan 18 19:12:39 2011 GMT
            Not After : Jan 18 19:12:39 2012 GMT
        Subject: C=US, ST=North Carolina, O=Red Hat, Inc., OU=Training,
 CN=serverX.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c5:5b:06:cd:ac:17:7c:53:09:69:04:11:a6:7e:
                    8e:27:35:57:4a:eb:87:70:c7:f4:15:93:00:9d:81:
...
```

- Examine *Subject* information:

```
# openssl x509 -in serverX.crt -noout -subject
subject= /C=US/ST=North Carolina/O=Red Hat, Inc./OU=Training/CN=serverX.example.com
```

- Examine *Issuer* information:

```
# openssl x509 -in serverX.crt -noout -issuer
issuer= /C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=Training/CN=GLS Example CA
 Certificate
```

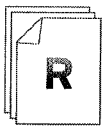
- Examine *Period of Validity* information:

```
# openssl x509 -in serverX.crt -noout -dates
notBefore=Jan 18 19:12:39 2011 GMT
notAfter=Jan 18 19:12:39 2012 GMT
```

Self-signed Certificates

Self-signed certificates are X.509 certificates with identical *Subject* and *Issuer*. While "normal" certificates allow a server to claim "you should trust that this public key belongs to me because this Certificate Authority you trust says so", self-signed certificates claim "you should trust that this public key belongs to me because I say so."

Self-signed certificates are often used on private networks where authenticated identities are not an issue. Also, certificate authorities generally distribute their public keys using self-signed certificates in order to "bootstrap" a trusted certification chain.



References

Red Hat Enterprise Linux Deployment Guide

- Section 11.6: Setting Up an SSL Server

> Apache.org: "Apache TLS/SSL Encryption"
> http://httpd.apache.org/docs/2.2/ssl/
> (if ***httpd-manual*** is installed and **`httpd`** is running):
> http://localhost/manual/ssl/

`x509`(1) man page

Practice Performance Checklist

Apache mod_ssl Basics

Deploy an SSL encapsulated Apache web server on serverX. It should use the default self-signed SSL certificate.

- ☐ Log into serverX as root.
- ☐ Install the Apache web server (*httpd*) package and the *mod_ssl* package, if necessary.
- ☐ Examine the **`/etc/httpd/conf.d/ssl.conf`** configuration file provided by the **`mod_ssl`** package.
  - What is the Apache directive that points to the SSL certificate?
  - What is its value?
- ☐ Use **`openssl`** to display the subject and issuer of the SSL certificate that Apache uses.
- ☐ Restart the **`httpd`** service.
- ☐ Launch Firefox and browse to https://serverX.example.com. When Firefox presents a warning, take further steps to examine the certificate with Firefox.
  - Click the "I Understand the Risks" link.
  - Click the "Add Exceptions..." button, then click "View..." when it becomes active.
  - Browse the information presented in both the "General" and "Details" tabs.
  - Click "Close" when you are finished inspecting the certificate information.

Customizing a Self-signed Certificate

When encrypted communication is important, but authenticated identity is not, administrators can avoid the complexity of interacting with a certificate authority by generating a *self-signed certificate*.

The following steps outline how to use the **genkey** utility, which is distributed in the *crypto-utils* package, to generate a self-signed certificate and it's associated private key. To simplify, **genkey** will create the certificate and it's associated key in the "correct" location (the **/etc/pki/tls** directory). Accordingly, it must be run as a privileged user (root).

Although not an explicit package prerequisite, **genkey** also requires the *mod_ssl* package.

Generate a Self-signed Certificate

The following steps should be executed as the root user.

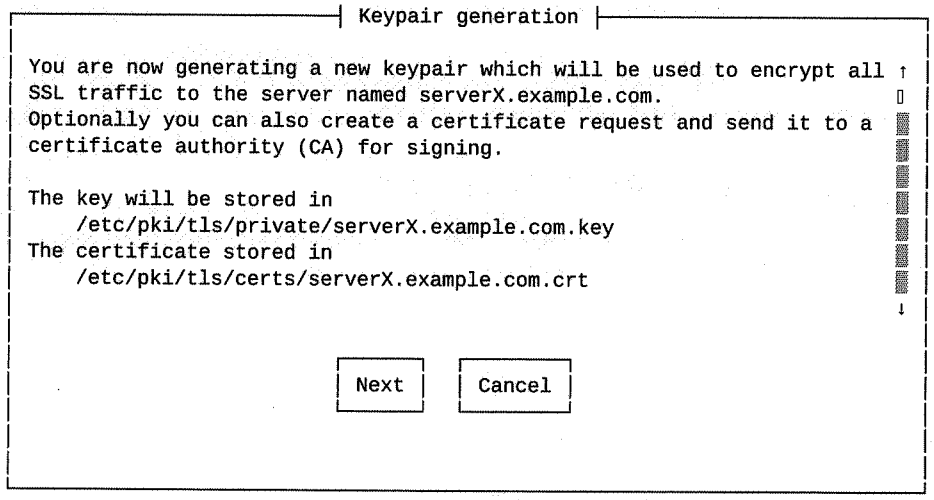
1. Ensure the *crypto-utils* and *mod_ssl* packages are installed.

```
[root@serverX ~]# yum install crypto-utils mod_ssl
```

2. Invoke **gen-utils**, specifying a unique name for the resulting files (for example, the server's fully qualified hostname).

   Use the **--days** command line switch to extend the period of validity to one year.

```
[root@serverX ~]# genkey --days 365 serverX.example.com
```

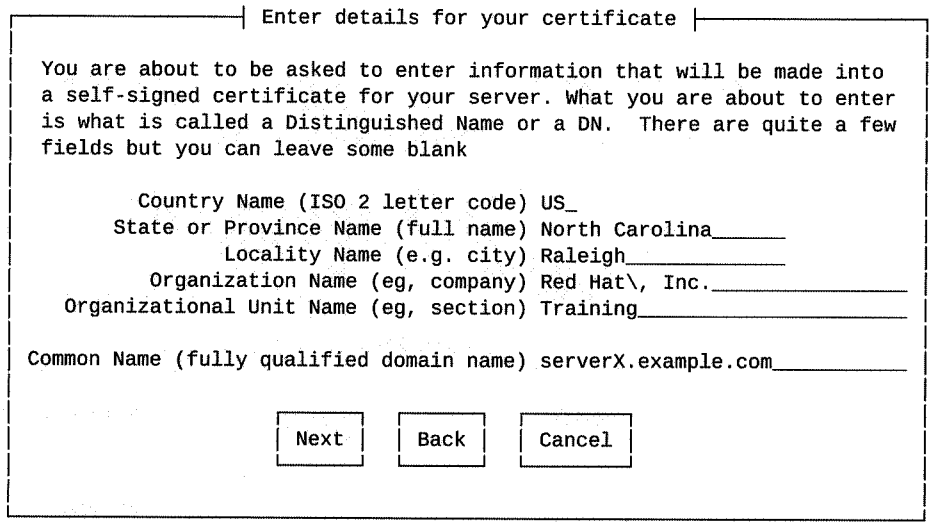


   Note the location of the resulting certificate (**serverX.example.com.crt**) and associated private key (**serverX.example.com.key**).

3. Continue through the dialogs, choosing the appropriate key size. (The default 1024 bit key should be sufficient.)

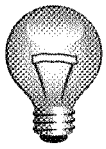
4. Decline to send a Certificate Request (CSR) to a certificate authority (CA).
5. Decline to encrypt your private key.
6. Supply the appropriate identity for your server. The *Common Name* must exactly match the fully qualified hostname of your server.

   (Note that any commas should be escaped with a leading backslash (\).)



7. Optionally, use the **openssl x509** command to confirm the details of your newly generated self-signed certificate.

```
[root@serverX ~]# openssl x509 -text < /etc/pki/tls/certs/serverX.example.com.crt
...
        Issuer: C=US, ST=North Carolina, L=Raleigh, O=Red Hat, Inc., OU=Training,
 CN=serverX.example.com
        Validity
            Not Before: Jan 20 15:26:35 2011 GMT
            Not After : Jan 20 15:26:35 2012 GMT
        Subject: C=US, ST=North Carolina, L=Raleigh, O=Red Hat, Inc., OU=Training,
 CN=serverX.example.com
...
```



Important

When using **genkey** to specify any field in the subject identity of the X.509 certificate, commas must always be escaped with a preceding \ character.

Example: Red Hat\, Inc.

Installing a Certificate and its Private Key

By default, the web server installs with a self-signed certificate bound to *localhost.localdomain*. Usually, administrators would like to replace this certificate with either a self-signed certificate bound to an appropriate external hostname, or a certificate verified by a certificate authority.

As prerequisites, the administrator must have already obtained:

- A X.509 certificate (in this case, **/etc/pki/tls/certs/serverX.example.com.crt**).
- The X.509 certificate's associated private key (in this case, **/etc/pki/tls/private/serverX.example.com.key**).

1. Ensure the *mod_ssl* package is installed.

```
[root@serverX ~]# yum install mod_ssl
```

2. Because the private key is sensitive information, ensure that it can only be read by the root user.

```
[root@serverX ~]# ls -l /etc/pki/tls/private
...
-r--------. 1 root root 937 Jan 20 10:26 serverX.example.com.key
...
```

3. Edit **/etc/httpd/conf.d/ssl.conf**, setting the `SSLCertificateFile` and `SSLCertificateKeyFile` directives to refer to the X.509 certificate and key file, respectively.

```
[root@serverX ~]# grep ^SSLCertificate /etc/httpd/conf.d/ssl.conf
SSLCertificateFile /etc/pki/tls/certs/serverX.example.com.crt
SSLCertificateKeyFile /etc/pki/tls/private/serverX.example.com.key
```

4. Restart the web server.

```
[root@serverX ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

5. To confirm, use a web client (such as **Firefox**) to access the web server using the *https* protocol (https://serverX.example.com).

   The web client will probably warn that it does not recognize the certificate's issuer. This is appropriate for a self-signed certificate. Ask the web client to bypass certificate authentication. (For **Firefox**, select "I Understand the Risks", "Add Exception", and "Confirm Security Exception".)

   Use the appropriate technique to examine the certificate details. (For **Firefox**, you may click on the padlock icon in the lower right corner, and choose "View Certificate" in the resulting dialog.)

Examining SSL Credentials

The **openssl** command provides a minimal SSL client which is useful for troubleshooting SSL related issues.

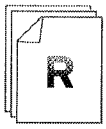
The following script illustrates using the **openssl s_client** command to obtain a SSL certificate from a remote server, and **openssl x509** to display the certificate's details.

```
#!/bin/bash

# usage: check-remote-cert.sh HOST:PORT

SERVER=$1

echo |    # echo is needed to make server hang up
openssl s_client -connect ${SERVER} 2>/dev/null |  # s_client connects and prints cert
openssl x509 -noout -subject -issuer -dates      # x509 reads the cert and prints info
```



References

Red Hat Enterprise Linux Deployment Guide

- Section 11.6: Setting Up an SSL Server

genkey(1) and **s_client**(1) man pages

Practice Performance Checklist

Creating a Custom Self-Signed Certificate

Deploy an SSL encapsulated Apache web server on serverX. It should use a personalized, self-signed SSL certificate.

- ☐ Login as **root** on serverX. Make sure the *crypto-utils* package is installed.
- ☐ Use the **genkey** utility to create a custom self-signed certificate for serverX.example.com that will expire after a year. Note where the key and the matching certificate are created.
- ☐ The certificate should have the following characteristics:
  - The key should be 1024 bits and should *not* be encrypted.
  - country code = *local country*
  - state = *local state*
  - locality = *local city*
  - organization = Red Hat Inc.
  - common name = serverX.example.com
- ☐ Modify the Apache configuration to point to the new key and certificate. Do *not* rename the files just created.
- ☐ Restart the Apache web service.
- ☐ Create the **check-remote-cert.sh** script (as shown previously in your workbook) on your desktopX machine.
- ☐ Make the script executable and run it so that it connects to serverX.example.com port 443.
- ☐ Confirm the certificate subject, issuer, and dates are correct and match what you specified earlier.

Generating a Certificate Signing Request

- Generate a certificate signature request and associated private key
- Install a certificate and associated private key
- Configure a SSL client to verify service with a CA certificate
- Monitor deployed SSL certificates for expiration

Generating a Certificate	Generating a CSR
Generating a Certificate	Generating a CSR
Run **genkey --days *no_of_days* *host_fqdn***	
Skim intro, click [next]	
Specify number of bits, then click [next]	
Generate random bits (via mouse and console keyboard)	
Generate CSR? Select No	
Leave checkbox for passphrase unchecked, click [next]	
Specify certificate details (country, state, locality, organization, org unit, common name (fqdn), email)	

Table 12.1. Compare and Contrast: Generating a Self-signed Certificate vs. a CSR

Note

The CSR file can be found in **/etc/pki/tls/certs/$(hostname).0.csr**

Using a CA Certificate to Verify Services

The following steps are used to obtain a X.509 certificate authenticated by a certificate authority.

1. Use an appropriate utility (such as **genkey**) to generate a certificate signature request (CSR) and associated private key.
2. Optionally, use the **openssl req** command to confirm the details of the CSR.

```
[root@serverX ~]# openssl req -text < /etc/pki/tls/certs/serverX.example.com.0.csr
Certificate Request:
...
```

```
        Subject: C=US, ST=North Carolina, L=Raleigh, O=Red Hat, Inc., OU=Training,
 CN=serverX.example.com
...
```

3. Send your CSR to the appropriate certificate authority, along with any fees and required supporting documentation to verify your identity. Do *not* send your associated private key.

4. Your certificate authority should return to you a X.509 certificate. Save your returned certificate to the appropriate location, conventionally the **/etc/pki/tls/certs** directory. Optionally, use the **openssl x509** command to verify the details of your returned certificate.

```
[root@serverX ~]# openssl x509 -text < /etc/pki/tls/certs/serverX.example.com.crt
...
        Issuer: C=US, ST=North Carolina, L=Raleigh, O=Example, Inc., CN=example.com
 Certificate Authority
        Validity
            Not Before: Jan 20 17:03:27 2011 GMT
            Not After : Jan 20 17:03:27 2012 GMT
        Subject: C=US, ST=North Carolina, L=Raleigh, O=Red Hat, Inc., OU=Training,
 CN=serverX.example.com
...
```

5. In **/etc/httpd/conf.d/ssl.conf**, update the `SSLCertificateFile` and `SSLCertificateKeyFile` directives to refer to your newly installed certificate and private key, respectively, as in the previous section.

6. Restart your web server, and use a web client to confirm the new certificates details, as in the previous section.

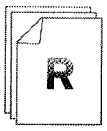
You can use **openssl s_client** to check the signature on an SSL certificate:

```
[root@serverX ~]# echo | openssl s_client -CAfile ${cacert} -connect ${server}:${port} 2>/
dev/null | grep Verify
```

Steps to Add a Non-standard CA to Firefox

1. Launch **Firefox**.
2. Select **Edit** → **Preferences**
3. Click the **Advanced** icon
4. Select the **Encryption** tab
5. Select the **Authorities** tab in the new window
6. Click the Import... button
7. Browse to find the CA's certificate file and **Open** it
8. Check the **Trust to identify web sites** checkbox
9. Click the OK, and then the Close buttons to return to the browser.

The **certwatch** utility (provided by the *crypto-utils* package) e-mails **root@localhost** when an Apache SSL certificate will expire in the next 30 days. It is run in an daily cron job.



References

Red Hat Enterprise Linux Deployment Guide

- Section 11.6: Setting Up an SSL Server

Apache SSL Documentation

http://httpd.apache.org/docs/2.2/ssl
http://localhost/manual/ssl

certwatch(1), **genkey**(1), and **s_client**(1) man pages

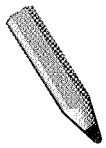
Practice Performance Checklist

Working with CA-Signed Certificates

Deploy an SSL encapsulated Apache web server on serverX. It should use a CA-signed SSL certificate.

- ☐ Use the **genkey** utility to create a new key and a certificate signing request (CSR) for serverX.example.com that will expire after a year.
- ☐ The requested certificate should have the following characteristics:
  - The key should be 1024 bits and should *not* be encrypted.
  - country code = *`local country`*
  - state = *`local state`*
  - locality = *`local city`*
  - organization = Red Hat Inc.
  - common name = serverX.example.com
- ☐ Use the web form at **`http://instructor.example.com/lab/csr_upload.html`** to upload your certificate signature request.
- ☐ Save the returned certificate to a local file. For **Firefox**, this can be accomplished by choosing **File : Save Page As....**

  Move the saved certificate to the appropriate location, and deploy it with your Apache web server.
- ☐ Get a copy of the instructor's CA certificate from **`http://instructor/pub/example-ca.crt`**.
- ☐ Use it with **`openssl`** to verify that the certificate presented by your web server is properly signed.
- ☐ Install the CA certificate in Firefox.
- ☐ Launch Firefox and browse **`https://serverX.example.com`**. What is different?



Test

Criterion Test

Case Study

SSL Encapsulated Web Services

Before you begin...

Run the **lab-setup-hacker** script on desktopX.

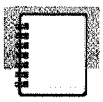
Marcelo Hacker is a successful private investigator. In fact, he is doing so well that it is becoming difficult to find the time to meet with prospective clients. Mr. Hacker has decided to set up a website where prospective clients can send him messages. As confidentiality is an important part of the private investigation business, the website must use a signed SSL certificate.

- Set up Apache on serverX to provide an SSL encrypted website for Marcelo Hacker.
- Your instructor will act as the Certificate Authority that will sign your SSL certificate. Create a certificate request for serverX.example.com and upload it using the form **http://instructor.example.com/lab/csr_upload.html**.

  (Note that the certificate authority will refuse to issue a duplicate certificate, so make sure to vary your identity information from previous labs.)
- Deploy the signed certificate for Apache on serverX.
- Leave the default placeholder website for content. Mr. Hacker will upload his custom content at a later date.

When you have fulfilled the requirements, run **lab-grade-hacker** on desktopX to check your work.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes

Unit Summary

Securing Apache with Encryption

In this section you learned how to:

- Install a certificate for the Apache web server and enable SSL encapsulation
- Examine preexisting SSL certificates
- Manually validate certificate presented by an existing SSL service

Customizing a Self-signed Certificate

Since you've completed this unit, you are able to:

- Generate a self-signed certificate and associated private key
- Manually validate certificate presented by an existing SSL service

UNIT THIRTEEN

WEB SERVER ADDITIONAL CONFIGURATION

Introduction

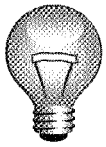
Topics covered in this unit:

- Name-based virtual hosting
- Dynamic CGI content
- Authenticating web users
- SELinux web security

Configure Name-Based Virtual Hosting

Virtual hosts allow you to serve multiple web sites at the same time from a single **httpd** server. In this section we will look at *name-based virtual hosts*, in which multiple hostnames all point to the same IP address, but the web server delivers a different web site with different content depending on the hostname used to reach the site.

The way this will work is that a particular IP address or addresses on the web server will be identified as being shared by the name-based virtual hosts. The web server will look in its configuration files for the first **VirtualHost** block for the IP address of each incoming request in which the **ServerName** or **ServerAlias** matches the hostname used by the request. It will serve content based on the configuration of that matching **VirtualHost** block. If the hostname does not match in any block, then the first **VirtualHost** block for the IP address of the request is used by default.



Important

When you add virtual hosts to your web server configuration, you need to make sure you have a **VirtualHost** block that matches your original main website if you intend to keep serving it out. The directives for the main website are used as *defaults* for the virtual hosts, which may be overridden by each **VirtualHost** block.

The following is a list of key concepts and settings you will need to know in order to set up name-based virtual hosts. Pay attention as the class discusses each one and take notes of where, when, and how you would use each along with any examples that the instructor gives. You will be asked to set up virtual hosts after this discussion.

```
# Name-based virtual hosts defined in /etc/httpd/conf/httpd.conf
NameVirtualHost *:80

<VirtualHost *:80>
    ServerName www.wonka-chocolates.com
    ServerAlias wonka-chocolates.com
    ServerAdmin webmaster@wonka-chocolates.com
    DocumentRoot /var/www/wonka-chocolates.com/html
</VirtualHost>
```

Name-Based Virtual Hosting Keywords

1. VirtualHost

2. NameVirtualHost

3. ServerName/ServerAlias

4. ServerAdmin

5. ServerAdmin

6. DocumentRoot

7. **`semanage fcontext`**

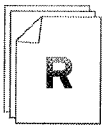
Configuring one **`DocumentRoot`** to be administered by a group:

It is useful to use set-GID directories to make it easier for a group of web administrators to manage content under a **`DocumentRoot`**. Use **`chgrp -R`** ***`webadminsDocumentRoot`*** to set all the files in **`DocumentRoot`** to be owned by group *`webadmins`*, that is, whatever group your webadmins are all in. Then make sure that group has set-GID on **`DocumentRoot`**: **`chmod 2775`** ***`DocumentRoot`***. (A fancier way to set set-GID and write for the group on *all* subdirectories of **`DocumentRoot`** is **`find`** ***`DocumentRoot`*** **`-type d -exec chmod g+ws '{}'\;`**, where *`DocumentRoot`* is replaced with the actual **`DocumentRoot`** directory.)

You should also ensure that the SELinux type on the contents of **`DocumentRoot`** is either **`httpd_content_t`** or **`public_content_t`** to allow the web server to serve the contents out.

Note

There is another type of virtual host, the *IP-based virtual host*, in which each virtual host has its own IP address on the server. These types of virtual hosts work better with TLS/SSL sites; note that the default SSL service configured in **`/etc/httpd/conf.d/ssl.conf`** is an IP-based virtual host. For more information on IP-based virtual hosts, see the **Apache HTTP Server** documentation.



References

Red Hat Enterprise Linux Deployment Guide
- Section 11.5: Setting Up Virtual Hosts

Red Hat Enterprise Linux Deployment Guide
- Section 15.5.1: Group Directories

Apache.org: "Name-Based Virtual Host Support"
http://httpd.apache.org/docs/2.2/vhosts/name-based.html
(if ***httpd-manual*** is installed and **`httpd`** is running)
http://localhost/manual/vhosts/name-based.html

Practice Performance Checklist

Configure Name-Based Virtual Hosts

For this exercise, wwwX.example.com is already set up as a **CNAME** alias to serverX.example.com.

When you finish the checklist, you will run a grading script, so make sure your web server serves up the content exactly as described in the steps.

- ☐ Install and start **httpd** on serverX.
- ☐ Create **/var/www/html/index.html** containing the text "**this is serverX**."
- ☐ From desktopX, use Firefox to verify that the websites wwwX, wwwX.example.com, serverX, and serverX.example.com all display your custom **index.html**.
- ☐ Create **/wwwX/html/index.html** containing the text "**this is wwwX**."
- ☐ Modify Apache to enable name-based virtual hosting. serverX and serverX.example.com should serve **/var/www/html/index.html** as the main page. wwwX and wwwX.example.com should serve **/wwwX/html/index.html** as the main page.
- ☐ Do not disable SELinux (Hint: You may need to modify the SELinux file context database, or change the SELinux type of certain files).
- ☐ When you finish, run the **lab-grade-virthost** evaluation script from serverX to make sure you have done everything correctly.

Stage a CGI executable

CGI, the Common Gateway Interface, is the easiest way to put dynamic content on a web site. The web server acts as an application gateway to a *CGI script* which runs on the server and generates HTML output in its response which the server sends back to the browser.

CGI scripts can be used for many purposes, but it is important to carefully control which CGI scripts are used and who is permitted to add and run them. A poorly written CGI script may provide a way for an external attacker to compromise the security of the web site and its contents. Therefore, both at the web server level and at the SELinux policy level there are settings which are used to restrict use of CGI scripts.

Install a local copy of the How-To by installing the **httpd-manual** RPM with **yum** on serverX. Restart Apache after you install the package to make the Apache documentation available for browsing. Consult the Apache.org "Dynamic Content with CGI" tutorial in the References below to see examples and sample commands that perform the following steps.

CGI Permissions

Given a CGI script to be located at **/wwwX/cgi-bin/hostinfo.cgi**, what configuration syntax, filesystem permissions, and SELinux context type would you need to know?

1. Create a directory outside of the web site's **DocumentRoot**:

```
[root@serverX ~]# mkdir -p /wwwX/cgi-bin
```

2. Configure Apache to recognize it as a source for CGI programs:

3. Change the SELinux context of the CGI script directory to **httpd_sys_script_exec_t**:

4. Restart Apache to put the changes into effect:

Deploy a CGI Script

Make sure you understand how to enable CGI on a directory if someone were to give you a CGI script.

1. Copy the script into the CGI script directory:

2. Ensure the script is not writable by the **httpd** daemon:

3. Make it executable:

Warning

One of the most common sources of security issues that lead to web site compromises are software bugs in web applications. Be careful when writing code for CGI applications!

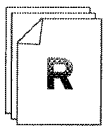
CGI programs must be written in a way they will produce content the Apache web server expects. Writing scripts that produce valid CGI output is the responsibility of web developers and is beyond the scope of this course.

Note

By default, the SELinux policy restricts CGI scripts from working unless labeled with the type **`httpd_sys_script_exec_t`**. Some locations on the system are set up by the SELinux policy to have this type by default. The most common locations used for CGI scripts include:

- **`/var/www/cgi-bin/`**
- **`/var/www/*/cgi-bin/`**
- **`/var/www/html/*/cgi-bin/`**

Others can be added manually.



References

Apache.org: "Apache Tutorial: Dynamic Content with CGI"
http://httpd.apache.org/docs/2.2/howto/cgi.html
(if ***httpd-manual*** is installed and **`httpd`** is running)
http://localhost/manual/howto/cgi.html

Red Hat Enterprise Linux Managing Confined Services
- Chapter 3: The Apache HTTP Server

`httpd_selinux`(8) man page

Practice Quiz

Apache CGI

1. CGI stands for

   (select one of the following...)
   a. Content Generated Interface
   b. Command Gateway Interface
   c. Common Generated Interface
   d. Common Gateway Interface

2. The last argument in **`ScriptAlias /cgi-bin/ /my/private/cgi-bin/`** is

   (select one of the following...)
   a. relative to **`DocumentRoot`**
   b. relative to **`/var/www/`**
   c. relative to **`ServerRoot`**
   d. an absolute path on the filesystem

3. The default **`ScriptAlias`** in **`/etc/httpd/conf/httpd.conf`** is pointing to ...

   (select one of the following...)
   a. /var/www/cgi-bin
   b. /var/html/cgi-bin
   c. /cgi-bin
   d. /var/www/html/cgi-bin

4. One of the built in SELinux context types for a generic CGI programs is

   (select one of the following...)
   a. **`httpd_t`**
   b. **`httpd_sys_script_exec_t`**
   c. **`script_t`**
   d. **`httpd_content_t`**

5. The Apache process should have the following filesystem permissions on CGI programs

   (select one of the following...)
   a. **`---`**
   b. **`r--`**
   c. **`r-x`**
   d. **`rwx`**

Configure User-Based Authentication

It can be useful to limit access to parts of a web site to authorized users only. One way to do this is to configure *user-based authentication* based on usernames and passwords, where the web server prompts the user to login when certain pages are accessed.

There are a number of different ways to implement this. We will look at two in this course: *flat-file authentication* where users are defined in a local password file, and *LDAP authentication*, where users are defined in a remote LDAP directory server.

Your instructor will demonstrate how to set up user-based authentication. Take notes as you will be asked to configure your web server to do the same in lab.

Apache Flat-file User Authentication

In this configuration, user accounts and passwords are stored in a local **.htpasswd** file. For security reasons, this file should not be kept in the **DocumentRoot** of the web site, but should be in some directory the web server does not serve out. The special **htpasswd** command is used to manage users in the **.htpasswd** file.

Example configuration procedure:

- Create an Apache password file with two accounts:

```
[root@serverX]# htpasswd -cm /etc/httpd/.htpasswd bob
[root@serverX]# htpasswd -m /etc/httpd/.htpasswd alice
```

- Assuming that the **VirtualHost** block has been defined previously, add a section such as the following into the **VirtualHost** block:

```
<Directory /var/www/virtual/wwwX/html>
    AuthName "Secret Stuff"
    AuthType  basic
    AuthUserFile /etc/httpd/.htpasswd
    Require valid-user
</Directory>
```

- Test access using a web browser; verify that access works for authenticated users but fails otherwise

Apache LDAP User Authentication

In this configuration, user accounts and passwords are stored in a remote LDAP directory service. The advantage of this configuration is that multiple web servers can use the same directory service to store user accounts and passwords, making it easier to keep them synchronized. In order to set this up, you need to know the location of the LDAP server that contains your account information, whether it uses TLS/SSL, and what LDAP prefix your user entries are under. Your LDAP administrator will need to appropriately manage account information in the directory. For the purposes of this course, we will focus on the web server part of this configuration and ignore how information is managed in the LDAP directory.

Example configuration procedure:

- Download the LDAP certificate, if necessary. In class it can be downloaded from *ftp://instructor.example.com/pub/example-ca.crt*. Copy it to a location that would make it available to the web server, such as **/etc/httpd/**
- Add the following to the **/etc/httpd/conf/httpd.conf** file:

```
LDAPTrustedGlobalCert CA_BASE64 /etc/httpd/example-ca.crt
AuthBasicProvider ldap
```

- Add a **Directory** block inside the **VirtualHost** block just like you did for flat-file authentication above. The **AuthUserFile** line, however, should be replaced with an **AuthLDAPUrl** line pointing to the LDAP directory to search:

```
AuthLDAPUrl "ldap://instructor.example.com/dc=example,dc=com" TLS
```

The stanza should look like the following:

```
LDAPTrustedGlobalCert CA_BASE64 cert-path

<Directory /var/www/html/private>
    AuthName "A very private place"
    AuthType basic
    AuthBasicProvider ldap
    AuthLDAPUrl "ldap://fqdn/prefix" TLS
    Require valid-user
</Directory>
```

where

- *cert-path* is the pathname of CA certificate
- *fqdn* is the fully qualified domain name of the LDAP server
- *prefix* is the LDAP prefix (**dc=example,dc=com** for example)

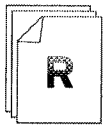
For example:

```
LDAPTrustedGlobalCert CA_BASE64 /etc/httpd/example-ca.crt

<Directory /var/www/html/private>
    AuthName "A very private place"
    AuthType basic
    AuthBasicProvider ldap
    AuthLDAPUrl "ldap://instructor.example.com/dc=example,dc=com" TLS
    Require valid-user
</Directory>
```

- Test that LDAP users can authenticate to Apache.

Use this space for notes



References

Apache.org: "Authentication, Authorization and Access Control"
http://httpd.apache.org/docs/2.2/howto/auth.html
(if the ***httpd-manual*** and **`httpd`** is running)
http://localhost/manual/howto/auth.html

Apache Module mod_authnz_ldap documentation
http://localhost/manual/mod/mod_authnz_ldap.html

Practice Performance Checklist

Configure LDAP-Based Authentication

You will configure the web server on serverX with a **/private** URL that is accessible by users in the LDAP directory on instructor.example.com.

- ☐ Configure LDAP authentication on serverX using instructor.example.com as the LDAP server, **dc=example, dc=com** for the base distinguished name and use the certificate found at *ftp://instructor/pub/example-ca.crt*. Choose LDAP passwords.
- ☐ Login as **root** on serverX. Create a new directory **/var/www/html/private**.
- ☐ In the **private** directory, create an **index.html** containing the text **Private Data**
- ☐ Download **ftp://instructor/pub/example-ca.crt** and place it in **/etc/httpd**
- ☐ Edit **/etc/httpd/conf/httpd.conf** and add LDAP authentication for the **private** directory.

```
LDAPTrustedGlobalCert CA_BASE64 /etc/httpd/example-ca.crt

<Directory /var/www/html/private>
    AuthName "Secret Stuff"
    AuthType  basic
    AuthBasicProvider ldap
    AuthLDAPUrl "ldap://instructor.example.com/dc=example,dc=com" TLS
    Require valid-user
</Directory>
```

- ☐ Restart Apache
- ☐ Browse to http://serverX.example.com/private. You should see an authentication dialog box pop up. If not, close all browser windows, check your configuration, and try again.
- ☐ Log in as user **ldapuserX** with a password of **password**

Troubleshooting Apache SELinux Issues

- List current port SELinux contexts

```
[root@serverX ~]# semanage port -l | grep http
http_cache_port_t              tcp      3128, 8080, 8118, 10001-10010
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

- Assign a port an SELinux context

  If you change Apache to run on a non-standard port, you will likely need to assign that port the **http_port_t** SELinux context. For example, if you changed Apache to run on port 777, you would do the following to allow Apache to run:

```
[root@serverX ~]# semanage port -a -t http_port_t -p tcp 777
```

- Apache log files and logging levels

  You can define the log level and define the log files using **LogLevel**, **ErrorLog** and **CustomLog** in **/etc/httpd/conf/httpd.conf**. The default **LogLevel** is **warn**. By default, the **ErrorLog** is sent to **/var/log/httpd/error_log** and **CustomLog** is sent to **/var/log/httpd/access_log**. These directives can be used to define directives for the main web site, or for virtual hosts.

- Make SELinux more verbose, SELinux log files

  There are rules in the SELinux policy that prevent error messages from being sent to the logs. These rules are called *dontaudit* rules. They prevent the logs from filling with useless messages, but they can prevent you from determining an issue if you are troubleshooting. To disable these **donaudit** rules, run the following command:

```
[root@serverX ~]# semanage dontaudit off
```

Warning

Disabling the **dontaudit** rules will also disable **setroubleshoot-server**. No messages will be sent to **/var/log/messages**, and **sealert** will be disabled. All of the SELinux error messages will be sent to **/var/log/audit/audit.log**.

- Restore or change the SELinux context type of an html file or directory

  If you are storing web data in a new location, use **semanage fcontext** to add new location to the context database and use **restorecon** to set the contexts of the files/directories:

```
[root@serverX ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
```

```
[root@serverX ~]# restorecon -RFvv /virtual/
```

If the content is in a known-good location (e.g., **/var/www/html/**), but you are getting permission denied errors, run the **restorecon** command on the directory as above. The **restorecon** command can take the **-F** option which will change the *customizable* types that **restorecon** normally ignores (**/etc/selinux/targeted/contexts/customizable_types** contains this list of types).

Note

If you get permission denied errors, remember that it could be a Linux permission issue, not an SELinux permission issue. Make sure that the **apache** user or group has at least read access to the files and directories in question.

- SELinux documentation on Apache context types and booleans

  The **httpd_selinux**(8) man page includes descriptions of the most common file contexts and booleans for the web server.

  httpd_sys_content_t is used for any general file/directory for the web server.
  httpd_sys_script_exec_t is used for scripts (e.g., CGI) executed by the web server.
  public_content_t is the context for files that will be shared with other services that are being restricted by SELinux such as FTP, rsync, Samba, etc.

- List Apache SELinux booleans

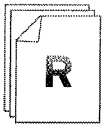
  Use the **getsebool** command to view booleans:

```
[root@serverX ~]# getsebool -a
abrt_anon_write --> off
allow_console_login --> on
allow_corosync_rw_tmpfs --> off
allow_cvs_read_shadow --> off
...
[root@serverX ~]# getsebool httpd_enable_cgi
httpd_enable_cgi --> on
```

- Make SELinux booleans persist

  To make booleans persistent, use **setsebool -P**:

```
[root@serverX ~]# setsebool -P httpd_enable_cgi off
[root@serverX ~]# getsebool httpd_enable_cgi
httpd_enable_cgi --> off
[root@serverX ~]# semanage boolean -l | grep httpd_enable_cgi
httpd_enable_cgi               -> off    Allow httpd cgi support
```



References

Red Hat Enterprise Linux Security-Enhanced Linux
- Section 5.6: Booleans

Red Hat Enterprise Linux Security-Enhanced Linux
- Chapter 8: Troubleshooting

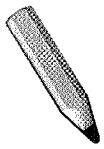
Red Hat Enterprise Linux Managing Confined Services
- Chapter 3: The Apache HTTP Server

semanage(8), **httpd_selinux**(8) man pages

Practice Quiz

Troubleshooting Apache Quiz

1. Complete the following command to list all port contexts: **semanage** _____ **-l**.
2. Complete the following command to enable Apache to use TCP port 8001: **semanage** _____ _____ **-t httpd_port_t** _____ _____ **8001**.
3. The two Apache config file directives to specify the severity (how verbose) error messages are and which file to write to are _____ and _____.
4. The Apache config file directive to specify the format and location of clients accessing content is _____.
5. Full (raw) SELinux AVC messages go to **/var/log**_____.
6. To make SELinux more verbose, you can run **semanage** _____ **off**.
7. The commands to get and set SELinux booleans are _____ and _____.
8. **man** _____ will present an SELinux man page specific to Apache.
9. **man -k** _____ lists all service specific SELinux man pages.
10. The **-F** option to **restorecon** will reset _____ types.



Test

Criterion Test

Case Study

Web Server Additional Configuration

Before you begin...

Run the **`lab-setup-website`** script on desktopX.

Example Industries, a fine example of a company, needs a new website. In fact, they need two! One will be the company website and the other will be for testing content. Additionally, the company website will need a password protected area and a special CGI application installed.

On your serverX machine, deploy a web server with two virtual hosts.

Virtual host 1: *http://serverX.example.com*

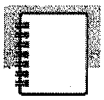
- Create a simple placeholder page for the base URL

Virtual host 2: *http://wwwX.example.com*

- Create a simple placeholder page for the base URL that is different from the one used on virtual host 1
- Make *http://wwwX.example.com/private* a password protected area
- Add user **`forrest`** with password **`trees`** to **`/private`**
- Download the CGI file ftp://instructor.example.com/pub/gls/special.cgi and install it as *http://wwwX.example.com/cgi-bin/special.cgi*

When you are ready to check your work, run the grading script on desktopX: **`lab-grade-website`**

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes

Unit Summary

Configure Name-Based Virtual Hosting

In this section you learned how to:

- Configure a name-based virtual host (separate DocumentRoot, ServerAdmin, ServerName)
- Establish a ServerAlias
- Configure one DocumentRoot to be managed by a group

Stage a CGI executable

In this section you learned how to:

- Stage a CGI executable

Configure User-Based Authentication

In this section you learned how to:

- Limit information to a small group of users using flat-file authentication
- Limit information to a centralized group of users using an LDAP server or relational database

Troubleshooting Apache SELinux Issues

In this section you learned how to:

- Monitor and audit web activity
- Add a persistent SELinux file context mapping to the policy
- Modify SELinux to allow a service to use a non-standard port
- Troubleshoot SELinux context and boolean conflicts

UNIT FOURTEEN

BASIC SMTP CONFIGURATION

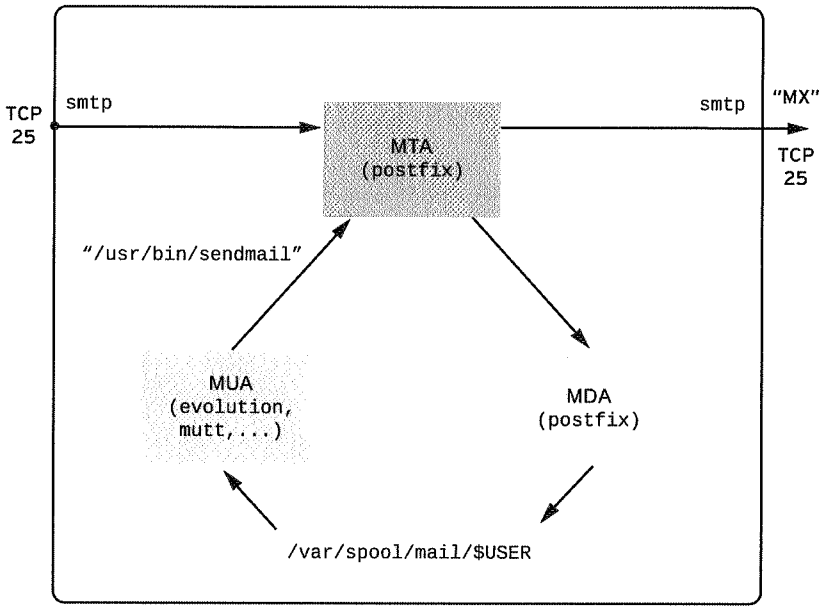
Introduction

Topics covered in this unit:

- Basic e-mail configuration
- Intranet server configuration

Basic E-mail Delivery

E-mail Delivery



- *MTA*: "Mail Transfer Agent". MTAs relay mail from point to point until it can be delivered. E-mail is submitted by other servers using the SMTP protocol to TCP port 25, or by local clients with the **`/usr/bin/sendmail`** program. If the MTA is the final destination, the message is passed to the MDA. If not, it looks up the next MTA in DNS using MX records and relays it there using SMTP.
- *MDA*: "Mail Delivery Agent". The MDA delivers mail to the recipient's local message store (by default, **`/var/spool/mail/`*`user`***). Postfix provides its own MDA to deliver to the default local file-based message store, **`/usr/libexec/postfix/local`**.
- *MUA*: "Mail User Agent". Clients used to send e-mail and read e-mail in the user's message store.

Key e-mail delivery concepts:

- *Relaying*: when an e-mail server (MTA) forwards submitted mail to another server for delivery
- *Queueing*: a failed delivery or relay attempt is queued and retried periodically by the MTA. (By default Postfix does this once every hour.)
- *Rejected*: when an e-mail message is refused by an e-mail server during the initial submission
- *Bounced*: when an e-mail message is returned by a remote server to the originating e-mail server and/or user after it has been accepted for delivery by the remote server

The Postfix MTA

A number of open source e-mail servers exist, including Postfix, Sendmail, and Exim. In this unit, we will focus on Postfix, a powerful but relatively easy to configure MTA, which happens to be used by default in Red Hat Enterprise Linux 6.

Note

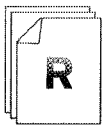
Sendmail was the default MTA in Red Hat Enterprise Linux 5 and earlier.

Postfix is provided by the *postfix* RPM package, and is controlled by the **postfix** service script. It is a modular program made up of several cooperating programs, its components controlled by the **master** process.

The main configuration file for Postfix is **/etc/postfix/main.cf**, which may be edited using a text editor or with the **postconf** command. The **postconf** command may also be used to determine current and default configuration settings, either for all of Postfix or on an option-by-option basis.

By default, Postfix only listens for incoming e-mail from localhost. To reconfigure Postfix to receive mail for local delivery sent from remote hosts, **inet_interfaces = all** must be set in **/etc/postfix/main.cf**.

When troubleshooting e-mail, a log of all mail-related operations is kept in **/var/log/maillog**, which includes information about rejections and successful deliveries. The **mailq** command (or **postqueue -p**) displays a list of any outgoing mail messages that have been queued. To attempt to deliver all queued messages again immediately, you can run the **postfix flush** command (or **postqueue -f**); otherwise Postfix will attempt to resend them about once an hour until they are accepted or expire.



References

Red Hat Enterprise Linux Deployment Guide

- Section 12.3.1: Postfix

postconf(5) and **postfix**(1) man pages

Workshop

Basic E-mail Delivery

Follow along with the instructor as you complete the steps below.

1. Start two terminal shells, one as **root** on desktopX, the other as **root** on serverX.

2. On each machine, confirm that the *postfix* and *mutt* packages are installed and that the *postfix* service is running.

```
[root@desktopX ~]# yum install -y postfix mutt
[root@desktopX ~]# service postfix status

[root@serverX ~]# yum install -y postfix mutt
[root@serverX ~]# service postfix status
```

3. On serverX, add the user **elvis**. Become **elvis** and open the **mutt** MUA to monitor incoming mail.

```
[root@serverX ~]# useradd elvis
[root@serverX ~]# yum install -y mutt
[root@serverX ~]# su - elvis
[elvis@serverX ~]$ mutt
```

4. Use the **mutt** MUA on desktopX to compose and send mail to elvis@serverX.example.com.

   Press **m** in **mutt** to compose a new message. Enter elvis@serverX.example.com in the **To:** field. Enter **Test** as the subject. Enter a few words in the body and save and exit (it opened the file in **vi**). Press **y** to send the mail.

5. Did **elvis** receive e-mail on serverX? Hmm...

6. On desktopX use **mailq** to examine the delivery queue. Also browse **/var/log/maillog** to look for any problems.

```
[root@desktopX ~]# mailq
[root@desktopX ~]# tail /var/log/maillog
```

7. On serverX, recall that **postfix** binds to **localhost** only by default according to Red Hat policy. Use **netstat** to confirm this is the case.

```
[root@serverX ~]# netstat -tuln | grep :25
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
```

8. Examine the current settings of the **inet_interfaces** directive in the main **postfix** configuration file **/etc/postfix/main.cf**.

```
[root@serverX ~]# grep inet_interfaces /etc/postfix/main.cf
inet_interfaces = localhost
```

9. Edit **/etc/postfix/main.cf**. Find the line that has **inet_interfaces = localhost** and change it to **inet_interfaces = all**. Restart the **postfix** service and confirm the daemon is listening on all interfaces.

```
[root@serverX ~]# service postfix restart
[root@serverX ~]# netstat -tuln | grep :25
tcp        0      0 0.0.0.0:25                  0.0.0.0:*                   LISTEN
tcp        0      0 :::25                       :::*                        LISTEN
```

10. On desktopX use **postfix flush** to manually flush the pending delivery queue. Confirm that the queue is now empty and that the mail was successfully delivered.

```
[root@desktopX ~]# postfix flush
[root@desktopX ~]# mailq
[root@desktopX ~]# tail /var/log/maillog
```

11. On serverX use the **mutt** MUA to confirm e-mail delivery. Also examine **/var/log/maillog** for evidence the mail was successfully delivered.

```
[elvis@serverX ~]$ mutt
[root@serverX ~]# tail /var/log/maillog
```

Intranet Configuration

In practice, most organizations do not have one mail server that handles all inbound and outbound e-mail any more. Instead, mail servers are specialized for particular roles for security reasons and so that they may better tune performance for their particular intended applications.

Some standard roles include:

- *null client*: A client machine that runs a local MTA, but only so that all e-mail can be forwarded to a central mail server for delivery. A null client does not accept local delivery for *any* e-mail messages. Users may run MUAs on the null client to read and send e-mail messages. Most machines will be null clients.

  In the diagram below, we use desktop*X*.example.com to stand in for all null clients.

- *inbound-only mail server*: A mail server that handles all incoming e-mail for users at the site and hands it to an MDA for delivery to user message stores. Outbound mail is forwarded to a central mail server in the same way as a null client. The inbound mail server may be integrated with an IMAP or POP3 server to allow user MUAs to access their message stores, or a separate host with access to the message store may run the IMAP or POP3 server.

  In a real-world scenario, there is usually an anti-spam mail server or appliance in front of the inbound-only mail server that filters out spam and only relays good messages to the inbound mail server. We will not discuss configuration of this device in this course.

  In the diagram below, mail.example.com is the inbound-only mail server, and it also runs the IMAP server.

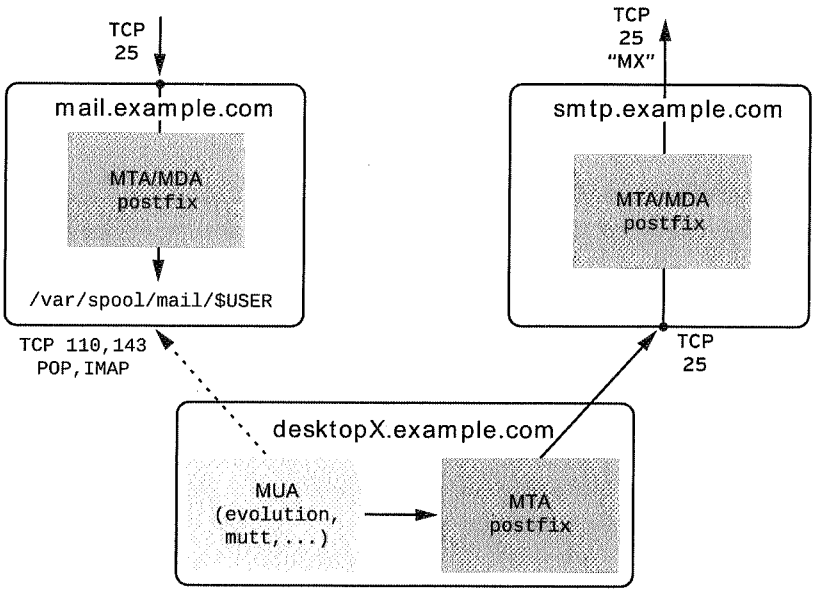
- *outbound mail relay*: The outbound mail relay, or "smarthost", accepts all outbound messages, and using MX records and the SMTP protocol relays them toward their destination. An outbound mail relay must only relay e-mail for authorized hosts; an "open mail relay" will be exploited by spammers, and other mail servers will likely block messages from a known open relay.

  For simplicity, in this class we will look at an internal workgroup outbound relay, which will accept messages sent to port 25/TCP from internal IP addresses for any destination without further authentication, but which does not accept messages from external IP addresses.

  In the diagram below, smtp.example.com is the outbound mail relay used by the null clients.

Note

A more sophisticated variation on the outbound mail relay is the *mail submission agent*. An MSA relays messages from internal or external machines which have been authenticated by username and password over an SSL-protected SMTP connection to port 587/TCP. This allows users not on the local network to relay e-mail securely through the outbound mail relay. It is more complex to configure, and we will not discuss it further in this class.



Important Postfix Configuration Directives

These are all found in the **/etc/postfix/main.cf** file.

myorigin

Rewrite locally posted e-mail to appear to come from this domain. This helps ensure responses return to the inbound mail server.

Default: **myorigin = $myhostname**

inet_interfaces

Controls which network interfaces Postfix listens on for incoming e-mail. If set to **loopback-only** listens only on 127.0.0.1 and ::1, and if set to **all** listens on all network interfaces. Can also specify particular addresses.

Default: **inet_interfaces = localhost**

mydestination

E-mail addressed to these domains are passed to the MDA for local delivery.

Default: **mydestination = $myhostname, localhost.$mydomain, localhost**

mynetworks

Comma-separated list of IP addresses and networks (in CIDR notation) that can relay through this MTA to anywhere, without further authentication.

Default: **mynetworks = 127.0.0.0/8**

relayhost
The smarthost to relay all outbound mail through. Normally given in square brackets to suppress MX record lookup.

Default: **relayhost =**

local_transport
How messages addressed to **$mydestination** should be delivered. By default, set to **local:$myhostname**, which uses the **local** MDA to deliver incoming e-mail to the local message store in **/var/spool/mail**.

Default: **local_transport = local:$myhostname**

These directives can be viewed and set with the **postconf** command:

```
[root@serverX ~]# postconf inet_interfaces
inet_interfaces = localhost
[root@serverX ~]# postconf -e "inet_interfaces = all"
[root@serverX ~]# postconf inet_interfaces
inet_interfaces = all
```

postconf -n shows changes that differ from the default. **postconf -v** shows all values, including those that differ from the default.

External DNS

It is expected that an MX record in DNS for example.com must point to the mailbox server, mail.example.com:

```
example.com.        IN MX 10    mail.example.com.
```

Concept	Directive	*mail.example.com*	*desktop.example.com*	*smtp.example.com*
Binding Interface	*inet_interfaces*			
Masquerade as	*myorigin*			
Indirect Delivery	*relayhost*			
Receive mail for ...	*mydestination*			
Local Delivery	*local_transport*			

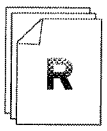
Concept	Directive	*mail.example.com*	*desktop.example.com*	*smtp.example.com*
Relay from ...	*mynetworks*			

Table14.1. Intranet Mail Server, Null Client, and Outbound Relay

A standards-compliant smart host must handle mail sent to postmaster@[*IP address*]; it is also good for it to handle abuse@[*IP address*]. These handle mail bounces from remote servers and reports of spam. The **smtp.example.com** machine should also include:

- **local_recipient_maps =** (empty) to disable accepting mail for most users.
- **virtual_alias_maps = hash:/etc/postfix/virtual** to specify a file in which we will say where to forward e-mail. The **/etc/postfix/virtual** file should contain:

```
postmaster    postmaster@example.com
abuse             abuse@example.com
```



References

postconf(5), **postconf**(1), and **transport**(5) man pages

/usr/share/doc/postfix-*/README_FILES/BASIC_CONFIGURATION_README

/usr/share/doc/postfix-*/README_FILES/STANDARD_CONFIGURATION_README

Practice Case Study

Intranet Configuration

Before you begin...

DNS has already been configured to overlay your hosts as members of the **`domainX.example.com`** domain.

Hostname	IP address	Also known as
mail.domainX.example.com	192.168.0.X+100	(serverX.example.com)
smtp.domainX.example.com	192.168.0.X+200	(hostX.example.com)
desktop.domainX.example.com	192.168.0.X	(desktopX.example.com)

Table 14.2. domainX.example.com

Also, the host mail.domainX.example.com is the **MX** recipient for the entire domainX.example.com domain.

Complete the following table with the appropriate directives to configure these hosts to act as an intranet mailbox server, smtp host, and client station, respectively.

Try to use only the the **`BASIC_CONFIGURATION_README`**, **`STANDARD_CONFIGURATION_README`** and **`main.cf`** files for reference.

Once complete, have another student review your work.

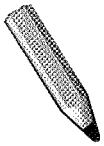
External DNS:

```
domainX.example.com.      IN MX 10    mail.domainX.example.com.
```

Concept	Directive	*mail.domainX*	*desktop.domainX*	*smtp.domainX*
Binding Interface	*inet_interfaces*			
Masquerade as	*myorigin*			
Indirect Delivery	*relayhost*			
Receive mail for ...	*mydestination*			
Local Delivery	*local_transport*			
Relay from ...	*mynetworks*			

Table 14.3. Intranet Mail Configuration for domainX.example.com

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Test

Criterion Test

Case Study

Intranet E-mail Configuration

Before you begin...

Run the script **lab-setup-email** on desktopX

The Hoffman Hair Supply company, a manufacturer of hair grooming products, wants to centralize the management of their internal e-mail.

DNS has already been configured so that your machines are members of the DNS domain **domainX.example.com** with the following addresses:

```
192.168.0.X      desktop.domainX.example.com   (a.k.a.  desktopX.example.com)
192.168.0.X+100    mail.domainX.example.com    (a.k.a.   serverX.example.com)
192.168.0.X+200    smtp.domainX.example.com    (a.k.a. hostX.example.com)
```

Also, the **mail.domainX.example.com** server is the **MX** recipient for the entire **domainX.example.com** domain.

Configure the **mail.domainX.example.com** host to act as an incoming mail-only server, so that all mail delivered to the **@domainX.example.com** domain is stored on this server.

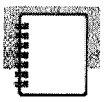
Configure the **smtp.domainX.example.com** server to act as an outgoing SMTP server, which is willing to relay mail from members of the **domainX.example.com** domain to outside networks.

Configure the **desktop.domainX.example.com** host to act as a "null client". It cannot receive e-mail from the network, local mail delivery is disabled, and all outgoing e-mail is sent indirectly via **smtp.domainX.example.com**.

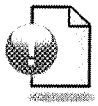
For all three hosts, make sure that any originating mail masquerades the sender's domain as **domainX.example.com**.

When you are ready to check your work, run **lab-grade-email** on desktopX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Basic E-mail Delivery

In this section you learned how to:

- Configure postfix to receive mail from the network for system administrative tasks.
- Monitor e-mail delivery, and diagnose potential delivery problems.

Intranet Configuration

In this section you learned how to:

- Determine if a mailbox server and/or an SMTP server is appropriate.
- Masquerade the domain of sender's addresses.
- Relay all mail to domain mail server.
- Disable local delivery.
- Relay mail only for local network.

UNIT FIFTEEN

CACHING-ONLY DNS SERVER

Introduction

Topics covered in this unit:

- Non-authoritative DNS servers

DNS Overview

In this section the class will review what you already know about DNS and ensure that you understand the types of DNS servers, DNS resource records, and the basics of DNS operation. Ask questions and take additional notes here as there will be a game after the class discussion.

Authoritative nameservers

Stores and serves actual data for a zone (all or part of a DNS domain). Types of authoritative nameservers include:

- *Master*: contains original zone data. Sometimes called a "primary" nameserver.
- *Slave*: backup server which gets copies of zone data from the master through *zone transfers*. Sometimes called a "secondary" nameserver.

Non-authoritative / recursive nameservers

Used by clients to look up data from authoritative name servers. Types of recursive nameservers include:

- *Caching-only nameserver*: only used for lookups, not authoritative for anything but trivial data

DNS Lookups

- *Stub resolver* on client sends query to the nameserver in **`/etc/resolv.conf`**
- If the nameserver is *authoritative* for the information requested, sends *authoritative answer* to client
- Otherwise, if the nameserver has the information requested in its cache, it sents a *non-authoritative* answer to the client
- If information is not in the cache, the nameserver searches for the authoritative nameserver for the information, starting with the root zone and working down the DNS hierarchy to the nameserver which is authoritative for the information, and gets the answer for the client. In this case the nameserver passes the information to the client and also keeps a copy in its own cache for future lookups.

Use this space for notes

DNS Resource Records

A DNS zone stores its information in the form of *resource records*. Each resource record has a *type*, which indicates what kind of data it holds:

- A: name to IPv4 address
- AAAA: name to IPv6 address
- CNAME: name to "canonical name" (another name with an A/AAAA record)
- PTR: IPv4/IPv6 address to name
- MX: mail exchanger for a name (where to send its e-mail)
- NS: name server for a domain name
- SOA: "start of authority", information for a DNS zone (administrative information)

Troubleshooting DNS lookups

The **dig** utility is one of the most useful utilities to troubleshoot issues with DNS lookups.

```
[student@desktopX ~]$ dig example.com
; <<>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41645
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;server3.example.com.           IN      A

;; ANSWER SECTION:
server3.example.com.    86400   IN      A       192.168.0.103

;; AUTHORITY SECTION:
example.com.            86400   IN      NS      instructor.example.com.

;; ADDITIONAL SECTION:
instructor.example.com. 86400   IN      A       192.168.0.254

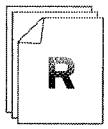
;; Query time: 2 msec
;; SERVER: 192.168.0.254#53(192.168.0.254)
;; WHEN: Mon Dec 13 10:06:48 2010
;; MSG SIZE  rcvd: 94
```

It shows detailed information from a DNS lookup, including why a query may have failed:

- **NOERROR**: query was successful
- **NXDOMAIN**: DNS server says no such name exists
- **SERVFAIL**: DNS server is down *or* DNSSEC validation of response failed
- **REFUSED**: DNS server refuses to answer (perhaps for access control reasons)

Parts of **dig** output:

- The header indicates information about the query and answer, including the response status and any special flags that are set (**aa** for authoritative answer, and so on)
- **QUESTION**: the actual DNS query made
- **ANSWER**: the response, if any
- **AUTHORITY**: the nameservers responsible for the domain/zone
- **ADDITIONAL**: additional information provided, usually about the nameservers
- The comments at the bottom indicate which recursive nameserver was sent the query and how long it took to get a response



References

Red Hat Enterprise Linux Deployment Guide
- Section 10.1: Introduction to DNS

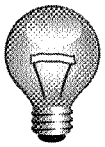
BIND Administrator's Reference Manual, Chapter 1
/usr/share/doc/bind-9.7.0/arm/

host(1), **dig**(1), and **resolv.conf**(5) man pages

Caching-only DNS Servers

In this section, the class will learn how to configure BIND 9.7 as shipped in Red Hat Enterprise Linux 6 as a caching-only nameserver. Highlight and take additional notes here on how to configure BIND as we will complete the game after discussion.

BIND is the most widely-used open source nameserver. In Red Hat Enterprise Linux it is provided by the *bind* software package. However, the main service it provides is the **named** program, which is controlled by the **named** service script.



Important

For BIND to work correctly, the firewall must allow connections to ports 53/UDP and 53/TCP on the nameserver. If 53/TCP is blocked, then large queries and zone transfers will probably break. This is a very common misconfiguration.

The main configuration file for BIND is **`/etc/named.conf`**. The **`/var/named`** directory contains additional data files used by the nameserver.

Syntax of **`/etc/named.conf`**:

- `//` or `#` to the end of a line is a comment; text between `/*` and `*/` is also a comment which can span multiple lines
- Directives end with a semi-colon (`;`)
- Many directives take *address match lists* as in curly braces; a list of IP addresses or subnets in CIDR notation, or named ACLs such as **`any;`** (all hosts) and **`none;`** (no hosts)
- The file starts off with an **`options`** block that contains directives that control how **named** works
- **`zone`** blocks control how **named** finds the root nameserver and the zones for which it is authoritative

Some important **`options`** directives:

- **`listen-on`** controls which IPv4 addresses **named** listens on
- **`listen-on-v6`** controls which IPv6 addresses **named** listens on
- **`allow-query`** controls which clients which clients can ask the DNS server for information
- **`forwarders`** contains a list of nameservers to which DNS queries will be forwarded (instead of directly contacting external nameservers; useful in firewalled scenarios)

All of these directives take semi-colon separated elements in curly braces as an address match list: **`listen-on { any; };`** or **`allow-query { 127.0.0.1; 10.0.0.0/8; };`** for example.

Configure a Caching-only Name Server Demonstration

- Install the *bind* software package:

```
[root@serverX]# yum install bind
```

- Edit **/etc/named.conf**:

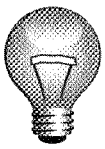
```
listen-on port 53 { any; };
listen-on-v6 port 53 { any; };
allow-query { 192.168.0.0/24; };
forwarders { 192.168.0.254; };
```

- Start and enable the DNS server:

```
[root@serverX]# service named start
[root@serverX]# chkconfig named on
```

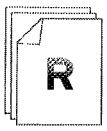
Note

The default configuration of BIND in Red Hat Enterprise Linux 6 integrates the settings formerly provided by the *caching-nameserver* package in Red Hat Enterprise Linux 5.



Important

BIND 9.7 in Red Hat Enterprise Linux 6 automatically attempts to validate DNS responses using DNSSEC by default. Since the root zone is signed, in a classroom without Internet access we have found that BIND may refuse to accept DNS responses from the classroom server because the real root nameservers cannot be contacted, causing DNSSEC validation errors. A workaround for this is to disable automatic DNSSEC validation by setting the option **dnssec-validation** to **no**.

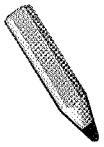


References

Red Hat Enterprise Linux Deployment Guide

- Unit 10: The BIND DNS Server

BIND Administrator's Reference Manual
/usr/share/doc/bind-9.7.0/arm/



Test

Criterion Test

Case Study

Caching-Only DNS Server

Before you begin...

Run **lab-setup-cachingdns** on desktopX to prepare serverX for the exercise.

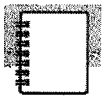
For his growing import/export business, Mr. Hnath would like to improve name resolution performance by deploying a caching name server at each of his business locations.

Recursive queries should be forwarded to the main name server at Hnath Import/Export headquarters.

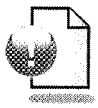
- Set up a caching name server on serverX.
- Configure the name server so that recursive queries are sent to instructor.example.com. Also, configure the name server to accept queries from anyone on the classroom network.

When you are ready to check your work, run **lab-grade-cachingdns** on desktopX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

DNS Overview

In this section you learned how to:

- Use **dig** to verify the functionality of your DNS server

Caching-only DNS Servers

In this section you learned how to:

- Install a caching nameserver
- Forward DNS requests to instructor.example.com

UNIT SIXTEEN

FILE SHARING WITH NFS

Introduction

Topics covered in this unit:

- NFS server configuration
- NFS client considerations

NFS Concepts and Configuration

NFS, the Network File System, is a network file system commonly used by Unix systems and network-attached storage filers to allow multiple clients to share access to files over the network. It may be used to provide access to shared directories of binaries, or to allow users to access their files from different clients in the same work group.

The NFS protocol comes in a number of versions: Linux supports versions 4, 3, and 2, with most system administrators being familiar with NFSv3. The protocol is insecure by default, but newer versions such as NFSv4 provide support for more secure authentication and even encryption with Kerberos. Fill in the table below as the instructor discusses some of the enhancements in NFSv4:

NFSv2	NFSv3	NFSv4
Original public NFS protocol	Extended NFSv2 architecture	
Still in use	Added features: TCP support 64-bit file sizes and offsets Larger read/write sizes	
	Some implementations (including Red Hat Enterprise Linux) support Kerberos	
Requires support services: nfsd, rpc.mountd, rpc.statd, lockd	Also requires support services: nfsd, rpc.mountd, rpc.statd, lockd	
More difficult to secure behind a firewall	More difficult to secure behind a firewall	
Useful for backward compatibility	Useful for backward compatibility	

NFS Server Configuration

To configure a basic NFS server, you should have the **nfs-file-server** package group (which includes the *nfs-utils* package) installed. You then should edit **/etc/exports** to list the file systems you intend to share to client systems over the network, and indicate which clients have what access to the export. For example:

```
/var/ftp/pub        192.168.0.0/24(ro,sync)
```

exports the directory **/var/ftp/pub** to all hosts on the 192.168.0.0/24 network with read-only permission. Likewise

```
/export/homes       *.example.com(rw,sync)
```

exports the directory **/export/homes** with read-write permission to all hosts in example.com. Each export is specified on its own line in the server's **/etc/exports** file.

Any time you make an edit to **/etc/exports** when the NFS server is running, you should ensure these changes get applied by executing **exportfs -r** after saving your changes. You can use **exportfs -v** to display all exports.

NFSv4 also exports something called the *pseudo-root*, the root of all exported file systems. If a client mounts *nfs-server*:/ this automatically mounts all exported file systems relative to their position under / on the NFS server. This is useful for browsing all the file systems exported from a server on a client. You can still mount file systems individually as well.

Note

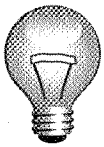
The **fsid=0** export option used in Red Hat Enterprise Linux 5 to manually export an NFSv4 pseudo-root is no longer necessary. The new NFS server automatically exports a pseudo-root without additional configuration or the need to configure bind mounts. (If for some reason you do specify the option, only the export with it set will be used as the top level of the pseudo-root.)

For file ownership and permissions to work properly, the user and group names that use NFS need to exist and be mapped consistently to the same UID and GID numbers on the clients and on the NFS server. These users can be manually configured in **/etc/passwd** and **/etc/group** using local tools, or you can keep them coordinated through a central LDAP authentication and user information directory.

By default, **root** on an NFS client is treated as user **nfsnobody** by the NFS server. That is, if **root** attempts to access a file on a mounted export, the server will treat it as an access by user **nfsnobody** instead. This is a security measure that can be problematic in scenarios where the NFS export is being used as / by a diskless client and **root** needs to be treated as **root**. To disable this protection, the server needs to add **no_root_squash** to the list of options set for the export in **/etc/exports**:

```
/exports/root-192.168.0.1      192.168.0.1(rw,no_root_squash)
```

Note that this particular configuration is not secure, and would be better done in conjunction with Kerberos authentication and integrity checking (which is beyond the scope of this course).



Important

If user names have different UID numbers on different clients or the server, you will have ownership and permission problems.

One symptom of getting this wrong is if a file appears to be owned and usable by a particular non-privileged user, yet when they try to read or write it permission errors occur. The user probably has different UID numbers on the client and the server; using **ls -ln** on the file to show the owner's UID number and **id** to find the user's local UID number can help diagnose this issue. Another symptom of username/UID inconsistencies seen when using NFSv3 or older is that a user creates a file on a mounted NFS file system, but it shows up as being owned by a different user. Note that **root** is purposely mapped to a non-privileged user (**nfsnobody**) by default.

For NFSv4, port 2049/TCP (for **nfsd**) must be open on the server. For NFSv3 and earlier, additional ports must be opened for **rpcbind**, **rpc.mountd**, **lockd**, and **rpc.rquotad**, which is complicated by the fact that many of these services start on "randomly" selected ports. In addition, NFSv2 and NFSv3 support UDP transport, which requires appropriate ports to be opened as well. For simplicity of configuration, in this course we will focus on NFSv4.

Note

The **rpcbind** service replaces **portmap** from Red Hat Enterprise Linux 5.

NFSv4 Demonstration

- Create a user on two machines with a common UID.
- Create a directory to share with NFS and set the proper permissions.
- Edit **/etc/exports**. For example:

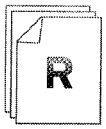
```
/exports/read   192.168.0.0/24(ro,sync)
/exports/write  192.168.0.0/24(rw,sync) 127.0.0.1(rw,sync)
```

- Configure the **nfs** service.

```
[root@serverX]# service nfs start
[root@serverX]# chkconfig nfs on
```

- Mount the NFS pseudo-root share from a client:

```
[root@desktopX]# mount -t nfs serverX.example.com:/ /mnt
```



References

Red Hat Enterprise Linux Storage Administration Guide

- Unit 10: Network File System (NFS)

exports(5) and **exportfs**(8) man pages

Links to NFSv4 Specifications and Resources

http://www.citi.umich.edu/projects/nfsv4/

Practice Quiz

NFS Concepts

1. Under what circumstances should NFSv2 or NFSv3 be used?

2. What is the syntax of the /etc/exports file?

3. What steps should be taken to publish a new export on an existing NFSv4 server?

4. Which option tells NFS to allow the root user on client systems to have root privileges in the share as well?

Using NFS

With NFSv4, you can mount the NFS server's pseudo-root export to see what file systems are being exported. If the server supports NFSv3 and earlier, use **showmount -e *nfsserver*** to talk to **rpc.mountd** and determine what exports are available to which machines.

The **nfs** file system type is used when mounting NFS exports on a client. In Red Hat Enterprise Linux 6, it will try NFSv4 first if supported, then fail back to NFSv3, then NFSv2. To determine the version of NFS in use for a mounted NFS file system, run **mount** with no options or arguments and look for the value of the **vers=** option in the file system's line in the resulting output.

To mount an NFS file system on a client:

- Create an empty directory for the mount point if it does not already exist
- To temporarily mount: **mount -t nfs *nfsserver:/export/mount-point***
- To mount immediately at boot, add an appropriate line to **/etc/fstab**:

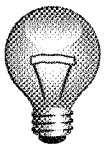
```
nfsserver:/exports    /mount-point    nfs    defaults    0 0
```

Note

NFS entries in **/etc/fstab** are mounted by the **netfs** service script after networking starts. It is more robust to mount NFS exports on demand with the automounter rather than by using entries in **/etc/fstab** to avoid issues when the client boots if the NFS server is offline or networking is not available.

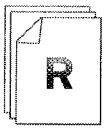
Client-side NFS mount options

- `rw`: mount the file system read-writable
- `ro`: mount the file system read-only
- `vers=4`: try to mount using the NFS version specified only. If this version is not supported by the server the mount request fails
- `soft`: If an NFS request times out, return an error after three retries. Trades off data integrity concerns for increased client responsiveness. (The default behavior is **hard** which will retry indefinitely).



Important

The **intr** mount option is no longer honored in Red Hat Enterprise Linux 6 (by the 2.6.25 Linux kernel and later). Only **SIGKILL** (**kill -9**) can interrupt a pending NFS operation on newer kernels.

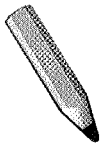


References

Red Hat Enterprise Linux Storage Administration Guide

- Section 10.2: NFS Client Configuration

nfs(5) man page



Test

Criterion Test

Case Study

File Sharing with NFS

Before you begin...

Run **lab-setup-strickland** on desktopX to prepare serverX for the exercise.

Strickland Pro Play is a store specializing in high end recreational equipment and accessories. The new sales software requires requires a file server with two shares that are mounted at each sales station in the store.

For the file server, deploy an NFSv4 service on desktopX. Create and share two exports on desktopX:

- The first export is for intake of current sales orders. On desktopX, export **/share/current** and make it writable. **root** on the client must be able to write to **/share/current** when mounted.
- The second export is to archive old orders. Again on desktopX, export the path **/share/archives** and make it read-only.
- Configure both exports so they are only available to the local classroom network.

Configure serverX to mount **desktopX:/share/current** as **/sales/current** and **desktopX:/share/archives** as **/sales/archives**. The mounts must be available after a reboot of serverX.

When you are ready to check your work, run **lab-grade-strickland** on serverX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Personal Notes

Unit Summary

NFS Concepts and Configuration

In this section you learned how to:

- Differentiate NFSv4 from older versions of NFS
- Configure an NFS server

Using NFS

In this section you learned how to:

- Export directories from multiple partitions with NFSv4
- Mount NFS filesystems automatically at startup
- Determine appropriate mount options for read-only and read-write access when mounting NFS filesystems as startup

UNIT SEVENTEEN

FILE SHARING WITH CIFS

Introduction

Topics covered in this unit:

- CIFS clients
- Essential CIFS configuration
- Collaborative CIFS shares

Accessing CIFS Shares

CIFS, the Common Internet File System, also known as SMB (Server Message Block), is the standard file and printer sharing system for Microsoft Windows servers and clients. Red Hat Enterprise Linux is able to act as both a client and as a server for CIFS file and printer shares.

Note

The below examples connecting to serverX will not function until you complete the next section with an initial configuration of Samba on serverX providing user home directory shares.

In this section, we will review four basic methods for connecting to a CIFS file share:

1. Graphical access to a CIFS share

   This technique uses Nautilus to set up an icon for the share which can be used for drag-and-drop access to its contents.

   Go to **Places → Connect to Server.** Fill in the following fields (leave the others blank):

```
Service type: Windows share
Server: serverX
Share: winuserX
User Name: winuserX
Domain Name: CLASSX
```

2. Command-line ftp-style access to a CIFS share: **smbclient**

```
[root@serverX ~]# smbclient -L instructor.example.com
Enter root's password:  Enter
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.4-68.el6]

        Sharename       Type      Comment
        ---------       ----      -------
        ftp             Disk      Instructor Public FTP
...
[root@serverX ~]# smbclient -L serverX -U winuserX
Enter winuserX's password: winpass
        Sharename       Type      Comment
        ---------       ----      -------
        winuser         Disk      Home Directories
...
[root@serverX ~]# smbclient //serverX/homes -U winuserX
Enter winuserX's password: winpass
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.4-68.el6]
smb: \> ls
  .bash_profile                       H      176  Tue Jun 22 11:49:51 2010
```

3. Manually mount a CIFS share

This technique treats the CIFS file share as a standard network file system from a Linux perspective, just like NFS.

```
mount -t cifs -o user=username //server/share /mntpoint
```

Example:

```
[root@serverX ~]# mount -t cifs -o user=winuserX //serverX/winuserX /mnt/
```

4. Persistently mount a CIFS share

This is a variation on the previous example that mounts the CIFS file share automatically at boot time. Note that a credentials file is used in order to provide the username and password for the share.

Add the following line to **/etc/fstab**:

```
//server/share /mntpoint cifs credentials=/etc/filename 0 0
```

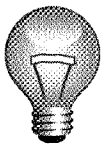
Example:

/etc/fstab:

```
//serverX/homes /mnt/serverX-share cifs credentials=/root/credentials 0 0
```

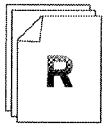
/root/credentials:

```
user=bob
pass=password
```



Important

The standard Microsoft Windows Uniform Naming Convention is ***\\ServerName\Share*** to represent a network resource. However, since \ is an escape character rather than a pathname separator in standard Linux shells such as **bash**, the / character is normally used in place of \ when expressing a UNC on Linux systems.



References

mount.cifs(8) man page

Practice Quiz

Accessing CIFS Share

1. What command-line would give ftp-style access to a CIFS share named "common" on a server named "nas2010", logging you in as a user named "winston"?

_____

2. What is wrong with the following line in /etc/fstab?

```
\\server\share /mnt/point cifs
user=ralph,pass=password 0 0
```

_____

3. How would you store the login credentials in a separate file to keep them out of **`/etc/fstab`**?

_____

4. When mounting a Windows-based CIFS share, what option allows you to specify the Linux ownership of all mounted files?

_____

Providing Home Directories as CIFS Shares

The Samba service can be used to share Linux file systems as CIFS/SMB network file shares, and Linux printers as CIFS/SMB printer shares. In this section, we will look at the basic configuration of a Samba server and specifically at how to share user home directories using Samba.

Parts of the Samba Service

- Packages:
  - **samba-common** - support files for Samba
  - **samba-client** - client applications
  - **samba** - server applications
  - **samba-doc** - documentation (in the "Optional" RHN channel)
- Service script name: **smb**
- Main configuration file: **/etc/samba/smb.conf**

The /etc/samba/smb.conf File

/etc/samba/smb.conf: [global] Section

- `workgroup`

  The **workgroup** is used to specify the Windows Workgroup or Domain name for the network.

- `hosts allow`

  hosts allow is a comma, space, or tab delimited set of hosts which are permitted to access a service. If specified in the **[global]** section then it will apply to all services, regardless of whether the individual service has a different setting.

  You can specify the hosts by name or IP number. For example, you could restrict access to only the hosts on a Class C subnet with something like **allow hosts = 150.203.5.** (using the *trailing dot* notation). The full syntax of the list is described in the man page **hosts_access(5)**.

- `security`

  This option affects how clients respond to Samba and is one of the most important settings in the **smb.conf** file.

  If your PCs use usernames that are the same as their usernames on the UNIX machine then you will want to use **security = user**. If you mostly use usernames that don't exist on the UNIX box then use **security = share**.

  With **security = share**, clients need not log onto the server with a valid username and password before attempting to connect to a shared resource. Instead, the clients send authentication information on a per-share basis, at the time they attempt to connect to that share.

For **security = user**, the client must log in with a valid user name and password before receiving share information or setting parameters like **guest only**.

security = domain will only work correctly if the machine has been added to the NT Domain. It expects the **encrypted passwords** parameter to be set to **yes**. In this mode Samba will try to validate the username/password by passing it to a Windows NT Primary or Backup Domain Controller, in exactly the same way that a Windows NT Server would do. Note that a valid UNIX user must still exist as well as the account on the Domain Controller to allow Samba to have a valid UNIX account to map file access to. You must set the **password server** parameter to give Samba the server to validate passwords.

Using **security = server**, Samba will try to validate the username/password by passing it to another SMB server. You must set the **password server** parameter to give Samba the server to validate passwords.

For **security = ads**, Samba will act as a domain member in an ADS realm. To operate in this mode, the machine running Samba will need to have Kerberos installed and configured and Samba will need to be joined to the ADS realm using the **net** utility. Read the chapter about **Domain Membership** in the HOWTO for details (this HOWTO is part of the **samba-doc** package). In addition to the **net** command, there is a **netdomjoin-gui** command provided by the **samba-domainjoin-gui** package in the Optional repository for joining a machine to ADS.

/etc/samba/smb.conf: Other Sections

- **[homes]**

  This share, which is enabled by default, is a special share that makes user's home directories available via CIFS. It includes **browseable = no**, so it will not show up as an available share until the user authenticates. The share name can either be specified as **homes** (in which case the Samba server will convert it to the home directory path of the user), or ***username*** .

- **[printers]**

  Also available by default, this will share out printers that are currently available.

- **[*share*]**

  If you want to make other shares, place the share name in brackets as above. The share requires at least a **path** parameter. There are several examples in the **smb.conf** file.

Example minimal **/etc/samba/smb.conf** file:

```
[global]
        workgroup = MYGROUP
        server string = Samba Server version %v
        log file = /var/log/samba/log.%m
        max log size = 50

        security = user
        passdb backend = tdbsam

        load printers = yes
        cups options = raw
```

```
[homes]
        comment = Home Directories
        browsable = no
        writable = yes

[printers]
        comment = All Printers
        path = /var/spool/samba
        browsable = no
        guest ok = no
        writable = no
        printable = yes
```

Other Configuration

Samba-only Users

- **useradd**

  security = user requires a UNIX and Samba account information. Either add a user (preferably using the same name as the Samba account), or place an entry in **/etc/samba/smbusers** (it has some examples). If you are creating a Samba-only user, set the UNIX password to **/sbin/nologin**.

```
[root@serverX ~]# useradd -s /sbin/nologin winuser
```

- **smbpasswd**

  If you do not have a Samba password server, you must create authentication data on the local machine. Use **smbpasswd** to create Samba accounts and passwords.

  If **smbpasswd** is passed a username without any options, it will attempt to *change* the account password. Passing the **-a** option will add the account and set the password.

```
[root@serverX ~]# smbpasswd -a winuser
New SMB password:  winpass
Retype new SMB password:  winpass
...
Added user winuser.
```

Securing Samba

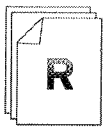
- samba_enable_home_dirs and use_samba_home_dirs SELinux booleans

  The **samba_enable_home_dirs** boolean allows local Linux home directories to be exported as CIFS file shares to other systems. The **use_samba_home_dirs** boolean, on the other hand, allows remote CIFS file shares to be mounted and used as local Linux home directories. It is easy to confuse the two options. See the **samba_selinux**(8) man page for more information.

```
[root@serverX ~]# setsebool -P samba_enable_home_dirs on
```

- Service Ports

  Samba normally uses TCP/445 for all connections. It also uses UDP/137, UDP/138 and TCP/139 for backward compatibility.



References

smb.conf(5), **smbd**(8), and **samba_selinux**(8) man pages

Red Hat Enterprise Linux Managing Confined Services
- Chapter 4: Samba

/usr/share/doc/samba-doc-*/ (from **samba-doc** package in the Optional repository)

Practice Performance Checklist

Samba Home Directories Configuration

Modify the default Samba configuration and security elements to support access to user home directories.

- ☐ Log in to serverX and become root.
- ☐ Install the necessary package(s) for a Samba server
- ☐ Start and enable the Samba service
- ☐ Configure system to be in the CLASSX workgroup (where X is your station number) with local user definitions.
- ☐ Add a Samba-only user named **winuserX** (where X is your station number) with a Samba password of **winpass**.
- ☐ Enable user home directory access in SELinux
- ☐ Enable the firewall and open up necessary ports to grant access.
- ☐ Test the configuration, by accessing your Samba-only user's home directory from desktopX.

Configuring Group and Print CIFS Shares

Search & Learn: Configuring CIFS Shares

Consult the reference documents below to answer the following questions:

Three Steps to Provide a Group Share:

1. The first step would be to create a collaborative directory in Linux; that is, a directory writable by a particular group in which all new files are automatically owned and writable by that group. From your previous experience, what three steps are needed to accomplish this?

2. Next, we need to set a correct SELinux context on this directory so that it may be shared using CIFS/Samba. What two steps are needed to set the correct context persistently (across SELinux relabels)?

3. How do we share this directory via Samba?

Two Steps to Provide an Individual Printer Share:

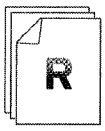
1. How do you prevent the automatic sharing of all locally defined printers as CIFS printer shares by Samba?

2. How would you share a particular printer manually?

Limiting Client System Access:

1. How can you limit which client systems can access a particular CIFS share by IP address?

2. Is there a non-Samba facility that can limit which client systems can access all CIFS shares by the client's source IP address?



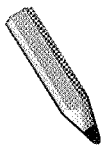
References

smb.conf(5), **smbd**(8), and **samba_selinux**(8) man pages

Red Hat Enterprise Linux Managing Confined Services
- Chapter 4: Samba

/usr/share/doc/samba-doc-*/ (from **samba-doc** package in the Optional repository)

Red Hat Enterprise Linux Deployment Guide
- Section 15.5.1: Group Directories



Test

Criterion Test

Case Study

File Sharing with CIFS

Before you begin...

Run **lab-setup-samba** on desktopX to prepare serverX for the exercise.

The School of Butler and Hacker has recently deployed several CIFS servers to allow their Windows client systems access to file shares.

The Color Guard, known as Green and Red is deploying a new server and needs to share information using CIFS. That share must be writable by members of the Color Guard, but other people can only have read access.

Enable the firewall and allow all clients on the local network to access the CIFS server.

Configure your serverX to function as a CIFS server, with the following information:

- Workgroup: BUTLER
- Linux Group: greenred
- CIFS Share Name: school
- Directory: /shared/school
- No printers shared

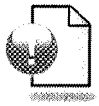
Test the configuration by:

- Creating a user as a member of **greenred** and ensuring they can write to the CIFS share, **school**
- instructor.example.com provides several printers that CUPS should automatically enable. Before you fulfill the printer requirement, check to verify they are available (they should be named printerX). Configure Samba so that no printers are shared and confirm that the user can NOT see them listed with **smbclient**
- Creating a second user not as a member of **greenred** and ensuring they can only read from the CIFS share, **school**

When you are ready to check your work, run **lab-grade-samba** on serverX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.

Personal Notes



Unit Summary

Accessing CIFS Shares

In this section you learned how to:

- Access CIFS-based file and print shares

Providing Home Directories as CIFS Shares

In this section you learned how to:

- Configure a CIFS based home directory server
- Add Samba-only users to the system

Configuring Group and Print CIFS Shares

In this section you learned how to:

- Create and configure a CIFS share which can be used for group collaboration

UNIT EIGHTEEN

FILE SHARING WITH FTP

Introduction

Topics covered in this unit:

- "Drop-box" anonymous upload

FTP Drop-box Anonymous Upload

FTP, the File Transfer Protocol, is one of the oldest network protocols still in use on the Internet. Many organizations still use FTP for basic file transfers that do not require strong security.

It can be useful to set up a FTP server that allows anonymous users to upload content. However, it is important to configure the server *not* to allow the anonymous users to download content from the upload directory. If the anonymous FTP user can download content from the FTP site that some other anonymous FTP user uploaded, your FTP site can be used by people as a way to transfer illegal or inappropriate content without oversight. It is good practice to ensure that any content served by your site has been approved in some way by an authorized administrator.

In this section, we will look at how to configure an FTP upload directory from which the anonymous user can not download or even list the contents.

FTP Buzz Groups

Using the documentation references below, research the answer to one or more of the below assigned questions. Take notes for the remaining unassigned questions when the class re-groups.

1. ***Create upload directory***

   What filesystem permissions could create a "drop-box"?

2. ***Modify SELinux for anonymous upload***

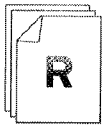
   Are there appropriate contexts and booleans to set?

3. ***Modify `/etc/vsftpd/vsftpd.conf`***

   What needed anonymous-related options are there?

4. ***Modify `iptables` to support inbound ftp connections***

   What ports does ftp use? How do we support passive ftp?



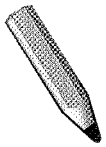
References

Red Hat Enterprise Linux Security Guide
- Section 2.2.6: Securing FTP

Red Hat Enterprise Linux Security-Enhanced Linux
- Section 5.6: Booleans

Red Hat Enterprise Linux Managing Confined Services
- Section 5.4.1: Uploading to an FTP site

ftpd_selinux(8) and **vsftpd.conf**(5) man pages



Test

Criterion Test

Case Study

FTP Drop Box

Before you begin...

Run **`lab-setup-dropbox`** on desktopX to prepare serverX for the exercise.

The Quiet Pleases company, a manufacturer of silence cones and other noise canceling devices, has a program to collect information about noise levels around the world. Volunteers have been collecting data about noise and need an easy way to send in reports.

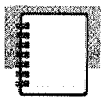
The company has decided to use an FTP server with an anonymous upload directory to collect the reports.

Deploy vsftpd on your serverX and configure a write-only upload directory that is accessible at: `ftp://serverX.example.com/dropbox`

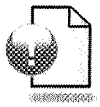
As the volunteers are located all over world, the FTP server must accept connections from anywhere on the internet.

When you are ready to check your work, run **`lab-grade-dropbox`** on desktopX.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

FTP Drop-box Anonymous Upload

In this section you learned how to:

- Configure FTP drop-box service
- Manage SELinux to support FTP uploads
- Manage the firewall to support FTP transfers

UNIT NINETEEN

TROUBLESHOOTING THE BOOT PROCESS

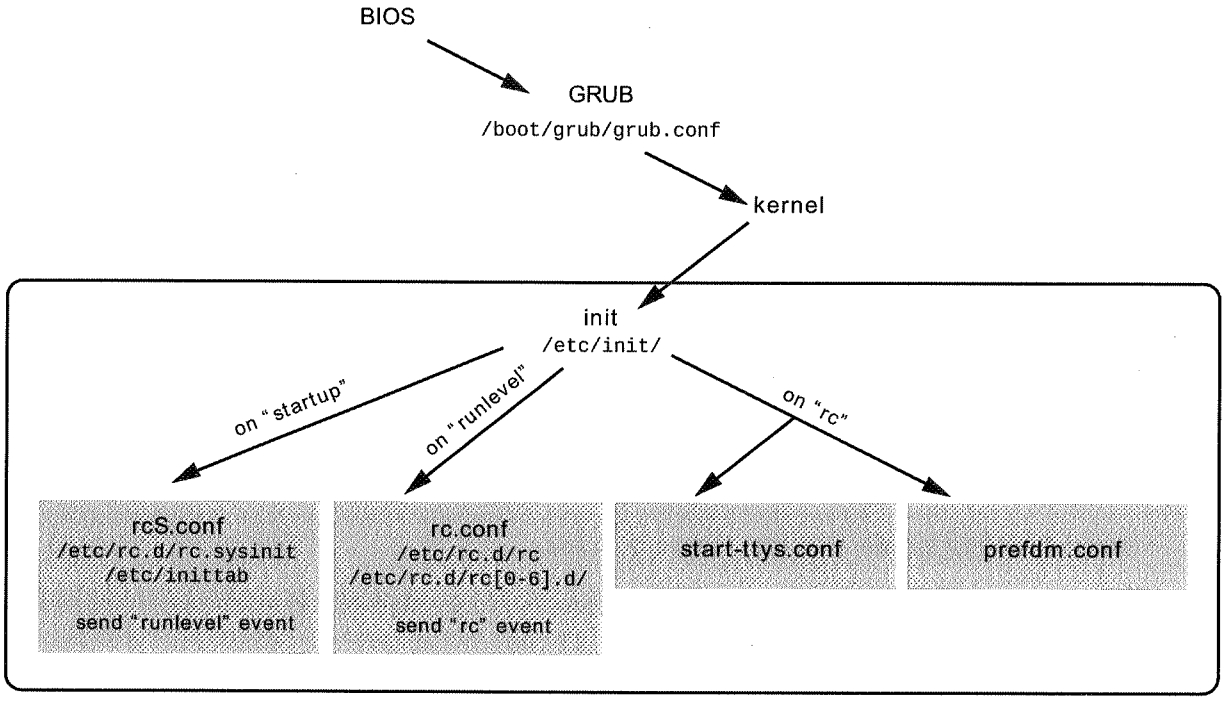
Introduction

In this unit you will learn how to:

- Understand the boot process and various levels of repair opportunities.
- Gain access to a system that is no longer bootable.
- Install software from inside rescue mode.
- Repair problems with the bootloader or a corrupted file system.
- Configure a serial TTY console.

The Boot Process and Rescue Mode

Below is a simplified diagram of the Red Hat Enterprise Linux 6 boot process, from power-on to the point at which a login prompt appears on the screen.



BIOS

The BIOS, or Basic Input/Output System, is the firmware interface built into standard x86/x86-64 hardware that puts the hardware into a known state and prepares the system to load an operating system.

What happens?

- Detects and initializes hardware
- Determines the device to boot from

What can go wrong?

- Incorrect or weird BIOS settings
- Bad boot device ordering

How can it be interrupted or influenced?

- Press a vendor-specific key
- Use a vendor-specific configuration utility
- Often, <F12> can perform a one-time override of the boot ordering

GRUB

GRUB, the GRand Unified Bootloader, is loaded by the BIOS and used to select and start the operating system, as we have already discussed.

What happens?

- Loads initial RAM file system ("initramfs")
- Loads and executes kernel
- Provides kernel's command-line

What can go wrong?

- Misconfiguration of the bootloader
- Bad kernel image or initramfs
- Bad kernel command line

How can it be interrupted or influenced?

- Choose an alternate pre-configured menu item
- Use "e" or "a" to select a different kernel image or edit the kernel command-line
- Edit the kernel command-line to boot from **`single`** user mode
- Boot with **`init=/bin/bash`**

Kernel

The Linux kernel is the heart of the operating system. It is responsible for managing hardware access for userspace processes. Drivers and the KVM hypervisor are integrated parts of the kernel.

What happens?

- Detect hardware devices
- Load device drivers (modules) for the devices

Note

Where does the kernel get modules to load at boot time?

1. Initially, it uses the initial RAM disk configured for the kernel in **`/boot/grub/grub.conf`**: **`/boot/initramfs-<VERSION>.img`**
2. Once the root file system is mounted, it uses **`/lib/modules/<VERSION>/`**

- Mount the root file system read-only

- Start the initial process, **init**

What can go wrong?

- Bad initial RAM file system image
- Badly identified root file system
- Corrupted root file system

How can it be interrupted or influenced?

- Generally only through GRUB options

init and Upstart

The first userspace process started on the machine is **/sbin/init**. The **init** process is responsible for starting all remaining userspace processes, directly or indirectly.

What happens?

- Once the kernel is running, it starts **init**. The **init** program is responsible for completing the boot process by starting all other non-kernel system processes.

  With **Upstart**, **init** starts "jobs" when various "events" happen, such as when the system boots, we enter a runlevel, or another **init** job starts or stops. These jobs are stored as scripts in the **/etc/init/** directory. At boot, the startup event causes **init** to run the **/etc/init/rcS.conf** job which:

  - Runs **/etc/rc.d/rc.sysinit** to start LVM, mount and check file systems, set the system clock, and do other housekeeping.
  - Looks in **/etc/inittab** to find the runlevel.
  - Sends an event to **init** telling it to enter that runlevel.

  The runlevel event causes **init** to run the **/etc/init/rc.conf** job which runs the **/etc/rc.d/rc** script with the desired runlevel as an argument:

  - Example: **rc.conf** runs **rc 5**, which runs **/etc/rc.d/rc5.d/K* stop** and **/etc/rc.d/rc5.d/S* start**.
  - The scripts are run in numeric order, first the K's and then the S's.
  - The **/etc/rc.d/rc5.d/** scripts are symlinks to the scripts used by service.
  - Whether the links start with a K or an S depends on whether the service has been turned **on** or **off** with **chkconfig**.

What can go wrong?

- Mangled **/etc/fstab** or **/etc/crypttab**
- Network or service misconfiguration leading to hung services
- Many more...

How can it be interrupted or influenced?

- Press "Alt-D" from graphical environment to view error messages
- Press "I" (capital i) during service startup to select services interactively

Note

Red Hat Enterprise Linux 5 and earlier used a different implementation of **init**, SysVinit, that was driven by directives in the **/etc/inittab** file. The basic boot process and scripts run by **init** were similar, however.

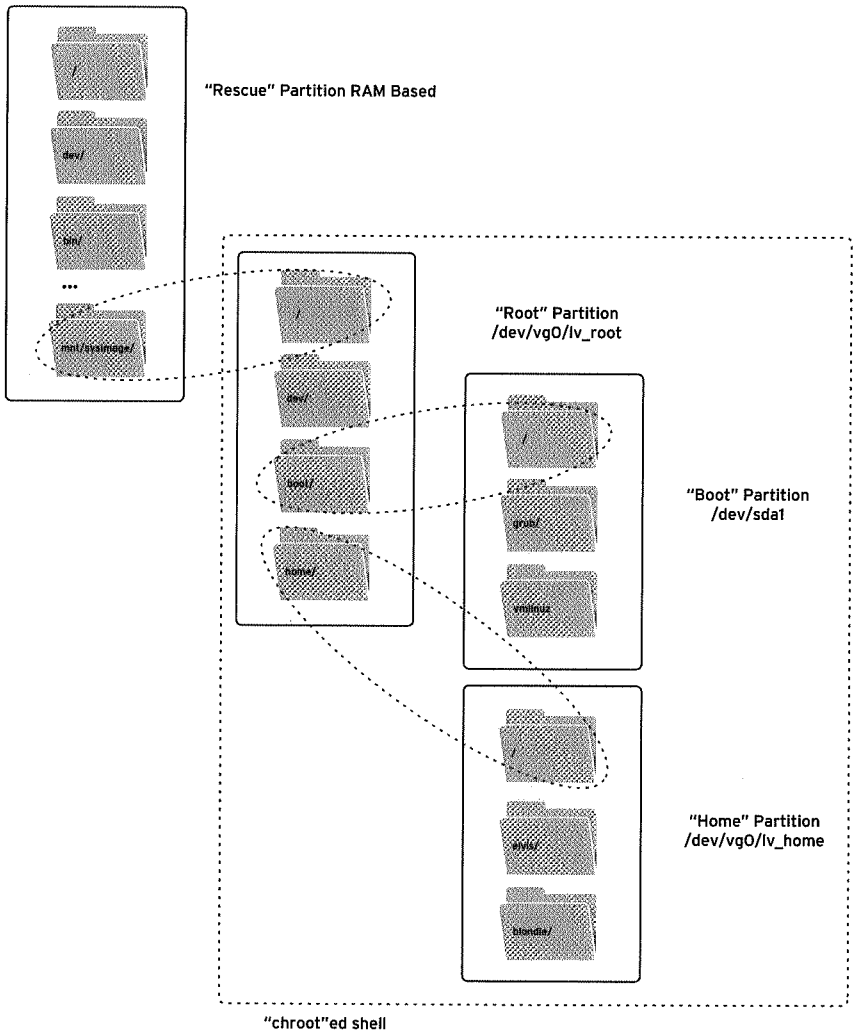
Note

Red Hat Enterprise Linux 6 supports systems that use UEFI and the UEFI Boot Manager to load the operating system instead of a BIOS and GRUB. For more information, see the *Red Hat Enterprise Linux Installation Guide*.

The Rescue Shell

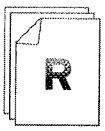
Recovery modes are useful when you are able to use GRUB, but what if GRUB is broken? The Rescue shell, a special mode of the installer, allows you to boot and recover the system when it is otherwise unbootable.

To enter the rescue shell, start as you would an installation, but choose *Rescue Installed System* from the initial menu or append **rescue** as an argument to the kernel.



The first thing you often want to do in the rescue shell is to access or recover file systems on your local hard drive. Rescue mode attempts to mount your system's root file system (and others) under **/mnt/sysimage**, as pictured above.

A useful trick is to use **chroot /mnt/sysimage** to start a sub-shell where / represents the root file system of your hard drive instead of the root directory of the rescue environment.



References

Red Hat Enterprise Linux Deployment Guide
- Section 3.2.2: Using RPM - Installing and Upgrading

Red Hat Enterprise Linux Deployment Guide
- Section 23.6: Verifying the Boot Loader

Red Hat Enterprise Linux Installation Guide
- Section 36.1: Rescue Mode

Red Hat Enterprise Linux Installation Guide
- Appendix E: The GRUB Boot Loader

Red Hat Enterprise Linux Installation Guide
- Appendix F: Boot Process, Init, and Shutdown

`info chroot`

`info grub`

Practice Exercise

Using the Rescue Environment

Before you begin...

Run **`lab-setup-bootbreak`** on desktopX.

Carefully perform the following steps. Ask your instructor if you have problems or questions.

This timed drill is designed to give you practice accessing the rescue environment. The installation path is *http://instructor.example.com/pub/rhel6/dvd*. The path for individual packages is *http://instructor.example.com/pub/rhel6/dvd/Packages/*

Perform the following steps on serverX unless directed otherwise.

1. After serverX has booted, run the **`lab-setup-bootbreak-0`** script as **`root`** on serverX. This script will alter your system and cause difficulties booting.
2. Boot into the rescue environment to diagnose and resolve the issue.
3. Confirm the problem has been solved by rebooting the system.
4. Repeat this process as often as possible during the allotted time.

Repairing Boot Issues

Reinstalling GRUB

First stage GRUB is a small binary that is installed into the Master Boot Record (MBR) of a bootable disk. Usually, GRUB is installed by the Anaconda installer, and never has to be re-installed. Occasionally, if a disk is damaged or moved, you may need to reinstall GRUB manually.

Procedure for Reinstalling GRUB

1. Invoke GRUB

```
[root@serverX ~]# grub
```

2. In some situations, map the Linux hard drive to a GRUB identifier:

```
grub> device (hd0) /dev/vda
```

3. Identify the **/boot** partitioning

```
grub> root (hd0,0)
```

GRUB refers to hard drives as **(hd0)** or **(hd1)**, which refers to "BIOS drive #0" or "BIOS drive #1". The actual drive can vary depending on BIOS. The first partition on **(hd0)** would be **(hd0,0)**, which might be **/dev/sda1** or **/dev/vda1**.

4. Install first stage grub into the MBR

```
grub> setup (hd0)
```

5. Exit grub

```
grub> quit
```

Repairing Damaged File Systems

In normal operation, the kernel keeps frequently accessed file system information in memory, and only periodically commits the information to disk. If a file system becomes unexpectedly unavailable (such as due to a power outage or physical connectivity problem), the on-disk file system will contain inconsistencies. If these errors are not fixed, they are likely to lead to corrupt data.

The **fsck** command will attempt to restore a file system to a self-consistent state. The guarantee of **fsck** is not full data recovery, but consistency of the file system. On boot, the startup scripts will automatically **fsck** all file systems (if tagged in **/etc/fstab**). Minor problems will be resolved without interaction. If an automatic repair to a file system risks data loss, the boot process will drop to a shell to allow an administrator to run **fsck** interactively. If the root partition is damaged, it might be necessary to boot the rescue environment to run **fsck**.

1. Unmount the **/boot** file system on **/dev/vda1**.

```
[root@demo ~]# umount /dev/vda1
```

2. Check the file system on **/dev/vda1**.

```
[root@demo ~]# fsck /dev/vda1
```

3. Remount the **/boot** file system.

```
[root@demo ~]# mount /dev/vda1
```

Procedure for Editing Files from the Maintenance Shell

This is sometimes necessary when a system can not mount file systems due to typos in **/etc/fstab**, **/etc/crypttab**, or related files, and the system drops to a maintenance shell with **/** mounted read-only.

1. Remount the root file-system read-write:

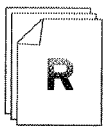
```
(Repair filesystem 1)# mount -o remount,rw /
```

2. Edit any needed files.
3. Mount all other file systems to confirm they mount:

```
(Repair filesystem 3)# mount -a
```

4. Exit the maintenance shell:

```
(Repair filesystem 4)# exit
```



References

Red Hat Magazine: "Using GRUB to overcome boot problems"
http://magazine.redhat.com/2007/03/21/using-grub-to-overcome-boot-problems/

Practice Quiz

Repairing Boot Problems

1. In maintenance mode, run _____ to mark the / partition as writable.

2. Assuming you have only one hard drive, and the first partition contains /boot, if you have minor MBR corruption, fix the corruption issue by booting into _____ mode, then run the _____ command. Sometimes you have to type the **`device (hd0) /dev/vda`** command to identify the disk you want to update. Type _____ followed by _____, then exit.

3. If you have file system corruption issues, the machine will boot into _____ mode.

4. In maintenance mode, run _____ to fix _____ corruption issues.

Configuring a Serial Console

Consoles

- *console*: a text-based output device for system administration
- *system console*: where kernel and boot messages are sent (**/dev/console**)
- *physical console*: the computer's attached display and keyboard (**/dev/tty0**). This is the default system console in Linux, **/dev/console** points here.
- *virtual console*: one of several consoles provided by the physical console on **/dev/tty1** and up; these provide independent "terminals" on the physical hardware. Six consoles allow logins out of the box. Typing **Ctrl**+**Alt**+**F*n*** (where *n* is a number) allows you to change from one virtual console to another.

Note

The graphical login normally runs on **/dev/tty1** in Red Hat Enterprise Linux 6.

- *serial console*: a terminal connected to a serial port (**/dev/ttyS0** and up) or a USB serial adapter (**/dev/ttyUSB0** and up). Setting up a serial console requires additional configuration.

Search & Learn: Configure a Serial Console

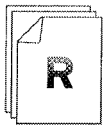
Your goal is to determine how to configure a Red Hat Enterprise Linux 6 machine to use a serial console. It should do the following:

- Use the serial console as the system console, outputting kernel and boot messages there
- Allow logins on the serial console

Use the **/usr/share/doc/kernel-doc/Documentation/serial-console.txt** and **/etc/init/serial.conf** files for reference.

When reading **serial-console.txt**, focus on kernel command line arguments. Do not worry about recompiling your kernel or making your own device nodes; this has been done for you already in Red Hat Enterprise Linux. Note that much of the setup discussion for **init** is for SysVinit, which was shipped in older versions of Red Hat Enterprise Linux but is not relevant in Red Hat Enterprise Linux 6.

Use this space for notes



References

Red Hat Enterprise Linux Virtualization section 34.4 (Troubleshooting with serial consoles)

/usr/share/doc/kernel-doc-2.6.32/Documentation/serial-console.txt

/etc/init/serial.conf

agetty(8) and **mingetty**(8) man pages

info grub serial (GRUB serial console redirection)

Practice Performance Checklist

Configuring a Serial Console

In this practice exercise, you will use the **minicom** terminal program on desktopX to act as a VT100 terminal connected to serverX.

Perform the following steps on serverX unless directed otherwise.

- ☐ Configure serverX to send console messages to the first serial port.

- ☐ Determine the source path associated with the virtual serial line. Record the source path in the space provided.

  _____

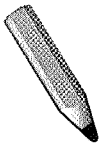
- ☐ Install the **minicom** package on desktopX. Invoke **minicom**, providing the terminal from the previous step as the argument to the **-p** option.

  Note that the **Ctrl+a**, **z** key sequence invokes a configuration menu. **Ctrl+a**, **x** key sequence exits **minicom**.

- ☐ While watching the **minicom** terminal, reboot serverX. As serverX reboots, you should be able to observe startup messages in the minicom terminal.

- ☐ When the startup process completes, you should be able to log into your machine using the minicom terminal.

- ☐ When finished, edit **/boot/grub/grub.conf**, removing the **console** kernel command line parameter, and reboot serverX.



Test

Criterion Test

Exercise

Troubleshooting the Boot Process

Before you begin...

Run the **lab-setup-bootbreak** command on desktopX to setup the lab.

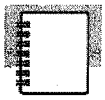
Carefully perform the following steps. Ask your instructor if you have problems or questions.

This practice exercise includes three break/fix challenges. For each, your serverX virtual machine will be modified in some way that prevents it from booting correctly and you will diagnose and correct the problem.

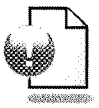
Perform the following steps on serverX unless directed otherwise.

1. Run **lab-setup-bootbreak-1** on serverX. After running the script, serverX should no longer boot correctly. Diagnose and correct the problem. You will know you have the solution when serverX boots normally again.

2. When you have resolved the first scenario, repeat the process with **lab-setup-bootbreak-2** and **lab-setup-bootbreak-3**.



Personal Notes



Unit Summary

The Boot Process and Rescue Mode

In this section you learned how to:

- Describe the boot process for Red Hat Enterprise Linux 6

Repairing Boot Issues

In this section you learned how to:

- Reinstall GRUB
- Check and repair errors on file systems
- Edit files from the read-only file system maintenance shell

Appendix A. Solutions

Getting Started with the Classroom Environment

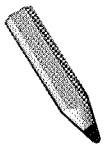
1. Install
2. Start
3. Enable
4. Test

Practice Quiz

System Administration Review

1. The virt-manager command will run the Virtual Machine Manager GUI, and virsh is the command line equivalent.
2. Press **Ctrl+Alt** to release the pointer in the virtual machine graphical console.
3. Passing the argument s to the kernel at boot executes the boot process up to the end of /etc/rc.sysinit and then opens a root shell without asking for a password.
4. In Red Hat Enterprise Linux 6 the System Security Services Daemon sssd manages access to remote directories and authentication mechanisms for clients.
5. The system-config-authentication command provides a GUI to configure user authentication.
6. In order to trust an LDAP server we use a Certificate Authority file to verify the TLS / SSL connection.
7. Run "service autofs reload" to have the automounter reload its main configuration file / etc/auto.master.
8. An indirect map is specified in the /etc/auto.master file by giving a directory which will contain mount points and the name of a second file containing information about the mount points managed in that directory.
9. An automounter map file can use the * wildcard character for the key and use the match in the source path with the & character.
10. The command to generate a new ssh key pair is ssh-keygen.
11. The command ssh-copy-id will copy the user's public key into a ~/.ssh/authorized_keys file.
12. To allow only certain usernames to login via ssh, edit /etc/ssh/sshd_config and add the AllowUsers directive, or to restrict root add PermitRootLogin no.

13. Run "yum install httpd" to install Apache, "service httpd start" to start it, and "chkconfig httpd on" to enable it to persist across reboots.



Test

Criterion Test

Case Study

Getting Started with the Classroom Environment

Before you begin...

Run **lab-setup-server** on desktopX to prepare serverX for the exercise.

Duffey Dynamics Institute, a research firm investigating deep water human habitation, recently dismissed their System Administrator. Several new Red Hat Enterprise Linux servers are installed but not configured for use.

One of the new systems, your serverX, is needed to provide services to the new Underwater Farming Lab.

The following services are needed:

- Configure serverX to authenticate users via the main company LDAP server using TLS.
  - LDAP Server: instructor.example.com
  - LDAP Search Base DN: **dc=example,dc=com**
  - LDAP Certificate: **ftp://instructor.example.com/pub/example-ca.crt**
  - Use LDAP Authentication
- Configure serverX to automount user home directories on demand via NFS.
  - NFS Server: instructor.example.com
  - NFS Export: **/home/guests/*username***
- The lab also needs a web server, so install Apache and set it to start at boot.
- Lastly, lock down the SSH server so that root cannot login.

1. `[root@desktopX ~]# lab-setup-server`

2. Use **system-config-authentication** to configure serverX to get network user information from the classroom LDAP server.
   - For "User Account Database", choose *LDAP*.
     1. For "LDAP Search Base DN:", enter *dc=example,dc=com*.
     2. For LDAP Server:, enter *instructor.example.com*
     3. Select the "Use TLS to encrypt connects" checkbox.

4. Choose "Download CA Certificate...", and enter *http://instructor.example.com/pub/example-ca.crt*.

- For "Authentication Method", choose *LDAP*.
- Choose *Apply*.

Optionally, you could use the following from the command-line:

```
[root@serverX ~]# authconfig --enableldap --enableldapauth --
ldapbasedn=dc=example,dc=com --enableldaptls --ldaploadcacert=ftp://
instructor.example.com/pub/example-ca.crt --ldapserver=instructor.example.com --update
```

3.
```
[root@serverX ~]# echo -e "/home/guests\t/etc/auto.guests" >> /etc/auto.master
[root@serverX ~]# echo '* -rw,hard,intr instructor.example.com:/home/guests/&' >> /
etc/auto.guests
[root@serverX ~]# service autofs reload
```

4.
```
[root@serverX ~]# yum -y install httpd
Already installed.
[root@serverX ~]# service httpd start

[root@serverX ~]# chkconfig httpd on
```

5.
```
[root@serverX ~]# sed -i.orig -r -e 's/#PermitRootLogin yes/PermitRootLogin no/' /etc/
ssh/sshd_config
[root@serverX ~]# service sshd restart
```

Enhance User Security

Search & Learn: Configuring sudo

Consult the reference documents below to answer the following questions. The instructor will review the solutions (as found in the appendix) with the class when the activity is complete.

1. What is the basic syntax of a sudoers rule (a *user specification* that controls which commands a user may run)?

   ```
   user  MACHINE=(RUNAS_USER) COMMANDS
   ```

   There are two kinds of sudoers entries. The kind that most people think of as a "sudoers rule" is a *user specification*, which controls which commands a user may run as which users on which hosts, which is the rule above. It specifies that *user* on *MACHINE* may run, as the user *RUNAS_USER* the comma separated list of commands *COMMANDS*.

   The *MACHINE* field allows the same **/etc/sudoers** file to be copied to multiple machines but have different effects on different machines.

   The other type of sudoers entry is an *alias* which allows a variable to be set which can be used in one of the four fields of a user specification. See the man page for details.

2. What sudoers entry would grant members of the *wheel* group access to any command?

   ```
   %wheel ALL=(ALL) ALL
   ```

3. How do you prevent a user from being prompted by **sudo** for their own password?

   Use **NOPASSWD:** before the command(s) in **/etc/sudoers**. For example:

   ```
   %wheel ALL=(ALL) NOPASSWD: ALL
   ```

4. How would you grant a list of users (who do not share a common group) **sudo** access to any command?

   ```
   User_Alias USERS = user1, user2, user3
   USERS ALL=ALL
   ```

   This approach uses a **User_Alias** entry to set the user list for the following user specification rule.

5. How could you set up sudoers entries to grant a list of users **sudo** access to a specific list of commands (rather than to all commands)?

   ```
   User_Alias USERS = user1, user2, user3
   Cmnd_Alias COMMANDS = command1, command2
   USERS ALL=COMMANDS
   ```

   Note that *COMMANDS* is any arbitrary string to name the **Cmnd_Alias** variable.

Practice Resequencing Exercise

sudo Rules Syntax

For each of the five requirements below, write down the number of the matching rule on the line in front of the definition.

5 Grants members of *wheel* group access to any command
3 Associates users *bob*, *mary*, *david*, and *george* with the name **ADMINS**
2 Grants user *joseph* access to commands in the **NETWORKING** group
9 Grants user *natalie* access to **SOFTWARE** commands without needing her password
8 Grants the *dba* group access to the **service** and **chkconfig** commands for the **mysql** service

1. `%wheel  ALL=(ALL) NOPASSWD:ALL`

2. `joseph  ALL=  NETWORKING`

3. `User_Alias ADMINS = bob, mary, david, george`

4. `%joseph  ALL= /sbin/ip, /bin/ping, /sbin/ifconfig, /sbin/route`

5. `%wheel  ALL=  ALL`

6. `natalie  ALL=(ALL) SOFTWARE`

7. `%dba  /sbin/service mysql, /sbin/chkconfig mysql ALL`

8. `%dba  ALL= /sbin/service mysql, /sbin/chkconfig mysql`

9. `natalie  ALL=  NOPASSWD: SOFTWARE`

Practice Performance Checklist

Kerberos Configuration Exercise

You will modify your previous LDAP-based configuration to now use only Kerberos for authentication. LDAP will still be used to provide account information.

Perform the following steps on serverX unless directed otherwise.

☐ Verify the necessary packages are installed for Kerberos authentication.

```
[root@serverX ~]# rpm -q krb5-workstation
krb5-workstation-1.8.2-3.el6.x86_64
```

If *krb5-workstation* is not installed, use **yum** to install it:

```
[root@serverX ~]# yum install -y krb5-workstation
```

- ☐ Configure system to use the following LDAP and Kerberos settings:
  - LDAP Server: instructor.example.com (uses TLS)
  - LDAP Certificate: `ftp://instructor.example.com/pub/example-ca.crt`
  - LDAP Base DN: dc=example,dc=com
  - Kerberos Realm: EXAMPLE.COM
  - Kerberos KDC: instructor.example.com
  - Kerberos Admin Server: instructor.example.com
  - Be sure the **sssd** service is enabled

  Launch **system-config-authentication**. Select LDAP for the User Account Database and Kerberos password for Authentication Method. Provide the information above for each of the two services. Once you complete the information and apply your changes, the sssd service should be started. To confirm the service is running, do the following:

```
[root@serverX ~]# service sssd status
sssd (pid 2634) is running...
```

- ☐ Test the change by logging in to serverX with ssh:
  - Username: **ldapuserX** (where X is your station number)
  - Password: **kerberos**

  The testing can be performed via ssh or by switching to an alternate virtual console on virt-manager.

Authentication Troubleshooting

As the instructor answers each of these questions, take notes in the space provided:

1. What is the drawback of using LDAP or Kerberos to authenticate desktop or laptop users versus locally-defined user accounts?

   When the system is "offline", credentials cannot be verified with the online services (LDAP or Kerberos), thus preventing login.

2. What can be implemented to resolve this problem?

   The SSSD service is used as a client for LDAP and Kerberos servers, that, by default, will cache previous credentials, thus, enabling offline login.

3. How does one configure SSSD?

   If the **sssd** package is installed, it is configured automatically when Kerberos or LDAP is setup with the Authentication Configuration tool (**system-config-authentication**).

4. What if I want to configure SSSD from the command line?

   The **authconfig** command line tool, like **system-config-authentication**, will transparently configure **SSSD**. Alternatively, manual settings for SSSD are found in the **sssd.conf**(5) man page and the Red Hat Enterprise Linux Deployment Guide, Section 8.2: The System Security Services Daemon (SSSD).

5. How will this affect troubleshooting the authentication process?

   Remember that answers may be serviced from a cache potentially hiding a down authentication server. The "cache" also means that **getent passwd** will no longer dump the entire list of users from the LDAP server, though **getent passwd *username*** will view one user's information.

6. Where can we see what the SSSD service is doing?

   Troubleshooting sssd:

   - *Look in **/var/log/sssd/** for logs.*
   - *Modify **/etc/sssd/sssd.conf** to increase information logged:*

   ```
   debug_level=10
   ```

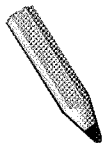
7. What if I cannot log in to view the log files or correct an authentication misconfiguration?

   Single-user mode or runlevel 1.

Practice Quiz

Troubleshooting Authentication Quiz

1. How does one normally configure SSSD? Authentication Configuration Tool, **authconfig**, or edit **/etc/sssd/sssd.conf**
2. Which directory holds log messages from sssd? **/var/log/sssd/**
3. How can we increase the logging detail that is generated? Change **/etc/sssd/sssd.conf** and set **debug_level=[0-10]** where higher means more detail. This is set in individual "service" areas in the file, allowing for unique levels.
4. When you cannot log in to correct an authentication misconfiguration, what approaches are available to you? Single-user mode or runlevel 1



Test

Criterion Test

Case Study

Enhance User Security

Before you begin...

Run **lab-setup-taylorlocke** on desktopX to prepare serverX for the exercise.

Taylor and Locke, a prestigious law firm, recently hired a security consultant to advise them regarding their servers. As the law firm's servers hold sensitive client information, security is a priority!

The security consultant recommended that all servers use LDAP for centralized accounts and Kerberos for authentication. Overall, the LDAP/Kerberos deployment went well. However, one of the servers that you manage appears to be mis-configured.

Correct the configuration on serverX so that LDAP users are able to login with Kerberos authentication (details below).

- LDAP Server: **instructor.example.com** (uses **TLS**)
- LDAP Certificate: **ftp://instructor.example.com/pub/EXAMPLE-CA-CERT**
- LDAP Base DN: **dc=example,dc=com**
- Kerberos Realm: **EXAMPLE.COM**
- Kerberos KDC: **instructor.example.com**
- Kerberos Admin Server: **instructor.example.com**

Test the change by logging in to serverX with ssh:

- Username: **ldapuserX** (where *X* is your station number)
- Password: **kerberos**

Once LDAP users can login, configure autofs to provide automounted home directories. The home directories are shared from instructor.example.com.

The security consultant suggested that administrative privileges on their servers be limited as much as possible.

On serverX, set up **sudo** to allow the following:

- Users **morris**, **borris**, and **horace** may use sudo to run only the **service** and **chkconfig** commands without supplying their password.

  Note: You will need to add accounts on serverX for **morris**, **borris**, and **horace**. Do not add **morris**, **borris**, and **horace** to any supplemental groups.

When you are ready to check your work, reboot serverX and run **lab-grade-taylorlocke** on serverX.

1. `[root@desktopX ~]# lab-setup-taylorlocke`

2. Take serverX to single user mode.

3. Configure LDAP/Kerberos authentication:

```
# authconfig --enableldap --ldapserver=instructor.example.com --enableldaptls
--ldaploadcacert=ftp://instructor.example.com/pub/example-ca.crt
--ldapbasedn="dc=example,dc=com" --disableldapauth --enablekrb5
--krb5kdc=instructor.example.com --krb5adminserver=instructor.example.com
--krb5realm=EXAMPLE.COM --enablesssd --enablesssdauth --update
```

4. Type **exit** to take serverX to multi-user mode.

5. Add the following line to **/etc/auto.master**:

```
/home/guests    /etc/auto.home
```

Create and add the following to **/etc/auto.home**:

```
*   -rw,hard,intr   instructor.example.com:/home/guests/&
```

Make the automounter reload its configuration:

```
[root@serverX ~]# service autofs reload
```

6. Enhance the configuration with sudo

Add the users (morris, borris, and horace) to the system:

```
useradd morris; useradd borris; useradd horace
```

Modify **/etc/sudoers** to include the following two lines:

```
User_Alias ADMINS = morris, borris, horace
Cmnd_Alias SERVICES=/sbin/service, /sbin/chkconfig
ADMINS ALL=NOPASSWD:SERVICES
```

7. When you are ready to check your work, reboot serverX and run **lab-grade-taylorlocke** on serverX.

Bash Scripting and Tools

Practice Case Study

Writing a Bash Shell Script

Perform the following steps on serverX unless directed otherwise.

Sam has asked you to write a shell script to generate a report called "$am's Report". The report should take a list of directories on the command line and will prompt for the name of the report requester.

If enough arguments are not given, an error message is displayed:

```
usage: ./samsreport.sh directories...
```

For each command line argument the script should print out the current argument, a colon, a space and then one of the following:

- is empty
- is not empty
- is not a directory

For example, when executing the following:

```
$ ./samsreport.sh Desktop Documents .bashrc
```

The script will prompt the user for a name:

```
Who is this report for?
```

Then, if "Jim" was entered, the script should output:

```
$am's Report
 Desktop: is not empty
 Documents: is empty
 .bashrc: is not a directory
___generated for Jim___
```

Sam is very particular and has asked that the output match exactly as above including the dollar sign for the letter S in Sam. For example, the footer should have exactly three underscores, the footer message, and three more underscores without any added whitespace.

To make development easier, it is recommended to write and test your script in this order:

- Print out the usage if there aren't any arguments
- Prompt "Who is this report for? " and read a NAME
- Print the header "$am's Report"
- Print the footer "___generated for NAME___"

- Insert code between the header and footer to print out the body of the report. Ask your instructor for help if you need some hints.
- The script below should meet Sam's requirements:

```
#!/bin/bash
#
# ... Comments Omitted ...

PATH=/bin:/usr/bin

# Complain and quit when no arguments are specified:
if [ $# -lt 1 ] ; then
    echo "usage: $0 directories..."
    exit 1
fi

# Prompt the user for a name:
echo -n 'Who is this report for? '
read name

# Print the header for Sam's Report.  (Note the quoting of $.)
echo "\$am's Report"

# Process each argument:
for dir in "$@" ; do

    # Confirm the current item is a directory.
    if [ -d "$dir" ] ; then
        # Empty directories have only two files: . and ..
        if [ "$(ls -a $dir | wc -l)" -eq 2 ] ; then
            echo " $dir: is empty"
        else
            echo " $dir: is not empty"
        fi
    else
        echo " $dir: is not a directory"
    fi

done

# Print the footer for Sam's Report.  (Note the variable reference.)
echo "___generated for ${name}___"
exit 0
```

Practice Quiz

Text Processing Commands

Fill in the appropriate command for solving each problem.

1. To update an Apache directive in **`httpd.conf`** across multiple machines with semi-unique configurations, you could use the diff utility to create a patchfile
2. To search and replace or delete strings in a file, regardless of context, use sed
3. The patch command will take either a copied context (diff **-c**) or unified context (diff **-u**) file as input

4. To list all users in **/etc/passwd** that use the bash shell, run grep **'/bin/bash$' /etc/passwd**

5. To remove all comments from **/etc/rc.sysinit** you could use grep **-v '^#'** or sed **'s/^#.*$//'**

6. To sort all lines in **/etc/passwd** by their UID, run sort **-n -t: -k3 /etc/passwd**

7. To list all of the shells from **/etc/passwd**, run cut **-d: -f7 /etc/passwd**

8. The tool used to strip out and count duplicate lines of text is uniq **-c**

9. To list the number of times a shell is used in **/etc/passwd**, run cut **-d: -f7 /etc/passwd** | sort | uniq **-c**

10. To change all upper case letters to lower case you could use the tr command

11. To get just the first two lines of **ls -tl**, pipe the command to head **-2**

12. To grab the last line of **du -h**, pipe the command to tail **-1**

13. To remove the first line of **ls -al**, pipe the command to tail **-n+2**

14. To remove the footer from **du -h**, pipe the command to head **-n-1**

Key Defaults Defined in /etc/login.defs

Fill in the following values:

Variable	*Definition*	*Default Value*
PASS_MIN_DAYS	Minimum number of days that must pass between password changes	0 - No restriction
PASS_MAX_DAYS	Maximum number of days that must pass between password changes	99999 days = almost 274 years - No restriction
PASS_WARN_AGE	Number of days of warning before a password expires	7 days

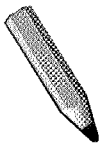
Table A.1. **/etc/login.defs** Definitions

Key Defaults Defined in /etc/default/useradd

Fill in the following values:

Variable	Definition	Default Value
INACTIVE	Number of days after a password expires when an account is permanently disabled	-1 - this restriction is disabled
EXPIRE	Date on which the account will be disabled, specified in the YYYY-MM-DD format	An empty string indicating no account expiration

Table A.2. **/etc/default/useradd** Definitions



Test

Criterion Test

Case Study

Scripting User Management Tasks

Perform the following steps on serverX unless directed otherwise.

Write a bash script that creates user accounts using data from a text file. Each line of the file contains four colon-separated fields that specify the information for a single user. The four fields represent the username, supplemental group membership, password, and maximum password age in days. Lines that start with "#" are ignored. Download a sample file containing the user list from **ftp://instructor.example.com/materials/user_list.txt**

Create any necessary supplemental groups before testing your script.

If your script does not work properly, then use text processing tools to extract the usernames from **user_list.txt** into a different file called **usernames.txt**. The following bash code will remove the user accounts so that you can test your program again:

```
#!/bin/bash
for u in $(cat usernames.txt)
do
    echo "Deleting $u"
    userdel -r $u
done
```

When you have completed the lab, copy your scripts to desktopX for future reference.

1. Download the file containing the user list: ftp://instructor.example.com/materials/user_list.txt

```
[root@serverX ~]# wget ftp://instructor.example.com/pub/materials/user_list.txt
--2010-12-21 22:59:48--  ftp://instructor.example.com/pub/materials/user_list.txt
           => "user_list.txt"
Resolving instructor.example.com... 192.168.0.254
Connecting to instructor.example.com|192.168.0.254|:21... connected.
```

```
... Output Omitted ...
```

2. Create any necessary supplemental groups before testing your script. Since **user_list.txt** refers to **staff** and **client** supplemental groups, create them:

```
[root@serverX ~]# groupadd staff
[root@serverX ~]# groupadd client
```

3. On serverX, write a bash script that uses a **for** loop to do the following for each user listed in the file:
   - Add the user to the system
   - Set the password
   - Set the maximum number of days between password changes
   - Add the user to the specified supplementary group

```
#!/bin/bash

# If the script was called with an argument, use it.  Otherwise, look for
# user_list.txt in the current working dir.
if [ -n "$1" ] ; then
    userfile=$1
else
    userfile=user_list.txt
fi

# Make sure the user file exists and is readable.
if [ ! -f $userfile -o ! -r $userfile ] ; then
    echo "error: unable to read $userfile"
    exit 1
fi

# Iterate over our list of users one at a time, ignoring lines that start
# with a '#' character.
for u in $(grep -v '^#' $userfile) ; do

     username="$(echo $u | cut -d: -f1)"
     password="$(echo $u | cut -d: -f2)"
     maxpwage="$(echo $u | cut -d: -f3)"
     supgroup="$(echo $u | cut -d: -f4)"

     # Create the user account.
     useradd -G $supgroup -K PASS_MAX_DAYS=$maxpwage $username

     # Set the account password.
     echo "$password" | passwd --stdin $username

done
```

   Read the note in the top of the user list file for information on the list format.

4. When your script is complete, run it and then check the following three files for the correct entries: **/etc/passwd**, **/etc/group**, and **/etc/shadow**.

```
[root@serverX ~]# getent passwd
```

```
[root@serverX ~]# getent group
[root@serverX ~]# getent shadow
```

5. If your script does not work properly, then use text processing tools to extract the usernames from **user_list.txt** into a different file called **usernames.txt**. The following bash code will remove the user accounts so that you can test your program again:

```
#!/bin/bash
for u in $(cat usernames.txt)
do
    echo "Deleting $u"
    userdel -r $u
done
```

File Security with GnuPG

gpg Options

Take some additional notes as you discuss these options:

1. Generate a key pair: **gpg --gen-key**
   Generates a new key pair for the user. You will be asked for: Kind of key (RSA/RSA is default); Key length in bits (longer is stronger); Expiration (in case of key compromise); Name, E-mail, Comment to identify the owner of the key; Passphrase (required to protect the private key so it can not be used if stolen).

2. List public keys: **gpg --list-keys**
   List the public keys the has: their own and any they have imported from other people they communicate with.

3. Export a public key: **gpg --export --armor *key-id* -o *file.key***
   Exports your public key into a file so that other people can have it. The **--armor** option puts the output in text rather than binary format. The ***key-id*** is the e-mail address *or* an eight hexadecimal digit number listed on the pub line from **--list-keys**.

4. Import a public key: **gpg --import *file.key***
   Imports another person's public key from a key file that has been sent to you.

5. Encrypt a file: **gpg --encrypt --armor -r *key-id file***
   Encrypts a message with the public key for *key-id*. If **-r *key-id*** is not given, the command will prompt for recipient. Default output file is *file.asc*.

6. Decrypt a file: **gpg --decrypt *file***
   Decrypts a message encrypted with the public key for one of your private keys.

Practice Quiz

Secure Files with GnuPG

1. Asymmetric (or public-key) encryption uses a pair of keys to encrypt/decrypt information. These keys are known as a public key (to be shared) and a private key (protected).

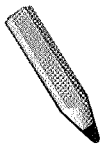
2. The **gpg** --gen-key command will create a key pair.

3. When creating a key pair you will specify a key type, the length of key in bits, an expiration, your name/e-mail information and a passphrase used to protect the private key.

4. The **gpg** --list-keys command displays the public keys that have been imported into your keyring.

5. **gpg --encrypt --armor myfile** will create an encrypted file named **myfile.asc**, after prompting for the intended recipient.

6. Using the sender's private key to encrypt a message is called a digital signature.



Test

Criterion Test

Performance Checklist

File Security with GnuPG

Partner with another learner to create an environment where you can encrypt files, share them, then decrypt them.

Perform these steps as **student** on your serverX machine.

☐ Create your own public/private key pair.

Note

You must have a graphical session available to successfully generate a GPG key. **gpg** now uses a graphical application to enter and validate the key.

```
[student@serverX ~]$ gpg --gen-key
gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc. This is free
 software: you are free to change and redistribute it. There is NO WARRANTY, to
 the extent permitted by law. Please select what kind of key you want: (1) RSA and
 RSA (default) (2) DSA and Elgamal (3) DSA (sign only) (4) RSA (sign only) Your
 selection?  Enter
RSA keys may be between 1024 and 4096 bits long. What keysize do you want?
 (2048)  Enter
Requested keysize is 2048 bits Please specify how long the key should be valid. 0
 = key does not expire <n> = key expires in n days <n>w = key expires in n weeks
 <n>m = key expires in n months <n>y = key expires in n years Key is valid for?
 (0)  Enter
Key does not expire at all Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key. Real name: My Name
Email address: student@serverX.example.com
Comment:  Enter
You selected this USER-ID: "My Name <student@serverX.example.com>" Change (N)ame,
 (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key. Enter passphrase
 Passphrase: testing123
Please re-enter this passpassphrase. Passphrase: testing123
We need to generate a lot of random bytes. It is a good idea to perform some other
 action (type on the keyboard, move the mouse, utilize the disks) during the prime
 generation; this gives the random number generator a better chance to gain enough
 entropy.

gpg: /home/student/.gnupg/trustdb.gpg: trustdb created
gpg: key 54AF5285 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
```

```
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
pub 2048R/54AF5285 2010-12-09
   Key fingerprint = 315F E90B 1745 2288 EBAE 4E7B 4BC6 4568 54AF 5285
uid    My Name <student@serverX.example.com>
sub    2048R/D08B2951 2010-12-09
```

To export the key, find the key ID from the output above. It can be found after the **pub 2048R/** output above. In this example, the key ID is **54AF5285**. The following examples will show the commands using this key ID. Your key ID will be different, so replace the example key ID with your key ID.

- ☐ Export your public key to share with your partner.

```
[student@serverX ~]$ gpg -a -o ~/KEY-serverX --export 54AF5285
```

- ☐ Copy your exported public key to your partner's serverX machine (perhaps use **scp**).

```
[student@serverX ~]$ scp ~/KEY-serverX serverY.example.com:~/
```

- ☐ Import your partner's public key.

```
[student@serverX ~]$ gpg --import ~/KEY-serverY
gpg: key DD7F4DFA: public key "Partner's Name <student@serverY.example.com>
 imported
gpg: Total number processed: 1
gpg:               imported: 1 (RSA: 1)
```

To use the key, find the key ID from the output above. It can be found after the **gpg: key** output above. In this example, the key ID is **DD7F4DFA**. The following examples will show the commands using this key ID. Your key ID will be different, so replace the example key ID with your key ID.

- ☐ Create a text file with a message for your partner to read.

```
[student@serverX ~]$ echo $(hostname) > ~/serverX.txt
```

- ☐ Encrypt the file with your partner's public key.

```
[student@serverX ~]$ gpg --encrypt --armor -r DD7F4DFA ~/serverX.txt
gpg: DD7F4DFA: There is no assurance this key belongs to the named user

pub  2048R/DD7F4DFA 2010-12-29 Foo Bar >student@serverY.example.com<
 Primary key fingerprint: 082A 55A4 CD5F E7C5 B627  94FD 606C F4B0 5B65 4D41
      Subkey fingerprint: 3C48 5B0D F655 2498 A83D  2E6E B805 043F 02E2 9705

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

- ☐ Exchange encrypted files with your partner (again, use **scp**).

```
[student@serverX ~]$ scp ~/serverX.txt.asc serverY.example.com:~/
```

- ☐ Decrypt the file encrypted by your partner and verify you can read the message they sent.

```
[student@serverX ~]$ gpg --decrypt ~/serverY.txt.asc
```

Package Management

Search & Learn: Yum Plugins

Consult the reference documents below to answer the following questions:

1. What is the purpose of the **yum-plugin-verify** package? Install it on serverX.
   Allows **yum** to check package integrity.

   ```
   [root@serverX ~]# yum -y install yum-plugin-verify
   ```

2. What new **yum** sub-commands does **yum-plugin-verify** introduce and what do the commands do?
   yum verify ***package_name***: Is the generic verification command, ignoring false matches due to multilib or changed configuration files.

   yum verify-rpm ***package_name***: 100% compatible with **rpm -V**.

   yum verify-all ***package_name***: Lists all the differences including some that **rpm** might normally ignore.

3. What is the purpose of the **yum-plugin-versionlock** package? Install it on serverX.
   Lists packages to lock to a specific version.

   ```
   [root@serverX ~]# yum -y install yum-plugin-versionlock
   ```

4. Which configuration file is used to lock down specific packages?
   By default, **/etc/yum/pluginconf.d/versionlock.list**

5. What is the format of that configuration file?
   With one package and version per line:

   epoch:name-version-release.arch

Practice Resequencing Exercise

Yum Plugins

For each of the four definitions below, write down the number of the matching file or command on the line in front of the definition.

3 This command lists all file changes from the original state of a package, including some that **rpm** would ignore.
4 This file contains packages that **yum** will refrain from updating (one package name per line).
5 This command is 100% compatible with **rpm -V** functionality.
2 This command verifies a package and ignores changes to configuration files.

1. **/etc/yum/pluginconf.d/versionlock.conf**

2. **yum verify** ***package***

3. **yum verify-all *package***
4. **/etc/yum/pluginconf.d/versionlock.list**
5. **yum verify-rpm *package***

Practice Quiz

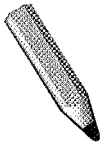
RPM Spec File

1. The package **Version** is usually derived from the open source project while the package **Release** is the packager's version.
2. The **Group** directive categorizes the type of package being built.
3. The name of the tarball containing the files used to build the package is specified with the **Source** directive.
4. The **BuildArch** directive specifies the target architecture the package is being built for. **noarch** will be its value when the package can be installed on any architecture.
5. The **Summary** directive specifies the 1-line description of a package while the **%description** section provides a more thorough explanation of what that package is for.
6. The **%install** section contains the code used to place files in the **$RPM_BUILD_ROOT** chroot directory structure.
7. The **%files** section defines which files and directories to package into the RPM.
8. The **%prep**, **%build**, and **%clean** sections contain shell code used to assemble a package and clean up after it has been built.

Practice Quiz

Create a Yum Repository

1. Install the **createrepo** package if necessary.
2. Create a directory that can be shared (via FTP or HTTP).
3. Create a subdirectory called Packages.
4. Copy all RPM packages to be published into Packages.
5. Execute **createrepo** on the top-level directory.



Test

Criterion Test

Performance Checklist

Part 1: Query with Yum Plugins

- ☐ Identify and install, if necessary, the **yum** plugin that will allow you to use **yum** to view a package changelog for a package that is not installed.

```
[student@serverX ~]$ su -
Password: redhat
[root@serverX ~]# yum -y install yum-plugin-changelog
```

- ☐ Use the plugin to view the changelog for the **zsh** package.

```
[root@serverX ~]# yum changelog all zsh
```

- ☐ Identify and install the **yum** plugin, if necessary, that will allow you to use **yum** to verify installed packages.

```
[root@serverX ~]# yum -y install yum-plugin-verify
```

- ☐ Use the plugin to verify the **initscripts** package. Show all changes.

```
[root@serverX ~]# yum verify-all initscripts
```

Performance Checklist

Part 2: Create an RPM

- ☐ Download the the file *ftp://instructor.example.com/pub/materials/hello.sh*.

```
[student@serverX ~]$ mkdir ~/hello-1.0
[student@serverX ~]$ cd ~/hello-1.0
[student@serverX hello-1.0]$ wget ftp://instructor.example.com/pub/materials/
hello.sh
```

- ☐ Create a simple RPM that installs **hello.sh** in **/root/bin**. Make sure that **hello.sh** is installed with a mode of 755.

```
[student@serverX hello-1.0]$ cd
[student@serverX ~]$ mkdir -p ~/rpmbuild/SOURCES
[student@serverX ~]$ mkdir -p ~/rpmbuild/SPECS
[student@serverX ~]$ tar -cvzf ~/rpmbuild/SOURCES/hello-1.0-1.tar.gz hello-1.0
```

~/rpmbuild/SPECS/hello.spec should look like:

```
Name:       hello
```

```
Version:    1.0
Release:    1
Summary:    Hello
Group:      RHCE
License:    GPL
URL:        http://www.redhat.com
Source0:    %{name}-%{version}-%{release}.tar.gz
BuildRoot:  /var/tmp/%{name}-buildroot

%description
Installs /root/bin/hello.sh

%prep
%setup -q -n %{name}-%{version}

%build

%install
rm -rf $RPM_BUILD_ROOT
mkdir -p $RPM_BUILD_ROOT/root/bin
install -m 755 hello.sh $RPM_BUILD_ROOT/root/bin/hello.sh

%clean
rm -rf $RPM_BUILD_ROOT

%files
%defattr(-,root,root,-)
/root/bin/hello.sh

%changelog
```

```
[student@serverX ~]$ su -
Password: redhat
[root@serverX ~]# yum install -y rpm-build
[root@serverX ~]# exit
[student@serverX ~]$ rpmbuild -ba ~/rpmbuild/SPECS/hello.spec
```

☐ Create a GPG key and sign the package with the key. Export the public GPG key.

Note

You must have a graphical session available to successfully generate a GPG key. **gpg** now uses a graphical application to enter and validate the key.

```
[student@serverX ~]$ gpg --gen-key
gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc. This is free
 software: you are free to change and redistribute it. There is NO WARRANTY, to
 the extent permitted by law. Please select what kind of key you want: (1) RSA and
 RSA (default) (2) DSA and Elgamal (3) DSA (sign only) (4) RSA (sign only) Your
 selection?   Enter
RSA keys may be between 1024 and 4096 bits long. What keysize do you want?
 (2048)  Enter
Requested keysize is 2048 bits Please specify how long the key should be valid. 0
 = key does not expire <n> = key expires in n days <n>w = key expires in n weeks
 <n>m = key expires in n months <n>y = key expires in n years Key is valid for?
 (0)  Enter
```

```
Key does not expire at all Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key. Real name: My Name
Email address: student@serverX.example.com
Comment:  Enter
You selected this USER-ID: "My Name <student@serverX.example.com>" Change (N)ame,
 (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key. Enter passphrase
 Passphrase: testing123
Please re-enter this passpassphrase. Passphrase: testing123
We need to generate a lot of random bytes. It is a good idea to perform some other
 action (type on the keyboard, move the mouse, utilize the disks) during the prime
 generation; this gives the random number generator a better chance to gain enough
 entropy.

gpg: /home/student/.gnupg/trustdb.gpg: trustdb created
gpg: key 54AF5285 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
pub 2048R/54AF5285 2010-12-09
   Key fingerprint = 315F E90B 1745 2288 EBAE 4E7B 4BC6 4568 54AF 5285
uid    My Name <student@serverX.example.com>
sub    2048R/D08B2951 2010-12-09
```

To export the key, find the key ID from the output above. It can be found after the **pub 2048R/** output above. In this example, the key ID is **54AF5285.** The following examples will show the commands using this key ID. Your key ID will be different, so replace the example key ID with your key ID.

```
[student@serverX ~]$ gpg -a -o ~/RPM-GPG-KEY-student --export 54AF5285
```

Create the **~/.rpmmacros** file and add the following content:

```
%_gpg_name 54AF5285
```

Sign the RPM package

```
[student@serverX ~]$ rpm --resign ~/rpmbuild/RPMS/x86_64/hello-1.0-1.x86_64.rpm
Enter pass phrase: testing123
Pass phrase is good. /home/instructor/rpmbuild/RPMS/x86_64/hello-1.0-1.x86_64.rpm:
```

- ☐ Deploy a web server and create a **yum** repository in **/var/www/html/Packages/**. Create a repository file that references *http://serverX/Packages*. Serve the GPG key from the web server and include the key in the repository file.

```
[root@serverX ~]# mkdir /var/www/html/Packages
[root@serverX ~]# cp ~student/rpmbuild/RPMS/x86_64/hello-1.0*.rpm /var/www/html/
Packages/
[root@serverX ~]# cp ~student/RPM-GPG-KEY-student /var/www/html/Packages/
[root@serverX ~]# createrepo -v /var/www/html/Packages/
[root@serverX ~]# service httpd start
```

Create the **/etc/yum.repos.d/hello.repo** file with the following content:

```
[hello]
name=hello
description=ServerX Yum Repo
baseurl=http://serverX.example.com/Packages
enabled=1
gpgcheck=1
gpgkey=http://serverX.example.com/Packages/RPM-GPG-KEY-student
```

- ☐ Install your RPM package using the **yum** repository above and run **/root/bin/hello.sh**.

```
[root@serverX ~]# yum -y install hello
[root@serverX ~]# hello.sh
```

Network Monitoring

Fill in the below **netstat** options for each description:

- **-t** - tcp protocol or open TCP ports
- **-u** - udp protocol or open UDP ports
- **-l** - listening ports only, not active, established connections
- **-n** - display hosts and ports by number, not name
- **-p** - display which local process controls a port

Fill in the below **nmap** options for each description:

- **-sP** - performs a ping scan to identify hosts that respond to an ICMP ping request
- **-sT** - performs a TCP connect scan
- **-sU** - performs a UDP scan (which can take a very long time)
- **-p** - limit to specific ports, which is particularly useful for UDP scans
- **-A** - enables OS detection and Version detection, Script scanning and Traceroute
- **-v** - verbose (use multiple **-v**'s to make it more verbose)

Practice Performance Checklist

Closing Open Ports

Examine how the **Avahi** service presents itself on serverX.

☐ Log into serverX as root.

☐ Locally detect which ports the **Avahi** service is listening on. What command, with options, did you use to identify the open ports and which ports **Avahi** is listening on?

```
[root@serverX ~]# netstat -tulnp | grep avahi
udp        0      0 0.0.0.0:5353                0.0.0.0:*
     1371/avahi-daemon:
udp        0      0 0.0.0.0:45620               0.0.0.0:*
     1371/avahi-daemon:
```

☐ Log into another window as root on desktopX.

☐ Scan the specific ports you identified in step 2 from desktopX using **nmap**. You should see the open **avahi-daemon** ports.

```
[root@desktopX ~]# nmap -sU -p 5353,45620 serverX

Starting Nmap 5.21 ( http://nmap.org ) at 2011-01-17 14:42 EST
```

```
Nmap scan report for server1 (192.168.0.101)
Host is up (0.00058s latency).
rDNS record for 192.168.0.101: server1.example.com
PORT      STATE         SERVICE
5353/udp  open|filtered zeroconf
45620/udp open|filtered unknown
MAC Address: 52:54:00:00:00:01 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.34 seconds
```

- ☐ Disable the **Avahi** service immediately and persistently on serverX.

```
[root@serverX ~]# service avahi-daemon stop
Shutting down Avahi daemon:                                [  OK  ]
[root@serverX ~]# chkconfig avahi-daemon off
```

- ☐ Redetect the open local ports on serverX. Does **avahi-daemon** have any open ports?

```
[root@serverX ~]# netstat -tulnp | grep avahi
```

- ☐ Rescan the specific ports from desktopX and confirm the **avahi-daemon** ports are unavailable.

```
[root@desktopX ~]# nmap -sU -p 5353,45620 serverX

Starting Nmap 5.21 ( http://nmap.org ) at 2011-01-17 14:47 EST
Nmap scan report for server1 (192.168.0.101)
Host is up (0.00064s latency).
rDNS record for 192.168.0.101: server1.example.com
PORT      STATE  SERVICE
5353/udp  closed zeroconf
45620/udp closed unknown
MAC Address: 52:54:00:00:00:01 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

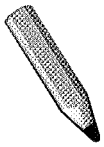
Practice Quiz

Packet Inspection

1. The **tcpdump** command is used on systems without X to capture network traffic to a file. The file can be copied to a system with the **wireshark** utility for further analysis.
2. **tcpdump -D** lists the available interfaces to capture network traffic from. The **-i** option is used to select the interface to use when capturing packets.
3. The ***-s size*** option to **tcpdump** specifies the maximum number of bytes per packet to output.
4. **tcpdump** creates a file with the captured packets when the ***-w filename*** option is specified.
5. A logical expression with keywords is the argument to **tcpdump** that more precisely specifies which network traffic to capture.

6. Files containing packet data can be analyzed graphically as an unprivileged user using **wireshark**. This utility is provided by the **wireshark-gnome** package.

7. The capture file to be analyzed can either be specified as an argument or opened within **wireshark** using the **File → Open** pull-down menu item.



Test

Criterion Test

Performance Checklist

Monitoring the HTTP Service

In this practice exercise, you will capture and analyze web traffic going between desktopX and serverX.

- ☐ Log in to serverX as root.

```
[student@serverX ~]$ su -
Password: redhat
```

- ☐ If necessary, start the **httpd** service.

```
[root@serverX ~]# service httpd start
```

- ☐ Determine which ports are listening for connections. What is the status of port 80?

```
[root@serverX ~]# netstat -tulnp | grep ':80'
tcp        0      0 :::80                       :::*                        LISTEN
     2646/httpd
```

- ☐ Log into desktopX and externally scan the open ports of serverX. Is port 80 open?

  If necessary, install the **nmap** package.

```
[root@desktopX ~]# yum -y install nmap
```

  Perform the scan.

```
[root@desktopX ~]# nmap -sT serverX | grep '80.*open'
80/tcp  open  http
```

- ☐ Use **tcpdump** on serverX to capture incoming HTTP connections to a capture file.

```
[root@serverX ~]# tcpdump -nn -l -s 2048 -w /tmp/http.dump -i eth0 'port 80' &
```

- ☐ While **tcpdump** is capturing packets on serverX, launch a web browser on desktopX and browse to http://serverX.example.com.

```
[student@desktopX ~]$ firefox http://serverX.example.com &
```

- ☐ Once you have browsed the site, stop **tcpdump** on serverX.

```
[root@serverX ~]# killall tcpdump
```

- ☐ Copy the capture file to desktopX.

```
[root@serverX ~]# rsync /tmp/http.dump student@desktopX:
```

- ☐ Use **wireshark** on desktopX to open the capture file for analysis.

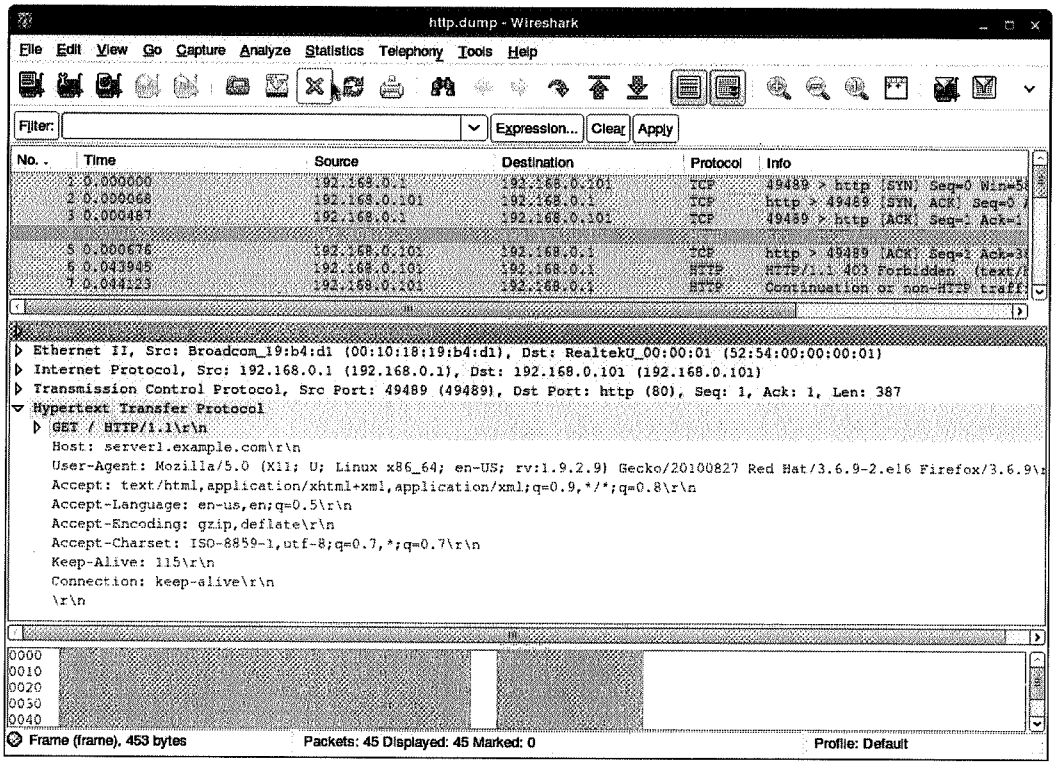
  If necessary, install the **wireshark-gnome** package as root:

```
[root@desktopX ~]# yum -y install wireshark-gnome
```

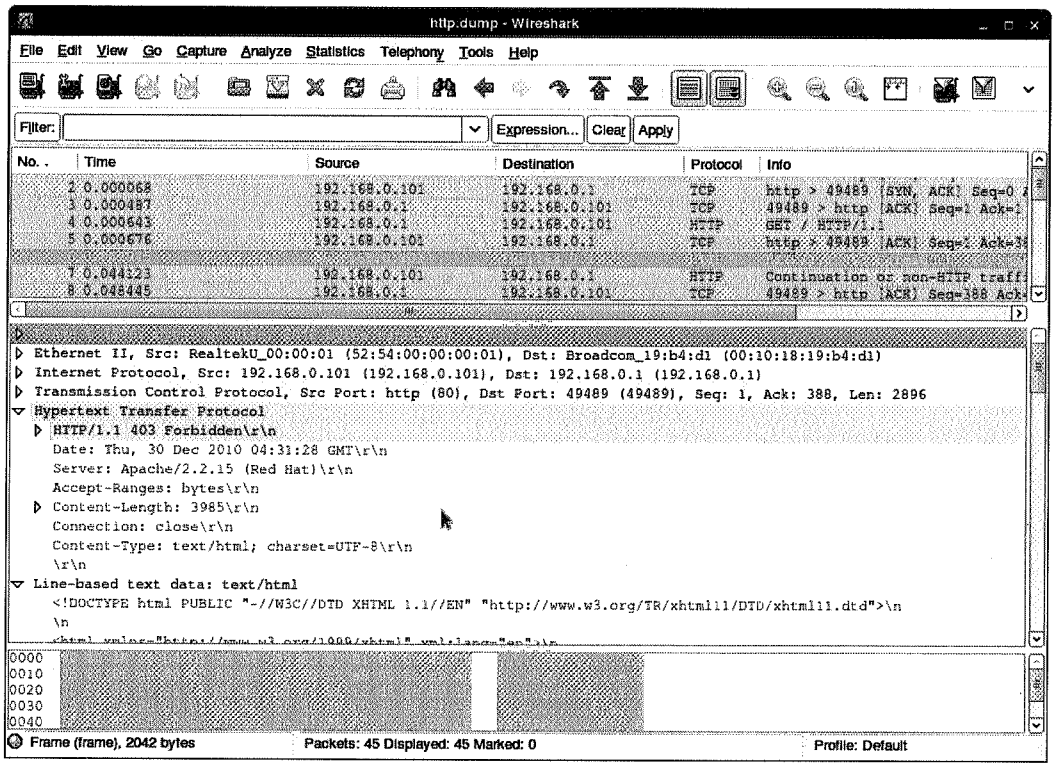
  Perform the analysis as student.

```
[student@desktopX ~]$ wireshark ~/http.dump
```

- ☐ Find these in the captured traffic:
  - a packet from desktopX to serverX requesting web content



  - the packet(s) from serverX to desktopX that fulfill the HTTP request


http.dump - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply
No. . Time Source Destination Protocol Info
2 0.000068 192.168.0.101 192.168.0.1 TCP http > 49489 [SYN, ACK] Seq=0
3 0.000487 192.168.0.1 192.168.0.101 TCP 49489 > http [ACK] Seq=1 Ack=1
4 0.000643 192.168.0.1 192.168.0.101 HTTP GET / HTTP/1.1
5 0.000676 192.168.0.101 192.168.0.1 TCP http > 49489 [ACK] Seq=1 Ack=3
7 0.044123 192.168.0.101 192.168.0.1 HTTP Continuation or non-HTTP traff
8 0.048445 192.168.0.1 192.168.0.101 TCP 49489 > http [ACK] Seq=388 Ack=
Ethernet II, Src: RealtekU_00:00:01 (52:54:00:00:00:01), Dst: Broadcom_19:b4:d1 (00:10:18:19:b4:d1)
Internet Protocol, Src: 192.168.0.101 (192.168.0.101), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: 49489 (49489), Seq: 1, Ack: 388, Len: 2896
Hypertext Transfer Protocol
HTTP/1.1 403 Forbidden\r\n
Date: Thu, 30 Dec 2010 04:31:28 GMT\r\n
Server: Apache/2.2.15 (Red Hat)\r\n
Accept-Ranges: bytes\r\n
Content-Length: 3985\r\n
Connection: close\r\n
Content-Type: text/html; charset=UTF-8\r\n
\r\n
Line-based text data: text/html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">\n
\n
0000
0010
0020
0030
0040
Frame (frame), 2042 bytes
Packets: 45 Displayed: 45 Marked: 0
Profile: Default

Advanced Network Configuration

Practice Quiz

Network Bonding Configuration

1. Which mode of Linux Ethernet bonding primarily uses one slave interface and changes interface upon failure?

   (select one of the following...)
   a. Mode 0 (balance-rr)
   b. Mode 1 (active-backup)
   c. Mode 3 (broadcast)

2. Which mode of Linux Ethernet bonding uses all interfaces in a round robin fashion to achieve more throughput?

   (select one of the following...)
   a. Mode 0 (balance-rr)
   b. Mode 1 (active-backup)
   c. Mode 3 (broadcast)

3. When creating a bonded network interface, which configuration file contains the IP address and netmask definitions for the interface?

   (select one of the following...)
   a. **`/etc/sysconfig/network`**
   b. **`/etc/sysconfig/network-scripts/ifcfg-bond0`**
   c. **`/etc/sysconfig/network-scripts/ifcfg-`*`iface`***
   d. None of the above

4. When creating a bonded network interface, which configuration file defines the type of bonding?

   (select one of the following...)
   a. **`/etc/sysconfig/network`**
   b. **`/etc/sysconfig/network-scripts/ifcfg-bond0`**
   c. **`/etc/sysconfig/network-scripts/ifcfg-`*`iface`***
   d. None of the above

5. When creating a bonded network interface, which variable definitions must be specified in the **`/etc/sysconfig/network-scripts/ifcfg-`*`iface`*** configuration file?

   (select one of the following...)
   a. **`GATEWAY`**
   b. **`IPADDR`**
   c. **`MASTER`**

d. None of the above

Search & Learn: Kernel Tuning

1. Perform the following steps on serverX.

2. Install the **kernel-doc** RPM if it is not already installed.

```
[root@serverX ~]# yum -y install kernel-doc
```

3. How would you use **sysctl** to identify kernel parameters that control ping, or ICMP echo, behavior?

```
[root@serverX ~]# sysctl -a | grep icmp
```

4. Which parameters look promising?

   net.ipv4.icmp_echo_ignore_all or
   net.ipv4.icmp_echo_ignore_broadcasts

5. What command would you use to identify and/or examine kernel documentation that describes what those parameters are for?

```
[root@serverX ~]# grep -A5 icmp /usr/share/doc/kernel-doc-*/Documentation/
networking/ip-sysctl.txt
```

6. How would you use **sysctl** to adjust kernel parameters to "hide" your system from ping requests?

```
[root@serverX ~]# sysctl -w net.ipv4.icmp_echo_ignore_all=1
```

7. How would you configure **sysctl** to persistently adjust kernel parameters to survive a reboot?

```
[root@serverX ~]# echo "net.ipv4.icmp_echo_ignore_all = 1" >> /etc/
sysctl.conf
```

Practice Performance Checklist

Enable Ping Broadcast

The default configuration for Red Hat Enterprise Linux 6 configures the kernel to ignore ping broadcast requests. You will work with a partner to tune the kernel on serverX to respond to them instead.

- ☐ Find a partner to work with. If there is an odd number of students, a group of three will work.
- ☐ Send a broadcast ping to the 192.168.0.0/24 network. Note which hosts respond to the ping request.

```
[root@serverX ~]# ping -b 192.168.0.255
```

- ☐ Tune serverX so that it will respond to ping broadcasts.

```
[root@serverX ~]# sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0
```

- ☐ Send another broadcast ping to the 192.168.0.0/24 network. Did your hosts respond?

```
[root@serverX ~]# ping -b 192.168.0.255
```

- ☐ Persistently configure your serverX machines to respond to ping broadcasts and reboot.

```
[root@serverX ~]# echo "net.ipv4.icmp_echo_ignore_broadcasts = 0" >> /etc/
sysctl.conf
[root@serverX ~]# reboot
```

- ☐ Send another ping broadcast. Did your configuration changes persist the reboot?

```
[root@serverX ~]# ping -b 192.168.0.255
```

Practice Performance Checklist

Static Route Configuration

Get back together with your partner(s). One of you will work on your system called desktopX and the other will be called desktopY for this lab. The virtual machines running on them will be serverX and serverY respectively.

- ☐ Log in to desktopX and desktopY and run the **lab-setup-routing** script on both systems.

```
[root@desktopX ~]# lab-setup-routing
```

- ☐ When the virtual machines finish rebooting, determine the IP addresses of serverX and serverY.

```
[root@serverX ~]# ip a show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
 1000
    link/ether 52:54:00:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet 10.X.0.149/24 brd 10.1.0.255 scope global eth0
    inet6 fe80::5054:ff:fe00:1/64 scope link
       valid_lft forever preferred_lft forever
```

- ☐ Tune desktopX and desktopY to support IP forwarding for IPv4 if it isn't configured already.

```
[root@desktopX ~]# cat /proc/sys/net/ipv4/ip_forward
```

```
1
```

- ☐ Interactively configure a route on desktopX to reach the 10.Y.0.0/24 network through desktopY. Perform the following tests and note your results:

  - Can desktopX ping serverY?
  - Can serverX ping serverY?
  - Can desktopY ping serverX?
  - Can serverY ping desktopX?

  ```
  [root@desktopX ~]# ip route add 10.Y.0.0/24 via 192.168.0.Y
  ```

  Yes, No, No, Yes

- ☐ Interactively configure a route on desktopY to reach the 10.X.0.0/24 network through desktopX. Repeat the following tests and note your results:

  - Can desktopX ping serverY?
  - Can serverX ping serverY?
  - Can desktopY ping serverX?
  - Can serverY ping desktopX?

  ```
  [root@desktopY ~]# ip route add 10.X.0.0/24 via 192.168.0.X
  ```

  Yes, Yes, Yes, Yes

- ☐ Persistently configure desktopX and desktopY to implement the static routes needed for serverX and serverY to communicate with each other. Reboot desktopX and desktopY.

  On each system, desktopX and desktopY, create a file named **/etc/sysconfig/network-scripts/route-virbr0** containing the following:

  On desktopX:

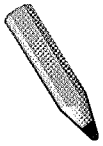
  ```
  ADDRESS0=10.Y.0.0
  NETMASK0=24
  GATEWAY0=192.168.0.Y
  ```

  On desktopY:

  ```
  ADDRESS0=10.X.0.0
  NETMASK0=24
  GATEWAY0=192.168.0.X
  ```

- ☐ Confirm that serverX and serverY can ping each other. If they cannot ping each other, then diagnose the routing issue and correct it.

- ☐ Execute the **lab-cleanup-routing** script on desktopX and desktopY to restore the virtual machines back onto the classroom network.



Test

Criterion Test

Case Study

Routing Network Traffic

Before you begin...

Run the **lab-setup-oshu** script on desktopX to prepare serverX for the exercise.

Operation Strategic Holistic Unusual (or OSHU), is an online chat system for fans of conspiracy fiction. In order to join the site they have two requirements, listed below.

1. To fulfill the first requirement, you must prove your ability to "disappear" a server. You will do this by modifying the configuration on serverX so that it does not respond to any ping requests. Make this change persistent so that it will still be in effect after a reboot.
2. The second requirement is to join the "secret" OSHU network. To join the network, add an additional IP address to serverX, where X is your desktop/server number:

   10.42.10.X/24

When you have fulfilled the requirements, run **lab-grade-oshu** on desktopX to check your work.

1.
```
[root@desktopX ~]# lab-setup-oshu
```

2. Add the following to **/etc/sysctl.conf**

```
net.ipv4.icmp_echo_ignore_all = 1
```

3. Enable the setting

```
[root@serverX ~] sysctl -p
```

4. Configure static network settings

```
[root@serverX ~] service NetworkManager stop
[root@serverX ~] chkconfig NetworkManager off
```

5. Create the **/etc/sysconfig/network-scripts/ifcfg-eth0:0** file and add the following content:

```
DEVICE=eth0:0
IPADDR=10.42.10.X
NETMASK=255.255.255.0
```

```
ONPARENT=yes
```

6. Enable the new network settings

```
[root@serverX ~] ifup eth0:0
```

Secure Network Traffic

Practice Exercise

Use SSH Port Forwarding

Before you begin...

Login as **student** on serverX and use **nc** to verify an SMTP service is accepting connections from localhost (i.e., **nc localhost 25** on serverX). You may have to install the **nc** package.

```
[student@serverX ~]$ su -
[root@serverX ~]# yum install -y nc
[root@serverX ~]# exit
[student@serverX ~]$ nc localhost 25
220 serverX.example.com ESMTP Postfix
 Ctrl+c
```

Your desktopX and serverX machines are running a local SMTP service that accepts connections only from localhost. In this exercise you will use **ssh** to connect to serverX's SMTP server from desktopX via a secure encrypted tunnel.

1. Login as **student** on desktopX.

```
[student@desktopX ~]$
```

2. Use **nc** to verify an SMTP service is running on desktopX.

```
[student@desktopX ~]$ su -
[root@desktopX ~]# yum install -y nc
[root@desktopX ~]# exit
[student@desktopX ~]$ nc localhost 25
220 desktopX.example.com ESMTP Postfix
 Ctrl+c
```

3. Use **nc** to verify that the SMTP service does not allow connections from desktopX to serverX.

```
[student@desktopX ~]$ nc serverX 25
[student@desktopX ~]$
```

4. Run **ssh** to forward desktopX port 2025 to port 25 on serverX through a tunnel.

```
[student@desktopX ~]$ ssh -L 2025:localhost:25 serverX
student@serverX's password: student
[student@serverX ~]$
```

Note that our ssh command simply opens an **ssh** connection as normal. As soon as you **exit** the shell, the **ssh** tunnel will be destroyed. It would probably be better to use the **-N** and **-f** options to **ssh**:

```
[student@serverX ~]$ exit
[student@desktopX ~]$ ssh -Nf -L 2025:localhost:25 serverX
```

```
student@serverX's password: student
[student@desktopX ~]$
```

5. Open another window on desktopX and use **nc** to verify desktopX port 2025 connects to serverX's SMTP service. Note the name of the host displayed in the SMTP welcome banner.

```
[student@desktopX ~]$ nc localhost 2025
220 serverX.example.com ESMTP Postfix
 Ctrl+c
[student@desktopX ~]$
```

iptables Basics

- Rule - criteria determining which packets to match and a target, or action, determining what to do with those packets.
- Chain - a list of *rules* which will be checked in order, first match takes effect.
- Policy - the default action, **ACCEPT** or **DROP**, taken if no *rule* matches in a built-in *chain*.
- Table - a set of *chains* used for a particular purpose: **filter** to block traffic, **nat** to modify the destination or apparent source of a packet.

Built-in Chains (filter table)

- INPUT - packets addressed to the firewall
- OUTPUT - packets originating from a service on the firewall (not forwarded)
- FORWARD - packets originating from another machine, that are not addressed to the firewall but are being forwarded (routed) elsewhere (when **net.ipv4.ip_forward=1**)

Targets

(Actions to take when packets match rules)

- ACCEPT - the packet passes the chain
- DROP - the packet is dropped as if it was never seen
- REJECT - the packet is rejected, and the firewall sends an error message (an ICMP port unreachable message by default)
- LOG - information about the packet is logged to syslog; we go on to the next rule in the chain

Practice Exercise

Implement a Firewall

In this exercise you will implement a firewall on serverX that rejects all packets, except that it will allow ICMP traffic for example.com and allow SSH for everyone.

1. Log into serverX as **root** using **virt-viewer** or **virt-manager**.

2. Create a simple deny all (except loop back) firewall by creating **/root/bin/resetfw.sh** that:

   1. sets the **INPUT** chain's default policy to **DROP**,
   2. flushes all rules in the filter table, and
   3. will **ACCEPT** all packets from the loopback interface

```
[root@serverX ~]# cat /root/bin/resetfw.sh
#!/bin/bash
# Set INPUT chain default policy to DROP
iptables -P INPUT DROP
# Flushes all rule in the filter table
iptables -F
# Will ACCEPT all packets from loopback interface
iptables -A INPUT -i lo -j ACCEPT
```

3. Run your script and record the results of the following:

   - **ping** and **ssh** serverX from desktopX and from remoteX.remote.test

   Both should fail since only traffic from serverX's loopback interface is being ACCEPTed.

4. What happens when you **ping** desktopX and 192.168.0.X from serverX now? Why?

   Again, both should fail since the replies from desktopX and 192.168.0.X are being DROPped.

5. Enable stateful firewalling by appending to your script a rule that will

   - **ACCEPT** all **ESTABLISHED, RELATED** packets

```
[root@serverX ~]# tail -n 2 /root/bin/resetfw.sh
# ACCEPT all ESTABLISHED, RELATED packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED
```

6. Run your script and record the results of the following:

   - **ping** desktopX and 192.168.0.X from serverX

   These should now work since the "replies" are part of an ESTABLISHED connection. Note, that the reverse is still not possible because the connection never gets started.

7. Reject all packets from remote.test by appending to your script a rule that will

   - **REJECT** all packets from the 192.168.1.0/24 network

```
[root@serverX ~]# tail -n 2 /root/bin/resetfw.sh
# REJECT all packets from 192.168.1.0/24 network
iptables -A INPUT -s 192.168.1.0/24 -j REJECT
```

8. Run your script and record the results of the following:

   - **ping** and **ssh** serverX from desktopX and from remoteX.remote.test

These should still not work from desktopX (being DROPped) since there are no applicable rules for the initial inbound packet. The attempts from remoteX.remote.test are being explicitly REJECTed and so will also not work.

9. Enable ICMP traffic for example.com by appending to your script a rule that will

   - **ACCEPT** all **icmp** traffic from 192.168.0.0/24

```
[root@serverX ~]# tail -n 2 /root/bin/resetfw.sh
# ACCEPT all icmp traffic from 192.168.0.0/24
iptables -A INPUT -p icmp -s 192.168.0.0/24 -j ACCEPT
```

10. Run your script and record the results of the following:

    - **ping** and **ssh** serverX from desktopX

    The **ping** should now work since the inbound **icmp** protocol is now ACCEPTed. However, **ssh** still does not work.

11. Enable SSH traffic for all hosts by modifying your script to

    - **ACCEPT** all **NEW** connections to **tcp** port **22**

    Notice that the direction did NOT say to append the new rule. For this to ACCEPT all NEW connections, we need to insert this into the script *above* the REJECT of 192.168.1.0/24.

```
[root@serverX ~]# cat /root/bin/resetfw.sh
#!/bin/bash
# Set INPUT chain default policy to DROP
iptables -P INPUT DROP
# Flushes all rule in the filter table
iptables -F
# Will ACCEPT all packets from loopback interface
iptables -A INPUT -i lo -j ACCEPT
# ACCEPT all ESTABLISHED, RELATED packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED
# ACCEPT all NEW connections to tcp port 22
iptables -A INPUT -m state --state NEW -p tcp --dport 22 -j ACCEPT
# REJECT all packets from 192.168.1.0/24 network
iptables -A INPUT -s 192.168.1.0/24 -j REJECT
# ACCEPT all icmp traffic from 192.168.0.0/24
iptables -A INPUT -p icmp -s 192.168.0.0/24 -j ACCEPT
```

12. Run your script and record the results of the following:

    - **ssh** to serverX from desktopX and from remoteX.remote.test

    These should now work since the "requests" are a NEW connection on port 22. If only desktopX works and remoteX.remote.test does not work, then check the order of your rules in the previous step.

13. Reject packets by default instead of dropping packets by appending to your script a rule that will:

    - **REJECT** all other traffic

```
[root@serverX ~]# tail -n 2 /root/bin/resetfw.sh
# REJECT all other traffic
iptables -A INPUT -j REJECT
```

14. Run your script and record the results of the following:

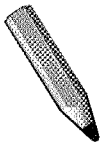
    - **ping** and **ssh** serverX from desktopX and from remoteX.remote.test

    Both **ping** and **ssh** will work from desktopX, but only **ssh** works from remoteX.remote.test.

Practice Quiz

Network Address Translation

1. The chains available in the **filter** table are **INPUT**, **FORWARD**, and **OUTPUT**
2. The chains available in the **nat** table are **PREROUTING**, **POSTROUTING**, and **OUTPUT**
3. **iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE**
4. **iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.0.1**
5. **iptables -t nat -A PREROUTING -i eth0 -m tcp -p tcp --dport 80 -j DNAT --to-destination 192.168.0.100:8080**
6. The **DNAT** target can only be used in the **PREROUTING** chain and the **OUTPUT** chain of the **nat** table
7. To enable forwarding persistently across reboots add **net.ipv4.ip_forward=1** to **/etc/sysctl.conf** and run **sysctl -p**



Test

Criterion Test

Case Study

The Morris Worm and Fish Supply Company

Before you begin...

Run **lab-setup-morrisworm** on desktopX to prepare serverX for the exercise.

The Morris Worm and Fish Supply company is finally looking to modernize its business by opening a website. The web server will run on a private network behind a firewall. The firewall will forward all TCP port 80 traffic to the web server and will perform NAT so that the web server can reach external hosts.

- desktopX.example.com will be the firewall, serverX.example.com will be the web server.

- Configure Apache to run on serverX.example.com. Put some custom content in **/var/www/html/index.html** that will uniquely identify the server.
- Configure the firewall on desktopX to perform NAT that will allow the web server to reach the outside network. You will be able to successfully **ping** instructor.example.com from serverX to confirm this works.
- Finally configure the firewall to forward all TCP port 80 traffic sent to it to the web server running on serverX. You will need to identify serverX's IP address to complete this step. Confirm this works by using a web browser from an external machine, NOT desktopX, to browse http://desktopX.example.com.

After you have successfully completed the lab, run **lab-cleanup-morrisworm** on desktopX to reset your network back to its original state.

1. Determine the IP address of serverX

```
[root@serverX ~]# ip a show eth0
```

2. Deploy a web server on serverX

```
[root@serverX ~]# yum install httpd
[root@serverX ~]# service httpd start ; chkconfig httpd on
[root@serverX ~]# echo serverX > /var/www/html/index.html
```

3. On desktopX set **net.ipv4.ip_forward=1**

```
[root@desktopX ~]# sysctl -w net.ipv4.ip_forward=1
```

4. On desktopX add these rules to the firewall (replace 192.168.122.*Z* with the IP found in the first step.

```
[root@desktopX ~]# NewServerIP=192.168.122.Z
[root@desktopX ~]# iptables -t nat -A POSTROUTING -s ${NewServerIP} -j MASQUERADE
[root@desktopX ~]# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-
destination ${NewServerIP}
```

NTP Server Configuration

Practice Quiz

Configuring NTP

Answer the questions below based upon the following NTP configuration file:

```
#/etc/ntp.conf

restrict default kod nomodify notrap nopeer noquery
restrict -6 default ignore

restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap nopeer
restrict 192.168.0.101 kod nomodify notrap
restrict 192.168.0.200

server 192.168.0.2
server 192.168.0.3
peer 192.168.0.101
```

1. The NTP client's time is off by 15 minutes, it will eventually sync with the servers.

   (select one of the following...)
   a. True
   b. False

2. The NTP client will use the computer's RTC (BIOS) as a time source.

   (select one of the following...)
   a. True
   b. False

3. 192.168.0.200 will be able to modify the time on this NTP server.

   (select one of the following...)
   a. True
   b. False

4. 192.168.0.4 will be able to query this NTP server.

   (select one of the following...)
   a. True
   b. False

5. 192.168.0.3 will be able to use this NTP server as a peer.

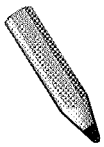
   (select one of the following...)
   a. True
   b. False

6. Anyone with an IPv4 address will be able use this NTP server as a time source.

   (select one of the following...)
   a. True
   b. False

7. Anyone with an IPv6 address will be able use this NTP server as a time source.

   (select one of the following...)
   a. True
   b. False



Test

Criterion Test

Case Study

NTP Server Configuration

Before you begin...

Run **lab-setup-howsonclock** on desktopX.

Howson Heavy Machine and Clock Manufacture, maker of clock tower parts and accessories, recently conducted an audit of all computer systems. The audit revealed several systems with out of sync clocks, including your serverX.example.com machine.

Set up NTP on your serverX to be a client of the NTP service running on instructor.example.com

In order to have additional time sources, work with a few neighbors so that all of your serverX systems are set up to synchronize as NTP peers

When you have finished, run **lab-grade-howsonclock** on serverX to check your work.

1. Identify two or three peer machines. In class, you are encouraged to peer with a neighbor, although these instructions will instead use your 3 assigned machines as peers: desktopX and hostX for the server running on serverX.

   Create the following **/etc/ntp.conf** configuration file, and distribute it to all 3 machines.

```
driftfile /var/lib/ntp/drift

restrict default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict 192.168.0.0 mask 255.255.255.0

server instructor.example.com
peer desktopX.example.com
peer serverX.example.com
peer hostX.example.com
```

On each of the machines, edit the file to omit itself from the peer list. For example, on desktopX, remove the peer line for desktopX.example.com

2. On each of the machines, enable and start the ntp service.

```
[root@serverX ~]# service ntpd restart
[root@serverX ~]# ssh desktopX service ntpd restart
[root@serverX ~]# ssh hostX service ntpd restart
[root@serverX ~]# chkconfig ntpd on
[root@serverX ~]# ssh desktopX chkconfig ntpd on
[root@serverX ~]# ssh hostX chkconfig ntpd on
```

3. For each of the machines, use the **ntpq -p** command to monitor the NTP service's interactions with its peers. For the first 5-10 minutes, you should see output similar to the following.

```
[root@serverX ~]# ntpq -p
remote           refid           st t when poll reach   delay   offset  jitter
==============================================================================
*instructor.exam LOCAL(0)        11 u   64   64   17    0.533   -0.096   0.223
 desktopX.exampl .INIT.          16 u   19   64    0    0.000    0.000   0.000
 hostX.example.c .INIT.          16 u   48   64    0    0.000    0.000   0.000
```

Hint: the **watch** command can be useful for monitoring the peering process. In each of three terminals, shell to each of the three machines, and run the command **watch -d ntpq -p**. Use **CTRL-C** to cancel the **watch** command when done.

4. After 5-10 minutes, the NTP services should peer, as reflected by the output of the **ntpq -p** command.

```
[root@serverX ~]# ntpq -p
remote           refid           st t when poll reach   delay   offset  jitter
==============================================================================
*instructor.exam LOCAL(0)        11 u   61   64   77    0.327    0.083   0.028
+desktopX.exampl 192.168.0.254   12 u    6   64   77    0.392    0.034   0.011
 hostX.example.c 192.168.0.254   12 u   36   64    2    0.420   -0.057   0.000
```

Note that the "st" column, for "stratum", has converted from considering the peers "stratum 16" (completely untrustworthy) to "stratum 12" (one higher than the best known time source, instructor.example.com).

System Monitoring and Logs

Practice Case Study

Usage Reports

Use the tool you investigated to create a simple report that logs information to a file.

Once you have used the tool to generate a report, the instructor will have you share the command you used and explain the output with the rest of the class.

- ```
  [root@serverX] ~]# df -h
  [root@serverX] ~]# iostat -dNk 2 10
  [root@serverX] ~]# vmstat 2 10
  ```

Practice Resequencing Exercise

Resequencing Quiz: Implementing AIDE

4 Copy **/var/lib/aide/aide.db.new.gz** to **/var/lib/aide/aide.db.gz**
2 Run **/usr/sbin/aide --init** to build the initial database
5 Run **/usr/sbin/aide --check** to check your system.
1 Customize **/etc/aide.conf**
3 Store **/etc/aide.conf**, **/usr/sbin/aide** and **/var/lib/aide/aide.db.new.gz** in a secure location.

Practice Exercise

Tuning tmpwatch and logrotate

Login as root and perform the following steps on serverX.

1. Adjust **/etc/cron.daily/tmpwatch** to remove files in **/tmp** that have not been accessed in seven (7) days.

   Change the line in **/etc/cron.daily/tmpwatch** that ends in **10d /tmp** to **7d /tmp**

2. Configure the global **logrotate** default setting to compress all log files when they are rotated.

   Add (or uncomment) the following line in the **/etc/logrotate.conf** file:

   ```
   compress
   ```
   ```

Remote Logging Search & Learn

Given the comments found in the **/etc/rsyslog.conf** file, complete the tasks listed below. Additional information relevant to these tasks can be found in **/usr/share/doc/rsyslog-*/rsyslog_conf_actions.html**.

1. Configure serverX to accept remote log messages using UDP.

   - Review the **/etc/rsyslog.conf** configuration file. What modifications would you have to make to accept remote log messages using UDP?
   - Make the necessary modifications and reload the **rsyslog** service on serverX.
     /etc/rsyslog.conf must contain the following to accept remote UDP log connections:

     ```
     $ModLoad imudp.so
     $UDPServerRun 514
     ```

     Restart the service:

     ```
     [root@serverX ~]# service rsyslog restart
     ```

2. Configure desktopX to send log messages via UDP to serverX.

   - What modification to **/etc/rsyslog.conf** on desktopX will cause all **info** priority and higher events to be sent to serverX using UDP?
   - Make the necessary modifications and reload the **rsyslog** service on desktopX.
     Add the following line to **/etc/rsyslog.conf** to send all **info** priority and higher events to serverX using UDP:

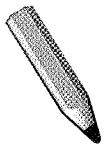
     ```
     *.info    @serverX
     ```

     Restart the service:

     ```
     [root@desktopX ~]# service rsyslog restart
     ```

3. Test your configuration by running **logger "Hello from desktopX"** on desktopX. Verify the message was received on both desktopX and serverX.

   ```
   [root@desktopX ~]# logger "Hello from desktopX"
   [root@desktopX ~]# tail /var/log/messages
   Jan 18 14:24:37 desktopX root: Hello from desktopX
   [root@serverX ~]# tail /var/log/messages
   Jan 18 14:24:37 desktopX root: Hello from desktopX
   ```



Test

Criterion Test

Case Study

System Monitoring and Logs

Before you begin...

Run the **lab-setup-bloomin** script on desktopX to prepare serverX for the assessment.

Every Bloomin' Thing is a nation wide cooperative of flower and plant growers. Among other things, the cooperative handles IT services for all members. The IT manager has decided to beef up security by requiring file integrity checking and remote logging on all servers, including your serverX.

Install **aide** on serverX and initialize the file integrity database. Do not wait for the database build to complete before continuing with the rest of the lab.

Configure rsyslog on desktopX to accept incoming log messages via UDP from serverX. Then configure **rsyslog** on serverX to send all ***.info** log messages to desktopX via UDP.

When you are ready to check your work, first run **lab-grade-bloomin** on serverX and then run it on desktopX.

1. On serverX, install the aide package and initialize the aide database.

```
[root@serverX ~]# yum -y install aide
[root@serverX ~]# aide --init
```

On desktopX, edit **/etc/rsyslog.conf**, removing the comment from lines 13 and 14.

```
# Provides UDP syslog reception
$ModLoad imudp.so
$UDPServerRun 514
```

2. On desktopX, restart the rsyslog service.

```
[root@desktopX ~]# service rsyslog restart
```

3. On desktopX, confirm that **rsyslogd** is bound to the external UDP port 514.

```
[root@desktopX ~]# lsof -i -n -P | grep rsyslogd
rsyslogd  2253    root    3u  IPv4  16673      0t0  UDP *:514
rsyslogd  2253    root    4u  IPv6  16674      0t0  UDP *:514
```

4. On serverX, edit the file **/etc/rsyslog.conf**, inserting the following line, replacing "X" with your station number. Although the exact location doesn't matter, around line 39, near the existing "info" configuration, would be reasonable.

```
*.info                                  @desktopX.example.com
```

On serverX, restart the rsyslogd service.

```
[root@serverX ~]# service rsyslog restart
```

5. Once **aide --init** has completed on serverX, prepare the database for use in integrity checks.

Note

Normally the database would be moved to read-only media, to be copied to disk when an integrity check (**aide -C**) is desired, and to be deleted from the disk thereafter. However, for the purposes of this lab you will leave the database in place.

```
[root@serverX ~]# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

Centralized and Secure Storage

Practice Performance Checklist

Configuring iSCSI

Configure your serverX to us iSCSI storage existing on instructor.example.com.

Perform the following steps on serverX unless directed otherwise.

- ☐ Verify that the *iscsi-initiator-utils* package is installed, and install it if needed.

```
[root@serverX ~]# yum install -y iscsi-initiator-utils
```

- ☐ Discover iSCSI targets on the iSCSI server on 192.168.0.254.

```
[root@serverX ~]# iscsiadm -m discovery -t st -p 192.168.0.254
Starting iscsid:                                           [  OK  ]
192.168.0.254:3260,1 iqn.2010-09.com.example:rdisks.serverX
```

- ☐ Log into the iSCSI target **iqn.2010-09.com.example:rdisks.serverX** on 192.168.0.254.

```
[root@serverX ~]# iscsiadm -m node -T iqn.2010-09.com.example:rdisks.serverX -p
 192.168.0.254 -l
Logging in to [iface: default, target: iqn.2010-09.com.example:rdisks.serverX,
 portal: 192.168.0.254,3260]
Login to [iface: default, target: iqn.2010-09.com.example:rdisks.serverX, portal:
 192.168.0.254,3260] successful.
```

- ☐ Identify the device file for your new iSCSI disk on your initiator.

```
[root@serverX ~]# dmesg | tail
scsi 2:0:0:0: RAID              IET      Controller       0001 PQ: 0 ANSI: 5
scsi 2:0:0:1: Direct-Access     IET      VIRTUAL-DISK     0001 PQ: 0 ANSI: 5
...
sd 2:0:0:1: [sda] Attached SCSI disk
```

- ☐ Set up a single partition on your new storage device, format the partition as ext4, and configure it to persistently mount on /mnt/iscsi at boot. *(Note: Do not forget to use **_netdev** as a mount option, or to mount by file system UUID and not by standard device name.)*

```
[root@serverX ~]# fdisk -cu /dev/sda
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-65535, default 2048):  Enter
Using default value 2048
```

```
Last sector, +sectors or +size{K,M,G} (2048-65535, default 65535):  Enter
Using default value 65535

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@serverX ~]# mkfs.ext4 /dev/sda1
[root@serverX ~]# blkid /dev/sda1
/dev/sda1: UUID="9ed75ae9-fe7c-4b2e-82fd-6a29df22c09d" TYPE="ext4"
```

Add the following line to **/etc/fstab**:

```
UUID="9ed75ae9-fe7c-4b2e-82fd-6a29df22c09d"  /mnt/iscsi  ext4  _netdev  1 2
```

Make the directory:

```
[root@serverX ~]# mkdir -p /mnt/iscsi
```

- ☐ Test your configuration.

```
[root@serverX ~]# reboot
[root@serverX ~]# df /mnt/iscsi
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1                30737      1407     27743   5% /mnt/iscsi
```

- ☐ Unmount the new file system and remove or comment the line in **/etc/fstab**.

```
[root@serverX ~]# umount /mnt/iscsi
```

Remove the line from **/etc/fstab**.

- ☐ Logout and delete the entry for the iSCSI target.

```
[root@serverX ~]# iscsiadm -m node -T iqn.2010-09.com.example:rdisks.serverX -p
 192.168.0.254 -u
[root@serverX ~]# iscsiadm -m node -T iqn-2010-09.com.example:rdisks.serverX -p
 192.168.0.254 -o delete
```

Encrypt Centralized Storage Worksheet

Create an encrypted block device

1. Use **fdisk** to partition the disk:

   ```
   [root@serverX ~]# fdisk /dev/sda
   ```

   Where **/dev/*sda*** is the name of the disk.

2. For better security, fill the device with random data from **/dev/urandom**:

   ```
   [root@serverX ~]# dd if=/dev/urandom of=/dev/sda1
   ```

Where **/dev/*sda1*** is the name of the partition created above.

This step is optional, but it greatly increases the strength of the encryption. The downside is that it can take a very long time (several minutes per gigabyte on most systems).

3. Encrypt the device with **cryptsetup luksFormat**:

```
[root@serverX ~]# cryptsetup luksFormat /dev/sda1
WARNING!
========
This will overwrite data on /dev/sda1 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: redhat
Verify passphrase: redhat
```

Decrypt and persistently mount an encrypted block device

1. Decrypt the device using **cryptsetup luksOpen**:

```
[root@serverX ~]# cryptsetup luksOpen /dev/sda1name
Enter passphrase for /dev/sda1: redhat
```

Where *name* will be used as the device mapper name: **/dev/mapper/*name***

2. Create a file system on the opened device:

```
[root@serverX ~]# mkfs.ext4 /dev/mapper/name
```

3. Add an entry to **/etc/crypttab**:

```
name  /dev/sda1
```

Note

You will have to enter the encryption password every time you boot.

4. Add an entry to **/etc/fstab**:

```
/dev/mapper/name  /name  ext4 _netdev 1 2
```

Note

Remember to use the **_netdev** option because we are using an iSCSI disk.

5. Mount the file system to ensure it is persistent:

```
[root@serverX ~]# mkdir /name
```

```
[root@serverX ~]# mount -a
```

Unmount and close an encrypted block device

1. Unmount the device:

```
[root@serverX ~]# umount /name
```

2. Close the encrypted device using **cryptsetup luksClose**:

```
[root@serverX ~]# cryptsetup luksClose /dev/mapper/name
```

Practice Performance Checklist

Encrypting iSCSI

Modify the serverX iSCSI-attached storage to be encrypted.

Perform the following steps on serverX unless directed otherwise.

- ☐ Rediscover and log in to the iSCSI target **iqn.2010-09.com.example:rdisks.serverX** on 192.168.0.254.

```
[root@serverX ~]# iscsiadm -m discovery -t st -p 192.168.0.254
[root@serverX ~]# iscsiadm -m node -T iqn.2010-09.com.example:rdisks.serverX -p
 192.168.0.254 -l
```

- ☐ Create and mount an encrypted block device:
  - Reuse the partition you created in the last section. Encrypt the partition using a passphrase of **redhat**

```
[root@serverX ~]# dd if=/dev/urandom of=/dev/sda1
[root@serverX ~]# cryptsetup luksFormat /dev/sda1
WARNING!
========
This will overwrite data on /dev/sda1 irrevocably.
Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: redhat
Verify passphrase: redhat
```

  - Open the new partition with the label **iscsi-secret**

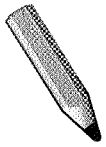
```
[root@serverX ~]# cryptsetup luksOpen /dev/sda1 iscsi-secret
Enter passphrase for /dev/sda1: redhat
```

  - Format **/dev/mapper/iscsi-secret** with an ext4 file system

```
[root@serverX ~]# mkfs.ext4 /dev/mapper/iscsi-secret
```

  - Manually mount **/dev/mapper/iscsi-secret** on **/storage**.

```
[root@serverX ~]# mkdir -p /storage
[root@serverX ~]# mount /dev/mapper/iscsi-secret /storage
```



Test

Criterion Test

Case Study

Centralized Storage

Before you begin...

Run **lab-setup-centralstore** on desktopX to prepare serverX for the exercise.

Cold Storage, an appliances retailer, recently acquired new SAN storage that utilizes the iSCSI protocol.

We will begin deployment with your serverX that will be configured to access its own dedicated iSCSI target:

- iSCSI Target IP Address: 192.168.0.254
- iSCSI Target: **iqn.2010-09.com.example:rdisks.serverX**

Partition (entire device), format, and persistently mount to **/coldstorage**.

When you are ready to check your work, run **lab-grade-centralstore** on serverX.

1. Run **lab-setup-centralstore** on desktopX to prepare serverX for the exercise:

   ```
   [root@desktopX ~]# lab-setup-centralstore
   ```

2. Log into serverX and become root
3. Install necessary packages:

   ```
   # yum install iscsi-initiator-utils
   ```

4. Connect to the target:

   ```
   [root@serverX ~]# iscsiadm -m discovery -t st -p 192.168.0.254
   [root@serverX ~]# iscsiadm -m node -T iqn.2010-09.com.example:rdisks.serverX -p
    192.168.0.254 -l
   ```

5. Partition, format and persistently mount new device with an ext4 file system:

   - Determine the device name (/dev/sda) of the iSCSI device in the log files

     ```
     [root@serverX ~]# tail /var/log/messages
     ```

- Create a partition table on the device

```
[root@serverX ~]# fdisk /dev/sda
```

- Format the partition with a file system

```
[root@serverX ~]# mkfs -t ext4 /dev/sda1
```

- Create a mount point

```
[root@serverX ~]# mkdir /coldstorage
```

- Determine UUID of partition

```
[root@serverX ~]# blkid /dev/sda1
```

- Create an entry in **/etc/fstab**

```
[root@serverX ~]# vim /etc/fstab # Add an entry that looks like (copying UUID from
 previous step):
UUID=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX  /coldstorage  ext4  _netdev  0 0
```

- Test

```
[root@serverX ~]# mount -a
```

6. When you are ready to check your work, run **lab-grade-centralstore** on serverX.

```
[root@serverX ~]# lab-grade-centralstore
```

SSL Encapsulated Web Services

Practice Performance Checklist

Apache mod_ssl Basics

Deploy an SSL encapsulated Apache web server on serverX. It should use the default self-signed SSL certificate.

- ☐ Log into serverX as root.

- ☐ Install the Apache web server (*httpd*) package and the *mod_ssl* package, if necessary.

```
[root@serverX ~]# yum install -y httpd mod_ssl
```

- ☐ Examine the **/etc/httpd/conf.d/ssl.conf** configuration file provided by the **mod_ssl** package.

  - What is the Apache directive that points to the SSL certificate?

  - What is its value?

```
[root@serverX ~]# less /etc/httpd/conf.d/ssl.conf
...
# Point SSLCertificateFile at a PEM encoded certificate.  If
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/pki/tls/certs/localhost.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
...
```

- ☐ Use **openssl** to display the subject and issuer of the SSL certificate that Apache uses.

- ☐ Restart the **httpd** service.

```
[root@serverX ~]# service httpd restart
```

- ☐ Launch Firefox and browse to https://serverX.example.com. When Firefox presents a warning, take further steps to examine the certificate with Firefox.

  - Click the "I Understand the Risks" link.

  - Click the "Add Exceptions..." button, then click "View..." when it becomes active.

- Browse the information presented in both the "General" and "Details" tabs.
- Click "Close" when you are finished inspecting the certificate information.

Practice Performance Checklist

Creating a Custom Self-Signed Certificate

Deploy an SSL encapsulated Apache web server on serverX. It should use a personalized, self-signed SSL certificate.

☐ Login as **root** on serverX. Make sure the *crypto-utils* package is installed.

```
[root@serverX ~]# yum install -y crypto-utils
```

☐ Use the **genkey** utility to create a custom self-signed certificate for serverX.example.com that will expire after a year. Note where the key and the matching certificate are created.

```
[root@serverX ~]# genkey --days 365 serverX.example.com
```

☐ The certificate should have the following characteristics:

- The key should be 1024 bits and should *not* be encrypted.
- country code = *local country*
- state = *local state*
- locality = *local city*
- organization = Red Hat Inc.
- common name = serverX.example.com

☐ Modify the Apache configuration to point to the new key and certificate. Do *not* rename the files just created.

Edit **/etc/httpd/conf.d/ssl.conf** and set the following values:

```
SSLCertificateFile /etc/pki/tls/certs/serverX.example.com.crt
SSLCertificateKeyFile /etc/pki/tls/private/serverX.example.com.key
```

☐ Restart the Apache web service.

```
[root@serverX ~]# service httpd restart
```

☐ Create the **check-remote-cert.sh** script (as shown previously in your workbook) on your desktopX machine.

Create a file named **check-remote-cert.sh** in **/usr/local/bin/** with the following content:

```
#!/bin/bash

# usage: check-remote-cert.sh HOST:PORT
SERVER=$1
echo |
openssl s_client -connect ${SERVER} 2>/dev/null |
openssl x509 -noout -subject -issuer -dates
```

- ☐ Make the script executable and run it so that it connects to serverX.example.com port 443.

```
[root@serverX ~]# chmod 755 /usr/local/bin/check-remote-cert.sh
[root@serverX ~]# /usr/local/bin/check-remote-cert.sh serverX.example.com:443
```

- ☐ Confirm the certificate subject, issuer, and dates are correct and match what you specified earlier.

Generating a Certificate	Generating a CSR
Generating a Certificate	Generating a CSR
Run **genkey --days *no_of_days host_fqdn***	Run **genkey --genreq --days *no_of_days host_fqdn*** The main difference in the procedure is that we add the **--genreq** option and say Yes to "Send a CSR to a CA?"
Skim intro, click [next]	Skim intro, click [next]
Specify number of bits, then click [next]	Specify number of bits, then click [next]
Generate random bits (via mouse and console keyboard)	Generate random bits (via mouse and console keyboard)
Generate CSR? Select No	Generate CSR? Select Yes
Leave checkbox for passphrase unchecked, click [next]	Leave checkbox for passphrase unchecked, click [next]
Specify certificate details (country, state, locality, organization, org unit, common name (fqdn), email)	Specify certificate details (country, state, locality, organization, org unit, common name (fqdn), email)

Table A.3. Compare and Contrast: Generating a Self-signed Certificate vs. a CSR

Practice Performance Checklist

Working with CA-Signed Certificates

Deploy an SSL encapsulated Apache web server on serverX. It should use a CA-signed SSL certificate.

- ☐ Use the **genkey** utility to create a new key and a certificate signing request (CSR) for serverX.example.com that will expire after a year.

```
[root@serverX ~]# genkey --genreq --days 365 serverX.example.com
```

- ☐ The requested certificate should have the following characteristics:
  - The key should be 1024 bits and should *not* be encrypted.
  - country code = *local country*
  - state = *local state*
  - locality = *local city*
  - organization = Red Hat Inc.
  - common name = serverX.example.com
- ☐ Use the web form at **http://instructor.example.com/lab/csr_upload.html** to upload your certificate signature request.
- ☐ Save the returned certificate to a local file. For **Firefox**, this can be accomplished by choosing **File : Save Page As....**

  Move the saved certificate to the appropriate location, and deploy it with your Apache web server.

  Once you have uploaded your CSR as explained above, move the certificate to **/etc/pki/tls/certs/serverX.example.com.crt**. Edit **/etc/httpd/conf.d/ssl.conf** and set the following values:

```
SSLCertificateFile /etc/pki/tls/certs/serverX.example.com.crt
SSLCertificateKeyFile /etc/pki/tls/private/serverX.example.com.key
```

- ☐ Get a copy of the instructor's CA certificate from **http://instructor/pub/example-ca.crt**.
- ☐ Use it with **openssl** to verify that the certificate presented by your web server is properly signed.

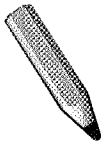
```
[root@serverX ~]# echo | openssl s_client -CAfile example-ca.crt -connect
 serverX.example.com:443 2>/dev/null | grep Verify
```

- ☐ Install the CA certificate in Firefox.

1. Launch **Firefox.**
2. Select **Edit** → **Preferences**
3. Click the **Advanced** icon
4. Select the **Encryption** tab
5. Select the **Authorities** tab in the new window
6. Click the Import... button
7. Browse to find the CA's certificate file and **Open** it
8. Check the **Trust to identify web sites** checkbox
9. Click the OK, and then the Close buttons to return to the browser.

☐ Launch Firefox and browse **`https://serverX.example.com`**. What is different?

It should no longer mention that the certificate is invalid.



Test

Criterion Test

Case Study

SSL Encapsulated Web Services

Before you begin...

Run the **`lab-setup-hacker`** script on desktopX.

Marcelo Hacker is a successful private investigator. In fact, he is doing so well that it is becoming difficult to find the time to meet with prospective clients. Mr. Hacker has decided to set up a website where prospective clients can send him messages. As confidentiality is an important part of the private investigation business, the website must use a signed SSL certificate.

- Set up Apache on serverX to provide an SSL encrypted website for Marcelo Hacker.
- Your instructor will act as the Certificate Authority that will sign your SSL certificate. Create a certificate request for serverX.example.com and upload it using the form **`http://instructor.example.com/lab/csr_upload.html`**.

  (Note that the certificate authority will refuse to issue a duplicate certificate, so make sure to vary your identity information from previous labs.)
- Deploy the signed certificate for Apache on serverX.
- Leave the default placeholder website for content. Mr. Hacker will upload his custom content at a later date.

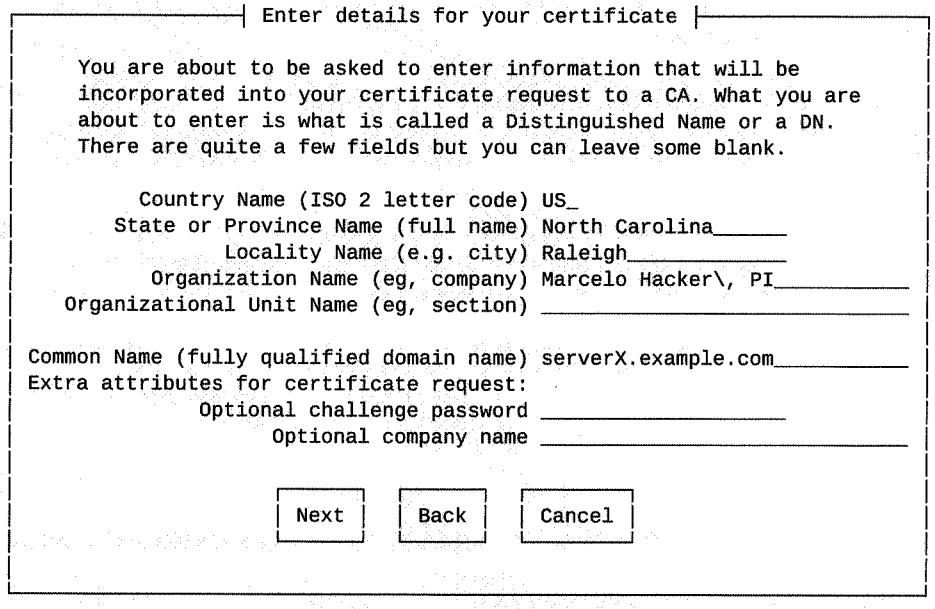
When you have fulfilled the requirements, run **lab-grade-hacker** on desktopX to check your work.

1. Make sure the *mod_ssl* and *crypto-utils* packages are installed.

```
[root@serverX ~]# yum install mod_ssl crypto-utils
```

2. Use the **genkey** command to create a new Certificate Signature Request, making sure to set the Common Name to **serverX**.

```
[root@serverX ~]# genkey --genreq serverX
```



3. Use **Firefox** to upload the resulting CSR file to **http://instructor.example.com/lab/csr_upload.html**. Save the returned certificate as a local file, using **File : Save Page As...**

   Move the saved certificate file to **/etc/pki/tls/certs/serverX.crt**.

4. Confirm the certificate and associated key have the appropriate SELinux context.

```
[root@serverX tls]# cd /etc/pki/tls/
[root@serverX tls]# ls -lZ certs/ private/
...
-rw-r--r--. root root unconfined_u:object_r:cert_t:s0  certs/serverX.crt
...
-rw-------. root root unconfined_u:object_r:cert_t:s0  private/serverX.key
...
```

5. Confirm that the certificate's Common Name is appropriate for your server.

```
[root@serverX tls]# openssl x509 -text < certs/serverX.crt | grep Subject:
```

```
        Subject: C=US, ST=North Carolina, O=Example, Inc., CN=serverX.example.com
```

6. Update your server's **/etc/httpd/conf.d/ssl.conf** configuration file, around line 100, setting the SSLCertificateFile and SSLCertifcateKeyFile to refer to your newly install certificate and key file.

```
...
#SSLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateFile /etc/pki/tls/certs/serverX.crt
...
#SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
SSLCertificateKeyFile /etc/pki/tls/private/serverX.key
...
```

7. Restart your web service. Note that a FAILED stop probably just indicates a service which was never originally started.

```
[root@serverX tls]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

8. (Optional) Use the **curl** command to confirm that clients can successfully verify encrypted connections with your server. You are not interested in the content, just that **curl** can download and verify the certificate without complaint.

```
[root@serverX tls]# CACERT=/net/instructor/var/ftp/pub/example-ca.crt
[root@serverX tls]# curl --cacert $CACERT https://serverX.example.com > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
102  3985  102  3985    0     0  62741      0 --:--:-- --:--:-- --:--:-- 1297k
```

9. In order to confirm that the Firefox web browser can successfully authenticate encrypted connections with your server, first install the local Certificate Authority's certificate by using Firefox to open the location *http://instructor/pub/example-ca.crt.*

   In the resulting dialog, trust the CA to identify (at least) websites.

   Browse *https://serverX.example.com*. Firefox should successfully verify the site using "Example, Inc.", which can be confirmed by "hovering" over the example.com prefix in the location bar.

Web Server Additional Configuration

Name-Based Virtual Hosting Keywords

1. VirtualHost

```
<VirtualHost  192.168.0.X+100:80>
...
</VirtualHost>
```

This is the block that defines a virtual host

2. NameVirtualHost

```
NameVirtualHost 192.168.0.X+100:80
```

This specifies on which address name-based virtual hosts are enabled (referenced in later **<VirtualHost>** blocks)

3. ServerName/ServerAlias

```
ServerName serverX.example.com
```

This specifies the server name. With name-based virtual hosts, the name here must match exactly with the client request.

4. ServerAdmin

```
ServerAlias serverX wwwX wwwX.example.com
```

A space-separated list of names to match like **ServerName** above.

5. ServerAdmin

```
ServerAdmin root@serverX.example.com
```

This specifies the administration email account. Web page errors will provide this email address.

6. DocumentRoot

```
DocumentRoot /var/www/html
```

Inside a **<VirtualHost>** block, specifies the directory from which to serve content.

7. **semanage fcontext**

```
semanage fcontext -l
semanage fcontext -a -t httpd_sys_content_t "/directory(/.*)?"
restorecon -vvFR /directory
```

semanage fcontext can set SELinux types on files persistently (**semanage fcontext -a ...** in the example above) and can list default file contexts (useful for seeing samples) with **semanage fcontext -l**.

Make sure *policycoreutils-python* is installed and review needed context in **httpd_selinux**(8).

httpd_sys_content_t is the standard SELinux type context for web pages. **public_content_t** and **public_content_rw_t** are types that are shared between services (HTTP, FTP, Samba, etc.). Use these types accordingly.

Practice Performance Checklist

Configure Name-Based Virtual Hosts

For this exercise, wwwX.example.com is already set up as a **CNAME** alias to serverX.example.com.

When you finish the checklist, you will run a grading script, so make sure your web server serves up the content exactly as described in the steps.

- ☐ Install and start **httpd** on serverX.

```
[root@serverX ~]# yum install -y httpd
[root@serverX ~]# service httpd start
[root@serverX ~]# chkconfig httpd on
```

- ☐ Create **/var/www/html/index.html** containing the text "**this is serverX.**"

```
[root@serverX ~]# echo this is serverX. > /var/www/html/index.html
```

- ☐ From desktopX, use Firefox to verify that the websites wwwX, wwwX.example.com, serverX, and serverX.example.com all display your custom **index.html**.

- ☐ Create **/wwwX/html/index.html** containing the text "**this is wwwX.**"

```
[root@serverX ~]# mkdir -p /wwwX/html
[root@serverX ~]# echo this is wwwX. > /wwwX/html/index.html
```

- ☐ Modify Apache to enable name-based virtual hosting. serverX and serverX.example.com should serve **/var/www/html/index.html** as the main page. wwwX and wwwX.example.com should serve **/wwwX/html/index.html** as the main page.

  Add the following content to the bottom of **/etc/httpd/conf/httpd.conf**:

```
# Enable name-based virtual hosting:
NameVirtualHost *:80

# serverX virtual host configuration
<VirtualHost *:80>
    ServerName serverX.example.com
    ServerAlias serverX
    ServerAdmin webmaster@serverX.example.com
    DocumentRoot /var/www/html
    ErrorLog logs/serverX.example.com-error_log
    CustomLog logs/serverX.example.com-access_log common
```

```
</VirtualHost>

# wwwX virtual host configuration
<VirtualHost *:80>
    ServerName wwwX.example.com
    ServerAlias wwwX
    ServerAdmin webmaster@wwwX.example.com
    DocumentRoot /wwwX/html
    ErrorLog logs/wwwX.example.com-error_log
    CustomLog logs/wwwX.example.com-access_log common
</VirtualHost>
```

Have the Apache server reload its configuration:

```
[root@serverX ~]# service httpd reload
```

- ☐ Do not disable SELinux (Hint: You may need to modify the SELinux file context database, or change the SELinux type of certain files).

```
[root@serverX ~]# semanage fcontext -a -t httpd_sys_content_t '/wwwX(/.*)?'
[root@serverX ~]# restorecon -vFR /wwwX
```

- ☐ When you finish, run the **lab-grade-virthost** evaluation script from serverX to make sure you have done everything correctly.

```
[root@serverX ~]# lab-grade-virthost
Good, www1 works
Good, www1.example.com works
Good, server1 works
Good, server1.example.com works
Grading PASSED!
```

CGI Permissions

Given a CGI script to be located at **/wwwX/cgi-bin/hostinfo.cgi**, what configuration syntax, filesystem permissions, and SELinux context type would you need to know?

1. Create a directory outside of the web site's **DocumentRoot**:

```
[root@serverX ~]# mkdir -p /wwwX/cgi-bin
```

2. Configure Apache to recognize it as a source for CGI programs:
   Add the following directive inside the relevant **<VirtualHost>** stanza:

```
ScriptAlias /cgi-bin/ "/wwwX/cgi-bin/"
```

3. Change the SELinux context of the CGI script directory to **httpd_sys_script_exec_t**:

```
[root@serverX ~]# semanage fcontext -a -t httpd_sys_script_exec_t '/wwwX/cgi-
bin(/.*)?'
[root@serverX ~]# restorecon -vFR /wwwX
```

4. Restart Apache to put the changes into effect:

```
[root@serverX ~]# service httpd restart
```

Deploy a CGI Script

Make sure you understand how to enable CGI on a directory if someone were to give you a CGI script.

1. Copy the script into the CGI script directory:

```
[root@serverX ~]# cp hostinfo.cgi /wwwX/cgi-bin/
```

2. Ensure the script is not writable by the **httpd** daemon:

```
[root@serverX ~]# chown root:root /wwwX/cgi-bin/hostinfo.cgi
```

3. Make it executable:

```
[root@serverX ~]# chmod 755 /wwwX/cgi-bin/hostinfo.cgi
```

Practice Quiz

Apache CGI

1. CGI stands for

(select one of the following...)

a. Content Generated Interface
b. Command Gateway Interface
c. Common Generated Interface
d. Common Gateway Interface

2. The last argument in **ScriptAlias /cgi-bin/ /my/private/cgi-bin/** is

(select one of the following...)

a. relative to **DocumentRoot**
b. relative to **/var/www/**
c. relative to **ServerRoot**
d. an absolute path on the filesystem

3. The default **ScriptAlias** in **/etc/httpd/conf/httpd.conf** is pointing to ...

(select one of the following...)

a. /var/www/cgi-bin
b. /var/html/cgi-bin
c. /cgi-bin
d. /var/www/html/cgi-bin

4. One of the built in SELinux context types for a generic CGI programs is

(select one of the following...)

a. **httpd_t**
b. **httpd_sys_script_exec_t**
c. **script_t**
d. **httpd_content_t**

5. The Apache process should have the following filesystem permissions on CGI programs

(select one of the following...)

a. **---**
b. **r--**
c. **r-x**
d. **rwx**

Practice Performance Checklist

Configure LDAP-Based Authentication

You will configure the web server on serverX with a **/private** URL that is accessible by users in the LDAP directory on instructor.example.com.

- ☐ Configure LDAP authentication on serverX using instructor.example.com as the LDAP server, **dc=example,dc=com** for the base distinguished name and use the certificate found at *ftp://instructor/pub/example-ca.crt*. Choose LDAP passwords.

  Run **system-config-authentication** on serverX. Choose **LDAP** in the **User Account Database** drop-down menu. Enter **ldap://instructor.example.com/** as the LDAP Server. Download the certificate from *ftp://instructor/pub/example-ca.crt*. Choose LDAP password as the **Authentication Method**.

- ☐ Login as **root** on serverX. Create a new directory **/var/www/html/private**.

```
[root@serverX ~]# mkdir /var/www/html/private
```

- ☐ In the **private** directory, create an **index.html** containing the text **Private Data**

```
[root@serverX ~]# echo "Private Data" > /var/www/html/private/index.html
```

- ☐ Download **ftp://instructor/pub/example-ca.crt** and place it in **/etc/httpd**

```
[root@serverX ~]# wget ftp://instructor/pub/example-ca.crt -O /etc/httpd/example-
ca.crt
```

- ☐ Edit **/etc/httpd/conf/httpd.conf** and add LDAP authentication for the **private** directory.

```
LDAPTrustedGlobalCert CA_BASE64 /etc/httpd/example-ca.crt
```

```
<Directory /var/www/html/private>
    AuthName "Secret Stuff"
    AuthType  basic
    AuthBasicProvider ldap
    AuthLDAPUrl "ldap://instructor.example.com/dc=example,dc=com" TLS
    Require valid-user
</Directory>
```

Add the stanza above to the bottom of the **/etc/httpd/conf/httpd.conf** file.

- ☐ Restart Apache

```
[root@serverX ~]# service httpd restart
```

- ☐ Browse to http://serverX.example.com/private. You should see an authentication dialog box pop up. If not, close all browser windows, check your configuration, and try again.

```
[root@serverX ~]# elinks http://serverX.example.com/private
```

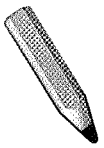
- ☐ Log in as user **ldapuserX** with a password of **password**

In the dialog box that pops up, enter the username and password above.

Practice Quiz

Troubleshooting Apache Quiz

1. Complete the following command to list all port contexts: **semanage** port **-l**.
2. Complete the following command to enable Apache to use TCP port 8001: **semanage** port -a **-t httpd_port_t** -p tcp **8001**.
3. The two Apache config file directives to specify the severity (how verbose) error messages are and which file to write to are LogLevel and ErrorLog.
4. The Apache config file directive to specify the format and location of clients accessing content is CustomLog.
5. Full (raw) SELinux AVC messages go to **/var/log** /audit/audit.log.
6. To make SELinux more verbose, you can run **semanage** dontaudit **off**.
7. The commands to get and set SELinux booleans are getsebool and setsebool.
8. **man** httpd_selinux will present an SELinux man page specific to Apache.
9. **man -k** _selinux lists all service specific SELinux man pages.
10. The **-F** option to **restorecon** will reset customizable types.



Test

Criterion Test

Case Study

Web Server Additional Configuration

Before you begin...

Run the **lab-setup-website** script on desktopX.

Example Industries, a fine example of a company, needs a new website. In fact, they need two! One will be the company website and the other will be for testing content. Additionally, the company website will need a password protected area and a special CGI application installed.

On your serverX machine, deploy a web server with two virtual hosts.

Virtual host 1: *http://serverX.example.com*

- Create a simple placeholder page for the base URL

Virtual host 2: *http://wwwX.example.com*

- Create a simple placeholder page for the base URL that is different from the one used on virtual host 1
- Make *http://wwwX.example.com/private* a password protected area
- Add user **forrest** with password **trees** to **/private**
- Download the CGI file ftp://instructor.example.com/pub/gls/special.cgi and install it as *http://wwwX.example.com/cgi-bin/special.cgi*

When you are ready to check your work, run the grading script on desktopX: **lab-grade-website**

1. Create a directory which will serve as the DocumentRoot for your virtual website.

```
[root@serverX ~]# mkdir -p /var/www/virtual/wwwX/html
```

2. Create placeholder "home pages" for each of your sites by creating a distinctive **index.html** file in their respective DocumentRoots.

```
[root@serverX ~]# echo serverX > /var/www/html/index.html
[root@serverX ~]# echo wwwX > /var/www/virtual/wwwX/html/index.html
```

3. Create the following minimal **/etc/httpd/conf.d/virtual.conf**, which contains virtual host definitions. Note that the filename does not matter, as long as it matches **/etc/httpd/conf.d/*.conf**.

```
NameVirtualHost 192.168.0.101

<VirtualHost 192.168.0.101>
        ServerName serverX.example.com
        ServerAlias serverX
</VirtualHost>
```

```
<VirtualHost 192.168.0.101>
        ServerName wwwX.example.com
        ServerAlias wwwX
        DocumentRoot /var/www/virtual/wwwX/html
</VirtualHost>
```

4. Restart your server, and confirm that the two virtual hosts serve the expected content.

```
[root@serverX ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@serverX ~]# curl http://serverX.example.com
serverX
[root@serverX ~]# curl http://wwwX.example.com
wwwX
```

5. Create a private area for your virtual site, and create some test content for that area.

```
[root@serverX ~]# mkdir /var/www/virtual/wwwX/html/private
[root@serverX ~]# echo "ssshhhhh" > /var/www/virtual/wwwX/html/private/secret
```

6. Add the following context stanza to your **virtual.conf** configuration file.

```
<Directory /var/www/virtual/wwwX/html/private>

        AuthName "Secret Hideout"
        AuthType basic

        AuthUserFile /etc/httpd/users
        require valid-user

        Options +Indexes

</Directory>
```

Note that enabling directory browsing was not specifically asked for, but mimics the "real server's" default behavior. More importantly, it will make the grading script happier.

7. Restart your server, and confirm that the context stanza has the anticipated effect.

```
[root@serverX ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@serverX ~]# curl http://wwwX.example.com/private/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Authorization Required</title>
</head><body>
...
```

8. Create the specified web user and password.

```
[root@serverX ~]# htpasswd -cm /etc/httpd/users forrest
New password: trees
Re-type new password: trees
```

```
Adding password for user forrest
```

9. Confirm the specified user can access the private area.

```
[root@serverX ~]# curl http://forrest:trees@wwwX.example.com/private/secret
sshhhhh
```

10. Stretch and take a deep breath.

11. Inside **/etc/httpd/conf.d/virtual.conf**, add the following ScriptAlias directive inside your wwwX VirtualHost stanza.

```
<VirtualHost 192.168.0.101>
        ServerName wwwX.example.com
        ServerAlias wwwX
        DocumentRoot /var/www/virtual/wwwX/html
        ScriptAlias /cgi-bin/  "/var/www/virtual/wwwX/cgi-bin/"
</VirtualHost>
```

12. Restart your server, so the new configuration takes effect.

```
[root@serverX ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

13. Create the specified **cgi-bin** directory. Download the specified CGI executable, and install it into the newly created directory, and make it executable.

```
[root@serverX ~]# mkdir /var/www/virtual/wwwX/cgi-bin
[root@serverX ~]# curl http://instructor/pub/gls/special.cgi > /var/www/virtual/wwwX/
cgi-bin/special.cgi
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0    77    0    77    0     0  17416      0 --:--:-- --:--:-- --:--:-- 77000
[root@serverX ~]# chmod 755 /var/www/virtual/wwwX/cgi-bin/special.cgi
```

14. Confirm that you can locally execute the CGI executable.

```
[root@serverX ~]# /var/www/virtual/wwwX/cgi-bin/special.cgi
Content-Type: text-html

<h1>Hello world</h1>
```

15. Confirm that the CGI executable can be accessed at the specified URL.

```
[root@serverX ~]# curl http://wwwX/cgi-bin/special.cgi
<h1>Hello world</h1>
```

16. Note: Because we chose "conforming" directory locations (locations which were registered in the SELinux policy) for our virtual host content, newly created files automatically obtained the correct SELinux context. Choosing different locations would require adjusting the

SELinux types. The simplest approach would be to copy the SELinux types from the "known good" base server locations.

```
[root@serverX ~]# cd /var/www/virtual/wwwX/
[root@serverX wwwX]# chcon -R --reference /var/www/html html/
[root@serverX wwwX]# chcon -R --reference /var/www/cgi-bin cgi-bin/
[root@serverX wwwX]# ls -lZ
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
```

Basic SMTP Configuration

External DNS

It is expected that an MX record in DNS for example.com must point to the mailbox server, mail.example.com:

```
example.com.      IN MX 10    mail.example.com.
```

Concept	Directive	*mail.example.com*	*desktop.example.com*	*smtp.example.com*
Binding Interface	*inet_interfaces*	inet_interfaces = all	inet_interfaces = loopback-only	inet_interfaces = all
Masquerade as	*myorigin*	myorigin = example.com	myorigin = example.com	myorigin = example.com
Indirect Delivery	*relayhost*	relayhost = [smtp.example.com]	relayhost = [smtp.example.com]	relayhost = ***(this is the relayhost)***
Receive mail for ...	*mydestination*	mydestination = localhost, $myhostname, $mydomain, example.com	mydestination = ***(no local delivery)***	mydestination = ***(no local delivery)***
Local Delivery	*local_transport*	local_transport = local: $myhostname ***(this is the default)***	local_transport = error: local delivery disabled	local_transport = error: local delivery disabled
Relay from ...	*mynetworks*	mynetworks = 127.0.0.1/8 ***(this is the default)***	mynetworks = 127.0.0.1/8 ***(this is the default)***	mynetworks = 192.168.0.0/24, 127.0.0.0/8

Table A.4. Intranet Mail Server, Null Client, and Outbound Relay

Practice Case Study

Intranet Configuration

Before you begin...

DNS has already been configured to overlay your hosts as members of the **domainX.example.com** domain.

Hostname	IP address	Also known as
mail.domainX.example.com	192.168.0.X+100	(serverX.example.com)
smtp.domainX.example.com	192.168.0.X+200	(hostX.example.com)
desktop.domainX.example.com	192.168.0.X	(desktopX.example.com)

Table A.5. domainX.example.com

Also, the host mail.domainX.example.com is the **MX** recipient for the entire domainX.example.com domain.

Complete the following table with the appropriate directives to configure these hosts to act as an intranet mailbox server, smtp host, and client station, respectively.

Try to use only the the **BASIC_CONFIGURATION_README**, **STANDARD_CONFIGURATION_README** and **main.cf** files for reference.

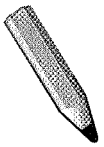
Once complete, have another student review your work.

External DNS:

```
domainX.example.com.      IN MX 10    mail.domainX.example.com.
```

Concept	Directive	*mail.domainX*	*desktop.domainX*	*smtp.domainX*
Binding Interface	*inet_interfaces*	inet_interfaces = all	inet_interfaces = loopback-only	inet_interfaces = all
Masquerade as	*myorigin*	myorigin = domainX.example.com	myorigin = domainX.example.com	myorigin = domainX.example.com
Indirect Delivery	*relayhost*	relayhost = [smtp.domainX.example.com]	relayhost = [smtp.domainX.example.com]	relayhost = ***(this is the relayhost)***
Receive mail for ...	*mydestination*	mydestination = localhost, $myhostname, $mydomain, domainX.example.com	mydestination = ***(no local delivery)***	mydestination = ***(no local delivery)***
Local Delivery	*local_transport*	local_transport = local: $myhostname ***(this is the default)***	local_transport = error: local delivery disabled	local_transport = error: local delivery disabled
Relay from ...	*mynetworks*	mynetworks = 127.0.0.1/8 ***(this is the default)***	mynetworks = 127.0.0.1/8 ***(this is the default)***	mynetworks = 192.168.0.0/24, 127.0.0.0/8

Table A.6. Intranet Mail Configuration for domainX.example.com



Test
Criterion Test

Case Study
Intranet E-mail Configuration

Before you begin...

Run the script **`lab-setup-email`** on desktopX

The Hoffman Hair Supply company, a manufacturer of hair grooming products, wants to centralize the management of their internal e-mail.

DNS has already been configured so that your machines are members of the DNS domain **`domainX.example.com`** with the following addresses:

```
192.168.0.X      desktop.domainX.example.com   (a.k.a.  desktopX.example.com)
192.168.0.X+100     mail.domainX.example.com   (a.k.a.   serverX.example.com)
192.168.0.X+200     smtp.domainX.example.com   (a.k.a. hostX.example.com)
```

Also, the **`mail.domainX.example.com`** server is the **`MX`** recipient for the entire **`domainX.example.com`** domain.

Configure the **`mail.domainX.example.com`** host to act as an incoming mail-only server, so that all mail delivered to the **`@domainX.example.com`** domain is stored on this server.

Configure the **`smtp.domainX.example.com`** server to act as an outgoing SMTP server, which is willing to relay mail from members of the **`domainX.example.com`** domain to outside networks.

Configure the **`desktop.domainX.example.com`** host to act as a "null client". It cannot receive e-mail from the network, local mail delivery is disabled, and all outgoing e-mail is sent indirectly via **`smtp.domainX.example.com`**.

For all three hosts, make sure that any originating mail masquerades the sender's domain as **`domainX.example.com`**.

When you are ready to check your work, run **`lab-grade-email`** on desktopX.

1. These instructions will refer to your three machines in the lab's context, *desktop.domainX.example.com*, *mail.domainX.domainX.eample.com*, and *smtp.domainX.example.com*, although the hostnames *desktopX*, *serverX*, and *hostX*, respectively, can still be used.

   For brevity, these three machines will usually be referred to a *desktop*, *mail*, and *smtp*, with the *domainX.example.com* implied.

   On *desktop.domainX.example.com* (*desktopX*), confirm that DNS has been properly pre-configured for your domain, by asking for a "domain dump" of *domainX.example.com*. (Note: in the "real world", most DNS servers will not allow clients to dump an entire domain.)

```
[root@desktop1 ~]# host -al domainX.example.com
...
;; QUESTION SECTION:
;domainX.example.com.           IN      AXFR

;; ANSWER SECTION:
domainX.example.com.    86400   IN      SOA     instructor.example.com.
 root.instructor.example.com. 2009062000 3600 300 604800 60
domainX.example.com.    86400   IN      NS      instructor.example.com.
domainX.example.com.    86400   IN      MX      10 mail.domainX.example.com.
desktop.domainX.example.com. 86400 IN   A       192.168.0.X
mail.domainX.example.com. 86400 IN      A       192.168.0.X+100
smtp.domainX.example.com. 86400 IN      A       192.168.0.X+200
domainX.example.com.    86400   IN      SOA     instructor.example.com.
 root.instructor.example.com. 2009062000 3600 300 604800 60
...
```

In particular, confirm the A records of *desktop*, *mail*, and *smtp* which perform the mapping to *domainX*, *serverX*, and *hostX*'s IP addresses, and the MX record that establishes *mail.domainX.example.com* as the "MX recipient" for the entire domainX.example.com domain. The rest of the DNS entries can be safely ignored.

2. In order to monitor your progress, it helps to monitor the **/var/log/maillog** on each of the three machines. As a suggestion, on the *desktop* machine, open 3 terminals. Leave one with a local shell, and open a remote shell to *mail* and *smtp* in each of the other two. In each of the terminals, run **less -F /var/log/messages**. (This places **less** in "tail -f" behavior, from which **CTRL-C** can be used to drop back into "normal" **less**, and **F** can be used to resume following the file.)

3. First, configure *mail* to be the MX recipient for the *domainX.example.com* domain. All of the following steps should be executed on *mail.domainX.example.com*.

   a. Ensure that postfix is installed, started, and enabled.

   ```
   [root@serverX ~]# yum -y install postfix
   [root@serverX ~]# service postfix restart
   [root@serverX ~]# chkconfig postfix on
   ```

   b. After making a backup of the primary configuration file, issue the following postfix configuration commands, and restart the server. (Of course, configuration changes can be made by editing the **/etc/postfix/main.cf** configuration file directly, which benefits from helpful comments.)

   ```
   [root@serverX ~]# cp /etc/postfix/main.cf /etc/postfix/main.cf.orig
   [root@serverX ~]# postconf -e inet_interfaces=all
   [root@serverX ~]# postconf -e myorigin=domainX.example.com
   [root@serverX ~]# postconf -e 'relayhost=[smtp.domainX.example.com]'
   [root@serverX ~]# postconf -e mydestination=domainX.example.com
   [root@serverX ~]# service postfix restart
   ```

   c. Confirm that mail sent locally to student@domainX.example.com is received by the host *mail*, which considers it a final destination.

```
[root@serverX ~]# date | mail -s test student@domainX.example.com
```

Examine the tail of **/var/log/maillog** on *mail* for lines similar to the following, where what is significant is "to=<student@domain1.example.com> ... (delivered to mailbox)".

```
...
Dec 10 05:39:48 serverX postfix/qmgr[28222]: 53948105A0:
 from=<root@serverX.example.com>, size=470, nrcpt=1 (queue active)
Dec 10 05:39:48 serverX postfix/local[28260]: 53948105A0:
 to=<student@domainX.example.com>, relay=local, delay=0.1,
 delays=0.07/0.02/0/0.01, dsn=2.0.0, status=sent (delivered to mailbox)
...
```

4. Secondly, configure *smtp* to be an outgoing mail relay. All of the following steps should be executed on *smtp.domainX.example.com*.

   a. Ensure that postfix is installed, started, and enabled.

   ```
   [root@hostX ~]# yum -y install postfix
   [root@hostX ~]# service postfix restart
   [root@hostX ~]# chkconfig postfix on
   ```

   b. After making a backup of the primary configuration file, issue the following postfix configuration commands, and restart the server.

   ```
   [root@hostX ~]# cp /etc/postfix/main.cf /etc/postfix/main.cf.orig
   [root@hostX ~]# postconf -e inet_interfaces=all
   [root@hostX ~]# postconf -e myorigin=domainX.example.com
   [root@hostX ~]# postconf -e local_transport="error:local delivery disabled"
   [root@hostX ~]# postconf -e mynetworks="127.0.0.0/8 192.168.0.0/24"
   [root@hostX ~]# service postfix restart
   ```

   c. Unfortunately, at this point, there is little that can be confirmed.

5. Thirdly, configure *desktop* to be a null client. All of the following commands are to be run on *desktop.domainX.example.com*.

   a. Ensure that postfix is installed, started, and enabled.

   ```
   [root@desktopX ~]# yum install postfix
   [root@desktopX ~]# service postfix restart
   [root@desktopX ~]# chkconfig postfix on
   ```

   b. After making a backup of the primary configuration file, issue the following postfix configuration commands, and restart the server.

   ```
   [root@desktopX ~]# cp /etc/postfix/main.cf /etc/postfix/main.cf.orig
   [root@desktopX ~]# postconf -e myorigin=domainX.example.com
   [root@desktopX ~]# postconf -e 'relayhost=[smtp.domainX.example.com]'
   [root@desktopX ~]# postconf -e local_transport="error:local delivery disabled"
   [root@desktopX ~]# service postfix restart
   ```

6. As a final confirmation of your work, from *desktop*, send a test message to student@domainX.example.com. Through the various **/var/log/message** files, trace the path of the email as it originates from *desktop*, passes through *smtp* as an outgoing relay, and is finally received by *mail* as its final destination.

```
[root@desktopX ~]# date | mail -s final_test student@domainX.example.com
```

   a. On *desktop*, examine the tail of **/var/log/maillog** for lines similar to the following, where what is significant is 'relay="smtp.domainX.example.com[192.168.0.X+200]:25"' and 'status=sent'.

```
Dec 10 14:32:04 desktopX postfix/qmgr[8860]: 7D35D247D7:
 from=<root@desktopX.example.com>, size=473, nrcpt=1 (queue active)
Dec 10 14:32:04 desktopX postfix/smtp[8905]: 7D35D247D7:
 to=<student@domainX.example.com>, relay=smtp.domainX.example.com[192.168.0.X
+200]:25, delay=0.18, delays=0.06/0.02/0.06/0.04, dsn=2.0.0, status=sent (250
 2.0.0 Ok: queued as 606C510594)
Dec 10 14:32:04 desktopX postfix/qmgr[8860]: 7D35D247D7: removed
```

   b. On *smtp*, examine the tail of **/var/log/maillog** for lines similar to the following, where both the reception ("client=desktopX.example.com[192.168.0.X] ... from=<root@desktopX.example.com>") and the subsequent transmission ("to=<student@domainX.example.com> ... relay=mail.domainX.example.com[192.168.0.X +100]:25, ... status=sent") of the email should be evident.

```
...
Dec 10 06:20:02 hostX postfix/smtpd[27799]: connect from
 desktopX.example.com[192.168.0.X]
Dec 10 06:20:02 hostX postfix/smtpd[27799]: 606C510594:
 client=desktopX.example.com[192.168.0.X]
Dec 10 06:20:02 hostX postfix/cleanup[27802]: 606C510594: message-
id=<20101210193204.7D35D247D7@desktopX.example.com>
Dec 10 06:20:02 hostX postfix/qmgr[27724]: 606C510594:
 from=<root@desktopX.example.com>, size=684, nrcpt=1 (queue active)
Dec 10 06:20:02 hostX postfix/smtpd[27799]: disconnect from
 desktopX.example.com[192.168.0.X]
Dec 10 06:20:02 hostX postfix/smtp[27803]: 606C510594:
 to=<student@domainX.example.com>, relay=mail.domainX.example.com[192.168.0.X
+100]:25, delay=0.13, delays=0.02/0.02/0.05/0.04, dsn=2.0.0, status=sent (250
 2.0.0 Ok: queued as 7EA04105A0)
...
```

   c. On *mail*, examine the tail of **/var/log/maillog** for lines similar to the following, where the reception (client=hostX.example.com[192.168.0.X +200] ... from=<root@desktopX.example.com> and final delivery of the email ("to=<student@domainX.example.com> ... status=sent (delivered to mailbox)") should be evident.

```
...
Dec 10 06:20:02 serverX postfix/cleanup[28475]: 7EA04105A0: message-
id=<20101210193204.7D35D247D7@desktopX.example.com>
Dec 10 06:20:02 serverX postfix/qmgr[28457]: 7EA04105A0:
 from=<root@desktopX.example.com>, size=897, nrcpt=1 (queue active)
Dec 10 06:20:02 serverX postfix/smtpd[28472]: disconnect from
 hostX.example.com[192.168.0.X+200]
```

```
Dec 10 06:20:02 serverX postfix/local[28476]: 7EA04105A0:
 to=<student@domainX.example.com>, relay=local, delay=0.05,
 delays=0.02/0.02/0/0.01, dsn=2.0.0, status=sent (delivered to mailbox)
Dec 10 06:20:02 server1 postfix/qmgr[28457]: 7EA04105A0: removed
...
```

d. Lastly, as the user *student* on *mail*, check your email using your favorite email client, such as **mutt**.

```
[root@serverX ~]# yum install -y mutt
[root@serverX ~]# su - student
[student@serverX ~]# mutt
```

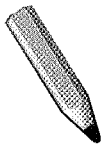
By listing *all* headers (within **mutt**, this requires typing **h** while viewing the message), you should be able to trace the path of the email in reverse order.

```
...
Received: from hostX.example.com (hostX.example.com [192.168.0.X+200])
        by serverX.example.com (Postfix) with ESMTP id 7EA04105A0
        for <student@domainX.example.com>; Fri, 10 Dec 2010 06:20:02 -0500
        (EST)
Received: from desktopX.example.com (desktopX.example.com [192.168.0.X])
        by hostX.example.com (Postfix) with ESMTP id 606C510594
        for <student@domainX.example.com>; Fri, 10 Dec 2010 06:20:02 -0500
        (EST)
Received: by desktopX.example.com (Postfix, from userid 0)
        id 7D35D247D7; Fri, 10 Dec 2010 14:32:04 -0500 (EST)
...
```

7. Lastly, from *desktop*, let your instructor in on the fun by sending mail to *instructor.example.com*.

```
[root@desktopX ~]# echo whew | mail -s done instructor@instructor.example.com
```

Caching-Only DNS Server



Test

Criterion Test

Case Study

Caching-Only DNS Server

Before you begin...

Run **lab-setup-cachingdns** on desktopX to prepare serverX for the exercise.

For his growing import/export business, Mr. Hnath would like to improve name resolution performance by deploying a caching name server at each of his business locations.

Recursive queries should be forwarded to the main name server at Hnath Import/Export headquarters.

- Set up a caching name server on serverX.
- Configure the name server so that recursive queries are sent to instructor.example.com. Also, configure the name server to accept queries from anyone on the classroom network.

When you are ready to check your work, run **lab-grade-cachingdns** on desktopX.

1. Make sure the bind package is installed.

```
[root@serverX ~]# yum install bind
```

2. Modify the named configuration (**/etc/named.conf**) to support connections from the network by modifying the **listen-on** lines to look like the following.

```
listen-on port 53 { any; };
listen-on-v6 port 53 { any; };
```

3. Modify the named configuration (**/etc/named.conf**) to ignore DNSSec by modifying the **dnssec-validation** line to look like the following.

```
dnssec-validation no;
```

4. Modify the named configuration (**/etc/named.conf**) to accept queries from anyone on the classroom network by modifying the **allow-query** line to look like the following. Also, modify the configuration so that recursive queries are sent to **instructor.example.com** by inserting a **forwarders** line below the **allow-query** line as the following.

```
allow-query     { localhost; 192.168.0.0/24; };
forwarders { 192.168.0.254; };
```

5. Restart the named service. Note that a FAILED stop probably just indicates a service which was never originally started.

```
[root@server1 ~]# service named restart
Stopping named:                                            [  OK  ]
Starting named:                                            [  OK  ]
```

6. Test from the desktopX system.

```
[root@desktop1 ~]# host server1.example.com 192.168.0.101
Using domain server:
Name: 192.168.0.101
Address: 192.168.0.101#53
Aliases:

server1.example.com has address 192.168.0.101
```

File Sharing with NFS

NFSv2	NFSv3	NFSv4
Original public NFS protocol	Extended NFSv2 architecture	Complete NFS redesign
Still in use	Added features: TCP support 64-bit file sizes and offsets Larger read/write sizes	Presents a root pseudo-filesystem
	Some implementations (including Red Hat Enterprise Linux) support Kerberos	All implementations support Kerberos
Requires support services: nfsd, rpc.mountd, rpc.statd, lockd	Also requires support services: nfsd, rpc.mountd, rpc.statd, lockd	Integrates/eliminates the need for auxiliary support services (rpc.mountd, etc.)
More difficult to secure behind a firewall	More difficult to secure behind a firewall	Improved user mapping support (requires rpc.idmapd)
Useful for backward compatibility	**Useful for backward compatibility**	**Use this for new deployments whenever possible**

Practice Quiz

NFS Concepts

1. Under what circumstances should NFSv2 or NFSv3 be used? Legacy installations and some network clients don't support NFSv4

2. What is the syntax of the /etc/exports file? `/directory/share host(options) host(options)`

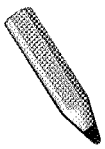
3. What steps should be taken to publish a new export on an existing NFSv4 server?

   Create an `/etc/exports` entry that defines access to the share

   Run `exportfs -r` as root on the server

4. Which option tells NFS to allow the root user on client systems to have root privileges in the share as well?

   `no_root_squash`



Test

Criterion Test

Case Study

File Sharing with NFS

Before you begin...

Run **lab-setup-strickland** on desktopX to prepare serverX for the exercise.

Strickland Pro Play is a store specializing in high end recreational equipment and accessories. The new sales software requires requires a file server with two shares that are mounted at each sales station in the store.

For the file server, deploy an NFSv4 service on desktopX. Create and share two exports on desktopX:

- The first export is for intake of current sales orders. On desktopX, export **/share/current** and make it writable. **root** on the client must be able to write to **/share/current** when mounted.
- The second export is to archive old orders. Again on desktopX, export the path **/share/archives** and make it read-only.
- Configure both exports so they are only available to the local classroom network.

Configure serverX to mount **desktopX:/share/current** as **/sales/current** and **desktopX:/share/archives** as **/sales/archives**. The mounts must be available after a reboot of serverX.

When you are ready to check your work, run **lab-grade-strickland** on serverX.

1. Run **lab-setup-strickland** on desktopX to prepare serverX for the exercise.

```
[root@desktopX ~]# lab-setup-strickland
```

2. First, create both directories to be shared on the NFS server, desktopX. Make the **current** directory writeable.

```
[root@desktopX ~]# mkdir -p /share/archives /share/current
[root@desktopX ~]# chmod 777 /share/current
```

(Another acceptible option is to export **/share/current** with the **no_root_squash** option set, in the next step.)

3. Define how both directories will be shared by creating **/etc/exports**.

```
[root@desktopX ~]# vi /etc/exports
[root@desktopX ~]# cat /etc/exports
/share/current 192.168.0.0/24(rw,sync)
/share/archives 192.168.0.0/24(ro,async)
```

4. Start the NFS service and configure it to start persistently.

```
[root@desktopX ~]# service nfs start
Starting NFS services:                     [  OK  ]
Starting NFS quotas:                       [  OK  ]
Starting NFS daemon:                       [  OK  ]
Starting NFS mountd:                       [  OK  ]
```

```
[root@desktopX ~]# chkconfig nfs on
```

5. Once the NFS server is configured and running, login as root on serverX and configure the NFS client. First, create the mountpoints.

```
[root@serverX ~]# mkdir -p /sales/archives /sales/current
```

6. Confirm the NFS shares exported by desktopX.example.com are visible to serverX and create **/etc/fstab** entries at the end to mount them.

```
[root@serverX ~]# showmount -e desktopX.example.com
Export list for desktopX.example.com:
/share/archives 192.168.0.0/24
/share/current  192.168.0.0/24
[root@serverX ~]# vi /etc/fstab
[root@serverX ~]# tail -n 2 /etc/fstab
desktopX.example.com:/share/current   /sales/current   nfs  rw  0 0
desktopX.example.com:/share/archives  /sales/archives  nfs  ro  0 0
```

7. Mount the NFS shares and confirm they mounted correctly. If all is well, then reboot serverX and run the lab-grade-strickland grading script.

```
[root@serverX ~]# mount -a
[root@serverX ~]# mount | grep desktop
desktopX.example.com:/share/current on /sales/current type nfs
 (rw,vers=4,addr=192.168.0.1,clientaddr=192.168.0.101)
desktopX.example.com:/share/archives on /sales/archives type nfs
 (ro,vers=4,addr=192.168.0.1,clientaddr=192.168.0.101)
[root@serverX ~]# reboot
```

File Sharing with CIFS

Practice Quiz

Accessing CIFS Share

1. What command-line would give ftp-style access to a CIFS share named "common" on a server named "nas2010", logging you in as a user named "winston"?

   `smbclient //nas2010/common -U winston`

2. What is wrong with the following line in /etc/fstab?

   `\\server\share /mnt/point cifs user=ralph,pass=password 0 0`

   Trick question: there is actually nothing wrong with this line. You can either use backslashes or forward slashes in **`/etc/fstab`**. However, on the command-line, you must double the backslashes if you choose to use them: **`smbclient \\\\serverX\\share`**

3. How would you store the login credentials in a separate file to keep them out of **`/etc/fstab`**?

   in **`fstab`** (column 4): **`credentials=filename`** and **`filename`** contains on three lines: **`username=value`**, **`password=value`**, and **`domain=value`**

4. When mounting a Windows-based CIFS share, what option allows you to specify the Linux ownership of all mounted files?

   `uid=value, gid=value` where value can be either the Linux UID/GID or username/groupname

Practice Performance Checklist

Samba Home Directories Configuration

Modify the default Samba configuration and security elements to support access to user home directories.

- ☐ Log in to serverX and become root.

  ```
  [student@serverX ~]# su -
  ```

- ☐ Install the necessary package(s) for a Samba server

  ```
  [root@serverX ~]# yum install -y samba
  ```

- ☐ Start and enable the Samba service

  ```
  [root@serverX ~]# service smb start
  [root@serverX ~]# chkconfig smb on
  ```

- ☐ Configure system to be in the CLASSX workgroup (where X is your station number) with local user definitions.

  Edit **/etc/samba/smb.conf** and set the following fields.

```
workgroup = CLASSX  # X is your desktop number
security = user             # default
passdb backend = tdbsam     # default
```

- ☐ Add a Samba-only user named **winuserX** (where X is your station number) with a Samba password of **winpass**.

```
[root@serverX ~]# useradd -s /sbin/nologin winuserX
[root@serverX ~]# smbpasswd -a winuserX
New SMB password:  winpass
Retype new SMB password:  winpass
...
Added user winuserX.
```

- ☐ Enable user home directory access in SELinux

```
[root@serverX ~]# setsebool -P samba_enable_home_dirs on
```

- ☐ Enable the firewall and open up necessary ports to grant access.

```
[root@serverX ~]# iptables -I INPUT -p udp --dport 137:138 -j ACCEPT
[root@serverX ~]# iptables -I INPUT -p tcp --dport 139 -j ACCEPT
[root@serverX ~]# iptables -I INPUT -p tcp --dport 445 -j ACCEPT
[root@serverX ~]# service iptables save
```

- ☐ Test the configuration, by accessing your Samba-only user's home directory from desktopX.

```
[root@serverX ~]# smbclient //serverX/winuserX -U winuserX%winpass
```

  To test using the GUI, go to **Places** → **Connect to Server**. Fill in the following fields (leave the others blank and remember to substitute your desktop number for X):

```
Service type: Windows share
Server: serverX
Share: winuserX
User Name: winuserX
Domain Name: CLASSX
```

  When prompted, enter **winpass** as the password.

Search & Learn: Configuring CIFS Shares

Consult the reference documents below to answer the following questions:

Three Steps to Provide a Group Share:

1. The first step would be to create a collaborative directory in Linux; that is, a directory writable by a particular group in which all new files are automatically owned and writable by that group. From your previous experience, what three steps are needed to accomplish this?

   Create directory:

```
[root@serverX ~]# mkdir -p /shared/dir
```

   Set ownership:

```
[root@serverX ~]# groupadd -r groupname
[root@serverX ~]# chgrp groupname /shared/dir
```

   Set permissions:

```
[root@serverX ~]# chmod 755 /shared
[root@serverX ~]# chmod 2770 /shared/dir
```

2. Next, we need to set a correct SELinux context on this directory so that it may be shared using CIFS/Samba. What two steps are needed to set the correct context persistently (across SELinux relabels)?

   Update policy:

```
[root@serverX ~]# semanage fcontext -a -t public_content_t '/shared(/.*)?'
[root@serverX ~]# semanage fcontext -a -t samba_share_t '/shared/dir(/.*)?'
```

   Reset context on existing files:

```
[root@serverX ~]# restorecon -FRvv /shared
```

Note

/shared in the example above is a top-level directory that may be shared using CIFS, NFS, FTP, etc. The **public_content_t** allows each of the services to access the top-level directory. The **/shared/dir** sub-directory is then given **samba_share_t** type that only CIFS can access.

3. How do we share this directory via Samba?

   To share this directory via Samba add the following to the bottom of **/etc/samba/smb.conf** and restart the **smb** service.

```
[dir]
  path = /shared/dir
  valid users = @groupname
  writeable = yes
```

```
  public = no
```

The previous share would only allow users of the group **groupname** to access the share. If you wanted to allow other read-only access to the share, change the Linux permissions on the directory:

```
[root@serverX ~]# chmod 2775 /shared/dir
```

...and change the share in **/etc/samba/smb.conf** to the following:

```
[dir]
  path = /shared/dir
  writeable = no
  write list = @groupname
  public = no
```

Two Steps to Provide an Individual Printer Share:

1. How do you prevent the automatic sharing of all locally defined printers as CIFS printer shares by Samba?

   To prevent the automatic sharing of all locally defined printers by Samba, remove or comment out the **printers** share in **/etc/samba/smb.conf**:

```
#[printers]
#        comment = All Printers
#        path = /var/spool/samba
#        browseable = no
#        guest ok = no
#        writable = no
#        printable = yes
```

   Alternately, change **load printers** from **yes** to **no** (the default is **yes**, so commenting out this line will not work).

2. How would you share a particular printer manually?

   To share a particular printer, add the following to **/etc/samba/smb.conf** and restart the service:

```
[myprinter]
  comment = My Printer Description
  path = /var/spool/samba
  read only = yes
  printable = yes
  printer name = cups_printer_name
```

Limiting Client System Access:

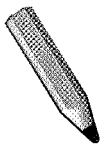
1. How can you limit which client systems can access a particular CIFS share by IP address?

   To limit which client systems can access a particular share by IP address, add something like the following to **/etc/samba/smb.conf** in the share section:

```
hosts allow = 192.168.0.1 10.2.12.
```

2. Is there a non-Samba facility that can limit which client systems can access all CIFS shares by the client's source IP address?

   iptables is a non-Samba facility that can limit which client systems can access any CIFS share by IP address.



Test

Criterion Test

Case Study

File Sharing with CIFS

Before you begin...

Run **lab-setup-samba** on desktopX to prepare serverX for the exercise.

The School of Butler and Hacker has recently deployed several CIFS servers to allow their Windows client systems access to file shares.

The Color Guard, known as Green and Red is deploying a new server and needs to share information using CIFS. That share must be writable by members of the Color Guard, but other people can only have read access.

Enable the firewall and allow all clients on the local network to access the CIFS server.

Configure your serverX to function as a CIFS server, with the following information:

- Workgroup: BUTLER
- Linux Group: greenred
- CIFS Share Name: school
- Directory: /shared/school
- No printers shared

Test the configuration by:

- Creating a user as a member of **greenred** and ensuring they can write to the CIFS share, **school**
- instructor.example.com provides several printers that CUPS should automatically enable. Before you fulfill the printer requirement, check to verify they are available (they should be named printerX). Configure Samba so that no printers are shared and confirm that the user can NOT see them listed with **smbclient**
- Creating a second user not as a member of **greenred** and ensuring they can only read from the CIFS share, **school**

When you are ready to check your work, run **lab-grade-samba** on serverX.

1. Run **lab-setup-samba** on desktopX to prepare serverX for the exercise.

```
[root@desktopX ~]# lab-setup-samba
```

2. Install, start and enable needed packages.

```
[root@serverX ~]# yum install samba samba-doc
[root@serverX ~]# service smb start
[root@serverX ~]# chkconfig smb on
```

3. Enable the firewall and add the following rules.

```
[root@serverX ~]# iptables -A INPUT -p udp --dport 137:138 -j ACCEPT
[root@serverX ~]# iptables -A INPUT -p tcp --dport 139 -j ACCEPT
[root@serverX ~]# iptables -A INPUT -p tcp --dport 445 -j ACCEPT
[root@serverX ~]# service iptables save
```

4. Create the Linux group and directory and configure SELinux support.

```
[root@serverX ~]# groupadd -r greenred
[root@serverX ~]# mkdir -p /shared/school
[root@serverX ~]# chgrp greenred /shared/school
[root@serverX ~]# chmod 2775 /shared/school
[root@serverX ~]# semanage fcontext -a -t public_content_t '/shared(/.*)?'
[root@serverX ~]# semanage fcontext -a -t samba_share_t '/shared/school(/.*)?'
[root@serverX ~]# restorecon -vvFR /shared
restorecon reset /shared context unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
restorecon reset /shared/school context unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
```

5. Edit **`/etc/samba/samba.conf`**.

   Modify or confirm the following:

```
[global]
...
 workgroup = BUTLER
...
 security = user
 passdb backend = tdbsam
```

   Add a section (at the end of the file) as follows.

```
[school]
 path = /shared/school
 write list = @greenred
 read only = yes
 guest ok = no
```

   Comment or remove the following lines, for printers.

```
[global]
...
load printers = no
```

```
-OR-
#[printers]
#       comment = All Printers
#       path = /var/spool/samba
#       browseable = no
#       guest ok = no
#       writable = no
#       printable = yes
```

6. Restart samba.

```
[root@serverX]# service smb restart
```

7. Test as mentioned above.

```
[root@serverX ~]# useradd -s /sbin/nologin -G greenred red
[root@serverX ~]# smbpasswd -a red
New SMB password:  red
Retype new SMB password:  red
Added user red.
[root@serverX ~]# useradd -s /sbin/nologin bernice
[root@serverX ~]# smbpasswd -a bernice
New SMB password:  bernice
Retype new SMB password:  bernice
Added user bernice.
[root@serverX ~]# smbclient -L serverX -U red%red | grep printer
Domain=[BUTLER] OS=[Unix] Server=[Samba 3.5.4-68.el6]
Domain=[BUTLER] OS=[Unix] Server=[Samba 3.5.4-68.el6]
[root@serverX ~]# smbclient //serverX/school -U red%red
smb: \> put /etc/hosts hosts
smb: \> ls
  hosts                               A      177  Tue Dec 21 10:10:03 2010
smb: \> exit
[root@serverX ~]# smbclient //serverX/school -U bernice%bernice
smb: \> mkdir test
NT_STATUS_MEDIA_WRITE_PROTECTED making remote directory \test
smb: \> get hosts
getting file \hosts of size 177 as hosts (57.6 KiloBytes/sec) (average 57.6 KiloBytes/
sec)
```

File Sharing with FTP

FTP Buzz Groups

Using the documentation references below, research the answer to one or more of the below assigned questions. Take notes for the remaining unassigned questions when the class re-groups.

1. ***Create upload directory***

   What filesystem permissions could create a "drop-box"?
   - Group ownership: **ftp**
   - Permissions: group **ftp** has write and execute, but not read access; "other" has no access

2. ***Modify SELinux for anonymous upload***

   Are there appropriate contexts and booleans to set?
   - File/directory type context: **public_content_rw_t**
   - Boolean: **allow_ftp_anon_write** must be enabled

3. ***Modify /etc/vsftpd/vsftpd.conf***

   What needed anonymous-related options are there?
   - **anon_upload_enable = YES**
   - **chown_uploads = YES**
   - **chown_username = daemon**
   - **anon_umask = 077** (default value)

4. ***Modify iptables to support inbound ftp connections***

   What ports does ftp use? How do we support passive ftp?
   - **/etc/sysconfig/iptables-config** change:

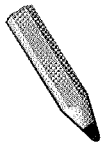
     ```
     IPTABLES_MODULES="nf_conntrack_ftp nf_nat_ftp"
     ```

   - Open new connections to TCP port 21 and allow ESTABLISHED and RELATED network traffic:

     ```
     # iptables -I INPUT -p tcp --dport 21 -j ACCEPT
     # iptables -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
     ```

Note

Remember that **iptables** rules are processed sequentially. The use of **-I** ensures that these rules appear before any default, global **REJECT** rule.



Test

Criterion Test

Case Study

FTP Drop Box

Before you begin...

Run **lab-setup-dropbox** on desktopX to prepare serverX for the exercise.

The Quiet Pleases company, a manufacturer of silence cones and other noise canceling devices, has a program to collect information about noise levels around the world. Volunteers have been collecting data about noise and need an easy way to send in reports.

The company has decided to use an FTP server with an anonymous upload directory to collect the reports.

Deploy vsftpd on your serverX and configure a write-only upload directory that is accessible at: `ftp://serverX.example.com/dropbox`

As the volunteers are located all over world, the FTP server must accept connections from anywhere on the internet.

When you are ready to check your work, run **lab-grade-dropbox** on desktopX.

1. Run **lab-setup-dropbox** on desktopX to prepare serverX for the exercise.

```
[root@desktopX ~]# lab-setup-dropbox
```

2. Install, start and enable needed packages.

```
[root@serverX ~]# yum install vsftpd
[root@serverX ~]# service vsftpd start
[root@serverX ~]# chkconfig vsftpd on
```

3. Create upload directory.

```
[root@serverX ~]# mkdir /var/ftp/dropbox
[root@serverX ~]# chgrp ftp /var/ftp/dropbox
[root@serverX ~]# chmod 730 /var/ftp/dropbox
```

4. Configure SELinux support.

```
[root@serverX ~]# semanage fcontext -a -t public_content_rw_t '/var/ftp/dropbox(/.*)'
[root@serverX ~]# restorecon -vvFR /var/ftp/dropbox
restorecon reset /var/ftp/dropbox context unconfined_u:object_r:public_content_t:s0-
>system_u:object_r:public_content_rw_t:s0
[root@serverX ~]# setsebool -P allow_ftpd_anon_write on
```

5. Edit **/etc/vsftpd/vsftpd.conf**.

   Modify, uncomment or confirm the following:

```
anon_upload_enable=yes
chown_uploads=yes
chown_username=daemon
anon_umask=077
```

   Restart **vsftpd**.

```
[root@serverX ~]# service vsftpd restart
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
```

6. Configure iptables (if enabled).

   Edit **/etc/sysconfig/iptables-config**:

```
IPTABLES_MODULES="nf_conntrack_ftp nf_nat_ftp"
```

   Edit **/etc/sysconfig/iptables**:

```
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p tcp --dport 21 -j ACCEPT
```

Note

Take care that the above two lines are placed above any global **REJECT**, or, if using the **iptables** command, perhaps use **-I** rather than **-A**.

Restart **iptables**.

```
[root@serverX ~]# service iptables restart
```

Troubleshooting the Boot Process

Practice Exercise

Using the Rescue Environment

Before you begin...

Run **lab-setup-bootbreak** on desktopX.

```
[root@desktopX ~]# lab-setup-bootbreak
```

This timed drill is designed to give you practice accessing the rescue environment. The installation path is *http://instructor.example.com/pub/rhel6/dvd*. The path for individual packages is *http://instructor.example.com/pub/rhel6/dvd/Packages/*

Perform the following steps on serverX unless directed otherwise.

1. After serverX has booted, run the **lab-setup-bootbreak-0** script as **root** on serverX. This script will alter your system and cause difficulties booting.

   ```
   [root@serverX ~]# lab-setup-bootbreak-0
   ```

   Try booting the system without the kernel arguments of **rhgb quiet**. Note the following error messages (Shift-PgUp can be used to scroll the screen back up):

   ```
   /etc/rc.d/rc.sysinit/: line 26: mount: command not found
   readahead: starting
                    Welcome to Red Hat Enterprise Linux Server
   ...
   Remounting root filesystem in read-write mode:  /etc/init.d/functions: line 536:
    mount: command not found
   ...
   ```

2. Boot into the rescue environment to diagnose and resolve the issue.

   - Boot into rescue mode.

     Rather than booting our virtual machine from an Installation DVD, cause it to boot from the network by pressing **Ctrl+B** at the gPXE dialog.

     ```
     gPXE> autoboot
     ```

     Choose "Rescue installed system"

     Choose Language and Keyboard Type as appropriate

     Choose "URL" for Rescue Method

     Use DHCP for IPv4 only

     http://instructor.example.com/pub/rhel6/dvd for URL Setup

     "Continue" to mount your hard drive under **/mnt/sysimage**

Finally, open a "shell"

- Verify problem based on earlier error message:

```
bash-4.1# chroot /mnt/sysimage
sh-4.1# mount
sh: mount: command not found
sh-4.1# yum provides /bin/mount
... output omitted ...
sh-4.1# yum reinstall util-linux-ng
... output omitted ...
sh-4.1# mount
... output omitted ...
```

3. Confirm the problem has been solved by rebooting the system.

```
sh-4.1# exit
exit
bash-4.1# exit
```

Choose "reboot"

4. Repeat this process as often as possible during the allotted time.

Practice Quiz

Repairing Boot Problems

1. In maintenance mode, run mount -o remount,rw / to mark the / partition as writable.
2. Assuming you have only one hard drive, and the first partition contains /boot, if you have minor MBR corruption, fix the corruption issue by booting into rescue mode, then run the grub command. Sometimes you have to type the **device (hd0) /dev/vda** command to identify the disk you want to update. Type root (hd0,0) followed by setup (hd0), then exit.
3. If you have file system corruption issues, the machine will boot into maintenance mode.
4. In maintenance mode, run fsck to fix file system corruption issues.

Search & Learn: Configure a Serial Console

Your goal is to determine how to configure a Red Hat Enterprise Linux 6 machine to use a serial console. It should do the following:

- Use the serial console as the system console, outputting kernel and boot messages there
- Allow logins on the serial console

Use the **/usr/share/doc/kernel-doc/Documentation/serial-console.txt** and **/etc/init/serial.conf** files for reference.

When reading **serial-console.txt**, focus on kernel command line arguments. Do not worry about recompiling your kernel or making your own device nodes; this has been done for you already in Red Hat Enterprise Linux. Note that much of the setup discussion for **init** is for SysVinit, which was shipped in older versions of Red Hat Enterprise Linux but is not relevant in Red Hat Enterprise Linux 6.

Add **console=ttyS0** to the end of the **kernel** line in the **/boot/grub/grub.conf** GRUB configuration file and reboot.

Practice Performance Checklist

Configuring a Serial Console

In this practice exercise, you will use the **minicom** terminal program on desktopX to act as a VT100 terminal connected to serverX.

Perform the following steps on serverX unless directed otherwise.

- ☐ Configure serverX to send console messages to the first serial port.

  As root on your server machine, edit **/boot/grub/grub.conf**, adding **console=ttyS0** to your kernel command line.

- ☐ Determine the source path associated with the virtual serial line. Record the source path in the space provided.

  As root on desktopX, open **virt-manager** and connect to vserver.

  View the "Details" panel by either clicking the "[i]" icon, or selecting the **View** → **Details** menu item.

  Select your serial device, and determine the source path associated with the virtual serial line. Record the source path in the space provided.

  The source path represents the terminal on your virtualization host which can be used to access the serial line on your virtualization guest. Think of it as the "serial port" for your guest.

- ☐ Install the **minicom** package on desktopX. Invoke **minicom**, providing the terminal from the previous step as the argument to the **-p** option.

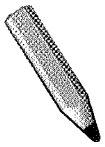
```
[root@desktopX ~]# yum install -y minicom
```

  Note that the **Ctrl+a**, **z** key sequence invokes a configuration menu. **Ctrl+a**, **x** key sequence exits **minicom**.

  From a terminal as root on desktopX, use **minicom** to mimic a VT100 terminal.

```
[root@dsk; ~]# minicom -p /dev/pts/1
```

- ☐ While watching the **minicom** terminal, reboot serverX. As serverX reboots, you should be able to observe startup messages in the minicom terminal.
- ☐ When the startup process completes, you should be able to log into your machine using the minicom terminal.
- ☐ When finished, edit **/boot/grub/grub.conf**, removing the **console** kernel command line parameter, and reboot serverX.



Test

Criterion Test

Exercise

Troubleshooting the Boot Process

Before you begin...

Run the **lab-setup-bootbreak** command on desktopX to setup the lab.

```
[root@desktopX ~]# lab-setup-bootbreak
```

This practice exercise includes three break/fix challenges. For each, your serverX virtual machine will be modified in some way that prevents it from booting correctly and you will diagnose and correct the problem.

Perform the following steps on serverX unless directed otherwise.

1. Run **lab-setup-bootbreak-1** on serverX. After running the script, serverX should no longer boot correctly. Diagnose and correct the problem. You will know you have the solution when serverX boots normally again.

   ```
   [root@serverX ~]# lab-setup-bootbreak-1
   ```

   lab-setup-bootbreak-1 replaces **UUID** with **UID** in **/etc/fstab**. Notice that **vim** will highlight that error in red. To fix the issue, make the **/** file system writable, then change **UID** to **UUID** in **/etc/fstab**.

2. When you have resolved the first scenario, repeat the process with **lab-setup-bootbreak-2** and **lab-setup-bootbreak-3**.

   ```
   [root@serverX ~]# lab-setup-bootbreak-2
   ```

   lab-setup-bootbreak-2 changes the default runlevel to 9. To fix this issue, boot into a standard runlevel (by adding **3** to the kernel line, for example), then edit **/etc/inittab** and change **id:9:initdefault:** to **id:3:initdefault:**

   ```
   [root@serverX ~]# lab-setup-bootbreak-3
   ```

lab-setup-bootbreak-3 introduces a misspelling in **/boot/grub/grub.conf**. To fix this issue, edit the kernel line from the grub prompt and change **rot=** to **root=**. Once the system boots, make the same change in **/boot/grub/grub.conf**.