RH253 - Red Hat Enterprise Linux Network Services and Security Administration

Introduction - RH253: Network Services and Security Administration

Unit 1 - System Performance and Security

Unit 2 - System Service Access Controls

Unit 3 - Network Resource Access Controls

Unit 4 - Organizing Networked Systems

Unit 5 - Network File Sharing Services

Unit 6 - Web Services

Unit 7 - Electronic Mail Services

Unit 8 - Securing Data

Unit 9 - Account Management

End of Unit 9

Appendix A - Installing Software

Software Installation

Introduction

RH253: Network Services and Security Administration

RH253-RH253-RHEL5-en-1-20070325

Copyright

Welcome

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

Participant Introductions

Please introduce yourself to the rest of the class!

Red Hat Enterprise Linux

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

Red Hat Enterprise Linux Variants

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

Red Hat Network

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

Other Red Hat Supported Software

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

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

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*
Hostile Network	cracker.org	192.168.1.0/24
Hostile Server	server1.cracker.org	192.168.1.254
Hostile Stations	station*X*.cracker.org	192.168.1.*X*
Trusted Station	trusted.cracker.org	192.168.1.21

Objectives of RH253

- To become a system administrator who can setup a Red Hat Enterprise Linux server and configure common network services and implement a security policy at a basic level.

Audience and Prerequisites

- Audience: System administrators, consultants, and other IT professionals
- Prerequisites: RH033 *Red Hat Linux Essentials* and RH133 *Red Hat Linux System Administration*, or equivalent skills and experience. A working knowledge of Internet Protocol(IP) networking.

Unit 1

System Performance and Security

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Understand System Performance Security Goals
- Describe Security Domains
- Describe System Faults
- Explain System Fault Analysis Methods
- Explain Benefits of Maintaining System State
- Describe Networking Resource Concerns
- Describe Data Storage Resource Concerns
- Describe Processing Resource Concerns
- Describe Log File Analysis

System Resources as Services

- Computing infrastructure is comprised of roles
  - systems that serve
  - systems that request
- System infrastructure is comprised of roles
  - processes that serve
  - processes that request
- Processing infrastructure is comprised of roles
  - accounts that serve
  - accounts that request
- System resources, and their use, must be accounted for as policy of
securing
the system

Security in Principle

- Security Domains
  - Physical
  - Local
  - Remote
  - Personnel

Security in Practice

- By design, the system serves available resources
- By policy, the system preserves available resources
- Host only services you must, and only to those you must
  - "Do I need or know to host this?"
  - "Do they need or know to access this?"
  - "Is this consistent with past records of system behavior?"
  - "Have I applied all relevant security updates?"
- Monitor system resources for vulnerabilities and poor performance

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?

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

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

System Faults and Breaches

- Both effect system performance
- System performance is *the* security concern
  - a system fault yields an infrastructure void
  - an infrastructure void yields opportunity for alternative resource access
  - an opportunity for alternative resource access yields unaccountable resource access
  - an unaccountable resource access is a breach of security policy

Method of Fault Analysis

- Characterize the problem
- Reproduce the problem
- Find further information

Fault Analysis: Hypothesis

- Form a series of hypotheses
- Pick a hypothesis to check
- Test the hypothesis

Method of Fault Analysis, continued

- Note the results, then reform or test a new hypothesis if needed
- If the easier hypotheses yield no positive result, further characterize the problem

Fault Analysis: Gathering Data

- **strace *command***
- **tail -f *logfile***
- *.debug in syslog
- **--debug** option in application

Benefits of System Monitoring

- System performance and security may be maintained with regular system monitoring
- System monitoring includes:
  - Network monitoring and analysis
  - File system monitoring
  - Process monitoring
  - Log file analysis

Network Monitoring Utilities

- Network interfaces (**ip**)
  - Show what interfaces are available on a system
- Port scanners (**nmap**)
  - Show what services are available on a system
- Packet sniffers (**tcpdump**, **wireshark**)
  - Stores and analyzes all network traffic visible to the "sniffing" system

Networking, a Local view

- The **ip** utility
  - Called by initialization scripts
  - Greater capability than **ifconfig**
- Use **netstat -ntaupe** for a list of:
  - active network servers
  - established connections

Networking, a Remote view

- **nmap** reports active services on ports open to remote connection attempts
  - Advanced scanning options available
  - Offers remote OS detection
  - Scans on small or large subnets
- Do not use without written permission of the scanned system's admin!
- Graphical front-end available (**nmapfe**)

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**

Typical Problematic Permissions

- Files without known owners may indicate unauthorized access:
  - Locate files and directories with no user or group entries in the `/etc/passwd` file:

```
find / \( -nouser -o -nogroup \)
```

- Files/Directories with "other" write permission (o+w) may indicate a problem
  - Locate other-writable files with:

```
find / -type f -perm -002
```

  - Locate other-writable directories with:

```
find / -type d -perm -2
```

Monitoring Processes

- Monitor processes to determine:
  - Cause of decreased performance
  - If suspicious processes are executing
- Monitoring utilities
  - **top**
  - **gnome-system-monitor**
  - **sar**

Process Monitoring Utilities

- **top**
  - 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**

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

Managing Processes by Account

- Use PAM to set controls on account resource limits:
  - `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

System Log Files

- Why monitor log files?
- Which logs to monitor?
- Logging Services:
  - Many daemons send messages to **syslogd**
  - Kernel messages are handled by **klogd**

syslogd and klogd Configuration

- **syslogd** and **klogd** are configured in `/etc/syslog.conf`
- Syntax:
`facility.priority log_location`
- Example:
`mail.info /dev/tty8`

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

End of Unit 1

- Questions and Answers
- Summary
  - Address questions
  - Preparation for Lab
  - Goals
  - Sequences
  - Deliverables
  - Please ask the instructor for assistance when needed

Unit 2

System Service Access Controls

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Understand how services are managed
- Learn common traits among services
- Describe Service Configuration Resources
- Implement Access Controls
- SELinux Overview
- SELinux Management

System Resources Managed by init

- Services *listening* for serial protocol connections
  - a serial console
  - a modem
- Configured in `/etc/inittab`
- Calls the command **rc** to spawn initialization scripts
- Calls a script to start the X11 Display Manager
- Provides respawn capability

```
co:23:respawn:/sbin/agetty -f /etc/issue.serial 19200 ttyS1
```

System Initialization and Service Management

- Commonly referred to as "System V" or "SysV"
  - Many scripts organized by file system directory semantics
  - Resource services are either enabled or disabled
- Several configuration files are often used
- Most services start one or more processes
- Commands are "wrapped" by scripts
- Services are managed by these scripts, found in `/etc/init.d/`
- Examples:
  - **/etc/init.d/network status**
  - **service network status**

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
- Used for standalone and transient services
- Called by other applications, including **system-config-services**
- To list run level assignments, run **chkconfig --list**

Initialization Script Management

- Determine which services are configured to run a system boot
  - **chkconfig --list**
- Shows which services should run
- Only reports the status of the symbolic links it manages

xinetd Managed Services

- Transient services are managed by the **xinetd** service
- Incoming requests are brokered by **xinetd**
- Configuration files: `/etc/xinetd.conf`, `/etc/xinetd.d/`*`service`*
- Linked with `libwrap.so`
- Services controlled with **chkconfig**:

```
chkconfig tftp on
```

xinetd Default Controls

- Top-level configuration file

```
# /etc/xinetd.conf
defaults
{
        instances           = 60
        log_type            = SYSLOG authpriv
        log_on_success      = HOST PID
        log_on_failure      = HOST
        cps                 = 25 30
}
includedir /etc/xinetd.d
```


xinetd Service Configuration

- Service specific configuration
  - /etc/xinetd.d/*service*

/etc/xinetd.d/tftp:

```
# default: off

service tftp
{
    disable    = yes
    socket_type        = dgram
    protocol           = udp
    wait               = yes
    user               = root
    server             = /usr/sbin/in.tftpd
    server_args        = -c -s /tftpboot
    per_source         = 11
    cps                = 100 2
    flags              = IPv4
}
```

xinetd Access Controls

- 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`

Host Pattern Access Controls

- Host masks for **xinetd** may be:
  - numeric address (`192.168.1.0`)
  - network name (from `/etc/networks`)
  - hostname or domain (`.domain.com`)
  - IP address/netmask range (`192.168.0.0/24`)
- Number of simultaneous connections
  - Syntax: `per_source = 2`
  - Cannot exceed maximum instances

The /etc/sysconfig/ files

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

Service and Application Access Controls

- Service-specific configuration
  - Daemons like **httpd**, **smbd**, **squid**, etc. provide service-specific 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

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*]

Daemon Specification

- Daemon name:
  - Applications pass name of their executable
  - Multiple services can be specified
  - Use wildcard `ALL` to match all daemons
  - Limitations exist for certain daemons
- Advanced Syntax:

daemon@host: *client_list ...*

Client Specification

- Host specification
  - by IP address (`192.168.0.1`,`10.0.0.`)
  - by name (`www.redhat.com`, `.example.com`)
  - by netmask (`192.168.0.0/255.255.255.0`)
  - by network name

Macro Definitions

- Host name macros
  - LOCAL
  - KNOWN, UNKNOWN, PARANOID
- Host and service macro
  - ALL
- EXCEPT
  - Can be used for client and service list
  - Can be nested

Extended Options

- Syntax:

```
daemon_list: client_list [:opt1 :opt2...]
```

- spawn
  - Can be used to start additional programs
  - Special expansions are available (%c, %s)
- Example:

```
in.telnetd: ALL : spawn echo "login attempt from %c to %s" \
       | mail -s warning root
```

- DENY
  - Can be used as an option in `hosts.allow`
- Example:

```
ALL: ALL: DENY
```

A tcp_wrappers Example

```
# /etc/hosts.allow
vsftpd :     192.168.0.
in.telnetd, sshd :    .example.com 192.168.2.5

# /etc/hosts.deny
ALL : ALL
```

xinetd and tcp_wrappers

- **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

SELinux

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

SELinux, continued

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

SELinux: Targeted Policy

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

SELinux: Management

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

SELinux: semanage

- Some features controlled by **semanage**
- Recompiles small portions of the policy
- **semanage *function* -l**
- Most useful in high security environments

SELinux: File Types

- A managed service type is called its *domain*
- Allow rules in the policy define what file types a domain may access
- The policy is stored in a binary format, obscuring the rules from casual viewing
- Types can be viewed with **semanage**
  - **semanage fcontext -l**
- **public_content_t**

End of Unit 2

- Questions and Answers
- Summary
  - Address questions
  - Preparation for Lab
  - Goals
  - Sequences
  - Deliverables
  - Please ask the instructor for assistance when needed
  - SELinux Management

Unit 3

Network Resource Access Controls

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Describe IP and Routing
- Compare IPv4 and IPv6
- Describe IPv6 Features
- Understand Netfilter Architecture
- Learn to use the **iptables** command
- Understand Network Address Translation (NAT)

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

IPv6 Features
IP version 6

- Larger Addresses
  - 128-bit Addressing
  - Extended Address Hierarchy
- Flexible Header Format
  - Base header - 40 octets
  - Next Header field supports Optional Headers for current and future extensions
- More Support for Autoconfiguration
  - Link-Local Addressing
  - Router Advertisement Daemon
  - Dynamic Host Configuration Protocol version 6

Implementing IPv6

- Kernel **ipv6** module enables stateless autoconfiguration
- Additional configuration implemented by **/etc/rc.d/init.d/network** initialization script
  - `NETWORKING_IPV6=yes` in `/etc/sysconfig/network`
  - `IPV6INIT=yes` in `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*

IPv6: Dynamic Interface Configuration

- Two ways to dynamically configure IPv6 addresses:
  - Router Advertisement Daemon
    - Runs on (Linux) Default Gateway - **radvd**
    - Only specifies prefix and default gateway
    - Enabled with `IPV6_AUTOCONF=yes`
    - Interface ID automatically generated based on the MAC address of the system
  - DHCP version 6
    - **dhcp6s** supports more configuration options
    - Enabled with `DHCPV6C=yes`

IPv6: Static Interface Configuration

- `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
  - `IPV6ADDR=<ipv6_address>[/prefix_length]`
  - Device aliases unnecessary...
  - `IPV6ADDR_SECONDARIES=<ipv6_address>[/prefix_length] [...]`

IPv6: Routing Configuration

- Default Gateway
  - Dynamically from **radvd** or **dhcpv6s**
  - Manually specified in `/etc/sysconfig/network`
    - `IPV6_DEFAULTGW=<IPv6_address[% interface]>`
    - `IPV6_DEFAULTDEV=<interface>` - only valid on point-to-point interfaces
- Static Routes
  - Defined per interface in `/etc/sysconfig/network-scripts/route6-`*`ethX`*
    - Uses **ip -6 route add** syntax
    - `<ipv6_network/prefix> via <ipv6_routeaddress>`

tcp_wrappers and IPv6

- tcp_wrappers is IPv6 aware
  - When IPv6 is fully implemented throughout the domain, ensure tcp_wrappers rules include IPv6 addresses
- Example: preserving localhost connectivity, add to /etc/hosts.allow
  - ALL: [::1]

New and Modified Utilities

- ping6
- traceroute6
- tracepath6
- ip -6
- host -t AAAA *hostname6.domain6*

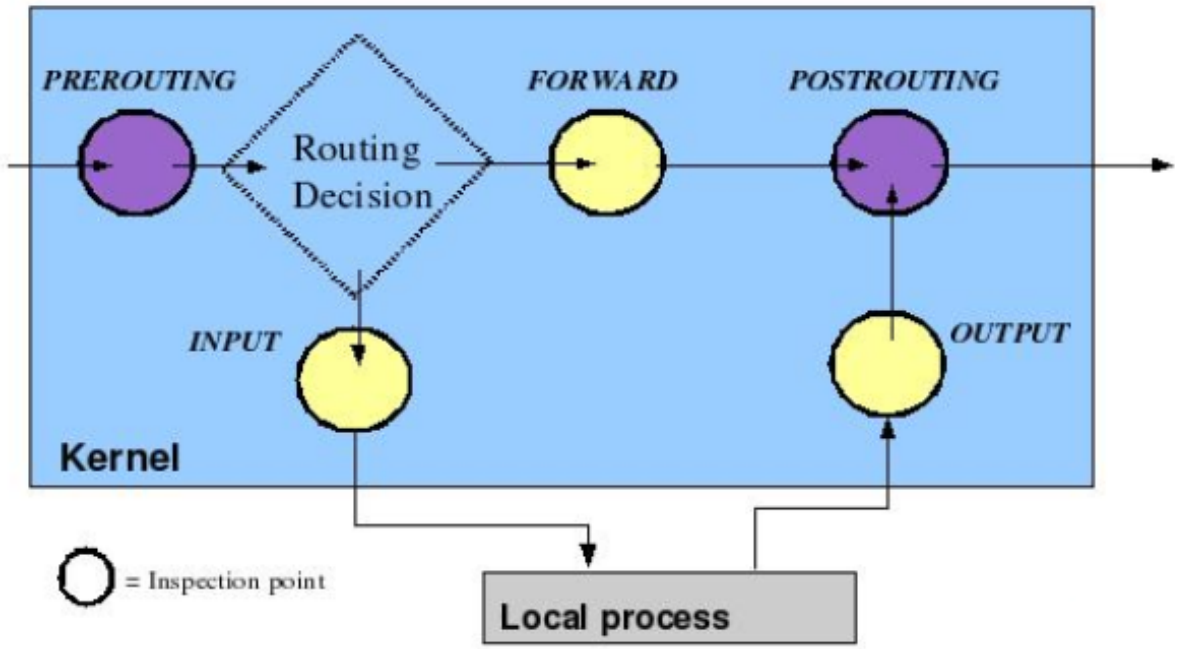
Netfilter Overview

- Filtering in the kernel: no daemon
- Asserts policies at layers 2, 3 & 4 of the OSI Reference Model
- Only inspects packet headers
- Consists of `netfilter` modules in kernel, and the **iptables** user-space software

Netfilter Tables and Chains

Filtering point	Table		
	filter	*nat*	*mangle*
INPUT	X		X
FORWARD	X		X
OUTPUT	X	X	X
PREROUTING		X	X
POSTROUTING		X	X

Netfilter Packet Flow



Rule Matching

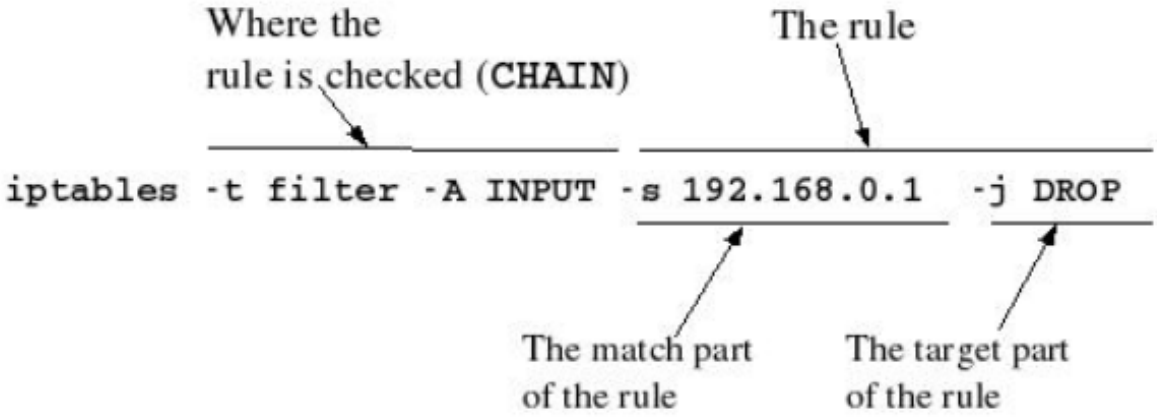
- Rules in ordered list
- Packets tested against each rule in turn
- On first match, the target is evaluated: usually exits the chain
- Rule may specify multiple criteria for match
- Every criterion in a specification must be met for the rule to match (logical AND)
- Chain policy applies if no match

Rule Targets

- Built-in targets: DROP, ACCEPT
- Extension targets: LOG, REJECT, custom chain
  - REJECT sends a notice returned to sender
  - LOG connects to system log kernel facility
  - LOG match does not exit the chain
- Target is optional, but no more than one per rule and defaults to the chain policy if absent

Simple Example

- An `INPUT` rule for the filter table:



Basic Chain Operations

- List rules in a chain or table (`-L` or `-vL`)
- Append a rule to the chain (`-A`)
- Insert a rule to the chain (`-I`)
  - `-I` *`CHAIN`* (inserts as the first rule)
  - `-I` *`CHAIN`* `3` (inserts as rule 3)
- Delete an individual rule (`-D`)
  - `-D` *`CHAIN`* `3` (deletes rule 3 of the chain)
  - `-D` *`CHAIN RULE`* (deletes rule explicitly)

Additional Chain Operations

- Assign chain policy (`-P` *`CHAIN TARGET`*)
  - `ACCEPT` (default, a built-in target)
  - `DROP` (a built-in target)
  - `REJECT` (not permitted, an extension target)
- Flush all rules of a chain (`-F`)
  - Does not flush the policy
- Zero byte and packet counters (`-Z` [*`CHAIN`*])
  - Useful for monitoring chain statistics
- Manage custom chains (`-N`, `-X`)
  - `-N` *`Your_Chain-Name`* (adds chain)
  - `-X` *`Your_Chain-Name`* (deletes chain)

Rules: General Considerations

- Mostly closed is appropriate
  - **iptables -P INPUT DROP** or
  - **iptables -A INPUT -j DROP**
  - **iptables -A INPUT -j REJECT**
- 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

Match Arguments

- Matches may be made by:
  - IP address, or host name
    - Warning: host names are resolved at the time of rule insertion
  - 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

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 more memory

Connection Tracking, continued

- Connection tracking modules
  - ip_conntrack_ftp
  - ip_conntrack_tftp
  - ip_nat_ftp
  - ip_nat_tftp (and others)
- `/etc/sysconfig/iptables-config`

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
```

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

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 --dport 80 -j DNAT \
     --to-dest 192.168.0.200:3128
```

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
```

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`(Ensure this file has proper SELinux context!)
  - System V management may be used, and is run before networking is configured

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
```

IPv6 and ip6tables

- Packet filtering for IPv6 traffic
- Provided by the `iptables-ipv6` package
- Rules stored in `/etc/sysconfig/ip6tables`
- Does not yet support:
  - **REJECT** target
  - **nat** table
  - connection tracking with the **state** module

End of Unit 3

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

Unit 4

Organizing Networked Systems

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Understand host name resolution and its impact on networked systems organization
- Use common utilities to explore and verify DNS server operation
- Describe the **Domain Name System (DNS)**
- Perform essential **BIND** DNS configuration
- DHCP Overview
- DHCP Configuration

Host Name Resolution

- Some name services provide mechanisms to translate host names into lower-layer addresses so that computers can communicate
  - Example: Name --> MAC address (link layer)
  - Example: Name --> IP address (network layer) --> MAC address (link layer)
- Common Host Name Services
  - Files (`/etc/hosts` and `/etc/networks`)
  - DNS
  - NIS
- Multiple client-side resolvers:
  - "stub"
  - **dig**
  - **host**
  - **nslookup**

The Stub Resolver

- Generic resolver library available to all applications
  - Provided through **gethostbyname()** and other glibc functions
  - Not capable of sophisticated access controls, such as packet signing or encryption
- Can query any name service supported by glibc
- Reads `/etc/nsswitch.conf` to determine the order in which to query name services, as shown here for the default configuration:

```
hosts: files dns
```

- The NIS domain name and the DNS domain name should usually be different to simplify troubleshooting and avoid name collisions

DNS-Specific Resolvers

- **host**
  - Never reads `/etc/nsswitch.conf`
  - By default, looks at both the **nameserver** and **search** lines in `/etc/resolv.conf`
  - Minimal output by default
- **dig**
  - Never reads `/etc/nsswitch.conf`
  - By default, looks only at the **nameserver** line in `/etc/resolv.conf`
  - Output is in RFC-standard zone file format, the format used by DNS servers, which makes **dig** particularly useful for exploring DNS resolution

Trace a DNS Query with dig

- **dig +trace redhat.com**
  - Reads `/etc/resolv.conf` to determine nameserver
  - Queries for root name servers
  - Chases referrals to find name records (answers)
  - See notes for sample output in case the training center's firewall restricts outbound DNS
- This is known as an *iterative* query
- Initial Observations:
  - Names are organized in an inverted tree with root (.) at top
  - The name hierarchy allows DNS to cross organizational boundaries
  - Names in records end with a dot when fully-qualified

Other Observations

- Answers in the previous trace are in the form of *resource records*
- Each resource record has five fields:
  - *domain* - the domain or subdomain being queried
  - *ttl* - how long the record should be cached, expressed in seconds
  - *class* - record classification (usually IN)
  - *type* - record type, such as A or NS
  - *rdata* - resource data to which the *domain* maps
- Conceptually, one queries against the *domain* (name), which is mapped to the *rdata* for an answer
- In the trace example,
  - The NS (name server) records are referrals
  - The A (address) record is the final answer and is the default query type for **dig**

Forward Lookups

- **dig redhat.com**
  - Attempts recursion first, as indicated by `rd` (recursion desired) in the *flags* section of the output: if the nameserver allows recursion, then the server finds the answer and returns the requested records to the client
  - If the nameserver does not allow recursion, then the server returns a referral to a top-level domain, which **dig** chases
- Observations
  - **dig**'s default query type is A; the rdata for an A record is an IPv4 address
  - Use **-t AAAA** to request IPv6 rdata
  - When successful, **dig** returns a status of NOERROR, an answer count, and also indicates which nameservers are authoritative for the name

Reverse Lookups

- **dig -x 209.132.177.50**
- Observations
  - The question section in the output shows that DNS reverses the octets of an address and appends `in-addr.arpa.` to fully qualify the domain part of the record
  - The answer section shows that DNS uses PTR (pointer) records for reverse lookups
  - Additionally, the rdata for a PTR record is a fully-qualified domain name

Mail Exchanger Lookups

- An MX record maps a domain to the fully-qualified domain name of a mail server
- **dig -t mx redhat.com**
- Observations
  - The rdata field is extended to include an additional piece of data called the *priority*
  - The priority can be thought of as a distance: networks prefer shorter distances
  - To avoid additional lookups, nameservers typically provide A records as additional responses to correspond with the FQDN's provided in the MX records
  - Together, an MX record and its associated A record resolve a domain's mail server

SOA Lookups

- An SOA record marks a server as a master authority
- **dig -t soa redhat.com**
- Initial Observations
  - The domain field is called the *origin*
  - The rdata field is extended to support additional data, explained on the next slide
  - There is typically only one master nameserver for a domain; it stores the master copy of its data
  - Other authoritative nameservers for the domain or zone are referred to as slaves; they synchronize their data from the master

SOA rdata

- Master nameserver's FQDN
- Contact email
- Serial number
- Refresh delay before checking serial number
- Retry interval for slave servers
- Expiration for records when the slave cannot contact its master(s)
- Minimum TTL for negative answers ("no such host")

Being Authoritative

- The SOA record merely indicates the master server for the origin (domain)
- A server is authoritative if it has:
  - Delegation from the parent domain: NS record plus A record
  - A local copy of the domain data, including the SOA record
- A nameserver that has the proper delegation but lacks domain data is called a *lame server*

The Everything Lookup

- **dig -t axfr example.com. @192.168.0.254**
- Observations
  - All records for the zone are transferred
  - Records reveal much inside knowledge of the network
  - Response is too big for UDP, so transfers use TCP
- Most servers restrict zone transfers to a select few hosts (usually the slave nameservers)
- Use this command from a slave to test permissions on the master

Exploring DNS with host

- For any of the following queries, add a **-v** option to see output in zone file format
- Trace: not available
- Delegation: **host -rt ns redhat.com**
- Force iterative: **host -r redhat.com**
- Reverse lookup: **host 209.132.177.50**
- MX lookup: **host -t mx redhat.com**
- SOA lookup: **host -t soa redhat.com**
- Zone transfer: **host -t axfr redhat.com 192.168.0.254** or **host -t ixfr=*serial* example.com. 192.168.0.254**

Transitioning to the Server

- Red Hat Enterprise Linux uses BIND, the **Berkely Internet Name Daemon**
- BIND is the most widely used DNS server on the Internet
  - A stable and reliable infrastructure on which to base a domain's name and IP address associations
  - The reference implementation for DNS RFC's
  - Runs in a chrooted environment

Service Profile: DNS

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

Access Control Profile: BIND

- Netfilter: tcp/udp ports 53 and 953 incoming; tcp/udp ephemeral ports outgoing
- TCP Wrappers: N/A

```
ldd `which named` | grep libwrap
strings `which named` | grep hosts
```

- Xinetd: N/A (**named** is a standalone daemon)
- PAM: N/A (no configuration in `/etc/pam.d/`)
- SELinux: yes - see notes
- App-specific controls: yes, discussed in later slides and in the ARM

```
/usr/share/doc/bind-*/arm/Bv9ARM.{html,pdf}
```

Getting Started with BIND

- Install packages
  - **bind** for core binaries
  - **bind-chroot** for security
  - **caching-nameserver** for an initial configuration
- Configure startup
  - **service named configtest**
  - **service named start**
  - **chkconfig named on**
- Proceed with essential **named** configuration

Essential named Configuration

- Configure the stub resolver
- Define access controls in `/etc/named.conf`
  - Declare client match lists
  - Server interfaces: `listen-on` and `listen-on-v6`
  - What queries should be allowed?
    - Iterative: `allow-query { match-list; };`
    - Recursive: `allow-recursion { match-list; };`
    - Transfers: `allow-transfer { match-list; };`
- Add data via zone files
- Test!

Configure the Stub Resolver

- On the nameserver:
  - Edit `/etc/resolv.conf` to specify `nameserver 127.0.0.1`
  - Edit `/etc/sysconfig/network-scripts/ifcfg-*` to specify `PEERDNS=no`
- Advantages:
  - Ensures consistent lookups for all applications
  - Simplifies access controls and troubleshooting
- Besides `/etc/resolv.conf`, where can an unprivileged user see what nameservers DHCP provides?

bind-chroot Package

- Installs a chroot environment under `/var/named/chroot`
- Moves existing config files into the chroot environment, replacing the original files with symlinks
- Updates `/etc/sysconfig/named` with a **named** option: `ROOTDIR=/var/named/chroot`
- Tips
  - Inspect `/etc/sysconfig/named` after installing `bind-chroot`
  - Run **ps -ef | grep named** after starting **named** to verify startup options

caching-nameserver Package

- Provides
  - named.caching-nameserver.conf
  - named.ca containing root server 'hints'
  - Forward and reverse lookup zone files for machine-local names and IP addresses (e.g., localhost. localdomain)
- Tips
  - Copy named.caching-nameserver.conf to named.conf
  - Change ownership to root:named
  - Edit named.conf
- The following slides describe essential access directives

Address Match List

- A semicolon-separated list of IP addresses or subnets used with security directives for host-based access control
- Format
  - IP address: 192.168.0.1
  - Trailing dot: 192.168.0.
  - CIDR: 192.168.0/24
  - Use a bang (!) to denote inversion
- A match list is checked in order, stopping on first match
- Example:

```
{ 192.168.0.1; 192.168.0.; !192.168.1.0/24; };
```

Access Control List (ACL)

- In its simplest form, an ACL assigns a name to an address match list
- Can generally be used in place of a match list (nesting is allowed!)
- Best practice is to define ACL's at the top of `/etc/named.conf`
- Example declarations

```
acl "trusted"     { 192.168.1.21; };
acl "classroom"   { 192.168.0.0/24; trusted; };
acl "cracker"     { 192.168.1.0/24; };
acl "mymasters"   { 192.168.0.254; };
acl "myaddresses" { 127.0.0.1; 192.168.0.1; };
```

Built-In ACL's

- BIND pre-defines four ACL's

```
none      - No IP address matches
any       - All IP addresses match
localhost - Any IP address of the name server matches
localnets - Directly-connected networks match
```

- What is the difference between the `localhost` built-in ACL and the `myaddresses` example on the previous page (assuming the server is multi-homed)?

Server Interfaces

- Option: `listen-on port 53 { match-list; };`
- Binds **named** to specific interfaces
- Example

```
listen-on port 53 { myaddresses; };
listen-on-v6 port 53 { ::1; };
```

- Restart and verify: **netstat -tulpn | grep named**
- Questions:
  - What if `listen-on` does *not* include `127.0.0.1`?
  - How might changing `listen-on-v6` to `::` (all IPv6 addresses) affect IPv4?
- Default: if `listen-on` is missing, **named** listens on all interfaces

Allowing Queries

- Option: `allow-query { match-list; };`
- Server provides both authoritative and cached answers to clients in match list
- Example:

```
allow-query { classroom; cracker; };
```

- Default: if `allow-query` is missing, **named** allows all

Allowing Recursion

- Option: `allow-recursion { match-list; };`
- Server chases referrals on behalf of clients in the match-list
- Example:

```
allow-recursion { classroom; !cracker; };
```

- Questions
  - What happens if 192.168.1.21 tries a recursive query?
  - What happens if 127.0.0.1 tries a recursive query?
- Default: if `allow-recursion` is missing, **named** allows all

Allowing Transfers

- Option: `allow-transfer { match-list; };`
- Clients in the match-list are allowed to act as slave servers
- Example:

```
allow-transfer { !cracker; classroom; };
```

- Questions
  - What happens if 192.168.1.21 tries a slave transfer?
  - What happens if 127.0.0.1 tries a slave transfer?
- Default: if `allow-transfer` is missing, **named** allows all

Modifying BIND Behavior

- Option: `forwarders { match-list; };`
- Modifier: `forward first | only;`
- Directs **named** to recursively query specified servers before or instead of chasing referrals
- Example:

```
forwarders { mymasters; };
forward only;
```

- How can you determine if `forwarders` is *required* ?
- If the `forward` modifier is missing, **named** assumes `first`

Access Controls: Putting it Together

- Sample `/etc/named.conf` with essential access control options:

```
// acl's make security directives easier to read
acl "myaddresses" { 127.0.0.1; 192.168.0.1; };
acl "trusted"     { 192.168.1.21; };
acl "classroom"   { 192.168.0.0/24; trusted; };
acl "cracker"     { 192.168.1.254; };
options {
    # bind to specific interfaces
    listen-on port 53 { myaddresses; };
    listen-on-v6 port 53 { ::1; };

    # make sure I can always query myself for troubleshooting
    allow-query { localhost; classroom; cracker; };
    allow-recursion { localhost; classroom; !cracker; };
    /* don't let cracker (even trusted) do zone transfers */
    allow-transfer { localhost; !cracker; classroom; };

    # use a recursive, upstream nameserver
    forwarders { 192.168.0.254; };
    forward only;
};
```

Slave Zone Declaration

```
zone "example.com" {
    type slave;
    masters { mymasters; };
    file "slaves/example.com.zone";
};
```

- Sample zone declaration directs the server to:
  - Act as an authoritative nameserver for `example.com`, where `example.com` is the origin as specified in the SOA record's *domain* field
  - Be a slave for this zone
  - Perform zone transfers (AXFR and IXFR) against the hosts in the `masters` option
  - Store the transferred data in `/var/named/chroot/var/named/slaves/example.com.zone`
- Reload **named** to automatically create the file

Master Zone Declaration

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

- Sample zone declaration directs the server to:
  - Act as an authoritative nameserver for `example.com`, where `example.com` is the origin as specified in the SOA record's *domain* field
  - Be a master for this zone
  - Read the master data from `/var/named/chroot/var/named/example.com.zone`
- Manually create the master file before reloading **named**

Zone File Creation

- Content of a zone file:
  - A collection of records, beginning with the SOA record
  - The @ symbol is a variable representing the zone's origin as specified in the `zone` declaration from `/etc/named.conf`
  - Comments are assembly-style (`;`)
- Precautions:
  - BIND appends the domain's origin to any name that is not properly dot-terminated
  - If the domain field is missing from a record, BIND uses the value from the previous record (Danger! What if another admin changes the record order?)
  - Remember to increment the serial number and reload **named** after modifying a zone file
- What DNS-specific resolver puts its output in zone file format?

Tips for Zone Files

- Shortcuts:
  - Do not start from scratch - copy an existing zone file installed by the `caching-nameserver` package
  - To save typing, put `$TTL 86400` as the first line of a zone file, then omit the TTL from individual records
  - BIND allows you to split multi-valued rdata across lines when enclosed within parentheses ()
- Choose a filename for your zone file that reflects the origin in some way

Testing

- Operation
  - Select one of **dig**, **host**, or **nslookup**, and use it expertly to verify the operation of your DNS server
  - Run **tail -f /var/log/messages** in a separate shell when restarting services
- Configuration
  - BIND will fail to start for syntax errors, so always run **service named configtest** after editing config files
  - **configtest** runs two syntax utilities against files specified in your configuration, but the utilities may be run separately against files outside your configuration

BIND Syntax Utilities

- **named-checkconf -t *ROOTDIR /path/to/named.conf***
  - Inspects `/etc/named.conf` by default (which will be the wrong file if the **-t** option is missing)
  - Example: **named-checkconf -t /var/named/chroot**
- **named-checkzone *origin /path/to/zonefile***
  - Inspects a specific zone configuration
  - Example:

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

Advanced BIND Topics

- Remote Name Daemon Control (**rndc**)
- Delegating Subdomains

Remote Name Daemon Control (rndc)

- Provides local and remote management of **named**
- The `bind-chroot` package configures **rndc**
  - Listens on the IPv4 and IPv6 loopbacks only
  - Reads key from `/etc/rndc.key`
  - If the key does not match, cannot start or stop the **named** service
  - No additional configuration is needed for a default, local install
- Example - flush the server's cache: **rndc flush**

Delegating Subdomains

- Steps
  - On the child, create a zone file to hold the subdomain's data
  - On the parent, add an NS record
  - On the parent, add an A record to complete the delegation
- Glue Records
  - If the child's canonical name is in the subdomain it manages, the A record is called a *glue* record

DHCP Overview

- DHCP: Dynamic Host Configuration Protocol, implemented via **dhcpd**
- **dhcpd** provides services to both DHCP and BOOTP IPv4 clients

Service Profile: DHCP

- Type: SystemV-managed service
- Package: `dhcp`
- Daemon: **/usr/sbin/dhcpd**
- Script: `/etc/init.d/dhcpd`
- Ports: 67 (bootps), 68 (bootpc)
- Configuration: `/etc/dhcpd.conf`, `/var/lib/dhcpd/dhcpd.leases`
- Related: **dhclient**, **dhcpv6_client**, **dhcpv6**

Configuring an IPv4 DHCP Server

- Configure the server in `/etc/dhcpd.conf`
- Sample configuration provided in `/usr/share/doc/dhcp-`*`version`*`/dhcpd.conf.sample`
- There must be at least one subnet block, and it must correspond with configured interfaces.
- Run **service dhcpd configtest** to check syntax

End of Unit 4

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

Unit 5

Network File Sharing Services

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Describe the FTP service
- Explain Network File Sharing
- Describe the NFS service
- Describe the Samba service
- Use client tools with each service

File Transfer Protocol(FTP)

- **vsftpd** - the default Red Hat Enterprise Linux ftp server
- No longer managed by **xinetd**
- Allows system, anonymous or virtual (FTP-only) user access
- The anonymous directory hierarchy is provided by the `vsftpd` RPM
- `/etc/vsftpd/vsftpd.conf` is the main configuration file

Service Profile: FTP

- Type: SystemV-managed service
- Package: `vsftpd`
- Daemon: **`/usr/sbin/vsftpd`**
- Script: `/etc/init.d/vsftpd`
- Ports: 21 (ftp), 20 (ftp-data)
- Configuration: `/etc/vsftpd/vsftpd.conf` `/etc/vsftpd.ftpusers` `/etc/pam.d/vsftpd`
- Log: `/var/log/xferlog`
- Related: `tcp_wrappers`, `ip_conntrack_ftp`, `ip_nat_ftp`

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** or **service nfs reload**
  - Shared directories are accessed through the **mount** command
  - The NFS server is an RPC service and thus requires **portmap**

Service Profile: NFS

- Type: System V-managed service
- Package: `nfs-utils`
- Daemons: **rpc.nfsd**, **rpc.lockd**, **rpciod**, **rpc.mountd**, **rpc.rquotad**, **rpc.statd**
- Scripts: `/etc/init.d/nfs`, `/etc/init.d/nfslock`
- Ports: 2049(nfsd), Others assigned by portmap (111)
- Configuration: `/etc/exports`
- Related: **portmap** (mandatory), `tcp_wrappers`

Port options for the Firewall

- **mountd**, **statd** and **lockd** can be forced to use a static port
- Set the `MOUNTD_PORT`, `STATD_PORT`, `LOCKD_TCPPORT` and `LOCKD_UDPPORT` variables in `/etc/sysconfig/nfs`

```
MOUNTD_PORT="4002"
STATD_PORT="4003"
LOCKD_TCPPORT="4004"
LOCKD_UDPPORT="4004"
```

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_squash)**
  - `root` mapped to UID 4294967294

NFS utilities

- exportfs -v
- showmount -e *hostname*
- rpcinfo -p *hostname*

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/init.d/netfs**
- **autofs** mounts NFS shares on demand and unmount them when idle

Samba services

- Four main services are provided:
  - authentication and authorization of users
  - file and printer sharing
  - name resolution
  - browsing (service announcements)
- Related
  - **smbclient** command-line access
  - Linux can mount a Samba share using the `cifs` or `smbfs` file system

Service Profile: SMB

- Type: System V-managed service
- Packages: `samba`, `samba-common`, `samba-client`
- Daemons: **/usr/sbin/nmbd**, **/usr/sbin/smbd**
- Script: `/etc/init.d/smb`
- Ports: [NetBIOS] 137(-ns), 138(-dgm), 139(-ssn), [SMB over TCP] 445(-ds)
- Configuration: `/etc/samba/*`
- Related: **system-config-samba**, **testparm**

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**
  - **samba-swat (`http://localhost:901`)**
  - Hand-editing `smb.conf` is recommended

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
- Use **testparm** to check the syntax of /etc/samba/smb.conf

Configuring File and Directory Sharing

- Shares should have their own [ ] section
  - Some options to use:
    - `public` - share can be accessed by `guest`
    - `browsable` - 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

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

Authentication Methods

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

Passwords

- Encrypted password considerations
  - Stored in `/etc/samba/smbpasswd`
  - Users added with **smbpasswd -a *user***
  - Users modified with **smbpasswd *user***
  - Users must have local accounts (or be translated to a local account through `/etc/samba/smbusers`), or implement **winbindd**, a separate service

Samba Syntax Utility

- **testparm** is used to check the syntax of `/etc/samba/smb.conf`
- Can check the allow/deny statements to verify that a host could access the server:

```
testparm /etc/samba/smb.conf station1.example.com 192.168.0.1
```

Samba Client Tools: smbclient

- Allows for simple view of shared services

```
smbclient -L hostname
```

- Can be used as an **ftp**-style file retrieval tool

```
[student@stationX]$ smbclient //machine/service
        > cd directory
        > get file
```

- `user%password` may be specified with `-U` or by setting and exporting the USER and PASSWD environment variables

Samba Client Tools: nmblookup

- List specific machine

```
nmblookup -U WINS_server -R name
```

- List all machines

```
nmblookup \*
```

Samba Clients Tools: mounts

- The SMB and CIFS file systems are supported by the Linux kernel
- Use **mount** to mount a Samba-shared resource:

```
mount -t cifs service mountpoint -o option1,option2
```

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, cifs as the file system type, and a user name.

```
//stationX/homes  /mnt/homes  cifs  username=bob,uid=bob  0 0
```

End of Unit 5

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

Unit 6

Web Services

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- 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

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

Service Profile: HTTPD

- Type: SystemV-managed service
- Packages: `httpd`, `httpd-devel`, `httpd-manual`
- Daemon: **/usr/sbin/httpd**
- Script: `/etc/init.d/httpd`
- Ports: 80(http), 443(https)
- Configuration: `/etc/httpd/*`, `/var/www/*`
- Related: `system-config-httpd`, `mod_ssl`

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/`

Apache Server Configuration

- Min and Max Spare Servers
- Log file configuration
- Host name lookup
- Modules
- Virtual Hosts
- user and group

Apache Namespace Configuration

- Specifying a directory for users' pages:

```
UserDir public_html
```

- MIME types configuration:

```
AddType application/x-httpd-php .phtml
AddType text/html .htm
```

- Declaring index files for directories:

```
DirectoryIndex index.html default.htm
```

Virtual Hosts

```
NameVirtualHost 192.168.0.100:80

<VirtualHost 192.168.0.100:80>
  ServerName   virt1.com
  DocumentRoot /virt1
</VirtualHost>
<VirtualHost 192.168.0.100:80>
  ServerName   virt2.com
  DocumentRoot /virt2
</VirtualHost>
```

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

Apache Syntax Utilities

- service httpd configtest
- apachectl configtest
- httpd -t
- Checks both `httpd.conf` and `ssl.conf`

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 -cm /etc/httpd/.htpasswd bob
htpasswd -m /etc/httpd/.htpasswd alice
```

.htaccess Advanced Example

```
AuthName      "Bob's Secret Stuff"
AuthType      basic
AuthUserFile  /var/www/html/.htpasswd
AuthGroupFile /var/www/html/.htgroup

<Limit GET>
require group staff
</Limit>

<Limit PUT POST>
require user bob
</Limit>
```

CGI

- 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`

Notable Apache Modules

- mod_perl
- mod_php
- mod_speling

Apache Encrypted Web Server

- Apache and SSL: *https* (port 443)
  - `mod_ssl`
  - `/etc/httpd/conf.d/ssl.conf`
- Encryption Configuration:
  - certificate: `/etc/pki/tls/certs/`*`your_host`*`.crt`
  - private key: `/etc/pki/tls/private/`*`your_host`*`.key`
- Certificate/key generation:
  - `/etc/pki/tls/certs/Makefile`
  - self-signed cert: **make testcert**
  - certificate signature request: **make certreq**

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

Service Profile: Squid

- Type: SystemV-managed service
- Package: `squid`
- Daemon: **/usr/sbin/squid**
- Script: `/etc/init.d/squid`
- Port: 3128(squid), (configurable)
- Configuration: `/etc/squid/*`

Useful parameters in /etc/squid/squid.conf

- http_port 3128
- cache_mem 8 MB
- cache_dir ufs /var/spool/squid 100 16 256
- acl all src 0.0.0.0/0.0.0.0
- acl localhost src 127.0.0.1/255.255.255.255
- http_access allow localhost
- http_access deny all

End of Unit 6

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

Unit 7

Electronic Mail Services

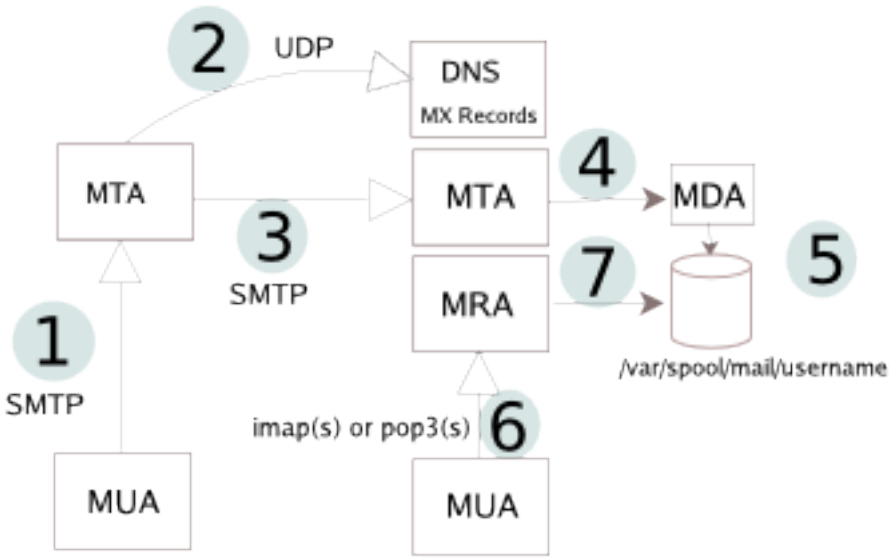
RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Understand electronic mail operation
- Use the alternatives system to select a mail server
- Perform basic configuration of a mail server
- Configure **Procmail**
- Configure Dovecot for encrypted and unencrypted protocols
- Debug email services

Essential Email Operation



Simple Mail Transport Protocol

- RFC-standard protocol for talking to MTA's
  - Almost always uses TCP port 25
  - Extended SMTP (ESMTP) provides enhanced features for MTA's
  - An MTA often uses Local Mail Transport Protocol (LMTP) to talk to itself
- Example MSP:

```
mail -vs 'Some Subject' student@stationX.example.com
```

- Use **telnet** to troubleshoot SMTP connections

SMTP Firewalls

- Network layer with **Netfilter** stateful inspection
  - Inbound and outbound *to* TCP port 25
- Application layer for relay protection
  - Internal MTA to which users connect for sending and receiving
  - DMZ-based outgoing *smart host* which relays mail *from* the internal MTA
  - DMZ-based inbound *mail hub* which relays mail *to* the internal MTA
  - Filtering rules within the DMZ MTA's or integrated applications (e.g., Spamassassin)

Mail Transport Agents

- Red Hat Enterprise Linux includes three MTA's
  - **Sendmail** (default MTA), **Postfix**, and **Exim**
- Common features
  - Support virtual hosting
  - Provide automatic retry for failed delivery and other error conditions
  - Interoperable with **Spamassassin**
- Default access control
  - Sendmail and Postfix have no setuid components
  - Listen on loopback only
  - Relaying is disabled

Service Profile: Sendmail

- Type: System V-managed service
- Packages: `sendmail`, `sendmail-cf`, `sendmail-doc`
- Daemon: **/usr/sbin/sendmail**
- Script: `/etc/init.d/sendmail`
- Port: 25 (smtp)
- Configuration: `/etc/mail/sendmail.mc`, `/etc/aliases`, and others
- Related: **procmail** (MDA), **spamassassin**, `tcp_wrappers`, `sendmail-doc`

Intro to Sendmail Configuration

- Red Hat uses and recommends the **m4** macro language
  - Use `dnl`**space** to comment a line within an `m4` macro file
- **service sendmail restart** uses `/etc/mail/Makefile`
  - Converts `/etc/mail/sendmail.mc` into `/etc/mail/sendmail.cf`
  - Rehashes various flat-file databases
  - **make** compares timestamps; **touch** a file to force a rebuild/rehash
- `sendmail-cf` is *not* installed by default
- The init script will *not* rebuild files unless `sendmail-cf` has been installed

Incoming Sendmail Configuration

- Modify `/etc/mail/sendmail.mc` to listen on all interfaces

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

- Add to `/etc/mail/local-host-names` each hostname by which the server may be referred
- Modify access control
  - Update `/etc/hosts.{allow,deny}`
  - Add an **Netfilter** rule to allow SMTP traffic
- Restart **sendmail**

Outgoing Sendmail Configuration

- Red Hat provides a default `/etc/mail/submit.cf`
  - rarely needs modification
  - enables sendmail to act as a client MSP
- To masquerade as a domain instead of a single host
  - Uncomment the following lines in `/etc/mail/sendmail.mc`

```
EXPOSED_USER(`root')dnl
FEATURE(masquerade_envelope)dnl
MASQUERADE_AS(`example.com')dnl
FEATURE(masquerade_entire_domain)dnl
```

  - These options work in conjunction with outbound address rewriting

Inbound Sendmail Aliases

- Local aliases: `/etc/aliases`
  - Programs must be linked under `/etc/smrsh` for the Sendmail Restricted Shell

```
fakename: realname
a-list:   fakename, otheruser
helpdesk: | mail2ticket
```

- Virtual aliases: `/etc/mail/virtusertable`

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


Outbound Address Rewriting

- Add the following lines to `/etc/mail/sendmail.mc`

```
FEATURE(genericstable)dnl
FEATURE(`always_add_domain')dnl
GENERICS_DOMAIN_FILE(`/etc/mail/local-host-names')dnl
```

- Create and populate `/etc/mail/genericstable`

```
paul@example.com     paul@otherexample.com
david@example.com    david.lastname@example.com
```

- Domains must be listed in `/etc/mail/local-host-names`
- Address rewriting occurs for SMTP and not LMTP

Sendmail SMTP Restrictions

1. Enable in `/etc/mail/sendmail.mc` using

```
FEATURE(`blacklist_recipients')dnl
```

2. Add restrictions in `/etc/mail/access`

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

- Use tags to indicate whether blacklisting affects sender, recipient, or MTA
- Untagged entries are deprecated in Sendmail

Sendmail Operation

- `/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
- **sendmail -q**
  - reprocess the email queue
- **tail -f /var/log/maillog**
  - View log in real-time

Using alternatives to Switch MTAs

- Overview of the `alternatives` system
  - displays or configures the preferred MTA and associated man pages based on a *`generic name`*
  - *`generic name`* is a link to a link in `/etc/alternatives/`
  - only the links in `/etc/alternatives/` are modified
- Switching between MTA's
  - Stop the current MTA and disable boot-time startup
  - **alternatives --config mta** and make a selection
  - Start the new MTA and enable boot-time startup
- Graphical interface: `system-switch-mail-gnome` package

Service Profile: Postfix

- Type: SystemV-managed service
- Package: `postfix`
- Daemons: `/usr/libexec/postfix/master` and others
- Script: `/etc/init.d/postfix`
- Port: 25 (smtp)
- Configuration: `/etc/postfix/main.cf` and others
- Related: `procmail`

Intro to Postfix Configuration

- `/etc/postfix/main.cf`
  - Well-commented key=value pairs, evaluated in the order in which they appear
  - White space at beginning of line is continuation character
  - Keys may be used as variables for subsequent key=value pairs

```
key1=value1
key2=$key1, value2
```

- **postconf**
  - Display defaults: **postconf -d**
  - Display current non-default settings: **postconf -n**
  - Modify `main.cf`: **postconf -e *key=value***
  - Show supported map types: **postconf -m**

Incoming Postfix Configuration

- Modify `/etc/postfix/main.cf`
  - Listen on all interfaces

    ```
    inet_interfaces = all
    ```

  - Specify each name and alias by which the server may be referred

    ```
    mydestination = $myhostname, localhost.$mydomain,
                    localhost, $mydomain
    ```

- Add **Netfilter** rules to allow SMTP traffic
- Restart **postfix**

Outgoing Postfix Configuration

- Red Hat provides a default `/etc/postfix/main.cf`
  - Enables Postfix to act as a client MSP
  - No further configuration needed for single host
  - Postfix automatically resolves local hostname and domain
- To masquerade as a domain

```
myorigin = $mydomain
masquerade_exceptions = root
```

Inbound Postfix Aliases

- Local aliases: `/etc/aliases` as in Sendmail
- Virtual aliases

1. Enable in `main.cf`

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

2. Define in `/etc/postfix/virtual` using the same format as Sendmail
3. Rehash the file: **postmap /etc/postfix/virtual**

Outbound Address Rewriting

1. Enable in `/etc/postfix/main.cf`

- `smtp` in the key name indicates SMTP only (not LMTP)

```
smtp_generic_maps = hash:/etc/postfix/generic
```

2. Define in `/etc/postfix/generic`

```
paul@example.com     paul@otherexample.com
david@example.com    david.lastname@example.com
```

3. Rehash the file: **postmap /etc/postfix/generic**

Postfix SMTP Restrictions

1. Create `/etc/postfix/access`

- untagged version of Sendmail `access` file
- rehash using **postmap /etc/postfix/access**

2. Edit `main.cf`

```
smtpd_TAG_restrictions =
   check_TAG_access hash:/etc/postfix/access, ...
```

- *TAG* is one of `sender`, `recipient`, or `client`
- Example:

```
smtpd_recipient_restrictions =
   check_recipient_access hash:/etc/postfix/access,
   permit_mynetworks, reject_unauth_destination
```

Postfix Operation

- `main.cf` settings
  - Server names: `mydestination` must contain server's name and aliases
  - Listening interfaces: `inet_interfaces = all`
  - Archive all messages: `always_bcc = address`
- View SMTP exchange: **mail -v user@domain.tld**
- View deferred messages: **postqueue -p**
- Flush deferred messages: **postqueue -f**
- Follow log: **tail -f /var/log/maillog**

Procmail, A Mail Delivery Agent

- 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
- Enabling Procmail
  - Sendmail: enabled by default
  - Postfix: modify `/etc/postfix/main.cf`

```
mailbox_command = /usr/bin/procmail
```

Procmail and Access Controls

- Initial controls
  - SELinux policy restricts mail utilies to certain directories
  - Procmail runs as *nobody*
  - Procmail is owned by the *mail* group
  - `/var/spool/mail` is writable only by root and the mail group
- Required: change the procmail binary to run setgid

```
chmod g+s $(which procmail)
```

Intro to Procmail Configuration

- Configuration files are processed in order if they exist

1. /etc/procmailrc
2. ~/.procmailrc

- Elements within a configuration file
  - Directives: `VERBOSE=yes`
  - Variables: `LOGFILE=/var/spool/mail/procmail.log`
  - Recipes
    - Begin with a "`:0`" line and flags
    - Zero or more match lines using regular expressions
    - One or more action lines

Sample Procmail Recipe

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

    :0:
    ADSL
}
```

- man pages: **procmailex**, **procmailrc**, **procmail**

Mail Retrieval Protocols

- Post Office Protocol
  - All data, including passwords, is passed in cleartext over TCP port 110
  - Use POP3s to provide SSL encryption of data over TCP port 995
- Internet Mail Access Protocol
  - All data, including passwords, is passed in cleartext over TCP port 143
  - Use IMAPs to provide SSL encryption of data over TCP port 993
- Dovecot supports POP3, POP3s, IMAP, and IMAPs

Service Profile: Dovecot

- Type: SystemV-managed service
- Package: `dovecot`
- Daemon: **/usr/sbin/dovecot**
- Script: `/etc/init.d/dovecot`
- Ports: 110 (pop), 995 (pop3s), 143 (imap), 993 (imaps)
- Configuration: `/etc/dovecot.conf`
- Related: `procmail`, `fetchmail`, `openssl`

Dovecot Configuration

- Listens on all IPv6 and IPv4 interfaces by default
- Specify protocols in `/etc/dovecot.conf`
  - `protocols = imap imaps pop3 pop3s`
- Make a private key and self-signed certificate before using SSL

1. Confirm system time to avoid date issues
2. Review `/etc/dovecot.conf` for key and cert locations
3. Run **make -C /etc/pki/tls/certs dovecot.pem**
   - Creates a single PEM file containing both the key and the cert
4. Copy the new PEM file to both locations

Verifying POP Operation

- Verify server operation
  - Graphical: **Thunderbird** and **Evolution**
  - Text-mode: **Mutt** and **Fetchmail**

```
mutt -f pop://user@server[:port]
mutt -f pops://user@server[:port]
```

  - Can also use **telnet** (POP3) or **openssl s_client** (POP3s)
    - Identify problems with certificate date or permissions

Verifying IMAP Operation

- Verifying server operation
  - Graphical: **Thunderbird** and **Evolution**
  - Text-mode: **Mutt** and **Fetchmail**

```
mutt -f imap://user@server[:port]
mutt -f imaps://user@server[:port]
```

  - Can also use **telnet** (IMAP) or **openssl s_client** (IMAPs)
    - Identify problems with certificate date or permissions

End of Unit 7

- Questions and Answers
- Summary
  - Inbound and outbound server configuration
  - Mail-related protocols: SMTP, IMAP, POP3
  - Preparation for Lab
  - Scenario
  - Deliverables
  - Please ask the instructor for assistance when needed

Unit 8

Securing Data

Objectives

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

- Understand fundamental encryption protocols
- Describe encryption implementations in Red Hat Enterprise Linux
- Configure encryption services for common networking protocols

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, NIS, etc.: insecure information
  - `rsh`, `rcp`, etc.: insecure authentication

Cryptographic Building Blocks

- Random Number Generator
- One Way Hashes
- Symmetric Algorithms
- Asymmetric (Public Key) Algorithms
- Public Key Infrastructures
- Digital Certificates
- Implementations:
  - `openssl`, `gpg`

Random Number Generator

- 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 pseudo-random generators
- **openssl rand [-base64]** ***num***

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
  - **sha1sum [--check]** ***file***
  - **md5sum [--check]** ***file***
  - **openssl**, **gpg**
  - **rpm -V**

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**

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))

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

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

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

Generating Digital Certificates

- X.509 Certificate Format
- Generate a public/private key pair and define identity
- Two Options:
  - Use a Certificate Authority
    - generate signature request (*csr*)
    - send csr to CA
    - receive signature from CA
  - Self Signed Certificates
    - sign your own public key

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/`

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

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`

Service Profile: SSH

- Type: System V-managed service
- Packages: `openssh`, `openssh-clients`, `openssh-server`
- Daemon: **/usr/sbin/sshd**
- Script: `/etc/init.d/sshd`
- Port: 22
- Configuration: `/etc/ssh/*`, `$HOME/.ssh/`
- Related: `openssl`, `openssh-askpass`, `openssh-askpass-gnome`, `tcp_wrappers`

OpenSSH Server Configuration

- SSHD configuration file
  - `/etc/ssh/sshd_config`
- Options to consider
  - `Protocol`
  - `ListenAddress`
  - `PermitRootLogin`
  - `Banner`

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***
  - **scp -r *user@host:remote-dir localdir***
- Secure ftp provided by **sshd**
  - **sftp *host***
  - **sftp -C *user@host***

Protecting Your Keys

- `ssh-add` -- collects key passphrases
- `ssh-agent` -- manages key passphrases

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 --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat***
  - **rpm --checksig *package_file_name*** (or **-K**)

End of Unit 8

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

Unit 9

Account Management

RH253-RH253-RHEL5-en-1-20070325

Objectives

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

- Understand the basics of authentication
- Understand the roles of NSS and PAM

User Accounts

- 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

Account Information (Name Service)

- 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`

Name Service Switch (NSS)

- 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`

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**

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

Pluggable Authentication Modules (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
- Documentation: `/usr/share/doc/pam-<version>/`

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

/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

/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
  - `include`returns the overall control value from tests configured in the file called

Example: /etc/pam.d/login File

```
auth        required     pam_securetty.so
auth        include      system_auth
account     required     pam_nologin.so
account     include      system_auth
password    include      system_auth
session     required     pam_selinux.so close
session     optional     pam_keyinit.so force revoke
session     include      system_auth
session     required     pam_loginuid.so
session     optional     pam_console.so
session     required     pam_selinux.so open
```


The system_auth file

- system-auth is widely used
  - Called by the *include* control-flag, not a module(i.e. pam_stack.so)
  - Contains standard authentication tests
  - Shared by many applications on the system
  - Allows easy, consistent management of standard system authentication

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

Network Authentication

- Central password management
  - `pam_krb5.so` (Kerberos V tickets)
  - `pam_ldap.so` (LDAP binds)
  - `pam_smb_auth.so` (old SMB authentication)
  - `pam_winbind.so` (SMB through winbindd)
- Some services use NSS/pam_unix.so
  - NIS, Hesiod, some LDAP configurations

auth Modules

- pam_securetty.so fails if logging in as root from a terminal not in /etc/securetty
- pam_nologin.so fails if the user is not root and the file /etc/nologin exists
- pam_listfile.so checks a characteristic of the authentication against a list in a file
  - A list of accounts can be allowed or denied

Password Security

- `pam_unix.so` MD5 password hashes
  - Makes password hashes harder to crack
- `pam_unix.so` shadow passwords
  - Makes password hashes visible only to `root`
  - Makes password aging available
- Other modules may support password aging mechanisms

Password Policy

- Password history
  - `pam_unix.so` with `remember=`*`N`* argument
- Password strength
  - `pam_cracklib.so`
  - `pam_passwdqc.so`
- Failed login monitoring
  - `pam_tally.so`

session Modules

- pam_limits.so enforces resource limits
  - Uses /etc/security/limits.conf
- pam_console.so sets permissions on local devices for console users
  - Can be used as an auth module as well
- pam_selinux.so helps set SELinux context
- pam_mkhomedir.so creates a home directory if it does not exist

Utilities and Authentication

- Local admin tools need authentication
  - **su**, **reboot**, **system-config-***, etc.
- `pam_rootok.so` passes if running as `root`
- `pam_timestamp.so` for **sudo**-like behavior
- `pam_xauth.so` forwards `xauth` cookies

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

End of Unit 9

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

Appendix A

Installing Software

Software Installation