**Part 1:**

┌──(kali㉿kali)-[~]

└─$ traceroute uww.edu

traceroute to uww.edu (140.146.22.212), 30 hops max, 60 byte packets

1 10.0.2.2 (10.0.2.2) 0.085 ms 0.066 ms 0.036 ms

2 10.0.2.2 (10.0.2.2) 1.563 ms 1.522 ms 1.511 ms

┌──(kali㉿kali)-[~]

└─$ sudo traceroute -I uww.edu

traceroute to uww.edu (140.146.22.212), 30 hops max, 60 byte packets

1 10.0.2.2 (10.0.2.2) 0.102 ms 0.088 ms 0.083 ms

2 192.168.0.1 (192.168.0.1) 1.394 ms 1.391 ms 1.386 ms

3 abdlwixahed12-dynamic-1001-secondary-1.network.tds.net (96.60.40.1) 4.000 ms 4.623 ms 4.736 ms

4 dnvvcohfcor52-bune100.network.tds.net (64.50.224.46) 18.944 ms 19.104 ms 19.100 ms

5 madix-uwsystem.net.wisc.edu (144.92.233.237) 21.133 ms 21.255 ms 21.251 ms

6 r-uwwhitewater-mcgraw-et-0-0-0-3441.uwsys.net (143.235.33.53) 23.025 ms 21.441 ms 22.029 ms

7 uwwhitewater-sitemanaged.uwsys.net (143.235.40.66) 22.143 ms 21.852 ms 22.240 ms

8 140.146.1.146 (140.146.1.146) 22.766 ms 22.879 ms 22.874 ms

9 \* \* \*

10 \* \* \*

11 \* \* \*

12 \* \* \*

13 \* \* \*

14 \* \* \*

15 \* \* \*

16 \* \* \*

17 \* \* \*

18 \* \* \*

19 \* \* \*

20 \* \* \*

21 \* \* \*

22 \* \* \*

23 \* \* \*

24 \* \* \*

25 \* \* \*

26 \* \* \*

27 \* \* \*

28 \* \* \*

29 \* \* \*

30 \* \* \*

┌──(kali㉿kali)-[~]

└─$ sudo traceroute -T uww.edu

traceroute to uww.edu (140.146.22.212), 30 hops max, 60 byte packets

1 10.0.2.2 (10.0.2.2) 0.140 ms 0.123 ms 0.085 ms

2 web-ns-vip.uww.edu (140.146.22.212) 22.422 ms 23.893 ms 23.547 ms

4)

The first command by using just traceroute is going to track the packets from an IP network(me) on their way to the given host(uww.edu). It uses the TTL field and attemp0ts to elicit a ICMP TIME\_EXCEEDED response from each gateway along to the host and then prints each hop out back to the terminal. Running traceroute by itself and then the desired IP will report the hops that it takes to get to the given host with a 60-byte packet length that can be changed if desired, but 60 is the default packet length for IPv4 and default of 80 for IPv6. The program traces the route of an IP packet and prints it back out to the user, by listening for that time exceeded reply and increasing the ttl by 1 each time it hits the time exceeded until the packet gets to a port unreachable (made it to the host) or hits a max of 30 hops. Each hop is printed out with the ttl, and the address of the gateway with round trip time of each probe. If there is no response within a certain timeout then an \* will be printed out for that probe.

The traceroute -I uww.edu command uses the ICMP ECHO for the probes. This is the most usual method that is used. This method is used when you are able to ping(8) the host, and then that means you would be able to use the traceroute command.

The next command that we administered was traceroute -T uww.edu which is equivalent to tcptraceroute and that gives us a little additional insight into what the command does differently. With the -T addition this command runs the TCP traceroute command. This command performs a TCP traceroute using TCP SYN for the probes. This can be run with -T or –tcp added to the traceroute command. This command is intended to bypass firewalls and can be used when ICMP echoes are being filtered. This method is also known as the “half-open technique” which prevents applications on the destination host from seeing our probes at all.

It makes sense that the traceroute -I command produced the most responses because it is ICMP echo probing and is the most used method for a reason. We can use the ping command on the desired host, so we are able to then add -I to the traceroute command to use ICMP echo for the probing to gather our gateway information.

I think that the -T command seems to be the most useful because it can be used to bypass firewalls that would otherwise filter out and block ping requests. If the ping requests are blocked, then we would be unable to use the traceroute -I command as well. Whereas we can access different ports using the traceroute -T command and potentially still gather the information that we are looking for.

**Part 2:**

┌──(kali㉿kali)-[~]

└─$ proxychains firefox www.hackers-arise.com

[proxychains] config file found: /etc/proxychains4.conf

[proxychains] preloading /usr/lib/x86\_64-linux-gnu/libproxychains.so.4

[proxychains] DLL init: proxychains-ng 4.16

[proxychains] DLL init: proxychains-ng 4.16

1. socks4 72.206.181.103 4145

socks4 98.188.47.150 4145

socks5 117.251.103.186 8080

socks5 193.252.50.1 3820

2. I chose my servers randomly, other than the fact that I wanted to get a few different geographical locations that were still listed as having a low latency/high speed. Otherwise, my choice was more or less random.

3. I did not encounter any issues with accessing the hackers-arise website through the proxychain. It opened successfully and relatively quickly.

4. I think that the connection speed was faster than what I was expecting, but now thinking about it with the question posed to me I would not have expected too much of a difference since the servers chosen were ones with low latency/high speed. Also we have the chain length set to 2, whereas if it was set higher I think that the connection speed would slow down a bit. I also think if the servers that were chosen were ones with low speeds that the connection speed when using the proxychains command would be much slower than the ones with high speeds.