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.

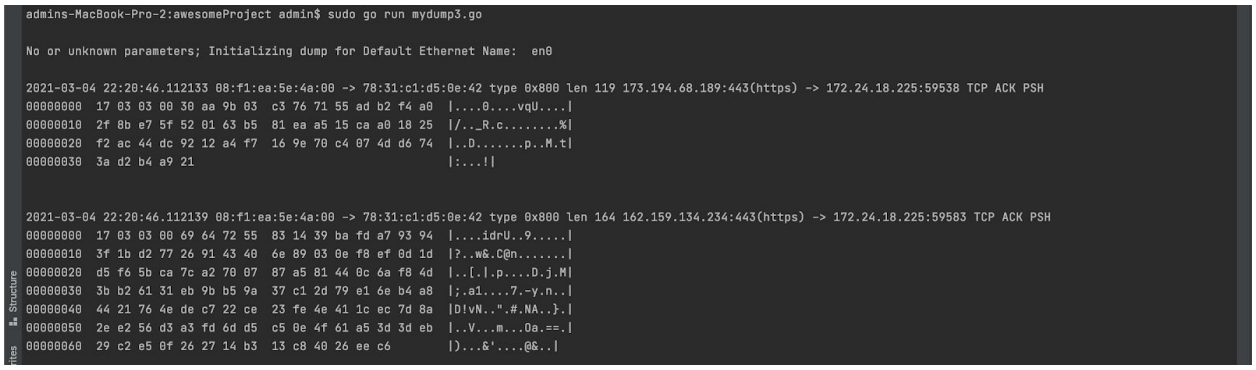
Output and Program Usage

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:

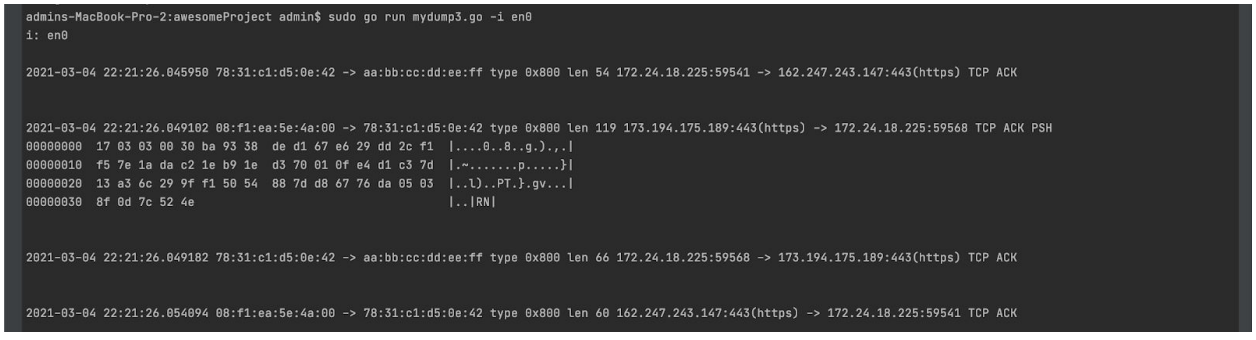


```
admins-MacBook-Pro-2:awesomeProject admin$ sudo go run mydump3.go

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 8b e7 5f 52 01 63 b5  81 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 89 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...Oa.==.|
00000060  29 c2 e5 0f 26 27 14 b3  13 c8 40 26 ee c6        |)...&'....@&..|
```

$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 (BIOCSETIF failed: Device not configured)".

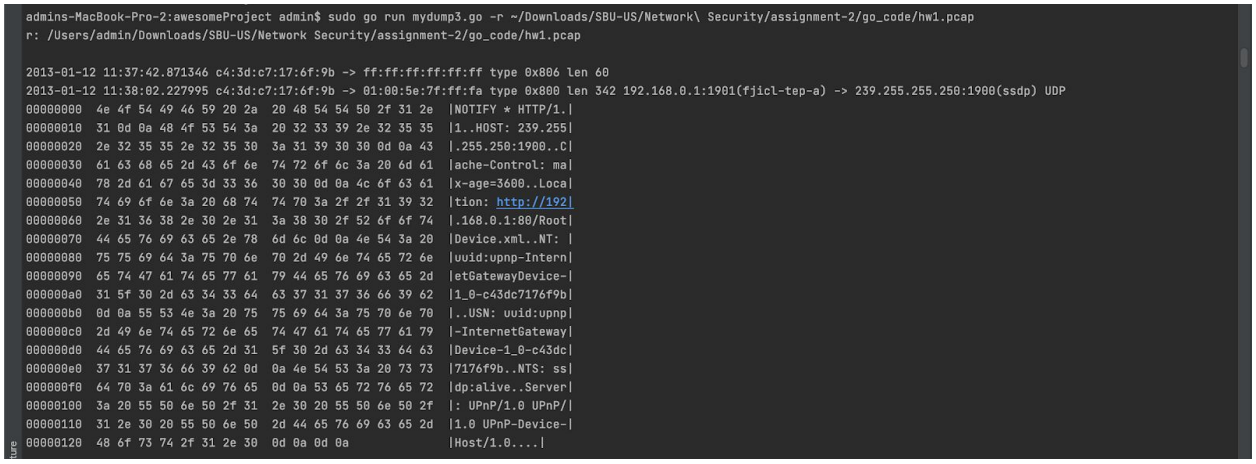
Sample Output:



```
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  8f 0d 7c 52 4e                                    |..|RN|


2021-03-04 22:21:26.049182 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

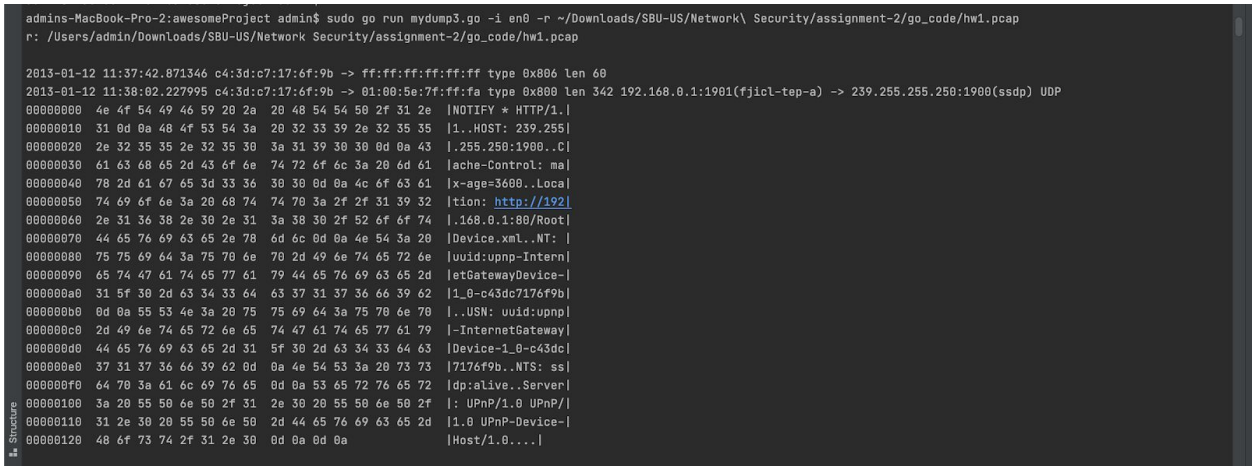
For the “r” parameter, it takes the file parameter. If the file parameter is wrong, it will exit.



$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:

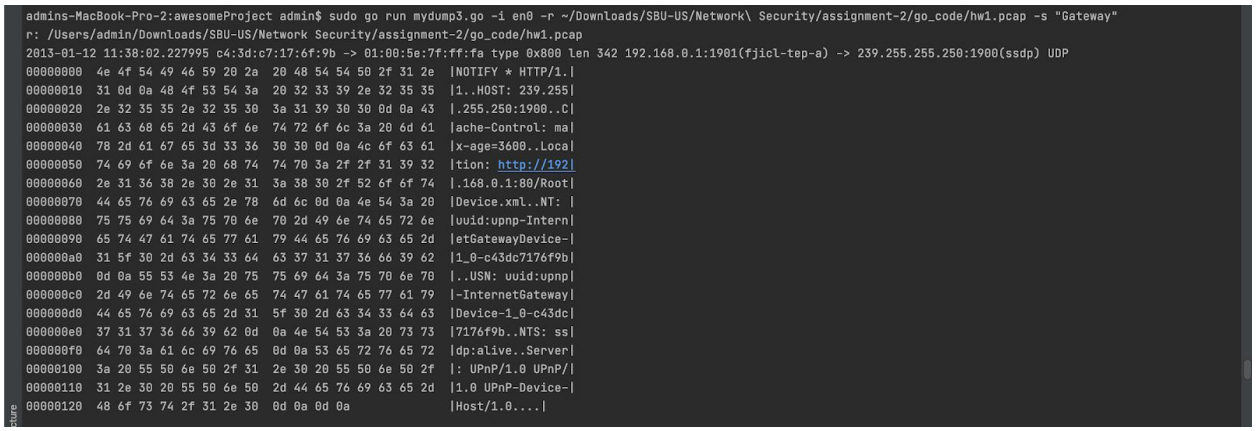


Here you can see that the above output and this output is the same as both are reading from the same file. This command has overridden the interface input provided by the user.

$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.

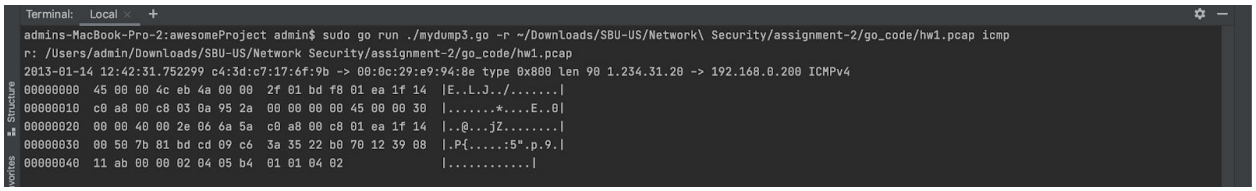
Sample Output



$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



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