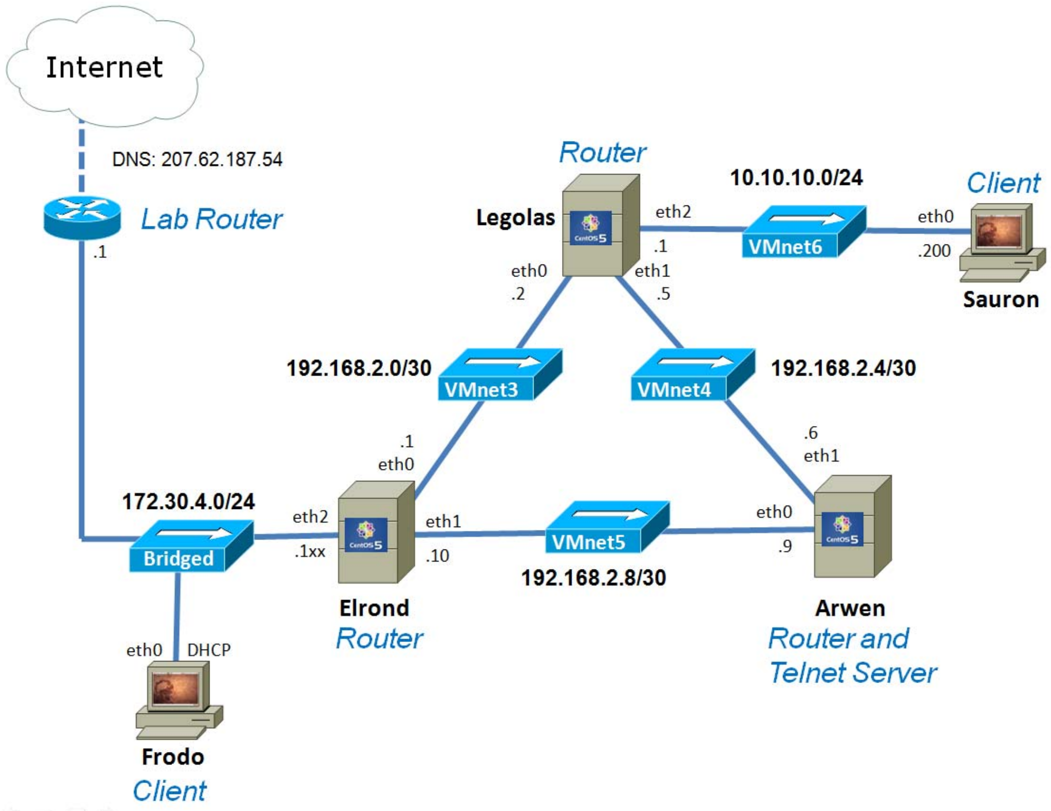
Cabrillo College

CIS 192 Linux Lab Exercise

Lab 4: Dynamic routing and tunnels
Spring 2009

Lab 4: Dynamic routing and tunnels

In this lab we will be using the Quagga package to implement dynamic routing across the three routers shown in the diagram below. For extra credit an SSH tunnel will be implemented through Elrond to the Telnet server on Arwen.



Supplies

- VMWare Server 1.08 or higher
- 192 VMs shown above

Preconfiguration

- Original versions of all VMs. Note, this will set the network configurations back to down or DHCP settings.
- You will need access to a DHCP server to assign addresses for the 172.30.4.0/24 network. This is already configured if the lab is done using the CIS VMware Stations in the CIS Lab (room 2504) or the CTC. If you plan to do this lab at home see: http://simms-teach.com/howtos/129-working-at-home.pdf

Forum

Use the forum to ask questions, collaborate, post tips and any lessons learned when you have finished. Forum is at: http://simms-teach.com/forum/viewforum.php?f=18

Background

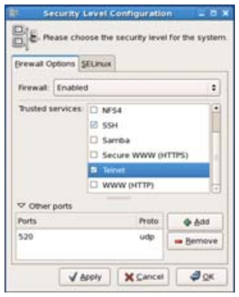
Quagga is a GPL licensed routing software suite that implements OSPF, RIP and BGP routing protocols on Linux. See http://www.quagga.net/ for details. For this

Part I – Revert VMs, add NIC cards, and cable

1. Revert the VMs back to their snapshots.
2. Add a third NIC to Elrond and Legolas.
3. Cable all VMs to the VMnets shown above.
4. Power on the VMs. If you run out of memory, modify the Host Settings (Memory Tab) to **Allow most virtual machine memory to be swapped.**

Part II – Punch some holes in the firewalls

1. On the Elrond, Legolas and Arwen routers go to run level 5 (use **init 5** or **startx**) and make the following firewall adjustments:
   a. Open a port for **RIP (UDP 520)** using the graphical **Security Level Configuration** utility. For **Arwen only**, make **Telnet** a trusted service.



Security Level Configuration

Please choose the security level for the system.

Firewall Options | SELinux

Firewall: Enabled

Trusted services:
- [ ] NFS4
- [✓] SSH
- [ ] Samba
- [ ] Secure WWW (HTTPS)
- [✓] Telnet
- [ ] WWW (HTTP)

Other ports

Ports	Proto
520	udp

   b. From the command line, backup the firewall settings, then disable filtering forwarded packets with:

```
iptables-save > /etc/sysconfig/iptables.bak
iptables -D FORWARD 1
iptables-save > /etc/sysconfig/iptables
```

   The default CentOS firewall filters all forwarded packets using the firewall INPUT filter. Unfortunately this blocks any DNS queries so we will just remove the rule that does this.

Part III – Configure NICs

Use the diagram above to configure each host. Frodo uses DHCP and should not need any additional configuration.

1. For all other hosts, configure:
   - IP Address, subnet mask, broadcast address on each interface
   - Default gateway on each host
     - Elrond to 172.30.4.1
     - Legolas and Arwen both to Elrond
     - Sauron to Legolas
   - DNS server (**207.62.187.54)** on each host
2. For Elrond's eth2 interface on Shire, use the **Static IP Address Table** in the Appendix.
3. Here is an example of the NIC configuration files Legolas. Note: **your MAC addresses will be different**:

Default Gateway	[root@legolas ~]# cat /etc/sysconfig/network
	NETWORKING=yes
	NETWORKING_IPV6=no
	HOSTNAME=legolas.localdomain
	GATEWAY=192.168.2.1
DNS	[root@legolas ~]# cat /etc/resolv.conf
	nameserver 207.62.187.54
eth0	[root@legolas ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
	# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
	DEVICE=eth0
	HWADDR=00:0C:29:7C:18:F5
	ONBOOT=**yes**
	BOOTPROTO=static
	IPADDR=192.168.2.2
	NETMASK=255.255.255.252
	BROADCAST=192.168.2.3
	[root@legolas ~]#
eth1	[root@legolas ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
	# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
	DEVICE=eth1
	HWADDR=00:0C:29:7C:18:FF
	ONBOOT=**yes**
	BOOTPROTO=static
	IPADDR=192.168.2.5
	NETMASK=255.255.255.252
	BROADCAST=192.168.2.7
	[root@legolas ~]#
eth2	[root@legolas ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth2
	# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
	DEVICE=eth2
	HWADDR=00:0c:29:7c:18:09
	ONBOOT=**yes**
	BOOTPROTO=static
	IPADDR=10.10.10.1
	NETMASK=255.255.255.0
	BROADCAST=10.10.10.255
	[root@legolas ~]#

Note: Be sure and use **service network restart** so the NIC settings take effect.

Part IV – Enable Packet Forwarding

Enable packet forwarding on Elrond, Legolas and Arwen so they can function as routers.

1. This is an example for Legolas:

Temporary	**echo 1 > /proc/sys/net/ipv4/ip_forward**
Permanent	[root@legolas ~]# cat /etc/sysctl.conf # Kernel sysctl configuration file for Red Hat Linux # # For binary values, 0 is disabled, 1 is enabled. # See sysctl(8) and # sysctl.conf(5) for more details. # Controls IP packet forwarding **net.ipv4.ip_forward = 1** *< snipped >* [root@legolas ~]#

2. Test what you have done so far using the ping command.

Part V – Dynamic routing table updates

Rather than use static routes, in this lab we will use dynamic routing. Quagga will be used to implement the RIPv2 protocol across Elrond, Legolas and Arwen.

1. We will use **yum install quagga** to install the routing software on Elrond, Legolas, and Arwen. Elrond will have Internet capability. Legolas and Arwen may or may not (in the Lab … no, at home (with Nosmo VM) … yes). Check the Appendix to see a quick way to get temporary Internet capability for Legolas and Arwen.
2. Edit the two configuration files. Here is an example for Legolas:

Edit the **zebra** configuration file	[root@legolas ~]# cat **/etc/quagga/zebra.conf** **hostname legolas** **password secret** **log file /var/log/quagga/zebra.log** [root@legolas ~]#
Create then edit the **ripd** configuration file	[root@legolas ~]# cat **/etc/quagga/ripd.conf** ! ! Zebra configuration saved from vty ! 2009/02/25 16:36:10 ! **hostname legolas** **password Cabri11o** **log file /var/log/quagga/ripd.log** ! **debug rip events**

	debug rip zebra ! **interface eth0** **no ip rip authentication mode text** **no ip rip authentication mode md5** **!** **interface eth1** **no ip rip authentication mode text** **no ip rip authentication mode md5** **!** **router rip** **redistribute connected** **redistribute static** **network eth0** **network eth1** ! **line vty** ! [root@legolas ~]#

3. Set ownership of the configuration files with:

   cd /etc/quagga/
   chown quagga:quaggavt ripd.conf zebra.conf

4. Start up the daemons:

   service zebra start
   service ripd start

5. Configure the daemons to run at system startup:

   chkconfig zebra on
   chkconfig ripd on

6. After all three routers have Quagga installed, configured and running check the routing tables. On Legolas you should see:

```
[root@legolas ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.8     192.168.2.1     255.255.255.252 UG    2      0        0 eth0
192.168.2.0     0.0.0.0         255.255.255.252 U     0      0        0 eth0
192.168.2.4     0.0.0.0         255.255.255.252 U     0      0        0 eth1
172.30.4.0      192.168.2.1     255.255.255.0   UG    2      0        0 eth0
10.10.10.0      0.0.0.0         255.255.255.0   U     0      0        0 eth2
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth2
0.0.0.0         192.168.2.1     0.0.0.0         UG    0      0        0 eth0
[root@legolas ~]#
```

7. Use Zebra to view the routing table. On Legolas you should see:

```
[root@legolas ~]# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is Quagga (version 0.98.6).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
legolas> enable
```

```
legolas# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

K>* 0.0.0.0/0 via 192.168.2.1, eth0
C>* 10.10.10.0/24 is directly connected, eth2
C>* 127.0.0.0/8 is directly connected, lo
K>* 169.254.0.0/16 is directly connected, eth2
R>* 172.30.4.0/24 [120/2] via 192.168.2.1, eth0, 09:37:17
C>* 192.168.2.0/30 is directly connected, eth0
C>* 192.168.2.4/30 is directly connected, eth1
R>* 192.168.2.8/30 [120/2] via 192.168.2.1, eth0, 09:37:17
```

8. We still need to add some static routes on Frodo to Elrond. Use:
   route add –net 10.0.0.0 netmask 255.0.0.0 gw 172.30.4.1xx
   route add –net 192.0.0.0 netmask 255.0.0.0 gw 172.30.4.1xx
9. Test and make sure all the VMs can ping each other now.

Part VI – Set up a SSH tunnel through Elrond (Extra Credit)

This part of the lab covers the use of the utility secure shell (ssh) to forward ports. We'll use port 8000 to forward to the remote port 23, but any port can be substituted. However it is important to remember that ports 1023 and lower are considered reserved and require root access to forward. Because this lab uses only port 8000, root access is not required. You need a regular user (cis192) account on the destination and pass-through computers.

Forwarding ports using ssh is a convenient way to protect information sent over the Internet, because unlike telnet and ftp, ssh encrypts data to protect against eavesdropping programs. Port forwarding is required in situations in which normal connections cannot be established. If a computer is part of a LAN and cannot be reached directly because of a firewall or some other barrier, it might be easier to use port forwarding. Telnet traffic travels in clear text and is not secure.

We will configure **Arwen** to be a Telnet server and then set up a secure SSH tunnel through **Elrond** to access it from the outside.

1. Use **yum install telnet-server** to install the telnet service on Arwen. See the Appendix if you don't have Internet access on Arwen.
2. Earlier you poked a hole in the firewall for Telnet. You can check for this with **iptables –L** and look for:
   ACCEPT tcp -- anywhere anywhere state NEW tcp dpt:telnet
3. Configure the Telnet service to be enabled but only accept logins from Elrond:

Edit telnet configuration file	[root@arwen ~]# cat **/etc/xinetd.d/telnet**
	# default: on
	# description: The telnet server serves telnet sessions; it uses
	# unencrypted username/password pairs for authentication.
	service telnet
	{
	flags = REUSE
	socket_type = stream
	wait = no
	user = root
	only_from = 192.168.2.10
	server = /usr/sbin/in.telnetd

	log_on_failure += USERID **disable = no** } [root@arwen ~]#

4. Finally, use **service xinetd restart** to make the settings take effect.
5. Go to **Frodo** now and let's try logging in through an SSH tunnel.
6. Use the following to create the tunnel:

   ssh -L 8000:192.168.2.9:23 cis192@172.30.4.1xx
   (where 172.30.4.1xx is Elrond)
7. From a **different terminal on Frodo** use the following to connect to the Telnet server and enter a command to sniff with Wireshark:

   telnet localhost 8000
   (login as cis192)
   echo this is a secret
8. Use Wireshark to capture the traffic on both sides of Elrond so you can see that it is encrypted from Frodo to Elrond and in clear text from Elrond to Arwen.

To turn in

Your *lab4* **text** file should contain the following sections.

- Standard boilerplate information:
  - CIS 192 Lab *XX*
  - *Name*
  - *Date*
  - TBA hours*: X.X*
  - Station number: CIS-Lab-*XX*
- ifconfig and route –n output for Elrond
- ifconfig and route –n output for Legolas
- ifconfig and route –n output for Arwen
- Successful ping output from Frodo to Sauron
- Successful ping output from Sauron to Frodo
- Command summary (documented examples of key commands for future reference)

Check your work for completeness then submit as many times as you wish up until the due date deadline. Remember, **late work is not accepted**, so start early, plan ahead for things to go wrong and use the forum to ask questions.

[p]scp lab4 cis192@opus.cabrillo.edu:lab4.lastname

Grading rubric (30 points)

3 points for complete submittal, professional appearance and quality
8 points for correct routing tables on routers
8 points for correct interface configuration on routers
8 points for successful end-to-end pings
3 points for useful command summary

Extra Credit (10 points)

Email me:
1) Two Wireshark screen captures showing:
   a) The encrypted (Frodo to Elrond) "Follow TCP stream" of logging in as cis192 and typing the **echo this is a secret** command.
   b) The clear text (Elrond to Arwen) "Follow TCP stream" of logging in as cis192 and typing the **echo this is a secret** command.
2) A screen capture of your terminal showing the telnet session for 1a or 1b above

Appendix

Getting temporary Internet capability for software downloads

In the Lab, Legolas and Arwen will not have Internet capability because Nosmo has no routes back to your private networks behind Elrond. To install quagga from the Internet do the following on Legolas and Arwen:

1. **ifconfig eth0 down**
2. Re-cable eth0 from VMnet3 to **Bridged** network.
3. Use **dhclient eth0** to get a Shire IP address.
4. Use **yum install quagga** to install the routing software.
5. **Arwen** additionally needs the Telnet service so use **yum install telnet-server** after installing quagga.
6. Use **dhclient –r** to release DHCP address.
7. **ifconfig eth0 down**
8. Re-cable eth0 from Bridged back to the **VMnet3** network.
9. Use **service network restart** to use static IP settings again.

Static IP Address Table

Station	IP	Static 1	Static 2
CIS-Lab-01	172.30.4.101	172.30.4.121	172.30.4.122
CIS-Lab-02	172.30.4.102	172.30.4.123	172.30.4.124
CIS-Lab-03	172.30.4.103	172.30.4.125	172.30.4.126
CIS-Lab-04	172.30.4.104	172.30.4.127	172.30.4.128
CIS-Lab-05	172.30.4.105	172.30.4.129	172.30.4.130
CIS-Lab-06	172.30.4.106	172.30.4.131	172.30.4.132
CIS-Lab-07	172.30.4.107	172.30.4.133	172.30.4.134
CIS-Lab-08	172.30.4.108	172.30.4.135	172.30.4.136
CIS-Lab-09	172.30.4.109	172.30.4.137	172.30.4.138
CIS-Lab-10	172.30.4.110	172.30.4.139	172.30.4.140
CIS-Lab-11	172.30.4.111	172.30.4.141	172.30.4.142
CIS-Lab-12	172.30.4.112	172.30.4.143	172.30.4.144

Pod 1		172.30.4.113	172.30.4.145
Pod 2		172.30.4.114	172.30.4.146
Pod 3		172.30.4.115	172.30.4.147
Pod 4		172.30.4.116	172.30.4.148