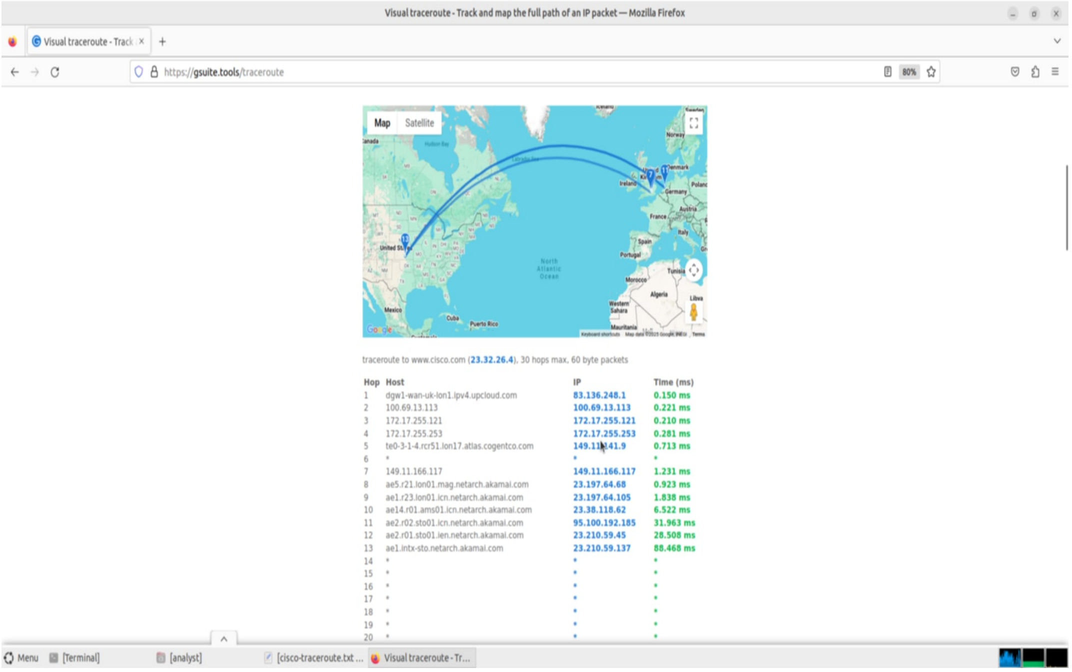
CyberOps Labs Submission

Lab 5.1.5 - Tracing a Route

- Learned how to trace packet paths across multiple routers and ISPs.
- Observed how geographical location changes the path results.

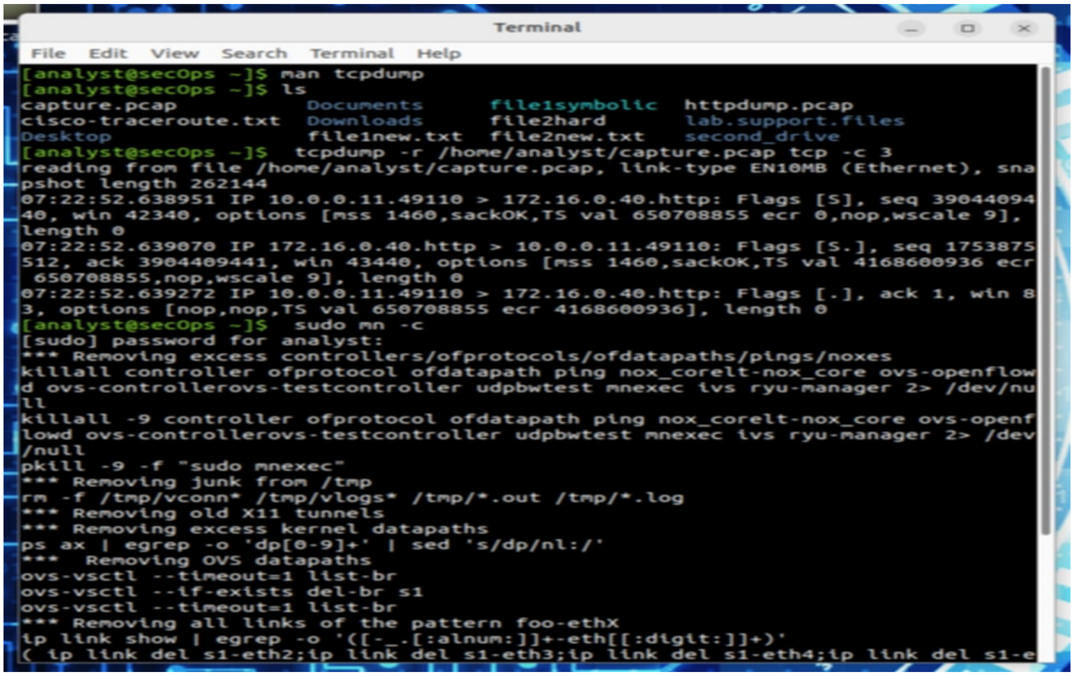

Visual traceroute - Track and map the full path of an IP packet — Mozilla Firefox
Visual traceroute - Track
https://gsuite.tools/traceroute
80%
Map
Satellite
Menu
[Terminal]
[analyst]
[cisco-traceroute.txt ...
Visual traceroute - Tr...

```
traceroute to www.cisco.com (23.32.26.4), 30 hops max, 60 byte packets

Hop  Host                                        IP               Time (ms)
1    dgw1-wan-uk-lon1.ipv4.upcloud.com           83.136.248.1     0.150 ms
2    100.69.13.113                               100.69.13.113    0.221 ms
3    172.17.255.121                              172.17.255.121   0.210 ms
4    172.17.255.253                              172.17.255.253   0.281 ms
5    te0-3-1-4.rcr51.lon17.atlas.cogentco.com    149.11.141.9     0.713 ms
6    *                                           *                *
7    149.11.166.117                              149.11.166.117   1.231 ms
8    ae5.r21.lon01.mag.netarch.akamai.com        23.197.64.68     0.923 ms
9    ae1.r23.lon01.icn.netarch.akamai.com        23.197.64.105    1.838 ms
10   ae14.r01.ams01.icn.netarch.akamai.com       23.38.118.62     6.522 ms
11   ae2.r02.sto01.icn.netarch.akamai.com        95.100.192.185   31.963 ms
12   ae2.r01.sto01.ien.netarch.akamai.com        23.210.59.45     28.508 ms
13   ae1.intx-sto.netarch.akamai.com             23.210.59.137    88.468 ms
14   *                                           *                *
15   *                                           *                *
16   *                                           *                *
17   *                                           *                *
18   *                                           *                *
19   *                                           *                *
20   *                                           *                *
```

Lab 9.2.6 - Using Wireshark to Observe the TCP 3-Way Handshake

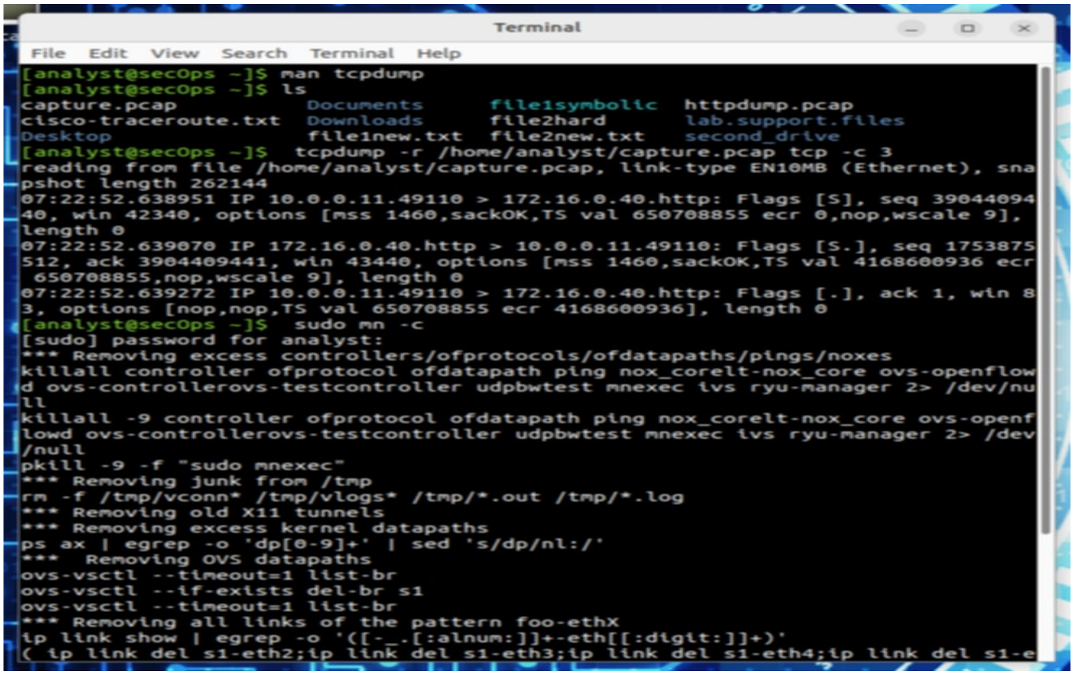
- Learned how SYN, SYN-ACK, and ACK packets establish a TCP connection.
- Observed tcpdump output of the handshake process.



```
[analyst@secOps ~]$ man tcpdump
[analyst@secOps ~]$ ls
capture.pcap           Documents      file1symbolic   httpdump.pcap
cisco-traceroute.txt   Downloads      file2hard       lab.support.files
Desktop                file1new.txt   file2new.txt    second_drive
[analyst@secOps ~]$  tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet), sna
pshot length 262144
07:22:52.638951 IP 10.0.0.11.49110 > 172.16.0.40.http: Flags [S], seq 39044094
40, win 42340, options [mss 1460,sackOK,TS val 650708855 ecr 0,nop,wscale 9],
length 0
07:22:52.639070 IP 172.16.0.40.http > 10.0.0.11.49110: Flags [S.], seq 1753875
512, ack 3904409441, win 43440, options [mss 1460,sackOK,TS val 4168600936 ecr
 650708855,nop,wscale 9], length 0
07:22:52.639272 IP 10.0.0.11.49110 > 172.16.0.40.http: Flags [.], ack 1, win 8
3, options [nop,nop,TS val 650708855 ecr 4168600936], length 0
[analyst@secOps ~]$  sudo mn -c
[sudo] password for analyst:
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflow
d ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/nu
ll
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openf
lowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev
/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --if-exists del-br s1
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
( ip link del s1-eth2;ip link del s1-eth3;ip link del s1-eth4;ip link del s1-e
```

Lab 10.6.7 - Using Wireshark to Examine HTTP and HTTPS Traffic

- Learned how HTTP traffic exposes login credentials in plaintext.
- Understood that HTTPS encrypts traffic and protects sensitive data.



```
[analyst@secOps ~]$ man tcpdump
[analyst@secOps ~]$ ls
capture.pcap           Documents      file1symbolic   httpdump.pcap
cisco-traceroute.txt   Downloads      file2hard       lab.support.files
Desktop                file1new.txt   file2new.txt    second_drive
[analyst@secOps ~]$  tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet), sna
pshot length 262144
07:22:52.638951 IP 10.0.0.11.49110 > 172.16.0.40.http: Flags [S], seq 39044094
40, win 42340, options [mss 1460,sackOK,TS val 650708855 ecr 0,nop,wscale 9],
length 0
07:22:52.639070 IP 172.16.0.40.http > 10.0.0.11.49110: Flags [S.], seq 1753875
512, ack 3904409441, win 43440, options [mss 1460,sackOK,TS val 4168600936 ecr
 650708855,nop,wscale 9], length 0
07:22:52.639272 IP 10.0.0.11.49110 > 172.16.0.40.http: Flags [.], ack 1, win 8
3, options [nop,nop,TS val 650708855 ecr 4168600936], length 0
[analyst@secOps ~]$  sudo mn -c
[sudo] password for analyst:
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflow
d ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/nu
ll
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openf
lowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev
/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --if-exists del-br s1
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
( ip link del s1-eth2;ip link del s1-eth3;ip link del s1-eth4;ip link del s1-e
```