S11/L2

Introduzione e contesto del laboratorio

Lo scopo del laboratorio è catturare e analizzare, tramite Wireshark e tcpdump, l'handshake TCP a tre vie tra un host client (H1) e un server web (H4) in un ambiente Mininet su una VM CyberOps. Questo consente di comprendere come TCP stabilisca una connessione affidabile prima dello scambio di dati di livello applicazione (ad es. HTTP).

Il laboratorio è suddiviso in tre parti principali:

1. Preparare gli host per catturare il traffico (impostazione VM, Mininet, avvio browser e tcpdump).
2. Analizzare i pacchetti catturati con Wireshark.
3. Visualizzare e filtrare i pacchetti con tcpdump.

Parte 1 – Preparare gli Host per Catturare il Traffico

a. Avvio della VM e accesso

- Avviare la VM CyberOps Workstation.
- Effettuare il login con utente **analyst** e password **cyberops**.
  Riferimento: istruzioni di login nel file di laboratorio.

b. Avvio di Mininet

- Da terminale dell'utente **analyst**, eseguire il comando per avviare Mininet.
  Obiettivo: creare la topologia definita, con host H1 e H4.

c. Avvio degli host H1 e H4

- Nel prompt di Mininet, lanciare:
- **xterm H1**
- **xterm H4**
- Ciò apre due finestre terminali collegate ad H1 e H4 rispettivamente, consentendo di eseguire comandi indipendenti su ciascun host.

d. Avvio del server web su H4

- Nella finestra terminale relativa a H4, eseguire lo script di avvio del server:
- **/home/analyst/lab.support.files/scripts/reg_server_start.sh**

  (eseguito come root o con permessi adeguati). In questo modo H4 fungerà da server HTTP per la sessione di test.

e. Cambio utente su H1 per eseguire Firefox

- Poiché non è consentito eseguire Firefox da root per motivi di sicurezza, in H1 (dove si è root) usare:
- **su analyst**

  In modo da passare all'utente analyst e poter lanciare Firefox.

f. Avvio del browser e di tcpdump su H1

1. Avviare Firefox:
2. **firefox &**
3. Occorrerà attendere qualche istante perché la GUI si apra.
4. Avviare subito dopo tcpdump sulla interfaccia di H1, con opzioni e conteggio limitato:
5. **sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.pcap**
   - **-i H1-eth0**: specifica l'interfaccia di rete.
   - **-v**: mostra informazioni di avanzamento e dettagli dei pacchetti catturati.
   - **-c 50**: cattura solo 50 pacchetti e poi termina automaticamente.
   - **-w capture.pcap**: scrive i pacchetti su file pcap per analisi successive .

g. Navigazione verso il server

- Con tcpdump in esecuzione, nella finestra di Firefox su H1 navigare rapidamente all'indirizzo IP del server HTTP: **http://172.16.0.40** (IP esemplificativo indicato nel laboratorio). Ciò innesca l'handshake TCP e il successivo traffico HTTP, che verrà catturato.

Note operative

- Il file capture.pcap si troverà in **/home/analyst/** e conterrà i pacchetti di interesse.
- Alla fine della cattura (dopo 50 pacchetti) tcpdump terminerà automaticamente; se necessario estendere il numero, modificare -c.

Parte 2 – Analizzare i Pacchetti usando Wireshark

1. Avvio di Wireshark e apertura del file pcap

- In H1, nel terminale come utente **analyst**, eseguire:
  wireshark-gtk &

- Confermare eventuali avvisi di esecuzione come superutente.
- In Wireshark: File > Open → selezionare **/home/analyst/capture.pcap**.

2. Applicazione del filtro

- Nella barra di filtro di Wireshark, inserire **tcp** e applicare il filtro per mostrare solo pacchetti TCP. In questo caso, i primi 3 pacchetti rappresentano l'handshake a tre vie.

3. Analisi dettagliata dell'handshake TCP

Wireshark mostra i pacchetti in frame numerati. Considerando che i primi tre frame TCP siano quelli dell'handshake:

Pacchetto 1 (Frame 1) – SYN iniziale dal client

- **Indirizzi IP**: sorgente = IP di H1 (es. 10.0.0.11), destinazione = IP del server (172.16.0.40).
- **Porte TCP**:
  - Porta di origine: un numero elevato **58716**.
    - **Classificazione**: porta dinamica, scelta casualmente dal sistema per la sessione client.
  - Porta di destinazione: **80**.
    - **Classificazione**: porta nota associata a servizio HTTP.
- **Flag impostato**: SYN = 1, indica richiesta di apertura connessione.
  In Wireshark, espandere il protocollo TCP nel pannello dettagli, quindi la sezione Flags e osservare **SYN: 1** e generalmente **ACK: 0**.
- **Numero di sequenza relativo**: Wireshark mostra un numero di sequenza "relativo" usualmente a 0 o a un valore di partenza predefinito; ad esempio "Seq=0" nell'interfaccia utente relativa.
  Questo valore indica l'iniziale sequence number (ISN) scelto dal client.
- **Significato**: il client invia un TCP SYN per iniziare la connessione.
  .

Domande da rispondere per il primo pacchetto:

- Qual è il numero di porta TCP di origine? → **58716**.

- Come classificheresti la porta di origine? → **Effimera e/o dinamica**.
- Qual è il numero di porta TCP di destinazione? → **80**.
- Come classificheresti la porta di destinazione? → **HTTP**.
- Quale flag è impostato? → **SYN**.
- A quale valore è impostato il numero di sequenza relativo? → **Tipicamente 0 nell'interfaccia di Wireshark (ISN relativo)**.

Pacchetto 2 (Frame 2) – SYN-ACK di risposta dal server

- **Indirizzi IP**: sorgente = IP server (172.16.0.40), destinazione = IP client (10.0.0.11).
- **Porte TCP**:
  - Origine: **80** (HTTP).
  - Destinazione: stessa porta del client **58716**.
- **Flag impostati**: SYN = 1 e ACK = 1.
  - Indica che il server accetta la richiesta di connessione e conferma l'ISN del client.
- **Numeri di sequenza e acknowledgment relativi**:
  - Sequence number relativo: in Wireshark "Seq=0" o "Seq=1" a seconda della visualizzazione relativa.
  - Number relativo: corrisponde a ISN client + 1 (ad es. Ack=1 se il client ISN relativo era 0).
- **Significato**: il server risponde con SYN-ACK per completare il secondo passo dell'handshake.

.

Domande per il secondo pacchetto:

- Quali sono i valori delle porte di origine e destinazione? → **Origine: 80 - Destinazione: 58716**.
- Quali flag sono impostati? → **SYN e ACK**.
- A quali valori sono impostati i numeri relativi di sequenza e acknowledgment?

  → **Sequenza relativa ≈ 0.**

  Acknowledgment relativo = ISN client + 1.

Pacchetto 3 (Frame 3) – ACK finale dal client

- **Indirizzi IP**: sorgente = IP client (10.0.0.11), destinazione = IP server (172.16.0.40).
- **Porte TCP**:
  - Origine: porta client.
  - Destinazione: 80 (HTTP).
- **Flag impostato**: ACK = 1, SYN = 0.
  - Conferma l'ISN del server (ISN server + 1).
- **Numeri relativi di sequenza e acknowledgment**:
  - Sequence relativo: ISN client + 1 (di solito 1 se ISN iniziale 0).
  - Acknowledgment relativo: ISN server + 1 (di solito 1).
- **Significato**: connessione TCP stabilita; a questo punto la sessione è pronta per lo scambio dati HTTP.
  .

Domande per il terzo pacchetto:

- Quale flag è impostato? → **ACK**.

I numeri relativi di sequenza e acknowledgment sono impostati a 1 come punto di partenza ed entrambi sono generalmente 1 nell'interfaccia di Wireshark se si usa la numerazione relativa con ISN iniziale 0.

Conferma che la connessione TCP è stabilita e la comunicazione applicativa può iniziare.

Osservazioni sull'analisi con Wireshark

- Espandere le sezioni IP e TCP nei dettagli del pacchetto per vedere:
  - Indirizzi IP e porte.
  - Flags di controllo TCP.
  - Sequence/Acknowledgment numbers (con o senza numerazione relativa).
  - Opzioni TCP (MSS, Window Scale, SACKOK, Timestamp) che possono comparire nel SYN/SYN-ACK.
- La numerazione relativa di Wireshark facilita la comprensione, mostrando valori a partire da 0 o 1.

Parte 3 – Visualizzare i pacchetti usando tcpdump

a. Consultazione delle pagine di manuale (man tcpdump)

- Digitare **man tcpdump** in un terminale Linux per vedere le opzioni disponibili.
- In particolare, ricercare **-r**:
  - **Cosa fa l'opzione `-r`?**
    - Permette di leggere pacchetti da un file pcap anziché dalla interfaccia di rete. In pratica, **tcpdump -r file.pcap** riproduce la cattura salvata.

b. Lettura del file pcap e filtraggio dei primi 3 pacchetti TCP

- Comando:
- **tcpdump -r /home/analyst/capture.pcap tcp -c 3**
  - **-r /home/analyst/capture.pcap**: legge dal file di cattura.
  - **tcp**: filtro di espressione BPF per considerare solo pacchetti TCP.
  - **-c 3**: mostra solo i primi tre pacchetti che soddisfano il filtro.

Domande di Riflessione

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

Un amministratore di rete può definire numerosi filtri Wireshark per isolare traffico di interesse. Di seguito tre esempi con descrizione:

1. **Filtro per indirizzo IP sorgente o destinazione**
   - Esempio: **ip.addr == Ip**
   - Utilizzo: mostra tutto il traffico in cui compare quell'IP, utile per investigare comunicazioni di un host specifico o verificare comportamenti anomali.
2. **Filtro per porta TCP/UDP**
   - Esempio: **tcp.port == 22** o **udp.port == 53**
   - Utilizzo: isolare traffico SSH (porta 22) o DNS (porta 53), per debug o monitoraggio di servizi critici.

3. **Filtro per flag TCP o specifiche fasi di handshake**
    - Esempio: **tcp.flags.syn == 1 and tcp.flags.ack == 0**
        - Mostra solo pacchetti SYN iniziali (prima fase handshake).
    - Oppure: **tcp.flags.reset == 1** per catturare e verificare connessioni chiuse in modo anomalo.
    - Utilizzo: analisi di connessioni sospette, verifica di tentativi di scansione di porte o handshake incompleti.

Questi sono esempi di alcuni filtri che semplificano l'individuazione di traffico rilevante in reti di grandi dimensioni.

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

Wireshark è uno strumento versatile oltre la semplice analisi di handshake. Alcuni usi in ambiente di produzione:

1. **Risoluzione di problemi di performance e latenza**
    - Analisi dei ritardi end-to-end TCP individuando (retransmission, congestione), tempi di round-trip, sovraccarichi di rete. Utile per ottimizzare applicazioni critiche.
2. **Analisi della sicurezza e rilevamento anomalie**
    - Identificare traffico sospetto: scansioni di porte, tentativi di attacco DoS, exploit conosciuti. Analisi di pacchetti malevoli o sospetti per rafforzare difese.
    - Verifica di eventuali comunicazioni non autorizzate o dati esfiltrati.
3. **Verifica e debug di protocolli applicativi**
    - Analizzare protocolli specifici (VoIP, HTTP/2, SMB, database proprietari) per malfunzionamenti o configurazioni errate.
    - Controllo di handshake di protocolli di cifratura e validità dei certificati.
4. **Analisi forense post-incidente**
    - In caso di violazione, usare catture archiviate per ricostruire sequenza di eventi, identificare host compromessi, flussi di dati.
5. **Monitoraggio e baseline del traffico**
    - Definire profili di traffico e confrontarli con traffico in tempo reale per individuare anomalie.

6. **Sviluppo e test di nuove applicazioni di rete**
    - Durante fasi di test, verificare conformità al protocollo, efficienza, gestione errori e correggere bug prima del deployment in produzione.
7. **Verifica di politiche QoS e routing**
    - Controllare se i pacchetti seguono le policy di qualità del servizio, identificare root cause di priorità errate o congestioni.
8. **Formazione e addestramento**
    - Ambienti di laboratorio per insegnare agli operatori e sviluppatori come funzionano i protocolli e come diagnosticare problemi di rete.

In tutti questi casi, occorre considerare aspetti di privacy oltre che di performance: catture estese possono generare grande volume di dati e includere informazioni sensibili. In produzione si usa spesso una combinazione di capture mirate (filtri BPF), mirroring di porte su switch e archiviazione controllata.