

Esercizio_S10_L2_Wireshark

Consegna

Obiettivi:

- Parte 1 Preparare gli Host per Catturare il Traffico
- Parte 2 Analizzare i Pacchetti usando Wireshark
- Parte 3 Visualizzare i Pacchetti usando tcpdump

Contesto / Scenario

In questo laboratorio, userai Wireshark per catturare ed esaminare i pacchetti generati tra il browser del PC che utilizza il protocollo HTTP (HyperText Transfer Protocol) e un server web, come www.google.com.

Quando un'applicazione, come HTTP o FTP (File Transfer Protocol), si avvia per la prima volta su un host, TCP utilizza l'handshake a tre vie per stabilire una sessione TCP affidabile tra i due host. Ad esempio, quando un PC utilizza un browser web per navigare in internet, viene avviato un handshake a tre vie e viene stabilita una sessione tra l'host del PC e il server web. Un PC può avere più sessioni TCP attive simultaneamente con vari siti web.

Risorse Richieste: Macchina virtuale CyberOps Workstation

Svolgimento

Preparazione Ambiente

Dopo aver avviato la Cyberops Workstation ed effettuato il login **User:cyberops** il primo passo da svolgere consisteva nell'avviare Mininet tramite:

```
sudo lab.support.files/scripts/cyberops_topo.py
```

```
[analyst@secOps ~]$ sudo lab.support.files/scripts/cyberops_topo.py
[sudo] password for analyst:

CyberOPS Topology:

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

*** Adding internal 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 Enabling IP forwarding on 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> 
```

Tramite questo tool è possibile simulare una rete interna di cui possiamo vedere sopra gli **HOST** (H1, H2, H3, H4, R1) e gli indirizzi **IP** di rete.

A questo punto possiamo avviare il terminale dell'host H1 ed H4 tramite:

```
xterm H1
xterm H4
```

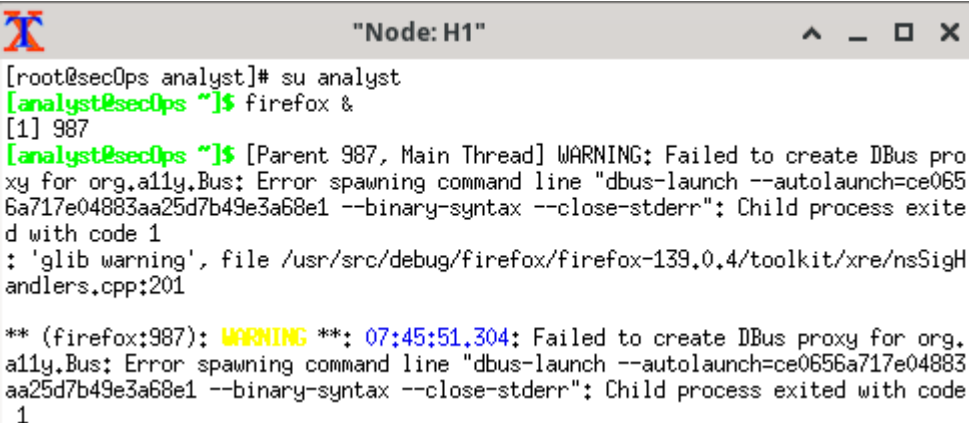
Fase 1 Cattura Traffico - tcpdump

All'interno dell'host H4 avvieremo un webserver tramite il comando /home/analyst/lab.support.files/scripts/reg_server_start.sh

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

Sull'host H1 invece, dopo aver cambiato utente da root ad analyst, avviamo il browser firefox:

[su analyst](#)
[firefox &](#)



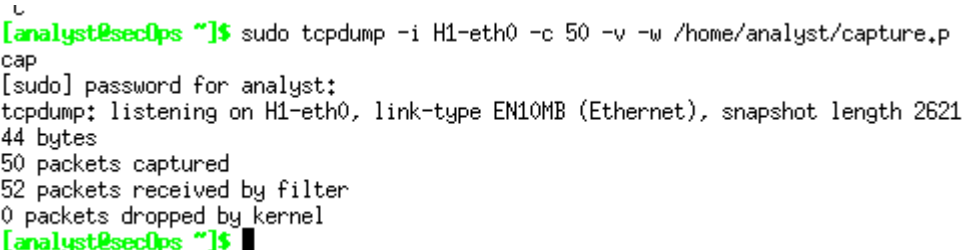
```
"Node: H1"
[root@secOps analyst]# su analyst
[analyst@secOps ~]$ firefox &
[1] 987
[analyst@secOps ~]$ [Parent 987, Main Thread] WARNING: Failed to create Dbus pro
xy for org.a11y.Bus: Error spawning command line "dbus-launch --autolaunch=ce065
6a717e04883aa25d7b49e3a68e1 --binary-syntax --close-stderr": Child process exite
d with code 1
: 'glib warning', file /usr/src/debug/firefox/firefox-139.0.4/toolkit/xre/nsSigH
andlers.cpp:201

** (firefox:987): WARNING **: 07:45:51.304: Failed to create Dbus proxy for org.
a11y.Bus: Error spawning command line "dbus-launch --autolaunch=ce0656a717e04883
aa25d7b49e3a68e1 --binary-syntax --close-stderr": Child process exited with code
1
```

Dopo l'apertura della finestra di Firefox, possiamo avviare una sessione tcpdump nel terminale Node: H1 che invii l'output a un file chiamato [capture.pcap](#).

Con l'opzione -v (verbose mode) è possibile osservare l'avanzamento del capturing. Il nostro scopo sarà catturare 50 pacchetti quindi utilizzeremo il parametro -c 50. Una volta lanciato il programma navigheremo sulla pagina web <http://172.16.0.40> per permettere la cattura di traffico web.

[sudo tcpdump -i H1-eth0 -c 50 -v -w /home/analyst/capture.pcap](#)



```
[analyst@secOps ~]$ sudo tcpdump -i H1-eth0 -c 50 -v -w /home/analyst/capture.p
cap
[sudo] password for analyst:
tcpdump: listening on H1-eth0, link-type EN10MB (Ethernet), snapshot length 2621
44 bytes
50 packets captured
52 packets received by filter
0 packets dropped by kernel
[analyst@secOps ~]$ █
```

Una volta completata la cattura troveremo il file pcap all'interno del percorso scelto. Possiamo ora all'analisi tramite Wireshark!

Fase 2 Analisi pcap - Wireshark

Dopo aver avviato wireshark possiamo aprire il pcap appena creato per analizzarne il contenuto.

`sudo wireshark-gtk &`

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.11	8.8.4.4	DNS	84	Standard
2	0.000046	10.0.0.1	10.0.0.11	ICMP	112	Destina
3	0.000067	10.0.0.11	8.8.4.4	DNS	84	Standard
4	0.000077	10.0.0.1	10.0.0.11	ICMP	112	Destina
5	0.000114	10.0.0.11	209.165.200.235	DNS	84	Standard
6	0.000123	10.0.0.11	209.165.200.235	DNS	84	Standard
7	1.828879	10.0.0.11	172.16.0.40	TCP	74	49192 →
8	1.828910	172.16.0.40	10.0.0.11	TCP	74	80 → 49
9	1.828916	10.0.0.11	172.16.0.40	TCP	66	49192 →
10	1.828952	10.0.0.11	172.16.0.40	HTTP	397	GET / H

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface eth0
Ethernet II, Src: da:dc:39:17:e3:6d (da:dc:39:17:e3:6d), Dst: 10:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 10.0.0.11, Dst: 8.8.4.4
User Datagram Protocol, Src Port: 52479, Dst Port: 53
Domain Name System (query)

Spostandoci all'interno della sezione [Statistics > Protocol Hierarchy](#) possiamo avere una prima panoramica delle tipologie di pacchetti catturati:

Protocol	Percent Packets	Packets	Percent Bytes
Frame	100.0	50	
Ethernet	100.0	50	
Internet Protocol Version 4	100.0	50	
User Datagram Protocol	60.0	30	
Domain Name System	60.0	30	
Transmission Control Protocol	28.0	14	
Hypertext Transfer Protocol	8.0	4	
Line-based text data	4.0	2	
Internet Control Message Protocol	12.0	6	

Procediamo analizzando inizialmente il protocollo TCP filtrando la ricerca tramite l'apposita sezione per rispondere alle seguenti domande relative al primo pacchetto dove viene inizializzata la connessione da H1 verso H5:

No.	Time	Source	Destination	Protocol	Length	Info
7	1.828879	10.0.0.11	172.16.0.40	TCP	74	49192 → 80 [SYN] Seq=0 Win=42340 Len=0 MSS=1460
8	1.829010	172.16.0.40	10.0.0.11	TCP	74	80 → 49192 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0

- Qual è il numero di porta TCP di origine?

Analizzando all'interno della sezione TCP possiamo notare che la source port è la **49192**.

```
► Internet Protocol Version 4, Src: 10.0.0.11, Dst: 172.16.0.40
▼ Transmission Control Protocol Src Port: 49192, Dst Port: 80
    Source Port: 49192
    Destination Port: 80
    [Stream index: 0]
    [Stream Packet Number: 1]
```

- Come classificheresti la porta di origine?

La porta d'origine è chiaramente una porta effimera ovvero una porta non standard utilizzata all'occorrenza per inizializzare connessioni.

- Qual è il numero di porta TCP di destinazione?

Sempre all'interno della stessa sessione possiamo trovare la porta di destinazione: **80**, la porta su cui è presente il webserver.

```
Destination Port: 80
[Stream index: 0]
```

- Come classificheresti la porta di destinazione?

La porta 80 è una porta standard utilizzata solitamente per l'hosting dei servizi web.

- Quale flag è impostato?

Il flag è un flag SYN ovvero il primo flag che compone una comunicazione three-way handshake

- A quale valore è impostato il numero di sequenza relativo?

Il numero di sequenza relativo è impostato a 0 in quanto è per l'appunto il primo pacchetto della comunicazione iniziata dal client verso uno specifico server.

```

▶ [Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number 0 (relative sequence number)
Sequence Number (raw): 154230802
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0

```

Spostiamoci ora all'interno del secondo pacchetto per poter rispondere alle seguenti domande:

```

8 1.828910 172.16.0.40 10.0.0.11 TCP 74 80 → 49192 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=

```

Quali sono i valori delle porte di origine e destinazione?

Ovviamente il secondo pacchetto avrà le porte invertite in quanto rappresenta una risposta del webserver verso l'host H1.

La source port sarà dunque la 80 mentre la destination port la n. 49192.

Quali flag sono impostati?

Le flag impostate sono in questo caso il **SYN/ACK** che rappresenta la seconda fase del three-way handshake dove il server dà risposta al client.

A quali valori sono impostati i numeri relativi di sequenza e acknowledgment?

Anche in questo caso il **relative sequence number** è 0 in quanto è la prima comunicazione che il server effettua verso il client mentre **l'ack n. è 1** in quanto conferma la ricezione del pacchetto e ne incrementa il valore di 1:

```

▼ Transmission Control Protocol, Src Port: 80, Dst Port: 49192
Source Port: 80
Destination Port: 49192
[Stream index: 0]
[Stream Packet Number: 2]
▶ [Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number 0 (relative sequence number)
Sequence Number (raw): 1439827823
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 154230803

```

Spostiamoci infine al pacchetto nr. 9 dove sarà presente l'ultima fase del three-way handshake per rispondere alla seguente domanda:

```

9 1.828916 10.0.0.11 172.16.0.40 TCP 66 49192 → 80 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSv

```

- Quale flag è impostato?

Il flag impostato è chiaramente un ACK di risposta da parte del client.

Fase 3 Analisi pcap - tcpdump

Per visualizzare il pcap tramite tcpdump è sufficiente richiamare il programma assieme al parametro -r seguito dal path del pcap da analizzare:

```
tcpdump -r /home/analyst/capture.pcap
```

Per visualizzare però i pacchetti di nostro interesse ovvero dal 7 al 9 dovremmo filtrare inserendo anche il protocollo e i pacchetti:

```
tcpdump -r /home/analyst/capture.pcap tcp | sed -n 7,9p
```

```
[analyst@secOps ~]$ tcpdump -r /home/analyst/capture.pcap tcp | sed -n 7,9p
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet), snapshot
07:55:34.623993 IP 10.0.0.11.49192 > 172.16.0.40.http: Flags [.], ack 239, win 83, c
07:55:34.624048 IP 172.16.0.40.http > 10.0.0.11.49192: Flags [P.], seq 239:854, ack
07:55:34.624049 IP 10.0.0.11.49192 > 172.16.0.40.http: Flags [.], ack 854, win 82, c
[analyst@secOps ~]$
```

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

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

Filtri per monitorare specifici endpoints: `ip.addr == <ipdaamonitorare>`

Filtri di protocollo: `http, tcp, dns, arp`

Filtri per monitorare specifiche porte `tcp.port== <portadamonitorare>`

Filtri per specifiche `flags tcp.flags.syn/reset/ack==1`

Filtri che ricercano parole specifiche all'interno dei pacchetti: `http contains "login"`

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

Oltre ai filtri, Wireshark può supportare un amministratore di rete in diversi scenari operativi:

- **Risoluzione dei problemi (troubleshooting):** Permette di individuare latenze, ritrasmissioni TCP, pacchetti persi o configurazioni errate.
- **Analisi di sicurezza:** Aiuta a rilevare attività anomale, come scansioni di rete, connessioni verso porte non standard o traffico cifrato su protocolli che dovrebbero essere in chiaro.
- **Verifica di configurazioni:** Consente di controllare che protocolli come DHCP, ARP o routing dinamico funzionino come previsto.
- **Monitoraggio delle prestazioni:** L'amministratore può usare le statistiche di Wireshark (es. *Statistics > Protocol Hierarchy, Conversations, IO Graphs*) per quantificare il volume di traffico, identificare i protocolli più usati e rilevare colli di bottiglia.
- **Formazione e documentazione:** È utile per spiegare protocolli a studenti o colleghi, grazie alla visualizzazione dettagliata dei pacchetti e all'interpretazione dei campi.

Conclusioni

L'esercizio ha permesso di sperimentare in un ambiente simulato l'intero ciclo di analisi del traffico di rete: dalla **preparazione degli host** con Mininet, alla **cattura dei pacchetti** con tcpdump, fino alla **verifica dettagliata** con Wireshark.

Nella prima fase è stato avviato un webserver interno e un browser client, simulando un classico scenario di comunicazione HTTP. Grazie a tcpdump è stato possibile catturare i pacchetti e salvarli in un file **.pcap**, strumento fondamentale per un'analisi successiva offline.

L'analisi con Wireshark ha permesso di osservare il **three-way handshake TCP**, interpretando correttamente porte di origine, porte di destinazione, flag e numeri di sequenza relativi. Successivamente è stato utilizzato nuovamente tcpdump in modalità di lettura per ispezionare selettivamente pacchetti d'interesse, dimostrando come strumenti a riga di comando e strumenti grafici possano integrarsi.

In sintesi, il laboratorio ha mostrato come Wireshark e tcpdump non siano soltanto strumenti per la cattura di pacchetti, ma veri e propri **alleati per l'amministratore di rete**, utili per garantire affidabilità, sicurezza e performance in ambienti reali.