



Utilizzare Wireshark per esaminare il traffico HTTP e HTTPS.

## Scenario

HyperText Transfer Protocol (HTTP) è un protocollo del livello applicativo che presenta i dati tramite un browser web. Con HTTP, non esiste alcuna protezione per i dati scambiati tra due dispositivi comunicanti.

Con HTTPS, viene utilizzata la crittografia tramite un algoritmo matematico. Questo algoritmo nasconde il vero significato dei dati che vengono scambiati. Ciò avviene tramite l'uso di certificati che possono essere visualizzati successivamente in questo laboratorio.

Indipendentemente dall'utilizzo di HTTP o HTTPS, è consigliato scambiare dati solo con siti web di cui ti fidi. L'uso di HTTPS, infatti, non garantisce che un sito sia affidabile. Gli attori malevoli utilizzano comunemente HTTPS per nascondere le proprie attività.

In questo laboratorio, esplorrai e catturerai il traffico HTTP e HTTPS utilizzando Wireshark.

# Parte 1: Cattura e Visualizzazione del Traffico HTTP

## Obiettivo

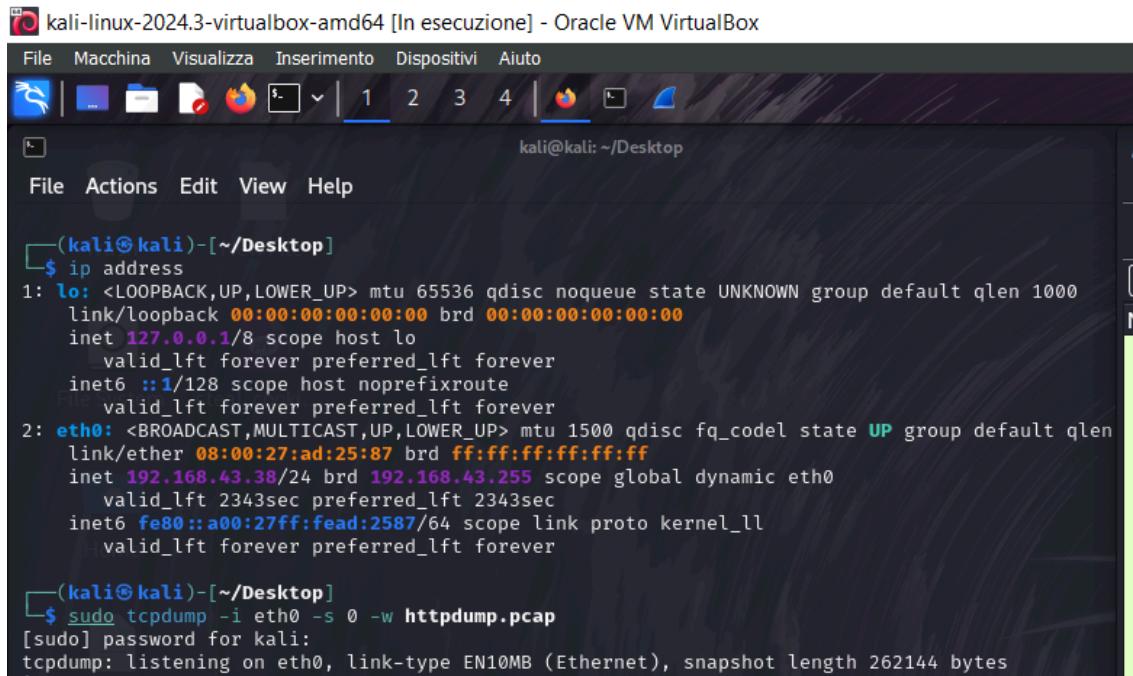
In questa fase, ci concentreremo sulla cattura del traffico HTTP e sull'analisi dei dati trasmessi in chiaro. L'obiettivo è comprendere la mancanza di protezione offerta dal protocollo HTTP e identificare le vulnerabilità associate.

## Passaggi

1. Configurazione della cattura del traffico HTTP:

Utilizzeremo tcpdump per catturare i pacchetti sulla rete. Il comando da eseguire sarà:

```
sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
```



The screenshot shows a terminal window on a Kali Linux desktop. The terminal displays the output of the 'ip address' command, showing network interfaces lo and eth0 with their respective IP configurations. Below this, the command 'sudo tcpdump -i eth0 -s 0 -w httpdump.pcap' is run, and the terminal prompts for a password. The terminal window has a dark theme with a purple header bar containing icons for file, machine, visualization, insertion, devices, and help. The title bar says 'kali-linux-2024.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox'. The bottom status bar shows the user 'kali@kali: ~/Desktop'.

eth0: Indica l'interfaccia di rete che monitoreremo.

-s 0: Garantisce la cattura completa del contenuto dei pacchetti.

-w httpdump.pcap: Salveremo i pacchetti catturati nel file **httpdump.pcap**.

2. Collegamento al sito web <http://www.altoromutual.com/login.jsp>  
 effettuiamo il login con le credenziali utente: Admin password: Admin

The screenshot shows a web browser window with the following details:

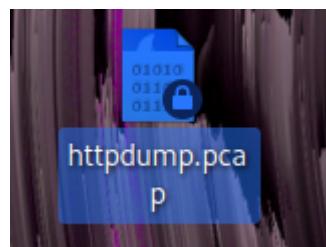
- Address Bar:** Non sicuro | www.altoromutual.com/login.jsp
- Header:** AltoroMutual
- Left Sidebar (ONLINE BANKING LOGIN):**
  - PERSONAL:**
    - Deposit Product
    - Checking
    - Loan Products
    - Cards
    - Investments & Insurance
    - Other Services
  - SMALL BUSINESS:**
    - Deposit Products
    - Lending Services
    - Cards
    - Insurance
    - Retirement
- Right Panel (PERSONAL):**

### Online Banking Login

Username:

Password:

3. Analisi dei pacchetti HTTP con Wireshark:
- Apriremo il file [httpdump.pcap](#) in Wireshark.



- Utilizzeremo il filtro [http](#) per visualizzare esclusivamente il traffico HTTP.

The screenshot shows the Wireshark interface with the following details:

- File Menu:** File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help
- Toolbar:** Includes icons for capture, stop, search, and various analysis tools.
- Search Bar:** http
- Table Headers:** No., Time, Source, Destination, Protocol, Length, Info
- Table Data:** A single row of data is visible: No. 357, Time 61.739553, Source 151.29.122.136, Destination 192.168.43.38, Protocol OCSP, Length 944, Info Response.

- Analizzeremo le richieste e risposte HTTP, osservando come i dati vengano trasmessi in chiaro.

The screenshot shows the Wireshark interface displaying the contents of the 'httpdump.pcap' file. The interface includes:

- File Menu:** File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help
- Toolbar:** Includes icons for capture, stop, search, and various analysis tools.
- Search Bar:** http
- Table Headers:** No., Time, Source, Destination, Protocol, Length, Info
- Table Data:** A list of approximately 15 rows of HTTP traffic, showing details such as source and destination IP addresses, protocol (HTTP or OCSP), length, and info about the response type (e.g., 411 GET /bank/main.jsp HTTP/1.1).

#### 4. Osservazioni:

- Nel traffico HTTP catturato, andremo a identificare una richiesta POST che trasmette dati sensibili in chiaro.

+	902	89.800950	192.168.43.38	65.61.137.117	HTTP	650	POST	/doLogin	HTTP/1.1	(application/x-www-form-urlencoded)
+	905	89.990784	65.61.137.117	192.168.43.38	HTTP	318	HTTP/1.1	302	Found	
+	907	89.993054	192.168.43.38	65.61.137.117	HTTP	633	GET	/bank/main.jsp	HTTP/1.1	
+	916	90.192930	65.61.137.117	192.168.43.38	HTTP	1158	HTTP/1.1	200	OK	(text/html)

- I dettagli osservati includono:
  - Credenziali utente visibili nel payload della richiesta (ad esempio: `uid=Admin`, `passw=Admin`).
  - Nessun meccanismo di crittografia applicato, rendendo i dati vulnerabili ad attacchi come il Man-in-the-Middle (MITM).

```
Hypertext Transfer Protocol
  ▶ POST /doLogin HTTP/1.1\r\n
    Host: www.altoromutual.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    Content-Length: 37\r\n
    Origin: http://www.altoromutual.com\r\n
    Connection: keep-alive\r\n
    Referer: http://www.altoromutual.com/login.jsp\r\n
    Cookie: JSESSIONID=7CE7ABE5D3704BCCA543A8805F8A82A4\r\n
    Upgrade-Insecure-Requests: 1\r\n
\r\n
  [Full request URI: http://www.altoromutual.com/doLogin]
  [HTTP request 1/1]
  [Response in frame: 905]
  File Data: 37 bytes
  ▶ HTML Form URL Encoded: application/x-www-form-urlencoded
    ▶ Form item: "uid" = "Admin"
    ▶ Form item: "passw" = "Admin"
    ▶ Form item: "btnSubmit" = "Login"
```

## Parte 2: Cattura e Visualizzazione del Traffico HTTPS

### Obiettivo

In questa seconda parte, andremo a catturare il traffico HTTPS seguendo la stessa procedura utilizzata per il traffico HTTP. L'obiettivo sarà osservare come i dati vengono trasmessi in forma crittografata e comprendere le differenze rispetto a HTTP.

### Passaggi

#### 1. Configurazione della cattura del traffico HTTPS:

Come per il traffico HTTP, utilizzeremo `tcpdump` per catturare i pacchetti di rete. Il comando sarà:

```
sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap
```

```
(kali㉿kali)-[~/Desktop]
$ sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes (7.51606
^C6446 packets captured
6446 packets received by filter
0 packets dropped by kernel
```

The terminal shows the command being run and the resulting file `httpsdump.pcap`. The file size is approximately 7.51606 MB.

eth0: L'interfaccia di rete monitorata.

-s 0: Per catturare i pacchetti nella loro interezza.

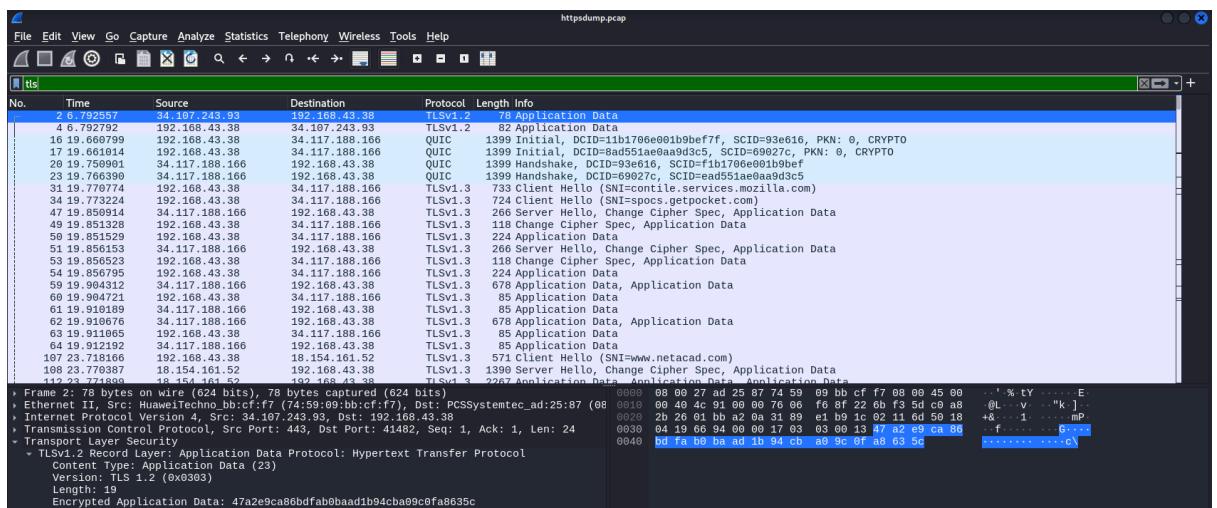
-w `httpsdump.pcap`: Salveremo i pacchetti catturati nel file `httpsdump.pcap`.

#### 2. Analisi dei pacchetti HTTPS con Wireshark:

- Apriremo il file `httpsdump.pcap` in Wireshark.
- Utilizzeremo il filtro `tls` per visualizzare il traffico HTTPS.
- A differenza del traffico HTTP, noteremo che i dati trasmessi non saranno visibili in chiaro, poiché protetti da crittografia.

#### 3. Osservazioni:

- Nel traffico HTTPS catturato, non sarà possibile leggere direttamente il contenuto delle richieste e delle risposte, poiché le informazioni sono protette da TLS.



- L'unica possibilità di analizzare il contenuto crittografato sarebbe utilizzare una chiave privata del server o configurare un proxy per intercettare e decrittare il traffico (ad esempio tramite Burp Suite).

## Conclusione

Seguendo la stessa procedura di cattura utilizzata per HTTP, abbiamo potuto osservare la principale differenza di HTTPS: i dati trasmessi sono crittografati, rendendoli inaccessibili senza una chiave di decrittazione. Questo dimostra come HTTPS protegga le informazioni sensibili durante la trasmissione, anche se non garantisce automaticamente che il sito sia affidabile.