

# Simulazione Architettura Client-Server e Analisi del Traffico HTTP/HTTPS con Wireshark

## 1. Configurazione dell'Ambiente

Macchine virtuali utilizzate:

**Kali Linux (Server)** – IP: 192.168.32.100

**Windows 7 (Client)** – IP: 192.168.32.101

**Obiettivo:** Simulare un'architettura client-server in cui il client (Windows) richiede una risorsa al server (Kali) tramite HTTP e HTTPS, intercettando e analizzando il traffico con Wireshark.

## 2. Configurazione del Server HTTP su Kali Linux

Configurazione della risoluzione dei nomi di dominio

Modificato il file /etc/hosts su Kali per risolvere l'hostname epicode.internal:

192.168.32.100 epicode.internal

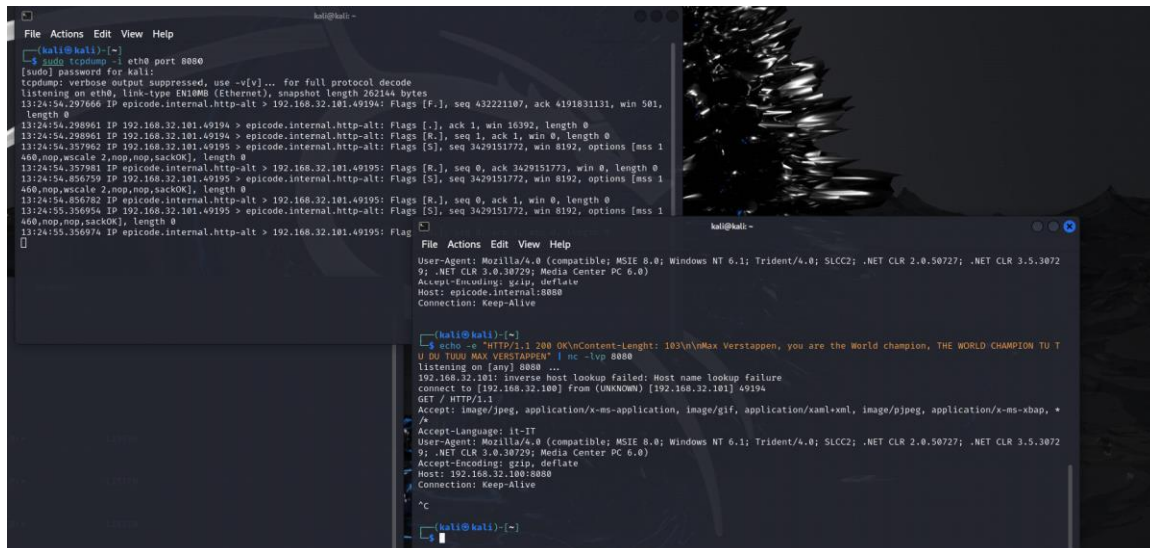
Modificato il file C:\Windows\System32\drivers\etc\hosts su Windows per la risoluzione locale:

192.168.32.100 epicode.internal

Verificata la corretta risoluzione del nome tramite ping epicode.internal da Windows.

Avvio del server HTTP su Kali

echo -e "HTTP/1.1 200 OK\nContent-Length: 103\n\nProf mi perdoni, ma non sono nato  
ferrarista!" | nc -lvp 8080



```
kali@kali:~$ sudo tcpdump -i eth0 port 8080
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type (Ethernet), snapshot length 262144 bytes
13:24:54.297666 IP epicode.internal.http-alt > 192.168.32.101.49194: Flags [F.], seq 432221187, ack 4191831131, win 501,
length 0
13:24:54.298961 IP 192.168.32.101.49194 > epicode.internal.http-alt: Flags [.], ack 1, win 16392, length 0
13:24:54.298961 IP 192.168.32.101.49194 > epicode.internal.http-alt: Flags [R.], seq 1, ack 1, win 0, length 0
13:24:54.357962 IP 192.168.32.101.49195 > epicode.internal.http-alt: Flags [S], seq 3429151772, win 8192, options [mss 1
460,nop,wscale 2,nop,nop,sackOK], length 0
13:24:54.357981 IP epicode.internal.http-alt > 192.168.32.101.49195: Flags [R.], seq 0, ack 3429151773, win 0, length 0
13:24:54.456759 IP 192.168.32.101.49195 > epicode.internal.http-alt: Flags [S], seq 3429151772, win 8192, options [mss 1
460,nop,wscale 2,nop,nop,sackOK], length 0
13:24:54.456782 IP epicode.internal.http-alt > 192.168.32.101.49195: Flags [R.], seq 0, ack 1, win 0, length 0
13:24:55.356954 IP 192.168.32.101.49195 > epicode.internal.http-alt: Flags [S], seq 3429151772, win 8192, options [mss 1
460,nop,nop,sackOK], length 0
13:24:55.356974 IP epicode.internal.http-alt > 192.168.32.101.49195: Flag

kali@kali:~$
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.3072
9; .NET CLR 3.0.30729; Media Center PC 6.0)
Accept-Encoding: gzip, deflate
Host: epicode.internal:8080
Connection: Keep-Alive

kali@kali:~$
echo -e "HTTP/1.1 200 OK\nContent-Length: 103\n\nMax Verstappen, you are the World champion, THE WORLD CHAMPION TU T
U DU TUUU MAX VERSTAPPEN" | nc -lvp 8080
listening on [any] 8080 ...
192.168.32.101: Inverse host lookup failed: Host name lookup failure
connect to [192.168.32.100] from (UNKNOWN) [192.168.32.101] 49194
GET / HTTP/1.1
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/png, application/x-ms-xap, *
/*
Accept-Language: it-IT
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.3072
9; .NET CLR 3.0.30729; Media Center PC 6.0)
Accept-Encoding: gzip, deflate
Host: 192.168.32.100:8080
Connection: Keep-Alive

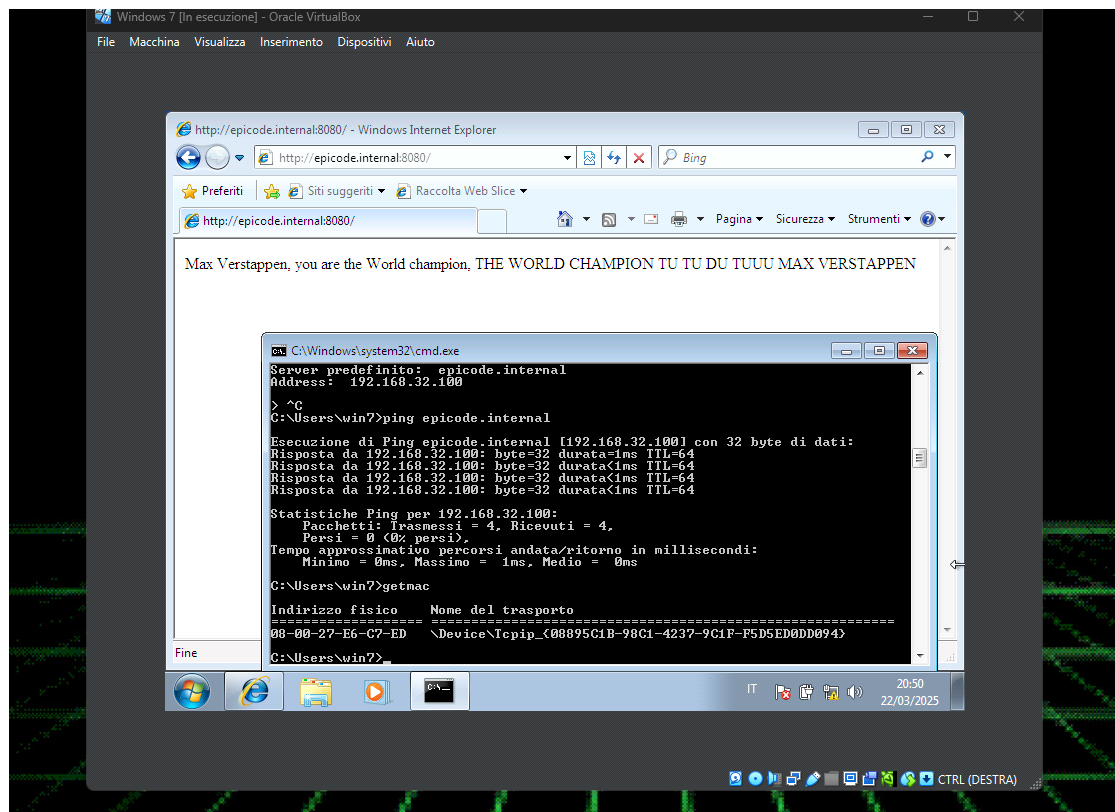
^C

kali@kali:~$
```

## Test della comunicazione

Apertura di Internet Explorer su Windows e accesso a <http://epicode.internal:8080>.

Verifica della risposta corretta: il messaggio che volevo che leggesse, viene restituito dal server.



### 3. Cattura e Analisi del Traffico HTTP con Wireshark

Avvio di Wireshark su Kali e selezione dell'interfaccia di rete eth0.

Applicazione del filtro HTTP: http.

Analisi del traffico intercettato:

Richiesta GET del client (Windows → Kali):

MAC sorgente: indirizzo MAC di Windows.

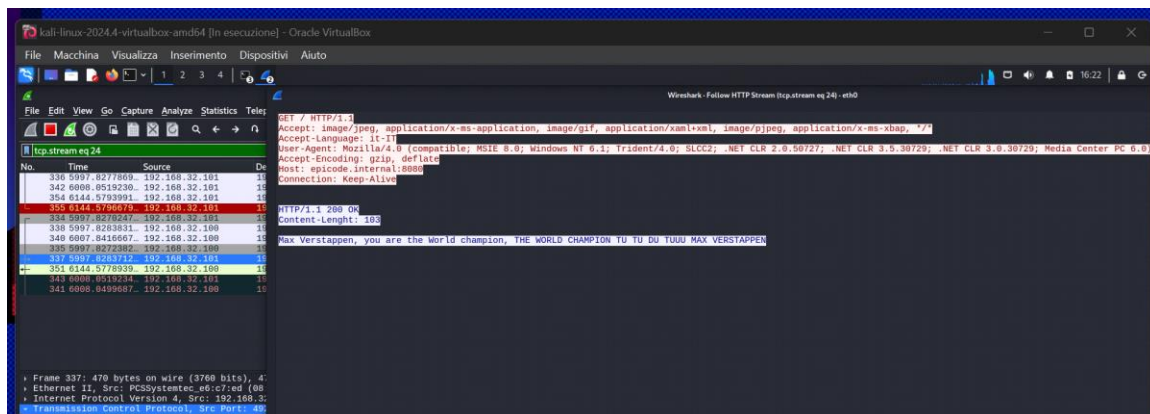
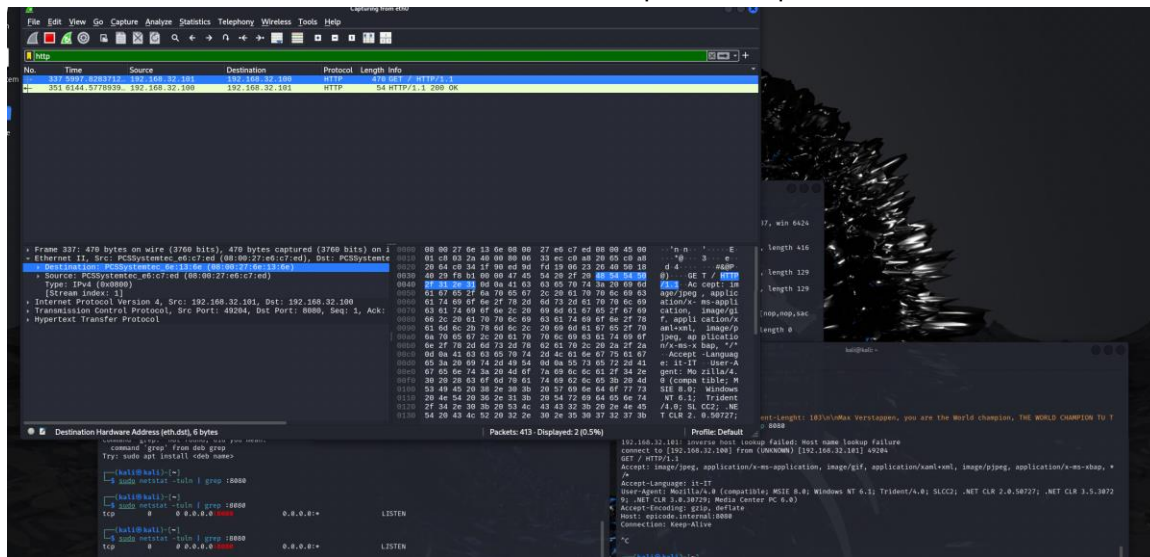
MAC destinazione: indirizzo MAC di Kali.

Risposta HTTP 200 OK del server:

MAC sorgente: indirizzo MAC di Kali.

MAC destinazione: indirizzo MAC di Windows.

Verifica del contenuto della richiesta e della risposta nel pacchetto intercettato.



Conclusione: Il traffico HTTP è stato correttamente identificato e analizzato, confermando la comunicazione tra client e server.

#### 4. Configurazione del Server HTTPS e Analisi del Traffico

Attivazione del supporto HTTPS su Apache:

```
sudo a2ensite default-ssl.conf
```

```
sudo systemctl reload apache2
```

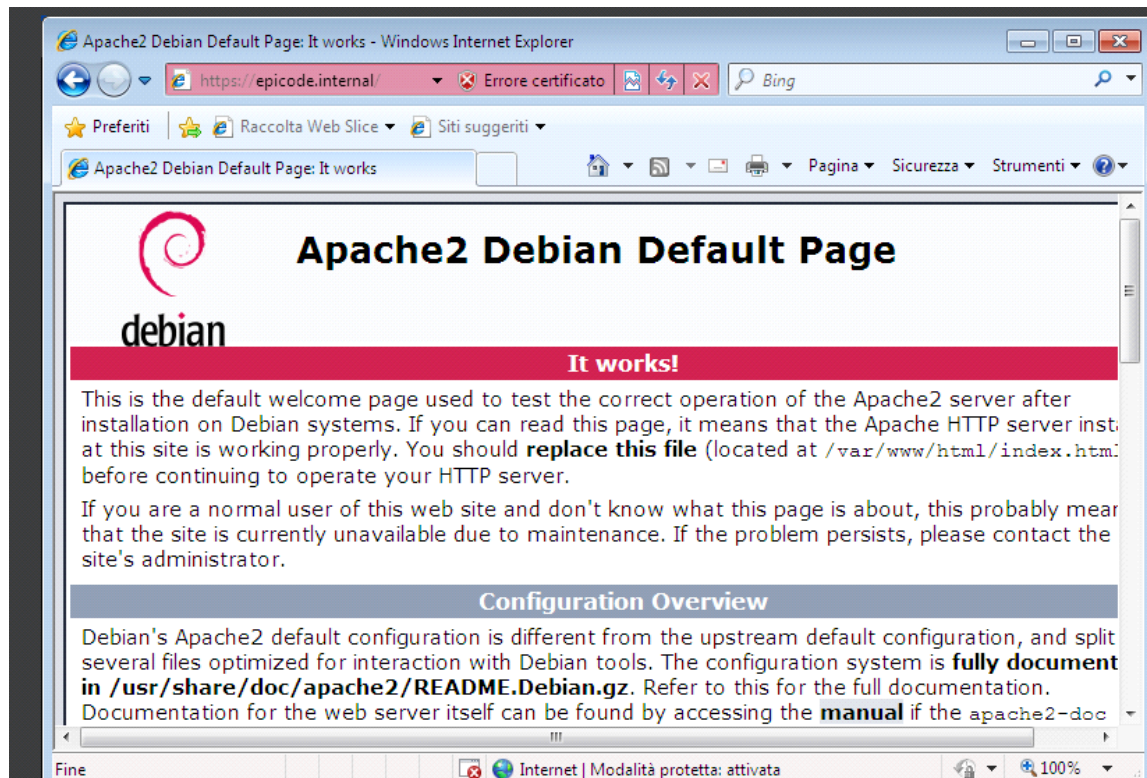
Verificato che Apache stia ascoltando sulla porta 443:

```
sudo ss -tln | grep 443
```

Test della comunicazione HTTPS

Accesso da Windows a <https://epicode.internal>.

Superamento dell'avviso di certificato non valido e verifica della risposta corretta dal server.



Cattura del traffico HTTPS con Wireshark

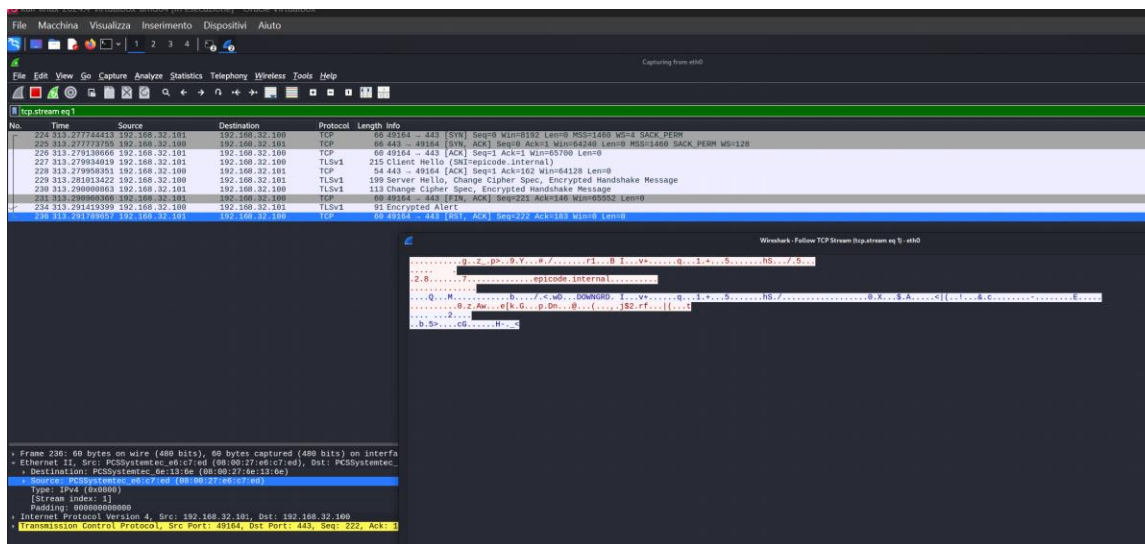
Applicazione del filtro: tcp.port == 443.

Analisi dei pacchetti SSL/TLS:

Client Hello e Server Hello.

Scambio di certificati e chiavi.

Dati della richiesta criptati.



## Confronto tra HTTP e HTTPS:

HTTP: pacchetti in chiaro, contenuto leggibile.

HTTPS: pacchetti crittografati, impossibile visualizzare i dati trasmessi.

## 5. Conclusioni

L'analisi del traffico ha evidenziato come il protocollo HTTPS garantisca la sicurezza delle informazioni rispetto a HTTP. In particolare:

L'handshake SSL/TLS stabilisce una connessione sicura tra client e server.

E' possibile vedere l'indirizzo ip, il macaddress del destinatario e del richiedente, tuttavia il contenuto della richiesta e della risposta non è visibile in chiaro in HTTPS, proteggendo i dati da intercettazioni.

In HTTP, invece, tutti i dati trasmessi sono leggibili direttamente nei pacchetti intercettati.

Questa simulazione ha permesso di comprendere l'importanza della crittografia nel traffico

web e il ruolo di Wireshark nell'analisi del traffico di rete.

*Panagiotis Diamantopoulos*