

## Preparazione per CyberOps.

## Usare Wireshark per Osservare l'Handshake a 3 Vie TCP.

**Qualche screen per il procedimento eseguito.**

```

analyst@secOps ~$ sudo lab.support.files/scripts/cyberops-topo.py
[sudo] password for analyst:

CyberOPS Topology:

      | R1 |-----| H4 |
      |
      |
      |
      |-----| S1 |-----|
      |
      |
      |
      |-----| H1 | | H2 | | H3 |
      |
      |-----|

```

```

*** Add links
*** Creating network
*** Adding hosts:
H1 H2 H3 H4 R1
*** Adding switches:
S1
*** Adding links:
(H1, s1) (H2, s1) (H3, s1) (H4, R1) (s1, R1)
*** Configuring hosts
H1 H2 H3 H4 R1
*** Starting controller

*** Starting 1 switches
S1...
*** Routing Table on Router:
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0         0.0.0.0         255.255.255.0   U        0    0    0 R1-eth1
172.16.0.0       0.0.0.0         255.240.0.0     U        0    0    0 R1-eth2

*** Starting CLI:
mininet> xterm H1
mininet> xterm H4

```

```
[root@secOps analyst]# /home/analyst/lab.support.files/scripts/reg_server_start
.sh
[root@secOps analyst]#
```

```
[root@secOps analyst]# su analyst
[analyst@secOps ~]$ firefox &
[1] 641
[analyst@secOps ~]$
```

### Node H4

## Node H1

```
[analyst@secOps ~]$ sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.pcap
[sudo] password for analyst:
tcpdump: listening on H1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
50 packets captured
55 packets received by filter
0 packets dropped by kernel
```

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

Thank you for using nginx.

## Parte 2: Analizzare i Pacchetti usando Wireshark.

Seguo la procedura fino ad ottenere questa schermata.

Filter: tcp		Expression...		Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info
15	4.492500	10.0.0.11	172.16.0.40	TCP	74	56438 → 80 [SYN] Seq=0 Win=29200 Len=0
16	4.492539	172.16.0.40	10.0.0.11	TCP	74	80 → 56438 [SYN, ACK] Seq=0 Ack=1 Win=28
17	4.492547	10.0.0.11	172.16.0.40	TCP	66	56438 → 80 [ACK] Seq=1 Ack=1 Win=29696
18	4.492717	10.0.0.11	172.16.0.40	HTTP	377	GET / HTTP/1.1
19	4.492730	172.16.0.40	10.0.0.11	TCP	66	80 → 56438 [ACK] Seq=1 Ack=312 Win=3020
20	4.718404	172.16.0.40	10.0.0.11	TCP	304	80 → 56438 [PSH, ACK] Seq=1 Ack=312 Win=

- 1) **Qual è il numero di porta TCP di origine?** Il numero di porta d'origine è 56438.
- 2) **Come classificherei la porta di origine?** La classificherei come porta dinamica/privata.
- 3) **Qual è il numero di porta TCP di destinazione?** Il numero di porta di destinazione è 80.
- 4) **Come classificherei la porta di destinazione?** La classificherei come porta nota dedicata ai servizi Web (al server Web **nginx** in questo caso).
- 5) **Quale flag è impostato?** È impostato il flag 0x002 (SYN).
- 6) **A quale valore è impostato il numero di sequenza relativo?** È impostato a 0.

## Frame 2:

- 7) **Quali sono i valori delle porte di origine e destinazione?** La porta di origine è 80 e la porta di destinazione è la 56438.
- 8) **Quali flag sono impostati?** 0x012 (SYN, ACK).
- 9) **A quali valori sono impostati i numeri relativi di sequenza e acknowledgment?** Il numero relativo di sequenza è impostato su 0 mentre il numero di acknowledgement è impostato su 1.

```
Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
```

- 10) Esaminare il terzo e ultimo pacchetto dell'handshake. **Quale flag è impostato?** Il flag impostato è 0x010 (ACK)

```
Sequence number: 1 (relative sequence number)
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
1000 .... = Header Length: 32 bytes (8)
▶ Flags: 0x010 (ACK)
```

## Parte 3: Visualizzare i pacchetti usando tcpdump

- 11) Cosa fa l'opzione -r?

L'opzione -r permette di leggere i pacchetti da un file.

```
-r file
Read packets from file (which was created with the -w option or by other tools that write pcap or pcap-ng files). Standard input is used if file is '-'.
```

```
[analyst@secOps ~]$ tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet)
08:27:31.663802 IP 10.0.0.11.56438 > 172.16.0.40.http: Flags [S], seq 591686644, win 29200, options [mss 1460,sackOK,TS val 1762803277 ecr 0,nop,wscale 9], length 0
08:27:31.663841 IP 172.16.0.40.http > 10.0.0.11.56438: Flags [S.], seq 3175599598, ack 591686645, win 28960, options [mss 1460,sackOK,TS val 1098819986 ecr 1762803277,nop,wscale 9], length 0
08:27:31.663849 IP 10.0.0.11.56438 > 172.16.0.40.http: Flags [.], ack 1, win 58, options [nop,nop,TS val 1762803277 ecr 1098819986], length 0
[analyst@secOps ~]$
```

Infine, navigo sul terminale dove ho attivato mininet e inserisco **quit** per terminare la sessione e **sudo mn -c** per pulire i processi avviati da mininet.

```
mininet> quit
*** Stopping 0 controllers

*** Stopping 2 terms
*** Stopping 5 links
.....
*** Stopping 1 switches
s1
*** Stopping 5 hosts
H1 H2 H3 H4 R1
*** Done
```

```
*** Done
[analyst@sec0ps ~]$ sudo mn -c
[sudo] password for analyst:
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec
ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec
ec ivs 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 --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_.:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

## Domande di riflessione.

- Ci sono centinaia di filtri disponibili in Wireshark. Una rete di grandi dimensioni potrebbe avere numerosi filtri e molti tipi diversi di traffico.

**Elenca tre filtri che potrebbero essere utili a un amministratore di rete.**

Su Wireshark si può filtrare attraverso i **protocolli** (per esempio: TCP, HTTP, DNS, ICMP), attraverso **indirizzo IP** sorgente e **IP destinatario** e attraverso **porte specifiche**.

- **In quali altri modi Wireshark potrebbe essere utilizzato in una rete di produzione?**

Wireshark può essere utilizzato per monitorare i tempi di risposta di applicazioni e servizi, per identificare problemi di connettività tra dispositivi e per l'analisi di incidenti di sicurezza.