Main Page

Minimize

User: e0540252 e0540252 (GUEST)
e0540252@u.nus.edu
Registered Account
Log off Account Links
Machine State: **RUN**
Refresh in: 0:08.
Time left: Holding
Boot progress: complete

control | **connect** | *stats* | *useful*

Home IP:	137.132.84.43
VM IP:	10.0.1.113
Direct:	telnet or ssh to linuxzoo.net
SSH:	unavailable
VM Web:	http://host-1-113.linuxzoo.net/
JScript Telnet:	Network / ~~Console~~
Java Telnet:	Auto
JavaScript SSH:	SSH
JavaScript VNC:	VNC
URI telnet:	linuxzoo.net
Connect:	Username: root, Password: secure

SHARED IP MODE

Active Scan

This practical covers an introduction to some elementary linux networking commands, along with an introduction to network discovery and active scanning. Remember if you try these in your own machine make sure you have permission, as running these on or over system you have no permissions for is likely illegal.

Your virtual machine sits inside a virtual network, and this network connect you to other virtual machines. One of those machines is your network gateway. The gateway itself is a router, and this connects together all the virtual networks from yourself and other cloud users. Above this another router sits, connecting together cloud users running on different servers, and this in turn links to the main system gateway, connecting the virtual world to the real network via a NAT. This gives many networks to explore, but in reality layered firewalls within the virtual network prevent most explorations.

To reset all the check buttons from a previous attempt click here

Question 1: Network IP

What is the primary IP address of your virtual machine network connection? If you have multiple connections for some reason this is the ethernet device related to eth0. You should use the "ip" command for this (not ifconfig as this is very old and weak). You can use "ip help" to discover the options. The "addr" option is what we want here, and again you can discover more with "ip addr help". If you want to jump to the answer, try "ip addr show".

Machine ip : 10.0.1.113

Check

Tests: Complete
Network Address PASSED

Question 2: Mac address

This time use "ip link help" and "ip link show" and find out the mac address of eth0.

Machine mac: 00:0c:71:00:01:90

Check

Tests: Complete
Layer 2 Address PASSED

Question 3: Network Gateway

This time use "ip route help" and "ip route show" and find out the IP address of your machine's gateway. The gateway is where your packets would be sent if it is destined for an IP address which is not in the local subnet.

Machine gateway: 10.0.1.118

Check

Tests: Complete
Gateway IP PASSED

Question 4: Gateway Mac

Look into /proc/net/arp. What is the mac address of the gateway?

Gateway mac: 1e:7f:da:34:99:58

Check

Tests: Complete
Gateway mac PASSED

Question 5: Wireshark

Run Wireshark in Kali.
Applications > Internet > Wireshark
Start a live capture, and in a kali terminal "ping" the machine 10.200.0.1 with 5 pings (you can use the "-c" option of ping for this, or press CTRL C to stop the ping). This ping should send a number of ping request packets, and if the network is operating correctly, you should receive a corresponding number of ping response packets.

What does wireshark say:
is the length of a ping packet: 98
is the ICMP type number of a ping request packet: 8

Check

Tests: Complete
Length in bytes PASSED
Type number for request PASSED

Question 6: tcpdump and host sweeping with ping

In this set of questions you will learn how to run a command multiple times by looping over a numerical range. This is particularly useful when you are using a network command which normally only targets a single machine but you want to target a range of IP numbers.

Firstly, ping 10.200.0.1 using "-c 1" (so only send a single packet exchange). What is the TTL of the response to this ping packet?
TTL: 63

Check

Tests: Complete
TTL Correct PASSED

It is possible to use bash variables in commands. One way is to set a variable using a for loop construct. Look at the following example:

```
for i in {1..10}; do mkdir /home/kali/newdir$i; done;
```

The "for" part till the first ";" sets up a loop which sets a variable called "i" to the values 1,2,3,4,5,6,7,8,9,10, one after another. For each value the "do" command is executed; "done" indicates the end of the loop. The $i is replaced by the current value of i.

In this example, "newdir$i" becomes "newdir1","newdir2", etc, up to "newdir10".

Consider another example:

```
for i in {1..10}; do echo 10.200.0.$i; done;
```

How many IP numbers get displayed? Count: 10

Check

Tests: Complete
Count Correct PASSED

In this question you will use the for loop and ping to investigate traffic. Wireshark is very graphical, and sometimes it is hard to see all the options. So this time use tcpdump to capture a pink sweep. The `tcpdump` command allows us to capture all or some of the network traffic on a particular network device.

Make sure you have two terminal windows (either vnc or via ssh or telnet). In one window, which will deal with the monitoring of the network, run this command:

```
tcpdump -vni eth0 icmp > /root/dump1
```

This will start the capture. Press CTRL C in this window when you want the capture to end. These flags indicate that it is a verbose capture (-v), and that IP numbers will be kept numeric and not generate DNS lookups (-n), and the capture is on interface eth0 (-i eth0). The captured information is saved to the dump1 file.

In the other terminal window, which in turn will generate the packets during the experiment, do the following REPLACING ?????? with suitable parameters to make the loop go from 1 to 30, and for the ping to send packets to 10.200.0.1 up to 10.200.0.30, depending on i.

```
for i in {?????}; do ping -c 2 ????? ; done;
```

This will run a ping sweep of ips in the range 10.200.0.1 to 10.200.0.30. It is not fast... may take around a minute.

Once complete stop the tcpdump capture.

Check

Tests: Complete
Capture seems to start PASSED
Capture seems to end PASSED

Use cat or less to examine the packet capture, and locate the echo reply packets. The ICMP packet is contained within an IP packet, and the related IP packet information is on the line just before the ICMP information. From that information, discover the IP packet id for the last received echo reply.

Packet sequence number: 14927

Check

Tests: Complete
dump1 seems ok test1 PASSED
dump1 seems ok test2 PASSED
Last reply seq PASSED

Use grep and wc. How many echo requests were sent?

Number of echo requests: 30

Check

Tests: Complete
dump1 seems ok test1 PASSED
dump1 seems ok test2 PASSED
Echo requests PASSED

Use grep, looking for packets with "echo reply", and use the '-B 1' to also show the line immediately before the line which matches (which contains the packet header information of the reply). Sometimes the ttl is 64, but sometimes the ttl is something else.

What is the other ttl: 63
Why? The ttl is smaller as the machine in question is further away in terms of network hops

Check

Tests: Complete
dump1 seems ok test1 PASSED
dump1 seems ok test2 PASSED
Other ttl PASSED

reason PASSED

Use tcpdump again, but this time capture only traffic you dont expect. So in our case ignore ssh and telnet traffic using a simple filter.

```
tcpdump -vni eth0 not port ssh and not port telnet
```

Direct that file to save the capture to /root/dump2, and terminate the capture after capturing an "arping -c 2 10.200.0.1". Replace the IP "10.200.0.1" with the IP of your virtual machine gateway address.

Check

Tests: Complete
dump2 seems ok test1 PASSED
dump2 seems ok test2 PASSED

What is the packet length of the is-at reply from your gateway?
packet length: 46

Check

Tests: Complete
dump2 seems ok test1 PASSED
dump2 seems ok test2 PASSED
dump2 byte len of is-at PASSED

Question 7: Host sweeping with arping

Use the "ip route show" command and identify the start and end address (not including the network number and broadcast address) of your virtual machine's network. Use the network number and netmask to make this calculation from the "ip route show" command output.

Start IP: 10.0.1.113
End IP: 10.0.1.118

Check

Tests: Complete
start ip is right PASSED
end ip is right PASSED

```
#!/bin/bash
for i in {0..150}; do
        arping -c 2 192.168.100.$i | grep 'bytes from' | awk '{print " possible target up at: " $4 " " $5}' | sort -u
done
```

Consider the above bash script. Use an editor and write this into a script /root/arpbash. After writing the script make sure it is executable by saying "chmod +x /root/arpbash". This script should do an arping scan of the local network, if configured correctly!

In the "for" loop, which currently goes from 0 to 150, change this to be the last octet of the start and end ip you identified in the previous question. Where it says "192.168.100." change this to the first three octets of the start ip you found in the previous question.

Check

Tests: Complete
for ok PASSED
subnet ok PASSED

Run the command using ./arpbash, but save the output to dump3 in /root.

Look at the file and identify the first IP number it has found.
IP: 10.0.1.118

Check

Tests: Complete
ip right PASSED

Question 8: Network tracing with traceroute

In order for your virtual machine to reach the internet, it's packets travels through a number of virtual networks. The final network node is 10.200.0.1.

Using traceroute, investigate the route to reach 10.200.0.1. Note you must use ICMP ECHO in traceroute, rather than the default. Find the right flag in the manual.

Num of hops: 2
Hostname of last hop: hub1-gw.linuxzoo.net

Check

Tests: Complete
Count the hops PASSED
Last hop hostname PASSED

Question 9: Network scanning with nmap

Run nmap as a network scanner, but save the data as a grepable data file.

```
nmap -PE -oG /root/out1 -sn 10.200.0.1-20
```

This saves the search information as a file called /root/out1 in a "grepable" format. Look at the file contents once you have created it.

Check

Tests: Complete
/root/out1 looks sensible check1 PASSED
/root/out1 looks sensible check2 PASSED

Process this output file out1, and generate a file "out2" with only the IP addresses of machines which are up.

```
grep Up /root/out1 | cut -d" " -f 2 > /root/out2
```

Check

Tests: Complete
/root/out1 looks sensible check1 PASSED
/root/out1 looks sensible check2 PASSED
/root/out2 seems right PASSED

Question 10: Port scanning with nmap and netcat

Use `nmap` to analyse the ports open on 10.200.0.1. As the nmap command can take quite a while to run, restrict your scan to the open tcp ports between port numbers 50 to 80 inclusive. List the open port numbers you find with commas between them in the box below (e.g. if ports 50 and 60 are open, the answer is "50,60"). The numbers in your list must be sorted (smallest number first). Remember the box is space sensitive so dont have extra spaces!

IMPORTANT. Linuxzoo security may shut you down if you produce too many packets too quickly! Use the following options for nmap or you may be kicked off the system. Even with these options the scan may take quite a few seconds.

nmap 10.200.0.1 -p 50-80 --max-retries 3

Open ports: 53,80

Check

Tests: Complete
Identify of open ports on 10.200.0.1 PASSED

Use tcpdump to only capture traffic involving port 80.

```
tcpdump -vni eth0 port 80
```

Capture the port 80 traffic generated when nmap scanning 10.200.0.1 using a fully connected scan, i.e. -sT. Store this information in /root/nmapfull. Note you should also use -Pn with nmap, as otherwise it will generate 2 extra packets when performing host discover, making the data more complex to understand.

Check

Tests: Complete

nmapfull looks ok PASSED

Using 'S' for SYN, '.' for ACK, and 'R' for RST (which is how tcpdump shows the "Flags" in the tcpdump), state the handshake involved in the full scan captured in the previous question. Seperate each packet with a comma. For instance, if the handshake on port 80 was SYN, then ACK, then SYN+ACK, then the answer would be "S,.,S.".

Handshake in nmapfull: S,S.,.,R.

Check

Tests: Complete
nmapfull looks ok PASSED
nmapfull analysis correct PASSED

Use tcpdump to only capture traffic involving port 80.

```
tcpdump -vni eth0 port 80
```

Capture the port 80 traffic generated when nmap scanning 10.200.0.1 using a SYN connected scan, i.e. -sS. Store this information in /root/nmapsyn. Note you should also use -Pn with nmap, as otherwise it will generate 2 extra packets when performing host discover, making the data more complex to understand.

Check

Tests: Complete
nmapsyn looks ok PASSED

Using 'S' for SYN, '.' for ACK, and 'R' for RST (which is how tcpdump shows the "Flags" in the tcpdump), state the handshake involved in the full scan captured in the previous question. Seperate each packet with a comma. For instance, if the handshake on port 80 was SYN, then ACK, then SYN+ACK, then the answer would be "S,.,S.".

Handshake in nmapsyn: S.,S.,S.

Check

Tests: Complete
nmapsyn looks ok PASSED
nmapsyn analysis correct FAILED

Use nmap, limiting yourself to port 80, and perform application version identification of 10.200.0.1 (-sV). Enter the version NUMBER returned by nmap, ignoring all other information. So if it returns "Apache httpd 3.0.0 ((Redhat) OpenSSL))" the answer is "3.0.0".

Port 80 version: 2.4.6

Check

Tests: Complete
Version correct PASSED

Repeat the above scan, but introduce "--version-trace". What is the GetRequest line number shown in the trace for the Service Scan Match which produced the application version information from nmap?

Line of nmap-service-probe: 5545

Check

Tests: Complete
Line No PASSED

Use "locate" to locate the nmap-service-probe definition file, and then do

```
head -LINE FILE | tail -1
```

where LINE is the line number from the previous question and FILE is the located filename. So if the file was /var/1 and the line number was 1000, you would do "head -1000 /var/1 | tail -1".

First 2 words of the line: match http

Check

Tests: Complete
Line words PASSED

Use nmap, limiting yourself to port 80, and perform OS fingerprinting of 10.200.0.1. Save the output of running this command to /root/finger. Have a look at the file too, and see what nmap thinks 10.200.0.1 is.

Check

Tests: Complete
OS Fingerprint test1 PASSED
OS Fingerprint test2 PASSED

Switch to using netcat, and perform a similar portscan.

```
nc -vv -n -z -w2 10.200.0.1 50-80
```

Note that there are many "version" of nc, and in some installs you may get an error message with the portscan. In this case switch to "nc.traditional" rather than "nc".

What message do you get for port 50? Connection timed out

Check

Tests: Complete
netcat output for port 50 PASSED

USe wireshark or tcpdump, and monitor the netcan scan of the previous question. What sort of scan does this command do by default?

What type of scan? Full-Connect Scan

Check

Tests: Complete
netcat output for port 50 PASSED
