Exercise 5

Abd El Rahaman Shehata

Exercise 5

1. Sniffing

Create syntactically and semantically valid command-lines for the `tcpdump` packet sniffer, in order to fulfill the following goals:

1.1 Capture all packets at a network interface, which use the UDP protocol and any destination port from 1 to 1024.

```
sudo tcpdump -n '(udp and portrange 1-1024)'
```

```
tcpdump: data link type PKTAP
tcpdump: verbose output suppressed, use -v or -vv for full
 ↪  protocol decode
listening on pktap, link-type PKTAP (Apple DLT_PKTAP), capture
 ↪  size 262144 bytes
13:32:26.709850 IP 10.29.17.169.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:27.007996 IP 10.29.17.130.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:27.019405 IP 10.29.19.127.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:27.317560 IP 10.29.17.188.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:27.317847 IP 10.29.17.169.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:27.323306 IP 10.29.19.54.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:27.623658 IP 10.29.17.169.137 > 10.29.19.255.137: UDP,
 ↪  length 50
13:32:28.853296 IP6 fe80::dd11:e731:9ab7:82e.546 > ff02::1:2.547:
 ↪  dhcp6 solicit
13:32:28.864043 IP 10.29.19.54.137 > 10.29.19.255.137: UDP,
 ↪  length 50
```

1.2 Capture all broadcast and multicast packets.

```
sudo tcpdump -n "broadcast and multicast"

tcpdump: data link type PKTAP
tcpdump: verbose output suppressed, use -v or -vv for full
 ↪  protocol decode
listening on pktap, link-type PKTAP (Apple DLT_PKTAP), capture
 ↪  size 262144 bytes
13:34:20.057706 ARP, Request who-has 169.254.255.255 tell
 ↪  10.29.19.212, length 46
13:34:20.057979 IP 10.29.19.212.54915 > 10.29.19.255.54915: UDP,
 ↪  length 263
13:34:20.094798 ec:43:f6:70:4c:11 > ff:ff:ff:ff:ff:ff, RRCP-0x23
 ↪  query
13:34:20.095423 58:8b:f3:fb:93:48 > ff:ff:ff:ff:ff:ff, RRCP-0x23
 ↪  query
13:34:20.114361 90:ef:68:df:d1:f1 > ff:ff:ff:ff:ff:ff, RRCP-0x23
 ↪  reply
13:34:20.116192 ec:43:f6:70:4b:6c > ff:ff:ff:ff:ff:ff, RRCP-0x23
 ↪  query
13:34:20.118093 ec:43:f6:70:47:85 > ff:ff:ff:ff:ff:ff, RRCP-0x23
 ↪  query
13:34:20.118095 58:8b:f3:fb:93:af > ff:ff:ff:ff:ff:ff, RRCP-0x23
 ↪  reply
13:34:20.118632 IP 10.29.19.54.137 > 10.29.19.255.137: UDP,
 ↪  length 50
```

1.3 Capture all packets, whose destination is host is given via a particular IP address (e.g. 192.168.1.1), except for packets to port 80 (http) or port 23 (telnet).

```
sudo tcpdump -n "(tcp and not port 80 and not port 23) and dst
 ↪  host 192.168.1.1"

tcpdump: data link type PKTAP
tcpdump: verbose output suppressed, use -v or -vv for full
 ↪  protocol decode
listening on pktap, link-type PKTAP (Apple DLT_PKTAP), capture
 ↪  size 262144 bytes
13:41:03.752962 IP 10.29.16.211.52545 > 192.168.1.1.25: Flags
 ↪  [S], seq 1228241491, win 65535, options [mss 1460,nop,wscale
 ↪  6,nop,nop,TS val 1308414052 ecr 0,sackOK,eol], length 0
13:41:04.753399 IP 10.29.16.211.52545 > 192.168.1.1.25: Flags
 ↪  [S], seq 1228241491, win 65535, options [mss 1460,nop,wscale
 ↪  6,nop,nop,TS val 1308415052 ecr 0,sackOK,eol], length 0
```

```
13:41:05.754983 IP 10.29.16.211.52545 > 192.168.1.1.25: Flags
 ↪  [S], seq 1228241491, win 65535, options [mss 1460,nop,wscale
 ↪  6,nop,nop,TS val 1308416052 ecr 0,sackOK,eol], length 0
13:41:06.756144 IP 10.29.16.211.52545 > 192.168.1.1.25: Flags
 ↪  [S], seq 1228241491, win 65535, options [mss 1460,nop,wscale
 ↪  6,nop,nop,TS val 1308417052 ecr 0,sackOK,eol], length 0
13:41:07.760818 IP 10.29.16.211.52545 > 192.168.1.1.25: Flags
 ↪  [S], seq 1228241491, win 65535, options [mss 1460,nop,wscale
 ↪  6,nop,nop,TS val 1308418052 ecr 0,sackOK,eol], length 0
13:41:08.765365 IP 10.29.16.211.52545 > 192.168.1.1.25: Flags
 ↪  [S], seq 1228241491, win 65535, options [mss 1460,nop,wscale
 ↪  6,nop,nop,TS val 1308419052 ecr 0,sackOK,eol], length 0
```

1.4 Find a way to capture all TCP packets, where the RST flag is set (abnormal connection closure).

```
sudo /usr/local/Cellar/tcpdump/4.99.1/bin/tcpdump -n "tcp and
 ↪  tcp[tcpflags] == tcp-rst"

tcpdump: verbose output suppressed, use -v[v]... for full
 ↪  protocol decode
listening on en0, link-type EN10MB (Ethernet), snapshot length
 ↪  262144 bytes
13:49:38.253176 IP 17.253.57.204.443 > 10.29.16.211.52618: Flags
 ↪  [R], seq 2779098861, win 0, length 0
13:49:38.255489 IP 17.253.57.204.443 > 10.29.16.211.52618: Flags
 ↪  [R], seq 2779098861, win 0, length 0
```

1.5 Set up `tcpdump` to capture HTTP packets only, and save the captured packets to a file. Browse the web for a few minutes and stop capturing. Then use the `tshark` utility to open and print the captured packets, and use common shell utilities to extract a list of all distinct destination hosts/IPs.

Own IP: 192.168.0.38

```
tshark -r http.pcap | cut -d '→' -f 2 | cut -d ' ' -f2 | sort -u
 ↪  | awk '!/192.168.0.38/'

188.184.21.108
34.223.124.45
```

We could also do this:

```
tshark -r http.pcap | cut -d'→' -f 2 | grep 'GET' | cut -d ' '
 ↪  -f2 | sort -u
```

but we may overlook a request that happens through JS that uses another

HTTP method

2. Application analysis

There are 3 things to find: a magic password, and 2 operations the program executes.

I'm using a mac, so I can't run the application, so I wasn't able to use `ltrace` or `strace`. I used `objdump` and a hex-viewer and the following Decompiler: https://dogbolt.org

Password:

- "pass word" `strcat("pass", " word")`

Operations:

- Opens a socket and starts listening for connections (server)
- Opens `/etc/passwd` in read mode