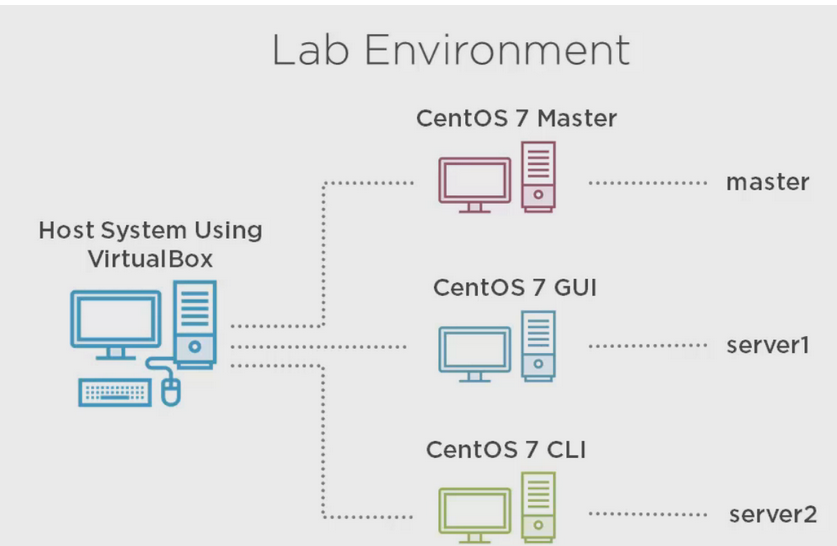
**Exercise Files**

[[https://www.evernote.com/images/file-generic.png](https://www.evernote.com/shard/s555/res/b9d49194-82bc-4ae7-bcb7-4f0e07606657/lfcs-linux-networking.zip)**lfcs-linux-networking.zip**2.2 MB](https://www.evernote.com/shard/s555/res/b9d49194-82bc-4ae7-bcb7-4f0e07606657/lfcs-linux-networking.zip)

PluralSight - LCFS Linux Networking Course Notes



**Introduction to Linux Networking and the Associated Certification Module Introduction**

Demo

ip vs ifconfig

**IP vs. IFCONFIG**

IFCONFIG

Ported from Unix

Displaying configuration and configuring the interface

ifconfig defined

Ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed. If no arguments are given, ifconfig displays the status of the currently active interfaces. If a single interface argument is given, it displays the status of the given interface only; if a single -a argument is given, it displays the status of all interfaces, even hose that are down. Otherwise, it configures an interface

ip defined

ip - show / manipulate routing, devices, policy routing and tunnels

$ ip addr

Default will show IP sub command, address and the show action

$ ip a

List the IP sub command only

$ ip r

List the root show command only

$ ip n

List the neighbor show command only

$ ip netns

List IP kernel network namespaces

Need root privileges to create network namespaces

$ sudo ip netns add development

Create the Development network stack

$ ip netns

List the development stack

**Configuring Hostnames Module Introduction**

Exam Objectives

Configure networking and hostname resolution statically or dynamically

Demo

Viewing the hostname

Using the hostnamectl

Host name resolution

**Viewing Hostname**

hostname

May include domain name or append domain name

Most of the time, will be included in teh command prompt of the distribution

domain name

DNS domain name the host belongs to

pretty name

Name that may not have any special characters. Good example is IPhone personal networks in IOS

$ echo $PS1

Display the format the prompt will be shown on terminal

$ hostname

Display the server name and the domain name but may only show the hostname element depending on the configuration

$ hostname -f

-f switch will extract the full hostname

$ uname -n

Return the hostname

$ hostnamectl

Return the full details including various other related information

**Configuring Hostnames**

$ su -

# hostname centos7

Change the hostname. This will be effective in a new shell

Once restart occurs, changes will be lost

# bash

New prompt will show up to update the hostname as original but the transient hostname is adjusted

# cat /etc/hostname

Need to update the file in order for changes to be permanent

For changes to be effective in transient and static, will need to update both the file from /etc/hostname and the hostname command

# hostnamectl set-hostname centos72.example.com

Configuring the hostname using this process is also effective and will effect both the transient and static

# bash

# hostname

# cat /etc/hostname

This verifies that the change has been effective in both file and hostname

# hostnamectl set-hostname "centos'72.example.com"

Using quotes help to assist in adding special characters to pretty names

Example of creating a pretty name

# hostnamectl

Verifies that the pretty name field has been added and is the indicated

# cat /etc/hostname

Verifies that the standard syntax is applied for the hostname

# cat /etc/machine-info

The prettyname is maintained in the machine-info file

PRETTY\_HOSTNAME field

**The Local Hosts File**

# hostnamectl set-hostname server1.example.com

Set the hostname but to original

# cat etc/hosts

# vim /etc/hosts

First line is the ipv4 didplayed by default

Second line with the colons is the ipv6

When adding host lines in this file standard formal is always IP Address, Qualified Name, Alias. Ensure that the lines adhere to the order as this is how it is processed

Edit the file to include aliases

192.168.56.105 nagios.example.com n

**DNS Name Resolution**

mDNS or Multi Cast DNS defined

multicast Domain Name System (mDNS) resolves host names to IP addresses within small networks that do not include a local name server. It is a zero-configuration service, using essentially the same programming interfaces, packet formats and operating semantics as the unicast Domain Name System (DNS). Although Stuart Cheshire designed mDNS to be stand-alone capable, it can work in concert with unicast DNS servers

# yum info avahi

Avahi is a system which facilitates service discovery on a local network -- this means that you can plug your laptop or computer into a network and instantly be able to view other people who you can chat with, find printers to print to or find files being shared. This kind of technology is already found in MacOS X (branded 'Rendezvous', 'Bonjour' and sometimes 'ZeroConf') and is very convenient

Referred to as plug and play

# getent hosts

This will list all the local hosts that are also in the /etc/hosts file

nsswitch.conf defined

The Name Service Switch (NSS) configuration file, /etc/nsswitch.conf, is used by the GNU C Library to determine the sources from which to obtain name-service information in a range of categories, and in what order. Each category of information is identified by a database name

# grep hosts /etc/nsswitch.conf

Look for the hosts database in the nsswitch.conf

resolv.conf defined

resolv.conf is the name of a computer file used in various operating systems to configure the system's Domain Name System (DNS) resolver. The file is a plain-text file usually created by the network administrator or by applications that manage the configuration tasks of the system

# cat /etc/resolv.conf

Display network services

dig defined

dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output

Testing responses and checking host name

# yum install -y bind-utils

RPM package for dig

# dig [www.pluralsight.com](http://www.pluralsight.com)

In the bottom of the results, the DNS server IP will be listed under"SERVER"

Queried the default server or local system

# dig [www.pluralsight.com](http://www.pluralsight.com) @8.8.8.8

Check record of another server using the @

Returns the server that being used, in this case, google

Query the remote system to debug name resolution

# dig +short [www.pluralsight.com](http://www.pluralsight.com) @8.8.8.8

+short will provide only the IP addresses

# dig+short [pluralsight.com](http://www.pluralsight.com) @8.8.8.8

Search by domain name

# dig+short [pluralsight.com](http://www.pluralsight.com) @8.8.8.8 MX

Search mail exchange

**Configuring Hostnames Module Recap**

Hostname

/etc/hostname

File stores hostname of the system

hostname -f

-f switch fully qualified domain name including DNS

hostname centos7.example.com

Set transient hostname only

Hostnamectl

hostnamectl

Display, Change both transient and persisistent name

hostnamectl set-hostname a1.test.com

Command to change the transient and persistent

/etc/machine-info

/etc/hosts

IP.....FQDN....ALIAS

Order that should be entered IP then Qualified domain name then alias

mDNS

Multicast DNS

Zeroconf / Plug and Play network discovery

Avahi package in CentOS

# yum install avahi

nsswitch.conf

/etc/nsswitch.conf

getent hosts

Controls the order of name resolution

DNS

/etc/resolv.conf

yum install -y bind-utils

dig [www.pluralsight.com](http://www.pluralsight.com) @8.8.8.8

**Configuring NTP Module Introduction**

Exam Objectives

Synchronize time using other network peers

Demo

Hardware time and system time

Using timedatectl

Implementing chronyd

Implementing ntpd

Three elements to Linux time

Hardware clock

System time

hwclock

**Working with Time**

# date

Lists date and time

# hwclock

List the hardware clock

# hwclock --systohc

Synch memory time to the hardware clock

# date --set="20160801 12:30"

Set the date and time for August 1 2016 12:30

# date

Verify the date has been updated

# hwclock --hctosys

# date

This will revert the clock back to the prior time

Clock in Linux will need to be set twice. to be permanent. Once in the system configurations and another in memory

**Using timedatctl**

# timedatectl

List the local time, system time, hardware time and RTC time

# date --set="20130801 12:30"

Adjust the date

# hwclock --systohc

Synch the time to the hardware clock

# timedatectl set-time "2016-08-01 11:47:00"

Set time using the timedatectl command. The command may prompt that the automatic time is synchronized

# timedatectl set-ntp false

# timedatectl set-time "2016-08-01 11:47:00"

This should allow the date to be changed because the value of ntp was set to false

# timedatectl

This will prompt that the synchronisation has been disabled and the time has been changed to the adjusted time

# date

Verify that the time has been changed

**Time Synchronization with Chrony**

# yum install -y chrony

Package necessary to install chrony daemon

# vi /etc/chrony.conf

Location of the configuration file

# systemctl enable chronyd

# systemctl start chronyd

# systemctl status chronyd

Enable and start the service and then verify that it has started correctly

# cronyc tracking

Checking if chrony is synchronised

# cronyc sources

List all servers that can be synchronised to

**Time Synchronization with NTPD**

# yum install -y ntp

Package that needs to be installed

# systemctl start ntpd

# systemctl enable ntpd

# systemctl start ntpd

Start. enable and test the ntpd service

# systemctl stop chronyd

Stop the chrony daemon if already started

# systemctl disable cnronyd.service

Disable the service at startup

# vi /etc/ntp.conf

Configuration file for the service

# ntpq defined

The ntpq utility program is used to monitor NTP daemon ntpd operations and determine performance. It uses the standard NTP mode 6 control message formats defined in Appendix B of the NTPv3 specification RFC1305. The same formats are used in NTPv4, although some of the variable names have changed and new ones added. The description on this page is for the NTPv4 variables. The program can be run either in interactive mode or controlled using command line arguments. Requests to read and write arbitrary variables can be assembled, with raw and pretty-printed output options being available. The ntpq can also obtain and print a list of peers in a common format by send- ing multiple queries to the server

# ntpq -p

Command to query the list of peers available

**Configuring NTP Module Recap**

hwclock

date

System time

hwclock

Hardware clock

hwclock --systohc

hwclock --hctosys

timedatectl set-date

chrony

yum install chrony

/etc/chrony.conf

chronyc tracking

See who we have that we can synchronize with

chronyc sources

NTPD

yum install ntp

/etc/ntp.conf

ntpq -p

Synchronization status

**Network Services Module Introduction**

Exam Objectives

Configure networking and hostname resolution statically or dynamically

Configure network services to start automatically at boot

Start, stop and check the status of network services

Demo

Display IP address Information

Working with the Network Manager

Working with the Network Service

Static interface configuration files

**Introduction to Network Services**

IP addresses

IPv4

32 Bit Dotted Decimal

192.168.1.1

127.0.0.1

Represents your local host

IPv6

128 bit Quad Hex

4 times the size of IPv4

Letters are allowed with hex

2001:ab2::a1

::1

**Configuring IP Addresses**

# ip addr show

List address and IP configuration

# ip a

Short method for displaying the IP configuration

# ip -4 a

List IP version 4 list

#ip -4 a s enp0s8

List only IPv4

List one interface by PCI bus type

Specifically looking at the PCI interface card on slot 8

#ip -6 a

List IPv6 types

Network mask

Half makes the number and half makes the node

# ip a s enp0s8

Show the IP address information for the enp0s8 device

# ip addr add 172.17.67.3/16 dev enp0s8

This will create an ip address with 16 bits for the IP and the rest for the mask

172.17.0.0 would be the IP

Node will be 67.3

Added to the enp0s8

# ip a s enp0s8

Will include the newly added IP address until the network service is restarted or turned off because it hasn't been configured in the system configuration

**Working with the Network Manager**

Default service that manages the network connectivity

Geared toward desktop and laptop Connection at home as well as connection at the work place

NetworkManager defined

The NetworkManager daemon attempts to make networking configuration and operation as painless and automatic as possible by managing the primary network connection and other network interfaces, like Ethernet, WiFi, and Mobile Broadband devices. NetworkManager will connect any network device when a connection for that device becomes available, unless that behavior is disabled. Information about networking is exported via a D-Bus inter- face to any interested application, providing a rich API with which to inspect and control network settings and operation. NetworkManager will execute scripts in the /etc/NetworkManager/dispatcher.d directory in alphabetical order in response to network events. Each script should be (a) a regular file, (b) owned by root, (c) not writable by group or other, (d) not set-uid, (e) and executable by the owner. Each script receives two arguments, the first being the interface name of the device just activated, and second an action

# systemctl status NetworkManager

check the status of the NetworkManager service

# nmcli connection

Double tab to see the options that can be performed

# nmcli connection

List out connections

# nmcli conn show enp0s3

Detail list of network enp0s3 device

# nmcli -p conn show enp0s3

-p switch for pretty format

# nmcli connection add con-name home ifname enp0s3 type ethernet ip4 192.168.0.99 gw4 192.168.0.1

Creating a new connection

con-name is connection name

ifname is interface that will be used

gw4 is gateway

# nmcli conn show

List the connections including the newly created

# ip a s enp0s3

List IP information for enp0s3 interface

# nmcli con down enp0s3

Place connection in down state

# nmcli con up home

start the newly created connection

# ip a s enp0s3

See details including the newly started connection

# nmcli con down home

Put down the connection

# nmcli con up enp0s3

Bring back up the proper connection

**Working with Network Configuration Files**

# systemctl status network

Checking the status of the network

# cd /etc/sysconfig/network-scripts/

Directory containing network configuration files

# cp ifcfg-enp0s3 ifcfg-enp0s9

Create a copy of the valid connection

# vi /etc/sysconfig/network-scripts/ifcfg-enp0s9

Edit the configuration file

Delete the UUID line

Change NAME and DEVICE to enp0s9

Add in Network Manager variable

NM\_CONTROLLED=no

Update BOOTPROTO variable

BOOTPROTO= "none"

Add IPADDR variable

IPADDR="192.168.56.104"

Add NETMASK variable

NETMASK="255.255.255.0"

Adjust the DEFROUTE variable

DEFROUTE="no"

Add DNS Server configurations under NETMASK variable

DNS1="8.8.8.8"

DNS2=8.8.4.4"

# nmcli con show

View connections including newly created connection

# nmcli con delete enp0s9

Delete the newly created connection

# ifdown enp0s8

Bring down the service for configuration to be updated

# ifup enp0s8

Bring up the adjusted connection

# systemctl stop NetworkManager.service

# systemctl start NetworkManager.service

Restart the service

# nmcli con show

List the updated connections managed by network manager

**Network Services Module Recap**

IP

ip addr show

ip -4 a s

ip -6 a s enp0s8

ip addr add 192.168.0.199/24 dev enp0s8

NetworkManager

systemctl status NetworkManager

nmcli con show enp0s8

Network

systemctl status network

/etc/sysconfig/network-scripts

NM\_CONTROLLED=no

**Routing with CentOS 7 Module Introduction**

Exam Objectives

Statically route IP traffic

Dynamically route IP traffic

Configure a system to perform Network Address Translation

Demo

Display route tables

Adding routes

Configuring a Linux system as router

Allowing access to the internet via NAT

**Reconfigure VM Networks**

Power off server1 and server 2 instances

On server 1 in the network section of the VirtualBox, disable all adapters except the host-only adapter

On server 2 in the network section of the VirtualBox, ensure a Bridge Adapter is connected for Adapter 1 and Adapter 2 is Host-only adapter

**Display Route Tables**

# ip a s

List connections which include the local and one link/ether

# ip route show

List that the local connections are OK but routing outside will be an issue

# ping [www.pluralsight.com](http://www.pluralsight.com)

This will result in unknown host

# route

List the route table

# netstat -r

Run command with -r switch for routing info

# netstat -rn

Run command to see the numbers instead of the names

# ip r

Same as ip route show

**Adding Routes**

# ip route add default via 192.168.56.104

Add the ip address of server2 and make it the default selection if no ip indicated

# ip r

This will now list the default route

This will provide a route to the outside world

The server being routed will need to be reconfigured

# cat /etc/resolv.conf

This will list the nameservers listed in the server

Could be possible to add the default gateway in this file

# cd /etc/sysconfig/network-scripts/

Navigate to the directory

# vi ifcfg-enp0s8

Edit the config file

Change the DEFROUTE variable

DEFROUTE="yes"

Add GATEWAY variable under NETMASK

GATEWAY="192.168.56.104"

**Enabling Routing**

# ip a s

List the internet configuration for server2

This will list the details of the connectivity of the external network and the details of the internal which includes the same network as server1

# ip r s

List the routing details on server2

# cat /proc/sys/net.ipv4/ip\_forward

List the current default configuration on server2

# vi /etc/sysctl.conf

Write through to the configuration file

insert a new line at the end of the file

net.ipv4.ip\_forward=1

# sysctl -p

Execute on server1

This command will help read into the file to verify changes have been reflected

# cat /proc/sys/net.ipv4/ip\_forward

Execute on server1

Using this command will also verify the contents

# ping 192.168.56.104

Execute on server1

Ping server2 from server1 to verify internal connectivity

# ping 172.18.10.156

Execute on server1

Ping server2 from server1 to verify the external connectivity

# ping 172.18.0.1

Execute on server1

This command will not work until we add IP to the routing table

**Enabling NAT**

# systemctl stop firewall.service

Stopping the firewall daemon on server2

# iptables -L

Verify using the iptables command on server2

# iptable

s -t nat -L

View prerouting and postrouting on server2

# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

Append and use the IP address in the public network for the users in the private network on server2

# iptables -t nat -L

In the list, the masquerade is added to the postrouting on server2

**Routing with CentOS 7 Module Recap**

Route Table

ip route show

route

netstat -nr

ip route add default via 192.168.56.104

Enable Routing

vi /etc/sysctl.conf

net.ipv4.ip\_forward=1

sysctl -p

NAT

iptables -t nat -A POSTROUTING \ -o nep0s3 -j MASQUERADE

**Firewalling and Firewalld Module Introduction**

Exam Objectives

Implement packet filtering

Configure firewall settings

Demo

Working with Firewall zones

Adding service and port rules to zone

Creating your own services

Masquerading with Firewalld

**Introduction and Zones**

# firewall-cmd --state

This is to ensure that the service is still off from last module on server2

# systemctl start firewalld.service

Turn the service back on in server2

# firewall-cmd --state

Verifying that the service is back on and running on server2

# firewall-cmd --get-default-zone

Display the default zone on server2

# firewall-cmd --get-active-zones

Check which zones have interfaces applied to them (server2)

# firewall-cmd --get-zones

List all the types of compatible zones (server2)

# firewall-cmd --permanent --zone=public --remove-interface=enp0s3

Remove the enp0s3 interface from firewall (server2)

# firewall-cmd --permanent --zone=public --remove-interface=enp0s8

Remove the enp0s8 interface from firewall (server2)

# firewall-cmd --permanent --zone=external --add-interface=enp0s3

Add enp0s3 interface as external zone (server2)

# firewall-cmd --permanent --zone=internal --add-interface=enp0s8

Add enp0s8 interface as internal zone (server2)

# systemctl restart firewalld.service

Restart the firewall to see the new adjustments (server2)

# firewall-cmd --get-active-zones

Check to see if the changes have been reflected (server2)

# firewall-cmd --set-default-zone=external

Change the default zone to external (server2)

# systemctl restart firewalld.service

Restart the service to ensure the changes have been reflected (server2)

# firewall-cmd --get-active-zones

Verify no changes (server2)

# firewall-cmd --get-default-zone

Verify that the change has been reflected (server2)

**Working with Services**

# firewall-cmd --list-all

Check the firewall settings on server2

# iptables -L

List the ip configurations on server2

# firewall-cmd --list-all --zone=internal

List the available internal zones server2

# firewall-cmd --list-all --zone=internal

List the available public zones server2

# firewall-cmd --list-all

Without specifying zone type default to external

# firewall-cmd --permanent --remove-service=ssh

Remove the ssh service permanently from the firewall allow list

# firewall-cmd --list-all

It still may be listed until the firewall settings have been reset

# firewall-cmd --reload

This will reload the service

# firewall-cmd --list-all

It still may be listed until the firewall settings have been reset or reloaded

# firewall-cmd --list-all --zone=internal

This will list the service with internal zone only

# firewall-cmd --permanent --remove-service=ipp-client --zone=internal

# firewall-cmd --reload

Remove the ipp client service permanently from internal zone and reload the firewall so changes are in effect

# firewall-cmd --list-all --zone=internal

The ipp-client service service has been removed permanently from the internal zone interface. This is a method of customizing the interfaces to specifically do what you need

# firewall-cmd --permanent --add-service=nfs --zone=internal

# firewall-cmd --permanent --add-service=glusterfs --zone=internal

These will add the nfs, glusterfs services permanently in interfaces with internal zones

# firewall-cmd --reload

# firewall-cmd --list-all --zone=internal

Reload and list the interfaces with the changes in effect

**Custom Services**

# firewall-cmd --list-services --zone=internal

List services related to internal zone interfaces

# firewall-cmd --remove-service={dhcpv6-client,mdns,samba-client} --zone=internal --permanent

Remove multiple services at one time for interfaces that have internal zones

# firewall-cmd --reload

# firewall-cmd --list-all --zone=internal

Reload and list the interfaces with the changes in effect

# firewall-cmd --list-services --zone=public

List the services for the interfaces that are public zone to see what we missed if we decided to mimic the public zone interface

# firewall-cmd --add-service={high-availability,http} --zone=internal --permanent

This command will add multiple services for the interfaces with internal zones

# firewall-cmd --reload

# firewall-cmd --list-services --zone=internal

Reload and list the services related to internal zone interfaces

# firewall-cmd --list-services --zone-public

Listing the services for he public zone interface for reference

# firewall-cmd --permanent --remove-service={dhcpv6-client,glusterfs,high-availability,http,nfs,ssh} --zone=public

Remove all services associated with the interfaces with public zones

# firewall-cmd --reload

# firewall-cmd --list-services --zone=public

Resulting in no response

# ls /usr/lib/firewalld/services

Directory where the service files are located

# firewall-cmd --permanent --new-service="puppet"

Add a new empty service called "puppet"

# cd /etc/firewalld/services/

# ls

Navigate to the directory where the created services are saved and list

# restorecon puppet.xml

Checking the SELinux contents for the puppet file

# chmod 640 puppet.xml

Ensure the rights are set correctly on the file

# vi puppet.xml

Add in details to file in between the <service> tags to complete the custom tag

<service>

<short>puppet</short>

<port protocol="tcp" port="443"/>

<port protocol="tcp" port="8140"/>

</service>

# firewall-cmd --permanent --add-service=puppet --zone=internal

Add the custom service to the interfaces with internal zones

# firewall-cmd --reload

# firewall-cmd --list-services --zone=internal

Reload and list the services, the newly created service will be listed

**NAT and Masquerade with Firewalld**

# firewall-cmd --List-all

Will show everything on the interfaces. Masquerade may still be shown as "yes" Everything from the private network may still be able to connect to the interface. To verify this, ping the server2 from server1

# firewall-cmd --permanent --remove-masquerade

Deactivate the Masquerade service

# firewall-cmd --reload

Reload the service and try to ping the server again. Should be 100% packet loss

# firewall-cmd --permanent --add-masquerade

# firewall-cmd --reload

This will activate the masquerade back to the default interface and reload

In the end, firewalld is a simpler way to manage the firewall settings and removes the need of manually configuring the iptables

**Firewalling with firewalld Module Recap**

Firewalld

The purpose of firewall is to manage and restrict a users access by the ports

# firewall-cmd --state

Check to see if it was running

# systemctl start firewalld

#firewall-cmd --get-default-zone

# firewall-cmd --get-active-zones

# firewall-cmd --get-zones

# firewall-cmd --set-default-zone=external

Zones

# firewall-cmd --permanent --zone=public --remove-interface={enp0s3,enp0s8}

# firewall-cmd --permanent --zone=internal --interface=enp0s8

# firewall-cmd --permanent zone=external --interface=enp0s3

Service

# firewall-cmd --permanent --remove-service=ssh

# firewall-cmd --permanent --add-service=http --zone=internal

#firewall-cmd --permanent --new-service=puppet

# vi /etc/firewalld/services/puppet.xml

Masquerade

# firewall-cmd --permanent --add-masquerade

**Using iptables Module Introduction**

Exam Objectives

Implement packet filtering

Configure firewall settings

Demo

Using the iptables toolset

Design a robust firewall

Installing the iptables configuration files

Editing the configuration

**Basic iptables Operations**

# iptables -L

List the default table and processes

Input is traffic coming into the machine

Output is traffic out f the machine

Forwarding usually involves the router

# iptables-save >fwoff

This command assists to save the current configuration in case you would like to revert back to the original settings

# iptables -A INPUT -i lo -j ACCEPT

-A switch to append

-i switch for interface

-j jump to switch

Append to the INPUT process

# iptables -A INPUT -m conntrack --ctstate ESTABLISHED, RELATED -j ACCEPT

-m switch for module

-ctstate for connector tracker status

This statement is ensuring that if we have traffic that if we have outgoing connection, it can come back, any established traffic is allowed back in

# iptables -A INPUT -p tcp --dport 22 -j ACCEPT

Ability to ssh into the machine for port 22 and allow port 22 traffic into the system

-p switch is for port

# iptables -L

List the iptables

# iptables -nvL

Provide a more thorough detail of the packets and whats allowed in and out

# iptables-save > fwon

Save the newly adjusted configurations

# iptables-restore < fwoff

Restore the configuration from the fwoff file

# iptables -L

List the old configurations before the adjustments

# iptables-restore < fwon

Restore the configuration from he fwon file

# iptables -L

List the newly adjusted configurations. This provides the ability to restore and bring back configurations

# iptables-restore < fwoff

# vi fwon

Add line below the information in the file after restoring the older file

-A INPUT -j DROP

# iptables-restore < fwon

# iptables -L

The DROP action should appear enabling only explicit traffic in

**Firewall Design**

#iptables -L

List the current firewall design

# iptables -F

Flush the current design. Clear the rules

# vi fwon

Edit the configurations add additional line underneath INPUT and above COMMIT

-A FORWARD -j DROP

# iptables -F

Flush the design again

# iptables-restore < fwon

Activate the configurations from fwon file

# iptables -L

The DROP rule will be applied to anything from the FORWARD chain

**Install the iptables Service**

# yum install -y iptables-services

Install the iptables service

# vi /etc/sysconfig/iptables

File that has the basic processes set out with the service

Close the fie no changes need to be made

# vi /etc/sysconfig/iptables-config

Change the IPTABLES\_SAVE\_ON\_STOP to "yes"

Change the IPTABLES\_SAVE\_ON\_RESTART to "yes"

# systemctl disable firewalld.service

# systemctl status firewalld.service

Stop the firewalld service and ensure that it has stopped

# systemctl enable iptables-service

# systemctl start iptables-service

Start and ensure that the new iptables-service has started and is set to enabled

# iptables -L

The new design with the iptables-service should be displayed

# iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT

-I switch to insert

the number signifies the order in how it is shown this being above the others

This command will add an input ACCEPT process to port 80

# iptables -L

Ensure the current configurations have not been altered

# systemctl restart iptables-service

A restart needs to be executed before the new process can be applied

# iptables -L

Traffic will still be the same

# vi /etc/sysconfig/iptables

The changes will be added to the file as shown

**Using iptables Module Recap**

Iptables

iptables -L

iptables -nvL

More detailed listing

iptables -F

iptables-save

iptables-restore

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

Policy

Keep policy to ACCEPT or DROP as your last rule

iptables -A INPUT -j DROP

Service

yum install iptables-services

/etc/sysconfig/iptables

/etc/sysconfig/iptables-config

**Methods to Tunnel Traffic Module Introduction**

Exam Objectives

Configure network traffic tunneling

Demo

Creating SSH tunnels

Installing openVPN server

Configuring openVPN server

Configuring openVPN client

**Creating SSH Tunnels**

Ensure you have a web server and w3m installed

# w3m localhost

List the contents of the webserver through Linux

Will require two virtual servers, one as the client and the other as the host

# ssh -f -L 8080:localhost:80 root@server2 -N

-f switch for process to be executed in the backend

-L switch for listening

Port will require to be above 1024 to be able to ssh without sudo privileges

This command will allow the user to listen to traffic from port 8080 and any outgoing traffic will be routed to the standard port 80 through ssh and encrypted

# w3m http://localhost:8080

The web page of server2 will be displayed

the entire http request that was sent is encrypted. The message is received by the server2 web server and will display the contents of web server2

# ps -ef | grep ssh

List processes that have ssh

# kill 12099

Terminate the process related to the ssh from root. The number associated will be dependant on the process type

# w3m http://localhost:8080

The page will not be able to load as the process has been killed

**Install openVPN Server**

# yum install epel-release

Package is required before installing openVPN

# yum install openvpn easy-rsa

Install both the openvpn and the easy rsa to quickly create certificates

# systemctl stop firewalld

Stop the firewall service if its started

# iptables -L

Result is no processes

# iptables -L -t nat

List the NAT specific processes which should also be empty. We still require masquerade and routing setup

# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

Add the routing and masquerade to local interface

**Configuring an openVPN Server**

# cp /usr/share/doc/openvpn-2.3.10/sample/sample-config-files/server.conf /etc/openvpn/server.conf

Copy the server configuration file to the local machine

# vi /etc/openvpn/server.conf

Remove the ";" key from the beginning of the redirect configuration. Configure it so that the user will have to be directed to our default gateway before reaching the vpn server

push "redirect-gateway def1 bypass-dhcp"

In the "DNS Servers" section of the configuration file, adjust the commented lines to the google public DNS servers and activate them by removing the ";" in the beginning

push "dhcp-option DNS 8.8.8.8"

push "dhcp-option DNS 8.8.4.4"

In the user and group section of the configuration file, un-comment the nobody user and group

user nobody

group nobody

# mkdir -p /etc/openvpn/easy-rsa/keys

Create directories and parent directories for the keys

# cp -rf /usr/share/easy-rsa/2.0/\* /etc/openvpn/easy-rsa/

Copy all required scripts into the parent directory of /keys

# vi /etc/openvpn/easy-rsa/vars

Setup variables for when generating keys

In X509 Subject Field section, change the KEY\_NAME to "server"

export KEY\_NAME="server"

Change the KEY\_CN to the host name of the created vpn server

# export KEY\_CN="server2.example.com"

# cd /etc/openvpn/easy-rsa/

# cp openssl-1.0.0.0.cnf openssl.cnf

Create and copy contents of the openssl-1.0.0.0.cnf file to openssl.cnf

# source ./vars

Source the variables file. A warning prompt will appear.

# ./clean-all

Confirm the clean of existing keys that will not be needed

# ./build-ca

Execute the script that will build the certificate authorities

Use the default settings for the demonstration

# ./build-key-server server

Execute script to create a key for server

Use default settings for demonstration

Press y to continue through the prompts

# ./build-dh

Execute the Deffie-Helman script

# cd keys

# ls

Navigate to keys directory and list. The serials of the keys and several other files will be added

# cp dh2048.pem ca.crt server.crt server.key /etc/openvpn

Copy the necessary files for the openvpn server to the openvpn directory

**Configuring an openVPN Client**

# cd /etc/openvpn/easy-rsa/

Navigate to teh directory where the configurations were made to host

# ./build-key client

Generate keys for the client by executing the client script

Accept the default values

# cd /keys

# ls

Now the files for the client have been created including the key and certificate

# cp client.key client.crt /etc/openvpn

Copy the files into the service directory

# systemctl enable openvpn@server

# systemctl start openvpn@server

Start and enable the service

# ip a s

A new interface will be listed as UNKNOWN with a common 10.8.0.1 ip

# yum install openvpn

Install openvpn on the client server. This is the only required packages for clients

# mkdir certs

# cd /certs

# scp root@server2:/etc/openvpn/ca.crt .

# scp root@server2:/etc/openvpn/client.key .

# scp root@server2:/etc/openvpn/client.crt

Create a certs directory in the root home of the client and copy the ca.crt, client.key, client.crt from the vpn server to the client server

# cp /usr/share/doc/openvpn-2.3.10/sample/sample-config-files/client.conf .

Create a client configuration by copying the share version

# vi client.conf

Edit the file and remove all but the below

client

dev tun

proto udp

remote server2 1194

resolv-retry infinite

nobind

user nobody

group nobody

persist-key

persist-tun

ca /root/certs/ca.crt

cert /root/certs/client.crt

key /root/certs/client.key

comp-lzo

verb 3

**Connecting to the openVPN Server**

The client may be in their home or in a public network. The private and public keys are required to vpn into the server. This will give a user access to the corporate network outside the corporation

# openvpn --config client.conf

The connection will be established if everything worked error free

**Methods to Tunnel Traffic Module Recap**

SSH Tunnel

ssh -f -L 8080:localhost:80 root@s2 -N

OpenVPN

yum install openvpn easy-rsa

/etc/openvpn/server.conf

push "redirect-gateway def1 bypass-dhcp"

push "dhcp-option DNS 8.8.8.8"

push "dhcp-option DNS 8.8.4.4"

user nobody

group nobody

Generate Keys

./clean-all

./build-ca

./build-key-server server

./build-dh

./build-key client

**Monitoring the Network Module Introduction**

Exam Objectives

Monitor Network Performance

Demo

Check network path

Monitor network with ip

Using netstat

Reading network statistics with sysstat

**Checking the Network Path**

tracepath Defined

It traces path to destination discovering MTU along this path. It uses UDP port or some random port. It is similar to traceroute, only does not not require superuser privileges and has no fancy options

# tracepath [www.pluralsight.com](http://www.pluralsight.com)

 Command will list the process of the traffic from one point to another. This will help determine if the process is behaving in the expected manner

**Display Network Statistics with ip**

# ip a s

Standard command

# ip -4 a s

# ip -6 a s

-4 switch is version 4

-6 switch is version 6

# ip link show

link switch is used for mac address

# ip link show enp0s3

Command will list only the enp0s3 interface

# ip -s link show enp0s3

-s switch will list statistics

Removing the interface in teh command will list all interfaces

**Using netstat to Display Network Statistics**

# netstat -tln

List the tcp, listening and port numbers

# netstat -i

-i switch is for interfaces

# netstat -s

List the stack statistics

**Working with Historical Data Using sysstat**

sysstat package will be required for this module

# ls /etc/cron.d

Location of the sysstat configuration file Collected every ten minutes

# ls /var/log/sa

Location of the log files

# sar -n DEV

Listing todays file of the device logs. this will list in 10 minute intervals if there is any information created

# sar -n DEV 1 1

List 1 capture of 1 second gaps

# sar -n DEV 1 3

List 1 second gaps with 3 captures

# sar -n DEV 1

List in 1 second gaps with no max count

**Using nmap**

nmap package is required for this module

nmap Defined

Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime

# yum install -y nmap

# nmap scanme.nmap.org

# nmap --iflist

--iflist switch lists the host interfaces and routes

**Monitoring the Network Module Recap**

tracepath

# tracepath [www.pluralsight.com](http://www.pluralsight.com)

IP

# ip link show enp0s3

# ip -s link show enp0s3

netstat

# netstat -i

# netstat -s

sysstat

sar -n DEV 1 3

nmap

# nmap scanme.nmap.org

# nmap --iflist