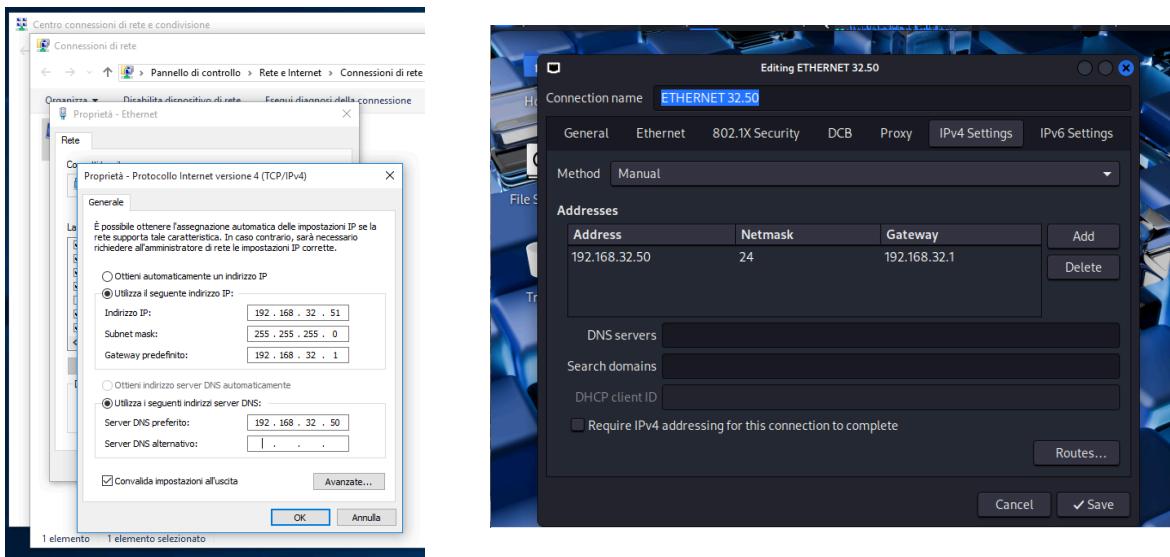


# PROGETTO FINE MODULO M1

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.51 (Windows) richiede tramite web browser una risorsa all'hostname episode.internal che risponde all'indirizzo 192.168.32.50 (Kali).

Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS. Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

Sono partito impostando rete interna delle due macchine virtuali su VirtualBox, poi ho impostato indirizzi IP manualmente su KALI 192.168.32.50 e poi su Windows 192.168.32.51 e infine sempre su Windows ho inserito come DNS predefinito quello dato a KALI ovvero 192.168.32.50. Immagini Windows a sinistra e a destra KALI.



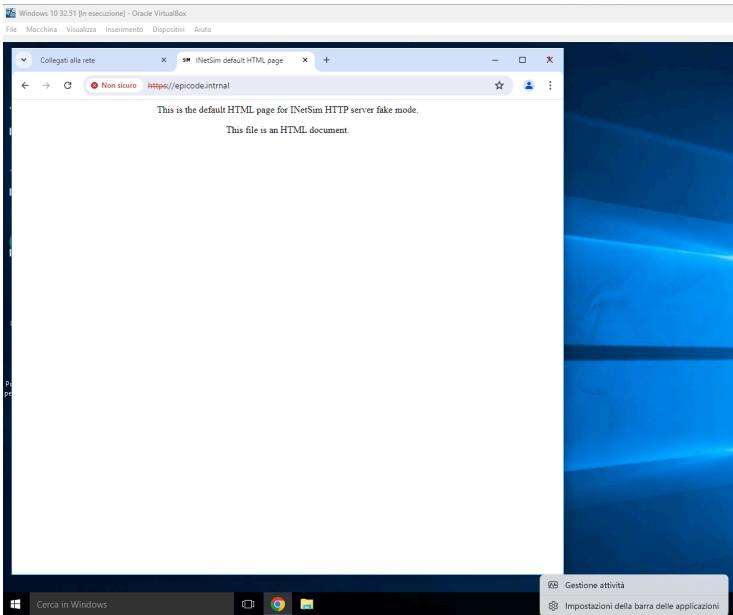
Dopo aver attribuito gli indirizzi IP alle macchine virtuali e aver messo un DNS di riferimento a Windows, su kali ho modificato le impostazioni di Inetsim rendendo attivi i servizi DNS, HTTP e HTTPS per poter simulare dei servizi web. Ho inserito su Inetsim nella riga dns\_default.ip e nella riga service\_bind\_address l'indirizzo 192.168.32.50. Poi ho inserito nella riga dns\_default\_name episode, mentre nella riga dns\_default\_domainname internal. ho cancellato i cancellietti ha inizio riga per poter rendere tutti i servizi necessari attivi. Dopodiché ho salvato le modifiche e sembrava tutto pronto ma una volta provato ad avviare Inetsim il DNS dava un errore:

```
(kali㉿kali)-[~]
└─$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file...
Configuration file parsed successfully.
== INetSim main process started (PID 4703) ==
Session ID: 4703
Listening on: 192.168.32.50
Real Date/Time: 2025-10-19 13:51:56
Fake Date/Time: 2025-10-19 13:51:56 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 4705)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at
/usr/share/perl5/INetSim/DNS.pm line 69.
* http_80_tcp - started (PID 4706)
* https_443_tcp - started (PID 4707)
done.
Simulation running.
```

Ho risolto l'errore copiandolo e incollandolo su google, sono giunto ad una pagina di discussioni dove veniva spiegato che per far funzionare quel servizio di Inetsim bisognava installare una versione più vecchia del servizio DNS, nella conversazione ho trovato il link per il download e tutti i comandi utili alla sua installazione, eccetto il comando che serviva per decomprimere il file scaricato, il quale mi ha fatto sudare per essere trovato. Dunque ho collegato Kali ad internet per poter scaricare il file e una volta eseguiti tutti i comandi per la sua installazione il servizio DNS di Inetsim ha iniziato a funzionare.

```
(kali㉿kali)-[~]
└─$ sudo nano /etc/inetsim/inetsim.conf
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file...
Configuration file parsed successfully.
== INetSim main process started (PID 1882) ==
Session ID: 1882
Listening on: 192.168.32.50
Real Date/Time: 2025-10-19 14:41:32
Fake Date/Time: 2025-10-19 14:41:32 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 1884)
* https_443_tcp - started (PID 1885)
done.
Simulation running.
```

Una volta avviato Inetsim senza errori ho tentato la ricerca da il Browser di Windows, e la ricerca è andata in porto:



Visto che tutto è andato come dovrebbe ho iniziato a osservare i pacchetti con wireshark, iniziando ad osservare una connessione HTTPS:

525 31.124489242	192.168.32.51	192.168.32.50	TCP	60 49506 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
526 31.124861733	192.168.32.51	192.168.32.50	TLSv1.3	1863 Client Hello (SNI=optimizationguide-pa.googleapis.com)
527 31.124867946	192.168.32.50	192.168.32.51	TCP	54 443 → 49506 [ACK] Seq=1 Ack=1810 Win=62464 Len=0
528 31.134480346	192.168.32.50	192.168.32.51	TLSv1.3	1497 Server Hello, Change Cipher Spec, Application Data, App
529 31.135555938	192.168.32.51	192.168.32.50	TLSv1.3	84 Change Cipher Spec, Application Data
530 31.135791278	192.168.32.51	192.168.32.50	TCP	60 49506 → 443 [FIN, ACK] Seq=1840 Ack=1444 Win=64256 Len=0
531 31.138456740	192.168.32.50	192.168.32.51	TCP	54 443 → 49506 [FIN, ACK] Seq=1444 Ack=1841 Win=62464 Len=0
532 31.138771095	192.168.32.51	192.168.32.50	TCP	60 49506 → 443 [ACK] Seq=1841 Ack=1445 Win=64256 Len=0
533 31.263514711	192.168.32.51	192.168.32.255	NBNS	92 Name query NB WPAD<00>

Questo è lo screenshot della risposta del server al client, si possono trovare gli indirizzi MAC ed attribuirli agli indirizzi IP, perché sapendo che è la risposta del server il source IP è 192.168.32.50 e nella riga evidenziata troviamo il source mac address che è 08:00:27:d1:f8:5d e troviamo anche il destination address quindi IP 192.168.32.51 e il suo MAC address 08:00:27:ce:fb:27 che sono gli indirizzi del client.

Da questa foto si possono anche rilevare il numero di bytes che sono passati nel cavo.

```
Acknowledgment number (raw): 78452651
0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x018 (PSH, ACK)
  Window: 488
    [Calculated window size: 62464]
    [Window size scaling factor: 128]
  Checksum: 0xc773 [unverified]
    [Checksum Status: Unverified]
  Urgent Pointer: 0
  ▶ [Timestamps]
  ▶ [SEQ/ACK analysis]
TCP payload (1443 bytes)
```

# OSSERVAZIONE PACCHETTI HTTPS CON WIRESHARK

Visto che tutto è andato come dovrebbe ho iniziato a osservare i pacchetti con wireshark, iniziando ad osservare una connessione HTTPS:

525	31.124489242	192.168.32.51	192.168.32.50	TCP	60	49506 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
526	31.124861733	192.168.32.51	192.168.32.50	TLSv1.3	1863	Client Hello (SNI=optimizationguide-pa.googleapis.com)
527	31.124867946	192.168.32.50	192.168.32.51	TCP	54	443 → 49506 [ACK] Seq=1 Ack=1810 Win=62464 Len=0
528	31.134486346	192.168.32.50	192.168.32.51	TLSv1.3	1497	Server Hello, Change Cipher Spec, Application Data, App
529	31.135555938	192.168.32.51	192.168.32.50	TLSv1.3	84	Change Cipher Spec, Application Data
530	31.135791278	192.168.32.51	192.168.32.50	TCP	60	49506 → 443 [FIN, ACK] Seq=1840 Ack=1444 Win=64256 Len=0
531	31.138456740	192.168.32.50	192.168.32.51	TCP	54	443 → 49506 [FIN, ACK] Seq=1444 Ack=1841 Win=62464 Len=0
532	31.138771095	192.168.32.51	192.168.32.50	TCP	60	49506 → 443 [ACK] Seq=1841 Ack=1445 Win=64256 Len=0
533	31.236514711	192.168.32.51	192.168.32.255	NBNS	92	Name query NB_WPAD<0>
▶ Frame 528: 1497 bytes on wire (11976 bits), 1497 bytes captured (11976 bits) on interface eth0, id 0						
▶ Ethernet II, Src: PCSSystemtec_d1:f8:5d (08:00:27:d1:f8:5d), Dst: PCSSystemtec_ce:fb:27 (08:00:27:ce:fb:27)						
▶ Destination: PCSSystemtec_ce:fb:27 (08:00:27:ce:fb:27)						
▶ Source: PCSSystemtec_d1:f8:5d (08:00:27:d1:f8:5d)						
Type:	IPv4 (0x0800)					
[Stream index:	2]					
▶ Internet Protocol Version 4, Src: 192.168.32.50, Dst: 192.168.32.51						
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 49506, Seq: 1, Ack: 1810, Len: 1443						
▶ Transport Layer Security						

Questo è lo screenshot della risposta del server al client, si possono trovare gli indirizzi MAC ed attribuirli agli indirizzi IP, perché sapendo che è la risposta del server il source IP è 192.168.32.50 e nella riga evidenziata troviamo il source mac address che è 08:00:27:d1:f8:5d e troviamo anche il destination address quindi IP 192.168.32.51 e il suo MAC address 08:00:27:ce:fb:27 che sono gli indirizzi del client. Da questa foto si possono anche rilevare il numero di bytes che sono passati nel cavo.

Da questa foto si possono anche rilevare il numero di bytes che sono passati nei cavi.

```
Acknowledgment number (raw): 78452651
0101 .... = Header Length: 20 bytes (5)
▶ Flags: 0x018 (PSH, ACK)
Window: 488
[Calculated window size: 62464]
[Window size scaling factor: 128]
Checksum: 0xc773 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
▶ [Timestamps]
▶ [SEQ/ACK analysis]
TCP payload (1443 bytes)
```

gli unici altri dati che si riescono a carpire su HTTPS sono la lunghezza dell'header e del payload in bytes il resto dei dati è tutto cifrato e impossibile da leggere.

## OSSERVAZIONE PACCHETTI CON WIRESHARK SU HTTP

The Wireshark interface displays a list of network frames. Frame 471 is selected, showing an HTTP GET request from 192.168.32.51 to 192.168.32.50. The details pane shows the request URL as /sample.txt. The bytes pane shows the raw HTTP traffic.

Frame 471 details:

- Source: 192.168.32.51
- Destination: 192.168.32.50
- Protocol: HTTP
- HTTP Request Method: GET
- HTTP Request Path: /sample.txt
- HTTP Version: HTTP/1.1
- HTTP Headers:
  - Host: episode.internal
  - Connection: keep-alive
  - Upgrade-Insecure-Requests: 1
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
  - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7
  - Accept-Encoding: gzip, deflate
  - Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
- HTTP Payload: [Response in frame: 476]
- [Full request URI: http://episode.internal/sample.txt]

Qui sopra abbiamo una richiesta del client ad un server in HTTP, si possono vedere in chiaro il tipo di richiesta, il formato del file richiesto, tipo di connessione, il nome dell'host, da quale programma sta effettuando la richiesta e in che lingua è stata effettuata la richiesta. e poi in fondo c'è anche l'URI completo

The Wireshark interface displays a list of network frames. Frame 476 is selected, showing an HTTP 200 OK response from 192.168.32.50 to 192.168.32.51. The details pane shows the response status code and content type. The bytes pane shows the raw HTTP traffic.

Frame 476 details:

- Source: 192.168.32.50
- Destination: 192.168.32.51
- Protocol: HTTP
- HTTP Response Status: 200 OK
- HTTP Response Reason: (text/plain)
- HTTP Headers:
  - Server: INetSim HTTP Server
  - Date: Sun, 19 Oct 2025 18:59:14 GMT
  - Connection: Close
  - Content-Length: 97
  - Content-Type: text/plain
- HTTP Payload: [Request in frame: 471]  
[Time since request: 0.017998683 seconds]  
[Request URI: /sample.txt]

Anche dalla risposta del server possiamo carpire molte informazioni: ad esempio che la risposta è stata positiva, informazioni sul server, la data con giorno e orario e la lunghezza del contenuto in bytes.

## CONCLUSIONE

Le principali differenze tra HTTP e HTTPS sono che utilizzano numeri di porta differenti HTTP (80) e HTTPS (403), e soprattutto i dati su HTTP possono essere visti in chiaro se intercettati da qualcuno, mentre su HTTPS i pacchetti vengono cifrati da TLS cosicché se venissero intercettati da qualcuno questi siano incomprensibili garantendo la privacy dei client che intendono usare i servizi di un sito web.