

# Esercizio 4: Usare Wireshark per Esaminare il Traffico HTTP e HTTPS

## Obiettivi:

- Parte 1 Catturare e visualizzare il traffico HTTP
- Parte 2 Catturare e visualizzare il traffico HTTPS

## Parte 1: Catturare e Visualizzare il Traffico HTTP

### PASSO 1: AVVIARE LA VM ED EFFETTUARE IL LOGIN

Iniziamo avviando la nostra VM Kali, inserendo nome Utente e Password.

Una volta all'interno della macchina, apriamo un terminale, digitiamo **'ip address'**, dove troveremo l'elenco delle interfacce con i vari indirizzi **IP**.

```
(kali㉿kali)-[~]
$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:b4:a1:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.188/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 42449sec preferred_lft 42449sec
```

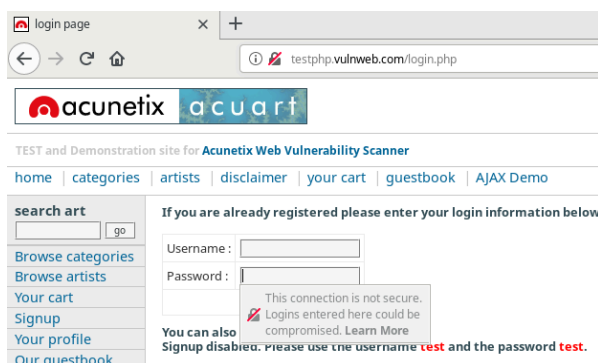
### PASSO 2: APRIRE UN TERMINALE E AVVIARE TCPDUMP

Inseriamo subito dopo il comando: **sudo tcpdump -i eth0s3 -s 0 -w httpdump.pcap**

```
(kali㉿kali)-[~]
$ sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Che avvierà il tcpdump registrando il traffico di rete sull'interfaccia eth0.

Il passaggio successivo sarà aprire un browser web navigando sul sito → <http://testphp.vulnweb.com/login.php> si aprirà questa pagina, che ci avviserà della connessione **non sicura** all'interno di questo sito visto che usa appunto HTTP.



Inseriamo un nome Utente e una Password, in questo caso **Admin - Admin** e poi premiamo → **login**.

Username :	<input type="text" value="Admin"/>
Password :	<input type="password" value="Admin"/>
<input type="button" value="login"/>	

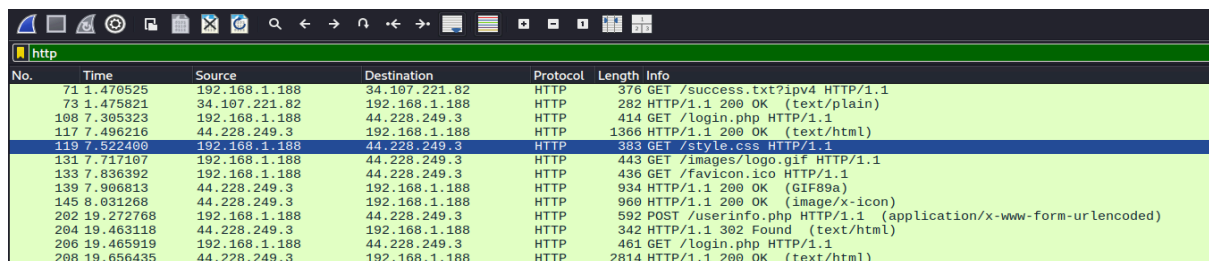
Chiudiamo il Browser. Torniamo sul terminale, e premiamo CTRL+C per terminare la cattura dei pacchetti, perchè adesso andremo a visualizzare e ad analizzare ciò che abbiamo catturato.

```
(kali㉿kali)-[~]
$ sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C306 packets captured
309 packets received by filter
0 packets dropped by kernel
```

### PASSO 3: VISUALIZZARE LA CATTURA HTTP

Nella nostra kali, una volta eseguiti i passaggi precedenti, andando in Home troveremo il file **.pcap** da noi creato, da analizzare con Wireshark.

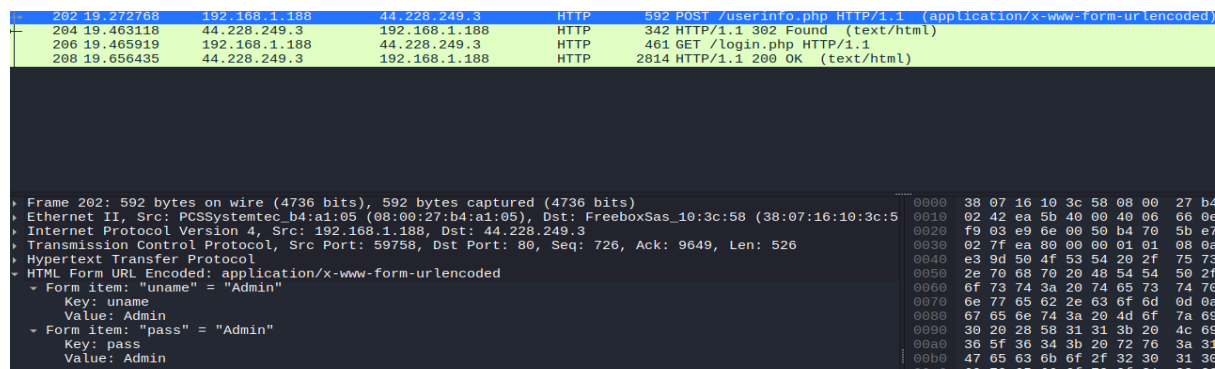
Apriamo dunque Wireshark, selezioniamo il file interessato e applichiamo un filtro **http**. Questo è quello che ci viene mostrato:



No.	Time	Source	Destination	Protocol	Length	Info
71	1.470525	192.168.1.188	34.107.221.82	HTTP	376	GET /success.txt?ip=192.168.1.188 HTTP/1.1
73	1.475821	34.107.221.82	192.168.1.188	HTTP	282	HTTP/1.1 200 OK (text/plain)
108	7.305323	192.168.1.188	44.228.249.3	HTTP	414	GET /login.php HTTP/1.1
117	7.496216	44.228.249.3	192.168.1.188	HTTP	1366	HTTP/1.1 200 OK (text/html)
119	7.522400	192.168.1.188	44.228.249.3	HTTP	383	GET /style.css HTTP/1.1
131	7.717107	192.168.1.188	44.228.249.3	HTTP	443	GET /images/logo.gif HTTP/1.1
133	7.836392	192.168.1.188	44.228.249.3	HTTP	436	GET /favicon.ico HTTP/1.1
139	7.906813	44.228.249.3	192.168.1.188	HTTP	934	HTTP/1.1 200 OK (GIF89a)
145	8.031268	44.228.249.3	192.168.1.188	HTTP	960	HTTP/1.1 200 OK (image/x-icon)
202	19.272768	192.168.1.188	44.228.249.3	HTTP	592	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
204	19.463118	44.228.249.3	192.168.1.188	HTTP	342	HTTP/1.1 302 Found (text/html)
206	19.465919	192.168.1.188	44.228.249.3	HTTP	461	GET /login.php HTTP/1.1
208	19.656435	44.228.249.3	192.168.1.188	HTTP	2814	HTTP/1.1 200 OK (text/html)

Andremo a selezionare il messaggio **POST**.

#### Domanda 1 : Quale due informazioni vengono visualizzate?



No.	Time	Source	Destination	Protocol	Length	Info
202	19.272768	192.168.1.188	44.228.249.3	HTTP	592	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
204	19.463118	44.228.249.3	192.168.1.188	HTTP	342	HTTP/1.1 302 Found (text/html)
206	19.465919	192.168.1.188	44.228.249.3	HTTP	461	GET /login.php HTTP/1.1
208	19.656435	44.228.249.3	192.168.1.188	HTTP	2814	HTTP/1.1 200 OK (text/html)

Hypertext Transfer Protocol	
Content-Type	application/x-www-form-urlencoded
Form item: "uname"	"Admin"
Key: uname	Value: Admin
Form item: "pass"	"Admin"
Key: pass	Value: Admin

**R:** Le due informazioni visualizzabili in chiaro sono il nome Utente da noi usato, ovvero: **Admin**. E la **password**, sempre in chiaro, che abbiamo usato per il login: **Admin**.

## Parte 2: Catturare e Visualizzare il Traffico HTTPS

### PASSO 1: AVVIARE TCPDUMP ALL'INTERNO DI UN TERMINALE

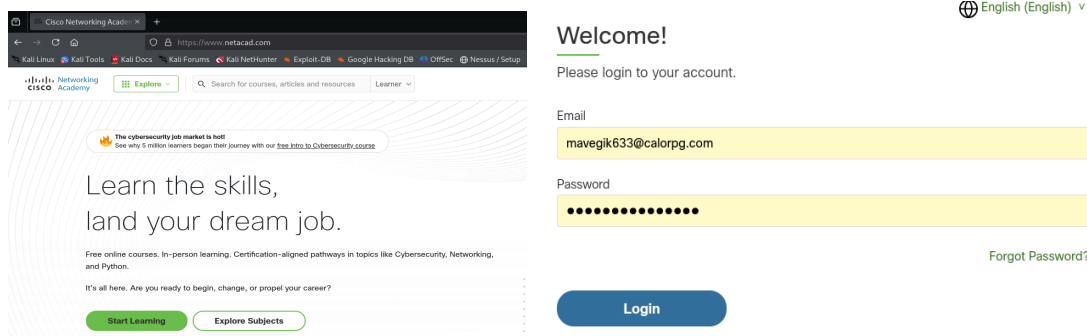
Come nella prima parte, apriamo un terminale e creiamo un altro file .pcap, questa volta per il traffico **HTTPS**.

Nel terminale scriveremo: **sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap**

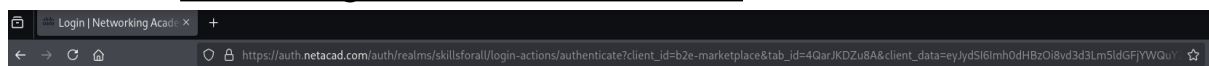
```
(kali㉿kali)-[~]  
$ sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap  
[sudo] password for kali:  
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Come per il comando usato prima, una volta digitato e fatto invio si avvierà il tcpdump che registrerà il traffico di rete, che verrà stampato nel file httpsdump.pcap, che troveremo nella home di kali.

Una volta in ascolto, apriamo il Browser nella VM e facciamo il login.



### Domanda 2: Cosa noti riguardo all'URL del sito web?



**R:** La prima cosa che ho notato nell'**URL** - clickando su '**login**' - è stata l'intestazione https. Ovvero "**https://auth.netacad.com/auth/realms...**"

**https://** → indica che il sito una comunicazione è **criptata** (via TLS/SSL).

**auth.** → è un **sottodominio**, spesso abbreviato di "**authentication**", ovvero **autenticazione**.

Quel **sottodominio** è **dedicato all'autenticazione degli utenti**, ad esempio:

- Inserimento di username/password
- Login via OAuth, SSO (Single Sign-On), SAML, ecc.

Il sito separa la logica di autenticazione dal sito principale (es: **www.**), spesso per:

- **Migliorare la sicurezza:** separando cookie/sessioni
- Centralizzare la gestione utenti
- **Ridurre la superficie d'attacco:** solo un componente gestisce login e accessi

## PASSO 2: VISUALIZZARE LA CATTURA HTTPS

tcp.port==443						
No.	Time	Source	Destination	Protocol	Length	Info
13	2.256805	192.168.1.188	34.160.144.191	TCP	74	33676 → 443 [SYN] Seq=
14	2.261406	34.160.144.191	192.168.1.188	TCP	74	443 → 33676 [SYN, ACK]
15	2.261426	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
16	2.261557	192.168.1.188	34.160.144.191	TLSv1.2	282	Client Hello (SNI=cont
17	2.266173	34.160.144.191	192.168.1.188	TCP	66	443 → 33676 [ACK] Seq=
18	2.268114	34.160.144.191	192.168.1.188	TLSv1.2	1466	Server Hello
19	2.268126	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
20	2.268171	34.160.144.191	192.168.1.188	TLSv1.2	1687	Certificate, Server Key
21	2.268174	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
22	2.269922	192.168.1.188	34.160.144.191	TLSv1.2	159	Client Key Exchange, Cl
23	2.274477	34.160.144.191	192.168.1.188	TLSv1.2	377	New Session Ticket, Ch
24	2.274478	34.160.144.191	192.168.1.188	TLSv1.2	135	Application Data
25	2.282733	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
26	2.282821	192.168.1.188	34.160.144.191	TLSv1.2	165	Application Data
27	2.282861	192.168.1.188	34.160.144.191	TLSv1.2	104	Application Data
28	2.285724	34.160.144.191	192.168.1.188	TCP	135	[TCP Spurious Retransm
29	2.285733	192.168.1.188	34.160.144.191	TCP	78	[TCP Dup ACK 25#1] 336
30	2.287205	34.160.144.191	192.168.1.188	TCP	66	443 → 33676 [ACK] Seq=
31	2.287205	34.160.144.191	192.168.1.188	TLSv1.2	104	Application Data
32	2.287215	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
36	2.537697	192.168.1.188	34.107.243.93	TCP	74	51616 → 443 [SYN] Seq=
37	2.541888	34.107.243.93	192.168.1.188	TCP	74	443 → 51616 [SYN, ACK]
38	2.541905	192.168.1.188	34.107.243.93	TCP	66	51616 → 443 [ACK] Seq=

Apriamo Wireshark poi → file → (selezioniamo il file da noi creato) **httpsdump.pcap**, premiamo invio. Aggiungiamo anche un filtro, come richiesto da esercizio, ovvero il **tcp.port==443** - così da filtrare direttamente il traffico HTTPS tramite la porta 443. Andiamo ora a cercare nella colonna info, un messaggio **'Application Data'**.

Come si può notare, c'è una grande differenza tra il primo file HTTP e questo che stiamo analizzando adesso HTTPS.

**Nel primo screenshot (HTTP):**

Tutto il traffico è in chiaro. Riusciamo a vedere direttamente i dati inviati dal form:

- **uname = Admin**
- **pass = Admin**
- Protocollo: **HTTP** su porta **80**

**Nello Screenshot 2 – HTTPS (cifrato)**

- Il traffico è cifrato tramite **TLSv1.2**.
- I dati del form non sono visibili, ma compaiono come:
  - **Encrypted Application Data**
- Protocollo: **HTTPS (HTTP su TLS)** su porta **443**

26	2.282821	192.168.1.188	34.160.144.191	TLSv1.2	165	Application Data
27	2.282861	192.168.1.188	34.160.144.191	TLSv1.2	104	Application Data
28	2.285724	34.160.144.191	192.168.1.188	TCP	135	[TCP Spurious Retransmission]
29	2.285733	192.168.1.188	34.160.144.191	TCP	78	[TCP Dup ACK 25#1] 33676 →
30	2.287205	34.160.144.191	192.168.1.188	TCP	66	443 → 33676 [ACK] Seq=3402
31	2.287205	34.160.144.191	192.168.1.188	TLSv1.2	104	Application Data
32	2.287215	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=447 A
36	2.537697	192.168.1.188	34.107.243.93	TCP	74	51616 → 443 [SYN] Seq=0 Win
37	2.541888	34.107.243.93	192.168.1.188	TCP	74	443 → 51616 [SYN, ACK] Seq=
38	2.541905	192.168.1.188	34.107.243.93	TCP	66	51616 → 443 [ACK] Seq=1 Ack

Frame 26: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits)	
Ethernet II, Src: PCSSystemtec_b4:a1:05 (08:00:27:b4:a1:05), Dst: FreeboxSas_10:3c:58 (38:07:16:10:3c:58)	
Internet Protocol Version 4, Src: 192.168.1.188, Dst: 34.160.144.191	
Transmission Control Protocol, Src Port: 33676, Dst Port: 443, Seq: 310, Ack: 3402, Len: 99	
Transport Layer Security	
TLSv1.2 Record Layer: Application Data Protocol: HyperText Transfer Protocol 2	
Content Type: Application Data (23)	
Version: TLS 1.2 (0x0303)	
Length: 94	
Encrypted Application Data: 00000000000000001521648ecac238914566e63573cbd9c792e5c0181f083af596708fda	
[Application Data Protocol: HyperText Transfer Protocol 2]	

**Domanda 3:** Cosa ha sostituito la sezione HTTP che era nel file di cattura precedente?

### In sintesi:

- Espandiamo completamente la sezione **Secure Sockets Layer** e poi su → **Encrypted Application Data**.

**Domanda 4: I dati dell'applicazione sono in formato plaintext o leggibile?**

```
0000000000000001f37df95d01627bf1f4311d360d81495a3473c5524586b46f06bf097ecc57fab9aca912f31b7769eea382a945e89ec16089af0d81f3182be6
```

### Dettagli tecnici:

- Quindi, in sintesi:**

- **No**, non sono dati leggibili.
- Sono il risultato della **crittografia TLS**, progettata proprio per **impedire l'ispezione dei contenuti da parte di terzi**.
- Solo il client e il server, che possiedono le chiavi corrette, possono **decifrare** questi dati.

## DOMANDE DI RIFLESSIONE:

### 1. Quali sono i vantaggi dell'uso di HTTPS invece di HTTP?

**R:** Vantaggi di HTTPS rispetto a HTTP:

- **Crittografia:** protegge i dati da intercettazioni (es. password, dati personali) tramite TLS, anche su reti pubbliche.
- **Integrità:** impedisce alterazioni ai dati durante la trasmissione.
- **Autenticazione:** verifica l'identità del sito tramite certificati digitali, prevenendo siti falsi o phishing.
- **Protezione da attacchi Man-in-the-Middle:** Rende difficile l'intercettazione o la modifica dei dati da parte di attaccanti.
- **Migliore SEO:** i motori di ricerca preferiscono e premiano i siti HTTPS con ranking superiore.
- **Fiducia degli utenti:** I browser segnalano il sito come sicuro (tramite il lucchetto di sicurezza) mentre HTTP segnala come **"Non sicuro"**
- **Funzionalità moderne:** alcune API web richiedono HTTPS, esempio Geolocation e Push Notifications.

**Rischi evitati con HTTPS:**

- Dati cifrati vs dati in chiaro
- Protezione dalla lettura da terzi
- Sicurezza delle credenziali
- Connessione autenticata e protetta da attacchi

**In breve:**

**HTTPS = HTTP + Sicurezza** — indispensabile per privacy, affidabilità e funzionalità web moderne.

### 2. Tutti i siti web che usano HTTPS sono considerati affidabili?

**R:** No, non tutti i siti HTTPS sono automaticamente affidabili. HTTPS garantisce solo che la connessione tra il browser e il server è sicura e cifrata, e che il sito ha un certificato digitale valido che conferma l'identità del server.

Tuttavia:

- Un sito può avere HTTPS ma contenere comunque contenuti pericolosi, malware o phishing.
- Il certificato HTTPS non valuta la bontà o l'onestà del contenuto, solo che il sito è chiuso in una connessione sicura.
- Esistono certificati con diversi livelli di verifica (Domain Validation, Organization Validation, Extended Validation), ma anche i più semplici non garantiscono che il sito sia affidabile o legittimo in senso ampio.

Quindi:

- HTTPS è una condizione necessaria per la sicurezza della connessione, ma non sufficiente per considerare un sito completamente affidabile.
  - Bisogna comunque fare attenzione al contenuto e alla reputazione del sito, oltre a usare altri strumenti di sicurezza (antivirus, controlli di phishing, recensioni).
- 

## Conclusione

L'attività svolta oggi ci ha permesso di approfondire il significato e l'importanza del protocollo HTTPS rispetto al tradizionale HTTP.

Rispondere alle domande proposte ci ha aiutato a comprendere che la vera differenza tra i due non riguarda solo aspetti tecnici, ma impatta direttamente sulla sicurezza, sull'affidabilità e sull'esperienza dell'utente nel navigare online.

**HTTP**, nato in un'epoca in cui il web era meno complesso e meno esposto a minacce, trasmette i dati in chiaro, rendendoli vulnerabili a intercettazioni e manipolazioni.

**HTTPS**, invece, rappresenta un'evoluzione necessaria e ormai imprescindibile: protegge le comunicazioni, rende più difficile per gli attaccanti accedere o alterare le informazioni e contribuisce a costruire un clima di fiducia tra utente e sito.

Tuttavia, ho anche imparato che la presenza del lucchetto verde o del prefisso "https://" non equivale automaticamente a un sito sicuro e affidabile.

Un sito malevolo può comunque sfruttare HTTPS per sembrare legittimo, quindi è fondamentale mantenere un atteggiamento critico, consapevole e di scetticismo, soprattutto davanti a link sconosciuti o offerte sospette.

In sintesi, l'analisi svolta mi ha fatto riflettere sul fatto che la sicurezza sul web non dipende solo dalla tecnologia utilizzata, ma anche dalla capacità dell'utente di interpretare correttamente i segnali che la rete ci offre.

HTTPS è uno standard di sicurezza indispensabile, ma non l'unico elemento da considerare per valutare l'affidabilità di un sito web.

---