



Analizzare i pacchetti utilizzando Wireshark

Per avviare Wireshark: `sudo wireshark`

Dopo aver avviato Wireshark seleziono eth0

Inizia la cattura dei pacchetti: Fai clic su "Start capturing packets" per iniziare la cattura dei pacchetti sulla rete.

Filtrare

tcp

oppure, se stai cercando solo la stretta di mano (3-way handshake), puoi usare il filtro:

`tcp.flags.syn == 1 and tcp.flags.ack == 0`

Analizza la stretta di mano TCP: La stretta di mano TCP a 3 vie si compone di:

SYN: Il client invia un pacchetto con il flag SYN per iniziare la connessione.

SYN-ACK: Il server risponde con un pacchetto che ha entrambi i flag SYN e ACK.

ACK: Il client risponde con un pacchetto che ha il flag ACK per completare la connessione.

Puoi vedere questi pacchetti nella cattura e analizzarli in dettaglio, esaminando i vari campi nelle intestazioni TCP come Sequence Number, Acknowledgment Number, Window Size, ecc.

Esamina la sequenza di pacchetti:

Seleziona un pacchetto SYN e usa la finestra di dettaglio per espandere le informazioni sulla sessione TCP.

Ripeti lo stesso per i pacchetti SYN-ACK e ACK, osservando i cambiamenti nei numeri di sequenza e di riconoscimento.

Visualizzare i pacchetti utilizzando tcpdump

tcpdump è un altro strumento potente per catturare il traffico di rete. Puoi usarlo per monitorare la stessa connessione TCP.

Usa il comando tcpdump per avviare la cattura dei pacchetti sulla tua interfaccia di rete
`sudo tcpdump -i eth0 -nn -v tcp`

Questo comando avvierà la cattura dei pacchetti TCP sull'interfaccia eth0, con un formato di output verboso e senza risoluzione degli indirizzi IP.

Durante la cattura dei pacchetti, cerca i pacchetti SYN, SYN-ACK e ACK che compongono la stretta di mano TCP.

Se vuoi concentrarti solo sul traffico relativo a una connessione specifica, puoi usare un filtro di porta, ad esempio per una connessione sulla porta 80 (HTTP):

```
sudo tcpdump -i eth0 -nn -v tcp port 80
```

Identificare i campi dell'intestazione TCP e UDP in Wireshark

Cattura una sessione FTP (TCP): Puoi avviare un client FTP (ad esempio ftp da terminale) o utilizzare un client grafico per stabilire una connessione a un server FTP. Una volta che la connessione è stabilita, cattura il traffico in Wireshark.

Filtrare i pacchetti TCP: Quando hai una sessione FTP in esecuzione, puoi filtrare il traffico per TCP. Seleziona un pacchetto che contiene la connessione e esamina i campi dell'intestazione TCP:

Source Port: la porta di origine (es. 21 per FTP)

Destination Port: la porta di destinazione

Sequence Number: il numero di sequenza del pacchetto

Acknowledgment Number: il numero di conferma

Cattura una sessione TFTP (UDP): Analogamente, puoi catturare una sessione TFTP che usa UDP come protocollo di trasporto. TFTP usa solitamente la porta 69.

Filtrare i pacchetti UDP: Filtra i pacchetti UDP in Wireshark:

udp

Esamina i campi dell'intestazione UDP:

Source Port: la porta di origine (di solito 69 per TFTP)

Destination Port: la porta di destinazione

Length: la lunghezza del pacchetto UDP

Checksum: il checksum UDP per la verifica dell'integrità del pacchetto

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ sudo tcpdump -i eth0 -nn -v tcp
[sudo] password for kali:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144
bytes
08:02:49.819561 IP (tos 0x0, ttl 64, id 58901, offset 0, flags [DF], proto TCP
(6), length 60)
  192.168.1.217.52124 > 34.117.188.166.443: Flags [S], cksum 0xalcb (incorec
t -> 0x603b), seq 645917957, win 64240, options [mss 1460,sackOK,TS val 1857813
406 ecr 0,nop,wscale 7], length 0
08:02:49.844113 IP (tos 0x0, ttl 119, id 0, offset 0, flags [DF], proto TCP (6)
, length 60)
  34.117.188.166.443 > 192.168.1.217.52124: Flags [S.], cksum 0x14fe (correct
), seq 47332953, ack 645917958, win 65535, options [mss 1412,sackOK,TS val 4136
504978 ecr 1857813406,nop,wscale 8], length 0
08:02:49.844290 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 40)
  192.168.1.217.52124 > 34.117.188.166.443: Flags [R], cksum 0x2d66 (correct)
, seq 645917958, win 0, length 0
08:02:50.279531 IP (tos 0x0, ttl 64, id 25991, offset 0, flags [DF], proto TCP
(6), length 60)
  192.168.1.217.57010 > 34.160.144.191.443: Flags [S], cksum 0x760f (incorec
t -> 0x832f), seq 240974042, win 64240, options [mss 1460,sackOK,TS val 1088826
228 ecr 0,nop,wscale 7], length 0
08:02:50.307964 IP (tos 0x0, ttl 122, id 0, offset 0, flags [DF], proto TCP (6)
, length 60)
  34.160.144.191.443 > 192.168.1.217.57010: Flags [S.], cksum 0x20a7 (correct
), seq 4018332273, ack 240974043, win 65535, options [mss 1412,sackOK,TS val 42
91988431 ecr 1088826228,nop,wscale 8], length 0
08:02:50.308004 IP (tos 0x0, ttl 64, id 25992, offset 0, flags [DF], proto TCP
(6), length 52)
  192.168.1.217.57010 > 34.160.144.191.443: Flags [.], cksum 0x7607 (incorec
t -> 0x4d32), ack 1, win 502, options [nop,nop,TS val 1088826256 ecr 4291988431
], length 0
08:02:50.309137 IP (tos 0x0, ttl 64, id 25993, offset 0, flags [DF], proto TCP
(6), length 268)
  192.168.1.217.57010 > 34.160.144.191.443: Flags [P.], cksum 0x76df (incore
ct -> 0x266f), seq 1, win 502, options [nop,nop,TS val 1088826258 ecr 4291988431
], length 0
```