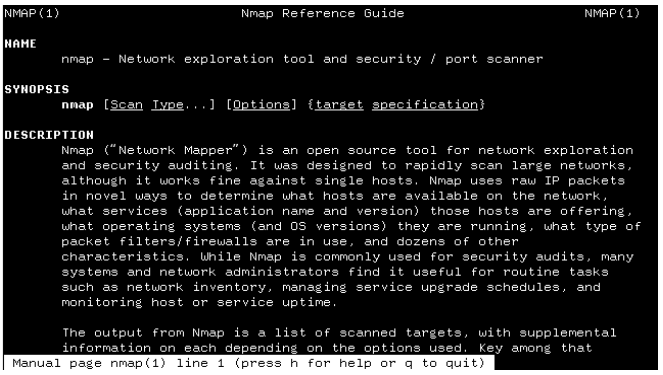
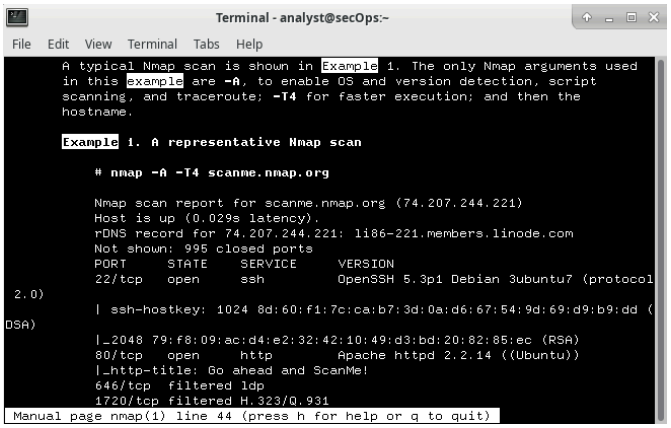
# Bonus progetto S11-L5

- ## Bonus 1 - Esplorazione Nmap

Apriamo il terminale sulla nostra macchina CyberOps WS ed inseriamo il comando **man nmap**.



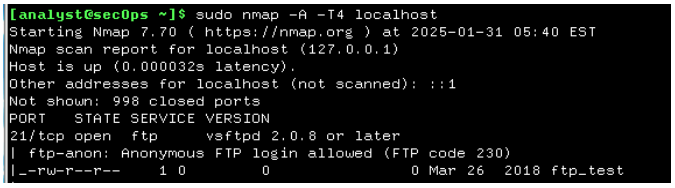

Tale comando ci permette di aprire un vero e proprio manuale su nmap, fornendoci anche una breve descrizione su cosa sia nmap.
E' possibile navigare all'interno del manuale tramite le frecce ed è possibile effettuare ricerche specifiche inserendo uno slash **/** seguito dalla parola da ricercare.
Facciamo una prova con la parola **/example.**

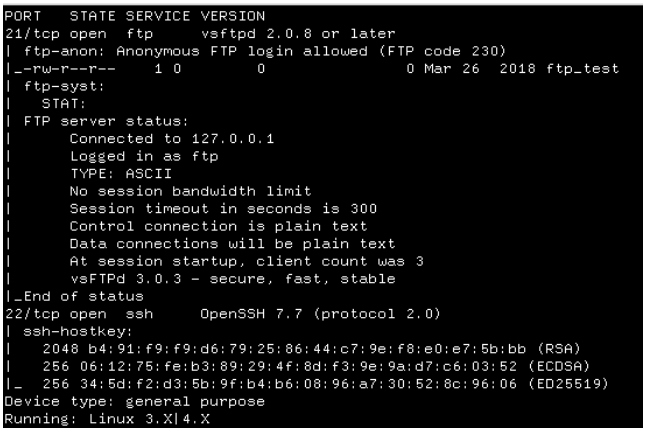



Cercando tale parola è possibile trovare un esempio effettuato utilizzando il comando **nmap -A -T4 scanme.nmap.org**.
Nell'immagine soprastante è possibile individuare anche una spiegazione per le opzioni utilizzate in questo comando:

1. **-A**: abilita il rilevamento del sistema operativo, il rilevamento delle versioni, la scansione tramite script e il traceroute.
2. **-T4**: per un'esecuzione più rapida della scansione.

Iniziamo effettuando una scansione dell'host locale, inserendo nella sezione del destinatario, la parola **localhost**.




Analizzando la scansione è possibile individuare 2 porte aperte:

1. **21**: ad essa è associato il servizio FTP nella versione vsftpd2.0.8
2. **22**: ad essa è associato il servizio SSH nella versione OpenSSH7.7

Lanciamo ora il comando **ip address** per avere informazioni sulla nostra configurazione di rete, scoprendo che l'indirizzo ip della nostra interfaccia di rete è **10.0.2.15/24**. Grazie a tale informazione è possibile risalire all'indirizzo di rete e il suo prefisso che in questo caso sarà **10.0.2.0/24**.
Ottenuta questa informazione, è possibile lanciare una scansione nmap su tutti i dispositivi collegati alla rete.


La nostra scansione individua tre differenti host:



1. Il primo host ha indirizzo IP **10.0.2.2** ed ha due porte aperte: 135 con servizio msrpc e 445 con servizio microsoft-ds.
2. il secondo host ha indirizzo IP **10.0.2.3** ed ha una porta aperta, la 53, alla quale corrisponde il servizio domain.
3. Il terzo host ha indirizzo IP **10.0.2.15** ed è il nostro host locale con (come visto prima) due porte aperte: la 21(FTP) e la 22 (SSH).

Proviamo ora ad effettuare la scansione di un server remoto. In particolare, effettuiamo la scansione del server del sito "**scanme.nmap.org**". Pertanto il comando è il seguente:

Dalla scansione riusciamo ad ottenere informazioni molto interessanti:

```
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.028s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 closed ports
```

1. **Indirizzo IPv4**: 45.33.32.156
2. **Indirizzo IPv6**: 2600:3c01::f03c:91ff:fe18:bb2f

```
PORT      STATE SERVICE    VERSION
22/tcp    open  ssh        OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0
)
| ssh-hostkey:
|   1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
|   2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
|   256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)
|_  256 33:fa:91:0f:e0:e1:7b:1f:6d:05:a2:b0:f1:54:41:56 (ED25519)
80/tcp    open  http       Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Go ahead and ScanMe!
9929/tcp  open  nping-echo Nping echo
31337/tcp open  tcpwrapped
```

Porte aperte sul server:

1. **22**: servizio SSH versione OpenSSH6.6.1p1
2. **80**: servizio HTTP versione Apache httpd2.4.7
3. **9929**: nping-echo
4. **31337**: tcpwrapped

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

**Sistema operativo del server**: Linux

**Come può Nmap aiutare nella sicurezza della rete? In che modo un attore malintenzionato può utilizzare Nmap come strumento dannoso?**

Nmap è uno strumento potente per l'analisi e la sicurezza delle reti. Può essere utilizzato dagli amministratori di sistema e dagli esperti di sicurezza per:

1. **Scansione delle porte aperte**:
   Identifica quali porte sono aperte su un host o su un'intera rete, aiutando a individuare servizi non necessari o esposti.
2. **Rilevamento del sistema operativo e delle versioni dei servizi**:
   Permette di scoprire il sistema operativo, le versioni dei software e i servizi in esecuzione su un dispositivo, facilitando la gestione e la protezione della rete.
3. **Esecuzione di script di sicurezza (NSE)**:
   Consente di individuare vulnerabilità note, configurazioni errate e servizi obsoleti attraverso script automatizzati.
4. **Traceroute e mappatura della rete**:
   Aiuta a visualizzare la topologia della rete e a individuare possibili punti di ingresso per attacchi o colli di bottiglia nelle comunicazioni.

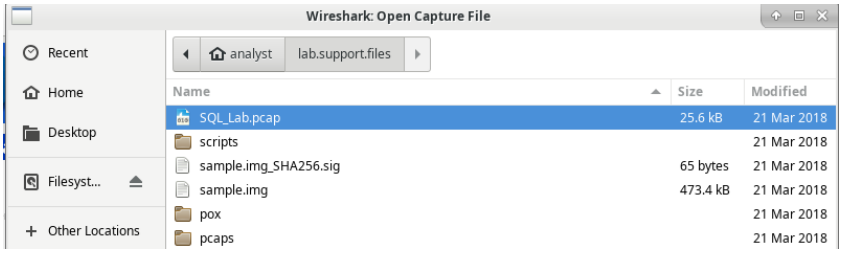
Sebbene Nmap sia uno strumento legittimo per la sicurezza informatica, può anche essere usato da hacker e attori malevoli per scopi dannosi, come:

1. **Ricognizione e raccolta di informazioni**:
   Gli aggressori possono utilizzare Nmap per raccogliere informazioni su un bersaglio, identificando sistemi, servizi e porte vulnerabili.

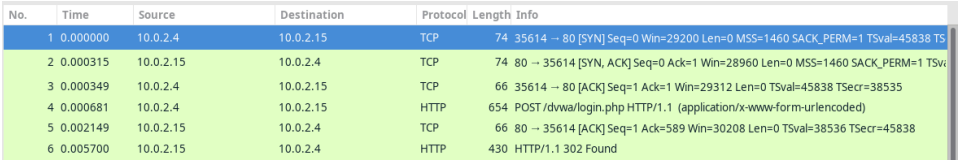
2. **Evasione dei sistemi di sicurezza**:
   Utilizzando opzioni avanzate, come la scansione lenta, un attaccante può mascherare il proprio traffico e aggirare firewall e sistemi di rilevamento delle intrusioni (IDS/IPS).
3. **Identificazione di vulnerabilità note**:
   Grazie agli script NSE, un attaccante può automatizzare la ricerca di falle di sicurezza in software obsoleti o mal configurati.
4. **Mappatura della rete**:
   Nmap può essere usato per creare una mappa dettagliata di una rete target.

- ## Bonus 2 - Analisi attacco SQL Injection

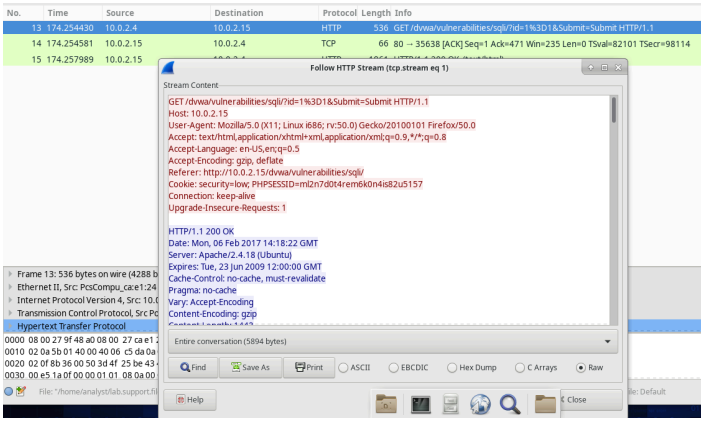
Apriamo il file **SQL_Lab.pcap** contenente la cattura di rete di un attacco **SQL injection** tramite Wireshark.




Analizzando il file, possiamo subito individuare i due indirizzi IP coinvolti: **10.0.2.15** e **10.0.2.4**.

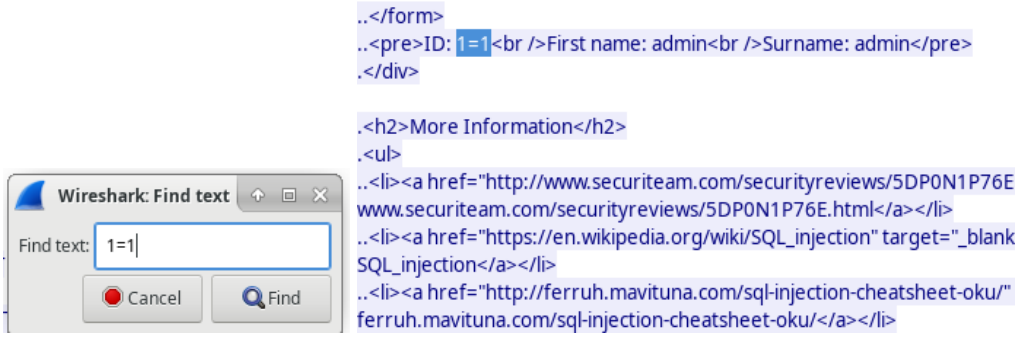



Notiamo qualcosa di strano sulla riga 13 di tale cattura.
Così ci clicchiamo sopra e con il tasto destro apriamo la finestra **HTTP stream** per analizzare lo scambio avvenuto tra i due sistemi.




Il testo rosso rappresenta la richiesta della macchina sorgente, mentre il testo blu rappresenta la macchina destinataria che a sua volta risponde alla richiesta.
Ciò che ha attirato la nostra attenzione è il possibile inserimento della stringa “**1=1**” all’interno della richiesta **GET**. Tale stringa è infatti tipicamente utilizzata per individuare

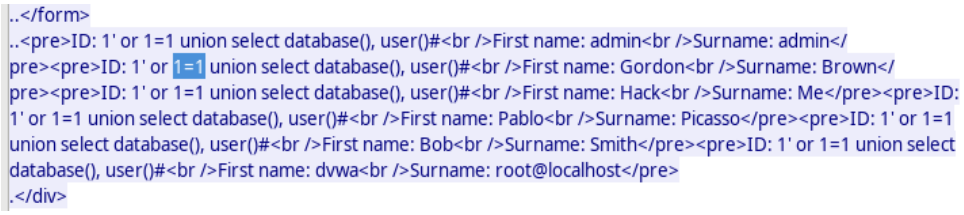
vulnerabilità all'SQL injection. Infatti, essendo una richiesta sempre vera, se il sistema risulterà vulnerabile, risponderà fornendo informazioni.
Per avere conferma, effettuiamo una ricerca:




Come possiamo notare, la nostra ricerca conferma l'inserimento di una stringa "1=1" all'interno di un campo. La macchina che riceve tale messaggio, non risponde ad esso con un messaggio di errore ma fornisce delle informazioni.
Questo vuol dire che essa è vulnerabile ad un attacco **SQL Injection**.

Analizziamo ora come continua l'attacco spostandoci sulla riga 19.


Come prima, apriamo HTTP stream e cerchiamo la stringa "1=1".

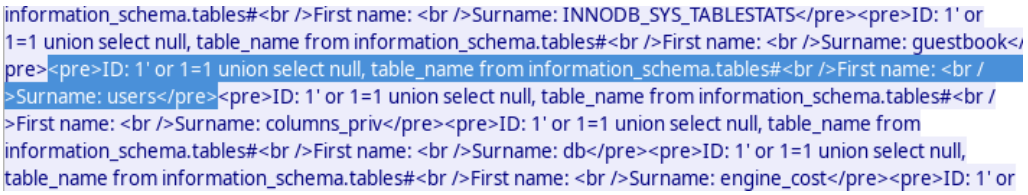



In questo caso l'attaccante, tramite una richiesta del tipo **1' or 1=1 UNION SELECT database(),user()**, riesce ad individuare il nome del database "**dvwa**" e il nome utente "**root@localhost**".

Andiamo avanti spostandoci ora sulla riga 22 e aprendo la relativa finestra HTTP stream. Effettuando anche qui una ricerca per la stringa "1=1", troviamo:

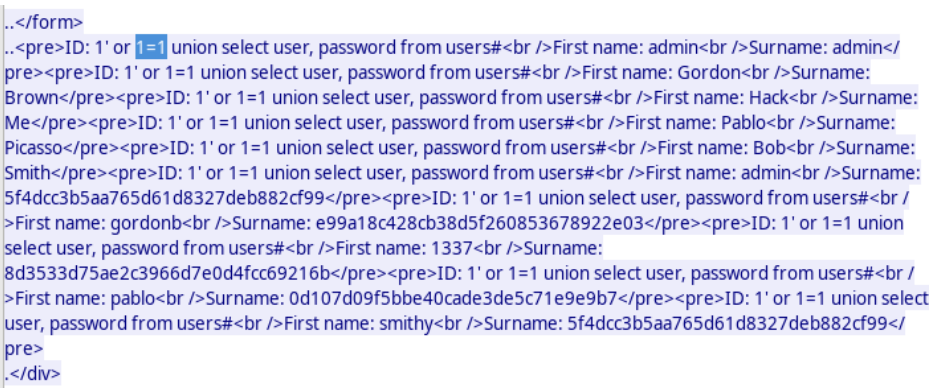

Ora l'attaccante inserisce la richiesta **1' or 1=1 UNION SELECT null,version()**. In questo modo viene a conoscenza della versione di MySQL che in questo caso è la 5.7.12-0.

Analizziamo ora la riga 25. Apriamo la finestra HTTP stream e cerchiamo la parola "**users**".




Qui l'attaccante inserisce la richiesta **1' or 1=1 UNION SELECT null,table_name from information_schema.tables**, per vedere tutte le tabelle presenti nel database.
L'attaccante in questo caso avrebbe potuto filtrare la ricerca inserendo un **WHERE**, ad esempio **WHERE table_name='users'**, in modo da ricevere come risposta solo le tabelle che rispettano quei parametri.

L'attacco si conclude alla riga 28. Apriamo la relativa pagina HTTP stream e cerchiamo nuovamente la stringa "1=1".




Grazie alla richiesta **1' or 1=1 UNION SELECT user, password FROM users**, l'attaccante riesce ad ottenere quello che voleva e cioè i nomi utenti e relativi hashes delle password presenti nella tabella users.
Tali hashes potranno poi essere tradotti per risalire alle password vere e proprie.

**Qual è il rischio di avere piattaforme che usano il linguaggio SQL?**
Il linguaggio SQL (Structured Query Language) è ampiamente utilizzato per gestire database, ma può rappresentare un rischio se non viene implementato correttamente. Ecco i principali rischi:

1. **SQL Injection**:
   Un attaccante può inserire codice SQL malevolo nei campi di input di un'applicazione per manipolare il database.
2. **Esposizione di dati sensibili**:
   Se il database SQL non è ben protetto, un attaccante potrebbe accedere a informazioni riservate senza bisogno di un exploit avanzato.
3. **Configurazione errata database**:
   Molti database vengono lasciati con impostazioni di default, come credenziali di amministratore deboli o accesso remoto abilitato.
4. **Denial of Service (DoS) tramite query pesanti**:
   Un attaccante può eseguire query molto pesanti o ripetitive per sovraccaricare il database e rallentare o bloccare il servizio.

**Quali sono 2 metodi efficaci per prevenire attacchi SQL Injection?**

1. **Uso di query parametrizzate**:
   Le query parametrizzate impediscono agli attaccanti di iniettare codice SQL malevolo perché i dati inseriti vengono trattati come semplici valori e non come parte della query SQL.
2. **Validazione e sanitizzazione degli input**:
   Riduce il rischio di SQL Injection bloccando o filtrando caratteri pericolosi prima che raggiungano il database. Questo può essere fatto: verificando che l'input sia nel formato corretto, usando funzioni specifiche per neutralizzare caratteri speciali, accettando solo valori predefiniti quando possibile (es. menù a tendina invece di input libero).