

Report: Network Traffic Sniffing

The program mydump.go provides implementation to carry live traffic or read from a tcp dump .pcap file with string pattern and BPF Filtering options.

Output and Program Usage

Program Usage:

\$sudo go run mydump.go

If we don't provide any parameters, it will take the default ethernet device and start sniffing live traffic

Sample Output:

```
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run mydump3.go

No or unknown parameters; Initializing dump for Default Ethernet Name:  en0

2021-03-04 22:20:46.112133 08:f1:ea:5e:4a:00 -> 78:31:c1:d5:0e:42 type 0x800 len 119 173.194.68.189:443(https) -> 172.24.18.225:59538 TCP ACK PSH
00000000 17 03 03 00 30 aa 9b 03 c3 76 71 55 ad b2 f4 a0 [...]0...vqU....|
00000010 2f 0b e7 5f 52 01 63 b5 01 ea a5 15 ca a0 18 25 [/...]R.c.....%|
00000020 f2 ac 44 dc 92 12 a4 f7 16 9e 70 c4 07 4d d6 74 [...]D.....p..M.t|
00000030 3a d2 b4 a9 21 [...]!:...!!

2021-03-04 22:20:46.112139 08:f1:ea:5e:4a:00 -> 78:31:c1:d5:0e:42 type 0x800 len 164 162.159.134.234:443(https) -> 172.24.18.225:59583 TCP ACK PSH
00000000 17 03 03 00 69 64 72 55 83 14 39 ba fd a7 93 94 [...]idrU..9.....|
00000010 3f 1b d2 77 26 91 43 40 6e 09 03 0e f8 ef 0d 1d [?]..w&.C@n.....|
00000020 d5 f6 5b ca 7c a2 70 07 87 a5 81 44 0c 6a f8 4d [...]..|.p....D.j.M|
00000030 3b b2 61 31 eb 9b b5 9a 37 c1 2d 79 e1 6e b4 a8 [;..a1....7..y.n..|
00000040 44 21 76 4e de c7 22 ce 23 fe 4e 41 1c ec 7d 8a [D|vN..."#.NA..|]
00000050 2e e2 56 d3 a3 fd 6d d5 c5 0e 4f 61 a5 3d 3d eb [..v...m...0a==..|
00000060 29 c2 a5 0f 26 27 14 b3 13 c8 40 26 ee c6 [)....6'....@&..|
```

\$sudo go run mydump.go -i en0

For the “i” parameter, it takes the ethernet device. If the ethernet device does not exist, the program stops giving an error “ No such device exists (BIOCKETIF failed: Device not configured)”.

Sample Output:

```
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run mydump3.go -i en0
i: en0

2021-03-04 22:21:26.045950 78:31:c1:d5:0e:42 -> aa:bb:cc:dd:ee:ff type 0x800 len 54 172.24.18.225:59541 -> 162.247.243.147:443(https) TCP ACK

2021-03-04 22:21:26.049102 08:f1:ea:5e:4a:00 -> 78:31:c1:d5:0e:42 type 0x800 len 119 173.194.175.189:443(https) -> 172.24.18.225:59568 TCP ACK PSH
00000000 17 03 03 00 30 ba 93 38 de d1 67 e6 29 dd 2c f1 [...]..0..8..g.)...|
00000010 f5 7e 1a da c2 1e b9 1e d3 70 01 0f e4 d1 c3 7d [..~.....p.....|]
00000020 13 a3 6c 29 9f f1 50 54 88 7d d8 67 76 da 05 03 [...]..L)..PT.}.gv...|
00000030 0f 0d 7c 52 4e [...]..|RN|

2021-03-04 22:21:26.049102 78:31:c1:d5:0e:42 -> aa:bb:cc:dd:ee:ff type 0x800 len 66 172.24.18.225:59568 -> 173.194.175.189:443(https) TCP ACK

2021-03-04 22:21:26.054094 08:f1:ea:5e:4a:00 -> 78:31:c1:d5:0e:42 type 0x800 len 60 162.247.243.147:443(https) -> 172.24.18.225:59541 TCP ACK
```

\$sudo go run mydump.go -r hw1.pcap

For the “r” parameter, it takes the file parameter. If the file parameter is wrong, it will exit.

```
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run mydump3.go -r ~/Downloads/SBU-US/Network\ Security/assignment-2/go_code/hw1.pcap
r: /Users/admin/Downloads/SBU-US/Network Security/assignment-2/go_code/hw1.pcap

2013-01-12 11:37:42.871346 c4:3d:c7:17:6f:9b -> ff:ff:ff:ff:ff:ff type 0x806 len 60
2013-01-12 11:38:02.227995 c4:3d:c7:17:6f:9b -> 01:00:5e:7f:ff:fa type 0x800 len 342 192.168.0.1:1901(fjicl-tep-a) -> 239.255.255.250:1900(ssdp) UDP
00000000 4e 4f 54 49 46 59 20 2a 20 48 54 54 50 2f 31 2e |NOTIFY * HTTP/1.1|
00000010 31 0d 0a 48 4f 53 54 3a 20 32 33 39 2e 32 35 35 |1..HOST: 239.255|
00000020 2e 32 35 35 2e 32 35 30 3a 31 39 30 30 0d 0a 43 |.255.250:1900..C|
00000030 61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61 |ache-Control: ma|
00000040 78 2d 61 67 65 3d 33 36 30 30 0d 0a 4c 6f 63 61 |x-age=3600..Loca|
00000050 74 69 6f 6e 3a 20 68 74 74 70 3a 2f 2f 31 39 32 |tion: http://192|
00000060 2e 31 36 38 2e 30 2e 31 3a 38 30 2f 52 6f 6f 74 |.168.0.1:80/Root|
00000070 44 65 76 69 63 65 2e 78 6d 6c 0d 0a 4e 54 3a 20 |Device.xml..NT: |
00000080 75 75 69 64 3a 75 70 6e 70 2d 49 6e 74 65 72 6e |uid:upnp-Intern|
00000090 65 74 47 61 74 65 77 61 79 44 65 76 69 63 65 2d |etGatewayDevice-|
000000a0 31 5f 30 2d 63 34 33 64 63 37 31 37 36 66 39 62 |1_0-c43dc7176f9b|
000000b0 0d 0a 55 53 4e 3a 20 75 75 69 64 3a 75 70 6e 70 |..USN: uid:upnp|
000000c0 2d 49 6e 74 65 72 6e 65 74 47 61 74 65 77 61 79 |-InternetGateway|
000000d0 44 65 76 69 63 65 2d 31 5f 30 2d 63 34 33 64 63 |Device-1_0-c43dc|
000000e0 37 31 37 36 66 39 62 0d 0a 4e 54 53 3a 20 73 73 |7176f9b..NTS: ss|
000000f0 64 70 3a 61 6c 69 76 65 0d 0a 53 65 72 76 65 72 |dp:alive..Server|
00000100 3a 20 55 50 6e 50 2f 31 2e 30 20 55 50 6e 50 2f |: UPnP/1.0 UPnP/I|
00000110 31 2e 30 20 55 50 6e 50 2d 44 65 76 69 63 65 2d |1.0 UPnP-Device-|
00000120 48 6f 73 74 2f 31 2e 30 0d 0a 0d 0a |Host/1.0....|
```

\$sudo go run mydump.go -i en0 -r hw1.pcap

If both the “i” and “r” parameters are given, the file will be taken into consideration and the program will start reading from the valid pcap file provided.

Sample Output:

```
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run mydump3.go -i en0 -r ~/Downloads/SBU-US/Network\ Security/assignment-2/go_code/hw1.pcap
r: /Users/admin/Downloads/SBU-US/Network Security/assignment-2/go_code/hw1.pcap

2013-01-12 11:37:42.871346 c4:3d:c7:17:6f:9b -> ff:ff:ff:ff:ff:ff type 0x806 len 60
2013-01-12 11:38:02.227995 c4:3d:c7:17:6f:9b -> 01:00:5e:7f:ff:fa type 0x800 len 342 192.168.0.1:1901(fjicl-tep-a) -> 239.255.255.250:1900(ssdp) UDP
00000000 4e 4f 54 49 46 59 20 2a 20 48 54 54 50 2f 31 2e |NOTIFY * HTTP/1.1|
00000010 31 0d 0a 48 4f 53 54 3a 20 32 33 39 2e 32 35 35 |1..HOST: 239.255|
00000020 2e 32 35 35 2e 32 35 30 3a 31 39 30 30 0d 0a 43 |.255.250:1900..C|
00000030 61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61 |ache-Control: ma|
00000040 78 2d 61 67 65 3d 33 36 30 30 0d 0a 4c 6f 63 61 |x-age=3600..Loca|
00000050 74 69 6f 6e 3a 20 68 74 74 70 3a 2f 2f 31 39 32 |tion: http://192|
00000060 2e 31 36 38 2e 30 2e 31 3a 38 30 2f 52 6f 6f 74 |.168.0.1:80/Root|
00000070 44 65 76 69 63 65 2e 78 6d 6c 0d 0a 4e 54 3a 20 |Device.xml..NT: |
00000080 75 75 69 64 3a 75 70 6e 70 2d 49 6e 74 65 72 6e |uid:upnp-Intern|
00000090 65 74 47 61 74 65 77 61 79 44 65 76 69 63 65 2d |etGatewayDevice-|
000000a0 31 5f 30 2d 63 34 33 64 63 37 31 37 36 66 39 62 |1_0-c43dc7176f9b|
000000b0 0d 0a 55 53 4e 3a 20 75 75 69 64 3a 75 70 6e 70 |..USN: uid:upnp|
000000c0 2d 49 6e 74 65 72 6e 65 74 47 61 74 65 77 61 79 |-InternetGateway|
000000d0 44 65 76 69 63 65 2d 31 5f 30 2d 63 34 33 64 63 |Device-1_0-c43dc|
000000e0 37 31 37 36 66 39 62 0d 0a 4e 54 53 3a 20 73 73 |7176f9b..NTS: ss|
000000f0 64 70 3a 61 6c 69 76 65 0d 0a 53 65 72 76 65 72 |dp:alive..Server|
00000100 3a 20 55 50 6e 50 2f 31 2e 30 20 55 50 6e 50 2f |: UPnP/1.0 UPnP/I|
00000110 31 2e 30 20 55 50 6e 50 2d 44 65 76 69 63 65 2d |1.0 UPnP-Device-|
00000120 48 6f 73 74 2f 31 2e 30 0d 0a 0d 0a |Host/1.0....|
```

Here you can see that the above output and this output is the same as both are reading from the same file. This command has overridden the interface input provided by the user.

\$sudo go run mydump.go -i en0 -r hw1.pcap -s "Gateway"

"S" parameter checks whether the packet payload has the provided string input or not. It will only dump the packet if the string is present in the packet payload.

Sample Output

```
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run mydump3.go -i en0 -r ~/Downloads/SBU-US/Network\ Security/assignment-2/go_code/hw1.pcap -s "Gateway"
r: /Users/admin/Downloads/SBU-US/Network Security/assignment-2/go_code/hw1.pcap
2013-01-12 11:38:02.227995 c4:3d:c7:17:6f:9b -> 01:00:5e:7f:ff:fa type 0x800 len 342 192.168.0.1:1901(fjicl-tep-a) -> 239.255.255.250:1900(ssdp) UDP
00000000 4e 4f 54 49 46 59 20 2a 20 48 54 54 50 2f 31 2e [NOTIFY * HTTP/1.]
00000010 31 0d 0a 48 4f 53 54 3a 20 32 33 39 2e 32 35 35 [1..HOST: 239.255]
00000020 2e 32 35 35 2e 32 35 30 3a 31 39 30 30 0d 0a 43 [255.250:1900..C]
00000030 61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61 [ache-Control: ma]
00000040 78 2d 61 67 65 3d 33 36 30 30 0d 0a 4c 6f 63 61 [x-age=3600..Local]
00000050 74 69 6f 6e 3a 20 68 74 74 70 3a 2f 2f 31 39 32 [tion: http://192]
00000060 2e 31 36 38 2e 30 2e 31 3a 38 30 2f 52 6f 6f 74 [168.0.1:80/Root]
00000070 44 65 76 69 63 65 2e 78 6d 6c 0d 0a 4e 54 3a 20 [Device.xml..NT: ]
00000080 75 75 69 64 3a 75 70 6e 70 2d 49 6e 74 65 72 6e [uid:upnp-Intern]
00000090 65 74 47 61 74 65 77 61 79 44 65 76 69 63 65 2d [etGatewayDevice-I]
000000a0 31 5f 30 2d 63 34 33 64 63 37 31 37 36 66 39 62 [1_0-c43dc7176f9b]
000000b0 0d 0a 55 53 4e 3a 20 75 75 69 64 3a 75 70 6e 70 [1..USN: uid:upnp]
000000c0 2d 49 6e 74 65 72 6e 65 74 47 61 74 65 77 61 79 [-InternetGateway]
000000d0 44 65 76 69 63 65 2d 31 5f 30 2d 63 34 33 64 63 [Device-1_0-c43dc]
000000e0 37 31 37 36 66 39 62 0d 0a 4e 54 53 3a 20 73 73 [7176f9b..NTS: ss]
000000f0 64 70 3a 61 6c 69 76 65 0d 0a 53 65 72 76 65 72 [dp:alive..Server]
00000100 3a 20 55 50 6e 50 2f 31 2e 30 20 55 50 6e 50 2f [1: UPnP/1.0 UPnP/]
00000110 31 2e 30 20 55 50 6e 50 2d 44 65 76 69 63 65 2d [1.0 UPnP-Device-I]
00000120 48 6f 73 74 2f 31 2e 30 0d 0a 0d 0a [Host/1.0....]
```

\$sudo go run mydump.go -i en0 -r hw1.pcap "icmp"

Expression at the end of the command gives the BPF Filter which needs to be applied while dumping packets. If no such value is provided, no BPF Filter is applied.

Sample Output

```
Terminal: Local +
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run ./mydump3.go -r ~/Downloads/SBU-US/Network\ Security/assignment-2/go_code/hw1.pcap icmp
r: /Users/admin/Downloads/SBU-US/Network Security/assignment-2/go_code/hw1.pcap
2013-01-14 12:42:31.752299 c4:3d:c7:17:6f:9b -> 00:0c:29:e9:94:8e type 0x800 len 90 1.234.31.20 -> 192.168.0.200 ICMPv4
00000000 45 00 00 4c eb 4a 00 00 2f 01 bd f8 01 ea 1f 14 [E..L.J./.....]
00000010 c0 a8 00 c8 03 0a 95 2a 00 00 00 00 45 00 00 30 [.....*....E..0]
00000020 00 00 40 00 2e 06 6a 5a c0 a8 00 c8 01 ea 1f 14 [..@...jZ.....]
00000030 00 50 7b 81 bd cd 09 c6 3a 35 22 b0 70 12 39 08 [P{.....5".p.9.]
00000040 11 ab 00 00 02 04 05 b4 01 01 04 02 [.....]
```

There could be any combination of the parameters given. Only expression or string parameters are also applicable if i and r parameters are not provided. If neither i nor r parameters are provided, string or BPF expression filter will be applied on the dump provided by the default ethernet device.

Code/ Implementation Walkthrough

1. Accept input parameter values , -i (for interface name), -r (for .pcap file), -s (for string) and flag.Args (for expression)
2. Depending on the input parameters, deciding whether to read from file, read online from device, or if unknown parameters, taking default device and start sniffing online
3. For default devices, **pcap.FindAllDevs()** returns all available interfaces. Took the first one in the list as the default.
4. Used **pcap.OpenOffline(filePointer)** to read from the pcap file
5. Used **pcap.OpenLive(interfacePointer, 1600, true, pcap.BlockForever)** to read from interface

6. After starting the handle to read from respective locations:

- a. Setting the BPF Filter using:

handle.SetBPFFilter(expressionPointer)

- b. Extracting each packet:

gopacket.NewPacketSource(handle, handle.LinkType())

- c. Running a loop for each packet

- d. If the string pattern parameter is present, checking the string in the payload to move forward:

strings.Contains(string(ip.Payload), stringValue)

- e. Printing time:

**time := packet.Metadata().Timestamp.Format("2006-01-02
15:04:05.000000")**

- f. Printing ethernet layer details -> SrcMac, DstMac, hex value of Ethernet Type:

ethernetPacket, _ := ethernetPacketLayer.(*layers.Ethernet)

- g. Printing packet length:

packet.Metadata().Length

- h. Printing IP layer details if present-> SrcIp, DstIp:

ipLayer := packet.Layer(layers.LayerTypeIPv4)

- i. Printing TCP/UDP layer port details if it is a TCP/UDP packet

tcpLayer := packet.Layer(layers.LayerTypeTCP)

udpLayer := packet.Layer(layers.LayerTypeUDP)

- j. Printing IP protocol:

ip.Protocol

- k. Printing TCP flags which are set to true if it is a TCP packet: **tcp.SYN**, **tcp.ACK**, etc.

- l. Printing ethernet layer payload as hex dump.

hex.Dump(ethernetPacket.Payload)

References

1. <https://pkg.go.dev/github.com/alicebob/pcap>
2. <https://www.devdungeon.com/content/packet-capture-injection-and-analysis-gopacket>
3. <https://pkg.go.dev/github.com/google/gopacket@v1.1.19/pcap>
4. <https://danielmiessler.com/study/tcpdump/>
5. <https://pkg.go.dev/github.com/google/gopacket@v1.1.19>