Red Hat

RH253

Network Security Services
and Security Administration

RH253-RHEL4-1-20050301

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

RH253

Red Hat Network Services and Security Administration

RH253-RHEL4-1-20030301

Table of Contents

RH253
Red Hat Network Services and Security Administration

Table of Contents - Labs

Welcome!

RH253

Red Hat Network Services and Security Administration

redhat

1

Rev RH253-RHEL4-1

Welcome to RH253

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

redhat 2

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Phone and network availability

Please only make calls during breaks Your instructor will show you which phone to use

Network access and analogue phone lines may be available; your instructor will provide information about these facilities.

Please turn pagers to silent and cell phones off during class.

Restrooms

Your instructor will notify you of the location of these facilities

Lunch and breaks

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

In case of Emergency

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

Access

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

Participant Introductions

Please introduce yourself to the rest of the class!

redhat

3

Rev RH253-RHEL4-1

Introduction

RH253

Red Hat Network Services
and Security Administration

redhat

1

Rev RH253-RHEL4-1

Copyright

- The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2005 Red Hat Inc.
- No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic electronic or other record without the prior written permission of Red Hat Inc.
- This curriculum contains proprietary information which is for the exclusive use of customers of Red Hat, Inc. and is not to be shared with personnel other than those in attendance at this course.
- This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.
- If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please email training@redhat.com or phone toll-free(USA) +1 866 626 2994 or +1 919 754 3700.

redhat 2

Red Hat Enterprise Linux

- Enterprise-targeted operating system
- Focused on mature open source technology
- 12-18 month release cycle
  - Certified with leading OEM and ISV products
- Purchased with one year Red Hat Network subscription and support contract
  - Support available for seven years after release
  - Up to 24x7 coverage plans available

redhat 3

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

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

Red Hat Enterprise Linux is released on a twelve to eighteen month cycle. It is based on code developed by the open source community and adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors such as Dell, IBM, Fujitsu, BEA, and Oracle. The longer release cycle allows vendors and enterprise users to focus on a common, stable platform and to effectively plan migration and upgrade cycles. Red Hat Enterprise Linux provides a high degree of standardization through its support for seven processor architectures: Intel x86-compatible, Intel Itanium 2, AMD AMD64/Intel EM64T, IBM PowerPC on eServer iSeries and eServer pSeries, and IBM mainframe on eServer zSeries and S/390

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

Red Hat Enterprise Linux AS: the top-of-the-line Red Hat Enterprise Linux solution, this product supports the largest x86-compatible servers and is available with the highest levels of support

Red Hat Enterprise Linux ES: for entry-level or mid-range departmental servers Red Hat Enterprise Linux ES provides the same core capabilities as AS, for systems with up to two physical CPUs and up to 8 GB of main memory.

Red Hat Enterprise Linux WS: the desktop/client partner for Red Hat Enterprise Linux AS and Red Hat Enterprise Linux ES on x86-compatible systems. Based on the same development environment and same software core as the server products, Red Hat Enterprise Linux WS does not include some network server applications. It is ideal for desktop deployments or use as a compute node in a HPC cluster environment.

Red Hat Network

- A comprehensive software delivery, system management, and monitoring framework
  - **Update Module,** included with Red Hat Enterprise Linux, provides software updates
  - **Management Module** adds more scalable management capabilities for large deployments
  - **Provisioning Module** provides bare metal installation, configuration management, and multi-state configuration rollback capabilities

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

About Red Hat Network

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

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

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

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

Red Hat Desktop

- High-quality, full-featured client system based on Red Hat Enterprise Linux
  - Includes desktop productivity applications
- Available in packages of 10 or 50 units for mass deployments of desktop systems
- Clients entitled to RHN Management Module
  - Package may also include RHN Proxy Server or Satellite Server

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

About Red Hat Desktop

Red Hat Desktop is the latest addition to the Red Hat Enterprise Linux product family. It provides a high-quality, full-featured client system suitable for use in a wide range of desktop deployments. Red Hat Desktop includes integrated third-party applications including Adobe Acrobat Reader and plugin, Macromedia Flash plugin, Citrix ICA client, Java (IBM and BEA) and plugin (IBM), and Real Player. Red Hat Desktop shares the same primary product features as the rest of the Red Hat Enterprise Linux family, including a twelve to eighteen month release cycle and one year of bundled software updates and support (annually renewable for the life of the product).

Red Hat Desktop provides mechanisms to help manage and secure large desktop deployments It is available in packages of either 10 or 50 units for mass deployments of consistently managed clients. Client systems are bundled with Red Hat Network Management Module entitlements for improved manageability. In addition, packages may include either Red Hat Network Proxy Server or Red Hat Network Satellite Server (with a Red Hat Enterprise Linux AS, Premium Edition entitlement) to ensure the highest levels of manageability and security.

Red Hat Desktop is currently available for client systems based on either the Intel x86 or the AMD AMD64/Intel EM64T processor architecture with a single CPU and up to 4 GB of main memory.

Red Hat Applications

- Open source applications provided separately from Red Hat Enterprise Linux
- Include a range of support options
- Installation media and updates provided through Red Hat Network
- More information on specific products at http://www.redhat.com/software/rha/

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Red Hat Applications

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

For more information on specific products which are currently available, please visit

`http://www.redhat.com/software/rha/`

The Fedora Project

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

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

About the Fedora Project

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

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

Objectives of RH253

- Learn skills of the system administrator who can configure Red Hat Enterprise Linux common network services and security at a basic level

8

Rev RH253-RHEL4-1

A person who has completed RH253 will have practiced network service configuration and security administration tasks The services include:

- DNS
- DHCP
- NFS
- FTP
- SMB
- HTTP
- `sendmail/postfix`
- `xinetd`
- Packet Filtering and Connection Tracking

Audience and Prerequisites

- Audience: Linux or UNIX operators who can perform system administration tasks to a level where he/she can install, configure, and attach a new Red Hat Linux workstation to an existing network.
- Prerequisites: experience in Linux or UNIX administration at the single-workstation level

redhat 9

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Audience for RH253:

The Red Hat Network Services and Security Administration course is designed for UNIX- and Linux-experienced system administrators who want a first course in networking services and security.

Prerequisites for RH253 include knowledge in the following areas:

- Installing Red Hat Linux
- Creation and maintenance of the Linux filesystem
- User and group administration
- Integrating a workstation as a client to network services, including NIS and DHCP
- Configuring a workstation as an NFS client
- Configuring X, Gnome, and KDE
- Performing basic performance, memory and process management
- Performing basic troubleshooting
- Configuring basic host security
- TCP/IP Networking Fundamentals

Classroom Network

	Names	IP Addresses
Our Network	example.com	192.168.0.0/24
Our Server	server1.example.com	192.168.0.254
Our Stations	station*X*.example.com	192.168.0.*X*
Their Network	cracker.org	192.168.1.0/24
Their Server	server1.cracker.org	192.168.1.254
Their Stations	station*X*.cracker.org	192.168.1.*X*

(For Stations, *X* is a number between 1 and 20)

UNIT 1

Introduction to System Services

redhat

1

Rev RH253-RHEL4-1

Objectives

- Understand how services are managed
- Learn common traits among services
- Introduce service fault analysis methods

Agenda

- Service management concepts
- System V-managed services
- **`xinetd`** managed services
- The **`/etc/sysconfig`** files
- Fault Analysis

3

Service Management

- Services are managed several ways:
  - by `init`
  - by System V scripts
  - by direct command
  - by `xinetd`

Services Managed by `init`

- Typically non-TCP/IP services, for example dial-in modems
- Provides respawn capability
- Configured in `/etc/inittab`

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`init` can automatically respawn a program that terminates The management of the X Window System at run level 5 is an example of this type of service.

Services involving dial-in modems or serial ports (for example, dumb terminals) are under `init` control. See `inittab(5)` for more information on the format Changes to `inittab` can be activated with `init q`.

System V Service Management

- Processes are "wrapped" by System V ('SysV') initialization script methods
- More than one script, and several configuration files are often used, per service
- The **`service`** command is a "wrapper of wrappers"
  - **`/etc/init.d/cups start`**
  - **`service cups start`**

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Invoking the service script can be done directly, or by use of the `service` command

Some services have more than one daemon managed by the `service` script.

The name of the script and the name of the daemon(s) it starts are often similar, but not always

Invoking the service program in this way is not persistent across reboots To ensure a persistent change between reboots, the daemon's start or kill symbolic link in `/etc/rc.d/rc[0-6].d/` must be changed

chkconfig

- Manages service definitions in run levels
- To start the **cups** service on boot:
  `chkconfig cups on`
- Does not modify current run state of System V services
- List run level definitions with
  `chkconfig --list`

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Using `chkconfig`, a service definition persists across reboots
`chkconfig` also manages `xinetd` services.

```
chkconfig cups --list
cups      0:off    1:off    2:off    3:on   4:off   5:on   6:off
```

`chkconfig <service> on` enables the service in runlevels 2, 3, 4, and 5

`chkconfig <service> off` disables the service in runlevels 2, 3, 4, and 5

`chkconfig <service> --add` ensures that either a *kill* or a *start* symbolic link is set for every runlevel

`chkconfig <service> --del` removes a service from `chkconfig` management

xinetd Managed Services

- Services are started by **xinetd** in response to incoming request
- Activated with **chkconfig**: **chkconfig cups-lpd on**
- Uses files in **/etc/xinetd.d/**

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`xinetd` is itself managed from a System V script. It monitors the ports used by all the services under its care, and starts services in response to incoming connections

`chkconfig` [de]activates installed `xinetd` services:

```
chkconfig cups-lpd on
```

This use of `chkconfig` does not start or stop the running instance of the service.

The use of `chkconfig` for `xinetd` services has no relation to run levels, except to the extent that `xinetd` must be running, which may be dependent on run level.

The **xinetd** daemon

- Manages network-specific resources and authentication
  - less-frequently needed services
  - host-based authentication + Time based Auth.
  - service statistics and logging
  - service IP redirection
- Replaces **inetd** (ubuNTu + DEBIAN)
- Linked with **libwrap.so**
- Configuration files: **/etc/xinetd.conf**, **/etc/xinetd.d/*service***

TELNET

redhat 9

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Services which are needed less frequently, or requiring additional resource management, are typically controlled by the `xinetd` daemon This daemon provides host-based authentication, resource logging, timed access, and address redirection among its many configuration options. These options may be service-specific, or generally applied across all `xinetd`-managed services. `xinetd` uses `/etc/services` in its configuration of port-to-service management

Red Hat Enterprise Linux(RHEL) `xinetd` is compiled with `libwrap` support This will cause `xinetd` to check for the names of all services that it spawns with `hosts.allow` and `hosts.deny`, regardless of whether or not the individual services were compiled with `libwrap.so` `xinetd` has its own internal security mechanism as well, which is described later in this unit `hosts.allow` and `hosts.deny` will be consulted first when a service is requested. If they allow access, then `xinetd`'s internal access control policies are checked.

The default installed configuration of `xinetd` is provided by the top-level configuration file `/etc/xinetd.conf` and service specific files under the `/etc/xinetd.d` directory tree

More information may be found at `http://www.xinetd.org`, the `xinetd-*` directory under `/usr/share/doc`, and `man xinetd.conf`.

xinetd default controls

- Top-level configuration file
  - /etc/xinetd.conf

The top-level configuration file /etc/xinetd.conf sets the global configuration options shared by all managed services. It also provides the path to service specific configurations Below is an annotated version of the default installed top-level configuration file

```
defaults
{
     instances              = 60 [No. of connections]
     log_type               = SYSLOG authpriv [security + login] /var/log/secure
     log_on_success         = HOST PID
     log_on_failure         = HOST
     cps (connections per second) = 25 30
}

includedir /etc/xinetd.d
```

xinetd service controls

- Service specific configuration
  - /etc/xinetd.d/<service>

redhat 11

Rev RH253-RHEL4-1

Below is the default service specific configuration file for `cups-lpd`. Note the first line (which appears "commented"): this boolean value determines whether the service is active. Service configuration utilities, like `ntsysv` and `chkconfig`, will edit the appropriate `xinetd` service configuration files for a given runlevel

```
# default: off
# description: Allow applications using the legacy lpd protocol to
# communicate with CUPS//service printer
{
        disable = yes
        socket_type = stream
        protocol = tcp
        wait = no
        user = lp
        server = /usr/lib/cups/daemon/cups-lpd
}
```

The /etc/sysconfig files

- Some services are configured for *how* they run
  - named
  - sendmail
  - dhcpd
  - samba
  - init
  - syslog

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Many files under /etc/sysconfig describe hardware configuration; however, some of these files configure service run-time parameters. By contrast, /etc/init.d/ files are executable scripts that configure the conditions of daemon execution, while those in /etc/sysconfig/ configure the manner of daemon execution

In the following units, we will discuss all the services listed in the slide above. Although we may not directly involve their run-time configuration here, you will find good references for them in /usr/share/doc/initscripts-<version>/sysconfig.txt, and service specific documentation. For example, the elsewhere in this course, we introduce the Berkeley Internet Domain(BIND) daemon We are limited here to discuss all BIND run-time configuration subtleties, and the "chrooted jail" method for running the named daemon is one such subtlety: to learn more about this, visit http://www.tldp.org/HOWTO/Chroot-BIND-HOWTO.html.

Fault Analysis

- Determine the severity of the fault
  - Is it the data?
  - Is it the program or application? rpm -V
  - Is it the operating system?
  - Is it the hardware?
- Inspect logs *before* configuration files
- Use command options for debugging
- Document your investigation

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Throughout this course you will be setting up a number of different services under Red Hat Enterprise Linux. More than likely, somewhere along the way a typo will be made, a file will be misplaced and the service will not work as expected. This is, ironically, to be expected and will provide you with practical hands-on troubleshooting experience. Self-induced problems, while frustrating, at least have the benefit of giving you some clue as to which component is faulty (what was the last thing you changed before the trouble started?). During lab exercises in this course, you may also be asked to run scripts that break some component of the system, provide you with a description of the resulting symptoms and then leave it to you to do the rest. This will provide valuable experience with troubleshooting service faults you did *not* have a hand in creating.

When analyzing a service (or system) fault, consider its severity. Look at, and read, error messages displayed, when presented to you. These messages may be in a log file too. They will very, *very* often describe the kind of fault encountered. It might be bad data, or the wrong data given to your program. Your program could be faulty. The operating system could be at fault, or your hardware. It is possible, but rare, that a combination of these potential causes would occur at the same time; however, a bad disk sector (hardware) may result in bad data(some part of a configuration file), which in turn results in a program's failure to start: no DNS services.

The process of troubleshooting any system, including those running RHEL, is both science and art. The science comes from the concepts of hypothesis testing, experimentation, comparison, and reproducing results. The art of troubleshooting comes from the realization that operating systems, services, and applications do not always work as we hope or anticipate, or even as their creators hope or anticipate. Science allows us to focus on likely causes, while art permits us consider the off-the-wall and unlikely as possibilities.

Regardless of whether the problem is something wrong in a single user's environment or a system-wide crisis that has rendered a system unusable, sensible troubleshooting begins with the *easy* fixes. This may mean running a configuration tool, or it may mean looking at the system or service-specific log. Logs often provide explicit information on problems, sometimes even identifying the exact line of a configuration file that is causing problems, so it is often far easier to look in a log for an answer then to parse a configuration file for what might be a trivial syntax error. Many services provide switches that enable higher levels of debugging output, and some may be run as foreground applications for debugging purposes. Some services, such as BIND, provide syntax checkers that may also prove useful. Begin looking in configuration files *after* you have exhausted these possibilities.

Also, develop a method which includes *documenting* your procedure... even if it's only a simple list, mark your trail!

Security Enhanced Linux

- Who can do what to which files?
  - Mandatory access control (SELinux)
    - Under the control of the security administrator
  - Discretionary access control (Traditional Linux)
    - Under the control of the user

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Security Enhanced Linux (SELinux) has been around for some time, but has recently seen a rise in popularity. It was developed by NSA, as a research project Linux was chosen because it is open source, and therefore easier toget people involved. It also makes it easier to prove the technology.

Traditional Linux leaves access control to the user. The user decides who can access their files, based on group membership. With the ACL option turned on in the filesystem, users can fine grain this control. However, as long as an attacker gets access to the account, he/she can change these options at will.

With SELinux we introduce a concept called Mandatory Access Control (MAC) With MAC, we let the security administrator decide who can do what to which files Users are put in domains, and even if a user wants to share a file with another user, this will fail unless they are both in the same domain.

A security administrator can decide that normal users cannot see certain files. If a user account is compromised, the attacker cannot access these files, unless they compromise an account which has access to them. This is due to the fact that even root accounts can belong to different domains, and a change in roles is necessary to get access to different files. This change of roles requires a password.

This is fundamentally different from traditional Linux, where an attacker can easily break the security put in place. Without knowing any passwords, just by knowing vulnerabilities in services running on the machine, an intruder can gain root access.

The performance impact of running SELinux is about 7%, so consider whether the security is needed, before you turn it on.

SELinux

- Each process or object(file,directory, network socket) also has a SELinux context
  - identity:role:domain/type
- The SELinux policy controls
  - What identities can use which roles
  - What roles can enter which domains
  - What domains can access which types

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

SELinux adds another layer of access control permissions on top of standard file permissions and ACLs, which are defined by the system's security policy. Each process or object (such as a file, directory, or network socket) on the system also has a SELinux security context. This context consists of a SELinux user identity, a role, and a domain (for processes) or a type (for objects).

The policy controls what SELinux identity a process is assigned when it starts. The identity determines which roles are accessible to the process, and the roles are used to determine which domains the process can switch to. Once running in a particular domain, the policy also determines what access a process will have to objects of particular SELinux types. The policy also determines the default type for a new object when it is created.

In general, the access a process will be granted to an object is determined by the domain of the process and the type of the object being accessed.

The default policy is set by the contents of the selinux-policy-targeted RPM.

SET ENFORCE=0

SELinux Installation Options and Control

- Installation Options
  - Disabled
  - Warn (Permissive)
  - Active (default) (Enforcing)
- Control Options when SELinux is enforced
  - Targeted (default)
  - Strict

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

During installation, SELinux is automatically activated. This is done on the same screen as firewalling. There are three options to choose from:

- `Disabled`: This turns enforcing off, which means that labelling and domains are not set up. This is the most efficient way of running your machine, but less secure.
- `Warn`: This option sets up policies and logging, so you can monitor what is happening in the machine, without actually running SELinux. This enables the possibility of writing new rules for testing purposes.
- `Active`: SELinux is now enforced, but it will only affect certain daemons. When active is chosen, only some daemons will be under control of SELinux, the "strict" option will affect all daemons.

To change between enforcing and permissive mode, you can do that either at boot time, at runtime or you can make it permanent:

- During boot, add "`enforcing=1`" to the kernel line to turn on,
  "`enforcing=0`" turns it off.
- At run time "`setenforce=1`" turns SELinux into enforcing mode,
  "`setenforce=0`" turns it permissive
- To make it permanent either edit:
  `/boot/grub/grub.conf`: edit the file and add "`enforcing=1`" to the kernel line to turn on,
  "`enforcing=0`" turns it off

OR

`/etc/sysconfig/selinux`: this file is well documented to help you choose the right option.

To fine tune your security settings, you can also use `system-config-securitylevel`

Controlling SELinux

- `system-config-securitylevel`
- `setenforce` and `setsebool`
- `/etc/sysconfig/selinux`
- `enforcing=0`
- `/selinux` virtual filesystem

redhat 17

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The SELinux policy may be adjusted or disabled through a number of utilities The easiest to use is the graphical `system-config-securitylevel` tool, which can turn on or off SELinux or place it into a warn-only mode. It also allows the adjustment of "booleans" which can fine-tune the rules enforced by the policy

SELinux enforcement may also be changed from on to warn-only and vice versa with the `setenforce` command-line tool The file `/etc/sysconfig/selinux` can be edited to make enforcement changes persist across reboot. The `setsebool` command-line tool can be used to adjust and save booleans, and `sestatus` will print out the current state of SELinux to standard output.

The kernel option `enforcing=0` can be passed through GRUB at boot time to put the kernel in warn-only mode; `enforcing=1` puts it in enforcing mode.

The `/selinux` virtual filesystem is similar to `/proc` and `/sys`. It presents information about the state of SELinux in the kernel to user programs like the ones above

SELINUX = 0 (off)

SELinux Contexts

- List process contexts: **ps -Z**
- List file contexts: **ls -Z**
- Change file contexts: **chcon**
  - **chcon -t http_sys_content_t index.html**
  - **chcon --reference=/var/www/html index.html**

redhat 18

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Process contexts can be listed by adding the `-Z` option to the ps command. Most processes run in the `unconfined_t` domain, which means that they are not restricted by the default SELinux policy. Processes running in other domains are most likely restricted by the default policy. Services affected include `dhcpd`, `httpd`, `mysqld`, `named`, `nscd`, `ntpd`, `portmap`, `postgres`, `snmpd`, `squid`, `syslogd`, and `winbindd`.

The security contexts of files can be displayed through the `ls -Z` command. A newly created file is assigned the context of its parent directory unless the policy specifies otherwise. The `chcon` command changes the context of a file, and works much like `chown` and `chmod`. Generally, the type of a file is the critical part of its context to set. To recursively set the type of all files in `/var/www/html` to `httpd_sys_content_t`, without chasing symlinks, run the command

```
chcon -R -t httpd_sys_content_t -h /var/www/html
```

The `--reference` option can be used with chcon to apply the current SELinux context of one file or directory to another file or directory.

Troubleshooting SELinux

- What is the error?
  - Check **/var/log/messages** for **avc** denials
- Is the process doing something it should not?
- Does the target have the right context?
- Does a "boolean" setting need adjustment?

redhat 19

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

SELinux policy violations are logged to `/var/log/messages`. An error might read like the following:

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

Role Type

This translates as:

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

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

Resources that can help troubleshoot SELinux problems include the `Red Hat Enterprise Linux 4: Red Hat SELinux Guide` on *www.redhat.com*, and `Understanding and Customizing the Apache SELinux Policy for Fedora Core 3` at *fedora.redhat.com*. The source used to build the SELinux policy is included in the selinux-targeted-policy-sources RPM.

End of Unit 1

- Address questions
- Preparation for Lab 1
  - Goals
  - Sequences
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 20

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 1
Introduction to Network Services

Estimated Duration: **30 minutes**

Goal: To become acquainted with your system.

Sequence 1: Inspecting your system

Scenario/Story:

You have "inherited" this system. The root password is "redhat." Enjoy

Task:

Before you use the system, inspect its configuration by answering the following questions with the command(s) you used to derive this answer. Consider *how* you derived your answers.

1. What is the IP address of the system? 192.168.0.9
2. How was this address configured? DHCP. /etc/sysconfig/network-scripts/ifcfg-eth0
3. What runlevel is the system currently in? 5
4. How was this configured? grub INITTAB 5
5. Which services, if any, are currently offered? dnt, chargen, etc.
6. How many "end user" accounts are there? 1
7. Which account are you now using? Jeff
8. How did you get access to the system? Login.
9. Has anyone else logged into your system recently? root

Sequence 2: Troubleshooting Practice

Scenario/Story:

Now that you have become a bit more familiar with the system, let's break it. This is a controlled environment: there are no "hints" in the "Real World" ... unless your intruder is interested in a game of "Cat and Mouse."

Task:

Practice correcting a problem with RHEL system.

1. Disable the system's "firewall," if neccessary.

   ```
   service iptables stop
   ```

2. Run the following command, following the instruction displayed:

   ```
   tsnetwork 2
   ```

3. This command will set up the problem and will explain the goal. Refer to the file `/etc/ts` to review the goal. Spend three to eight minutes trying to solve the problem.

4. If you have not yet solved the problem, you may need a hint. Hints can be displayed by running the `tshint` command:

   ```
   tshint network 2 1
   ```

   This will display the first hint for the first `tsnetwork` problem. Continue to invoke hints until you get enough information to solve the problem or until you run out of hints:

   ```
   tshint network 2 2
   tshint network 2 3
   ```
   [and so on ...]

   The `tshint` command will tell you when you have reached the end of the hints. Again, do not spend more than five to ten additional minutes on this problem.

5. Whether or not you have solved the problem, run the `tslesson` command:

   ```
   tslesson network 2
   ```

   This command will tell the lesson(s) intended to be taught by the problem.

6. If, after reading the hints and the lesson, you are unable to solve the problem, ask the instructor for assistance.

netconfig ifup
ifdown

UNIT 2

Organizing Networked Systems

redhat

1

Rev RH253-RHEL4-1

Objectives

- Explain networked systems organization
- Describe the Domain Name System (DNS)
- Explain the BIND DNS service
- Learn how to configure BIND
- Understand BIND utilities
- Explain the DHCP service

Agenda

- DNS operational overview
- Configuring BIND
- Creating BIND databases
- Additional DNS methods
- Using BIND tools
- Configure DHCP services

redhat 3

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Domain Name System(DNS)

- Resolves hostnames into IP addresses (forward lookup)
- Resolves IP addresses into hostnames (reverse lookup)
- Allows machines to be logically grouped by name *domains*
- Provides email routing information

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

DNS makes it possible to refer to Internet Protocol (IP)-based systems (*hosts*) by human-friendly names (*domain names*). *Name resolution* is the act of determining the IP address (or addresses) of a given hostname. The benefits of DNS are two-fold. First, domain names can be logical and easily remembered. Second, should an IP address for a host change, the domain name can still resolve transparently to the user or application.

DNS name resolution is a critical Internet service. Many network services require functional name service for correct operation.

Domain names are separated by dots, with the topmost element on the right, whereas IP address have the topmost element on the left. Each element may be up to 63 characters long; the entire name may be at most 255 characters long. Letters, numbers, or dashes may be used in an element.

The right-most element of a domain name is called the *top-level domain* (TLD). If a domain name is not shortened, it is said to be a *fully-qualified domain name* (FQDN). For example, *lava.redhat.com* may be specified by a machine in the *redhat.com* domain as either *lava.redhat.com* (the FQDN), or as *lava*.

Host names map to IP addresses in a many-to-many relationship. A host name may have one or more IP addresses. Conversely, a particular IP address may have multiple host names associated with it.

Hosts that are designed to perform email routing -- *mail exchangers* -- have special-purpose records in DNS (*MX* records). When a SMTP server, or mail server, needs to send mail to a remote domain it does a DNS lookup for the Mail Exchanger (MX) of that remote domain. A domain can and should have multiple mail exchangers. Mail that cannot be sent to one mail exchanger, can instead be delivered to an alternative server, thus providing failsafe redundancy.

Zones, Domains & Delegation

- A domain is a complete sub-tree of the hierarchical namespace
- A zone is the part of the domain managed by a particular server
- Subdomains may be delegated into additional zones
- A zone may directly manage some subdomains

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

In DNS subdomains are not the same as zones. A zone is a part of the namespace administered by a single name server. A zone may include all the subdomains of a particular domain, or it may delegate authority for some subdomains as separate zones on other name servers.

A zone represents the scope of administration for which one body is responsible.

Name Server Hierarchy

- Master name server
  - Contains the master copy of data for a zone.
- Slave name server
  - Provides a backup to the master name server
  - All slave servers maintain synchronization with their master name server

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Master and slave DNS servers are sometimes referred to as "primary" and "secondary" servers The terminology *primary* and *secondary* is deprecated, as these terms are typically used to refer to nameservers that a client will use for hostname resolution Both master and slave servers contain authoritative data. A zone may have multiple slave servers. A slave server may get its zone data from another slave server There is normally only one master per zone

The DNS Server

- Server receives request
- If server doesn't have answer, either asks root server or forwards request
- Response from upstream server may be final answer or referral to another name server
- `lwresd`

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

A name server may know the answer to a query because either:

- the answer is cached locally, or
- the server is authoritative for the zone queried.

If the name server does not know the answer, then it usually asks a root name server, specified in the hints file (by default, `named.ca`). The upstream name server may respond with a final answer to the query, or with a referral to another name server. This may happen a number of times until a name server that knows the answer is contacted

Alternatively, the name server may be configured to forward the request to another local name server, requesting that the upstream server resolve the request. The upstream server will normally respond with a final answer.

When the server gets a final answer, it puts a copy in its local cache and returns a copy to the requesting client.

`lwresd` is the LightWeight RESolver daemon which provides caching nameserver lookup capability to local client applications which use BIND9's lightweight resolver protocol rather than standard DNS protocol.

dns ~~dm~~ masq 9

Berkeley Internet Name Domain (BIND)

- BIND is the most widely used DNS server on the Internet
  - Red Hat Enterprise Linux uses BIND 9
  - Provides a stable and reliable infrastructure on which to base a domain's name and IP address associations
  - Runs in a chrooted environment

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

BIND has gone through numerous revisions over the years. The most common BIND used since about 1995 was version 8. BIND 9 was released in September 2000. The development of BIND 9 made important improvements in security and robustness. In addition it provides IPv6 support, allows eight-bit clean names, and better multi-threading. The Internet Software Consortium (isc.org), who are the maintainers of BIND recommend that all users of older versions of BIND upgrade to version 9 because of its greatly improved security.

For more information on BIND 9 features, see `/usr/share/doc/bind-<version>/README`.

For more information on migration to BIND 9, see `/usr/share/doc/bind-<version>/misc/migration`.

BIND URL: `http://www.isc.org/products/BIND`

A great reference, the BIND version 9 administrator's manual is at `/usr/share/doc/bind-<version>/arm/Bv9ARM.html`.

Service Profile: DNS

MANAGED BY INITSCRIPT

- Type: System V-managed service
- Packages: *bind, bind-utils, bind-chroot*
- Daemons: **named, rndc**
- Script: **named**
- Ports: 53 (domain), 953(rndc)
- Configs: (Under **/var/named/chroot**)
  /etc/named.conf,
  /var/named/*,
  /etc/rndc.*
- Related: *caching-nameserver, openssl*

NB INSTALL

9

redhat

Rev RH253-RHEL4-1

Unlike the software implementing many network services, the *bind* package installs to an unconfigured state. The *caching-nameserver* package applies a simple configuration to *bind* if required. The *bind-chroot* package assists in setting up *bind* to run in a chrooted environment.

The *openssl* package provides cryptographic libraries needed for the new security features of BIND 9.

If the *bind-chroot* pacakge is installed, `named` will run in a chroot'ed environment. This is the default when choosing the DNS Name Server package group at install time. The *bind-chroot* package creates the chroot'ed environment under `/var/named/chroot`. See the next page for more information on this package

Name service is an SELinux restricted service when enforcing the default targeted policy on a Red Hat Enterprise Linux, version 4 system. It uses a number of contexts, including:

`system_u:object_r:named_cache_t`
For slave data, temporary files.

`system_u:object_r:named_conf_t`
For the named.conf file and other files needed to operate in the chroot'ed environment (for example, devices, other files in `/etc`).

`system_u:object_r:named_var_run_t`
For the service's PID file.

`system_u:object_r:named_zone_t`
For zone files. The `/var/named/chroot/var/named` file uses this context and so any files created in this directory will have the proper context for a zone file

If a file is **moved** into this directory, it may not have the proper context and so it may need to have its context changed. The following command will accomplish this:

```
chcon --reference=/var/named/chroot/var/named \
                    /var/named/chroot/var/named/zonefile
```

bind-chroot

- Config file:
  /etc/sysconfig/named
- Define chroot directory:
  ROOTDIR=/var/named/chroot

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

If you select the *DNS Server* package group, either during initial installation or from the `system-config-packages` gui, the `bind-chroot` package will be installed. This package contains a tree of files which can be used as a `chroot` jail for the `named` program from the `bind` package

When the `bind-chroot` package is installed, `named` runs in a chrooted environment. This environment is defined in the `/etc/sysconfig/named` file by setting a variable called ROOTDIR:

```
ROOTDIR=/var/named/chroot
```

When this variable is set the `named` process chroots to the directory specified in the variable prior to reading any configuration files. The default `chroot` directory is `/var/named/chroot`. This means that all of `bind`'s configuration files will be stored relative to this directory. So, for example, the main configuration file, `named.conf`, will be in `/var/named/chroot/etc/named.conf` instead of `/etc/named.conf`. All other configuration and zone files will also be relative to this directory.

If you remove the `ROOTDIR` setting then named runs without chrooting first, so all files and directories are relative to the root directory.

Configuring BIND

- Default configuration file is
  `/var/named/chroot/etc/named.conf`
- Read by **named** (BIND daemon) during startup or **`service named reload`**
- Text-file specifying directives: zones, options, access control lists, etc.
- Comments can be in C, C++ or shell style

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The BIND 9 configuration file, `named.conf`, is only slightly different in format to BIND 8

At startup or whenever the name server is restarted, always check the system log for error diagnostics and warnings. An invalid directive or option will prevent the server from starting, halting name service. Error messages will be reported on standard error, but the log should be checked even if the server started up successfully.

You should comment your configuration file Comments may be in C, C++ or shell-like syntax:

```
/* this is a C style comment */
// this is a C++ style comment
# this is a shell style comment
```

Directives (`options`, `server`, and `zone`) precede blocks that are delineated by braces. All statements, including blocks delimited by braces, end with a semicolon.

Relative pathnames will be prefixed with value of *directory* option if specified, otherwise `/var/named`.

Global Options

- Declared with the **options** directive:

```
acl "mynetwork" { 192.100.100/24; };
options {
   directory       "/var/named";
   forwarders      { 203.50.0.137; };
   allow-query     { mynetwork; };
   allow-transfer  { mynetwork; };
};
```

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Commonly-used global options are:

`directory`	Base directory of all relative paths specified in `named.conf`. This base directory will be relative to the `CHROOT` directory if it has been set.
`forwarders`	Server forwards queries it can't answer to the name servers at the IP addresses in this list. If it gets no answer, it will try a root name server unless the `forward-only` option is also set.
`allow-query`	Specifies an address match list of hosts allowed to query this server. If this option is not set, any host can query the server. In the example above, only hosts in the 192.100.100.0 network may query
`allow-transfer`	Like allow-query, specifies hosts that may copy the database. Should be used to limit zone transfers.

Access Control Lists (acl)

- Access control list is a list of semi-colon separated IP addresses, networks, or named access control lists
- Can use **acl** directive to create a custom named access control list

```
acl "mylist" { 192.168.0/24;
               192.168.1.12; };
```

- Trailing, non-significant zeros may be dropped
- Makes the configuration easier to read and maintain

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Some global options and directives such as `allow-query` take an address list as an argument. Access control lists are semi-colon delimited lists of IP addresses, network prefixes, or named address match lists.

A named access control list is a shortcut that can be used to quickly specify a set of machines. There are four pre-defined named access control lists available:

`none`	No IP address matches
`any`	All IP addresses match.
`localhost`	Any IP address of the name server matches.
`localnets`	Any network on which the name server has an IP address matches.

One of the main benefits of using ACLs is that they make the configuration file easier to maintain and more human readable. They provide a central place where IP addressed may be changed which is considerably easier than replacing those IP(s) throughout the file if a change needs to be implemented

Name Daemon Control Utility (rndc)

- Provides secure and remote management of running name server
- Uses TSIG security

```
include "/etc/rndc.key";
controls {
    inet 127.0.0.1 allow { localhost; } \
    keys { rndckey; };
};
```

- **rndc** only listens to the loopback interface, or "localhost" by default

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`rndc` is based on the `ndc` utility of earlier versions of BIND, but it supports network client management using port 953, and TSIG keys for security. It's configuration file, `/etc/rndc.conf`, and "key file" are installed with the bind RPM package. During installation, a unique key is generated. The directive listed below is included in the `/etc/named.conf` file. Remember that these locations are relative to the `ROOTDIR` directory if it is being used (i.e. `/var/named/chroot/etc/rndc.conf`).

```
include "/etc/rndc.key";
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};
```

While starting the BIND daemon `named`, a child-process thread is spawned to start the `rndc` utility. If the key for the designated "localhost" acl is missing, or does not match, `named` will not start. Similarly, to stop the `named` daemon, `rndc` is called to verify this key. If the key is invalid, `named` will not *stop*. While not discussed in detail here, `rndc` is also called by a remote BIND, or DHCP server to dynamically "update" the local server's configuration. Again, proper configuration includes the correct key(s) for these servers are available to the `rndc` utility.

Note: `rndc` is not a daemon, but called by `named` to verify keys.

/var/named/chroot/~~etc~~/var/named/redhat.com.zone

Master and Slave Zones

- Declared with the `zone` directive:

```
zone "redhat.com" {
    type            master;
    file            "redhat.com.zone";
  };
zone "kernel.org" {
    type            slave;
    masters         { 192.168.192.168; };
    file
  "slaves/kernel.org.zone";
  };
```

- File name should indicate the zone

CREATED FOR MASTER

redhat 15

Master zones are the central player in BIND configuration. Every non-cached domain name must have a master zone so that authoritative records can be generated for queries. A zone□s name (i.e., `redhat.com` from above) is important; the host name given in a query will be matched against all configured zones.

Slave zones look similar to their master counterparts. The `masters` sub-directive must occur if the `type` sub-directive equals `slave`. A `file` directive may be used to store a local copy of the database which lessens the load on the master server, however, it is not required. For consistency's sake, the `file` sub-directive should be set to the same value as the corresponding master's (except in the `slaves` directory).

When a slave nameserver starts, it tries to contact a master and get a current copy of the database. If the slave stores a local copy of the database in a file, it will just ask the master for the serial number of the current zone file and compare it to the serial number of the stored copy. If the serial number hasn't changed, the slave will use its stored copy, reducing network traffic and server load.

The *file* directive specifies the text file that holds the zone□s database (the zone file). The name of the zone file is arbitrary but common examples include:

```
redhat.com.zone
redhat.com.db
db.redhat.com
redhat.com
```

Reverse Lookup Zones

- Zone name ends with special domain: **.in-addr.arpa**
- Declared with the **zone** directive:

```
zone "10.100.172.in-addr.arpa" {
  type          slave;
  masters       { 172.100.10.1; };
  file          "slaves/172.100.10.zone";
};
```

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The `in-addr.arpa` domain provides for reverse lookups. The associated zone files mainly hold PTR records. The reverse lookup domain for a network is based upon the network□s IP address with the octets in reverse order. This is so the resultant domain name has the topmost element at the right-hand side.

Generally, a reverse lookup zone should be named as follows:

1. Determine the network the zone should cover. In the example above, the zone covers the 172.100.10/24 network.
2. Reverse the order of the octets in the network address. From above, we take 172.100.10 and reverse it to 10.100.172.
3. Append `in-addr.arpa` to the reversed string. Appending onto the result of step 2, we have `10.100.172.in-addr.arpa`.

Special Zones

- Root zone: "."

```
zone "." {
  type            hint;
  file            "named.ca";
};
```

- Loopback zone: "0.0.127.in-addr.arpa"
  - Specified like other reverse lookup zones

redhat 17

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Every BIND configuration should include a root zone. The root zone is used when a query is unresolvable by any other configured zones; in essence, it is the default. Note that the type of the root zone is `hint` (unless the server being configured is a root name server.)

Loopback zones should also be specified, although they aren't strictly required. Many programs --most notably the X Window System -- use local UNIX sockets to emulate IPC queues between cooperating processes. These sockets are bound to 127.0.0.1, the loopback address. Loopback zones should never be slaves.

The file `named.ca` contains information about root servers on the Internet. This information rarely changes, but the latest version can always be obtained from `ftp://rs.internic.net/domain`.

A typical `dig @<ROOT SERVER IP>` will also output a properly formatted `named.ca` file to standard out. This can be used to get a current list of the root nameservers as necessary.

Zone Files

- Files usually reside in **`/var/named/chroot/var/named`**
- Begins with **`$TTL`** (time to live)
- First *resource record* is zone's start of authority (SOA)
- Zone data in additional resource records

redhat 18

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Use semi-colons in database files to signify a comment to the end of line. For example:

```
; this is a comment and will be ignored
```

All zone files must start with a TTL directive. This determines the default length of time in seconds which you want resolving servers to cache your zone's data. The TTL directive takes the form:

```
$TTL  86400
```

Fully-qualified domain names in zone files must end with a dot. BIND will assume that names that don't end with a dot should end with the name of the current domain. Always use a dot at the end of a name that is fully-qualified

Errors in a zone file will not prevent the *named* daemon from starting. However, they can prevent the zone from loading and being read properly causing the entire zone to be inaccessible.

Resource Records (RR)

- Syntax:
  `[domain] [ttl] [class] <type> <rdata>`
  - `[domain]` specify domain or use current
  - `[ttl]` how long record will be cached
  - `[class]` record classification (usually `IN`)
  - `<type>` record type (`SOA`, `MX`, `A`, etc)
  - `<rdata>` specific data for record

redhat 19

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

When the database is parsed, the current domain is set to the value specified in the start of authority record (for example, `redhat.com`). If a '@' appears in the name field, the current default domain for the zone will be used. If whitespace appears in the name field, then the name from the preceding resource record will be used.

Time to live values may be set on a per-record basis, overriding the default set by `$TTL`.

By far, the most widely used class is `IN`, which specifies the Internet class; under most circumstances it□s the only class you'll encounter. The `IN` class is assumed if the class is omitted from the resource record (RR).

SOA (Start of Authority)

- Every zone file must have one

```
@ IN SOA ns.redhat.com. root.redhat.com. (
 2001042501 ; serial number
 300        ; refresh
 60         ; retry
 1209600    ; expire
 43200      ; minimum TTL for negative answers
)
```

- Values no longer need be in seconds

redhat 20

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The SOA designates the beginning of a zone's data, and sets default parameters for that zone It is normally the first resource record in a zone database.

The @ is interpreted as the name of the originating domain - redhat.com in this example The @ is not itself mandatory, but the domain must be indicated.

Values of fields between the brackets, except for the first, are time periods.

serial numbers are used for version control on DNS database files. Every time data in the database is changed, the serial number ***must be increased*** in order that slave servers know the zone has changed. A common practice is to use serial numbers based on ISO dates.

refresh is the delay time that slave name servers should wait between checking the master name server□s serial number for changes. A good value is one hour.

retry is the delay time that a slave name server should wait to refresh its database after a refresh has failed One minute is a good value

expire is the upper time limit that a slave name server should use in serving DNS information for lack of a refresh from the master name server. A good value is 7 days.

The *minimum time to live for negative answers* specifies how long a nameserver should cache a "no such host" response from an authoritative server of the domain. This reduces load on that server.

Note that all times are in seconds by default. However, the following may be used:
W = weeks D = Days H = Hours M = Minutes
(Must use capital letters, no space between the number and the unit is allowed)

The last string in the first line of the SOA record (`root.redhat.com.` from the slide example) specifies the contact person for the domain. Conventionally, the responsible party□s email address is used, replacing the @ with a dot

NS (Name Server)

- There should be an NS record for each master or slave name server serving your zone
- **NS** records point to any slave servers that should be consulted by the client's name server if the master should fail

```
@                IN NS ns.redhat.com.
redhat.com.      IN NS ns1.redhat.com.
```

redhat 21

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

NS records designate name servers to use for this domain. It should contain at least one DNS server that is authoritative for the zone(which may be a slave server, masking the identity of the master zone server). A list of slave servers that can be referenced is commonly included.

Fully-qualified names must be used for NS resource records

Note, you may sometimes see that the first field is left blank. This takes advantage of a BIND zone file shortcut which allows the name for the first field to be assumed from the preceding record. This shortcut can be dangerous because the meaning of the resource record can change if the ordering of the file changes

The @ notation allows the domain name to be taken as the originating domain for the zone.

Main Record Types

- **A** records map hostname to IP address

```
mail                IN A 192.100.100.3
login.redhat.com.   IN A 192.100.100.4
```

- **CNAME** records map address aliases

```
pop     IN CNAME mail
ssh     IN CNAME login.redhat.com.
```

- **PTR** records map IP address to hostname

```
4.100          IN PTR   login.redhat.com.
```

MAIL EXCHANGER

- **MX** records map mail servers for a domain

```
redhat.com.   IN MX  5 mail.redhat.com.
redhat.com.   IN MX 10 lava.redhat.com.
```

PRIORITIES

redhat 22

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`A`, `CNAME`, and `PTR` resource records comprise the bulk of resources seen in database files. Incorrect setup of these records can cause many problems, so they should always be evaluated carefully before changes are committed.

Either short or fully-qualified names may be used when a hostname is needed (don't forget the terminating dot for fully-qualified names.)

An `A` resource record maps a hostname -- which may or may not be fully qualified -- and an IP address. `CNAME` records usually point to names with A records. It's considered a bad idea in most cases to point a `CNAME` to another `CNAME`, because it slows down name resolution, is easy to misconfigure, and is incompatible with older(and still present) DNS server software. However, BIND 9 does support this configuration.

`PTR` resource records are the inverse of `A` records - they map an IP address to a hostname. For reverse lookups -- that is, `PTR` records -- specify the octets of the domain in the reverse order. For example, if the zone were defined as `100.192.in-addr.arpa`, then the name server would expand the `PTR` reference in the slide into `4.100.100.192.in-addr.arpa`. A lookup of 192.100.100.4 would find this reference and would return `login.redhat.com.`

`MX` resource records are used to define mail handlers (or, *exchangers*) for a zone. `MX` records must have a positive integer listed immediately after the `MX` and before the host name. This integer is used by remote Mail Transport Agents (MTA) to determine which host has delivery priority for the zone.

Precedence is given to the mail exchanger with the lowest priority. If that host is not up, then the next lowest priority mail exchanger will be used. If none of the mail exchangers are up, then the mail will be returned to the forwarding SMTP server to be queued for later delivery.

Example Zone File

- **SOA** record
- **NS** records
- **A** records
- **CNAME** records
- **MX** records

redhat 23

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.


```
; Example zone file for redhat.com zone
$TTL 86400
@   IN SOA   ns.redhat.com. root.redhat.com. (
                2001042501      ; serial number YYYYMMDDCC
                3H              ; refresh H = hours
                1M              ; retry M = minutes
                2W              ; expiration W = weeks
                1D )            ; minimum time to live D = days

@               IN  NS  ns.redhat.com.
redhat.com.     IN  NS  ns1.redhat.com.

ns.redhat.com.  IN A  192.100.100.1  ; simple A association
ns1             IN A  192.100.100.2  ; FQDNs aren't required
mail            IN A  192.100.100.3
login           IN A  192.100.100.4
lava            IN A  192.100.100.10
www       0     IN A  192.100.100.5 ; note zero(0) TTL
www       0     IN A  192.100.100.6 ; note zero(0) TTL
www       0     IN A  192.100.100.7 ; note zero(0) TTL

pop             IN CNAME mail        ; alias pop to mail
imap            IN CNAME pop         ; bad idea!

@               IN MX  5 mail.redhat.com. ; used 1st
redhat.com.     IN MX 10 lava.redhat.com. ; used if mail is down
```

Round-Robin Load Sharing Through DNS

- Load balancing can be achieved through the simple use of multiple **A** records:

```
www    0   IN   A    192.168.34.4
www    0   IN   A    192.168.34.5
www    0   IN   A    192.168.34.6
```

- DNS traffic will increase as a TTL of 0 is never cached

redhat 24

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

This is a useful feature for heavily loaded servers. This will allow the duplication of A records to evenly distribute incoming requests. In this example, we have increased the amount of web traffic the site at http://www can handle by a factor of three (notwithstanding network traffic limitations). At the same time, this technique provides a simple method for service continuity in the case of a web server failure.

Note: DNS traffic will increase because a TTL of 0 means queries will never be cached

Delegating Subdomains

- Configure the subdomain as a zone on the new server
- On delegating server, set up **NS** record for the subdomain pointing to the new server
- If new server is in subdomain it manages, on delegating server need a "glue" **A** record for new server

redhat 25

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

It's possible to set up and manage an entire subdomain as part of a zone that includes its parent domain. However, sometimes it's necessary to delegate management of DNS for a subdomain to another name server. As an example, we'll delegate authority for the `support.example.com` subdomain from `example.com`, managed by `ns.example.com`, to a new server called `ns.support.example.com`.

First, we need to set up a zone to manage `support.example.com` on the new `ns.support.example.com` name server.

Second, the parent zone needs to delegate authority for the subdomain to the new server. Do this by creating NS records for the subdomain in the parent zone's database that point to the new name server or servers. In the `example.com` database on `ns.example.com`:

```
support.example.com.        IN NS       ns.support.example.com.
```

But to get the address of this name server for `support.example.com`, we need to look up an A record on the name server for `support.example.com`! To fix this problem, we'll also need to add a "glue record" in the zone file for `example.com` on `ns.example.com` pointing to that server:

```
ns.support                  IN A        192.100.100.10
```

The parent zone will need NS records (and possibly A records) for all authoritative name servers managing the delegated zone.

BIND Syntax Utilities

- BIND will fail to start for syntax errors
  - **named-checkconf**
    - Inspects `/var/named/chroot/etc/named.conf` by default
  - **named-checkzone**
    - Inspects a specific zone configuration

```
named-checkzone redhat.com
        /var/named/chroot/var/named/redhat.com.zone
```

redhat 26

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

In order to prevent problems with BIND starting after a configuration change one should always use named-checkconf before restarting BIND. This is especially important on production servers because syntax errors can prevent BIND from starting, thereby causing a loss of service

`named-checkconf` checks `/var/named/chroot/etc/named.conf` by default, but another file can be specified on the command line

```
$ named-checkconf ~/configs/named.conf
```

`named-checkzone` also allows you to specify a path to a file to be checked. This also allows off line checking before a new or modified zone file is deployed.

Caching-only Name Server

- The caching name server configuration; forwards queries and caches results.
  - *caching-nameserver* RPM package provides a working **named.conf** BIND configuration
  - Also provides Internet root server "hints" or references via **named.ca**

redhat 27

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The *caching-nameserver* package doesn't serve as an authoritative name server for any domain. Its job is to fulfill client requests, and to cache the retrieved information for subsequent lookups. It must be installed with the BIND packages. The name server entry in `/etc/resolv.conf` should then be set to 127.0.0.1.

The *caching-nameserver* package does not contain any binary files. It simply applies a standard configuration to BIND. It is recommended that a `forwarders` entry be added to the `options` section of `named.conf`. This will provide better performance, and reduce the usage of shared network resources.

The *caching-nameserver* configuration can be a good starting point for configuring name servers which are not caching-only.

BIND Utilities

- Many useful utilities are included in the *bind-utils* RPM package, including:
  - **host**: gather host/domain information

    ```
    host -a ns.redhat.com
    host -al redhat.com
    ```
  - **dig**: send queries to name server directly

    ```
    dig @ns redhat.com any
    ```
  - **nslookup**

redhat 28

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

`host` is a particularly useful command, as it is capable of showing all information about a host and/or listing an entire domain. The first example above shows all information about `ns.redhat.com`, while the second example shows all information for every host in the `redhat.com` domain. Listing an entire domain's contents is known as performing a "total zone transfer."

`dig` is used to send queries directly to the name server, bypassing any system resolver libraries. This direct access is useful for problem isolation. `dig` is short for "domain information groper." The output of `dig` is in zone file format, which can make it a useful tool.

`nslookup` has been the standard DNS query tool in Unix and Linux for years. It has both an interactive and non-interactive mode which include a number of useful features. For more information, see `man nslookup`.

Advanced BIND Features

- Integration with **dhcpd** to implement Dynamic DNS(DDNS) updates from the DHCP server
- DDNS updates directly from clients
- Transaction Signatures(TSIG) for secure exchanges between name servers

redhat 29

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Dynamic DNS allows dhcp clients who's IP addresses change frequently to update their DNS records. This can be done on behalf of the client by the `dhcpd` and `named` daemons shipped with Red Hat Enterprise Linux(RHEL).

TSIG, or transaction signatures, provides a method for secure transfers of information using a shared key. This is used for authenticated communication between name servers and `DHCP` servers to update DNS records dynamically. This is also the method used by `rndc` discussed earlier.

More information about these advanced features can be found in the BIND Administrators reference or by attending the Red Hat RHS333 *Red Hat Enterprise Security Network Services* class.

DHCP Overview

- DHCP: Dynamic Host Configuration Protocol, implemented via **dhcpd**
- **dhcpd** provides services to both DHCP and BOOTP clients

redhat 30

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

DHCP provides a method for hosts on a network to request, and be granted, configuration information including the addresses of routers and nameservers. Usually, there is a single DHCP server per network segment, but in some cases there may be more than one. DHCP forwarding agents allow clients to receive addresses from a server which is not located on the same network segment.

IP addresses are either dynamically assigned from a range or pool of addresses, or statically assigned by MAC address. The assignments are made for a configurable amount of time (termed a □lease□ period) and may be renewed by the client. The server can be configured to accept requests from only a specific set of MAC addresses, if desired.

Typically, the server will supply information about the network□s subnet address and netmask, its default gateway, domain name and DNS server, and locations of kickstart configuration files.

DHCP is a superset of BOOTP; dhcpd has been designed to answer requests from BOOTP clients. BOOTP clients will retain their configuration information indefinitely; there is no notion of a lease in BOOTP.

Service Profile: DHCP

- Type: SystemV-managed service
- Packages: *dhcp*
- Daemons: **dhcpd**
- Script: **dhcpd**
- Ports: 67 (bootps), 68 (bootpc)
- Configuration: `/etc/dhcpd.conf`, `/var/lib/dhcp/dhcpd.leases`
- Related: *dhclient*

redhat 31

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The *dhcp* package installs without any configuration The daemon will not start if a `dhcpd.leases` file does not exist. An empty(commented) file is installed with this package.

The DHCP server is an SELinux restricted service when enforcing the default targeted policy on a Red Hat Enterprise Linux, version 4 system. The server uses a number of SELinux contexts for its various files. For purposes of configuration, the following contexts are important:

`system_u:object_r:dhcpd_state_t`
For the `/var/lib/dhcp` directory and the `dhcpd.leases` file within it.

`system_u:object_r:dhcp_etc_t`
For the `/etc/dhcpd.conf` file.

`system_u:object_r:etc_t`
For `dhcpd` related files in `/etc/sysconfig`, including the `dhcpd` and `dhcrelay` files.

After creating a `dhcpd.conf` file and placing it in `/etc`, run the following command when using the default enforced targeted policy to set the proper context for the configuration file:

```
chcon system_u:object_r:dhcp_etc_t /etc/dhcpd.conf
```

Configuring a DHCP Server

- Configure the server in **/etc/dhcpd.conf**
- Sample configuration provided under **/usr/share/doc/dhcp-<version>/**
- There must be at least one subnet block, and it must correspond with configured interfaces.

redhat 32

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`dhcpd` is configured in `/etc/dhcpd.conf` and managed with `service`.

To configure a `dhcpd` server, first use `ifconfig` to verify that a BROADCAST address is specified in your network configuration; Initial DHCP requests are broadcast and not sent to a specific server.

Next, create the `/etc/dhcpd.conf` file. You□ll need to configure lease times, optional subnet masks, router addresses, DNS servers and IP addresses or ranges of addresses for your clients Leased IP addresses are kept in `/var/lib/dhcp/dhcpd.leases` as they are assigned. Refer to the sample configuration file under `/usr/share/doc/dhcp-<version>/`, or the manual page (`man 5 dhcpd.conf`) when creating your site-specific DHCP server Here is a sample:

```
# global definitions
ddns-update-style none;            # turn off DDNS updates
option domain-name "example.com"; # domain name given to client
option domain-name-servers 192.168.0.254;
default-lease-time 21600;          # seconds till expire
max-lease-time 43200;              # maximum lease time
subnet 192.168.0.0 netmask 255.255.255.0
{
    # definitions in this block applicable only to given net
    option routers 192.168.0.253;      # local gateway
    option subnet-mask 255.255.255.0; # local subnet mask
    # Range configuration DHCP
    range 192.168.0.2 192.168.0.250;
    # static configuration for each host BOOTP
    host station1
    {
        hardware ethernet 00:a0:cc:3d:0b:39;
        fixed-address 192.168.0.1;
    }
}
```

End of Unit 2

- Address questions
- Preparation for Lab 2
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 33

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 2
The Domain Name System

Estimated Duration: 1.5 hours

Goal: To install and configure a DNS server

Introduction

This lab steps you through the process of configuring Domain Name System(DNS) using the Berkeley Internet Name Domain(BIND). Using template files as a guide, you will

- Implement a caching-only name server
- Configure `named` as a slave name server for example.com
- Configure `named` as a master name server for forward and reverse IP resolution.

> Throughout this lab, the host and domain names that you use will be based upon the IP address of your machine. Any time the lab refers to a name that contains ***X***, you should replace ***X*** with your station number (the last segment of your IP address). For example, if your station's IP address is `192.168.0.3`, you would replace references to `station`***X***`.domain`***X***`.example.com` with `station`**3**`.domain`**3**`.example.com`.

Disable packet filtering. Before beginning this lab, make sure all packet filtering is turned off on your host (obviously, you should take advantage of the Linux kernel's firewalling capabilities in practice, but for our purposes disabling packet filtering lessens the potential for problems). These and the following commands in this lab will need to be run as the `root` user.

```
service iptables stop
chkconfig iptables off
```

Initial Setup

A. Confirm software installation

The *bind*, *bind-utils*, *bind-chroot*, and *caching-nameserver* packages are required. Use `rpm -q` to determine whether these packages are installed. If not, install them(see Appendix item 1). The *bind* RPM contains the DNS daemon `named` and supporting scripts, but no configuration or zone files. *caching-nameserver* provides a generic caching-only configuration and zone files. The *bind-chroot* RPM provides the files and setup necessary to run `bind` in a chrooted environment.

B. Download Template Files

Template files are available to *aid* in preparing your configuration and zone files You can download them by anonymous ftp from `ftp://192.168.0.254/pub/namedfiles/`

You may choose to work with these files, or create the configuration based on the following instructions. *Remember to replace every X in the templates and in the file names with your station number*

Note-The instructions state that you can either create your own files or you can use these download template files It is important to understand that if you use the downloaded template files, you should *copy* the files to the proper directories. Do not *move* the files!

Reason: If you *copy* the files, they inherit the SELinux context from the directory in which they reside If you *move* them, they will keep their current context and so SELinux will deny access and consequently the `named` server will not be able to read the files

C. Configuring the local resolver

Configure your host so that your system is used for name service, instead of `192.168.0.254`

Note Until your name server is properly installed and configured, this will effectively break DNS for your machine

Edit your resolver configuration file as follows:

/etc/resolv.conf

```
search domainX.example.com
nameserver 192.168.0.X
```

(Remember to replace **X** *with your station number)*

The first line defines the default domain that should be appended to simple hostnames that are not fully qualified. The second line specifies that the host 192.168.0.*X* (your machine) should be used to resolve all DNS queries.

To simplify the situation, remove all but the `localhost` hostname definitions from your hostname configuration file:

/etc/hosts

```
127.0.0.1                    localhost localhost.localdomain
```

This step is not necessary, but may simplify DNS debugging. Sometimes the installer will place your system's fully-qualified domain name on the localhost line, which can make it more difficult to determine whether your name server is functioning properly

Append the following to your `/etc/sysconfig/network-scripts/ifcfg-eth0` configuration to avoid resetting your DNS server setting if accidentally re-requesting a DHCP address.

/etc/sysconfig/network-scripts/ifcfg-eth0

```
[ previous entries ]
PEERDNS=no
```

Sequence 1: Configuring a caching-only name server

The first configuration you will set up is a caching-only name server. This type of name server is not authoritative for any zone. The caching-only name server is set up as a host's primary name server. When a hostname or IP address needs to be resolved, the caching-only name server forwards a request to another name server or to the root name servers in order to determine the authoritative name server for the resolution. Once resolution has taken place, the caching-only name server stores the resolved information in a cache for the designated time-to-live period. Consequently, subsequent lookups should be very fast. You have already installed the files necessary for this configuration. Follow the steps below to configure the name server.

1. Add the following lines to the "options" section in the file called `/var/named/chroot/etc/named.conf` provided by the *caching-nameserver* RPM:

   ```
   forwarders { 192.168.0.254; };
   forward only;
   ```

   This will cause the caching-only nameserver on your workstation to forward DNS requests it can't resolve to the name server on `192.168.0.254`, and not to contact the root name servers directly if that times out.

2. Start `named`: `service named start`

3. Test your configuration using `host` or `dig` and querying some example.com names and some "real" names on the Internet (if you have Internet access).

Sequence 2: Configuring a slave name server

A slave name server can provide authoritative answers for a zone, but is not the zone's start of authority. You will now reconfigure your name server as a slave server for the `example.com` and `0.168.192.in-addr.arpa` zones

1 Append the following lines to your `/var/named/chroot/etc/named.conf` file:

```
zone "example.com" {
   type slave;
   masters { 192.168.0.254; };
   file "slaves/slave-example.com.zone";
};
zone "0.168.192.in-addr.arpa" {
   type slave;
   masters { 192.168.0.254; };
   file "slaves/slave-192.168.0.zone";
};
```

2 Restart named: `service named restart`

3 Examine `slave-example.com.zone` and `slave-192.168.0.zone` (these files are in the `/var/named/chroot/var/named/slaves` directory). These files should contain copies of the appropriate zone database transferred from the master server on `192.168.0.254`.

Before you start the next section, remove the two slave zones you just added in step 1 from `/var/named/chroot/etc/named.conf`.

Sequence 3: Configuring a master name server

Now you will configure your name server to take responsibility for the "domainX.example.com" zone. You will also take responsibility for a corresponding reverse lookup zone The following steps will be involved:

A. Editing the configuration file (named.conf)

B. Preparing database files for the "domainX.example.com" zone and the "X.0.168.192.in-addr.arpa" zone

C. Restarting the name server

D Testing your configuration

A. The primary configuration file

There are three zones that we want to consider:

1. The " " (root level) zone

   The " " zone is at the top of the DNS hierarchy Root servers provide information on what name servers are authoritative for a given domain. The " " stanza should appear as follows:

```
zone "." {
    type hint;
    file "named.ca";
};
```

2. The "domainX.example.com" (forward lookup) zone

   Add the following lines to establish your name server as the master name server for the zone.

```
zone "domainX.example.com" {
    type master;
    file "domainX.example.com.zone";
};
```

3. The "X.0.168.192.in-addr.arpa" (reverse lookup) zone

   Now add these lines to establish your name server as the master name server for the reverse lookup zone.

```
zone "X.0.168.192.in-addr.arpa" {
    type master;
    file "192.168.0.X.zone";
};
```

The following is a sample configuration file for `station2` at `192.168.0.2`:

/var/named/chroot/etc/named.conf

```
options {
    directory "/var/named";
    forwarders { 192.168.0.254; };
    forward only;
};

zone "." {
        type hint;
        file "named.ca";
};

zone "localhost" {
    type master;
    file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};

zone "domain2.example.com" {
    type master;
    file "domain2.example.com.zone";
};

zone "2.0.168.192.in-addr.arpa" {
    type master;
    file "192.168.0.2.zone";
};
```

Note-There may be other items in the `/var/named/chroot/etc/named.conf` file. You should not change them. Just make sure that you change the options and zone statements as shown above (edited for your workstation, of course).

B. The database files

Your primary configuration file specifies /var/named as the database directory (remember that this is relative to the chrooted directory /var/named/chroot). You must now create database files for your zone and the reverse lookup zone and place them in this directory These database files will contain your SOA, NS, A, CNAME, MX, PTR, and possibly other resource records. All of the zone database files should start with the line

```
$TTL 86400
```

The numeric value here is the default time-to-live period in seconds that will apply to all of the records in that zone.

1. The "domainX.example.com" zone

   In the primary configuration file, the database file for the "domainX.example.com" zone is defined to be /var/named/domainX.example.com.zone (which really means /var/named/chroot/domainX.example.com.zone) That file contains records similar to the following:

   Start Of Authority Record

```
@   IN SOA stationX.domainX.example.com. root.stationX.domainX.example.com.   (
                2001101100 ; Serial
                28800      ; Refresh
                14400      ; Retry
                3600000    ; Expire
                0 )        ; Negative
```

   The "Start Of Authority" (SOA) record should be the first resource record in the database files, but it may be preceded by directives such as $TTL (default time to live). The SOA record establishes this database file as the authoritative source of information for this zone The first token is the domain for which the following records are appropriate, and is often specified by the "@" abbreviation, which expands to the domain name defined in the "zone" stanza in named conf (or the current zone defined by an $ORIGIN directive in the file, if such a definition exists).

   The fourth token is the master name server for this domain, and the fifth is the email address of an administrator responsible for maintaining the database, with the first period replacing the "@" in the address (Can you explain why?) The remaining entries in the record specify the dynamics of interactions with resolving name servers

   Name Server Record(s)

```
@   IN   NS   stationX.domainX.example.com.
```

   Name Server (NS) records identify hosts as name servers authoritative for the specified domain They specify both master and slave name servers for this zone, and delegate authority for subdomains to zones on other name servers (For example, server1.example.com has NS records for all the domainX.example.com name servers) As you have only a single name server for the "domainX.example.com" zone, you have only a single NS record.

Address Records

```
domainX.example.com.            IN      A       192.168.0.X

stationX.domainX.example.com.   IN      A       192.168.0.X
www                             IN      A       192.168.0.X
ftp                             IN      A       192.168.0.X
pop                             IN      A       192.168.0.X
```

Address (A) records map a hostname to an IP address (the primary function of the name server) A database file would normally contain A records for many IP addresses In our classroom setting, however, there is only a single host in your domain. Note that the first A record establishes a "default IP address" for the domain. The subsequent A records establish multiple hostnames for a single IP address

Hostnames can be specified as either fully qualified domain names (FQDNs), or as abbreviations All hostnames that do not end in a period are interpreted as abbreviations, and the zone name is appended to the hostname For example, the third A record is for the hostname `www.domainX.example.com`

Canonical Name (Alias) Records

```
www1                    IN      CNAME   stationX.domainX.example.com.
www2                    IN      CNAME   stationX.domainX.example.com.
www3                    IN      CNAME   stationX.domainX.example.com.
```

Alias (CNAME) records establish aliases for hostnames. Note that aliases map to other hostnames, not to IP addresses.

CNAMEs should not be used in place of "true names" in the data portion on the right side of resource records Resolving multiple aliases slows name lookups.

Mail Exchanger Records

```
@                       IN MX 10 stationX.domainX.example.com.
domainX.example.com.    IN MX 10 stationX.domainX.example.com.
```

Mail Exchanger (MX) records define a host that will handle email transfers for a given domain or host. When a Mail Transport Agent (MTA) attempts to deliver mail, it will first perform a DNS lookup for the destination host's MX record. If a MX record exists, mail will be sent to the host specified by the MX record. Otherwise, if no MX record exists, the MTA will usually attempt to perform a standard DNS lookup for the destination host, and deliver the mail to that host directly. MX records are used to establish mail gateways, and as default destinations for mail addressed to domains.

2. The "`X.0.168.192.in-addr.arpa`" zone

In `/var/named/chroot/etc/named.conf`, we specified `/var/named/192.168.0.X.zone` (which really means `/var/named/chroot/var/named/192.168.0.X.zone`) as the database file for the reverse lookup zone `X.0.168.192.in-addr.arpa`. It should contain a `SOA` record, a `NS` record, and appropriate `PTR` records.

Start Of Authority and Name Server Records

```
@ IN SOA     stationX.domainX.example.com. root.stationX.domainX.example.com. (
                 4          ;
                 10800      ;
                 3600       ;
                 604800     ;
                 86400 )

  IN NS      stationX.domainX.example.com.
```

The SOA and NS records are identical in form to those in the previous zone file.

Note that the leading whitespace in the NS record is significant, and is interpreted as an abbreviation for "same name as the previous record". In this case, the previous record's name was the "@" symbol, which is itself an abbreviation for the zone's domain as defined in the primary configuration file.

Pointer Records

```
X.0.168.192.IN-ADDR.ARPA.      IN  PTR    stationX.domainX.example.com.
```

Pointer (PTR) records map names to IP addresses through an indirect mechanism. Instead of using a separate technique to perform a reverse lookup on IP addresses, BIND performs a modified forward lookup on a specially named domain. This "reverse lookup domain" is formed by reversing the IP address and appending the "in-addr.arpa" domain. This allows the name server to perform both forward and reverse lookups using similar mechanisms.

3. Putting it all together

The following are sample configuration files for `station2` at `192.168.0.2`:

/var/named/chroot/var/named/domain2.example.com.zone

```
$TTL 86400
@  IN SOA station2.domain2.example.com. root.station2.domain2.example.com. (
                                    2001101100 ; Serial
                                    28800      ; Refresh
                                    14400      ; Retry
                                    3600000    ; Expire
                                    0 )        ; Negative

@                   IN   NS    station2.domain2.example.com.

@                   IN   A     192.168.0.2

station2.domain2.example.com. IN      A    192.168.0.2
www                           IN      A    192.168.0.2
ftp                           IN      A    192.168.0.2
pop                           IN      A    192.168.0.2

www1                 IN       CNAME   station2.domain2.example.com.
www2                 IN       CNAME   station2.domain2.example.com.
www3                 IN       CNAME   station2.domain2.example.com.

@              IN     MX 10    station2.domain2.example.com.
station2       IN     MX 10    station2.domain2.example.com.
```

/var/named/chroot/var/named/192.168.0.2.zone

```
$TTL 86400
@  IN SOA station2.domain2.example.com. root.station2.domain2.example.com. (
                                    4 10800 3600  604800 86400 )
   IN NS  station2.domain2.example.com.

2.0.168.192.IN-ADDR.ARPA.       IN  PTR   station2.domain2.example.com.
```

C. Restarting the name server

Once more, we'll restart the name server. Then, confirm that it's running with the `pidof` command:

```
service named reload
pidof named
```

Skim through the entries that the server appended to the `/var/log/messages` file. Confirm that your domain loaded without errors.

D. Testing the name server

Make the following DNS queries. Can you interpret all of the results?

```
host stationX
dig stationX.example.com
dig stationX.example.com @192.168.0.254
dig stationX.domainX.example.com
host server1.example.com
host 192.168.0.X
dig -x 192.168.0.X
dig -x 192.168.0.254
host www
host www1
```

Remember that `dig` expects to be given FQDNs for lookups, while `host` will look at the search information in `/etc/resolv.conf`. Try additional queries on other people's name servers and subdomains. If set up correctly, you should be able to perform forward and reverse lookups on other classroom systems.

Challenge Projects:

Configure a "round robin" hostname by adding multiple "A" records to different IP addresses for a single hostname How does the name server handle this situation? *Hint*: Try setting the TTL for just those A records to zero

Add the subdomain "support.domain*X*.example.com" to your domain. Add appropriate resource records that refer back to your station's IP address.

Partner with another station, and become a slave name server for each other's domain. Add a new CNAME for your station to your zone, and verify that the change propagates to the secondary server

Challenge Break:

1. Disable the system's "firewall," if neccessary.

   ```
   service iptables stop
   ```

2. Run the following command, following the instruction displayed:

   ```
   tsnetwork 1
   ```

3. This command will set up the problem and will explain the goal. Refer to the file `/etc/ts` to review the goal. Refer to Lab 1, if you need help.

Cleaning Up:

Subsequent labs will flow more smoothly if you reset your station so all DNS queries are made from the classroom server. In order to clean up, make sure that you have reset your `/etc/resolv conf` file to its original state, and configure `/etc/hosts` as indicated below

/etc/resolv.conf

```
search example.com
nameserver 192.168.0.254
```

/etc/hosts

```
127.0.0.1       localhost.localdomain localhost
192.168.0.X     stationX.example.com
```

Comment, or remove the entry added earlier in the lab:

/etc/sysconfig/network-scripts/ifcfg-eth0

```
[ previous entries ]
#PEERDNS=no
```

UNIT 3

Network File Sharing Services

redhat

1

Rev RH253-RHEL4-1

Objectives

- Explain Network File Sharing
- Describe the NFS service
- Describe the FTP service
- Describe the SMB/CIFS service
- Use client tools with each service

2

Agenda

- Introduction to NFS
- Configuring the NFS service
- Introduction to FTP
- Configuring the FTP service
- Introduction to Samba (SMB)
- Configuring the SMB service

Network File Service(NFS)

- The Red Hat Enterprise Linux NFS service is similar to other BSD and UNIX variants
  - Exports are listed in **/etc/exports**
  - Server notified of changes to exports list with **'exportfs -r'**
  - Shared directories are accessed through the **mount** command
  - The NFS server is an RPC service and thus requires **portmap**
- Red Hat Linux supports NFS version 3.0 on the client, and most 3.0 features on the server

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

NFS server software included with Red Hat Enterprise Linux(RHEL) is composed of three facilities, included in the *portmap* and *nfs-utils* rpms:

`portmap`: maps calls made from other machines to the correct RPC service
`nfs` (in kernel): translates NFS requests into requests on the local filesystem
`rpc.mountd`: mounts and unmounts filesystems

These all run as daemons and are started at boot time from the `portmap` and `nfs` System V initialization scripts in the usual way. The filesystems to share are listed in `/etc/exports`.

To verify that these services are running, use `rpcinfo -p` or `service`:

```
service portmap status
service nfs status
```

To verify that these services are running on remote host `bigserver` use:

```
rpcinfo -p bigserver
```

`exportfs -r` refreshes the server's share list after modifying *`/etc/exports`*
`exportfs -v` displays a list of the shared directories and options on a server
`exportfs -a` exports all shares listed in `/etc/exports`, or a share named as an argument
`exportfs -u` unexports the share named as an argument, or all shares with no argument and `-a`
`showmount -e` *`host`* shows the available shares on *`host`*

`portmap`, `rpc.nfsd` and `rpc.mountd` are required to run an NFS server

Service Profile: NFS

faster THAN samba
NOT good over DISTANCE.

- Type: System V-managed service
- Packages: *nfs-utils*
- Daemons: `nfsd, lockd, rpciod, rpc.{mountd,rquotad,statd}`
- Scripts: `nfs, nfslock` nfs
- Ports: Assigned by portmap (111)
- Configuration: `/etc/exports`
- Related: *portmap* (mandatory) NEED

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

NFS Server

- Exported directories are defined in **/etc/exports**
- Each entry specifies the hosts to which the filesystem is exported plus associated permissions and options
  - options should be specified
  - default options: **(ro, sync)**
  - **root** mapped to UID 65534(**nfsnobody**)

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Filesystems to be exported via NFS are defined in `/etc/exports`. Here is an example:

```
/var/ftp/pub              *.example.com(ro,sync) bigserver.redhat.com(rw,sync)
/root/presentations     server2.example.com(rw,sync)
/data                   192.168.10.0/255.255.255.0(sync)
```

Ro - default

Each entry specifies one exported directory and its access permissions. One or more host/permission pairs can be specified, but entries cannot be split into multiple lines. Hostnames can contain wildcards, as in the above example. Wildcards can also be used to match hostnames or domain names: `station1*` will match `station1, station10, station11,` etc.; `*.example.com` will match `station1.example.com` and `station1.corp.example.com`. The ? wildcard, indicating a match of exactly one character, is also supported.

IP addresses of hosts can be specified individually or using a network/netmask specification. Entries in `/etc/exports` are exported read-only by default.

Options must not be separated from hostnames with whitespace. If whitespace exists between a hostname and an option, it is treated as two distinct export destinations and the option will apply to a "world" export. This is probably not what is intended.

Entries in `/etc/exports` are exported with *root_squashing* turned on. This ensures that requests from the root user on a client machine are denied root access to root-owned files on a server machine. Such requests are mapped instead to a uid such as 65534. This behavior can be defeated with the `no_root_squash` option, but this is not recommended.

`service nfs status` and `exportfs -v` will help confirm proper operation of your NFS server. For more information see the man page for `exports`

Client-side NFS

- implemented as a kernel module
- **/etc/fstab** can be used to specify network mounts
- NFS shares are mounted at boot time by **/etc/rc.d/init.d/netfs**
- **autofs** mounts NFS shares on demand and unmount them when idle

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

To associate a shared directory on the network with a mount point in your local filesystem, use `mount`. When you mount an exported directory from an NFS server, you can access it as if it was local to your machine. Shares can be mounted manually by root, or automatically at boot time

`/etc/fstab` allows you to specify network directories to be mounted at boot Here's a sample `fstab` entry that defines a shared filesystem `/var/ftp/pub` on server1 to be mounted locally as `/mnt/pub`.

```
server1:/var/ftp/pub   /mnt/pub          nfs      defaults      0 0
```

`/etc/rc.d/init.d/netfs` mounts any network filesystems that are configured to be mounted at boot time, such as the one defined above.

Some NFS-specific options that can be used with `mount` or in `/etc/fstab` include:

- `rsize=8192` and `wsize=8192` - will speed up NFS throughput considerably
- `soft` - processes return with an error on a failed I/O attempt
- `hard` - will block a process that tries to access an unreachable share
- `intr` - allows NFS requests to be interrupted or killed if the server is unreachable
- `nolock` - disables file locking (`lockd`) and allows interoperation with older NFS servers

The kernel automounter facility, *autofs*, provides the ability to mount NFS shares on demand and unmount them when they are idle in a way that is transparent to the end user. Install the *autofs* RPM, then examine `/etc/auto.master` and `/etc/auto.misc` for examples of how *autofs* is configured. *autofs* is a kernel service, but the capability must be enabled by configuring *autofs* to run in the appropriate runlevels.

CONNECTION PORT 21
20 - } ACTIVE, PASSIVE - PORT 20

File Transfer Protocol(FTP)

FC. Script.

- **vsftpd** - the default RHEL ftp server
- No longer managed by **xinetd**
- Allows anonymous or real user access only
- The anonymous directory hierarchy is provided by the *vsftpd* RPM
- **/etc/vsftpd/vsftpd.conf** is the main configuration file

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The Very Secure FTP Daemon, `vsftpd`, was designed to be a secure, stable, fast, and scalable FTP daemon It provides two levels of user access:

Anonymous access. Users can log in as user `ftp` or as user `anonymous` to get access to an anonymous ftp site. By default, anonymous users are change-rooted into `/var/ftp` for security.

User access. Users with accounts on the target system can connect via FTP and log in using their username and password. They can download any file they can read, and upload to any directory which they have write access

The configuration file for `vsftpd` is `/etc/vsftpd/vsftpd.conf` By default, anonymous users may download files but may not upload them.

To *disable* anonymous user access:

```
anonymous_enable=NO
```

To *enable* anonymous users to upload file:

```
anon_upload_enable=YES
```

Two access files are used by `vsftpd`. Individual users can be denied access by placing their username in `/etc/vsftpd.ftpusers` A second file, `/etc/vsftpd.user_list`, is only examined if `userlist_enable=YES` is set in `/etc/vsftpd/vsftpd.conf`. It can be used either to list users which will be allowed access or users which will be denied access, depending on whether the option `userlist_deny=NO` is set(default is `YES`). In order to gain access to the ftp daemon, a user must satisfy the requirements of both access files.

If a file called `.message` exists in a directory, the contents of that file will be displayed to FTP clients accessing that directory

Service Profile: FTP

- Type: SystemV-managed service
- Packages: *vsftpd*
- Daemons: **vsftpd**
- Script: **vsftpd**
- Ports: 21 (ftp), 20 (ftp-data)
- Configuration: **/etc/vsftpd/vsftpd.conf**
  /etc/vsftpd.ftpusers
  /etc/pam.d/vsftpd
- Logs: **/var/log/vsftpd.log**

vsftpd.USER LIST

redhat 9

Rev RH253-RHEL4-1

Samba services

smbd

nmbd.

- Four main services are provided:
  - authentication and authorization of users PAM
  - file and printer sharing
  - name resolution
  - browsing (service announcements)
- Related
  - *smbclient* command-line access
  - *smbfs* Linux can mount an SMB share

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The Samba server will provide user authentication via passwords and optionally domains Samba can also try `AFS` (Andrews File System) authentication, granting `AFS` rights if successful or falling back to native password checking otherwise.

Samba also has the ability, through the `winbindd` daemon, to attach to a Microsoft domain password server. When a RHEL machine is running the `winbindd` service, user accounts defined in the Microsoft Domain can be used to authenticate to the RHEL machine Additionally, in Samba version 3 0 and later, the Samba services can use some Microsoft "Active Directory" resources

File and printer sharing is probably the most attractive Samba feature to many users. With this functionality, users can easily retrieve files or print to any printer on the network.

Browsing is the capability of participating systems to view the contents of other participating systems in the same "neighborhood". With the proper authorization, users may look at and use devices and files of other computers as if they were local.

Name resolution is an integral component of browsing. With name resolution, a computer in the "Network Neighborhood" can receive the neighborhood name of other computers; the "neighborhood" name is not necessarily the network name of the computer Additionally, name resolution comprises a portion of `WINS` (Windows Internet Name Service), which allows centralized mapping of `NetBIOS` names to `IP` addresses. This name service is independent of `DNS`.

`smbclient` is a standard utility in the Samba suite to provide command-line `SMB` client access This is useful for testing access, and for using in scripts The `smbclient` command interface is very similar to that of a command line `ftp` client.

`smbfs` is an optionally-configured kernel component (`CONFIG_SMB_FS`). It permits direct mounts of an SMB share, in a similar fashion to mounting an NFS share. `smbfs` is not required to provide file and print services.

Samba Daemons

- **smbd** : SMB/CIFS server
  - authentication and authorization
  - File and printer sharing
- **nmbd** : NetBIOS name server
  - resource browsing
  - WINS server

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

When a SMB/CIFS client starts, it may need to know what IP address a specified host is using. The client will broadcast this request on the network and will receive a response from nmbd, informing the client of its NetBIOS information.

Broadcast requests can saturate a network, making it virtually unusable. To combat this, Microsoft developed the WINS protocol, which pulls all the broadcast isolated subnets into a single NetBIOS scope. nmbd will act as a WINS server, maintaining a database of all computers on the network. Note that you should not mix the use of the Samba WINS server and the Windows NT one. In a mixed NT and Samba environment, Samba recommends that you use the NT server□s WINS capabilities.

smbd provides file space and printer services to clients using the SMB/CIFS protocol. Using the notion of "shares," or logical volumes of disk or printers, smbd can provide access to these facilities with the proper authorization

Service Profile: SMB

- Type: System V-managed service
- Packages: *samba{,-common,-client}*
- Daemons: **nmbd, smbd**
- Script: **smb**
- Ports:(netbios) 137(-ns), 138(-dgm), 139(-ssn)
- Configuration: **/etc/samba/***
- Related: *system-config-samba*

redhat 12

Configuring Samba

- Configuration in **/etc/samba/smb.conf**
  - Red Hat provides a well-commented default configuration, suitable for most situations
- Configuration tools are available
  - **system-config-samba**
  - Hand-editing **smb.conf** is recommended

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Red Hat has authored the system-`config-samba` tool to offer administrators the abilities to create new shares, add new samba users, and perform basic server configuration.

Additionally, samba provides a syntax checking program to review the `/etc/samba/smb.conf` file. The `testparm` command will test the configuration in `/etc/samba/smb.conf`. If `testparm` finds there are syntax errors, you will be told of the error and on which line the error was encountered. You will *not* be informed of non-existant users, groups nor directory paths!

Red Hat Enterprise Linux 4 includes another gui interface for configuring `samba`. This interface is `samba-swat`. `samba-swat` uses a browser which connects through port 901 on the server system. The `samba-swat` interface has many more features than the `system-config-samba` interface.

One caveat that the user should be aware of is that `samba-swat` interface removes all of the comments from the `smb.conf` configuration file when it reads and saves the file.

Overview of `smb.conf` Sections

`smb.conf` is styled after the "ini" file format and is split into different [ ] sections

- **`[global]`** : section for server generic or global settings
- **`[homes]`** : used to grant some or all users access to their home directories
- **`[printers]`** : defines printer resources and services

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Red Hat provides the Samba suite on the standard distribution

Open `/etc/samba/smb.conf` in an editor.

The `smb.conf` file consists of sections and parameters A section begins with the name of the section in square brackets and continues until the next section begins Sections contain parameters of the form:

```
name = value
```

Parameters of an individual section override `[global]` section parameters

Comments may either be shell or assembly style; that is, all text after a hash (#) or semi-colon (;) to the end of the line are ignored:

```
# this is a comment
; this is also a comment
```

Configuring File and Directory Sharing

- Shares should have their own [ ] section
  - Some options to use:
    - **public** - share can be accessed by **guest**
    - **browseable** - share is visible in browse lists
    - **writable** - resource is read and write enabled
    - **printable** - resource is a printer, not a disk
    - **group** - all connections to the share use the specified group as their primary group

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The example Samba configuration file shows many different configurations of shares. As a simple example, consider Joe, who wants to share his home directory. Access control would be determined by normal group memberships and file permissions. The system administrator might then add the following to `/etc/samba/smb.conf`:

```
# share Joe□s home directory
[joe-home]
  comment = Joe□s Home Directory
  path = /home/joe
  public = no
  writable = yes
  printable = no
```

Printing to the Samba Server

- All printers defined in `/etc/cups/printers.conf` are shared as resources by default
- Can be changed to allow only explicitly publicized printers

redhat 16

Rev RH253-RHEL4-1 Copyright © 2006 Red Hat, Inc.

Printing is enabled by default for all valid users. You may restrict the printer use on a global or per-printer basis. To setup global printer options, you may use something like:

```
[printers]
  comment = All the Printers
  path = /var/spool/samba
  browsable = no
  public = yes
  guest ok = yes
  writable = no
  printable = yes
```

To setup a specific printer, use something like:

```
[3rdfloor]
  comment = Research Printer (3rd Floor Laser Printer)
  printer = 3rdfloor
  valid users = bob jane joe mary
  path = /var/spool/3rdfloor
  public = no
  writable = no
  printable = yes
```

By default the samba server assumes that the machine uses cups as the printing sub-system. This can be changed by modifying the `printing =` parameter in the `[global]` section, see the comment above the printing parameter for other acceptable printing sub-systems.

Authentication Methods

- Specified with **`security = <method>`**
- Valid methods are:
  - **`user`** : validation by user and password (this is the default)
  - **`share`** : user validation on per-share basis
  - **`domain`** : a workgroup with a collection of authentication data is used
  - **`ads`** : acts as an "Active Directory" member with Kerberos authentication

redhat 17

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Most Samba administrators will set `security = user` in their configuration file. Using this setting usually requires the setup of the `smbpasswd` file and possibly the setup of the username map file, `smbusers`.

If you choose to use `share` authentication, then public access will be denied to all shares, even if they are explicitly set to allow public access. In this scenario, only shares with specified users or groups will be accessible at all

If you choose to use `domain` authentication, also set:

```
# need an appropriate domain name (about 4 chars)
workgroup = MINE
encrypt passwords = yes
# server(s) to validate this domain; searched in order
password server = host1 host2 host3
```

If you choose to use the `ads` authentication, you must also specify your Kerberos "realm," and possibly the location of your "Active Directory" server:

```
realm = YOUR_KERBEROS_REALM
password server = your_kerberos_server
```

With `ads`, you must also configure your Samba server's account in the Microsoft® domain

Passwords

- Encrypted password considerations
  - Stored in **/etc/samba/smbpasswd**
  - Users managed with **smbpasswd**
  - Users must have local accounts, or implement **windbindd**, a separate service

redhat 18

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

To add a user, first use the `smbpasswd` script:

```
smbpasswd -a joe
```

Remember that `joe` must exist in `/etc/passwd` before `smbpasswd` will add the entry. Use `smbpasswd` for subsequent password changes for all your users.

It is also possible to define user accounts in `/etc/samba/smbpasswd`, but specify that a Primary Domain Controller will manage passwords.

Samba Client Tools: **smbclient**

- Can be used as an **ftp**-style file retrieval tool
  - **smbclient //machine/service**
    > cd directory
    > get file
- Allows for simple view of shared services
  - **smbclient -L hostname**
- **user%password** may be specified with **-U** or by setting and exporting the **USER and PASSWD** environment variables

redhat 19

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Sometimes you will see *service* used in place of *share*; these two words are synonymous A path of the form `//machine/service` is called a UNC path.

You may specify a user name to `smbclient` with which to connect. If you do not, `smbclient` will use the upper-case version of the `USER` or `LOGNAME` environment variables, in that order, and `PASSWORD` if it exists. If you do specify the `-U` option or the `USER` environment variable is set, then a user name will be formed from all characters up to, but not including, a separating percent (`%`) symbol Any characters after the percent symbol will be treated as the password for the user For example:

```
smbclient -L somehost -U 'bob%foobar'
```

Supplying passwords on the command line should be avoided, since those passwords would then be available to anyone able to retrieve previous commands.

nmblookup

- list specific machine
  `nmblookup -U server -R 'name'`
- list all machines
  `nmblookup \*`

redhat 20

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`nmblookup` queries a WINS server in much the same way that `nslookup` queries a DNS server; the same kind of information -- hostname and IP -- will be returned

When querying a broadcast area, use * as the machine name Note that you will have to precede the * with a backslash (\ ) to protect it from shell expansion See the example in the slide above.

smbmount

- The SMB file system is supported by the Linux kernel
- Use **smbmount** to mount a SMB-shared resource:

```
smbmount service mountpoint -o options
```

redhat 21

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`smbmount` allows a remote SMB filesystem to be mounted. Because the kernel support for the Samba filesystem has to be present before `smbmount` will work, Linux is the only operating system that supports this feature in Samba.

To connect to `share` on host `server` as `smbuser`, the command below can be used:

```
smbmount //server/share /mnt/smb_mountpoint -o username=smbuser
```

`smbmnt` (a helper application used by the `smbmount` program) must be set SUID `root` for non-root users to mount the samba share. This will also allow non-root users to un-mount the share. See `man smbmnt` for further information.

The `smbumount` command may also be used by non-privileged users to unmount there Samba filesystems. The format is:

```
smbumount mount-point
```

For more information on this command see `man smbumount`.

Samba Mounts in **/etc/fstab**

- Samba mounts can be performed automatically upon system boot by placing an entry in **/etc/fstab**
- Specify the UNC path to the samba server, local mount point, **smbfs** as the file system type, and a user name.

redhat 22

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

For example: To mount `//server1/public` on `/mnt/smb` automatically upon system boot, place the following line in `/etc/fstab`:

```
//server1/public /mnt/smb smbfs defaults,username=nobody 0 0
```

Rather than put the username and password for Samba authentication in the `/etc/fstab` file, it might be more advisable to use the following method to protect them from prying eyes.

In the `/etc/fstab` file the entry might look like:

NB!!

```
//servername/share  /mntpt smbfs  defaults,credentials=/etc/samba/cred.txt 0 0
```

The `/etc/samba/cred.txt` file should only be readable by root. This ~~file is an ascii text file~~ which contains the following:

```
username=<samba username>
password=<samba password>
```

End of Unit 3

- Questions and Answers
- Preparation for Lab 3
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 23

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 3
Network File Sharing Services

Estimated Duration: 1 hour

Goal: Share file and printer resources over the network

Disable packet filtering Before commencing, ensure that packet filtering is not active The default installation will have a file called `/etc/sysconfig/iptables`*, which configures the* `iptables` *facility. Run* `"chkconfig iptables off"`*. To remove any rules that may be in place, then run* `"service iptables stop"`*.*

Sequence 1: Samba configuration for user connectivity

Tasks:

1 If necessary, install the *samba*, *samba-common*, and *samba-client* RPMs and start the `smb` service(see Appendix 1). A default configuration will apply. Verify that Samba is working correctly with:

```
smbclient -L localhost -N
```

You should get a response back from the server, but no indication of available shares (Make sure `smbd` is running or else the command will not work)

2. Create a group named "legal" to which the users `karl`, `joe`, `mary` and `jen` belong. These users do not yet exist on your system, so you'll need add them too. The following commands will create the group, and add each of these users with a "nologin" shell. *Do not give them passwords!* They are permitted access to this system only through Samba

```
groupadd -g 30000 legal

for user in karl joe mary jen; do
    useradd -G legal -s /sbin/nologin $user
done
```

3. By default, Samba is configured to receive encrypted passwords, but no passwords have been set in `/etc/samba/smbpasswd`. To test your Samba server, you must run `smbpasswd` for each user you wish to grant access to your server. Add the users from step #2, `root`, and another user of your choosing(e.g., `student`).

4 Notice the first share defined in `/etc/samba/smb.conf`, `[homes]` , has no path specified. This share is configured to share a user's home directory to them if they connect and authenticate Explore one or two of the user's home directory shares. Upload a file to the `joe` user's home share

Deliverable:

A working Samba server accessible to several users with `smbclient`

Sequence 2: Providing access to a group directory

Scenario/Story:

It turns out that in addition to having their own private shares on the server, our four users are part of the same department and require a place to store their departmental files. We will need to set up a linux group for the users, create a directory for the users to store their content, and configure the Samba server to share the directory.

Tasks:

1 Create the directory path `/home/depts/legal`. Set the ownerships and permissions on the directory such that people in the `legal` group can add/delete files, but others can not. Also set the SGID and Sticky permissions so that the group ownership on all files created in this directory will be set to the group owner of `legal` and so that one user can not remove another's files.

2 Create a Samba share called `[legal]` in `/etc/samba/smb.conf`. Only the members of the `legal` group should have access to the share. In addition, ensure that the files placed in the `[legal]` share are created with permissions 0660.

3 Restart the `smb` service and test users of the legal group (and users *not* of this group) with `smbclient`.

Deliverables:

1. A Linux directory that only the `legal` group can use.

2. A Samba share that only `legal` group users can access and modify.

Sequence 3: Providing access to printers

Scenario/Story:

One of the primary fuctions of Samba besides sharing files is sharing access to print queues that have been defined on your Linux machine. In fact, by default, all configured print queues on the Linux machine are shared to the network with the `[printers]` share. In this sequence you will create a print queue, which will be shared by the Samba server. Then explore printing to shared printers with `smbclient`.

Tasks:

1. Create a new print queue using system-config-printer. Call the print queue `printerX` where `X` is your station number. Configure the printer to be locally connected to `/dev/lp0`. Configuring the print queue this way will ensure that any submitted jobs will remain in the queue. Don't forget to restart the Samba server.

2. Connect to the `printerX` share on the samba server with `smbclient`. Use the `print` command to submit print jobs to the queue. Verify that the jobs have been queued.

Deliverables:

1. A defined Linux print queue `printerX`

2. A Samba share allows authenticated users to print to the `printerX` share.

Sequence 4: Anonymous upload with `vsftpd`

1. The following package is required: *vsftpd* Check, and install if necessary from CD or from `ftp://server1/pub/RedHat/RPMS/` Activate the `vsftpd` service

2. The *vsftpd* package provides `/var/ftp` for downloads of files by the anonymous FTP user It does not set up an upload directory. Configure `vsftpd` to permit uploads by anonymous users. Prepare a directory for incoming files in this way:

```
cd /var/ftp
mkdir incoming
chown root.ftp incoming
chmod 730 incoming
```

Now verify the permissions on the new directory.

```
ls -ld /var/ftp/incoming
```

3 Set the following lines in `/etc/vsftpd/vsftpd.conf`:

```
anon_upload_enable=YES
chown_uploads=YES
chown_username=daemon
anon_umask=077
```

In addition, `anonymous_enable=YES` should be set already by default.

Restart `vsftpd`.

4 The result of these changes should be that the anonymous FTP user is able to upload files to `/var/ftp/incoming`, but cannot download files from that directory or list files in it This is to stop "warez" traders from using our upload directory as a "drop box" for stolen software or data. Upload a file as the anonymous FTP user It should end up in `/var/ftp/incoming`, owned

by user `daemon` and group `ftp`, with permissions 600 (read-write by user `daemon` only)

Deliverables:

1 A working anonymous FTP server.

2. An available, but "invisible" upload directory via FTP

Sequence 5: NFS

Tasks:

1. The following packages are required: *nfs-utils*. Check and install if necessary from CD or from `ftp://server1/pub/RedHat/RPMS/`. The provides the `nfs` and `nfslock` services.

2. Create a user and configure NFS to share the user's home directory to `example.com` read-write:

   a) Before configuring the NFS server, observe the RPC services you are now running.

   ```
   rpcinfo -p
   showmount -e localhost
   ```

   b) Create a test user.

   ```
   useradd nfstest
   ```

   c) Edit `/etc/exports` such that `/home/nfstest` is shared to `example.com`. See the man page for `exports` if you are unsure about the appropriate format for entries

   d) Installation of the NFS server packages configures NFS to start in runlevels 3 through 5, but the absence of an `/etc/exports` of non-zero size when you booted your system means that NFS will not have started Consequently, start the `nfs` and `nfslock` services manually (it will start automatically after subsequent reboots).

   e) Observe which RPC services are running now and see if you are exporting `/home/nfstest`:

   ```
   rpcinfo -p
   showmount -e localhost
   ```

   f) Work with one or two partners, taking turns mounting each other's shares. Try reading the contents of the share and writing to it as both root and as `nfstest` (modify the `nfstest` user's UID and GID to match your partner's `nfstest` user if they are not the same.) What happens? Why?

Deliverables:

1. A working NFS share of the `/home/nfstest` directory

Challenge Break:

1. Disable the system's "firewall," if neccessary.

   ```
   service iptables stop
   ```

2. Run the following command, following the instruction displayed:

   ```
   tsservices 2
   ```

3. This command will set up the problem and will explain the goal. Refer to the file `/etc/ts` to review the goal. Refer to Lab 1, if you need help.

One Solution:

Sequence 1:

- `rpm -ivh` ftp://server1.example.com/pub/RedHat/RPMS/samba-c* `rpm -ivh` ftp://server1.example.com/pub/RedHat/RPMS/samba-2* `service smb start`
  `smbclient -L localhost -N`

- `useradd -s /bin/false karl`
  `useradd -s /bin/false joe`
  `useradd -s /bin/false mary`
  `useradd -s /bin/false jen`

- `smbpasswd -a karl`
  `smbpasswd -a joe`
  `smbpasswd -a mary`
  `smbpasswd -a jen`

- `smbclient //localhost/joe -U joe`
  You should now see the `smb: \>` prompt
  `put /etc/hosts hosts`

Sequence 2:

- `groupadd -g 30000 legal`
  `usermod -G legal karl`
  `usermod -G legal joe`
  `usermod -G legal mary`
  `usermod -G legal jen`

- `mkdir -p /home/depts/legal`
  `chgrp legal /home/depts/legal`
  `chmod 3770 /home/depts/legal`

- In the `/etc/samba/smb.conf` file, Share Definitions section:

```
[legal]
 comment      = Legal's files
 path         = /home/depts/legal
 public       = no
 write list   = @legal
 create mask  = 0660
```

- `service smb restart`

Sequence 3:

- `system-config-printer`

- `service smb restart`

- `smbclient //localhost/printerX -U joe`

UNIT 4

Electronic Mail Services

redhat

1

Rev RH253-RHEL4-1

Objectives

- Understand electronic mail(email) operation
- Review email transmission
- Basic Sendmail server configuration
- Evaluate the `m4` macro language
- Learn debugging techniques for email servers
- Evaluate Postfix
- Learn to configure Procmail

Agenda

- Sendmail features
- Email overview
- Basic Sendmail configuration
- Using the `m4` macro language
- Debugging Sendmail
- Basic Postfix configuration
- Configuring Procmail

Sendmail Features

- Allows many different types of email addresses to be routed
- Supports virtual domains and users
- Allows masquerading of users and machines
- Provides automatic retry for failed delivery and other error conditions

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Electronic mail(email) supports many old email address and transfer standards, such as BITNET addresses and UUCP email transfer. It can "hide" email sent from users and machines, and can masquerade this email as coming from a given domain, or host It also supports receiving and sending email from virtual domains. Many domains could be mapped to one IP address

Security and "Anti-spam" Features

- Many security features and options:
  - rejects email from unresolvable domains
  - full access control for users, machines, and domains
  - default configuration allows only local connections
  - no longer a setuid root program
- "Anti-spam" features
  - no relaying by default
  - access databases
  - Email header checks
  - interoperability with `spamassassin`

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Unsolicited and unwanted email is called "spam " For more information on spam and spam prevention, refer to `http://spam.abuse.net/`.

While rejecting email for unresolvable domains is often an appropriate policy, this configuration can deny email from dialup connections or laptop users with no reverse name resolution Because of this, Red Hat has included the `FEATURE(`accept_unresolvable_domains')` `m4` macro directive in the default `sendmail` configuration

`spamassassin` is a mail filtering tool which conducts post receipt processing on electronic mail messages. It uses a variety of techniques to examine the headers and the body content of email messages. These include heuristic analysis on mail headers and body text and the use of several real time "black hole" lists and DNS to identify " open relay" hosts, a regular source or relay for spam

An Email Review

- Mail user agent (MUA) passes message to mail transport agent (MTA)
- MTA routes message to destination, giving to other intermediate MTAs as necessary
- Domain MTA passes message to mail delivery agent (MDA)
- User receives message

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The mail delivery process begins when the user decides to send a composed message. The user□s mail agent passes the message along to its configured MTA, usually a central mail gateway. With Sendmail 8.12, the user program calls `sendmail` as a non-privileged mail submission program (MSP) which will relay it to the MTA. This gateway reads the message and extracts the destination addresses from it. The MTA will unravel each email address, gathering networks, machines, and users to whom to send the message.

Once the MTA has verified all destination email addresses, it will notify the MUA that the mail was sent. Next, the MTA will deliver the message to the configured mail exchanger (MX) for each domain; should the primary one be down, the next MX for the domain will be chosen. If no mail exchangers are available (e.g., they□re all down), then the MTA will queue the message and attempt delivery later.

When the message reaches the final destination, the target MTA hands the message to the system MDA. Under many systems, the MTA and the MDA are the same program, `sendmail`. The MDA will store the message in a spool file, or pass it through filters, or any perform whatever other instructions it is given for the particular site.

Users may then retrieve their mail either locally by reading from a spool file, or remotely, by using a protocol such as POP or IMAP.

Server Operations

- User's email agent connects to the local MTA as an unprivileged mail submission program (MSP)
- Local MTA queries DNS for destination's MX
- Local MTA opens a TCP/IP connection to port 25 of the target MX
- Both email servers negotiate a SMTP (Simple Mail Transport Protocol) connection
- Target MX allows or rejects email delivery or relaying based upon its own rulesets

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

You can simulate an SMTP delivery manually with `telnet`. Say you wanted to send email to `bob@example.com`:

```
$ dig example.com mx | grep "MX"
example.com.            1D IN MX            0 lava.example.com.

$ telnet lava.example.com 25
Trying 199.44.172.1
Connected to lava.example.com.
Escape character is '^]'
220 example.com ESMTP Sendmail 8.11.6/8.11.6; Wed, 17 Oct 2001 12:31:04 -0400
HELO mynet.com
250 mynet.com Hello user@mynet.com [199.32.75.1], pleased to meet you
MAIL From: user@mynet.com
250 user@mynet.com... Sender ok
RCPT To: bob@example.com
250 bob@example.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
test
.
250 RAA21966 Message accepted for delivery
QUIT
221 example.com closing connection
```

Service Profile: Sendmail

- Type: System V-managed service
- Packages: *sendmail{,-cf,-doc}*
- Daemons: **`sendmail`**
- Script: **`sendmail`**
- Ports: 25 (smtp)
- Configuration: **`/etc/mail/sendmail.cf`**, **`/etc/mail/submit.cf`**, **`/etc/aliases`**, **`/etc/mail/`**, **`/usr/share/sendmail-cf/`**
- Related: *procmail*

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Sendmail Configuration with the **m4** Macro Language

- **m4** is a macro language that can help configure the **sendmail.cf** file
- Red Hat's default Sendmail configuration is generated from the **m4** specification in **/etc/mail/sendmail.mc**
- Red Hat recommends configuring Sendmail with **m4** using **sendmail.mc** as a starting point

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

To install the `m4` macro compiler and the base `sendmail m4` libraries, install the *m4* and *sendmail-cf* RPM packages

Sendmail **m4** Macro File: Introduction

- All **sendmail.mc** macro configuration files should define the OS type, file locations, desired features, and mailer and user tables
- Step through header and definitions in the **sendmail.mc** below

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.


```
divert(-1)
dnl  Each dnl is a comment: Delete-characters-to-NewLine
dnl  Pull in definitions from distribution
include(`/usr/share/sendmail-cf/m4/cf.m4')
VERSIONID(`linux setup for Red Hat Linux')dnl
dnl  Pull in operating system pre-defines
OSTYPE(`linux')
dnl
dnl  Tell sendmail to run as UID 8 (mail), GID 12 (mail)
define(`confDEF_USER_ID', ``8:12'')dnl
dnl  Disable UUCP relaying, which is on by default
undefine(`UUCP_RELAY')dnl
dnl  Disable BITNET relaying, also on by default
undefine(`BITNET_RELAY')dnl
dnl  Define a 1 minute time for SMTP connections
define(`confTO_CONNECT', `1m')dnl
dnl  If no MX record exists, contact host directly
define(`confFRY_NULL_MX_LIST',true)dnl
dnl  Do not accept email bound for you directly; valid hosts
dnl  should only go in mailertable
define(`confDONT_PROBE_INTERFACES',true)dnl
dnl  Define the location of the procmail delivery agent
define(`PROCMAIL_MAILER_PATH', `/usr/bin/procmail')dnl
dnl  Specify the location of the aliases file
define(`ALIAS_FILE', `/etc/aliases')dnl
dnl  Place some security restrictions
define(`confPRIVACY_FLAGS',`authwarnings,novrfy,noexpn,restrictqrun')dnl
define(`confAUTH_OPTIONS', `A')dnl
define(`confTO_IDENT', `0')dnl
```

Sendmail **m4** Macro File: Features

- Investigate the features enabled and disabled in the continuing example below:


```
FEATURE(`no_default_msa',`dnl')dnl
dnl  Specify the location of the sendmail restricted shell
FEATURE(`smrsh', `/usr/sbin/smrsh')dnl
dnl  Set up mailer table and virtual user table files
FEATURE(`mailertable',`hash -o /etc/mail/mailertable.db')dnl
FEATURE(`virtusertable', `hash -o /etc/mail/virtusertable.db')dnl
dnl  Reject e-mail destined for address.REDIRECT
FEATURE(redirect)dnl
dnl  Append the local hostname on locally delivered mail
FEATURE(always_add_domain)dnl
dnl  Read the list of aliases to use from /etc/mail/local-host-names
FEATURE(use_cw_file)dnl
dnl  Use procmail as the local mailer
FEATURE(local_procmail,`',`procmail -t -Y -a $h -d $u')dnl
dnl  Use /etc/mail/access to accept or reject mail
FEATURE(`access_db',`hash -T<TMPF> -o /etc/mail/access.db')dnl
dnl  Allows blocking of email based on destination
FEATURE(`blacklist_recipients')dnl
dnl  Do not masquerade the domain of mail sent by root
EXPOSED_USER(`root')dnl
dnl  Only listen for connections on the 127.0.0.1 address
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
dnl  Accept mail from IP addresses that do not have a reverse DNS lookup
FEATURE(`accept_unresolvable_domains')dnl
dnl  Set up supported mailers
MAILER(smtp)dnl
MAILER(procmail)dnl
dnl  Act like localhost.localdomain is in /etc/mail/local-host-names
Cwlocalhost.localdomain
```

Sendmail Client Configuration

- Often, clients do not accept incoming mail themselves
  - A central mail server accepts all incoming mail and relays all outgoing mail
    - `MAIL_HUB`, `SMART_HOST` defines
    - Central mail server must allow relaying from the client and have `local-host-names` set up
  - Useful for client to "masquerade" as the server in `From:` addresses
    - `MASQUERADE_AS(`example.com')`

14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

In large organizations, it is common for a central mail gateway to act as a relay for all outbound e-mail, and as a central mail collection point for inbound e-mail Client workstations "masquerade" as that mail server or as the domain (which has an appropriate MX record). This can be set up on the client by adding a few lines to the `/etc/mail/sendmail.mc` file and regenerating `sendmail.cf` from it

To forward all incoming mail to the host mail.example.com for delivery, set

```
define(`MAIL_HUB', `mail.example.com')dnl
```

The central mail server's `/etc/mail/local-host-names` file must include the name of the host to which the incoming mail is addressed for this to work.

To deliver local mail locally but relay outgoing mail through mail.example.com, set

```
define(`SMART_HOST', `mail.example.com')dnl
```

The central mail server must allow the client host to relay through it (possibly by setting up the server's `/etc/mail/access` file) for this to work.

To masquerade addresses so that mail sent by a user at your host looks like it actually came from user@example.com, set

```
MASQUERADE_AS(`example.com')dnl
FEATURE(`allmasquerade')dnl
FEATURE(`masquerade_envelope')dnl
```

It is often a good idea to not to masquerade mail sent by certain users on your host, particularly `root`, `postmaster`, and `mailer-daemon`. You can set this up with `EXPOSED_USER` directives:

```
EXPOSED_USER(`root')dnl
```

Other Valuable **m4** directives

- **FEATURE(`dnsbl')**
  - checks a DNS implemented blackhole list to block email spammers
- **FEATURE(`relay_based_on_MX')**
  - Automatically allows relaying if **sendmail** server is listed as the target domain's MX record

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`FEATURE(`dnsbl')` is used to check a DNS based blackhole list to reject connections from known spammers By default, it points at a subscription service run by MAPS at `blackholes.mail-abuse.org` but can take an argument to point at other blackhole lists. It can be specified several times to point at multiple lists. This feature replaces the older `FEATURE(`rbl')`.

The MAPS (Mail Abuse Prevention System) Realtime Blackhole List homepage at `http://www.mail-abuse.org/rbl/` is an excellent resource that discusses the problems related with spam email and presents some great solutions to identify and prevent spammers.

The relay_based_on_MX feature should only be used in private, well protected networks. Enabling that feature is dangerous because it tells sendmail that if a DNS check reveals that it is supposed to be the MX for a domain then it should relay any mail sent from the domain, regardless of origin. Mail servers on the public internet would then be vulnerable to relay from anyone as any domain's DNS records could be setup to use that host as a relay

While these features are useful, they are not enabled by default.

Additional Sendmail Configuration Files

- **`/etc/mail`** is now considered the default Sendmail configuration directory
- **`virtusertable`** maps virtual addresses to real addresses
- **`access`** specifies rejection or acceptance criteria for email from specified domains

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

There are other files in `/etc/mail` as well. Two of them, `mailertable` and `domaintable` are not used by the default `sendmail` configuration, but are included for completeness.

Another file present in `/etc/mail` is the `Makefile`. Notice that all files in `/etc/mail` must be hashed before use by `sendmail`. This allows the `sendmail` daemon faster access to the information, but requires the system administrator to rehash all files after modification. With the presence of the Makefile, this is trivial; simply type `make`. Restarting `sendmail` using the System V startup script or the `service` command will also rebuild these files.

/etc/mail/virtusertable

Allows multiple virtual domains and users to be mapped to other addresses:

admin@123.com	shopper
admin@xyz.org	jdj
pageme@he.net	lmiwtc@pg.com
@cba.com	cba@aol.com
@dom1.org	%1@dom2.org

redhat 17

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The simplest type of entry in `/etc/mail/virtusertable` maps an email address to a local user This is shown above in the first two lines. Notice that there are no difficulties with a domain alias of `admin` for more than one domain.

Another type of entry is that of a simple forwarder. This allows an incoming email to be sent to the target specified This is often used by people needing a simple to remember email address that are stuck with a complex one. A `sendmail` administrator can simply point the desired email address at the individual user□s email account The third line in the above example is an instance of an email-to-pager gateway with an assigned address, and a simple alias pointing to it.

The last type of entry is the advanced feature of forwarding an entire domain□s email to a given (local or remote) address This is a very useful feature for administrators of virtual domains that want to avoid the setup of specific aliases. This can be seen in that last two lines of the example `virtusertable` file above The last line adds a twist to this concept, and forwards the incoming email to the user specified in the `dom1.org` target as a user at `dom2.org` For example, `jonny@dom1.org` will map to `jonny@dom2.org`

Any changes made to the `virtusertable` file must be mapped before sendmail can use them. If the admin adds a new virtual alias he or she must run `make` before it will take effect.

/etc/mail/access

Used to accept or deny incoming email:

```
90trialspammer@aol.com   REJECT
spamRus.net              REJECT
204.168.23               REJECT
10.3                     OK
virtualdomain1.com       RELAY
user@dom9.com    ERROR:550 mail discarded
nobody@          ERROR:550 bad name
```

redhat 18

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The `/etc/mail/access` database file can reject email from individual users (the first line above), entire domains (the second line), or an entire IP subnet (third line).

As shown on the fourth and fifth lines, the access database can also direct `sendmail` to accept email from subnets and domains. The second column of this file can be one of a few different values:

`REJECT`
: rejects the sender with a general purpose message

`OK`
: accepts mail (for receipt, *not relay*) even if other rules, such as failed DNS lookup, might reject it

`RELAY`
: accept mail for relaying

`DISCARD`
: discard the message completely (harsher than reject)

`ERROR:550` *`your message`*
: like `REJECT` but returns with your specific message

Email addresses which are not listed in the `access` file are simply processed according to what ever other rulsets are in place. Additions to the `access` file require that the administrator run `make`. Essentially the `access` file gives the administrator the ability to create rules which may explicitly block or allow addresses, domains or dotted quad ip addresses or networks.

Blacklisting Recipients

- `FEATURE('blacklist_recipients')`
  - Block mail destined for certain recipients
- Any entry in the `access` file that has a `REJECT` or returns an error code will be a blacklisted recipient

Blacklisting blocks both to and from the blacklisted party.

Note that this `m4` Macro feature is enabled by default in `sendmail.mc`

The `access` file allows the admin to disable mail for a local user as well as remote users or mail servers trying to send mail inbound. Adding the `blacklist_recipents` feature means that an admin may also disable all mail inbound or outbound for a user with a local account on the mail server so that they cannot send or receive any email at all.

Debugging Sendmail

- **/etc/mail/local-host-names**
  - must contain server's name and aliases
- **mail -v user**
  - view SMTP exchange with local relay
- **mailq** and **mailq -Ac**
  - view messages queued for future delivery
- **tail -f /var/log/maillog**
  - View log in real-time

redhat 20

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Sendmail will not accept mail for local delivery for hosts that are not specified by name in `/etc/mail/local-host-names`. This file contains a list of host names which, if seen in an e-mail address, should be delivered locally. The host names should resolve to your server or have a MX record pointing to your server.

In Sendmail 8.12, the `mail -v` command displays the SMTP exchange between the MSP and the local relay MTA only. It is useful to debug local `sendmail` daemon configuration. To send a test message and view the SMTP exchange, use

```
mail -v user
```

type the message and press Control-D to send. (In older versions of Sendmail, there was no unprivileged MSP – the user's e-mail program ran a set-uid MTA directly, and `mail -v` instead displayed the local MTA to remote MTA communication.)

The `mailq` command is useful for displaying messages waiting in a queue for delivery. By default, it displays the queue of messages waiting to be processed by your local MTA for delivery or relay to a remote MTA. Messages can sit in this queue for a long time if the remote host is refusing connections. If you run `mailq -Ac`, the queue of messages waiting to be sent by your MSP to the local MTA relay will be displayed instead. Messages can sit in this queue for a long time if your local host is having problems with name resolution.

During any server configuration or testing, monitoring the appropriate logs with `tail -f` can be invaluable. Note that the default `/etc/syslog.conf` file precedes the `/var/log/maillog` target with a '-' (dash) symbol to disable syncing the file after every logging. You can remove the dash to enable syncing and get real-time logging from the `tail -f` command.

Using **alternatives**

- **alternatives** configures the server software through a *generic name*
  - generic name is a link to a link in /etc/alternatives/
  - only the links in /etc/alternatives/ are modified
- **alternatives** displays and sets link groups
  - **alternatives --display** *name*
  - **alternatives --config** *name*
- **system-switch-mail**

redhat 21

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The `alternatives` system manages many packages in the distribution that provide the same service. For instance, both Sendmail and Postfix provide electronic mail support, but only one of them should be used at a time on a single machine.

With `alternatives`, an executable with a generic name on the filesystem is used to access a particular service. This executable is really a symbolic link to another symlink in the `/etc/alternatives/` directory. For example, `/usr/sbin/sendmail` is really a link to `/etc/alternatives/mta`. In order to select between Sendmail or Postfix, we just change the symlink for `/etc/alternatives/mta`. This is normally done with the `alternatives` command. Some example commands are listed below.

To display which MTA alternative is in use:

```
alternatives --display mta
```

To choose from the available MTA alternatives from the command line:

```
alternatives --config mta
```

To set up Postfix as the default mail system:

```
alternatives --set mta
```

You may use the GUI tool `system-switch-mail`, if installed, to make these changes. Note that it calls alternatives to effect the configuration.

Postfix

- A replacement for Sendmail
- Project goals:
  - Sendmail-compatible
  - Speed
  - Ease of Administration
  - Security
- Efficient application design based on a modular suite of programs

redhat 22

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Sendmail-compatible

Postfix is an alternative mail server implementation for Unix and Linux. Postfix is designed to be compatible with Sendmail, and can use `/var/spool/mail`, `/etc/aliases`, NIS maps, and `.procmailrc` files. This simplifies migration of a server using Sendmail to Postfix. It is designed to improve on Sendmail in several key areas:

Speed

A desktop-class system running Postfix can receive and deliver as many as a million different messages per day. Postfix uses techniques used in web servers to reduce process creation overhead, and other tricks to reduce file system overhead.

Ease of Administration

Unlike the complex and unwieldy `sendmail.cf`, Postfix configuration files consist of well-commented, human-readable directives.

Security

There is no direct connection from the network to the security-sensitive local delivery programs. Almost every program that makes up Postfix can run in a `chroot` "jail" with minimal privileges. No part of Postfix is set-uid.

Postfix comprises several individual programs to process mail, managed by a supervisor daemon called `master`. The file `/etc/postfix/master.cf` configures how the subsidiary processes should be run. The `master` daemon does not handle mail directly, but manages the other daemons that do.

Some of these other daemons are:

`smtpd`	Listens on port 25 for incoming messages and submits them to the "incoming" queue.
`pickup`	Moves messages sent by the local Postfix server from the "maildrop" queue to the "incoming" queue.
`nqmgr`	Passes messages from the "incoming" queue to various processes for transmission, relaying, or local delivery.

A diagram of the Postfix system is available at `/usr/share/doc/postfix-*/html/big-picture.html`.

Service Profile: Postfix

- Type: SystemV-managed service
- Packages: *postfix*
- Daemons: `master`, `nqmgr`, `smtpd`, `pickup`, (others)
- Script: `postfix`
- Ports: 25 (smtp)
- Configuration: `/etc/postfix/main.cf` `/etc/postfix/master.cf`
- Related: *procmail*

redhat 23

Rev RH253-RHEL4-1

Configuring Postfix

- Activate with `alternatives`
- Set up minimal configuration directives
  - using `postconf`
  - using a text editor
- Start with `service`

redhat 24

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Setup

Once Postfix is installed, it may be activated with alternatives:

```
# alternatives --set mta /usr/sbin/sendmail.postfix
```

Configure

While Postfix has several hundred configuration paramaters , most of the defaults are sensible. At a minimum, you should configure the following values, listed below. Here we use `postconf` to set configuration options You may also use a text editor. In either case, edits are made to one file (`/etc/postfix/main.cf`) and applied by reloading the service.

```
# postconf -e "myorigin = redhat.com"
# postconf -e "mydestination = redhat.com dawg.redhat.com"
# postconf -e "mynetworks = 192.168.0.0/24, 127.0.0.1"
# postconf -e "inet_interface = all"
```

Use `postconf` again to inspect your results, as recorded in `/etc/postfix/main.cf`:

```
# postconf -n
[ ... output of non-default configuration omitted ]
```

Start

Once Postfix is configured, start the mail system with `service`:

```
# service postfix start
```

Additional Postfix Configuration

- **/etc/postfix/** files share syntax and function with those of **/etc/mail/**
  - **virtual** - virtual domain mapping
  - **access** – mail routing controls
- **/etc/aliases** can be used by postfix, as is
- Postfix command utilities
  - **postmap**
  - **postalias**

redhat 25

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

One of the goals of `postfix` is to be very `sendmail` compatible You may use the `sendmail` style *alias* and *virtualuser* files. As the format of these files is the same used by `sendmail`, refer to the previous `sendmail`discussion(c.f. 10, 17 and 18). Configuration does vary slightly; for example: virtual domains must be added to the `mydestination` line in `/etc/postfix/main.cf`.

Like `sendmail`, `postfix` requires that it's *access* and *alias* files be hashed prior to use. RHEL's `sendmail` implementation provides two different tools to do this (the `Makefile` in `/etc/mail` and the `newaliases` command) The `postfix` tools used to accomplish this same task are `postmap` and `postalias`. The `postmap` command is used for the *access* file, and any other file which requires hashing, **except** the *alias* file (this may also include a separate *alias* file if you choose to run `listserv` software with `postfix`) To hash the access file, the appropriate command would be:

```
# /usr/sbin/postmap /etc/postfix/access
```

To hash the *aliases* file simply run the `postalias` command (no argument is required as the command reads the postfix configuration to identify which file should be hashed)

In addition to supporting `sendmail` style files, postfix offers other configuration options and methods. Documentation on available options may be found in `/usr/share/doc/postfix-*` and at the website `http://www.postfix.org`.

Enhanced Postfix Configuration

- Pre-receipt header and body checks
- Multiple transports (uucp, X.400)
- Virtual domain support
- UCE controls (blacklists, helo/sender)
- Table lookups (SQL, LDAP)

redhat 26

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Postfix is rapidly reaching critical mass in both installations and functionality. It's unique combination of performance and light resource usage makes it an excellent choice as a mail server. The pre-receipt header and body checks are unique and make postfix an overwhelming favorite among knowledgable and security conscious electronic mail system administrators

Using header and body checks is done using regular expressions or PERL-compatible regular expressions (PCRE). The important thing to note here is that this is *pre-recipt* functionality which means that while the email is being received it's header and/or body text are analyzed and if there is a regular expression match the email is immediately rejected prior to receipt. This makes postfix a great tool with which to prevent the transmission of viruses and spam!

Here is a sample header and body check configuration for `/etc/postfix/main.cf`:

```
header_checks = regexp:/etc/postfix/header_checks
body_checks = regexp:/etc/postfix/body_checks
```

Once these directives are added, regular expressions must be entered into the two files listed in the directives. Here are a pair of examples for each:

```
/etc/postfix/header_checks:

/^Date: .* 200[0-2]/                          REJECT
/^Subject:\s+.*your account\s{5,}.{8}/        REJECT

/etc/postfix/body_checks:

/Lenders Compete For Your Business/           REJECT
/Cheapest Viagra Anywhere/                    REJECT
```

Procmail Delivery

- Procmail is a very powerful delivery tool
- Different uses include
  - sorting incoming email into different folders or files
  - preprocessing email
  - starting an event or program when email is received
  - Automatically forwarding email to others
- Additional MTA configuration may be required

redhat 27

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Information on `procmail` configuration can be found in /usr/share/doc/procmail-<version>, or the `procmailex` manual page (`man procmailex`). For further information on `procmail`, refer to `http://www.procmail.org/`.

Both sendmail and postfix have to be configured to use procmail as their local delivery agent. For sendmail ensure that the following lines are enabled in your `/etc/mail/sendmail.mc` file:

```
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl
FEATURE(local_procmail,``,`procmail -t -Y -a $h -d $u')dnl
MAILER(procmail)dnl
```

The default sendmail configuration provided by the RHEL `sendmail` package provides this configuration.

By default postfix uses its own delivery agent. To enable postfix to use `procmail` for email delivery, configure the following in the file `/etc/postfix/main.cf`:

```
# postconf -e " mailbox_command = /usr/bin/procmail"
```

Once your MTA has been configured to use `procmail` you may implement a system-wide configuration (`/etc/procmailrc`), or by individual user (`$HOME/.procmailrc`) to sort mail or forwarded mail to a remote host.

Note: the use of `~/.forward` files for this purpose is deprecated.

Procmail Sample Configuration

- Usually located in a user's home directory:
  `/home/bob/.procmailrc`
- To forward mail from Joshua about ADSL to Jim, but also copy to the ADSL folder:

```
:0
*^From.*joshua
*^Subject:.*ADSL
{  :0 c
! Jim@somedomain.org
:0:
ADSL
}
```

redhat 28

RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

For more information and examples of using `procmail` in many scenarios, refer to:
`http://www.ling.helsinki.fi/users/reriksso/procmail/mini-faq.html`.

Many administrators may find `spamassassin` useful to help with the large amount of unsolicited commercial email (UCE) their domain receives daily. A user may also call `spamassasin` individually. RHEL provides the `spamassasin` package, and if installed, the following lines at the beginning of the user's `procmail` configuration enables this software:

```
SPAMFOLDER=spam
#
# SpamAssassin check
:0 wf
| /usr/bin/spamassassin
#
# File as SPAM
:0 w :$SPAMFOLDER/.lock
* ^X-spam-status: Yes
$SPAMFOLDER/.
```

MAN PROCMAIL
PROCMAILEX

End of Unit 4

- Address questions
- Preparation for Lab 4
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 29

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 4
Electronic Mail

Estimated Duration: 1 hour

Goal: To build skills at basic `MTA` configuration

Instructor: *Make sure* `server1.example.com` *can receive mail from other hosts*

Introduction

This lab serves as in introduction to installation and configuration of an MTA. Instructions are provided for both `sendmail` and `postfix`. You may choose to setup either MTA, or if time permits, both. In the following sequences, you will

- Install and verify `sendmail` "out of the box"
- Add new aliases to your `sendmail` installation
- Use the `m4` utilities to change your relaying behavior
- Install a POP server and configure a POP client

Throughout this lab, the host and domain names that you use will be based upon the IP address of your machine. Any time the lab refers to a name that contains *X*, you should replace *X* with your station number (the last segment of your IP address). For example, if your station's IP address is 192.168.0.2, you would replace references to `stationXX.example.com` with `station2.example.com`.

Disable packet filtering. Before beginning this lab, make sure all packet filtering is turned off on your host (obviously, you should take advantage of the Linux kernel's firewalling capabilities in practice, but for our purposes disabling packet filtering lessens the potential for problems). These and the following commands in this lab will need to be run as the root user.

Initial Setup - Installing the necessary packages

The following packages are required for `sendmail`: *sendmail*, *sendmail-cf*, *m4*, and *procmail*. For `postfix` you will need: *postfix*. Check and install them as needed (see Appendix item 1)

Sequence 1: Configuring the MTA to receive mail

For security reasons, the default configuration of `sendmail` and `postfix` allows sending, but not receiving email over the net (by default it will only accept connections on the loopback interface) Configure your chosen MTA to accept incoming connections by as follows:

1. **For `sendmail`:** Make a backup copy of your original `sendmail.cf` and `sendmail.mc`:

```
cp /etc/mail/sendmail.cf /etc/mail/sendmail.cf.orig
cp /etc/mail/sendmail.mc /etc/mail/sendmail.mc.orig
```

2. Modify `/etc/mail/sendmail.mc`

   Comment out the line below by prepending it with "`dnl `", like so:

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
```

3. Build a new `sendmail.cf` in the same directory.

```
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

4. Restart Sendmail with:

```
service sendmail restart.
```

For `postfix`: modify `/etc/postfix/main.cf`

A. Run the command:

```
postconf -e "inet_interfaces = all"
```

B. Proceed to the next Postfix instructions toward the end of Sequence 2

Sequence 2: Starting and verifying MTA operation

For `sendmail`: there are several steps that you should take to confirm that `sendmail` has been properly installed.

A. Confirm that `sendmail` is enabled for the appropriate runlevels

Check to ensure your machine is properly configured to restart `sendmail` upon reboot. This is most conveniently done with `chkconfig`

```
chkconfig --list sendmail
sendmail 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

If `sendmail` is disabled for the standard user runlevels, use a utility like `chkconfig`, `ntsysv`, or `serviceconf` to enable the service.

B. Confirm that `sendmail` started without error

The Red Hat Linux installation of sendmail is configured to use the syslog facility to log all messages to the file `/var/log/maillog`. Examine this file, perhaps searching for the last instance of the word "starting". Ensure that `sendmail` started without error.

The sendmail executable is `/usr/sbin/sendmail`. In order to determine if `sendmail` is identifying your station's hostname correctly, invoke it with its debugging command line switch set to 0:

```
sendmail -d0 < /dev/null

Version 8.11.2
Compiled with: LDAPMAP MAP_REDGEX LOG MATCHGECOS MIME7TO8 MIME8TO7
NAMED_BIND NETINET NETUNIX NEWDB NIS QUEUE SASL SCANF SMTP
TCPWRAPPERS USERDB

============ SYSTEM IDENTITY (after readcf) ============
      (short domain name) $w = station2
  (canonical domain name) $j = station2.example.com
         (subdomain name) $m = example.com
              (node name) $k = station2.example.com
=======================================================

Recipient names must be specified
```

If `sendmail` is returning your station name as `localhost`, you probably have a misconfigured `/etc/hosts`. Examine your `/etc/hosts`, and remove all *but* the `localhost` hostname references, and try again. If `/etc/hosts` appears correct, check the definition of `HOSTNAME` in `/etc/sysconfig/network`.

Try mailing a sample message to `root@server1`. You should see a reasonable SMTP exchange with your local relay server:

```
echo "hello root" | mail -v -s hello root@server1

root@server1.... Connecting to localhost.localdomain. via relay...
220 station6.example.com ESMTP sendmail 8.12.5/8.12.5; Thu, 24 Oct 2002
14:54:43 -0400
>>> EHLO station6.example.com
250-station6.example.com Hello localhost.localdomain [127.0.0.1], pleased
to meet you
...
>>> MAIL From:<root@station6.example.com> SIZE=44
250 2.1.0 <root@station6.example.com>.... Sender ok
>>> RCPT To:<root@server1.example.com>
250 2.1.5 <root@server1.example.com>.... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 2.0.0 g9OIGEJk014797 Message accepted for delivery
root@server1.... Sent (g9OIGEJk014797 Message accepted for delivery)
Closing connection to localhost.localdomain.
>>> QUIT
221 2.0.0 station6.example.com closing connection
```

If the SMTP exchange completes correctly, as above, then the message has been passed to the local relay server on your station, and `mailq -Ac` should report an empty queue. Then check `mailq` (with no arguments) to see if the message was passed from the local relay to server1. That queue should also be empty.

Did your message get logged appropriately in `/var/log/maillog`? During the following sequence, monitor the `/var/log/maillog` file. The following command will be helpful:

```
xterm -e tail -f /var/log/maillog &
```

For postfix:

A. Run 'service sendmail stop,' then use `system-switch-mail` to make `postfix` the active MTA. You may also do this step at the command line:

```
alternatives --set mta /usr/sbin/sendmail.postfix
```

B. Confirm that `postfix` is enabled for the appropriate runlevels

```
chkconfig --list postfix
postfix          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

C. Confirm that the `hostname` command returns the correct value for your station. It should return your FQDN.

If `hostname` is returning your station name as `localhost`, you probably have a misconfigured `/etc/hosts`. Examine your `/etc/hosts`, and remove all but the `localhost` hostname references, and try again. If `/etc/hosts` appears correct, check the definition of HOSTNAME in `/etc/sysconfig/network`. When this value is correct, start the `postfix` service.

D. Confirm that `postfix` started without error

Like sendmail, the Red Hat Linux installation of `postfix` is configured to use the syslog facility to log all messages to the file `/var/log/maillog`. Examine the end this file and look for any error messages

Try mailing a sample message to `root@server1` and check the results in `/var/log/maillog`:

```
mail -s `echo $USER` root@server1 < /etc/redhat-release
```

Look for lines like the following:

```
Mar  8 09:57:59 station6 postfix/pickup[24288]: F147040024: uid=0 from=<root>
Mar  8 09:57:59 station6 postfix/cleanup[24636]: F147040024: message-
id=<20030308145759.F147040024@station6.example.com>
Mar  8 09:58:00 station6 postfix/nqmgr[1141]: F147040024:
from=<root@station6.example.com>, size=325, nrcpt=1 (queue active)
Mar  8 09:58:16 station6 postfix/smtp[24708]: F147040024: to=<root@server1>,
relay=server1.example.com[192.168.0.254], delay=17, status=sent (250 ok dirdel)"
```

Sequence 3: Adding new aliases

For sendmail:

Before sendmail determines the destination of a message recipient, it undergoes an attempt at alias resolution. The primary configuration file for sendmail aliases is /etc/aliases. In order to optimize lookups, sendmail generates a hashed database for its alias records, /etc/aliases.db This file is generated with the newaliases command (which is a synonym for sendmail -bi).

The following command will add the user student(if not already present).

```
useradd student
```

Add the following lines to /etc/aliases.

```
me: student
wizards: root, me
methere: student@stationX.example.com
```

Now run the newaliases command to update the database, and try sending mail to the recepient aliases that you defined

```
newaliases
echo "hello there" | mail -s "hello" me
echo "hello there" | mail -s "hello" wizards
echo "hello there" | mail -s "hello" methere
```

Did you get the expected results? Did all recepients on the "wizards" list receive a copy of the mail? If not, su - to a user *other than* root and try again.

For postfix:

Before postfix determines the destination of a message recipient, it makes an attempt at alias resolution. The primary configuration file for postfix aliases is /etc/aliases In order to optimize lookups, postfix generates a hashed database for its alias records(just like sendmail) named /etc/aliases.db This file is generated with the postalias command.
The following command will add the user student(if not already present).

```
useradd student
```

Add the following lines to /etc/aliases.

```
me: student
wizards: root, me
methere: student@stationX.example.com
```

NOTE: Be sure to edit the line that aliases root to postfix! Set it to *your* (non-root) account.

Now run the command 'postalias /etc/aliases' to update the database, and try sending mail to the recipient aliases that you defined

```
echo "hello there" | mail -s "hello" me
echo "hello there" | mail -s "hello" wizards
echo "hello there" | mail -s "hello" methere
```

Did you get the expected results? Did all recipients on the "wizards" list receive a copy of the mail?

Sequence 4: Controlling Relaying

Relaying allows mail to be directed to its destination by use of an intermediary "relay" machine. Although at times useful, relaying has been a source of abuse on the internet by "spammers" People wanting to send unsolicited mail often make use of relaying in order to make the mail's origin harder to determine.

The following sequence will make use of the following hosts. Replace X, Y, and Z with appropriate station numbers :

- `stationY`: the source machine, where the mail originates.
- `stationX`: the relaying machine, where the mail is sent by the originator
- `stationZ`: the destination machine, the final destination of the mail

This sequence assumes that you are at `stationX`, the relaying machine, and have partnered with someone at `stationY`, the machine from which mail originates. During the sequence, keep an eye on `/var/log/maillog`. The following command will help:

```
xterm -e tail -f /var/log/maillog &
```

Scenario A: allowing relaying

For `sendmail`:

You have the ability to control who you allow to relay from your machine. By configuring your machine to use promiscuous relaying, you allow anyone to use your machine as a relay. (We advise against this practice, and hope that this exercise will make its pitfalls evident.) Configure the `/etc/mail/sendmail.mc` `m4` preprocessor file to enable promiscuous relaying by adding the following line just above the last lines of the file beginning with "MAILER":

`/etc/mail/sendmail.mc`

```
(... other entries...)
FEATURE(promiscuous_relay)dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

Use the `m4` preprocessor and this template file to generate a new `sendmail` configuration file, and then compare the newly generated file to the `sendmail.cf` file supplied by the *sendmail* RPM:

```
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.test-relay
diff /etc/mail/sendmail.test-relay /etc/mail/sendmail.cf
```

How much difference did allowing the promiscuous relaying feature make? Now move the newly created `sendmail.test` file into place and restart `sendmail`:

```
mv /etc/mail/sendmail.cf /etc/mail/sendmail.cf.accept-mail
cp /etc/mail/sendmail.test-relay /etc/mail/sendmail.cf
service sendmail restart
```

Have your partner take the role of a malicious spammer, who will now be able to use your `sendmail` to spoof the origins of unwanted mail by telnetting directly to your `smtp` (`sendmail`) port 25, and typing the following sequence of commands from `stationY`:

This example is for `stationY` *(the originating host)* = `station2`, *and* `stationX` *(the relaying, and in this case, the destination host)* = `station1`.

```
[root@station2 /root]# telnet station1 25
Trying 192.168.0.1...
Connected to station1.example.com.
Escape character is '^]'.
220 station1.example.com ESMTP sendmail 8.11.2/8.11.2; Mon, 7 May 2001
09:26:44 -0400
helo mail.cracker.org
250 station1.example.com Hello IDENT:root@station2.example.com
[192.168.0.2], pleased to meet you
mail From: spammer@cracker.org
250 spammer@faked.com... Sender ok
rcpt To: root@station1.example.com
250 root@station1.example.com... Recipient ok
data
354 Enter mail, end with "." on a line by itself
Subject: Faked
this was faked!
.
250 JAA00748 Message accepted for delivery
quit
221 station1.example.com closing connection
Connection closed by foreign host.
```

Spammed mail has now been sent to your machine Next, see if your partner can use your machine to relay mail to a third machine:

This example is for stationY *(the orginating host)* = station2 *and* stationX *(the relaying host)* = station1, *and* stationZ *(the destination host)* = station3

```
[root@station2 /root]# telnet station1 25
Trying 192.168.0.1...
Connected to station1.example.com.
Escape character is '^]'.
220 station1.example.com ESMTP sendmail 8.11.2/8.11.2; Mon, 7 May 2001
09:28:58 -0400
helo mail.cracker.org
250 station1.example.com Hello IDENT:root@station2.example.com
[192.168.0.2], pleased to meet you
mail From: spammer@cracker.org
250 spammer@cracker.com... Sender ok
rcpt To: root@station3.example.com
250 root@station3.example.com... Recipient ok
data
354 Enter mail, end with "." on a line by itself
Subject: Relayed
this was faked and relayed!
.
250 JAA00752 Message accepted for delivery
quit
221 station1.example.com closing connection
Connection closed by foreign host.
```

Because your machine is configured for promiscuous relaying, the spammer was able to relay mail through your machine

For postfix:

By default, Postfix disallows relaying To enable relaying to systems on your subnet, run:

```
postconf -e "mynetworks_style = subnet"
service postfix restart
```

Have your partner take the role of a malicious spammer on your subnet, who will now be able to use your postfix to spoof the origins of unwanted mail by telnet directly to your SMTP port 25, and typing the following sequence of commands from stationY:

```
[root@station2 /root]# telnet station1 25
Trying 192.168.0.1...
Connected to station1.example.com.
Escape character is '^]'.
220 station1.example.com ESMTP Postfix
helo mail.cracker.org
250 station1.example.com
mail From: spammer@cracker.org
250 Ok
rcpt To: root@station1.example.com
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: Faked
this was faked!
.
250 Ok: queued as 2F7BDCB882
quit
221 Bye
Connection closed by foreign host.
```

Spammed mail has now been sent to your machine. Next, see if your partner can use your machine to relay mail to a third machine:

This example is for `stationY` (the orginating host) = `station2`, and `stationX` (the relaying host) = `station1`, and `stationZ` (the destination host) = `station3`

```
[root@station2 /root]# telnet station1 25
Trying 192.168.0.1...
Connected to station1.example.com.
Escape character is '^]'.
220 station1.example.com ESMTP Postfix
helo mail.cracker.org
250 station1.example.com
mail From: spammer@cracker.org
250 Ok
rcpt To: root@station3.example.com
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: Relayed
this was faked and relayed!
.
250 Ok: queued as 2F7BDCB884
quit
221 Bye
Connection closed by foreign host.
```

Because of your machine's default configuration, the spammer was able to relay mail through your machine.

Scenario B: disallowing relaying

For sendmail:

Restore the default sendmail configuration by replacing the new sendmail.cf with the configuration that accepts incomming mail and restarting sendmail:

```
mv /etc/mail/sendmail.cf.accept-mail /etc/mail/sendmail.cf
service sendmail restart
```

Have your partner at stationY attempt to relay spammed mail again Did your sendmail act as a relay? An attempt to relay should have produced a message like the following:

```
550 root@station3.example.com... Relaying denied
```

For postfix:

To disable relaying, execute the following commands:

```
postconf -e "mynetworks_style = host"
service postfix reload
```

Have your partner at stationY attempt to relay spammed mail again. Did your postfix act as a relay? An attempt to relay should have produced a message like the following:

```
554 <root@station3.example.com>: Recipient address rejected: Relay access denied
```

Scenario C: Selectively allowing relaying

For sendmail:

To allow relaying from a specific host, domain, or network, edit /etc/mail/access and restart sendmail. To allow only hosts in the example.com domain to relay mail through your machine, add an entry to /etc/mail/access that permits this. Test with your partner using the commands in *Scenario A*.

For postfix:

To allow relaying from only certain hosts through your system, run:

```
postconf -e "mynetworks_style = host"
```

Then run the following to allow a specific network, or host to relay(here, stationY).

```
postconf -e "mynetworks = 192.168.0.Y, 127/8"
service postfix reload
```

Test with your partner using the commands in Scenario A.

Sequence 5: Installing a POP server and client

This sequence will have you configure your machine, `stationX`, as a mail POP server, and have a partner at `stationY` take the role of a POP client.

Step A: Installing the POP server

Configuring a POP server tends to be straightforward, involving only two steps:

- Installing the relevant RPMs
- Starting the service

Installing the relevant packages

The POP daemon is bundled with another daemon that provides similar functionality, the IMAP daemon, and both are found within the *dovecot* package. Load the *dovecot, mysql, postgresql perl-DBD-MySQL,* and *perl-DBI* packages and examine what the *dovecot* package contains.

Three daemons are included: `imapd`, `ipop2d`, and `ipop3d`. POP3 is used by many Internet Service Providers; POP2 is provided for backwards compatibility. The IMAP daemon provides more sophisticated capabilities, including folder management on the server.

Starting the service:

Edit the `/etc/dovecot.conf` file and add the following line (to start the POP3 and secure POP3 servers):

```
protocols = pop3 pop3s
```

`ipop3d` is started on demand. To activate, run the following command:

```
service cyrus-imapd start
```

Confirming the service:

Confirm that the POP3 daemon has been properly installed by running the following sequence of commands. Use the following commands as a guide:

```
echo "mail to be popped" | mail -s "hello student" student
```

Then, connect to the POP server:

```
telnet localhost 110

Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
+OK POP3 localhost.localdomain v7.64 server ready
USER student
+OK User name accepted, password please
```

```
PASS student
+OK Mailbox open, 1 messages
STAT
+OK 1 384
TOP 1 99999
+OK Top of message follows
Return-Path: <student>
Received: (from student@localhost)
   by station1.example.com (8.11.2/8.11.2) id IAA00917
   for student1; Mon, 7 May 2001 08:00:00 -0400
Date: Mon, 7 May 2001 08:00:00 -0400
From: student@station1.example.com
Message-Id: <200001121300.IAA00917@station1.example.com>
To: student@station1.example.com
Subject: hello student
Status:
mail to be popped
.
DELE 1
+OK Message deleted
QUIT
+OK Sayonara
```

If all went well, you should now have a properly installed POP server.

Step B: Using a POP client

Most popular Mail User Agents (MUAs) today, such as *mozilla*, *mutt*, *pine*, and *evolution*, are POP-aware, and can be used as POP clients The configuration of each depends upon the implementation There is also a popular command-line POP client called `fetchmail` `fetchmail` is highly configurable, can query multiple mailboxes, and can run in daemon mode, such that it could query a user's mailbox every five minutes. `fetchmail` delivers the mail passing it off to the Mail Transport Agent (MTA) on the local host, usually `sendmail`. We will outline the steps for installing `fetchmail` and using it to query the POP server which we just installed.

If neccessary, install the *fetchmail* package.

Note that there is a wide range of options available for configuring `fetchmail`'s behavior. Create a `~/.fetchmailrc` file that looks like the following:

~student/.fetchmailrc

```
poll stationX.example.com with protocol pop3:
   user student there is user student here
   password "password"
```

Because passwords are stored in this file, `fetchmail` will refuse to run unless you make the file readable only by owner Note that you should also chown the file such that it is owned by `studentXX` if you created it as root.

```
chmod 600 ~student/.fetchmailrc
chown student.student ~student/.fetchmailrc
```

Attempt to "pop" your mail as `studentXX`.

```
echo "hello student" | mail -s "Hola" student
su - student
fetchmail -v
exit
```

Was `fetchmail` able to POP `student`'s mail? Where did it deliver `student`'s mail to? Does it make much sense to POP mail from the localhost?

Have a partner at another machine set up a similar `~/.fetchmailrc` file (or, alternately, configure some other MUA such as `mozilla`) and then try retrieving mail from your server using POP3

Challenge Break:

1. Disable the system's "firewall," if necessary. Also switch back to using `sendmail`.

   ```
   service iptables stop
   ```

2. Run the following command, following the instruction displayed:

   ```
   tsservices 4
   ```

3. This command will set up the problem and will explain the goal. Refer to the file `/etc/ts` to review the goal. Refer to Lab 1, if you need help.

UNIT 5

The HTTP Service

redhat

1

Rev RH253-RHEL4-1

Objectives

- Learn the major features of the Apache HTTP server
- Be able to configure important Apache parameters
- Learn per-directory configuration
- Learn how to use CGI with Apache
- Identify key modules
- Understand proxy web servers

redhat 2

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Tux web server - built in

Agenda

- Introduce Apache Features
- Apache configuration files and important parameters
- Using CGI with Apache
- Key modules
- Squid proxy server

Apache Overview

- Process control:
  - spawn processes before needed
  - adapt number of processes to demand
- Dynamic module loading:
  - run-time extensibility without recompiling
- Virtual hosts:
  - Multiple web sites may share the same web server

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The Apache server has a flexible mechanism for accepting requests and dispatching children to process them which is abstracted into "Multi-Processing Modules". The MPM used by default in Red Hat Enterprise Linux(RHEL) is `prefork`, which spawns multiple child processes when needed just like Apache 1.3. Other MPMs are not yet available, although directives for some appear in the configuration file.

Dynamic module loading allows a web server administrator to change the behavior of Apache. This can be done without recompiling any source code, and simply specifying the use of a given module. An example of a commonly used module is `mod_perl`, used to increase Perl CGI script execution speed.

The Apache HTTP Server project web site is `http://httpd.apache.org`.

Service Profile: HTTPD

- Type: SystemV-managed service
- Packages: *httpd, httpd-devel*
- Daemons: **`httpd`**
- Script: **`httpd`**
- Ports: 80(http), 443(https)
- Configuration: **`/etc/httpd/*, /var/www/*`**
- Related: *system-config-httpd, mod_ssl*

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The Apache web server is an SELinux restricted service when enforcing the default targeted policy on a Red Hat Enterprise Linux, version 4 system. The server uses a number of SELinux contexts for its files. For purposes of web server configuration, the following contexts are important: Not Cool

`system_u:object_r:httpd_config_t`
For configuration files, particularly in `/etc/httpd/conf` and `/etc/httpd/conf.d`.

`system_u:object_r:etc_t`
For the files in `/etc` that are not in `/etc/httpd`, such as the `httpd` files in `/etc/sysconfig` and `/etc/logrotate.d`.

`system_u:object_r:httpd_log_t`
For log files in `/etc/httpd/logs`.

`system_u:object_r:httpd_modules_t`
For modules used with `httpd`.

For purposes of web content creation, the following context is vital:

`system_u:object_r:httpd_sys_content_t`
For web content. The `/var/www/html` file is set to this context. Any web content created outside this directory must have the context set to this. This includes standard error messages, icons, and any other file that will be distributed by the `httpd` server.

If you move a directory tree to `/var/www/html` (or a subdirectory thereof), you will want to set the context. For example, perhaps you copy the data directory to this tree. Run:

```
chcon -R --reference=/var/www/html /var/www/html/data
```

The `httpd` server also has an option which will check the syntax of the configuration files only (`httpd -t`). The program immediately exits after these syntax parsing tests and displays either a `Syntax OK` or `Syntax Error` message

Apache Configuration

- Main server configuration stored in
  `/etc/httpd/conf/httpd.conf`
  - controls general web server parameters, regular virtual hosts, and access
  - defines filenames and mime-types
- Module configuration files stored in
  `/etc/httpd/conf.d/*`
- DocumentRoot default
  `/var/www/html/`

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Configuration files are read from top to bottom. For some directives, order is important, so you should keep parsing order in mind

Apache Server Configuration

- Min and Max Spare Servers
- Log file configuration
- Host name lookup
- Modules
- Virtual Hosts
- user/group

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The number of spare server processes provided through the `prefork` MPM can be tightly controlled in Apache. Setting a maximum and minimum number of spare servers allows a balance between high request readiness and lower memory utilization.

Log files can consist of several different elements, as nearly everything passed to or from a web server can be logged. For a complete list of all log file elements, refer to:
`http://httpd.apache.org/docs-2.0/mod/mod_log_config.html`

Host name lookups are turned off by default. When turned on, the host names of clients will be logged, not just their IP. In general, host name lookups should remain off unless you are interested in knowing their results on a continual basis.

Apache module files live in `/etc/httpd/modules`. For detailed information on all included and generally available modules, refer to the official Apache modules registry at `http://modules.apache.org/`.

Apache supports both name and multiple IP based virtual hosts. It also enables Virtual Hosts to run associated CGI processes as a specific user and group.

Virtual Hosts

```
NameVirtualHost 192.168.0.100

<VirtualHost 192.168.0.100>
    ServerName   virt1.com
    DocumentRoot /path-to-document-root
</VirtualHost>
```

NAME

CONTENT

8

Rev RH253-RHEL4-1

The virtual host sections of `httpd.conf` usually have more directives than just `ServerName`, and `DocumentRoot`. Other likely directives include `ErrorLog`, `TransferLog`, and `ScriptAliases`. `ScriptAliases` defines which directories CGI programs are permitted to run from.

Once any virtual hosts are defined, then all content served from the server must be moved into a virtual host.

SSL virtual hosts are now configured in `/etc/httpd/conf.d/ssl.conf`.

Webdav.

Apache Namespace Configuration

- Specifying a directory for users' pages:
  `UserDir public_html`
- MIME types configuration:
  `AddType application/x-httpd-php .phtml` → TELL WEB BROWSER WHAT THE FILE TYPE IS.
  `AddType text/html .htm`
- Declaring index files for directories:
  `DirectoryIndex index.html default.htm`

redhat 9

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

The `UserDir` directive allows users to have a separate space for their own web documents. For example, if `/home/bob/public_html` existed and `UserDir` was set to `public_html`, then a request to `http://server/~bob/` would read documents from `/home/bob/public_html/`.

When a browser receives a file from a web server, it also receives the associated MIME type. The `AddType` directive allows a web server to intelligently supply the MIME type based on the file name extension.

The `DirectoryIndex` directive defines the names of files which are to be considered indices for a directory. If a URL specifies a directory rather than a named file, then the directory will be searched for one of the listed index files. Multiple index file names are supported, with preference given to those that appear first in the `DirectoryIndex` list.

Apache Access Configuration

- Apache provides directory- and file-level host-based access control
- Host specifications may include dot notation numerics, network/netmask, and dot notation hostnames and domains
- The **`Order`** statement provides control over "order", but not always in the way one might expect

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Directories and even files can have host-based access restrictions applied to them. Access is controlled through the use, either alone or in combination, of `allow`, `deny`, and `order` statements. The following lines, for example, access to `/var/www/html/internal` would be granted to only those hosts whose IPs reverse to a name in the example.com domain:

```
<Directory /var/www/html/internal>
        order allow,deny
        allow from .example.com
</Directory>
```

The order statement can take two arguments: order allow,deny and order deny,allow.

These arguments suggest that one can control the order in which Apache evaluates its access lists. It would be more accurate to understand these as statements of default behavior for hosts that are not included in the access control lists or when hosts that are included are specified in a contradictory fashion. If a host is included in a deny or allow list, but not both, then Apache will apply the access control. If a host is not in a deny or allow list, then the host will be handled by the default mechanism for the given order policy. If a host is included in both allow and deny lists, then it is handled by the default mechanism. It breaks down as follows:

`order allow,deny`	allows explicitly allowed clients, denies everyone else; clients matched by both allow and deny are denied
`order deny,allow`	denies explicitly denied clients, allows everyone else, clients matched by both allow and deny are allowed

For more information, see
`http://httpd.apache.org/docs-2.0/mod/mod_access.html`

Using .htaccess Files

- Change a directory's configuration:
  - add mime-type definitions
  - allow or deny certain hosts
- Setup user and password databases:
  - **AuthUserFile** directive
  - **htpasswd** command:
    htpasswd -c /etc/httpd/mypasswd bob

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`AllowOverride authconfig` is used to specify which and how much configuration can be overridden by directory specific `.htaccess` files. These files can be created by individual users to change certain Apache configurations that they wouldn't otherwise be able to change.

Essentially anything that can be configured in `httpd.conf` can be changed with the use of `.htaccess` files. These files have the same format as `httpd.conf`, and are placed in the directories in which the desired changes in configuration should occur.

One of the most common tasks performed in users' `.htaccess` files is adding authorization. Typically, a user will setup authorization for directories that hold sensitive information with a configuration such as:

```
# sample authentication .htaccess file

AuthName        "Bob's Secret Stuff"
AuthType        basic
AuthUserFile    /home/bob/mypasswd
require user    bob
```

With the `.htaccess` above, any access into the directory holding this file will require a user and password to proceed. In particular, only the user `bob` will be allowed, even if other users are listed in the `/home/bob/mypasswd` file. Note that you should not store your password file in a directory accessible from the web; it may be downloaded and cracked if access is gained.

To add entries to the password file, use the `htpasswd` command. The `-c` option will create the file if it doesn't previously exist.

CGI - Common Gateway Interface

- CGI programs are restricted to separate directories by ScriptAlias directive:
  `ScriptAlias /cgi-bin/ /<path>/cgi-bin/`
- Apache can greatly speed up CGI programs with loaded modules such as **`mod_perl`**

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

CGI, or Common Gateway Interface, is a defined method of passing information referring to dynamic web server content to web server programs. These programs often use the Perl programming language, which normally take slightly longer to start than compiled programs. These CGI programs greatly benefit from `mod_perl`. `mod_perl` keeps a copy of the Perl interpreter in memory, drastically reducing the startup time incurred from Perl CGIs.

Notable Apache Modules

- **mod_perl**
- **mod_php**
- **mod_speling** - spelling check.

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`mod_perl` embeds the Perl interpreter within a running `httpd` process to reduce process creation overload

`mod_php` enables accelerated parsing of pages containing PHP code

`mod_speling` will correct misspellings of URLs that users might have entered, by ignoring capitalization and by allowing up to one misspelling.

www ~~free~~ CA CERT ONLY

Apache Encrypted Web Server

- Apache and SSL: `https` (port 443)
  - `mod_ssl`
  - `/etc/httpd/conf.d/ssl.conf`
- Encryption Configuration:
  - certificate: `conf/ssl.crt/server.crt`
  - private key: `conf/ssl.key/server.key`
- Certificate/key generation:
  - `/usr/share/ssl/certs/Makefile`
  - self-signed cert: `make testcert`
  - certificate signature request: `make certreq`

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Apache can provide encrypted communications using the `mod_ssl` Apache module. To make use of encrypted communications, a client must request the `https` protocol, which is uses port 443. The configuration file for `mod_ssl` in RHEL is `/etc/httpd/conf.d/ssl.conf`.

Encryption is based on either the RSA or DSA algorithm. Private keys, self-signed certificates, or certificate signature requests can be generated using the `openssl` utility. RHEL provides a pre-configured `Makefile`, found in the `/usr/share/ssl/certs` directory, that can be used to aid in key generation.

A PEM-encoded private key should be saved in `/etc/httpd/conf/ssl.key/server.key`, and a PEM-encoded signed certificate should be stored in `/etc/httpd/conf/ssl.crt/server.crt`. If the private key is additionally encrypted, the Apache server will prompt for the appropriate password upon initialization and re-initialization.

Squid Web Proxy Cache

- Squid supports caching of FTP, HTTP, and other data streams
- Squid will forward SSL requests directly to origin servers or to one other proxy
- Squid includes advanced features including access control lists, cache hierarchies, and HTTP server acceleration

redhat 15

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

Squid is an internet object cache that can act as a proxy server for HTTP, FTP, and other requests. Clients request URLs from Squid, which then either serves cached copies of the URLs if they have been previously requested. URLs associated with dynamic content (CGI executables, server-parsed pages) get forwarded, rather than being served out of the cache.

Squid may be used as an HTTP accelerator. Just as Squid makes URL requests on behalf of a client when it acts as a proxy, squid *serves* URL requests on behalf of a server when acting as an accelerator. For example, a site whose URL is `http://www.notreallyhttpd.com` may actually have a squid process listening for requests on port 80 of `www.notreallyhttpd.com`. It will either serve the page itself out of cache, or else it will forward the request to the web server that handles that site.

Squid's configuration file is `/etc/squid/squid.conf`, and the *squid* RPM includes the Squid project's well-commented example configuration file. Key configuration elements include the port number on which it will listen for requests, whether it is inside a firewall, timeout settings, and ICP request port number. Squid uses port 3128 by default, but can easily be changed to 8080 if required.

Like Sendmail, the default Squid configuration only accepts connections on the system's loopback interface. To allow local network access, add an acl directive corresponding to the local network (for example, `acl mynet src 192.168.0.0/255.255.255.0`) and add an `http_access` directive corresponding to the new acl before the line `http_access deny all`.

The ICP (Internet Cache Protocol) request port number relates to Squid's ability to participate in cache hierarchies. A Squid cache can share the contents of its cache, or can request URLs from the cache of other squid processes if it belongs to a cache hierarchy. A Squid cache may act as a parent, sibling, or child of another Squid cache. The default maximum size of the cache is 100 MB, which would need to be increased (using the `cache_dir directive`) for most realistic situations.

Service Profile: Squid

- Type: SystemV-managed service
- Packages: *squid*
- Daemons: **`squid`**
- Script: **`squid`**
- Ports: 3128(squid), (configurable)
- Configuration: **`/etc/squid/*`**

redhat 16

End of Unit 5

- Address questions
- Preparation for Lab 5
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 17

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 5
The HTTP Service

Estimated Duration: 1 hour

Goal: To install a basic web server with CGI and virtual host capabilities

Throughout this lab, the hostnames and domain names that you use will be based upon the IP address of your machine. Any time the lab refers to a name that contains `XX`, *you should replace "X" with your station number (the last segment of your IP address). For example, if your station's IP address is* `192.168.0.3`, *you would replace references to* `stationX.domainX.com` *with* `station3.domain3.com`.

Disable packet filtering. Before commencing, ensure that packet filtering is not active. The default installation will have a file called `/etc/sysconfig/iptables`, which configures the `iptables` facility. Run "`chkconfig iptables off`". To remove any rules that may be in place, run "`service iptables stop`"

Sequence 1: Server installation and basic configuration

Scenario/Story:

Your organization needs a web server in one hour, complete with CGI capability and the ability to serve different content from different virtual hosts.

Tasks:

1. The *httpd* package is required (see Appendix item 1). Use `chkconfig` to enable the service

2. Start `httpd` using the default configuration: `service httpd start`

3. Verify the `DocumentRoot` directive in `/etc/httpd/conf/httpd.conf` reads as below:

   ```
   DocumentRoot /var/www/html
   ```

4. Open a web browser and set the URL to

   `http://stationX.example.com`.

   If your browser is working, you will see the default server index page. Note that this page is not actually stored as an HTML file, but will be generated by server for those directories that do not have a default index.html file

5. Create a new directory hierarchy and some new content:

```
mkdir -p /var/www/virtual/wwwX.example.com/html
cd /var/www/virtual/wwwX.example.com/html
cat > index.html <<EOF
<b>wwwX.example.com</b>
EOF
```

(This creates a one-line HTML page.)

6. Create a virtual host by adding the following lines to the bottom of `/etc/httpd/conf/httpd.conf`:

```
NameVirtualHost 192.168.0.X

<VirtualHost 192.168.0.X>
   ServerName       wwwX.example.com
   ServerAdmin root@stationX.example.com
   DocumentRoot  /var/www/virtual/wwwX.example.com/html
   ErrorLog    logs/wwwX.example.com-error_log
   CustomLog   logs/wwwX.example.com-access_log combined
   <Directory  /var/www/virtual/wwwX.example.com/html>
      Options  Indexes Includes
   </Directory>
</VirtualHost>
```

7. Ensure that your DNS system resolves your virtual host domain name.

```
dig wwwX.example.com
```

8. Reload `httpd`: `service httpd reload`

9. In your browser, change the URL to point to the new virtual host:

`http://wwwX.example.com`.

Do you see the content of your customized page?

Sequence 2: Using CGI

Tasks:

1 Add the following **one line** to the `<VirtualHost>` block defined in Sequence 1:

```
ScriptAlias /cgi-bin/
     /var/www/virtual/wwwX.example.com/cgi-bin/
```

This is only one line, and should not be line wrapped in `httpd.conf`. *Ensure that a SPACE exists between the two elements listed above.*

2. Create the directory, then create a file inside it named `test.sh`, with the following contents:

`/var/www/virtual/wwwX.example.com/cgi-bin/test.sh`

```
#!/bin/bash
echo Content-Type: text/html
echo

echo "<pre>"
echo My username is:
whoami
echo
echo My id is:
id
echo
echo My shell settings are:
set
echo
echo My environmental variables are:
env
echo
echo Here is /etc/passwd:
cat /etc/passwd
echo "</pre>"
```

3. Try to execute this CGI script by pointing your web browser to

`http://wwwX.example.com/cgi-bin/test.sh`

Why didn't the script execute? Check the log files in `/var/log/httpd/` for information to help you answer why. (Did you remember to restart or reload server?).

4 Make the script read-executable for user, group and other:

```
chmod 555 test.sh
```

Does the script execute now?

In the `test.sh` script, the `id` command is executed. When you test this and the script otherwise runs correctly, you will note that the `id` command fails. Can you figure out why?

Remove the `id` command from the `test.sh` script and confirm that the `test.sh` script can run without errors

Challenge 1: Securing access to your web site documents

Tasks:

1. Create a file named `.htaccess` in the `wwwX.example.com` document root with the following contents:

/var/www/virtual/wwwX.example.com/html/.htaccess

```
AuthName     "restricted stuff"
AuthType     Basic
AuthUserFile /etc/httpd/conf/wwwXX.htpasswd
require      valid-user
```

2. Create your domain's password file. The file must be readable by the group `apache`

```
htpasswd -mc /etc/httpd/conf/wwwX.htpasswd user_name
chgrp apache /etc/httpd/conf/wwwX.htpasswd
chmod g+r /etc/httpd/conf/wwwX.htpasswd
```

Set the SELinux context of the new `virtual` directory tree so that SELinux will permit the web server to access it:

```
chcon -R -reference=/var/www/html /var/www/virtual
```

What would happen if you attempted to access pages under `virtual` without setting this context? Try it and investigate the errors.

3. Access the `http://wwwX.example.com` web page. Are you prompted for a user name and password? Check the server logs in `/var/log/httpd/` for clues.

4. Add the following line to server's configuration file `httpd.conf`, inside of a `<Directory>` block for the `wwwX.example.com` virtual host:

```
AllowOverride AuthConfig
```

5. Try to access the `http://wwwX.example.com` web page again. Are you prompted for a user name and password now? If so, are you able to gain access to the page with the user name and password you created in Step 2?

Sequence 3: Basic squid configuration

Tasks:

1. Install `squid` on your system, if neccessary(see Appendix item 1):

```
rpm -Uvh ftp://server1.example.com/pub/RedHat/RPMS/squid*
```

2. Start the service (`service squid start`), then configure your web browser to use your proxy `localhost` with the port set to `3128`.

3. Try accessing a web page somewhere. If the classroom does not have Internet access, try `http://server1.example.com`, which should return the server test page.

4. Now have a neighbor configure his or her web browser to use your proxy. This should not work

   The page that `squid` returns, and the bottom of `/var/log/squid/access.log` indicate why.

5. Open `/etc/squid/squid.conf` in your preferred text browser. As you can see, it is mostly comments and documentation. You will also note that `squid` is extremely configurable and tunable. For this lab, we will configure a basic setup that will be adequate for many settings.

6. Search for the *second* occurance of `Recommended minimum configuration` in the file. This will take you to the default access control lists, or acls. Add an acl for the local network below the acl `CONNECT method CONNECT` line:

   ```
   acl    example    src    192.168.0.0/24
   ```

   You can now refer to this network as `example` elsewhere in the configuration file. `src` means that the IP specified is the source IP(s) for this `acl`.

7. Search further down in the file for `INSERT YOUR OWN RULE(S) HERE`. Add a line *above* the `localhost` access rule, as follows:

   ```
   http_access    allow    example
   ```

   Restart `squid`. Your neighbor should now be able to access your cache

8. Some URLs are best avoided completely. Return back the acls section, and add the following lines beneath the line you added earlier (use `example.com` if you do not have Internet access in your classroom):

   ```
   acl    otherguys    dstdomain    .yahoo.com
   acl    otherguys    dstdomain    .hotmail.com
   ```

   There are a couple of things to mention here. First, note that the additive property of acls. Both of the domains are added to the acl. Second, note the `dstdomain` acl type, which specifies that this definition concerns destination domains. Third, note the use of dot notation in specifying the domain name. Make sure to include the leading dot

9. Add a rule to deny access to these problematic domains. Return to where you added the allow rule for example, and below it add the following:

   ```
   http_access    deny    otherguys
   ```

   Restart `squid` again, then check one or more of the web sites associated with those domains. Unfortunately, you find that access is not denied.

10. Open the configuration file again, and move the deny rule you added so that it is before the allow rule for example. Order matters, so by having the allow rule for example before the deny rule for the `otherguys` destinations, access was allowed and the deny rule never took effect. After moving the rule, restart `squid` once more. This time it should deny access to any site within the prohibited domains.

Challenge Break:

1. Disable the system's "firewall," if neccessary.

   `service iptables stop`

2. Run the following command, following the instruction displayed:

   `tsservices 1`

3. This command will set up the problem and will explain the goal. Refer to the file `/etc/ts` to review the goal. Refer to Lab 1, if you need help.

Answers to Questions

Sequence 2, Task 4:

In the `test.sh` file, the id command is run. When you test this and the script otherwise runs correctly, you will note that the id command fails. Can you figure out why?

> Answer: SELinux prevents the web server from accessing the id command. Note the SELinux error messages in the `/var/log/messages` file.

Challenge 1, Task 2:

Set the SELinux context of the new `virtual` directory tree so that SELinux will permit the web server to access it:

```
chcon -R --reference=/var/www/html /var/www/virtual
```

What would happen if you attempted to access pages under virtual without setting this context? Try it and investigate the errors.

> Answer: Without modifying the context, SELinux would prevent the web server from accessing any files or directories under the `/var/www/virtual` directory.

UNIT 6

Security Concerns and Policy

redhat

1

Rev RH253-RHEL4-1

Objectives

- Be able to define security
- Understand Security Components
- Be able to develop a Security Policy

Agenda

- Define Security
- Where are the Vulnerabilities?
- Developing a Security Policy
  - System Activity
  - Human Activity
- Response Strategies

redhat 3

Definition of Security

- Types of security
  - Network(external)
  - Local(internal)
  - Physical

4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Many security tutorials focus on protecting a network against attacks from the public Internet. However, there are other forms of security to consider

Many of the most damaging attacks come not from the outside, but from employees and contractors with access to a company's internal network. Since more company-confidential information is likely to be accessible from the internal network than the Internet, overlooking internal security can be a very costly mistake

Physical security, likewise, is very important. Physical security is the limiting of physical access to systems containing sensitive information. Almost any software-based security measure can be circumvented if someone has physical access to a machine. You should realize that someone with physical access to your machine could boot the machine into single user mode very easily if your machine does not have a boot loader password. However, even if you have a boot loader password, someone with physical access could boot your machine from installation media and access it from Rescue Mode. Unfortunately, even a BIOS password to prevent people from changing the boot order of the machine may not be enough. Potentially, someone could remove the disk drive from your machine and access the data on it from another machine. Luckily, physical security is the most straight forward to enforce. By locking the machine in a tamper proof case, or locating it in a locked room one could limit the people that have physical access to the machine.

Attacks from the Network

- Exploits and "script kiddie" attacks
- Denial of Service(DoS) attacks
- Distributed Denial of Service(DDoS) attacks
- Hijacking, "Man-in-the-Middle" attacks
- Trojans and "Root Kits"

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

When a vulnerability in software is found, small programs that exploit the vulnerability are often released shortly thereafter. Sometimes these "exploits" are written as proof-of-concept demonstrations and sometimes they are written expressly for use in compromising vulnerable systems. Often the people using exploits to break into systems do not really understand what they are doing, they are just running code that someone else wrote. The derisive term "script kiddie" is often used to describe such people.

The Denial of Service (DoS) attack does not necessarily involve exploiting a particular vulnerability, but instead attempts to exhaust or otherwise deprive a system of necessary resources like CPU time, memory or network "bandwidth". A distributed denial of service (DDoS) attack involves a coordinated attempt by multiple systems launching DoS attacks on a single system at once. The attacking systems are often compromised machines running "zombie" daemons remote controlled by someone without their administrator's knowledge.

Hijacking attacks involve one machine attempting to impersonate another machine on the network. For example, a NIS client that determines its NIS server by broadcast can be tricked by a fake NIS server that responds faster than the real one. The impostor then enjoys easy access to user names and passwords sent to it as well as the ability to serve out misleading host resolution information. A variation on hijacking is a "man in the middle" attack, wherein a machine manipulates the keys being used in an encrypted connection to trick each side of the connection into thinking it (the "middle" machine) is the other side. The middle machine can then read and even manipulate the content of the communication.

Upon gaining root level access to a system, attackers will sometimes install a "root kit". A "root kit" is a tool or set of tools that removes evidence of the intrusion (sometimes even securing the hole that was used to get in). It may install "back doors" so that the intruder can get in later and possibly modified("trojan")versions of common system utilities like `ls`, `ps`, `rpm` and `md5sum`, which do not report the "back doors". Some "root kits" even install modified kernel modules or otherwise alter the kernel's system calls without altering the system binaries. This makes the "root kit" even harder to detect.

Principles of Security

- No such thing as 100% protection
- Myth: "We're too small to be at risk"
- Every service is a liability
- Processes running as root are a liability

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Some intrusion attempts are aimed at obtaining specific information or at attacking a specific person or organization. However, many -probably the majority-- of attacks are the result of random scans that search a network for vulnerable systems In some cases the results of such scans are catalogued for future reference so that when new vulnerabilities are discovered, potential targets can be located without re-scanning

cybeCERT

Security Practices

- Do not run services you do not need, lock down services you do need
- For processes that run as root:
  - "Do I need to be running this?"
  - "Does it need to be running as root?"
  - "Have I applied all relevant security updates"
- Regularly scan for vulnerable files
- Compromising a user often leads to root
  - Educate users!

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Services that are unnecessary can be turned off with '`service <service name> stop`' and prevented from restarting at boot time with '`chkconfig <service name> off`'.

It is a good idea to periodically check which services are running on your system using `netstat` for the "internal" (i.e. independent of host-based access controls, firewalls, etc) and `nmap` run from a system outside of your network for the "external" perspective. These utilities are discussed next. Remember that local tools such as `netstat` may have been trojan'd by an attacker so as to not show you things like listening "back door daemons". Thus, doing periodic port scans from an external host can yield valuable information.

A system administrator is responsible for keeping all active services running in a secure manner. This includes both keeping services updated against bugs and security holes and running each service with as little access to the rest of the system as possible.

User accounts must also be protected. Weak, easily determined passwords can give attackers access to your system. Such access can then be leveraged to either allow intruders deeper access into the network or to begin launching attacks against other networks while disguised as one of your users. Teaching users the importance of strong passwords, not writing down their passwords, not running untrusted binaries and not setting files to be world-writable can save future headaches.

Diagnostic Utilities

- Port scanners (nmap)
  - Show what services are available on a system
- Packet sniffers (tcpdump, ethereal)
  - Stores and analyzes all network traffic visible to the "sniffing" system
  - Availability is also a liability

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Port scanners can list everything from what operating system a host is running to which daemons (including versions) are offering services on that machine This is as useful to you, the system administrator, in identifying potential points of entry as it is to an attacker Whoever finds the vulnerable service first "wins:" that is, either you, or your adversary.

Likewise, packet sniffers can be useful for troubleshooting as well as detecting potential attacks. Some sniffing programs serve as intrusion detection systems (IDS) that scan network traffic in real time looking for know-malicious types of traffic. Two utilities included in the RHEL distribution are `tcpdump` and `ethereal` These tools may also be viewed as a liability and are often installed on the administrator's system(s) only.

nmap -sV -O server1

ethereal

ettercap

Which Services Are Running?

- Use **`netstat -taupe`** for a list of:
  - active network servers
  - established connections

redhat 9

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

One of the most used diagnostic utilities is `netstat`. It can list active network connections, routing tables, interface statistics, and other vital network information. With `netstat -taupe` you can get extended information about which network services are available, regardless if they were started automatically by script, via `xinetd` or manually.

important `netstat` options:

`-t`	list TCP connections
`-u`	list UDP connections
`-a`	show listening and connected sockets
`-p`	show PID and name of the program
`-e`	show extended information, use twice for even more information

other `netstat` options include:

`-s`	statistics on the network stack (e.g. number of delivered packets)
`-r`	kernel routing table

`netstat` only shows programs using network sockets (and UNIX-domain types, with `-a`).

Remote Service Detection

- **nmap** scans for active services
  - Advanced scanning options available
  - Offers remote OS detection
  - Scans on small or large subnets
- Used by intruders for the same purpose
- Do not use without written permission of the scanned system's admin!
- Graphical front-end available (**nmapfe**)

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The utility nmap is a full-featured port scanner that can be used to detect which ports are open on a remote system. It supports a large number of scanning techniques that are able to detect which services are available on a networked host

Special analysis of the TCP stack of the target system (sequence numbers, response to errors, etc.) allows nmap to detect which operating system is being used.

With cloaking technologies like half-open scans and decoys ("scans" from spoofed addresses), nmap port scans are hard to detect and therefore nmap is one of the most commonly-used tools by intruders

There are many options. Below is an nmap command specifying just a few of them. It will perform a TCP syn scan (-sS), UDP scan, (-sU) and rpc/portmap scan (-sR), with operating system and service version detection (-A) on station1. It will print verbose diagnostic information (-v) and will not attempt to ping the system before scanning (-P0):

```
nmap -sS -sU -sR -P0 -A -v station1
```

Port scanners and penetration testing tools should be used with discretion. Make sure that you have permission to probe a machine with these tools before employing them.

Many scanning programs will start by pinging an IP address. If the host is deemed to be down, because it is not responding to pings, the scan will skip the host. Therefore, you can eliminate a large amount of scans on your machine by turning off responses to ping. This can be done in the kernel by adding:

```
net.ipv4.icmp_echo_ignore_all = 0
```

to /etc/sysctl.conf.

Isolate Vulnerabilities

- Isolate processes
  - Process runs as own user (RHEL default)
  - System users should only have access to service's files and nothing else
- Isolate networks
  - Implement a "firewall"
  - Avoid services that authenticate without encryption
    - telnet, pop, imap, authenticated ftp
    - alternatives: ssh, apop, imaps, sftp, anonymous ftp
- Keep systems 'up2date'

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

It was once considered good security to have all services running as a 'non-privileged' user like `nobody` or `daemon`. But the service's "user" must have access to all of the files that service needs. As a result, the more services you run as `nobody`, the more privileged the `nobody` user actually is. Red Hat Enterprise Linux runs each service as its own user. For example, the `httpd` web daemon runs as the user `apache`. This user should have access to the `/var/www/*` files, but as little else as possible. That way, if an `httpd` process is compromised, the extent to which it can affect the rest of the system is limited.

Such illicit access can be further limited by 'chrooting' the daemon processes. Chrooting a process tricks it into thinking that a specified directory on your system is actually the root filesystem directory, '/'. The process lives entirely in the chrooted directory. As a result, even if a process is compromised in such a way that the attacker is given extra privileges, the process's "perspective" is limited so that it cannot "see", much less affect, the rest of the system. Chroot "jails" can be broken out of by root-level processes, but it is one more hurdle the prospective intruder must anticipate and overcome.

In keeping with the attitude of trusting internal network clients as *little* as external clients, a "de-militarized zone", or DMZ firewall layout has all workstations on one network, all servers on another network and both connected to each other and the Internet through a common firewall. Thus, the administrator is afforded as much control over workstation/server communications as over Internet/server communications.

Many old, insecure services enjoy a great deal more popularity than they probably should on today's Internet. The services `telnet`, `pop`, `imap` and `ftp` all transmit authentication information unencrypted over the network. It is trivially easy for someone with root-level access to any system between the client and the server to intercept this information. Secure alternatives exist for each of these services. In fact, `ssh` can be used to tunnel any tcp-based service in a secure fashion using the `-L` option.

When run regularly, the `up2date` utility can be used to keep systems current with the Red Hat errata. Check RHN for updates regularly or schedule a cron job that runs `up2date -u`, which installs all relevant updates regularly. `up2date` sends a daily log email message to the root account, cataloging all actions it has taken.

Security Policy: the System

- Managing system activities
- Regular system monitoring
  - Log to an external server in case of compromise
  - Monitor logs with logwatch
  - Monitor bandwidth usage inbound and outbound
- Regular backups of system data

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Even after doing everything possible to prevent an intrusion, it is always possible that an attacker will succeed in getting into a system. The key to limiting the attacker's success is to identify compromises quickly and be ready with a recovery and response plan. Have a security policy that defines 'warning signals' such as services mysteriously dying, unexpected spikes in bandwidth usage and modifications to important system files and know who is expected to do what in response to them

Always have a recent backup on-hand for quick recovery to a known-good state.

Security Policy: the People

- Managing human activities
  - includes Security Policy maintenance
- Who is in charge of what?
- Who makes final decision about false alarms?
- When is law-enforcement notified?

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Mark Cox (head of Red Hat's security response team) and Michael Tiemann (CTO of Red Hat) developed a white paper on security best practices at the Best Practices section of the Red Hat's website. Regarding security policy they have this to say:

"The first and foremost best security practice is to have a documented security policy. Regardless of how complete or incomplete this policy may be, the policy is the objective reference against which one can measure "what did we say we would do, what did we do, and what do we need to do in the future to do better?" If the policy is found to have had a hole, the policy can be amended and security improved. However, if there is no policy, then every incident becomes a fire drill: can it ever happen again? Will it ever happen again? What if it does happen again? Trying to make those decisions in the absence of a documented security policy leads most often to either spending a lot of money on a solution that doesn't really fix the root problem, or spending nothing, learning nothing, and hoping it doesn't happen again."

Response Strategies

- Assume suspected system is untrustworthy
  - Do not run programs from the suspected system
  - Boot from trusted media to verify breach
  - Analyze logs of remote logger and "local" logs
  - Check file integrity against read-only backup of rpm database
- Make an image of the machine for further analysis/evidence-gathering
- Wipe the machine, re-install and restore from backup

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Remember, one of the first things that an intruder will do is install a "*root kit*" that replaces the binaries she suspects you will use with compromised versions that leave out important information. Boot the suspected system from read-only, trusted media(like your Red Hat installation CD) in rescue mode. Having booted from a trusted kernel and using only the utilities on the CD, you can generally rely on the information you gather

One example of valuable forensic information obtained from within a rescue environment is determining whether or not important system binaries have been altered. The md5 sum (`md5sum <filename>`) of every file on the system provided by a Red Hat RPM is available in the system's RPM database as well as on Red Hat's RHN website. If regular backups of the RPM database files(`/var/lib/rpm/*`) are made to trusted media, then the following example test of the `/bin/ps` binary (part of the *procps* RPM) is a reliable "sanity check." Note: `/bin/ps` is often chosen for replacement by intruders, and replaced with a variant which "hides" their rogue processes.

```
rpm -V -root=/mnt/sysimage -define '_dbpath /path/to/backup' procps
```

This will compare the size, md5sum, ownership, permissions and other attributes of `/bin/ps` and the rest of the *procps* utilities as they exist on the hard disk (`/mnt/sysimage`) to their recorded values in the backed-up RPM database. If discrepancies are found, especially with the md5sum, it may be the result of a trojan'd binary. Note that in rescue mode the `-dbpath` option is not available, so it is important to use `-define` instead.

Attempts by an intruder to remove evidence from the system logs can be thwarted by having your system log to a remote log host as well as to the local `/var/log/` directory. Of course, it is possible for the log host to be compromised as well, but we have double the amount of work an intruder must do to cover his or her tracks

Determining how someone got into a system and how much damage was done in the process can be very time-consuming. For production systems minimizing downtime is often of paramount importance. Creating an "image," or exact copy of the suspect system's disks permits thorough investigation, while facilitating rapid re-deployment of a system's resources. Computer-forensics investigators will also take snapshots of what is in the system's memory as well

Additional Resources

- Security Education
  - Red Hat Security Guide (on Documentation CD and at redhat.com's docs section)
- Keeping up with vulnerabilities
  - Red Hat Network
  - Red Hat Errata
  - Bugtraq mailing list
- Keeping track of "the other side"

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The Red Hat Security Guide provides a good introduction to securing Red Hat systems. There are different editions for each Red Hat distribution, so look on your Documentation CD or look in the appropriate documentation section of the Red Hat website.

The Linux Security and Security Focus websites both offer security-oriented news, tutorials and vulnerability alerts. Securityfocus is also the home of the bugtraq mailing list archives. Bugtraq is usually the first place that new vulnerabilities are announced. Errata RPMs incorporating security fixes are announced and made available at the Red Hat Network's errata website and through the up2date utility (c.f. http://rhn.redhat com, http://rhn redhat.com/errata, http://www linuxsecurity com, http://www.securityfocus com).

The Honeynet Project is run by a group of security experts who intentionally set up machines with known vulnerabilities, hoping that someone will break into them. These "honey pots" are loaded with monitoring software that records everything the attacker does in an attempt to learn about the latest methods being employed by crackers. The Honeynet group publish the results of their research, develop tools to aid in security investigation and make available scans of compromised systems, allowing people to practice tracking down real-life security breaches.

As the Honeynet project illustrates, it is always useful to have an idea about what the "other side" is up to. There are a number of "black hat" web sites that publish papers on the latest ideas with regard to how to compromise and maintain control of a system. Phrack magazine, in particular, tends to have in-depth, technical coverage of the "hows" behind the latest security compromises (c.f. http://www honeynet.org, http://www phrack.com)

ASTALA VISTA. box.SK
OSVDB. org

End of Unit 6

- Address questions
- Preparation for Lab 6
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 16

Lab 6
Security Concerns and Policy

Estimated Duration: 20 minutes

Goal: To build skills with diagnostic utilites

Sequence 1: Service Detection

Scenario/Story:

You have done your best to understand which services you must offer, and to whom, and now must verify this configuration. This is a mere exercise in a long and on-going effort to manage your system within the definitions of your Security Policy. If the `nmap` program is not currently on your system you will need to install the *nmap* rpm.

If you are located in an Internet-enabled classroom, please do not attempt to use `nmap` to scan machines outside the example.com domain or outside the 192.168.0/24 subnet unless instructed to do so. Thank you for your cooperation.

Tasks:

1. Work with a lab partner, and perform a port scan of one system from the other. For puposes of this lab, instructions will refer to `stationX` and `stationY`, where `stationX` is the "local" system and `stationY`, the "remote."

```
[stationX]# netstat -tpane | grep tcp | wc -l
[stationY]# nmap -sS -P0 <stationX> | grep open | wc -l
```

11
8, 17, 4 (5, 3, 7) (9)
11

What are the results? Are they the same? Should they be?

22

```
[stationX]# netstat -tupane | grep 0.0 | wc -l
[stationY]# nmap -sSU -P0 <stationX> | grep open | wc -l
```

Which ports are open and listening? Through which ports are you connecting to other systems? Remove the `wc -l` from the commands above to gather more data. Which kinds of commands would you use in place of `wc -l` to make best use of these investigations?

2. From each host, explore your subnet:

```
ping -c 6 -b 192.168.0.255 | grep '^64' | sort -u -k 3,4
```

What were the results? Were you surprised? Are these the only systems "out there?" Why not?

Now run `chkconfig` to audit your system. What is *planned* to run, and now running?

```
chkconfig --list | grep $(runlevel | cut -d" " -f2):on
```

Again, is your machine configured to run services you don't know about? Use `chkconfig` to turn off services that you don't want running. Reboot your machine, and repeat step #1 above. What is `netstat` reporting about the ports associated with the services you disabled?

Deliverables:

1 A system audit reveals only neccessary services are available

UNIT 7

Authentication Services

redhat 1

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Objectives

- Understand the basics of authentication
- Understand the roles of NSS and PAM
- Use NIS to centrally manage user information and authentication through NSS and PAM

redhat

2

Rev RH253-RHEL4-1

Agenda

- User Information and NSS
- Authentication and PAM
- Network Information Service (NIS)
  - Configuring NIS master servers, slave servers, and clients

User Authentication

- Two types of information must always be provided for each user account
  - **Account information:** UID number, default shell, home directory, group memberships, and so on
  - **Authentication:** a way to tell that the password provided on login for an account is correct

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

There are two basic types of information that must always be available for user authentication to work properly. First, *account information* is needed which provides basic information about the account: its UID number, primary GID number, default shell, and so on. Second, *authentication* information is needed which can be used to prove that a particular password is the correct password to grant access to a particular account. For local users, account information is stored in `/etc/passwd`, while authentication information is stored in `/etc/shadow`. For centrally-managed network users, both types of information may be stored in one or more network services.

In this unit we will look at mechanisms which control how both types of information are provided to the system.

Account Information

- Name services accessed through library functions map names to information
- Originally, name service was provided only by local files like `/etc/passwd`
- Adding support for new name services (such as NIS) required rewriting `libc`

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Basic information about the characteristics of a user account must be provided at login time This information includes the UID number of the account, the account's primary GID number, the account's home directory, its default shell, and so on Originally, this information was stored as individual lines in local files such as `/etc/passwd`

Applications normally do not look directly in `/etc/passwd` for this information. Instead, they use standard library functions provided by the system's standard C library, `libc`, to look it up For example, the getpwnam() function looks up the account information normally stored in `/etc/passwd` as a single line, by user name Adding support for new sources of information ("name services") such as NIS or LDAP involves making appropriate changes to the C library, and all applications using these library functions can immediately take advantage of the change

Name Service Switch

- NSS allows new name services to be added without rewriting `libc`
  - Uses `/lib/libnss_service.so` files
- `/etc/nsswitch.conf` controls which name services to check in what order
  - `passwd: files nis ldap`

redhat 6

Rev RH253-RHEL4-1

Name Service Switch, or **NSS**, is the mechanism which allows configuration and extension of the name services supported by libc. Auxiliary name service libraries are installed as `/lib/libnss_`*service*`.so` files; for example, `/lib/libnss_files.so` provides support for retrieving name service information from the standard local files

The `/etc/nsswitch.conf` file controls which name services will be used for what name services in which order to look up information. Each line represents an administrative database of names ("name service") used by libc functions. The first item in each line specifies which name service is being configured, followed by a colon:

aliases, ethers, group, hosts, netgroup, networks, passwd, protocols, rpc, services, shadow

with the database name usually matching the name of the file in `/etc` usually used to store the information. The remaining items on the line consist of a space-delimited list of name services to check, in order, for the information. The first match that is found will take effect. For example, the line

```
passwd: files nis ldap
```

specifies that for information typically stored in `/etc/passwd`, look first in local files, then use the NIS server, then the LDAP server

getent

- `getent` *`database`*
  - Lists all objects stored in the specified database
  - `getent services`
- `getent` *`database name`*
  - Looks up the information stored in the specified database for a particular name
  - `getent passwd smith`

gfouf Legal

redhat 7

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

The `getent` utility is invaluable for checking name service operation. `getent` can perform lookups of names in any of the standard C library databases, and it can also provide a dump of the appropriate database to standard output. This can be useful in determining answers to questions like "Are there two users with the same UID on the system?" or "Is there an entry for this user in NIS as well as in the local files?"

For example, the command `getent passwd root` might produce the output:

```
root:x:0:0:root:/root:/bin/bash
```

Authentication

- Applications traditionally authenticated passwords by using `libc` functions
  - Hashes password provided on login
  - Compare to hashed password in NSS
  - If the hashes match, authentication passes
- Applications had to be rewritten to change how they authenticated users

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Originally, applications used NSS to store password hashes as well as general account information. On a login attempt, the application would encrypt the password entered, use `getpwnam()` to look up the `passwd` file information for that account in NSS, and compare the encrypted passwords. If the password hashes matched, the correct password was entered and the user is authenticated. This same basic approach is still used for local user authentication.

However, other mechanisms to authenticate users were developed. If modern applications still called *libc* functions directly to authenticate users, then each application would have to be rewritten to support password authentication mechanisms that do not store hashes in NSS.

md5sum

PAM

- Pluggable Authentication Modules
- Application calls `libpam` functions to authenticate and authorize users
- `libpam` handles checks based on the application's PAM configuration file
  - May include NSS checks through `libc`
- Shared, dynamically configurable code

redhat 9

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The **Pluggable Authentication Modules** system, or **PAM**, provides a generic way for applications to implement support for authentication and authorization. A PAM-enabled application calls `libpam` functions to perform all authentication tests for it. These tests may include traditional NSS-based authentication. These functions each return an overall pass or fail result which the application then act upon appropriately.

New modules can be added or removed by the system administrator to adjust authentication methods supported. Changes to PAM take effect the next time the program is run, as soon as the configuration files are saved. Implementations of authentication mechanisms can be performed once and shared by many programs, and the smaller, shared code base is more easily audited for problems.

System-Auth-config

PAM Operation

- `/lib/security` PAM modules
  - Each module performs a pass or fail test
  - Files in `/etc/security` may affect how some modules perform their tests
- `/etc/pam.d` PAM configuration
  - Service files determine how and when modules are used by particular programs

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

PAM modules are stored in `/lib/security`. Each PAM module implements some test which may pass or fail Some PAM modules may use supplementary configuration files in `/etc/security` that control how they perform their tests Others may use traditional files like /etc/securetty or may only be configurable through options

The `/etc/pam.d` directory contains *service files*. Each application that is PAM-aware has a service file which controls what modules it uses and how those modules affect the overall authentication results. Generally the service file that is used has the same name as the PAM-aware application. If the service file for an application is missing, `/etc/pam.d/other` is used as a default.

column 1

/etc/pam.d Files: Tests

- Tests are organized into four groups:
  - **auth** authenticates that the user *is* the user
  - **account** authorizes the account may be used
  - **password** controls password changes
  - **session** opens, closes, and logs the session
- Each group is called as needed and provides a separate result to the service

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

PAM actually organizes tests into four management groups which are checked independently by different `libpam` library functions The `auth` management group is used by PAM functions which authenticate users. The `account` management group is used to verify that an account is valid at this time and that passwords have not expired The `password` management group is used to control password changes. The `session` management group is called by PAM at the start and at the end of a session

Each management group returns independent results. A particular PAM function will only call PAM tests from one management group, but an application will generally perform tests from all management groups in turn. For example, on login it would be normal to perform `auth`, `account`, and `session` tests If the password of an account had expired, it would not be unusual for the login program to prompt the user to change the password and perform `password` checks as well.

An individual PAM module may support all four management groups, but it does not need to Some modules may be useful only for `auth` but not for `session`, or vice versa

column 2

/etc/pam.d/ Files: Control Values

- Control values determine how each test affects group's overall result
  - **required** must pass, keep testing even if fails
  - **requisite** as **required**, except stop testing on fail
  - **sufficient** if passing so far, return success now if fails, ignore test and keep checking
  - **optional** whether test passes or fails is irrelevant

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The second column of an `/etc/pam.d` service file lists the control value that determines how the result of a module's test will affect the overall result of the management group. Modules are checked in order in the configuration file. A **required** module must pass for the overall result to pass. If it fails, the remaining modules are checked anyway to disguise why the failure happened from a potential attacker. A **requisite** module also must pass, but if it fails the failure is returned to the application immediately without checking other modules. This can be useful to stop a user from entering a password over an insecure terminal. If a **sufficient** module passes and the overall result so far is passing, then the pass is returned to the application immediately without checking other modules. If it fails, then the result is ignored. Whether an **optional** module passes or not has no effect on the return value.

Sometimes, a module will return *ignore* rather than pass or fail. This indicates that this test should be ignored when PAM figures out the overall return code for this management group.

An advanced control value syntax also exists for more complex scenarios. For instance

```
[default=bad success=ok ignore=ignore user_unknown=ignore]
```

would work much like `required` except that errors due to the user not existing would be ignored rather than treated like a fail result. The advanced syntax is best avoided if possible.

Example /etc/pam.d/ File

auth	requisite	pam_securetty.so
auth	sufficient	pam_unix.so likeauth
auth	required	pam_deny.so
account	required	pam_unix.so
password	required	pam_cracklib.so retry=3
password	sufficient	pam_unix.so use_authtok
password	required	pam_deny.so
session	required	pam_unix.so
session	required	pam_limits.so
session	optional	pam_console.so

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

This example might be appropriate for the `login` program. The user is prompted for the account's password by the `pam_unix.so` auth module. Notice that if the `pam_securetty.so` auth module fails, the user will not be prompted for the password.

Two special PAM modules are `pam_deny.so` (which always fails), and `pam_permit.so` (which always passes). These modules can be used to structure more complex sets of tests.

pam_stack

- Special module that bases result on the tests in another /etc/pam.d service file
- system-auth is widely used
  - Contains standard authentication tests
  - Shared by many applications on the system
  - Allows easy, consistent management of standard system authentication

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

This module gets its result by performing the tests listed in a separate /etc/pam.d service file The overall results of those tests will be the result of this module This is useful to share the same basic authentication tests between multiple applications.

Many applications use this in conjunction with the /etc/pam.d/system-auth file This file contains standard authentication tests for system users, and changes under the "Authentication" tab in the system-config-authentication utility change that file. This allows easy and consistent management of standard authentication tests For example, in /etc/pam.d/sshd,

```
auth required pam_stack.so service=system-auth
```

might cause the following three lines to be tested from /etc/pam.d/system-auth:

```
auth required pam_env.so
auth sufficient pam_unix.so
auth required pam_deny.so
```

pam_unix

SO

- Module for NSS-based authentication
  - **auth** gets hashed password from NSS and compares it to hash of entered password
  - **account** checks for password expiration
  - **password** handles password changes to local files or NIS
  - **session** records login and logout to logs

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

One of the most critical PAM modules is *pam_unix*, which implements traditional NSS-based authentication The *pam_unix* module performs traditional authentication through the libc library functions on behalf of the application, and it can be used in all four functional groups:

As an **auth** module, it checks to see if the password entered on login is correct through NSS library functions and password hash comparisons.

As an **account** module, it uses NSS information to check if the password has expired or the account is locked.

As a **password** module, it updates password information The `md5` option is set by default, indicating that passwords should be encrypted as MD5 hashes The `shadow` option is also set by default, indicating that password aging should be turned on and that password hashes should be hidden from non-root users in `/etc/shadow`. The `nis` option can be added to allow updates of NIS passwords through communicating with a remote `rpc.yppasswdd` service
As a **session** module, it logs through `syslogd` all login and logout events. AUTH PRIV

/usr/share/doc/PAM-0.77/html/pam.html

Network Authentication

- Central password management
  - pam_krb5 (Kerberos V tickets)
  - pam_ldap (LDAP binds)
  - pam_smb_auth (old SMB authentication)
  - pam_winbind (SMB through *winbindd*)
- Some services use NSS/pam_unix
  - NIS, Hesiod, some LDAP configurations

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Additional modules exist which operate much like *pam_unix* but support alternative, network-based methods of authenticating users:

pam_krb5 authenticates users by using the password to decrypt a Kerberos V ticket for the user

pam_ldap authenticates users by using the password to LDAP simple bind to a directory server This module is insecure unless used with TLS encryption, since the password is sent cleartext!

pam_smb_auth and **pam_winbind** use different approaches to authenticate users through information stored in a Microsoft Windows domain controller.

Some name services, like NIS, Hesiod, and some LDAP configurations may use pam_unix to authenticate users through NSS information stored in the name service instead of using one of the additional modules listed above. For Hesiod or LDAP, supplementary PAM modules may still be needed in the password management group to properly change passwords in this mode.

auth Modules

- pam_securetty fails if logging in as root from a terminal not in `/etc/securetty`
- pam_nologin fails if the user is not root and the file `/etc/nologin` exists
- pam_listfile checks a characteristic of the authentication against a list in a file
  - A list of accounts can be allowed or denied

redhat 17

Rev-RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Some additional modules are useful in the `auth` managment group:

pam_securetty fails if the user is attempting to log in as root from a virtual console or serial terminal not listed in `/etc/securetty` This is intended to protect the root password from being captured from an insecure terminal.

pam_nologin fails if the user is attempting to log in as a non-root user and the file `/etc/nologin` exists. This can be used to run in multi-user mode for maintenance purposes while still locking non-privileged users out of the system.

pam_listfile checks an arbitrary characteristic of the authentication against a list of items in a file, such as the user or group that the program user is attempting to log in as. This list can either allow or deny access to the items on that list:

```
auth required pam_listfile item=user sense=deny \
          file=/etc/vsftpd.ftpusers onerr=succeed
```

Password Security

- pam_unix MD5 password hashes
  - Makes password hashes harder to crack
- pam_unix shadow passwords
  - Makes password hashes visible only to root
  - Makes password aging available
- Other modules may support password aging mechanisms

redhat 18

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The *pam_unix* module has two features which are useful to enforce more secure passwords. The `md5` option is set by default, indicating that passwords should be encrypted as MD5 hashes This is more secure than the standard DES-based hash method, and allows for passwords more than eight characters long. The `shadow` option is also set by default, indicating that that password hashes should be hidden from non-root users in `/etc/shadow` and that password aging should be turned on. Password and account expiration parameters can be set through the `chage` command For example,

```
chage -M 90 username
```

will cause the password for the user *`username`* to expire if not changed at least once every 90 days.

Other modules may also support password aging and account expiration mechanisms. For example, most Kerberos V key distribution centers support principal policies which control not just when the passwords for user principals expire but also other password policy rules.

Generally, `account` management group tests are used to see if an account or password has expired

Password Policy

- Password history
  - pam_unix with `remember=N` argument
- Password strength
  - pam_cracklib
  - pam_passwdqc
- Failed login monitoring
  - pam_tally

redhat 19

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Password policies define the basic requirements for how passwords may be used on a system. There are a number of PAM modules to implement various policies:

pam_unix can use the option `remember=N` in the `password` management group to store the last *N* password hashes for a user in `/etc/security/opasswd` and prohibit the passwords from being reused

pam_cracklib is a `password` module that enforces that passwords have at least a specific complexity. Among other tests, it can require that passwords are of a certain length, which may be shortened if the password contains digits, punctuation, and other characters besides letters It also requires that passwords are not dictionary words or variations on dictionary words.

```
password required pam_cracklib.so retry=3 minlength=12 \
                  ocredit=1 dcredit=1 lcredit=1 ucredit=1
```

pam_passwdqc is similar to pam_cracklib except that it does not mandate that passwords are not dictionary words (in order to allow use of pass-phrases) It can also be configured to suggest random passwords This module ships as a separate RPM package.

```
password required pam_passwdqc.so retry=3 min=11,10,10,9,8
```

pam_tally as an `account` module tracks the number of failed login attempts in `/var/log/faillog`, and as an `auth` module can deny access if there have been too many failed login attempts. If used, remember that an attacker can use this to mount a denial-of-service attack to lock valid users out of the system. The `/sbin/pam_tally` utility can be used to adjust the count of failed login attempts for a user.

session Modules

- pam_limits enforces resource limits
  - Uses `/etc/security/limits.conf`
- pam_console sets permissions on local devices for console users
  - Can be used as an **auth** module as well
- pam_selinux helps set SELinux context

Session modules are often used to help prepare a session for use.

pam_limits enforces resource limits like those set by the `ulimit` command at login. The configuration file `/etc/security/limits.conf` controls the exact limits that are enforced. This can include the number of simultaneous logins, the amount of memory used, the number of running processes, and so on. Soft limits can be overridden by the `ulimit` command, hard limits can not.

pam_console sets permissions on specific device files for the first non-root user to log in at the console. This allows the console user to have access to do things like mount the floppy drive or CD. The exact permissions are controlled by `/etc/security/console.perms`. In addition, *pam_console* can be used as an auth module; it passes if the user is on a local console, it fails if the user is logging in remotely.

pam_selinux is a helper module to set the correct SELinux security context (user, role, and domain) on login and logout. Under the default "targeted" policy, root's SELinux security context will be set to root:system_t:unconfined_t and most other users will have their SELinux security context set to user_t:system_t:unconfined_t. If a particular user has multiple roles available, pam_selinux can prompt the user to select between them on login.

Utilities and Authentication

- Local admin tools need authentication
  - su, reboot, system-config-*, etc.
- pam_rootok passes if running as root
- pam_timestamp for sudo-like behavior
- pam_xauth forwards xauth cookies

redhat 21

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Login is not the only time at which authentication may be required. Operations like using su, rebooting the system, or running a configuration utility may require authentication or that certain conditions are true about an account before the operation is authorized. There are a number of PAM modules useful for these programs:

pam_rootok passes if the calling program is run as root. This is useful for allowing root to su without entering a password.

pam_timestamp is used to implement sudo-like authentication timestamps. As a session module, it sets a timestamp file in /var/run/sudo after normal authentication is successful. As an auth module, if the timestamp file exists and is less than five minutes old, *pam_timestamp* passes. This allows authentication to succeed if it succeeded for this session within the last five minutes.

pam_xauth forwards xauth cookies. Used with su, this allows X clients run as the user you switched to to have access to your X display. By default, only non-root users forward their cookies. Note that any user that has your xauth cookie has full access to your X display and all its windows! A configuration file can be set up as ~/.xauth/export, that controls which users you will forward cookies to.

PAM_ACCESS.SO
PAM_TIME.SO.

PAM Troubleshooting

- Check the system logs
  - `/var/log/messages`
  - `/var/log/secure`
- PAM mistakes can lock out the root user
  - Keep a root shell open when testing PAM
  - Single-user mode bypasses PAM
  - Boot the system using a rescue disc

redhat 22

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

PAM generally logs login/logout events and errors to either `/var/log/messages` or `/var/log/secure`. Be sure to check these log files if you are having problems with authentication.

If you suspect that `system-config-authentication` has been run, that may overwrite manual configuration changes in `/etc/pam.d/system-auth` and also in the `/etc/nsswitch.conf` file used by NSS.

Misconfigurations of PAM can break all authentication. It is a very good idea to keep a root shell open at all times when testing changes to PAM. If you do manage to lock out all accounts, there are still ways to recover the system. Remember that single-user mode bypasses PAM authentication. Reboot the system and go into GRUB menu editing mode. Pass the kernel the argument `single` on the end of its command line, and boot. You should be dropped to a root prompt without being prompted for a password. If an unknown GRUB password is set, you can attempt to boot from a rescue CD and repair the problem from there. If a BIOS password is set or you otherwise are not able to boot the rescue environment on that computer, you may be able to physically remove the hard drive and install it temporarily into another computer which is under your control.

When debugging problems involving NSS, do not forget that the `getent` utility is very useful for troubleshooting.

NIS Overview

- Simple directory service for system and account information
- All NIS servers and clients are members of a named NIS domain
  - Single master server, multiple slave servers
- Minimal network security
- Support for NIS version 1 and 2

redhat 23

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The **Network Information Service**, or **NIS**, is one popular network service which can be used to manage system and account information on multiple systems from a central server. NIS uses a single master server and optionally one or more slave servers, each running `ypserv`, to share information with NIS clients running `ypbind`. The NIS protocol is based on Sun RPC, and therefore clients and servers must also run a local service called `portmap` which helps remote systems contact the local `ypserv` or `ypbind` program. Clients and servers which are bound to each other are normally members of the same *NIS domain*, identified by an arbitrary string selected by the system administrator.

NIS servers are typically used to synchronize account information. They can share the contents of `/etc/passwd`, `/etc/shadow`, and `/etc/group` files by converting them into NIS maps. Each NIS map consists of a set of key/value pairs. For instance, one typical NIS map used is passwd.byname, where the key is an account name and the value is the matching line of user information for that account in `/etc/passwd` format.

The network connection between servers and clients is not encrypted, and there are no integrity or secrecy guarantees on NIS information. Passwords are sent as hashes, but nevertheless security of NIS information is minimal. NIS is still widely used in low-security environments because setup and maintenance is relatively simple.

Red Hat Enterprise Linux supports version 1 and 2 of the NIS protocol on both clients and servers. The client software also may work with NIS+ (NIS version 3) servers, but this functionality is minimally maintained.

Service Profile: NIS

- Type: System V-managed services
- Packages: *ypserv*
- Daemons: `ypserv`, `rpc.yppasswdd`, `rpc.ypxfrd` {uses Portmap
- Scripts: `ypserv`, `yppasswdd`, `ypxfrd`
- Ports: Dynamically assigned by `portmap`
- Configuration: `/var/yp/*`, `/etc/ypserv.conf` (`/etc/yp.conf` for *ypbind*)
- Related: *portmap*, *ypbind*, *yp-tools*

Yellow Page.

redhat 24

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The NIS ypserv server is an SELinux restricted service when enforcing the default targeted policy on a Red Hat Enterprise Linux, version 4 system. The server uses a number of SELinux contexts for its files. For purposes of server configuration, the following contexts are important:

`system_u:object_r:var_yp_t`
For the contents of the `/var/yp` directory, including the NIS maps and the Makefile.

`system_u:object_r:etc_t`
For NIS configuration files in `/etc`, including the `ypserv.conf` file and the `yppasswdd` file in `/etc/sysconfig`.

When moving files to `/var/yp`, always set the context for those files:

```
chcon -R --reference=/var/yp /var/yp/newfile
```

NIS Server Configuration

- Install the *portmap* and *ypserv* RPMs
- Set the NIS domain name
  - Run `nisdomainname mydomain`
  - In `/etc/sysconfig/network` insert the line: `NISDOMAIN=mydomain`
- In `/var/yp/securenets`, specify the networks that may use your server
- Start `ypserv`

redhat 25

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

Whether installing a master server or a slave server, the initial configuration steps for a NIS server will be similar. Before starting, make sure that the *portmap* and *ypserv* packages are already installed on the machine.

Choose an arbitrary string for your NIS domain name. This does not need to be the same as your DNS domain name, and some system administrators feel that it is a good idea not to make them the same for security reasons. (Any client that can communicate with your server and knows its NIS domain name can bind to it and get full access to the NIS directory.) Set this by running `nisdomainname your-domain-name` as root. To make sure this happens at boot, add the line `NISDOMAIN=your-domain-name` to `/etc/sysconfig/network`.

To limit which clients can communicate with your server, create a `/var/yp/securenets` file. Each line in this file should be a netmask and network number for networks that contain NIS clients:

```
255.255.255.255     127.0.0.1
255.255.255.0       192.168.0.0
```

Finally, make sure `ypserv` is running:

```
chkconfig ypserv on
service ypserv start
```

Configuring a Master Server

- To share only user, group, and host name information, edit **/var/yp/Makefile**
  all: passwd group hosts netid
- Build the NIS maps from local files by using the makefile:
  /usr/lib/yp/ypinit -m
- Start **yppasswdd** to allow password updates

redhat 26

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Many simple NIS configurations only share passwd, shadow, group, and host information. There are a multitude of other resources that may be shared, but they are often not needed by clients. A Makefile in `/var/yp` controls how NIS maps are built from local files. You can limit which NIS maps are built by editing the `all` target:

```
all: passwd group hosts netid
```

While editing this makefile, there are additional variables (or *macros*) that may need to be set:

```
NOPUSH=true
MERGE_PASSWD=true
MERGE_GROUP=false
```

`NOPUSH` should be set to `true` if you have no slave servers. If you have slave servers, it should be set to `false`. `MERGE_PASSWD` should be set to `true` if you want to merge password hashes from `/etc/shadow` into the NIS `passwd` map. `MERGE_GROUP` should be set to `true` if you want to merge group password hashes from `/etc/gshadow` into the NIS `group` map.

Once `/var/yp/Makefile` has been edited, build the NIS maps by running `/usr/lib/yp/ypinit -m` as root on the master server. This will store the NIS maps as Berkeley DB files in `/var/yp/`*`domainname`*.

If you want to allow users to change NIS passwords, start `rpc.yppasswdd`:

```
chkconfig yppasswdd on
service yppasswdd start
```

Configuring a Slave Server

- Include the names of all slave servers in the master's **/var/yp/ypservers** file
- On the slave, transfer the initial NIS maps from the master server:
  /usr/lib/yp/ypinit -s *master*
- To rebuild and push NIS maps from master to slave, on the master run
  cd /var/yp; make

redhat 27

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

To set up a slave server, first make sure that the slave is listed in the master server's `/var/yp/ypservers` file. Also make sure that the master server's `/var/yp/Makefile` has `NOPUSH=false` set.

Then on the slave server, transfer an initial copy of the domain's NIS maps from the master server by running the command `/usr/lib/yp/ypinit -s` *`master's-host-name`* as root

Whenever you change the files used to build the NIS maps on the master server, you should `cd /var/yp` on the master server and run `make` as root. This will rebuild the NIS map DB files and run `yppush` to push the changes out to all slave servers.

You can also run the `ypxfr` command on the slave server to pull maps from the master server. (An optional service for the master server, `rpc.ypxfrd`, is intended to speed up `ypxfr` transfers, but may not work if processor architecture, operating system, or Berkeley DB library version differs between master and slave.)

NIS Client Configuration

- Must install *ypbind* and *portmap* RPMs
- **`system-config-authentication`**
  - Enable NIS to provide "User Information"
  - Specify NIS server and NIS domain name
  - Keep default "Authentication" (using NSS)
- What does this actually do?
  - Modifies four configuration files

redhat 28

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The easiest way to set up a client to use an existing NIS server is to install the *portmap* and *ypbind* packages on the client, and then run `system-config-authentication` to modify NSS and PAM. Under "User Information", enable NIS, and then a NIS domain and a NIS server for that domain must be specified. No changes are necessary under "Authentication" if NIS will be used for authentication, since it provides password hash information through NSS.

What does this change? In `/etc/sysconfig/network`, the variable `NISDOMAIN` is set to the NIS domain name, and the `nisdomainname` command is run to set it on the system immediately. The `/etc/yp.conf` file has a line added to it which specifies which server to use for that NIS domain. The `/etc/nsswitch.conf` file is modified to specify that NIS should be used as a source of information for passwd, shadow, and group lookups. The `/etc/pam.d/system-auth` file is modified so that password change requests for NIS accounts are sent to the master server's `rpc.yppasswdd` service. Finally, the `ypbind` service is started up and is set to automatically restart on reboot.

NIS is a relatively insecure service. To improve password security, Kerberos can be used in conjunction with NIS as an authentication service. A better solution might use LDAP protected with TLS (SSL) encryption to store name service information in place of NIS. However, these solutions can be more complex to set up and manage.

NIS Troubleshooting

- Is the default firewall still turned on? IPTABLES
- Are services running and registered with `portmap`?
  - `rpcinfo hostname`
- Use `ypwhich` to verify which server a client is bound to, if any
- Use `ypcat` and `getent` to verify that NIS data is available

redhat 29

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Whenever troubleshooting a network service, do not forget to check log files in `/var/log` for errors first. You can also view many of these log files with `system-logviewer`.

Since NIS uses dynamic port assignment by default, it can be challenging to use it in conjunction with a modified default firewall. Most NIS services can be bound to fixed ports by command-line options read from files in `/etc/sysconfig` by the service startup scripts.

The `portmap` service can be queried to determine if NIS services are properly registered and what port each service is currently using. Use the command `rpcinfo hostname` to get this information. In particular, look for `ypbind` on clients and `ypserv` and `rpc.yppasswdd` on servers. If there is no response, log into the server and verify that `portmap` is running with `service portmap status`. By default, `portmap` listens on ports tcp/111 and udp/111.

On a client system, after making sure `portmap` and `ypbind` are running, you can run the `ypwhich` command to see if the client is bound to a NIS server and which server it is currently using. If the client is bound, `ypcat passwd` should dump the NIS passwd map to standard output. If that works, verify that NIS information shows up in the output of the `getent passwd` command. If it does, but users do not appear to be using that information, make sure there are no duplicate entries coming from other name services (like local files) that are overriding the NIS information.

End of Unit 7

- Address questions
- Preparation for Lab 7
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 30

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 7
Authentication Services

Estimated Duration: 45 minutes

Goal: To build skills in authentication

Disable packet filtering. If you have not done so already, ensure the default host-based firewall is disabled. Either use `system-config-securitylevel` to do so, or as root, run the commands "`chkconfig iptables off; service iptables stop`". You may leave SELinux running.

Sequence 1: Authenticating users with NIS

Tasks:

You should partner with one of your neighbors and decide which of your workstations will be the NIS server and which will be the NIS client. Throughout this exercise, both you and your partner should configure the server and client together. You will need to decide on a name for your NIS domain, and note each workstation's name and IP address. Please make sure that authentication does work properly before starting this sequence.

1. Configure the NIS Server

   a. Install the following RPMs on your system if they are not already installed: *ypserv*, *portmap*, and *make*.

   b. Edit `/etc/sysconfig/network` so that the following line appears in the file, replacing *ourdomainname* with the NIS domain name you chose.

      `NISDOMAIN=`*`ourdomainname`*

      This will take effect the next time you reboot your machine. To set the NIS domain name without rebooting, type the command

      `domainname` *`ourdomainname`*

   c. Restrict access to `ypserv` to your networks. As root, create a `/var/yp/securenets` file containing the lines:

      ```
      255.255.255.255    127.0.0.1
      255.255.255.0      192.168.0.0
      ```

   d. Verify that the `portmap` service is running: `service portmap status`. If it is not, restart it and configure it to start on boot:

      ```
      service portmap start
      chkconfig portmap ~~start~~ on
      ```

e. Start the `ypserv` service and configure it to start on boot:

```
service ypserv start
chkconfig ypserv on
```

f. Edit the `/var/yp/Makefile`. Look for the `all:` target and edit it to read

```
all: passwd group hosts netid
```

g. Generate the NIS maps (databases) by running `ypinit`. Watch it's output for any error messages that might be generated.

```
/usr/lib/yp/ypinit -m
```

(*Note* You don't need to add any other machines to the list of hosts, just press **<CTRL-D>**)

h. Start the rpc yppasswdd service used to update passwords, and configure it to start on boot:

```
service yppasswdd start
chkconfig yppasswdd on
```

i. Verify the services have started: `ps auxf | grep yp`
Verify the services have registered with portmap: `rpcinfo -p localhost`
Check `/var/log/messages` for any errors.

Deliverable:

A working NIS server.

2 Configure the NIS Client

So far, only half of our task is complete You and your partner must now configure the client workstation to use the NIS server.

a. Install the following RPMs on your system if they are not already installed: *portmap*, *ypbind*, *yp-tools*, *authconfig*, and *authconfig-gtk*.

b Use `system-config-authentication` to confure your host to use NIS for authentication.

If you are using the GUI, check "`Enable NIS Support`" on the "`User Information`" tab You can leave the "`Authentication`" tab alone. Click the "`Configure NIS...`" button, then in the "`NIS Settings`" window specify your NIS domain name for "`NIS Domain`" and your NIS server's hostname for "`NIS Server`". Click on OK for the "`NIS Settings`" window and then on the main window.

If you are using a virtual console or the —nox option, under "`User Information`" check "`Use NIS`" and under "`Authentication`" check "`Use MD5 Passwords`" and "`Use Shadow Passwords`". Select the "`Next`" button, then on the "`NIS Settings`" screen specify your NIS domain name for "`Domain`" and your NIS server name for "`Server`" Select "`OK`".

Either way, ypbind should now start up successfully. If it does not, check `/var/log/messages` for errors.

c. Restart the `sshd` service to make sure that it registers the changes to authentication:

```
service sshd  restart
```

d. Test your NIS client. Run `ypwhich`, the name of your NIS server should be printed.

Run `ypcat passwd`, the `/etc/passwd` lines (and password hashes for `/etc/shadow`) for all users with UIDs of 500 and higher on the NIS server should be printed out

Run `getent passwd`, you should see all the `/etc/passwd` lines for local users on your system, as well as the information from the NIS password maps.

e. Use `useradd` to create a new user account on the client and a different new user account on the server. Then use `passwd` to set passwords for each.

(On the client):
```
useradd -u 1024 localguy
passwd localguy
```

(On the server):
```
useradd -u 1025 nisuser
passwd nisuser
```

f. Verify that you can log in on the client using `localguy` and on the server using `nisuser`. Then try logging in on the client using the `nisuser` account; this shouldn't work. This is because the information about the new user has not been added to the NIS password maps. Use `ypcat passwd` or `getent passwd` to verify this

g. In the `/var/yp` directory on the server, type `make`. Once this command completes, try logging into the client as `nisuser` again. It should succeed this time -- why? The NIS map has been rebuilt, and nisuser has been added to the map. Use `ypcat` or `getent` again to verify this

h. Use `passwd` to change the password of `nisuser`. Does this change information in the server's `/etc/passwd` or `/etc/shadow` file? Does the information in the NIS password database change? You can check this by examining the output of

```
ypcat passwd | grep nisuser
```

i. When you log in on the client as `nisuser`, what is your home directory? What directory are you actually in? NIS only provides information about where the user's home directory should be. It depends on the rest of the system to ensure that it is present and on other services such as NFS to share files between multiple workstations

Deliverables:

A client which can get authentication information from a NIS server

Sequence 2: Limiting NIS users

Tasks:

For security reasons, some of the users managed by the NIS server should be allowed to use the NIS client, but some should not. In this sequence, you will modify PAM on your NIS client to allow all local users and selected NIS users to log in, while prohibiting all other NIS users from logging in.

1. On your NIS server, add an additional account named `baduser` that will be prohibited from accessing your NIS client. Give `baduser` some valid password.

```
useradd -u 1026 baduser
passwd baduser
cd /var/yp;   make
```

2. On your NIS client, try to log in as baduser on a virtual console. You should get a shell prompt. Log out.

3. On your NIS client, open `/etc/pam.d/system-auth` in an editor. Examine the account section. It should read something like

```
account  required    pam_unix.so
  account  sufficient  pam_succeed_if.so uid < 100 quiet
  account  required    pam_permit.so
```

This says that on login, pam_unix.so should check the NSS shadow name service to make sure the account or password is valid. Then, `pam_succeed_if` checks to see if the account's UID is below 100(implying it is a local program account in `/etc/passwd`). If both checks pass, the account checks return success.

If the account's UID is 100 or higher, then the results of the `pam_succeed_if` module are ignored and we go on. (If you set up Kerberos or LDAP authentication in `system-config-authentication`, account checks using *pam_krb5* and *pam_ldap* are performed next). Finally, the last check is performed. The `pam_permit` check should always pass. If both `pam_unix` and `pam_permit` pass, then the account checks return success.

If `pam_unix` fails, then the account checks return failure.

4. First, let's lock out all non-local users. There is a PAM module, `pam_localuser`, that passes if an account's information is stored in local file (like `/etc/passwd`) and fails if it is not. On your NIS client, in `/etc/pam.d/system-auth`, edit the account section to read:

```
account  required    pam_unix.so
account  sufficient  pam_succeed_if.so uid < 100 quiet
account  required    pam_localuser.so
account  required    pam_permit.so
```

Save the file.

5. In a virtual console, try to login in as `baduser` and as `nisuser`. Both should fail. Try to log in as `localguy`. That should work.

6. Now we need to allow `nisuser` but not `baduser` to login One way to do that is with the `pam_listfile` module. You will create a file containing a list of NIS users allowed to use this machine In `/etc/pam.d/system-auth` on the NIS client, edit the account section to read:

```
account  required    pam_unix.so
account  sufficient  pam_succeed_if.so uid < 100 quiet
account  sufficient  pam_listfile.so item=user \
     sense=allow file=/etc/nisusers onerr=fail
account  required    pam_localuser.so
account  required    pam_permit.so
```

So now, after checking to make sure the account is not expired, and that it is not a local system account, PAM checks to see if it is a user listed in `/etc/nisusers` If it is, account checks pass and we stop. If it is not, PAM checks to see if the account is in `/etc/passwd`. If it is not, `pam_localuser.so` fails and account checks will fail.

Save the file.

7. You still have to create `/etc/nisusers` on your NIS client Each line in that file should be a non-local user to whom we want to grant access Create a file with a text editor, containing the line:

```
nisuser
```

Save the file Make sure the file is not world-writeable.

8. On your NIS client, try to log in as `baduser` on a virtual console. That should fail. Then try to log in as `nisuser`. That should work. Check `/var/log/messages` and `/var/log/secure` if you have problems

Challenge Question:

What would the effect be if you misconfigured the `/etc/pam.d/system-auth` file so that `pam_localuser` is a `sufficient` account test instead of a `required` account test? Try logging in as various NIS and local users to test your theory

Clean Up:

Run `system-config-authentication` on your NIS client and un-check NIS This will also wipe out your hand-edited changes to the `/etc/pam.d/system-auth` file On the NIS server, stop and `chkconfig` off `ypserv` and `yppasswdd`

UNIT 8

System Monitoring

redhat

1

Rev RH253-RHEL4-1

Objectives

- Learn to identify file statistics
- Ensure filesystem integrity
- Understand system log configuration
- Learn log file analysis
- Understand process monitoring

Agenda

- File system analysis with **`find`**
- Common log files
- Configuration of **`syslogd`** and **`klogd`**
- Process monitoring and accounting

redhat

3

Introduction to System Monitoring

- Security breaches can be detected with regular system monitoring
- System monitoring includes:
  - File system monitoring
  - Log file analysis
  - Process monitoring

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

In order to detect possible security breaches or system malfunctions it is imperative to monitor the system regularly System monitoring includes log file analysis, process, resources and file system monitoring.

Many problems can be detected and resolved before becoming critical if a system is closely monitored

File System Analysis

- Regular file system monitoring can prevent:
  - Exhausting system resources
  - Security breaches due to poor access controls
- File system monitoring should include:
  - Data integrity scans
  - Investigating suspect files
- Utilities: **df, du, logwatch**

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

File system analysis and monitoring is one of the key elements of a secure system. File system analysis can answer questions such as "Is there enough diskspace available?" or "What accounts have write access to this directory?". One situation that can often lead to security breaches are files and directories that are "left over" after the account or application that owned those files has been deleted. As new accounts are added to the system, these left over files can wind up being owned by one of the new accounts. Directories with write access for the '`other`' category are also potential security problems.

Red Hat Enterprise Linux provides several standard utilities that can be used to monitor file permissions, file system usage and other system activities that canaffect security.

One simple utility that can be used to monitor file system utilization is the '`df`' command. The '`df`' command reports on the total amount of space available on each mounted partition and can be used to catch situations in which a system is about to run out of space. A useful option for '`df`' is the '`-h`' option which converts disk units to gigabytes and megabytes.

The '`du`' command can be used to display the actual amount of storage required bya given file as opposed to the *length* of the file, which is reported by '`ls -l`'.

Set User and Group ID Permissions

- Programs owned by root with SUID or SGID permissions can be dangerous
- Security policy should include monitoring SUID programs
- Prevent SUID and SGID permissions on filesystems with **nosuid** mount option

redhat 6

Rev RI-253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Set User ID (SUID) and Set Group ID (SGID) programs run with the permission of the program's owner or group -- not those of the executing user. SUID and SGID files are prime targets for crackers, and should be carefully monitored.

When any user executes a SUID program owned by root, all the privileges given to root become effective for that program during its execution lifetime. For example, while the permissions for `/etc/passwd` are:

```
-rw-r--r-- 1 root  root    556  Mar 24 18:30 /etc/passwd
```

the file may still be updated by non-root users through the uses of the SUID program `passwd`:

```
-r-sr-xr-x 1 root  bin   15613  Apr 27  1998 /usr/bin/passwd
```

As part of your regular system monitoring practice, you should determine what files are SUID or SGID:

```
find / -type f -perm +6000
```

and compare those files returned with a list of authorized SUID and SGID programs. The presence of a SUID or SGID program not authorized by a system administrator could be a warning sign of your system having been successfully attacked and compromised.

SUID and SGID programs can be prevented from running at all on a filesystem-wide basis with the '`nosuid`' option for the mount command:

```
mount -o nosuid /dev/sdd3  /mnt/datadisk
```

The '`nosuid`' option can be added to the options listed for a given filesystem in `/etc/fstab` to make it 'permanent'. You should consider using the '`nosuid`' option when mounting any sort of removable media, such as floppy disks, which might contain SUID root programs designed to allow a malicious user to gain access to your system as root.

Typical Problematic Permissions

- Files without known owners may indicate unauthorized access:
  - Locate with: `find / \( -nouser -o -nogroup \)`
- Files/Directories with "other" write permission (`o+w`) may indicate a problem:
  - Locate with: `find / -type f -perm -2`
  - Locate with: `find / -type d -perm -2`

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The first `find` command on the slide above, will locate all files which have an owner UID or GID that cannot be located in either `/etc/passwd` or `/etc/group`, respectively The presence of such files could indicate illicit activity on the part of a cracker

The second `find` command on the slide above will locate all files on a system that have write permisson set. Although some files may need to have this permission set, they should be rare enough that you want to know which files have this permission set.

The final `find` command on the slide above will locate all directories on a system that have write permission set. Recall that having write permission to a directory permits a user to delete files in that directory, regardless of the access the user may have to the files in that directory

You may want to make scans of your filesystem with these commands a part of your regular filesystem maintenance procedure, perhaps by including them as part of a `cron` job that is executed in `/etc/cron.daily` or `/etc/cron.weekly`

EXT2/3 Filesystem Attributes

- EXT2/3 supports several special attributes that affect the behavior of files
- Show attributes with `lsattr`
- Set attributes: `chattr <file>`
- Some attributes not currently supported by the Linux kernel
- Some attributes unavailable for users

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Understanding EXT2/3 filesystem attributes

To set attributes on files, use `chattr +|-attributes <file>` where the letters "`AacdijsSu`" select the new attributes for the files:

- `A` atime record is not modified upon access or modification.
- `a` File can only be opened in *append* mode for writing. (`root` only)
- `d` File is excluded in backup by dump
- `j` File data is journaled to ext3 journal (`root` only)
- `i` File is *immutable*. It cannot be deleted or modified (`root` only)
- `S` All changes to this file are written synchronously on the disk; this is equivalent to the `sync` mount option applied to a subset of the files.

The following attributes are currently *not* supported by Red Hat Enterprise Linux(RHEL)kernels:

- `c` File is automatically *compressed*.
- `s` The file's content is zeroed upon deletion.
- `u` File's content is saved when it is deleted. Thus undelete is possible.

System Log Files

PROBLEMS.

- Why monitor log files?
- Which logs to monitor?
- Logging Services:
  - Many daemons send messages to **syslogd**
  - Kernel messages are handled by **klogd**

redhat 9

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Monitoring log files will help detect:

- equipment problems such as hard disk crashes or power outages
- user problems such as repeated login failures
- security breaches from outside the system

Log files to monitor include:

- `/var/log/messages`: logs most system messages
- `/var/log/secure`: authentication messages, `xinetd` services
- `/var/log/vsftpd.log`: FTP (`vsftpd`) transactions
- `/var/log/maillog`: mail transactions

The information contained in `/var/log/messages` includes:

- date and time the message was written
- name of the utility, program, or daemon that caused the message
- action that occurred
- executing program's hostname

The exact information in a log file depends on the application. Many applications create their own log files which may also need to be monitored

SYSTEM KERNEL

syslogd and klogd Configuration

- **syslogd** and **klogd** are configured in **/etc/syslog.conf**
- Syntax:
  facility.priority log_location
- Example:
  mail.info /dev/tty8

redhat 10

Rev RH253-RHEL4-1

The standard system logging daemons `syslogd` and `klogd` are both configured with `/etc/syslog.conf`. It is possible to configure what kind and what amount of system messages is stored in specific log files.

The format is straightforward. The first entry specifies a semi-colon delimited list of facility priority declarations. The second field specifies the log location, which is usually a file.

Facilities:

`authpriv`	security/authorization messages	`lpr`	printing system
`cron`	clock daemons (`atd` and `crond`)	`mail`	mail system
`daemon`	other daemons	`news`	news system
`kern`	kernel messages	`syslog`	internal `syslog` messages
`local[0-7]`	reserved for local use	`user`	generic user level messages

Priorities:

`debug`	debugging information	`err`	error condition
`info`	general informative messages	`crit`	critical condition
`notice`	normal, but significant, condition	`alert`	immediate action required
`warning`	warning messages	`emerg`	system no longer available

Example:

```
kern.info      /var/log/kernel
```

This will log all kernel-related messages of priority `info` and higher to `/var/log/kernel`.

Advanced **syslogd** Configuration

- Operators
  - ***facility.priority***
    facility messages with equal or higher *priority*
  - ***facility.=priority***
    facility messages with exact *priority*
  - ***facility.=!priority***
    facility messages except those with *priority*
  - ***facility1,facility2.priority***
    priority messages from *facility1* and *facility2*
  - ***.priority***
    messages with equal or higher *priority* regardless of facility
- Special Targets
  - Comma-separated list of users
  - Remote machine **(@hostname)**

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Logging can be further specified with certain operators:

=	log on only this exact priority
!	Exclude this facility or priority
*	Log all facilities/priority

Also it is possible to use advanced log locations You can specify a comma-separated list of users who will be notified or a named pipe for use with external logging programs (`| /name/of/pipe`). The pipe has to exist before `syslogd` starts

It is also possible to log to different machines. This can be achieved by defining `@hostname` as log location. In addition to local log files, a central logging server simplifies log analysis. However, local logfiles should not be abandoned, since the network or logging host may not be always accessible.

Remote and local logfile content should be compared occasionally Differences between the files indicate network errors or tampering with one of the files.

A remote syslog daemon only accepts incoming messages if remote functionality is enabled using the `-r` command line switch This and other startup options can be specified in `/etc/sysconfig/syslog`.

CCZE

Log File Analysis

- Should be performed on a regular basis
- **`logwatch`** can be installed to run by **`crond`** every hour to report possible issues
- When looking for anomalies, **`logwatch`** uses negative lists
  - Discard everything normal
  - Analyze the rest

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Log file analysis should be performed on a regular basis. Since log files can grow very big on larger servers, it is nearly impossible to look through all the data by hand. It is therefore practical to use log analysis tools like `logwatch`, which is installed by default on most systems.

`logwatch` runs daily, although on a busy server and even on many workstations, it should be run much more often, perhaps every hour. It reports any activity, except those actions that it is programmed to ignore. It is configured in `/etc/log.d`.

In highly-customized environments, it is often necessary to implement custom log analysis tools. The programming language Perl was designed exactly for this purpose. Its powerful regular expressions make it very easy to write such programs. Many available tools in this area are programmed in Perl for this very reason.

Monitoring Processes

- Monitor processes to determine:
  - Cause of decreased performance
  - If suspicious processes are executing
- Monitoring utilities
  - `top`
  - `gnome-system-monitor`
  - `sar`

redhat 13

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

Monitoring processes is key to understanding how your system behaves. Processes that are taking up a large percentage of your system resources are either written poorly, being abused, or simply not designed to run on your system. Correcting these issues before they cripple or cause damage to your system is the heart of process monitoring.

Process Monitoring Utilities

- **top** z - change colors
  - view processor activity in real-time
  - interactively **kill** or **renice** processes
  - watch system statistics update through time, either in units or cumulatively
- GUI system monitoring tools:
  - **gnome-system-monitor**: GNOME process, CPU, and memory monitor
  - **kpm**: KDE version of **top**

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`top` is an invaluable tool for actively monitoring the processes running on a RHEL system. With `top`, you are able to view the most CPU-intensive tasks, manipulate processes (e.g., kill) or track the system performance over time.

`top` updates its output every 5 seconds by default; the delay may be changed interactively with the `s` key. The output consists of a header block of information showing overall system statistics, and a process block showing a list of processes and their associated information.

The header block contains such information as load average, number of processes, available RAM and swap space, and so on. This information is useful for getting a quick picture of the system state.

The process block contains information such as process ID, user who started the process, priority and nice level, memory, CPU and frame consumption, activity time and command. You can change the sort order of the list with `M` (sort by memory), `T` (sort by time), and `P` (sort by CPU usage).

To show processes owned only by a particular user, use the `u` key followed by the user name or UID you wish to see. To interactively kill a process, use the `k` key, followed by the process ID of the victim, and then the signal to send. Simply pressing `<enter>` for the signal number will use the default of 15 (`SIGTERM`). To renice a process, use the `r` key, enter the PID number, and then select a nice value.

`gnome-system-monitor` displays process information. Processes can be listed in a tree like fashion or independently. By selecting column headers, processes can be sorted a variety of different ways. The `system monitor` tab also displays the percentage of CPU, memory, swap space, and filesystem space in use.

`kpm` is a program which is similar to `gnome-system-monitor`. The process control features of `kpm` include re-nice, kill, re-scheduling by root, and sending signals. To send a signal to a process, use the "Signal" menu. To re-nice a process, use the "Process" menu.

Other graphical tools help system administrators monitor their system. These tools are traditional X11 commands, and can be found in `/usr/X11R6/bin`. Among them are `xosview` and `xload`.

Be aware that these require more resources themselves than command-line utilities (which also consume some resources), and thus may not be very useful in situations in which resources are already strained.

System Activity Reporting

- Frequent reports, over time
  - `cron` spawns `sa1` and `sa2`
  - `sar` reads and generates "human friendly" logs
- Commonly used for performance tuning
  - more accurate statistics
    - binary "database" collection method
    - regular intervals
  - Evidence of pattern establishes "normal" activity

redhat 15

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

Installation of the *sysstat* RPM automatically configures frequent system activity reports through the `cron` service (see `/etc/cron.d/sysstat`). Every 10 minutes `sa1` runs, passing options to the `sadc` program which takes a one second snapshot of system activity. Each night `sa2` runs `sar` which reads the binary "database" generated by `sadc`. Both the binary database and "human friendly" `sar` reports are found in `/var/log/sa`.

While `sadc` and `sar` may be run interactively, the `cron` invoked implementation combined with the default seven (7) day log life-cycle provides more accurate evidence of system activity patterns. Administrators often use these reports—and the accompanying `iostat` and `mpstat` utilities—to optimize system resources. In the context of system and data security however, these system resource usage patterns may also describe unusual system behavior when it occurs. Abnormal system activity may be the cause, or effect of a security breech

SYSTAT . RPM

Limiting Processes

- Use PAM to set resource limits for processes:
  - **pam_access.so** can be used to limit access by account and location
  - **pam_time.so** can be used to limit access by day and time
  - **pam_limits.so** can be used to limit resources available to process

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Monitoring may turn up processes that are using more than their fair share of resources or you may have a need to restrict access to the system during certain hours so that backups and other maintenance procedures can be performed. Certain PAM modules can be used to limit or control access to the system. Each of these modules has an associated configuration file in '`/etc/security`'.

The '`pam_access.so`' module can be used to restrict access to the system from only certain terminals or types of terminals. The configuration file for this module is '`/etc/security/access.conf`'

The '`pam_time.so`' module can be used to restrict the time of day that a user may log in to the system or run certain commands. The configuration file for this module is '`/etc/security/time.conf`'.

The '`pam_limits.so`' module can be used to limit the number of processes a given user may create, the amount of CPU time a process may consume, the default nice value for a process, and other limits. The configuration file for this module is '`/etc/security/limits.conf`'

Process Accounting Tools

- **`history`** shell built-in command listing
- **`last`** displays user's login history
- Process accounting
  - provided by *psacct* package
  - Activated by **`accton`**
  - Potential performance impact
  - `ac` displays user connect times from `/var/log/wtmp`

ac
LastComm

redhat 17

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

The `/var/log/wtmp` file only maintains rudimentary accounting information. To obtain more detailed accounting information, you will need to install the *psacct* package

To enable process accounting, issue the command:

```
accton /var/account/pacct
```

To disable accounting, issue the command:

```
accton
```

Note that running the comman without an argument will always disable process accounting.

The statistics gathered by the process accounting utility can be displayed with the '`lastcomm`' utility

The '`ac`' utility, which is also part of the process accounting package, prints information about users' connect times contained in `/var/log/wtmp`. Some useful options include:

- `-p`: display totals for each user
- `-d`: display totals for each day
- `--complain`: list ttys that do not have records

The '`sa`' utility summarizes information in '`/var/account/pacct`' and displays the results to standard output. With the `-s` option however, information will be written to the following files:

- `/var/account/savacct` – contains a summary of accounting information by command
- `/var/account/usracct` – contains a summary of accounting information by user

You should also note that since process accounting requires that the kernel record information about every process that it executes, it can negatively impact system performance. You may want to consider using process accounting only when it is absolutely necessary to be able to track specifically what andwhen processes are doing. For example, if you suspect an application on your system has been compromised by an attacker, you might enable processing accounting to help establish an audit trail for the application. Once the problem has been corrected, you would disable process accounting to avoid the uneccessary overhead on the system

End of Unit 8

- Questions and Answers
- Preparation for Lab 8
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 18

Lab 8
System Monitoring

Estimated Duration: 45 minutes

Goal: To build skills at enforcing file and file system security by locating files prone to security vulnerabilities.

Sequence 1: Locating vulnerable files and directories

Scenario/Story:

Search the filesystem for vulnerable files and directories.

Tasks:

1. Locate SUID and SGID files and store their names in `/root/stickyfiles`:

```
find / -type f -perm +6000 2> /dev/null > /root/stickyfiles
```

2. Locate world-writable files and store their names in `/root/world.writable.files`:

```
find / -type f -perm -2 2> /dev/null > /root/world.writable.files
```

3. Examine `/root/stickyfiles` and `/root/world.writable.files` to see which files fall into each category

Sequence 2: Controlling access to files

Scenario/Story:

You need to make some files available to users. However, you want to control the type of access users have to those files.

Tasks:

1. Create a user named `supervisor`

2. Create two files in `supervisor`'s home directory:
```
touch /home/supervisor/{payroll,old.employees}
```

3. Prevent the `payroll` file from being deleted:
```
chattr +i /home/supervisor/payroll
```

4. Allow data to only be appended to the `old.employees` file:
```
chattr +a /home/supervisor/old.employees
```

5. Verify that the attributes have been changed:
```
lsattr /home/supervisor/*
```

6. Try to remove the payroll file:
```
rm /home/supervisor/payroll
```

   What error message do you receive?

7. Try to edit the `old.employees` file. Do you receive an error message when trying to save your changes? Why or why not? Does the error make sense? Give the command:
```
echo "foobar" >> /home/supervisor/old.employees
```

   Why does this command work?

Deliverables:

1. The `/home/supervisor/payroll` file cannot be deleted.

2. The `/home/supervisor/old.employees` file can have records added to it, but no data can be removed from the file

Sequence 3: Logging to a centralized loghost

Scenario/Story:

Your boss thinks it is a great idea to have one central logging host

Tasks:

Work together with your neighbor

1. First, set up `syslogd` to accept remote messages. Edit `/etc/sysconfig/syslog`:
```
SYSLOGD_OPTIONS="-r -m 0"
```

2. Restart `syslogd`:
```
service syslog restart
```

   Now your machine will accept logging messages from other machines.

3. Set up `syslogd` to send some messages to another machine. Append in `/etc/syslog.conf` the following line:
```
user.*          @stationX
```

   Where `stationX` is your neighbor's machine.

4 Restart `syslogd`:

```
service syslog restart
```

Now your machine sends messages from user programs to your neighbor's machine

5. Test the new setup by using `logger` to generate a `syslog` message:

```
logger -i -t yourname "This is a test"
```

Does the message appear in your neighbor's `/var/log/messages` ?

Questions:

Why does this message also appear in your own `/var/log/messages`?

How can you prevent it?

UNIT 9

Securing Networks

1

Rev RH253-RHEL4-1

Objectives

- Explain packet filtering architecture
- Explain primary filtering command syntax
- Explain Network Address Translation
- Provide examples
- Show how to maintain configuration

redhat 2

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Agenda

- Introduce packet filtering architecture
- Describe Netfilter configuration
- Demonstrate rules by example
- Describe NAT
- Making rules persistent

redhat 3

IP Forwarding

- Effectively makes your Linux box a router
- Usually used with two network interfaces
- Can be used with dynamic routing and firewall services
- Configure by setting `net.ipv4.ip_forward` kernel variable
  - `/etc/sysctl.conf`
  - `/proc/sys/net/ipv4/ip_forward` (not persistent)

4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

A process creates a *packet* to carry data over an Internet Protocol (IP) network. Among the many components which make up each packet, the "to" and "from" IP addresses are carried in the packet *header*. If the destination host IP address is not on the same network(or subnet) as the originating host, a router, or gateway is required. It forwards incoming packets on one interface to another interface based on the packet's destination address. each interface is configured with a unique IP address. If two networks use separate media (Ethernet/X.25/PPP), packets are automatically transformed to match the other medium's specification. If a packet is too large to fit, it is fragmented. Even these "fragments" have a "to" and "from" address; however, the header does not contain any information about the route, or "journey," taken by this packet.

A Red Hat Enterprise Linux machine can be configured to serve as a router/gateway. Usually such a system will be configured with a separate interface card for each network it is attached to. However, the RHEL kernel does not forward packets between interfaces automatically. Before routing functionality will work, the `net.ipv4.ip_forward` kernel feature must be turned on:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

or persistantly in `/etc/sysctl.conf`:

```
net.ipv4.ip_forward = 1
```

A router system can also serve as a firewall, filtering traffic between networks (systems that are not routers can also run system-specific firewalls). It may also support dynamic routing protocols such as OSPF and BGRP, which allow it to communicate with other routers in determining the best path to send packets by

Routing

- Routers transport packets between different networks
- Each machine needs a default gateway to reach machines outside the local network
- Additional routes can be set using the **`route`** command
- Permanent entries can be placed in **`/etc/sysconfig/static-routes`**
- Dynamic routing protocols are used for greater flexibility

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The current routing table can be displayed with `route`:

```
[root@server1 /]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 eth1
```

Permanent static routes can be defined in the `/etc/sysconfig/static-routes` file. For example, adding the line:

```
any net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.190 dev eth0
```

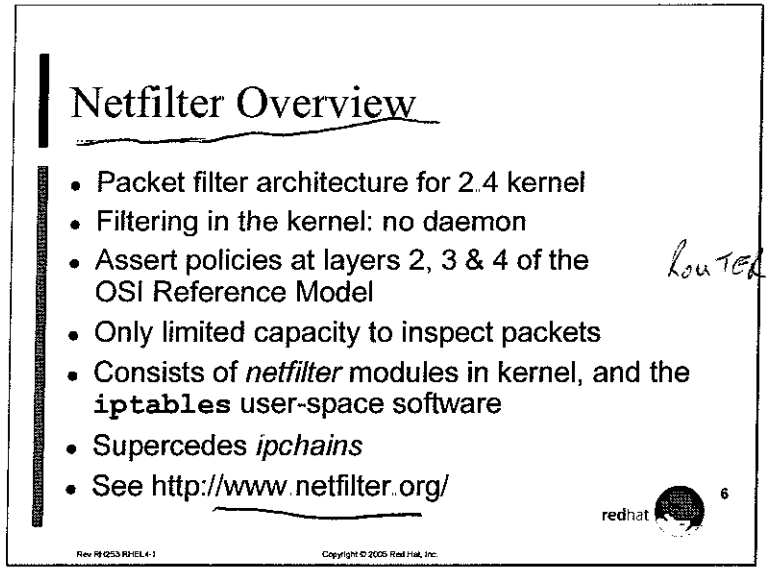
would (after running "`service network restart`") cause a new route to be added to the kernel routing table, specifying that packets destined for the 10.X.X.X network be sent out *eth0* via the 192.168.0.190 gateway. The "`any`" at the beginning of the line means that the route is started when networking is started, not when a particular interface is. The syntax of the rest of the line is similar to of the "`route add`" command, which can be used to create temporary static routes. See "`man route`" for details.

Routing must work in both directions. All routers between the hosts must be configured correctly. You can use `traceroute` to diagnose routing problems.

Dynamic routing allows greater flexibility, because routes are automatically negotiated between participating routers. Routes can be chosen by least cost, lowest latency or using other options. If a route fails, alternatives are used if available.

The '*quagga*' package provides a set of daemons that together implement a large number of dynamic routing protocols Dynamic routing protocolsallow a Red Hat Enterprise Linux server acting as a router to communicate with other routers in determining the best path to send a packet along. For example, if such a system were connected to two other routers and traffic could reach itsdestination by going across either, the RHEL system could use information gathered from the other routers to determine which was closer to the destinationnetwork, had a faster connection and so forth On busy or regularly-changing networks this allows routers to handle traffic much more efficiently than with static routes. Supported protocols include OSPF, OSPF2, RIP, RIPng and BGP. Eachprotocol is implimented by its own daemon (ospfd, ripd, etc), which communicateswith a master daemon called zebra The zebra daemon is the only one with the autority to modify the kernel routing table

For more information, see `/usr/share/doc/quagga-<version>/quagga.html`



Packet filtering is an operation performed by the network stack within the kernel itself. Packet filtering occurs without any user-space daemon process.

Filtering occurs in the kernel and at Open Systems Interconnect(OSI) layer "4" and below It is very fast because only the packet headers are inspected. Inspection of packet contents can be useful for things like tests for protocol correctness, however this usually happens at layers above layer "4", and involve user-space processes.

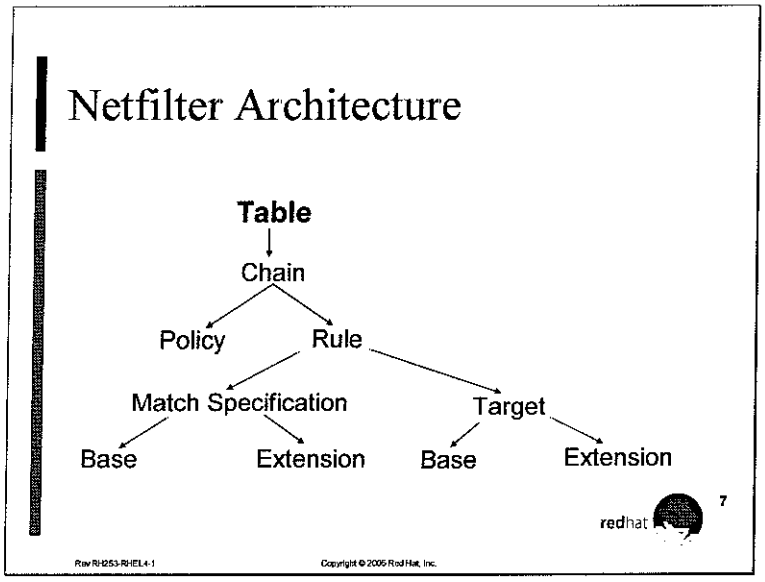
Most of the packet filtering is provided by Netfilter/IP Tables at layers "3" and "4" (*network* and *protocol*). However it can also use the interface (e g. eth0), a layer 2(*data-link*) attribute.

The packet filtering architecture has evolved rapidly with the Linux kernel, and has changed names in the process:

Kernel version	Packet filter
2.0	ipfwadm
2.2	ipchains
2.4	Netfilter(iptables)

Netfilter is an open source project in its own right, and contributes the kernel-space and user-space components.

The capabilities of Netfilter can be extended through an open process described in the `netfilter-extensions-HOWTO` at http://www.netfilter.org/ This normally involves creation of new kernel code (usually modules) and use-space code (patches to iptables command). Extensions normally implement new packet match capabilities (match extensions) or a new target capabilities (target extensions)



The core capability is implemented in the *ip_tables* and *iptable_filter* kernel modules in `/lib/modules/<version>/kernel/net/ipv4/netfilter/` Extensions (target or match) usually involve unique kernel modules

Match specifications for rules are also refereed to as *selectors*

Netfilter Tables and Chains

- Built-in Chains:

Filtering Point	Table		
	filter	*nat*	*mangle*
INPUT	X		X
FORWARD	X		X
OUTPUT	X	X	X
PREROUTING		X	X
POSTROUTING		X	X

redhat 8

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

The filtering framework consists of eleven built-in chains in the three tables as in the above diagram. The table in which a chain resides relates to the role of the rules as follows

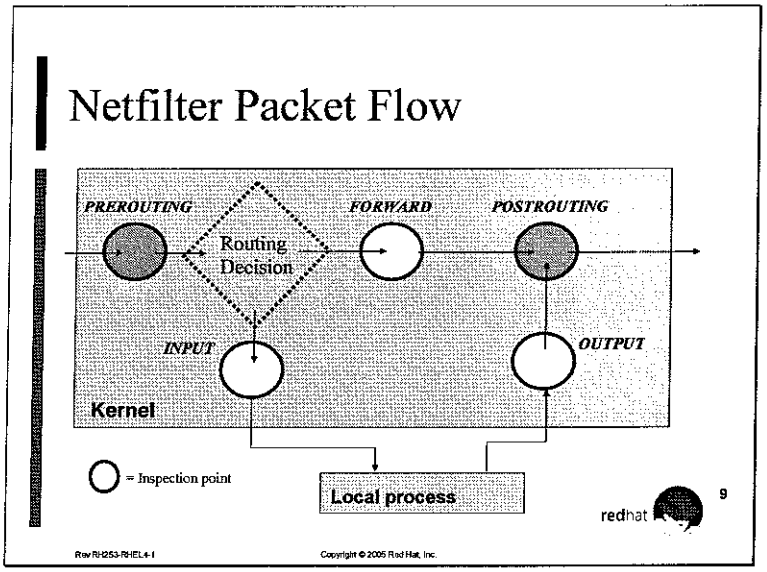
filter The main packet filtering is performed in this table.

nat This is where Network Address Translation(NAT) occurs.

mangle This is where a limited number of 'special effects' can happen This table is rarely used

Note the names of the tables are case-sensitive and are in lower case

Additional chains (the so-called *custom* chains) can be created at runtime. The built-in chains are always present, even if, in some cases, they might not be used.



Packet filtering takes place within the kernel at the five packet filtering points shown. Note that the filtering point names are case-sensitive and are in upper case

PREROUTING This filtering point deals with packets first upon arrival. (nat)

Routing Decision If a packet's destination address corresponds to the local system, then packets are routed to be handled there, by a local process. If the packet is to be delivered to another system, and packet forwarding is enabled in the kernel (see `/proc/sys/net/ipv4/ip_forward`) then packets are directed in accordance with the routing table.

FORWARD This filtering point handles packets being routed through the local system (filter)

INPUT This filtering point handles packets destined for the local system, after the routing decision (filter)

OUTPUT This filtering point handles packets after they have left their sending process, and prior to POSTROUTING (nat & filter)

POSTROUTING This filtering point handles packets immediately prior to leaving the system (nat)

Rule Matching

- Rules in ordered list
- Packets tested against each rule in turn
- On first match, the target is evaluated: usually exits the chain
- Rule may specify multiple criteria for match
- Every criterion in a specification must be met for the rule to match (logical 'and')
- Chain policy applies if no match

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Rules are in an ordered list. The position of a rule in the list has a bearing on when and if the rule will be used. Packets are tested against each rule (starting at the top of the list) in turn.

Netfilter matching is on a first match basis. If a packet's characteristics match a rule, then the rule's target is evaluated, which usually means that packet checking stops. Packet filtering on some operating systems work on a last-match basis.

If a rule specifies multiple criteria in the match specification, then packets must match every one for the packet to be considered matched by that rule.

If a built-in chain is traversed entirely and no match is found then the chains default policy applies. In the case of a custom chain, if there is no match then control returns to the chain from which the custom chain was called.

Rules: General Considerations

- Defaults to mostly open (**ACCEPT**). Mostly closed is more appropriate
  - **`iptables -P INPUT DROP`** or
  - **`iptables -A INPUT -j DROP`**
- Criteria also apply to loopback interface
  - The example rules above will have the side effect of blocking localhost!
- Rules, like routes, are loaded in memory and must be saved to a file for persistence across reboots

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

A default installed RHEL system without any iptables rules asserted, will have empty built-in chains with a policy of `ACCEPT`. In this way the packet filtering facility can be present without having any effect on system resources.

Approaches to rule design can be classified as either *mostly open* or *mostly closed*. A mostly open approach allows all packets by default and only blocks known-bad traffic. A mostly closed approach blocks all packets by default andonly allows known-good. A mostly closed approach to rule design is more cautious and usually considered more appropriate.

There are two techniques for creating a mostly closed ruleset for a chain. One is to set the chain's policy to `DROP`:

```
iptables -P INPUT DROP
```

The other is to create a "catch-all" `DROP` rule at the bottom of the chain:

```
iptables -A INPUT -j DROP
```

Both techniques will cause all traffic that is not explicitly allowed to be blocked. Note that this means NO network services, including ones that listen on localhost (127.0.0.1) will work if there are no other rules in the chain. The difference between the two techniques has to do with iptables' behavior when the chain is "flushed" and all the rules are deleted. Since a chain's policy is not affected by a flush, flushing a chain with a `DROP` policy will cause all network services to become unavailable (very bad if you are administering the system remotely). However, flushing a chain with a catch-all `DROP` rule will remove the rule so that the chain reverts to its default `ACCEPT` policy. This will not interrupt access to any services but it will leave the system wide open, as if no firewall at all were running. Which technique you use should be dependent upon your level of access to the machine and the number of sensitive services being protected by the firewall.

Packet filtering rules applied using the iptables commands alone, are not automatically re-applied on reboot(not persistent). However they can be made persistent as described later in this unit.

Match Criteria (filter table)

- A rule can match many characteristics of a packet:
  - Incoming interface (`-i`)
  - Outgoing interface(`-o`)
  - Layer 4 protocol (`-p`)
  - Source IP address (`-s`)
  - Destination IP address (`-d`)
- The above are base capability

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Interface matching examples:

```
iptables -A INPUT -i eth0 -j DROP

iptables -A OUTPUT -o eth1 -j DROP

iptables -A FORWARD -i eth1 -o eth2 -j DROP
```

Protocol matching examples:

```
iptables -A INPUT -p icmp  -j ACCEPT
```

Address matching examples:

```
iptables -A INPUT -s 127.0.0.1 -j ACCEPT

iptables -A INPUT -d 123.123.123.1 -j DROP
```

TCP Match Extensions (filter table)

- Additional criteria can be used as the basis for packet matching:
  - Protocol `-p`
  - Source port `--sport`
  - Destination port `--dport`
  - TCP flags `--tcp-flags`
  - Initial connection request `--syn`

redhat 17

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Some of Netfilter's capacity to match packets is not built-in, but may be called by the match extension, using the `-m` option to the iptables command

An example of invoking a match extension is:

```
iptables -A INPUT -m tcp  -p tcp --dport 80 -j DROP
```

In this example, the `-m tcp` invokes the TCP match extension, and the `-p tcp` calls up a feature of the extension to match based on the TCP protocol. This is the generalized operation of match extensions, but some cases like this can be simplified. The above command could alternatively have been cast as:

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

thus leaving out the `-m tcp`. In other words, the match extension may sometimes be implicit. When in doubt, use the former example for clarity, and *security*.

UDP and ICMP Match Extensions

- Match source and destination ports with UDP extensions:

```
iptables -A INPUT -m udp -p udp --sport 123 -j DROP
```

- Match ICMP types:

```
-p icmp --icmp-type destination-unreachable
```

redhat 18

Rev RH253 RHEL4-1

Match Arguments

- Matches may be made by:
  - IP address, or host name
  - Port number, or service name
  - Arguments may be negated with '!'
- Inclusive port range may be specified '0:1023'
- Masks may use VLSN or CIDR notation

redhat 19

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Host or service names are resolved at the point of rule insertion. Service names can be used in place of port numbers. Valid names are those found in `/etc/services`

Logical negation can be accommodated by prefixing the match element with "!", e.g.:

```
iptables -A INPUT -s  ! 123.123.123.1 -j DROP
```

which is the same as:

```
iptables -A INPUT ! -s  123.123.123.1 -j DROP
```

"Dotted quad," or Variable Length SubNet(VLSN) mask notation refers to:

```
192.168.0.0/255.255.255.0
```

"Prefix length," or Classless Inter-domain Routing(CIDR) mask notation refers to:

```
192.168.0.0/24
```

The above two mask examples are equivalent.

Chain Criteria

- Outgoing interface (`-o`) may only be used in the **`FORWARD`**, **`OUTPUT`** and **`POSTROUTING`** chains
- Incoming interface (`-i`) may only be used in **`FORWARD`**, **`INPUT`** and **`PREROUTING`** chains
- Owner match (**`--*-owner`**) may only be used in the **`OUTPUT`** chain

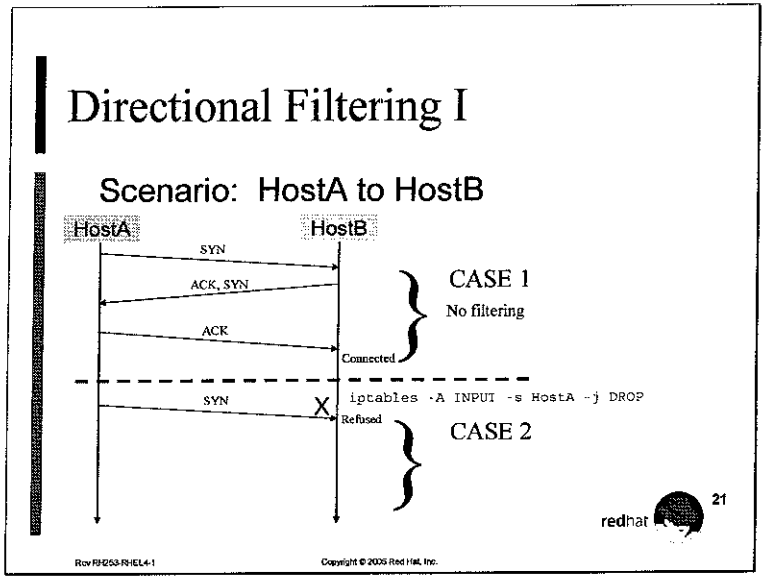
redhat 20

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Some selectors only apply in certain situations The ability to specify the incoming interface is meaningless in the `OUTPUT` chain. Similarly, the outgoing interface cannot be specified in the `INPUT` chain. The ability to select based on both incoming and outgoing interface is unique to the `FORWARD` chain

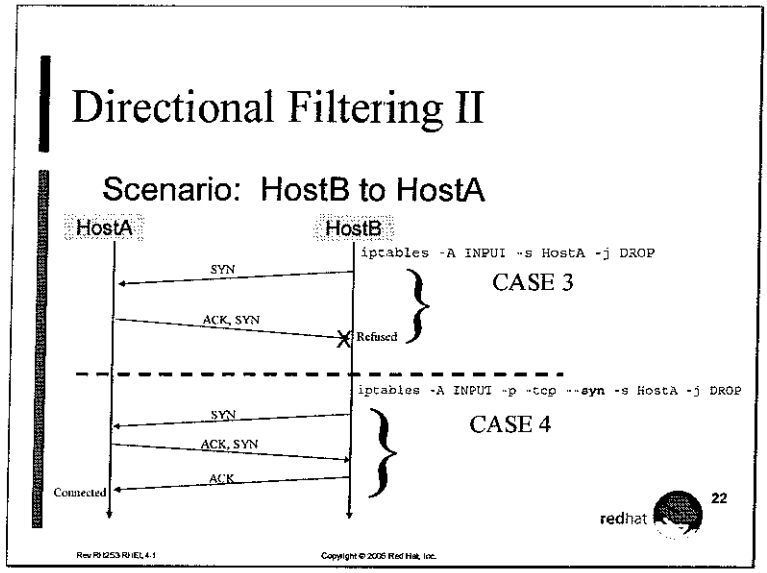
Another available extension is the `owner` extension The owner match capability only applies in the `OUTPUT` chains, since it is most certain *who* owns the local process generating the packet Even then, this extension may not match: see `iptables(8)`. An example of owner matching is:

```
iptables -A OUTPUT  -m owner --uid-owner 501 -j ACCEPT
```



Case 1 shows normal TCP connection behavior, regardless of the TCP payload, *ie* examples of this could be telnet, ssh, http, or ftp traffic.

Case 2 introduces a simple filtering rule applied on HostB which drops packets based on the source address of HostA. The first packet, with the source address of HostA, matches the filtering criterion and is dropped.



Case 3 shows the implications of the same filtering rule as Case 2, when HostB attempts to connect to HostA using TCP. Blocking based on source address of A prevents TCP connections initiated from either end.

Case 4 introduces an additional criterion in the filtering rule with the –syn flag. With the syn criterion in a rule, packets will only match if the SYN flag is set, and the ACK flag is not set So in this case B attempts to connect to A and succeeds because the second packet, with ACK and SYN flags set, no longer matches the rule. Thus, with the syn criterion a rule can be directional for TCP traffic (well, at least well-behaved TCP traffic, but more about that later).

Connection Tracking

- Provides inspection of packet's "state"
  - a packet can be tested in a specific context
- Simplifies rule design
  - without connection tracking, rules are usually in pairs(inbound & outbound)
- Implemented in `state` match extension
- Recognized states: **NEW, ESTABLISHED, RELATED, INVALID**
- Requires much more memory

redhat 23

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Connection tracking is arguably the most important new feature of Netfilter over what was in ipchains The term *connection* could be thought to indicate that it only relates to TCP traffic, because TCP is connection-based However this is not the case, and the connection tracking capability also applies to (connectionless) UDP traffic Instead of thinking in terms of connections (because it suggests TCP-style notions of what constitutes a connection), it is more appropriate to think in terms of exchanges of packets Netfilter understands how exchanges are stateful

When connection tracking is not employed, it is usually necessary to open up high ports fairly indiscriminately to provide for return packets With connection tracking, this approach can be avoided

Although connection tracking makes the kernel work harder, it can often be faster overall because after a connection is established, the chain traversal (ie the number of rules tested in a chain before a match) for remaining packets in the same connection can be minimized

Connection Tracking Example

- One rule to permit established connections:
```
iptables -A INPUT -m state \
--state ESTABLISHED,RELATED -j ACCEPT
```
- Many rules; one for each permitted service:
```
iptables -A INPUT -m state --state NEW \
-p tcp --dport 25 -j ACCEPT
```
- Lastly, one rule to block all others inbound:
```
iptables -A INPUT -m state --state NEW \
-j DROP
```

redhat 24

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

This example shows a typical use of connection tracking. The first rule allows any packets which are part of, or related to an existing exchange to be accepted. Since most packets will be part of existing exchanges, this arrangement is very efficient. The subsequent rules on the `INPUT` chain characterize the first packet of exchanges which are permitted. The last rule listed above refuses all other, undefined attempts to connect. This last rule may be replaced by setting the `INPUT` chain policy to `DROP`. Outbound packets are dealt with similarly in the `OUTPUT` chain.

Network Address Translation (NAT)

- Translates one IP address into another (inbound and/or outbound)
- Allows "hiding" internal IP addresses behind a single public IP
- Rules set within the `nat` table
- Network Address Translation types:
  - Destination NAT (`DNAT`)
    Set in the `PREROUTING` chain where filtering uses translated address
  - Source NAT (`SNAT`, `MASQUERADE`)
    Set in the `POSTROUTING` chain where filtering *never* uses translated address

redhat 25

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Network Address Translation "NAT" is a netfilter feature that allows a Red Hat Enterprise Linux firewall/gateway to alter the source or destination address of packets that pass through it. This is most commonly done to allow alltraffic going out the gateway to appear as though it is coming from a single address. This reduces the number of routable IPs an organization must purchase and also makes it more difficult for outsiders to learn details about the numberof machines on an internal network the addressing scheme used therein and so forth.

Source NAT translates the source address of outbound packets, and the destination address of return packets So the sense of source or destination is with respect to the first packet of the exchange.

The translation performed by Netfilter deals with both addresses and with ports, *ie* it is possible that source address and source port of a packet will change as the result of an SNAT translation. This is what is referred to as Network Address and Port Translation (NAPT).

From the `iptables` man page: *If no port range is specified, then source ports below 512 will be mapped to other ports below 512: those between 512 and 1023 inclusive will be mapped to ports below 1024, and other ports will be mapped to 1024 or above. Where possible, no port alteration will occur.*

Although translation of addresses needs to happen symmetrically in two places (outbound and return), only a single rule with respect to the first packet is required to perform this. So with SNAT, the translation of the destination address of return packets is done automatically, and without an explicit rule

`SNAT` and `MASQUERADE` are subtly different. The `MASQUERADE` target is appropriate for use with interfaces with an address which is assigned dynamically by DHCP, and only requires that the relevant interface be specified. On the other hand, `SNAT` requires a specific address to refer packets. `MASQUERADE` also differs from `SNAT` in that with `MASQUERADE`, connections are forgotten if the interface goes down.

SNAT Examples

- **MASQUERADE**

```
iptables -t nat -A POSTROUTING \
  -o eth0 -j MASQUERADE
```

- **SNAT**

```
iptables -t nat -A POSTROUTING \
  -j SNAT --to-source 1.2.3.45
```

redhat 26

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The SNAT example above can be extended The following are also possible:

Map to a specific source port:

```
iptables -t nat -A POSTROUTING -j SNAT -to-source 1.2.3.45:1234
```

Map using a range of possible IP addresses:

```
iptables -t nat -A POSTROUTING -j SNAT -to-source 1.2.3.45-1.2.3.55
```

Map using a range of possible ports:

```
iptables -t nat -A POSTROUTING -j SNAT -to-source 1.2.3.45:1234-1334
```

These last two specified by a range will randomly assign a value for that connection or *socket*

DNAT Examples

- **INBOUND**

```
iptables -t nat -A PREROUTING \
  -p tcp --dport 80 -j DNAT  \
--to-dest 192.168.0.20
```

- **OUTBOUND (with port redirection)**

```
iptables -t nat -A OUTPUT  \
-p tcp -j DNAT  \
--to-dest  192.168.0.200:3128
```

redhat 27

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Destination *nat* can be used to allow selective access to internal systems. In the example pictured on the slide any packets coming into the gateway on tcp port 80 will be forwarded to an internal web server at 192.168.0.20. Responses from the webserver will be `SNAT`ed on the way out.

The `DNAT` target can accept multiple `--to-dest` arguments. For example:

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80   \
     -j DNAT --to-dest  192.168.0.20 --to-desk 192.168.0.21   \
          --to-dest  192.168.0.22
```

would do round-robin load balancing. In other words, the first matching request would be sent to 192.168.0.20, the next to 192.168.21, the next to 192.168.0.22, then back to 192.168.0.20 and so on. Be warned that this can cause problems if dynamic or otherwise sever-centry content is being served.

Iptables can also change the destination port of a packet:

```
iptables -t nat -A PREROUTING -i -p tcp --dport 2201 -j DNAT  \
     --to-dest 192.168.0.1:22
```

This rule would cause any packets hitting this system's port 2201 to be forwarded to port 22 (*ssh*) of the internal system at 192.168.0.1.

Finally, destination *nat* can also be done for outbound packets as well as inbound. The following rule would cause all web traffic sent out from this machine to be transparently forwarded to a proxy server at port 3128 of the machine at 192.168.0.200:

```
iptables -t nat -A OUTPUT -p tcp --dport 80 -j DNAT   \
     --to-dest 192.168.0.200:3128
```

Rules Persistence

- **`iptables`** is not a daemon, but loads rules into memory and exits
- Rules are not persistent across reboot
  - **`service iptables save`** will store rules to **`/etc/sysconfig/iptables`**
  - System V management may be used, and is run before networking is configured
- Conflicts with **`ipchains`**

redhat 28

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

RHEL provides a convenient way of saving and restoring the state of all Netfilter rules through the use of a service management script called `iptables` This allows `iptables` to be operated like a service, even though no "listening" daemon is started.

Backwards compatibility with the earlier `ipchains` facility is provided through a system service of the same name. However, the `ipchains` and `iptables` methods of packet filtering are mutually exclusive. Each facility (`ipchains` and `iptables`) involves the use of its own set of kernel modules Each service (`ipchains` and `iptables`) depends on the corresponding kernel modules to operate

Example

- Sample **/etc/sysconfig/iptables**

```
*filter
:INPUT DROP [573:46163]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [641:68532]
-A INPUT -i lo  -j ACCEPT
-A INPUT -p tcp -m tcp --dport 143 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22  -j ACCEPT
-A INPUT -p tcp -m tcp --dport 25  -s 123.123.123.1 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 53  -j ACCEPT
-A INPUT -p udp -m udp --dport 53  -j ACCEPT
-A INPUT -p udp -m udp --dport 123 -s 123.123.123.1 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m tcp --dport 113 -j REJECT --reject-with tcp-reset
COMMIT
```

redhat 29

Rev Rh253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

This is a sample file saved by the command `service iptables save` It is a simplification for basic network protection, as might be used on a bastion host on the internet. Some points to consider:

- The default policy on the `INPUT` chain is `DROP`, while the others is `ACCEPT`
- There are no rules in the `FORWARD` chain, as is appropriate on a system which has only one interface and does no packet forwarding.
- There are no rules in the `OUTPUT` chain, allowing unrestricted outbound connections (notwithstanding other restrictions unrelated to packet filtering).
- There are no restrictions on packets arriving on the loopback interface.
- Incoming `SSH`, `DNS` and `IMAP` connections are allowed from any host
- Incoming `NTP` and `SMTP` connections are allowed, but only from one particular host
- Connections for the `IDENT` service (also known as the `AUTH` service) are rejected instead of being dropped, so that time-outs are avoided.

End of Unit 9

- Address questions
- Preparation for Lab 9
  - Goals
  - Sequences
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 30

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 9
Securing Networks

Estimated Duration: 30 minutes

Goal: Learn how to build a simple firewall with `iptables`.

Sequence 1: Applying simple packet filtering to a host

Scenario/Story:

A host ("`stationX`") requires protection by packet filtering. This host has only one network interface, so no packet forwarding is involved.

Tasks:

Work with a lab partner, and assign each of you with the roles of `stationX` (`192.168.0.X`), and `stationY` (`192.168.0.Y`).

1 On ***both*** `stationX` and `stationY`, clear any existing rules and custom chains, and confirm:

```
service iptables stop
iptables -nvL
```

2 Ensure the SSH service is running on `stationX`.

```
service sshd status
```

Ensure the HTTP service is running on `stationY`

```
service httpd status
```

3. Confirm exposed ports on `stationX` from `stationY`:

```
nmap stationX
```

Then confirm `stationX` can establish SSH connections to `stationY`. Note: use an earlier configured user account. If none exists, then create one for each system for this lab.
Do not use the root account! That's a poor practice!

4 On `stationX`, apply a new default policy on the INPUT chain of the filtering table:

```
iptables -P INPUT DROP
```

5. Allow local connections on `stationX`:

```
iptables -A INPUT -i lo -j ACCEPT
```

6. Allow return packets, again, on stationX:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
-j ACCEPT
```

7. On stationX, allow connections to the SSH service:

```
iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

8. Have your lab partner(stationY) confirm that only the SSH port is *not* exposed.

```
nmap -v -sF -P0 stationX
```

This may take some time to complete, but give it a couple of minutes to complete. Would merely an attempt to connect to stationX, from stationY using ssh satisfy the efficacy of the Netfilter configuration? Why? Why not?

9. Confirm that stationX can still establish connections to stationY

```
links http://stationY.example.com/
```

Confirm that stationX can still resolve host names.

```
dig stationY.example.com
```

10. Save your configuration:

```
service iptables save
```

And view the iptables configuration file just created:

```
cat /etc/sysconfig/iptables
```

11. Make the configuration persistent across reboots:

```
chkconfig iptables on
```

Deliverables:

1. Packet filter rules successfully limit connections to stationX for SSH services only
2. The HTTP service on stationY is available to stationX.

Clean-up:

Once you have verified that you have completed the lab successfully, remove the rules you just configured by entering the command:

```
service iptables stop; chkconfig iptables off
```

UNIT 10

Securing Services

1

Rev RH253 RHEL4-1

Objectives

- Analyze service activity
- Implement security policy
  - within the service
  - with *tcp_wrappers*
  - with **`xinetd`**

Agenda

- Inspect local network services
- Configure *tcp_wrappers*
- Secure **`xinetd`** managed services

System V Startup Control

- Determine which services are running from SysV startup scripts or **xinetd**
- **chkconfig --list**
  - shows which services should run.
  - cannot be used to get a list of running services
- Disable all unneeded services

redhat 4

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

A host is configured to provide a limited range of services. All other services should be disabled for security reasons. `chkconfig --list` provides a list of all services that are started via System V scripts or `xinetd`.

Example:

```
[root@server1 /]# chkconfig --list
anacron    0:off   1:off   2:on    3:on    4:on    5:on    6:off
apmd       0:off   1:off   2:on    3:on    4:on    5:on    6:off
[...]
xinetd based services:
        finger: on
        talk:   off
[...]
```

To disable a service, use `chkconfig <service-name> off`

Keep in mind that `chkconfig` only maintains the System V links in `/etc/rc.d/rcX.d` and the `xinetd` configuration. It does not start or stop the services or control the behavior of other services. `chkconfig` does not work with services started via `rc.local` or `inittab`.

Lib what

Ldd – display what are being used by program

Securing the Service

- Service-specific configuration
  - Daemons like **`httpd`** provide special security mechanisms
- General configuration
  - All programs linked with **`libwrap.so`** use common configuration files
  - Because **`xinetd`** is linked with **`libwrap.so`**, its services are effected
  - Checks for host and/or remote user name

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Many daemons provide their own set of security mechanisms These mechanisms are usually far more advanced than the simple functionality that *tcp_wrappers* provides On the other hand, is it much easier to use one central location for your service security policy. The `libwrap.so` library, more commonly referred to as *tcp_wrappers*, provides host-based access control lists for various network services The library, executables, and documentation are distributed in *tcp_wrappers-<version> rpm*.

When a client connects to a "tcp wrapped" service, the access control lists `/etc/hosts.allow` and `/etc/hosts.deny` are examined. The server will then either choose to accept or drop the connection, depending on the control list configuration. Policies can be specified for individual services, and are usually configured in terms of the client's IP address.

All processes controlled by `xinetd` automatically use `libwrap.so` The `tcpd` wrapper is no longer needed to protect these services

Many RHEL services are compiled with `libwrap.so` dynamically linked Use the command `ldd` to display a list of libraries dynamically linked to a particular program. Standalone daemons linked with `libwrap.so` include:

- `sendmail`
- `slapd`
- `sshd`
- `stunnel`
- `xinetd`
- `gdm`
- `gnome-session`
- `ORBit`
- `portmap`

tcp_wrappers Configuration

- Three stages of access checking
  - Is access explicitly permitted?
  - Otherwise, is access explicitly denied?
  - Otherwise, by default, permit access!
- Configuration stored in two files:
  - Permissions in `/etc/hosts.allow`
  - Denials in `/etc/hosts.deny`
- Basic syntax:

```
daemon_list: client_list [:options]
```

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`/etc/hosts.allow` and `/etc/hosts.deny` each have two or more colon-separated fields. The first field specifies a comma-delimited list of executable names (*not* service names), possibly with the wildcards `ALL` and `EXCEPT`. The second field contains a comma-separated list of client specifications, using IP address, hostname, "trailing dot" networks, "leading dot" domains, or network/netmask pairs. Again, the keywords `ALL` and `EXCEPT` are recognized.

When parsing the files, `libwrap.so` implements a "stop on first match" policy: as soon as a daemon/client configuration line is matched, the configuration line is implemented, and then no further action occurs. A matching line in `/etc/hosts.allow` would allow the connection. A matching line in `/etc/hosts.deny` would deny the connection. First, `/etc/hosts.allow` is examined. If access is not explicitly allowed, `/etc/hosts.deny` is examined. If access is not explicitly denied, the connection is allowed, *by fault of omission*: that is, the connection request meets no rule criterion.

Changes to the access files are effective immediately for all new connections

man hosts-access

Daemon Specification

- Daemon name:
  - Applications pass name of their executable
  - Multiple services can be specified
  - Use wildcard **ALL** to match all services
  - Limitations exist for certain daemons
- Advanced Syntax:

```
daemon@host: client_list ..
```

redhat 7

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The first field specifies a comma-delimited list of daemons *tcp_wrappers* usually takes argv[0] (the name of the process without path) as the daemon name

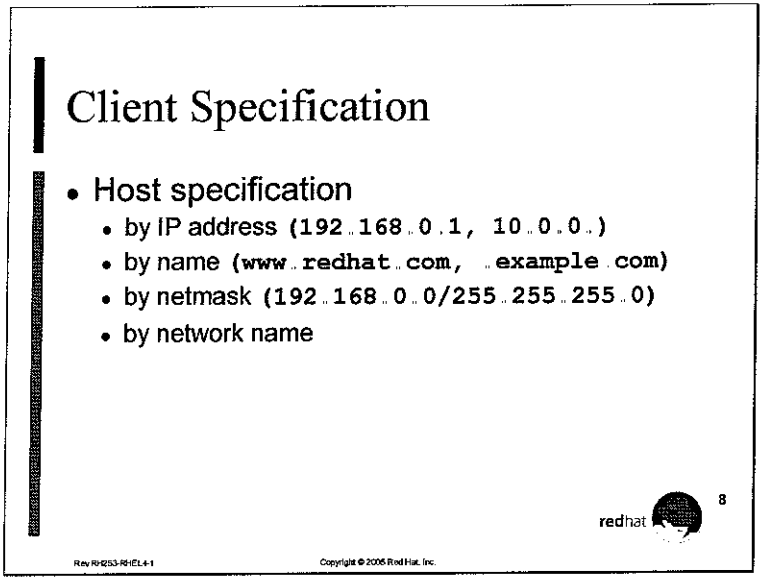
Examples:

```
in.telnetd:          192.168.0.1
sshd, gdm:           192.168.0.1
```

If your host has more than one network interface and you want to implement different policies for them, use the following syntax:

```
in.telnetd@192.168.0.254:  192.168.0.
in.telnetd@192.168.1.254:  192.168.1.
```

To block access to RPC-based services like NFS or NIS, block the underlying `portmap`. Unlike `xinetd`-managed services or `gdm`, for which changes to the access control lists take place immediately, it takes a brief interval of time for changes to rules concerning `portmap` to take effect. Remember that both NFS and NIS rely on the `portmap` daemon.



Client Specification

- by IP-address
  - Full or partial IP addresses
  - Rightmost components are treated as zero if omitted
  - Example: `192.168.1.` (all hosts within the class C network `192.168.1.0`)
- by network / netmask
  - Specify the complete network address plus netmask.
  - Netmask must be in the long format.
  - Example: `192.168.0.0/255.255.255.0`
- by host name
  - Performs a reverse lookup every time a client connects
  - Is not always supported
  - Example: `.example.com` (all hosts in the `example.com` domain)
- by network name
  - network names from `/etc/networks` or NIS.
  - Does not work together with usernames.
  - Example: `@mynetwork`

Advanced Syntax

- Wildcards
  - ALL, LOCAL
  - KNOWN, UNKNOWN, PARANOID
- **EXCEPT** operator
  - Can be used for client and service list
  - Can be nested

redhat 9

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Wildcards

ALL	always matches
LOCAL	all hosts without a dot in their name
UNKNOWN	all hosts or users that cannot be looked up
KNOWN	all hosts or users that can be determined
PARANOID	matches all hosts where lookup and reverse lookup do not match

EXCEPT Operator

The EXCEPT operator can be used in daemon and client lists to exclude some hosts from your match. It can be nested For example, consider the following:

```
hosts.allow
  sshd: ALL EXCEPT .cracker.org EXCEPT trusted.cracker.org

hosts.deny
  sshd: ALL
```

Because of the catch-all rule in hosts.deny this ruleset would allow only those who have been explicitly granted access to *ssh* into the system In hosts.allow we grant access to everyone except for hosts in the `cracker.org` domain, but to this rule we make an exception: We will allow the host `trusted.cracker.org` to *ssh* in despite the ban on `cracker.org`.

Options

- Syntax
  `daemon_list: client_list [:option1 :option2 ...]`
- **Spawn**
  - Can be used to start additional programs
  - Example: `in.telnetd: ALL : spawn echo \ "login attempt from %c to %s" | mail -s \ warning root`
  - Special expansions are available (%c %s)
- **DENY**
  - Can be used as an option in `hosts.allow`
  - `Example: ALL: ALL: DENY`

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The version of *tcp_wrappers* included with Red Hat Enterprise Linux supports the extended options documented in the `hosts_options`(5) man page. One of the more commonly used options is `spawn`, which closes the connection and executes an external program when the rule is matched.

The options utilize special `%` replacements. Common replacements are:

`%c`	client information (`user@host`)
`%s`	server information (`server@host`)
`%h`	the client's host name (IP address if name unavailable)
`%p`	server PID

The `%` expansions which are supported are documented in the `hosts_access`(5) man page. Use the command `man 5 hosts_access` so you do not get the `hosts_access`(3) man page by mistake.

Example

Consider the following example for the machine 192.168.0.254 on a class C network:

```
# /etc/hosts.allow
vsftpd:    192.168.0.
in.telnetd, portmap: 192.168.0.8

# /etc/hosts.deny
ALL: .cracker.org EXCEPT trusted.cracker.org
vsftpd, portmap: ALL
pop3d: 192.168.0. EXCEPT 192.168.0.4
```

redhat 11

Observations:

1. Only stations on the local network can `ftp` to the machine
2. Only `station8` could NFS-mount a directory from the machine (remember that NFS relies on `portmap`)
3. All hosts in `cracker.org`, except `trusted.cracker.org`, are denied access to any tcp-wrapped services
4. Only the host `192.168.0.4` is able use pop from the local network

Questions:

1. What stations from the local network can receive a telnet connection to this machine? ALL
2. Can machines in the cracker org network access the web server? No, tcp wrappers NOT used by Apache.
3. What services are available to a system from someother.net? What's "wrong" with these rules from the perspective of a security policy? only covers some services

A more realistic example would be to do something like the following using a mostly closed approach:

```
/etc/hosts.allow
     vsftpd : 192.168.0.
     in.telnetd,  sshd :  .example.com    192.168.2.5
/etc/hosts.deny
     ALL : ALL
```

The above example denies access to all services for everyone, except those which are explicitly allowed In this case *ftp* access is allowed to all hosts in the 192.168.0 subnet while *telnet* and *ssh* access are allowed by everyone in the `example.com` domain as well as host 192.168.2.5. This is a better a method for tightening down a system. It is a simpler, more direct approach and is much easier to maintain.

Securing **xinetd**-managed services

- **xinetd** provides its own set of access control functions
  - host-based
  - time-based
- *tcp_wrappers* is still used
  - **xinetd** is compiled with *libwrap* support
  - If **libwrap.so** allows the connection, then **xinetd** security configuration is evaluated

`xinetd` offers its own mechanism in addition to the security provided by *tcp_wrappers*. `xinetd` can implement host and simple time-based protection. Since `xinetd` is linked with `libwrap.so`, `/etc/hosts.allow` and `/etc/hosts.deny` are still authoritative.

First *tcp_wrappers* is checked. If access is granted, `xinetd`'s security directives are evaluated. If `xinetd` accepts the connection, then the requested service is called and *its* service-specific security is evaluated.

xinetd Access Control

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

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

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

`xinetd` access control is implemented with two directives, `only_from` and `no_access`. If neither directive is specified for a service, the service is available to anyone. If both are specified for a service, the one that is the "better match" for the address of the remote host is authoritative. In some cases, what the "better match" is may be ambiguous, so it may be a good idea to only use one of these directives when possible

Example:

```
service telnet
{
        disable         = yes
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        only_from       = 192.168.0.0/24
        no_access       = 192.168.0.1
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
}
```

This will block access to the `telnet` service to everyone except hosts from the `192.168.0.0/24` network, and of those, `192.168.0.1` will be denied access.

Host Patterns

- Host masks for **xinetd** may be:
  - numeric address (192.168.1.0)
    - rightmost zeros are treated as wildcards
  - network name (from /etc/networks)
  - hostname or domain (.domain.com)
  - IP address/netmask range (192.168.0.0/24)

redhat 14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

- Numeric Address
  - Full or partial IP addresses.
  - Rightmost zeros are treated as wildcards
  - Example: 192.168.1.0 (all machines within the class C network 192.168.1.0)
- By network name
  - network names from /etc/networks or NIS.
  - Does not work together with usernames.
  - Example: @mynetwork
- by host name
  - performs a reverse lookup every time a client connects.
  - Example: .example.com (all hosts in the example.com domain)
- IP-address / netmask
  - Must specify the complete network address **and** netmask
  - Example: 192.168.0.0/255.255.255.0 192.168.0.0/24

Advanced Security Options

- Access by time
  - Syntax: `access_times = 9:00-18:00`
  - `pam_time.so` for more advanced scenarios
- Number of simultaneous connections
  - Syntax: `per_source = 2`
  - Cannot exceed maximum instances

redhat 15

Rev RH253 RHEL4-1 Copyright © 2005 Red Hat, Inc.

`xinetd` features additional security functions. It is possible to restrict access to a service to certain times

Example:

```
service telnet
{
        access_times     = 08:00-16:00
        server           = /usr/sbin/in.telnetd
}
```

`per_source` limits the number of simultaneous connections per IP address This number may not exceed the total number of instances.

Example:

```
service telnet
{
        wait             = no
        instances        = 60
        per_source       = 5
        server           = /usr/sbin/in.telnetd
}
```

End of Unit 10

- Address questions
- Preparation for Lab 10
  - Goals
  - Sequences
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Lab 10
Securing Services

Estimated Duration: 30 minutes

Goal: To build skills in restricting user access to the system by using *tcp_wrappers* and `xinetd`.

Sequence 1: Restricting services for certain hosts

Scenario/Story:

Certain hosts on your network and from one specific network are compromised. To prevent possible attacks against your machine you decide to block access to certain critical services

Tasks:

Configure your system to fulfill the following criteria (you will need to work with a couple of other students for testing.) Note: you will need to install the *telnet-server* and *openssh-server* packages if you have not already done so.

1. `ssh` should be available to every host in the local subnet, but no other networks.
2. telnet should be available to exactly three of your neighbors, but no one else
3. No services are accessible from `cracker.org` (how might you find the appropriate IP address range?)

If you have difficulty fulfilling these criteria, one possible solution appears on the following page.

One Solution:

Assume that you are testing your configuration using neighbors at `stationX.example.com`, `stationY.example.com`, and `stationZ.example.com`.

1. Install *telnet-server*:

```
rpm -Uvh /mnt/server1/RedHat/RPMS/telnet-server*
chkconfig telnet on
```

Install *openssh-server*:

```
rpm -Uvh /mnt/server1/RedHat/RPMS/openssh-server*
chkconfig sshd on
service sshd start
```

2 `/etc/hosts.deny:`

```
sshd : ALL EXCEPT 192.168.0.
```

3. `/etc/xinetd.d/telnet:`

```
only_from = 192.168.0.X 192.168.0.Y 192.168.0.Z
```

4 `/etc/xinetd.conf:`

```
no_access = 192.168.1.0/24
```

In order to determine the IP addresses for cracker.org, you could use host and give the following command:

```
host -l cracker.org server1.example.com
```

This procedure queries the nameserver server1.example.com for the zone information associated with cracker.org. From the IPs returned, you could gather that they were all in the 192.168.1.0 subnet indicated above. If only life were so simple! Very often nameservers will have that particular capability (zone transfers) disabled for all hosts except slave servers. Providing the ability to list an entire domain makes life much too easy for people who wish to compromise your security.

UNIT 11

Securing Data

1

Rev RH253-RHEL4-1

Objectives

- Understand fundamental encryption protocols
- Describe encryption implementations in Red Hat Enterprise Linux
- Configure encryption services for common networking protocols

Agenda

- Introduction to data encryption
- Contrast encryption methods
- Red Hat encryption implementations
  - OpenSSH
  - RPM

redhat

3

Rev RH253-RHEL4-1

The Need For Encryption

- Susceptibility of unencrypted traffic
  - password/data sniffing
  - data manipulation
  - authentication manipulation
  - equivalent to mailing on postcards
- Insecure traditional protocols
  - `telnet`, `ftp`, `pop3`, etc. : insecure passwords
  - `sendmail`, `nfs`, etc.: insecure information
  - `rsh`, `rcp`, etc.: insecure authentication

redhat 4

Rev RI 253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

While early networking protocols have provided an indispensable infrastructure, secure authentication and privacy were often not part of their design. As a result, today's networking implementations often provide inadequate protection for the people who use them.

The mathematical field of number theory has provided cryptographic algorithms, protocols, and techniques which provide various forms of networking security, including secure authentication, assurance of data integrity, and privacy.

Cryptographic Building Blocks

- Random Numbers
- One Way Hashes
- Symmetric Algorithms
- Asymmetric (Public Key) Algorithms
- Public Key Infrastructures
- Digital Certificates
- Implementations:
  - `openssl, gpg`

redhat 5

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Encrypted communication protocols are often assembled from a collection of cryptographic techniques in a building block fashion. A protocol will usually provide a selection of one of several similar algorithms to fill a particular role. For example, public key encryption (used for email and other things) uses symmetric encryption for the bulk of the work. A selection of symmetric algorithms (such as 3DES, CAST5, Blowfish, and others) is available for use in this sort of encryption.

An understanding of these underlying cryptographic building blocks is important to understanding higher level protocols The various building blocks will be examined in the next few pages, including a discussion of commonly-used algorithms and examples of useful applications

Red Hat Enterprise Linux(RHEL) ships with two different implementations of cryptographic services:

- *Gnu Privacy Guard*, or `gpg`,which is used to implement file-based encryption: ensuring file integrity, encrypting email messages, etc.
- `openssl`, which provides cryptographic libraries that are used in conjunction with network communications, such as `ssl`-enabled services and `ssh` The `openssl` command provides a command line interface to many of the services provided by the *openssl* library.

Random Numbers

- Pseudo-Random Numbers and Entropy Sources
  - keyboard and mouse events
  - block device interrupts
- Kernel provides sources
  - **/dev/random**:
    - best source
    - blocks when entropy pool exhausted
  - **/dev/urandom**:
    - draws from entropy pool until depleted
    - falls back to pseudorandom generators
- **openssl rand [-base64]**

redhat 6

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Many cryptographic routines rely on random numbers, which are notoriously difficult to generate using computers. Generally, computers use "pseudo-random" random number generators, which are often seeded with integers. In order to improve random number generation, the kernel collects an "entropy pool" based on user events, such as mouse movements, keyboard hits, and disk I/O operations. The state of the entropy pool is maintained in the file `/var/lib/random-seed` between boots, as implemented by the `random` service script.

Two character devices allow access to the kernel's random number facilities.

- `/dev/random` will generate random numbers exclusively from the collected entropy pool. If the entropy pool is exhausted, the process will block until the pool is replenished (i.e., the mouse is moved - try `cat /dev/random`, then move your mouse).
- `/dev/urandom` will use the entropy pool until it is exhausted, and then resort to a fallback pseudo-random algorithm. While reads from `/dev/urandom` will not block, the random numbers are not as statistically well-formed as those provided by `/dev/random` (though they are still thought to be cryptographically strong.)

The *openssl* library also provides a random number generator, accessible through the `openssl rand` command. `openssl rand` also accepts a `-base64` option which uses base64 encoding to generate printable ASCII text.

One-Way Hashes

- Arbitrary data reduced to small "fingerprint"
  - arbitrary length input
  - fixed length output
  - If data changed, fingerprint changes ("collision free")
  - data cannot be regenerated from fingerprint ("one way")
- Common Algorithms
  - md2, md5 mdc2, rmd160 sha sha1
- Common Utilities
  - `md5sum [ --check ]`
  - `openssl, gpg`
  - `rpm -V`

redhat 7

Rev RH253-RHEL4-1

One-way hashes are used to confirm the integrity of information using simple strings known as *fingerprints*, or *message digests*. The contents of a given block of data (generally a file) are used to generate a small fingerprint, which can easily be recorded for later use. The integrity of the data can later be verified by regenerating the fingerprint, and comparing it to the stored form. If the data has changed in any way, the generated fingerprint is statistically guaranteed to change as well. Additionally, the original contents of the data cannot be deduced from the fingerprint, thus the name "one-way hash."

One-way hashes are commonly used to ensure file integrity. For example, the `rpm` command makes use of the MD5 hash to ensure the integrity of RPM package files and the files installed on your system.

One-way hashes are also used to secure passwords. A password entered at login is hashed and compared with the user's hashed password stored on the system. If the two hashes match, the password is correct.

Some command line utilities allow you to generate and confirm one way hashes directly, such as `sum` (CRC-32), `md5sum` (MD5), and `sha1sum`. Additionally, the *openssl* library provides access to several hashing algorithms (MD2, MD5, MDC2, RMD160, SHA, and SHA1).

Symmetric Encryption

- Based upon a single Key
  - used to both encrypt and decrypt
- Common Algorithms
  - DES, 3DES, Blowfish, RC2, RC4, RC5, IDEA, CAST5
- Common Utilities
  - **passwd** (modified DES)
  - **gpg** (3DES, CAST5, Blowfish)
  - **openssl**

redhat 8

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Symmetric encryption routines provide basic privacy. A single key (generally a short passphrase) is used to encrypt data in a way that the "plaintext" cannot be deduced from the encrypted "ciphertext." The same key is also used to decrypt the ciphertext back into plaintext.

Symmetric encryption algorithms can also be used to generate a one-way hash. This is used to secure passwords. The user's password, with some characters of random "salt" prepended, is used to encrypt a fixed block of text (generally all zeros). The result, with the same salt prepended, is stored as the "encrypted form" of the user's password. When the system wants to verify a user-supplied password, it prepends the appropriate salt (stored along with the encrypted password), then performs the encryption again. If the result matches the stored value, the user is authenticated. The user's password (the symmetric key) is never directly stored on the system.

By default, Red Hat Linux uses a strong MD5 based algorithm to secure passwords. This method allows passwords to be 256 characters long. A weaker method based on a DES variant ("crypt") is supported for compatibility purposes. The method used can be selected with `authconfig`, and both are implemented through PAM modules.

`gpg` and `openssl` also provide access to symmetric encryption protocols. Because of its speed, symmetric encryption is often used internally in asymmetric protocols.

Asymmetric Encryption I

- Based upon public/private key pair
  - What one key encrypts, the other decrypts
- Protocol I: Encryption without key synchronization
  - Recipient
    - generate public/private key pair: P and S
    - publish public key P guard private key S
  - Sender
    - encrypts message M with recipient public key
    - send P(M) to recipient
  - Recipient
    - decrypts with secret key to recover: M = S(P(M))

redhat 9

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Asymmetric, or *public key* encryption, relies on two complementary keys: the *public* and *private* keys What one key encrypts, the other key can decrypt The public key is made publicly available, generally through some sort of directory service The private key is carefully guarded, since the security of many protocols depends on the private key's owner having sole possession of it.

Cryptographic protocols are traditionally discussed using the fictional participants Alice and Bob. Assume that Bob has already generated a public/private key pair, and has published his public key. If Alice wanted to send Bob a private message, she would encrypt the message using Bob's public key, which she looked up from the appropriate key directory (or had Bob send to her) She would then send the encrypted message to Bob. Only Bob has the complementary private key to decrypt the message, so only Bob is privy to its contents.

This first of two basic protocols based on asymmetric encryption overcomes a fundamental problem with symmetric encryption: for two parties to communicate, they must first agree on a common symmetric key.

Asymmetric Encryption II

- Protocol II: Digital Signatures
  - Sender
    - generate public/private key pair: P and S
    - publish public key P, guard private key S
    - encrypt message M with private key S
    - send recipient S(M)
  - Recipient
    - decrypt with sender's public key to recover M = P(S(M))
- Combined Signature and Encryption
- Detached Signatures

redhat 10

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Asymmetric encryption also allows for digital signatures. Consider again the fictional participants Alice and Bob. Alice has a document she would like to send to Bob; she would like to assure Bob that the message came from her, and that no one has manipulated it in transit. Assuming she has generated an appropriate public/private key pair, she would encrypt the message with her private key. Upon receipt, Bob would decrypt the message with Alice's public key. If the message successfully decrypts, then Bob is assured that Alice sent the message, and he has received it unmodified.

A combination of asymmetric I and II is often employed. Alice would first sign (encrypt) the document with her secret key, and then encrypt it with Bob's public key. Upon receipt, Bob would perform the complementary operation, first decrypting the message with his private key, and then Alice's public key. The message has the advantages of both encrypted delivery and digital signatures.

A related protocol is known as a "detached signature". Rather than encrypting the document directly, Alice generates a hash of the document using a well known one-way hash algorithm such as MD5. She then encrypts the hash with her private key, and sends Bob both the original document and the encrypted hash. Bob can use the document directly, but if he would like to test its authenticity, he can generate his own hash of the document, and compare it against the encrypted hash that Alice sent to him.

Public Key Infrastructures

- Asymmetric encryption depends on public key integrity
- Two approaches discourage rogue public keys:
  - Publishing Key fingerprints
  - Public Key Infrastructure (PKI)
    - Distributed web of trust
    - Hierarchical Certificate Authorities
      - Digital Certificates

redhat 11

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The distribution of public keys is a fundamental weakness in asymmetric encryption schemes. In addition to Alice and Bob, we now introduce a malicious character, Mallet, whose wayward actions must be guarded against

When Alice wants to send a message to Bob, she looks up Bob's key from a public key directory. However, if Alice is unfamiliar with Bob, or more importantly, unfamiliar with the public key directory, she has no guarantee that the key she received, clearly labeled "Bob's key" in the directory, was actually generated by Bob. Mallet may have just as easily generated a public/private key pair, and, in an attempt to impersonate Bob, published his public key as "Bob's key"

Fingerprints

One approach for Bob to ensure that he is not impersonated is for Bob to generate a fingerprint (one way hash) of his public key, and include it copiously in all his communications (including it in his email signatures and on his business cards). When Alice obtains "Bob's Key" from a public key server, she can then compute her own fingerprint and compare it against the well known value

Public Key Infrastructures

A more secure approach would be for Bob to have a third party that Alice trusts sign his public key. When Alice obtains Bob's key, she can verify the signature of the trusted third party. Since she trusts the third party to sign keys only for legitimate owners, she can trust that the public key belongs to Bob. Such a signed public key is referred to as a Certificate, and the trusted third party is referred to as a Certificate Authority.

Digital Certificates

- Certificate Authorities
- Digital Certificate
  - Owner: Public Key and Identity
  - Issuer: Detached Signature and Identity
  - Period of Validity
- Types
  - Certificate Authority Certificates
  - Server Certificates
- Self-Signed certificates

redhat 12

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Certificate Authorities play the role of trusted third parties for the purpose of associating identities with their legitimate public keys. We now add the fictional character Trent to take the role of the trusted third party. Assume that Trent has generated his own public/private key pair, and signed his own public key. The self-signed public key is referred to as a "Certificate Signer's Certificate", or "Certificate Authority Certificate". Trent is now ready to act as a Certificate Authority (CA)

Bob is a retailer who would like (previously unknown) customers to be able to exchange sensitive information with him in a secure manner. He first generates a public/private key pair. He then convinces Trent of his identity, and provides Trent with his public key. Trent will combine his own identity, Bob's identity, Bob's public key, and a period of validity. He will then create a one-way hash of the information, and then encrypt this hash with his own private key, thus creating a detached digital signature. The combined information and detached signature compose a digital certificate, which Trent gives to Bob. Trent is referred to as the "issuer" of the certificate, and Bob as the "owner" of the certificate

Alice is a potential customer who would like secure communications with retailers such as Bob Knowing that she is going to want Trent to authenticate such retailers for her, she has already obtained Trent's "Certificate Authority Certificate" Standard web browsers, such as Netscape, come with such CA certificates built in

Alice now connects to Bob's web site. Bob will give Alice his certificate Alice will verify Bob's certificate by extracting the issuer's detached signature, and verifying it with the CA certificate that was already in her possession Effectively, Alice is trusting the CA's word that this is in fact Bob that she is talking to With Bob's identity established, Alice and Bob can now use public key encryption techniques to exchange information securely.

Generating Digital Certificates

- X.509 Certificate Format
- Generate a public/private key pair
- Define Identity
- Two Options:
  - Use Certificate Authority
    - generate signature request (*csr*)
    - send csr to CA
    - receive signature from CA
  - Self Signed Certificates
    - sign your own public key

redhat 13

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Currently, the Internet has standardized on a certificate format known as X.509. The X.509 format associates a public key with an identity, and is generally only valid for a given time period. The identity can be composed of arbitrary name/value pairs, but is currently generally composed the following minimum set: Country, Province or State, Organization Name, Common Name, and Email.

The first step is to generate a public/private key pair, which can be done with the `openssl` command (assuming a 1024-bit key):

```
openssl genrsa -out server1.key.pem 1024
```

Next, in order to generate a certificate signature request (csr), an identity must be established. The following command will read in the generated key pair, prompt for identity information, and generate a request:

```
openssl req -new -key server1.key.pem -out server1.csr.pem
```

The request would then be sent to a Certificate Authority, who would (we hope) take additional steps to verify the identity of the requester. The CA would then return to you a signed certificate, which could be saved in a file such as `server.crt.pem`.

Alternatively, a "self-signed" certificate could be generated, by using the `-x509` switch (note the change in name of the output file):

```
openssl req -new -key server1.key.pem -out server1.crt.pem -x509
```

For a self-signed certificate, the owner is also the issuer. Such certificates are appropriate for root level CA's, or in situations where encryption is desired by authenticated identity is not necessary.

Red Hat provides a Makefile to assist with the creation of certificates. From within the directory `/usr/share/ssl/certs`, we may generate a self-signed certificate by simply entering:

```
#make dovecot.pem
```

Running `make` without a "target," or argument will display a usage statement.

OpenSSH Overview

- OpenSSH replaces common, insecure network communication applications
- Provides user and token-based authentication
- Capable of tunneling insecure protocols through port forwarding
- System default configuration (client and server) resides in `/etc/ssh`

14

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The OpenSSH project provides support for the secure shell protocol, a mechanism for providing secure authentication, remote execution, and remote login capabilities In addition to the capabilities that the OpenSSH packages provide themselves, other packages, such as *rsync* and *rdist*, can use secure shell as their transport mechanism.

If a system does not need to provide remote shell access, but does need shell access to other hosts, then install the *openssh, openssl* and *openssh-clients* packages at a minimum, and add the *openssh-askpass* and *open-askpass-gnome* packages if you are running X. In order to provide remote `ssh` access to other systems, install the *openssh-server* RPM, which provides `sshd`. The □askpass□ packages are used in conjunction with `ssh-agent` in an X session.

Below is a list of the RPM packages and what they provide

```
openssh:                ssh-keygen, scp
openssl:                cryptographic libraries and routines required by openssh
openssh-clients:        ssh, slogin, ssh-agent, ssh-add, sftp
openssh-server:         sshd
openssh-askpass:        X11 passphrase dialogue
openssh-askpass-gnome:  GNOME passphrase dialogue
```

OpenSSH Authentication

- The **sshd** daemon can utilize several different authentication methods
  - password (sent securely)
  - RSA and DSA keys
  - Kerberos
  - s/key and SecureID
  - host authentication using system key pairs

redhat 15

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

sshd provides several options for authentication besides user-provided passwords Some of these, such as Kerberos, s/key, and SecureID, provide mechanisms for expiring tokens or one-time passwords Public/Private key pairs are another option, and these provide a means for logging into remote hosts or executing remote commands without needing to provide a password while at the same time using a secure mechanism for restricting this capability Key pairs may also be used to establish host identities for a somewhat more secure approach than hosts equiv.

Public/Private key authentication, ssh v2 compatible configuration (default in Red Hat Enterprise Linux):

Create a DSA key pair, setting an empty passphrase:

```
ssh-keygen -t dsa
```

Transfer `$HOME/.ssh/id_dsa.pub` to the remote host, appending it to `$HOME/.ssh/authorized_keys` if it already exists, or renaming it to `authorized_keys` if it does not.

`ssh` to the remote host without entering a password (or passphrase)

Public/Private key authentication, ssh v1 compatible configuration:

Create an RSA key pair, setting an empty passphrase:

```
ssh-keygen -t rsa
```

Transfer `$HOME/.ssh/id_rsa.pub` to the remote host, appending it to `$HOME/.ssh/authorized_keys` if it already exists, or renaming it to `authorized_keys` if it does not.

The OpenSSH Server

- Provides greater data security between networked systems
  - private/public key cryptography
  - compatible with earlier restricted-use commercial versions of SSH
- Implements host-based security through **`libwrap.so`**

redhat 16

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

The secure shell daemon (`sshd`) is installed with the *openssl, openssh,* and *openssh-server* RPMs

openssl provides certificate management -- a public/private-key technology -- and cryptographic libraries and routines required by *openssh*, the "common" or base RPM of the *openssh* software. *openssh-server* provides services for providing secure services to other systems via clients such as `ssh` and `scp`. The structure of secure connections is the agreement between the client and server through `ssh`. Once this agreement is reached, subsequent connections are "pre-authenticated," or "trusted "

To establish this trust relationship, the authentication process and data transmission between the client and server is encrypted. This encryption is based on an exchange of public keys. Data that is encrypted with these keys can only be decrypted with matching private keys held by only the other machine. The size of these keys is configurable, but should not be set so small that they are easily cracked

The *openssl* and *openssh* packages are compatible with proprietary versions of the software available from third-party vendors.

RHEL *openssh* software is compiled with `libwrap` support. If you are using `/etc/hosts.allow` and `/etc/hosts.deny` to provide additional host-based authentication, be sure to include a line similar to the following in `/etc/hosts.allow`:

```
sshd: IP_addr_of_host(s)
```

Service Profile: SSH

- Type: System V-managed service
- Packages: *openssh{,-clients,-server}*
- Daemons: `sshd`
- Scripts: `sshd`
- Ports: 22
- Configuration: `/etc/ssh/*, $HOME/.ssh`
- Related: *openssl, openssh-askpass, openssh-askpass-gnome*

OpenSSH Server Configuration

- SSHD configuration file
  - **/etc/ssh/sshd_config**
- Options to consider
  - **Protocol** ONLY USE PROTOCOL 2
  - **ListenAddress**
  - **PermitRootLogin**
  - **Banner**

redhat 18

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Several options are available to customize SSHD's operation. The configuraton file (`/etc/ssh/sshd_config`) shipped with RHEL has most of its entries "commented." These commented entries are the default configuration which might require modification to meet your security policy. Note, for example, that `X11Forwarding` has two entries. That commented is the software default. Red Hat installs a server that forwards X11 connections back to the connecting client.

SSHD is configured by default to listen for two protocols, SSH2(DSA) and the older SSH1(RSA). SSH1 has inherent security risks and is only provided for compatibility with older systems. It should be avoided whenever possible. SSH2 is the "preferred" protocol as configured by the "`Protocol 2,1`" option. To disable SSH1 protocol, change the option to read:

```
Protocol 2
```

By default SSHD is configured to listen on port 22. You can configure SSH to listen on multiple interfaces and multiple ports. This below configures a system to listen on tcp port 22 on a spcific interface:

```
ListenAddress 192.168.0.250:22
```

By default `root` is allowed to login via `ssh`. To disable this feature, set "`PermitRootLogin no`". Some administrators want to disable logging in as root with a password, but allowing root to log in using public-key authentication. To enforce this policy, set the option "`PermitRootLogin without-password`". A malicious user will not know that the password option has been disabled for `root`.

If you set "`PermitRootLogin forced-commands-only`", `root` can execute commands on a remote system using public key authentication only.

It is a good idea to warn users of your policies as they make a connection to your system, perhaps that their connection is being logged. The example below would display the contents of `/etc/issue.net` when a connection is made, before authentication starts.

```
Banner /etc/issue.net
```

The OpenSSH Client

- Secure shell sessions
  - `ssh hostname`
  - `ssh user@hostname`
  - `ssh hostname remote-command`
- Secure remote copy files and directories
  - `scp file user@host:remote-dir` - cblowtfish
  - `scp -r user@host:remote-dir localdir`
- Secure ftp provided by `sshd`
  - `sftp host`
  - `Sftp -C user@host`

redhat 19

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Several common, and notoriously insecure applications may be replaced with `ssh` and `scp`. To suggest that telnet and ftp are popular applications to the current network management administrator brings certain scorn. While well known, and at times, useful, these applications pass authentication information and data through similarly well known ports and through these ports as "clear text" or unencrypted packets. Once the "trust" has been established between participating `ssh` systems, all data is encrypted. What's more, the initial authentication is encrypted. Also, where `rsh`, `rcp` and `rlogin` are intended to obviate the authentication requirement (through the use of plain text files `$HOME/.rhosts` and `/etc/hosts.equiv` on the client and server), the data passed between such authenticated systems is unencrypted. With `ssh`-based communications, both authentication and application transmission is encrypted.

`ssh` is fairly simple to use. The user name that is sent to the server by default when one issues an ssh command is the local user name, but it is possible to override this default using the *user@* syntax or the *-l user* switch. When an `ssh` connection is made for the first time to a remote system, an entry will be appended to the local `~/.ssh/known_hosts` file that consists of the hostname plus the remote host's public `ssh` key. This key validates the host, in a sense: if the key changes, or if one has connected to a host that is not the original remote host, but now has that name, then `ssh` will exit with an error. `scp` uses a syntax much like that of `ssh`. It is possible to `scp` entire directories, and it is a more secure mechanism for transferring files than `ftp` or `rcp`. There are a few rules about the `-r` option that bear mention:

`scp -r localdir/ host:remote-dir` would copy the contents of *localdir* into *remote-dir*, but there would not be a *remote-dir/dir* (a copy from the remote host would work the same.)

`scp -r localdir host:remote-dir` would copy both *localdir* and its contents to *remote-dir* on the remote host if it existed, but would behave like the previous example if the remote directory did not exist (a copy from the remote host would work the same.)

`sftp -C user@host` would, using optional compression, log into *host* with a username of *user*, and present a password prompt and an interactive ftp-style session.

Note: system-wide default client configuration is set in `/etc/ssh/ssh_config`.

Protecting Your Keys

- **ssh-agent**
  - manages key passphrases
- **ssh-add**
  - collects key passphrases

redhat 20

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

An `ssh` public key (e.g. `~/.ssh/id_dsa.pub`) may be used for authentication in place of a simple plaintext password either by "exporting" your public key to a remote host, or by calling the *`ssh-agent`* at login. While this use of keys *is* convenient, it transfers the authentication vulnerability back to the "physical" key. `ssh` keys created *without* a passphrase are as vulnerable as the older, less secure "r-commands" (`rsh`, `rlogin`) if access to the exporting host or local account is compromised. Just as "forgetting" your house keys in a taxi-cab, so too a forgotten laptop or `ssh`-enabled PDA exposes your data and identity to someone *who is not you*!

Combining the use of an `ssh` passphrase with `ssh-agent` makes authentication both more secure and more convenient. Yes, you must enter your passphrase--use another key for another lock--to gain access to your things, but the passphrase must be typed only once per session. With the `ssh-add` program, you "import" your keys and their passphrases for that session. This method provides you a more secure "virtual key-ring" for that session only.

Example for use within display-managed X sessions

Generate your SSH DSA and, if needed RSA v2 keys ***with passphrases!***

```
ssh-keygen -t dsa
ssh-keygen -t rsa
```

The default Red Hat X11 configuration starts `ssh-agent` (see `/etc/X11/xinit/xinitrc` and `/etc/X11/xdm/Xsession`). With this agent "listening" through the wrapped session, prepare your account configuration to run the `/usr/bin/ssh-add` program at login (i.e. GNOME's Session Properties and Startup Programs tool). `ssh-add` will call the GUI tool `/usr/libexec/openssh/gnome-ssh-askpass` and prompt for the passphrase(s) that will be used by `ssh-agent` for this session.

Applications: RPM

- Two implementations of file integrity
- Installed Files
  - MD5 One-way hash
  - **rpm --verify *package_name*** (or **-V**)
- Distributed Package Files
  - GPG Public Key Signature
  - **RPM-GPG-KEY**
  - **rpm --checksig *package_file_name***

redhat 21

Rev RH253-RHEL4-1 Copyright © 2005 Red Hat, Inc.

Red Hat's `rpm` uses two encryption techniques to help maintain file integrity.

The first technique is implemented within the RPM database. When you install an RPM, an MD5 hash of every installed file is catalogued You can compare the files currently on your system against their original form (as installed from the RPM) with:

```
rpm --verify package_name
```

Files that differ from the their original RPM versions will be reported as having a differing MD5 fingerprint. This can be used as a debugging aid and security check for your trusted base of libraries and binaries

Second, Red Hat signs all RPM package files that it distributes with a `gpg`-generated private key. The complementary public key is distributed in the file `RPM-GPG-KEY`, found in the base directory of the distribution and `/usr/share/rhn`. This public key must be added to a user's `rpm` and `gpg` public key ring with:

```
gpg --import RPM-GPG-KEY
rpm --import RPM-GPG-KEY
```

Once added, a user can verify that a RPM package file was in fact packaged by Red Hat, and has not been modified during distribution, with `rpm --checksig package_file_name`. Checking the signature of package files obtained from the Internet is particularly advised

End of Unit 11

- Address questions
- Preparation for Lab 11
  - Goals
  - Scenario
  - Deliverables
- Please ask the instructor for assistance when needed

redhat 22

Lab 11
Securing Data

Estimated Duration: 45 minutes

Goal: Gain familiarity with encryption-based utilities

Sequence 1: Creating a certificate for IMAPS

Scenario/Story:

IMAP is a very useful protocol, but it lacks encryption. The *dovecot* package distributed with RHEL includes the ability to use IMAP over SSL This requires the creation of a PEM format certificate

Tasks:

1. Observe the credentials associated with the `dovecot.pem` certificate installed by the `dovecot` package.

```
cd /usr/share/ssl/certs
openssl x509 -subject -noout < dovecot.pem
```

```
subject= /C=-
/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizationalUnit/CN=local
host.localdomain/Email=root@localhost.localdomain
```

These are "bogus" values (output in a single line, despite the above) that are put in so that a key will be in place.

We need to generate a new key with our information.

2 Generate a new `dovecot.pem` certificate

```
rm dovecot.pem
make dovecot.pem
```

You will the be prompted for identity information. Enter the appropriate response for the first five, but don't enter anything yet for the sixth value For example:

```
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Tennessee
Locality Name (eg, city) [Newbury]:Knoxville
Organization Name (eg, company) [My Company Ltd]:Red Hat, Inc.
Organizational Unit Name (eg, section) []:GLS
```

The sixth prompt is for the Common Name of your server This should be the name your clients use to connect to your server.

Use the name of your imap server as the Common name: stationX.example.com.

```
Common Name(eg,your name or your server's hostname)[]:stationX.example.com
Email Address []:root@stationX.example.com
```

3 Observe the credentials of the newly created certificate

```
openssl x509 -subject -noout < dovecot.pem
```

```
subject= /C=US/ST=Tennessee/L=Knoxville/O=Red Hat,
Inc./OU=GLS/CN=stationX.example.com/Email=root@stationX.example.com
```

Move the `dovecot.pem` file to the `/usr/share/ssl/private` directory:

```
cp /usr/share/ssl/certs/dovecot.pem /usr/share/ssl/private
```

4. Enable IMAPS and test your server

Edit the `/etc/dovecot.conf` file and add `imap imaps` to the `protocols` line.

```
service dovecot start (or restart)
mutt -f {student@stationX.example.com}
```

Notice that you are presented with information about the machine you are connecting to. This allows you to verify the server's certificate manually since we are using a self-signed server certificate. Clients are usually unable to verify self signed certificates.

Deliverables:

1 You are using SSL as a transport for IMAP.

Sequence 2: Using SSH for Encrypted Communications

Scenario A

alice and bob are users on possibly different stations who would like to establish account equivalency In other words, alice would like to be able to access bob's account without needing to issue a password, and vice versa You will use ssh to provide this equivalency

This sequence will refer to stationX, *which contiaines the user* alice, *and* stationY, *which contains the user* bob. *When performing this lab, you will need to adjust the steps so the hostnames reflect your situation If you are working with a partner* stationX *and* stationY *would be the respective machine names of you and your partner If you are using a single machine, both hostnames may be replaced with* localhost.

1 Have root on bob's machine ensure that the sshd daemon is running

```
[root@stationY]$ service sshd start
[root@stationY]$ service sshd status
```

2 If alice knows bob's password, she can use ssh to access his account Note that all transactions with bob's account are encrypted, including the password exchange. As alice, run the following commands, supplying bob's password when appropriate.

```
[alice@stationX]$ ssh bob@stationY ls /tmp
[alice@stationX]$ ssh bob@stationY
[alice@stationX]$ scp bob@stationY:/etc/services .
[alice@stationX]$ scp -r bob@stationY:/etc/xinetd.d .
```

3 Assuming that alice and bob would like a more secure authentication scheme, have alice generate an ssh public/private key pair. Note that ssh-keygen should be invoked with the -t command-line switch, so that keys appropriate for the DSA algorithm are generated. Have alice examine her private key (id_dsa) and public key (id_dsa.pub).

```
[alice@stationX]$ ssh-keygen -t dsa
[alice@stationX]$ ls ~/.ssh
[alice@stationX]$ less ~/.ssh/id_dsa
[alice@stationX]$ less ~/.ssh/id_dsa.pub
```

Choose default options for key locations. Also choose a null passphrase by pressing <ENTER> *when prompted.*

4. Have alice mail bob a copy of her public key. Have bob save a copy of the public key as the file ~/.ssh/authorized_keys.

```
[alice@stationX]$ mail -s "my key" bob@stationY < ~/.ssh/id_dsa.pub
[bob@stationY]$ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/bob": 1 message 1 new
>N  1 alice@stationY.exa  Sun Sep 24 23:00  71/3947  "my key"
& w alices_key
"alices_key" [New file]
& q
```

```
[bob@stationY]$ mkdir ~/.ssh; chmod 700 ~/.ssh
[bob@stationY]$ cat alices_key >> ~/.ssh/authorized_keys
[bob@stationY]$ chmod 600 ~/.ssh/authorized_keys
```

5. Assuming all pieces are in place (i.e., bob now has a copy of alice's public key in his list of authorized keys), alice should now be able to access bob's account without needing to supply a password.

```
[alice@stationX]$ ssh bob@stationY id
501(bob) gid=501(bob) groups=501(bob)

[alice@stationX]$ ssh bob@stationY tar cvzf - /home/bob > \
  /tmp/bob.stationb.tgz
```

If things are not properly configured, ssh will fall back to password authentication, and prompt alice for a password. There are several steps you can take to help debug the situation. First, examine /var/log/messages and /var/log/secure on the server for helpful information. Second, use the -v command-line switch with the ssh client. This will output useful debugging information.

6. Perform the equivalent configuration for bob, so that he has access to alice's account

Scenario B

If someone should find a way to get alice's private key, they would be indistinguishable from alice. A passphrase associated with alice's key would prevent this confusion, but then "she" (you) must enter a passphrase for the commands above in Task #6. ssh-agent/ssh-add allows us to enter a passphrase only once per "session " (Note: the following

1. Have alice generate a new key, this time with a passphrase. When asked if you want to overwrite id_dsa answer yes. Use a passphrase of your own choosing... of at least 6 characters.

```
[alice@stationX]$ ssh-keygen -t dsa
```

2. Send bob a copy of your newly generated public key. Refer to Task #4 in Scenario A if you need help.

3. As alice, enter the commands in Scenario A, Task #5. Notice that now you are prompted for the passhrase for /home/alice/.ssh/id_dsa.

4. Start the ssh-agent running by typing the command:

```
[alice@stationX]$ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-BEKcxu4520/agent.4520; export SSH_AUTH_SOCK;
SSH_AGENT_PID=4521; export SSH_AGENT_PID;
echo Agent pid 4521;
[alice@stationX]$
```

If this command fails try: eval 'ssh-agent' or ssh-agent bash.

Provide your ssh-agent with the key for /home/alice/.ssh/id_dsa. You will be required to enter your passphrase so ssh-agent can access the key.

```
[alice@stationX]$ ssh-add
Enter passphrase for /home/alice/.ssh/id_dsa: <enter your passphrase>
Identity added: /home/alice/.ssh/id_dsa (/home/alice/.ssh/id_dsa)
```

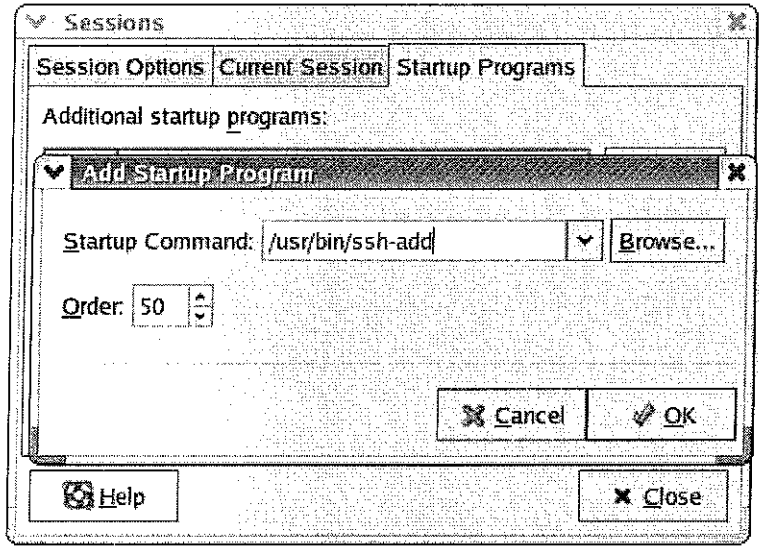
5 As `alice`, enter the commands in Scenario A, Task #5. Notice that now `ssh-agent` is able to reply to the challenge provided by `stationY`. No passphrase is required.

Note: if `alice` is logged in at a RHEL graphical user interface(GUI), then the previous Scenario will play out as described. However, if all she has is a text terminal session, then before executing `ssh-add` in Task #4, first run:

```
[alice@stationX]$ ssh-agent bash
```

This will "wrap" the bash shell with the `ssh-agent` utility.

What makes the RHEL GUI so SSH friendly? The desktop sessions are similarly "wrapped" as the `bash` shell is in the previous command example. In fact, as described in the Unit notes, by selecting from the "Red Hat Menu"->Preferences->More Preferences->Sessions->Start Up Programs, and selecting the "Add" button the following dialog is displayed. Enter the path to `ssh-add` as shown, then save and close the dialog.



Logout `alice` and login again. After the initial "splash screen" is displayed, a dialog will display asking for `alice`'s SSH passphrase. All SSH connections spawned from processes within this X11 session will be provided the passphrase... you can even skip Step #4 above.

Scenario C

`alice` has established public key authenticated shell access to `bob's` account She would now like to securely access the (plaintext) web server that is running on `bob's` machine.

1 Ensure that a webserver is running on `bob's` machine If not, as `root` on `bob's` machine, install and start the web server.

```
[alice@stationX]$ links http://stationY/
```

2. Using `ssh`, have `alice` connect to `bob's` account, and as a side effect, establish an encrypted tunnel between `alice's` port 12345 (or any other unused port) and `bob's` webserver (port 80).

```
[alice@stationX]$ ssh bob@stationY -L 12345:stationY:80
```

(and from another terminal)

```
[alice@stationX]$ links http://localhost:12345
```

`alice` *should see the same web page in both step 1 and step 2 In step 1, however, data traveled from the webserver to* `alice's links` *client in plaintext, and were subject to packet sniffing In step 2, packets traveled from the web server, through* `bob's ssh` *daemon, across the network in ciphertext to* `alice's ssh` *client, and then deciphered and passed to* `alice's links` *client*

Deliverables:

1. Users `bob` and `alice` are using SSH secured communications.

Appendix 1

Software Installation

In this course, you will be asked to configure software that may not be installed on the system. Below are a few simple methods to locate and install required packages.

Using NFS (may use the "*" wildcard in RPM name)

```
mkdir /mnt/server1
mount -t nfs -o ro server1:/var/ftp/pub/ /mnt/server1
rpm -Uvh /mnt/server1/RedHat/RPMS/<package-name>
```

Using FTP (may use the "*" wildcard in RPM name)

```
rpm -Uvh ftp://server1/pub/RedHat/RPMS/<package-name>
```

Using HTTP (may NOT use the "*" wildcard in RPM name)

```
rpm -Uvh http://server1/pub/RedHat/RPMS/<package-name>
```

Always verify that intended installations were successful, especially when using the wildcard character!