Hands-on Lab Description

CS-NET-00002 – Computer Network Gateway Setup

Category:

```
CS-NET: Computer Networking
```

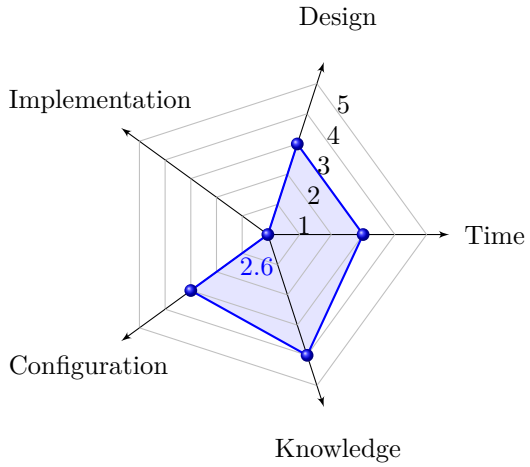
Objectives:

```
1 Setup the gateway to allow traffic flows to be transmitted among networks connected
      through it.
2 Check multi-hop network connectivity.
```

Estimated Lab Duration:

```
1 Expert: 5 minutes
2 Novice: 30 minutes
```

Difficulty Diagram:



Difficulty Table.

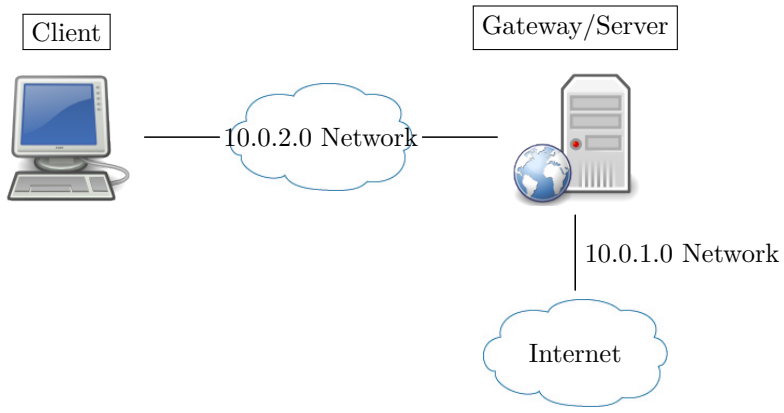
Measurements	Values (0-5)
Time	3
Design	3
Implementation	0
Configuration	3
Knowledge	4
Score (Average)	2.6

Required OS:

```
Linux: Ubuntu 18.04 LTS
```

Lab Running Environment:

```
VirtualBox https://www.virtualbox.org/ (Reference Labs: CS-SYS-00101)
```




```
1  Client: Linux (Ubuntu 18.04 LTS)
2  Gateway/Server: Linux (Ubuntu 18.04 LTS)
3  Network Setup:
   Both VMs connected to Local NAT Network: 10.0.2.0/24
   Internet is connected to Gateway/Server VM through network: 10.0.1.0/24
```

Lab Preparations:

```
  Initial setup: no packet forwarding rules are setup on the Gateway.
  Pre-knowledge: A short Linux tutorial (CS-SYS-00001)
```

Lab Overview

In this lab, you will need to explore a multi-hop network where a network gateway connects to two different networks. This lab will practice the basic networking concepts such as packet forwarding, default gateway, routing, networking services, and network configuration and diagnostic tools.

Task 1 Initial setup and connectivity testing

```
  Optional: use Lab CS-NET-00001 to learn the network detials of Client and
      Gateway/Server VM; and test the connectivity between interfaces and to the
      Internet.
```

“nobreak

Task 2 Check the network setup on the Gateway/Server VM

Using the following command to check the IP address and interfaces configured on the gateway machine.

```
$    ifconfig % you can use ifconfig interface to check a specific interface setup
```

There should be two interfaces created and configured for the gateway: one to the local network 10.0.2.0/24 network, one to the public network 10.0.1.0/24. Each network, there must be an IP address assigned to it. Using the following command to test the connectivity to external and client:

```
$    ping ip_address
```

The client addresses can be obtained from ifconfig command result in the client VM. We can use Google’s public server 8.8.8.8 to test the connectivity from the gateway/server to external systems.

Using the following commands to check the routing table:

```
$    route -n
$    netstate -nr
```

The *-n* option is to only display numerical values. A example of routing table is shown in Figure CS-NET-00002.2:

```
ubuntu@ubuntu:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.0.1      0.0.0.0         UG    100    0        0 ens3
0.0.0.0         192.168.0.1     0.0.0.0         UG    20101  0        0 ens4
0.0.0.0         10.0.0.1        0.0.0.0         UG    20102  0        0 ens5
10.0.0.0        0.0.0.0         255.0.0.0       U     102    0        0 ens5
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 ens3
169.254.169.254 172.16.0.1      255.255.255.255 UGH   100    0        0 ens3
169.254.169.254 192.168.0.2     255.255.255.255 UGH   101    0        0 ens4
169.254.169.254 10.0.0.3        255.255.255.255 UGH   102    0        0 ens5
172.16.0.0      0.0.0.0         255.240.0.0     U     100    0        0 ens3
192.168.0.0     0.0.0.0         255.255.255.0   U     101    0        0 ens4
```

Figure CS-NET-00002.2
Checking VM’s IP address.

Based on the routing table, you need to understand the following concepts and where the routing table shows you the relevant information:

```
Gateway (default gw) % How many default gateways that you can have?
Genmask (network mask) % Understand host and network addresses
Flags % Under the meanings of U (up), G (through gateway), H (to host), D (created
    by a redirect), and M (modified by a redirect)
Metric % Understand how to use metric to prioritize the routing, e.g., use
    ifmetric command to modify the metric value.
Ref % Number of references to this route.
Use %Count of lookups for the route. Depending on the use of -F and -C this will be
    either route cache misses (-F) or hits (-C).
Iface % Interface to which packets for this route will be sent.
```

Task 3 Enable Packet forwarding on Gateway

In this task, you need to allow the gateway to transfer packets from the client to/from the Internet to install the necessary software package. The following command set up a value 1 in the *ip_forward* file in the directory */proc/sys/net/ipv4/* to enable packet forwarding on the Gateway/Server VM.

```
$    echo "1" > /proc/sys/net/ipv4/ip_forward
```

This command basically change the value from 0 to 1 in the file *ip_forward*. Or we can use the command *sysctl* to enable the IP forwarding.

```
$    sysctl -w net.ipv4.ip_forward=1
```

To enable IP forwarding permanently edit */etc/sysctl.conf* and add the following line. This will enable IP forwarding even after the system reboot.

```
net.ipv4.ip_forward = 1
```

Task 4 Configure default routes

In the lab topology figure, the Gateway/Server VM is required to forward traffic from/to the client. The local networks between the client and Gateway/Server VM, the Client VM and Gateway/Server VM must be configured as different network, i.e., they have different network address. As a result, the client cannot communicate to Internet directly, and it must send packet through the gateway to a Internet destination. If it does not know how to forward the packet to the destination, the packet forwarding gateway is also called "default gateway" for client.

The syntax to set up a default gateway is given as follows:

```
route add default gw {IP-ADDRESS} {INTERFACE-NAME}
```

Where, *IP-ADDRESS* is the default gateway (or router) IP address and *INTERFACE-NAME* is the packet outgoing interface name such as *ens3*.

For example, if the client is on *10.0.2.0/24* network and the interface connected to it is *enp0s3*, and the gateway's IP address is *10.0.2.15*, then you can type the following command as the root user:

```
$    route add default gw 10.0.2.15 enp0s3
```

Or use the ip command (newer syntax) to route all traffic via *10.0.2.15* gateway connected via *enp0s3* network interface:

```
$    ip route add 10.0.2.0/24 dev enp0s3
```

OR

```
$    ip route add 10.0.2.0/24 via 10.0.2.15
```

Note that in this project, we do not need to set the default gateway on the gateway machine, in which the system had already set it up to allow all unknown destinations be forwarded to its default gateway on the *10.0.1.0/24* network. You can use the command $route - n$ on the gateway to check its setup.

Task 5 Enable network traffic to Internet

In the lab running environment, we mimic a real networking scenario, where the client-side network represents a private network domain. However, you might want to enable the Internet access for client VM to install additional software packages. To do so, you should allow the gateway to forward the packets to Internet and transfer the response back to the client. Then you need to configure *iptables* on the gateway.

To setup an *iptables* firewall to forward network traffic, we need to configure its *FORWARD* chain of the *FILTER* table or *NAT* table. We must note that, in Ubuntu LTS (Long-Term Support) versions, a software package called *ufw* was introduced since 16.04 to simplify the firewall configuration. However, this package is mostly to setup host-based firewall to enable and disable traffic goes out or comes into the host. For network firewall, you should still use native *iptables* configurations to setup proper network traffic filtering policies.

To enable a quick traffic forwarding policy from the lab environment to Internet, you can do the follows:

```
$    iptables -P FORWARD ACCEPT % set up default packet filtering policy for foward
       chain to "accept"
$    iptables -t nat -A POSTROUTING -o exit_interface -j MASQUERADE % enable packet
       forwarding in masquerade mode to the out-going interface: exit_intferface,
       replace exit_intferface with the interface name on Gateway/Server VM that is
       connected to 10.0.1.0 network.
```

In the above configuration, the *exit_interface* is the gateway's interface connected to the external network, and *-j MASQUERADE* will change source IP addresses of all forwarded packets to the gateway's IP address that is assigned to the *exit_interface*.

Once the system reconfiguration task is done, to delete the *iptables* configuration, you can issue the following command:

```
$    iptables -t nat -L POSTROUTING -n -v --line-number % -t nat: show nat table; -L
       POSTROUTING: show postrouting chain; -n: numerical value; -v: verbose;
       --line-number: show line number
```

This command will return the recent configured *POSTROUTING* rule given in Figure CS-NET-00002.3.

```
ubuntu@ubuntu:~$ sudo iptables -t nat -L POSTROUTING -n -v --line-number
Chain POSTROUTING (policy ACCEPT 121 packets, 8940 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      355 27124 MASQUERADE  all  --  *      ens5    0.0.0.0/0            0.0.0.0/0
```

Figure CS-NET-00002.3
Using iptables command to show the POSTROUTING chain of NAT table.

Next, you can delete the *POSTROUTING* rule by issuing one of the following commands:

```
$    iptables -t nat -D POSTROUTING 1 % Here 1 is the line number learned from the
       above command; -D: delete
$    iptables -t nat -D POSTROUTING -o ens5 -j MASQUERADE % or, if you know the exact
       deleting rule; -D: delete
```

Finally, turn of the default policy to *DROP*, which is a safer way to enable *Whitelist*-based firewall policy, i.e., only accept known good traffic.

```
$    iptables -P FORWARD DROP
```

Task 6 Check DNS service to access Internet

Once, the default gateway is set up correctly on client, and the IP forwarding is enabled on Gateway, you need to check if you can access to Internet by using DNS service. The first step is to make sure the packet forwarding work properly and you can issue the following command:

```
$    ping 8.8.8.8   % 8.8.8.8 is a well-known google dns server.
```

If you can ping this address, that means you can access to Internet without using DNS service: Next, you should issue the following command to test the DNS service:

```
$    ping www.google.com % this command requires the dns service to translate the URL
       to its IP address
```

If you ping is successful, that means you DNS service works properly to allow you to access Internet by using URL. This step is critical since many of our services use URLs to access to remote systems.

Alternatively, you can directly issue an DNS request to test if your DNS service is set up properly:

```
$    nslook www.google.com % use nslook up to send a dns request
$    dig www.google.com    % use dig to send a dns request
```

If you cannot get a response, then you need to check with version of Linux you are using by issuing the following command:

```
$    lsb_release -a
```

From Ubuntu 17.10, the default is to list the 127.0.0.53 DNS stub as only DNS server and it is specified */etc/resolv.conf* in order to connect all local clients that bypass local DNS APIs to *systemd-resolved*. For example, the Thoth lab is built on the Openstack and the DNS default system DNS resolver is set to Google's dns servers: 8.8.8.8 and 8.8.4.4. To check your system-sub DNS service you can issue the following command (this command may require to *apt install gnome-system-tools*):

```
$    systemd-resolve --status % shows how dns request to be handled and list used
       system dns servers
```

By sending DNS requests to 127.0.0.53, the system will send the DNS requests to the Google's DNS server as long as the IP forwarding system works properly. This can simplify the system setup. However, if you want to involve local DNS services, you can simply add your local DNS server in the file */etc/resolv.conf*, such as:

```
nameserver 127.0.0.53
nameserver 10.0.0.9 # a local dns server
nameserver 192.168.0.10 # a local dns server
```

Related Information and Resource

```
Understand routing table: https://www.cyberciti.biz/faq/what-is-a-routing-table/
```