

PROGETTO FINALE M1

Introduzione all'hacking

Nel seguente report andremo ad analizzare e confrontare il traffico di rete da una richiesta HTTPS e una richiesta HTTP tra un client Windows e un server Kali Linux, con l'obiettivo di comprendere le differenze tra i due servizi.

Per l'esecuzione andremo a simulare le richieste nell'ambiente virtuale, utilizzando dei tool forniti dal sistema Kali, nello specifico:

inetsim, che permette di simulare servizi internet, con cui quindi simuleremo i servizi HTTP e HTTPS.

dnsmasq, che utilizzeremo per risolvere le richieste DNS della nostra rete interna, nel nostro caso indirizzeremo le richieste all'indirizzo `epicode.internal` verso l'IP dei servizi HTTP e HTTPS. Anche *Inetsim* permette di configurare un servizio di DNS, ma per un difetto del programma non risulta una soluzione stabile. Ho scelto quindi *dnsmasq* dato che è un tool che presenta una notevole la semplicità di configurazione.

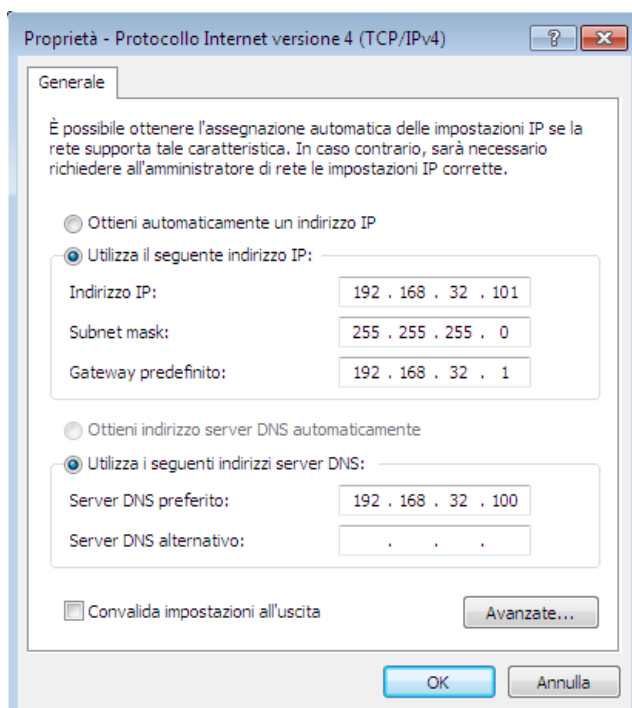
Wireshark, un analizzatore di protocolli di rete, lo utilizzeremo per catturare e analizzare il traffico di rete in tempo reale.

CONFIGURAZIONE DEL LABORATORIO

Per l'esecuzione dell'esercitazione andiamo a configurare i sistemi operativi client e server, e gli strumenti di simulazione dei servizi.

CLIENT (WINDOWS)

Sul sistema Windows 7 andiamo ad impostare nelle configurazioni di rete:



Indirizzo IP:	192.168.32.101
Subnet mask:	255.255.255.0
Gateway:	192.168.32.1
Server DNS Predefinito:	192.168.32.100

In questo modo abbiamo dato un indirizzo al sistema Windows, e impostato come DNS di riferimento l'IP del server Kali, che andremo a configurare nei passaggi successivi.

SERVER (KALI)

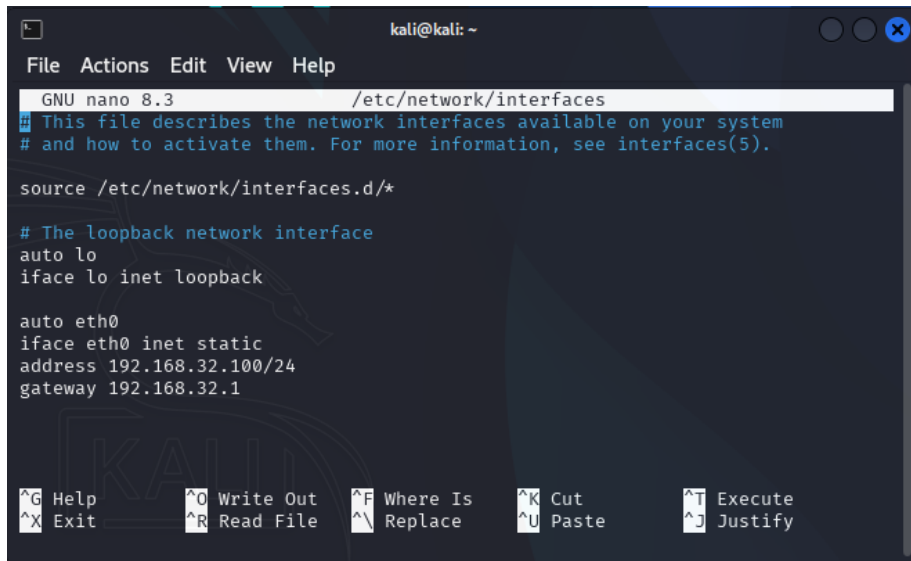
CONFIGURAZIONE DI RETE

Passiamo ora alla configurazione del Server, come prima cosa impostiamo l'indirizzo IP al nostro sistema, quindi dal prompt dei comandi digitare:

```
sudo nano /etc/network/interface
```

Si aprirà il file di configurazione dell'interfaccia di rete, lo modifichiamo aggiungendo:

```
auto eth0
iface eth0 inet static
address 192.168.50.100/24
gateway 192.168.50.1
```



```
kali@kali: ~
File Actions Edit View Help
GNU nano 8.3 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1

^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

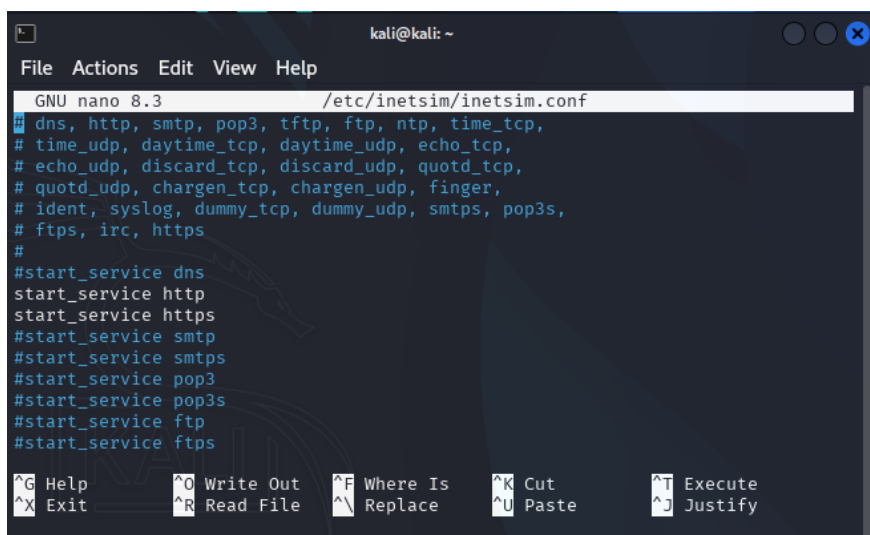
In questo modo abbiamo impostato un IP statico sull'interfaccia di rete eth0.

INETSIM

Adesso passiamo alla configurazione del tool di simulazione di servizi Inetsim. Apriamo il file di configurazione dal prompt dei comandi digitando:

```
sudo nano /etc/inetsim/inetsim.conf
```

All'apertura del file troveremo la lista di tutti i servizi disponibili simulabili da questo tool, noi lasceremo attivi solo HTTP e HTTPS, che sono quelli che andremo ad analizzare, quindi disattiviamo tutti gli altri aggiungendo un simbolo del cancelletto [#] di fronte ad ogni riga, come mostrato in figura.



```
kali@kali: ~
File Actions Edit View Help
GNU nano 8.3 /etc/inetsim/inetsim.conf
# dns, http, smtp, pop3, tftp, ftp, ntp, time_tcp,
# time_udp, daytime_tcp, daytime_udp, echo_tcp,
# echo_udp, discard_tcp, discard_udp, quotd_tcp,
# quotd_udp, chargen_tcp, chargen_udp, finger,
# ident, syslog, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, irc, https
#
#start_service dns
start_service http
start_service https
#start_service smtp
#start_service smtps
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service ftps

^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Scorrendo in basso nel file di configurazione andiamo a modificare il `service bind address`. Questa impostazione determina l'indirizzo IP sul quale il server simula i servizi e ascolta le richieste del client.

```
#####  
# service_bind_address  
#  
# IP address to bind services to  
#  
# Syntax: service_bind_address <IP address>  
#  
# Default: 127.0.0.1  
#  
service_bind_address 192.168.32.100
```

Andiamo quindi a impostare l'indirizzo IP del server `192.168.32.100`, come mostrato in figura

DNSMASQ

Questo tool non è presente di default sul sistema Kali, va quindi installato eseguendo dal prompt dei comandi:

```
sudo apt update  
sudo apt install dnsmasq
```

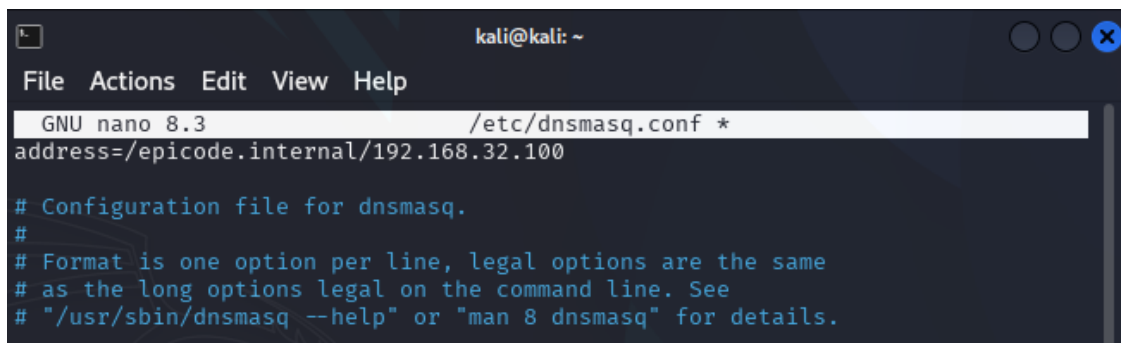
Una volta completata la rapida installazione, passiamo alla configurazione del servizio andando a modificare il file di configurazione del programma. Digitiamo sul prompt:

```
sudo nano /etc/dnsmasq.conf
```

Una volta aperto il file, aggiungiamo semplicemente la riga:

```
address=/epicode.internal/192.168.32.100
```

Questa riga definisce una mappatura DNS personalizzata. Qualsiasi richiesta per il dominio `epicode.internal` verrà risolta all'indirizzo IP `192.168.32.100`.



```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 8.3 /etc/dnsmasq.conf *  
address=/epicode.internal/192.168.32.100  
  
# Configuration file for dnsmasq.  
#  
# Format is one option per line, legal options are the same  
# as the long options legal on the command line. See  
# "/usr/sbin/dnsmasq --help" or "man 8 dnsmasq" for details.
```

AVVIO DELLA SIMULAZIONE

Ora siamo pronti per l'avvio della simulazione, sul sistema server andiamo ad attivare:

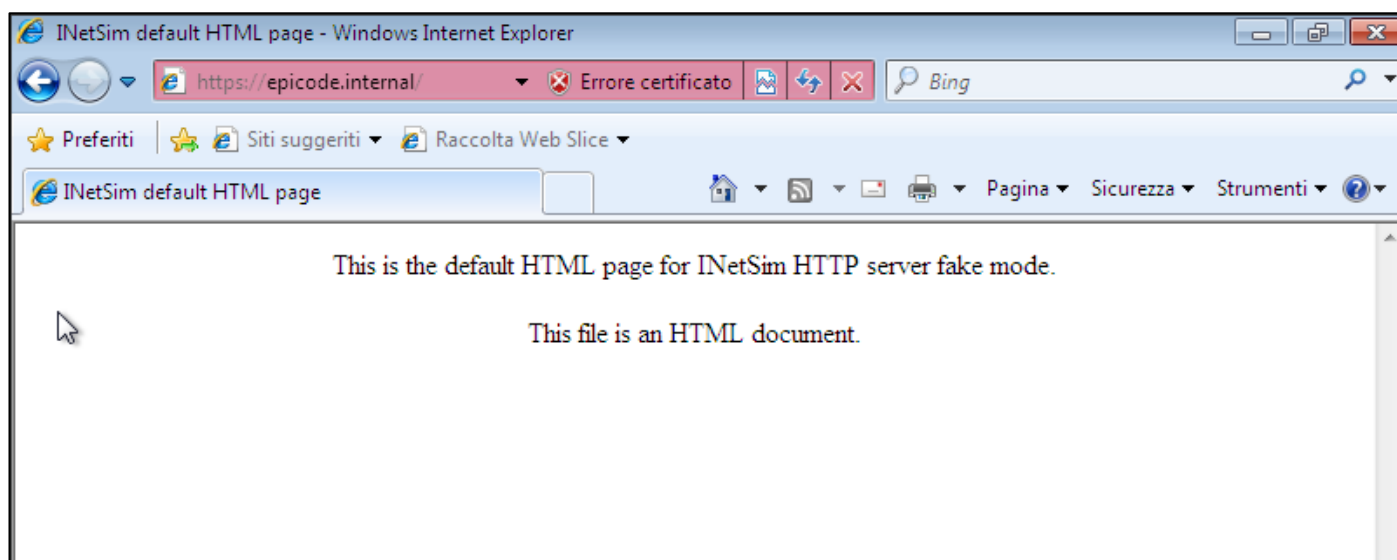
- inetsim, digitando su prompt: `sudo inetsim`
- dnsmasq, aprendo un'altra finestra di prompt e digitando: `sudo dnsmasq`
- wireshark, cerchiamo nella barra delle applicazioni questo programma già installato nel sistema Kali, e nella finestra di avvio, quando viene chiesta la rete su cui intercettare le comunicazioni selezioniamo la rete `eth0` precedentemente configurata.

In questo momento i servizi HTTPS, HTTP, e DNS sono in funzione, e wireshark è in ascolto in attesa dei dati trasmessi

Spostiamoci sul sistema client Windows

Da windows ora apriamo il browser di internet e nella barra delle applicazioni digitiamo:

<https://epicode.internal>



Se tutte le configurazioni sono corrette e i servizi funzionano ci troveremo di fronte a questa pagina web fittizia, fornita da alcuni file sample del tool *Inetsim*.

Analizzando il percorso delle richieste, possiamo notare i seguenti passaggi:

Richiesta di Conversione del Nome di Dominio:

Il client invia una richiesta di conversione del dominio epicode.internal al server DNS configurato.

Risposta del DNS:

Il server DNS, dopo aver elaborato la richiesta, converte il dominio nell'indirizzo IP corrispondente, che è quello del server.

Richiesta al Servizio HTTPS:

Una volta ottenuto l'indirizzo IP, il browser del client invia una richiesta direttamente al servizio HTTPS, richiedendo i pacchetti necessari per visualizzare la pagina.

ANALISI CON WIRESHARK

Per analizzare nel dettaglio tutti questi passaggi, ci sposteremo nuovamente su Kali, dove Wireshark è rimasto in ascolto e ha catturato i pacchetti delle comunicazioni. Questo ci permetterà di esaminare i dettagli delle richieste e delle risposte, fornendo una visione chiara del flusso di dati e delle interazioni tra il client, il DNS e il server

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A display filter bar shows 'Apply a display filter ... <Ctrl-/>'. The main packet list pane shows a table of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets include ARP requests, TCP SYN and ACK sequences, TLSv1 client hello and key exchange, and LLMNR/NBNS queries. The bottom pane shows the detailed view of the selected packet (Frame 1), displaying the raw bytes in hexadecimal and ASCII, and the packet structure tree on the left.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.100
2	0.000216234	PCSSystemtec_04:42:...	PCSSystemtec_4f:8d:...	ARP	42	192.168.32.100 is at 08:00:27:04:42:...
3	0.000901811	192.168.32.101	192.168.32.100	TCP	66	49161 → 443 [SYN] Seq=0 Win=8192 Len=0
4	0.001267807	192.168.32.100	192.168.32.101	TCP	66	443 → 49161 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
5	0.002024101	192.168.32.101	192.168.32.100	TCP	60	49161 → 443 [ACK] Seq=1 Ack=1 Win=0 Len=0
6	0.003481556	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello (SNI=epicode.internal)
7	0.003506031	192.168.32.100	192.168.32.101	TCP	54	443 → 49161 [ACK] Seq=1 Ack=162 Win=0 Len=0
8	0.034868297	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange
9	0.045671171	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec
10	0.046535679	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake
11	0.065996585	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.1
12	0.239114660	192.168.32.101	192.168.32.100	TCP	60	49161 → 443 [ACK] Seq=296 Ack=1379 Len=0
13	1.034754037	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.1
14	2.034461457	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.1
15	3.205848777	fe80::a4fb:a3dd:cf1...	ff02::1:3	LLMNR	84	Standard query 0x7b04 A wpad
16	3.205849178	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x7b04 A wpad
17	3.317536813	fe80::a4fb:a3dd:cf1...	ff02::1:3	LLMNR	84	Standard query 0x7b04 A wpad
18	3.317537484	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x7b04 A wpad
19	3.519666493	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
20	4.268085998	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
21	5.018404608	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
22	5.779001084	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.1
23	6.522500524	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.1

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0
Ethernet II, Src: PCSSystemtec_4f:8d:cd (08:00:27:4f:8d:cd), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)

0000 ff ff ff ff ff ff 08 00 27 4f 8d cd 08 06 00 01
0010 08 00 06 04 00 01 08 00 27 4f 8d cd c0 a8 20 65
0020 00 00 00 00 00 00 c0 a8 20 64 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00

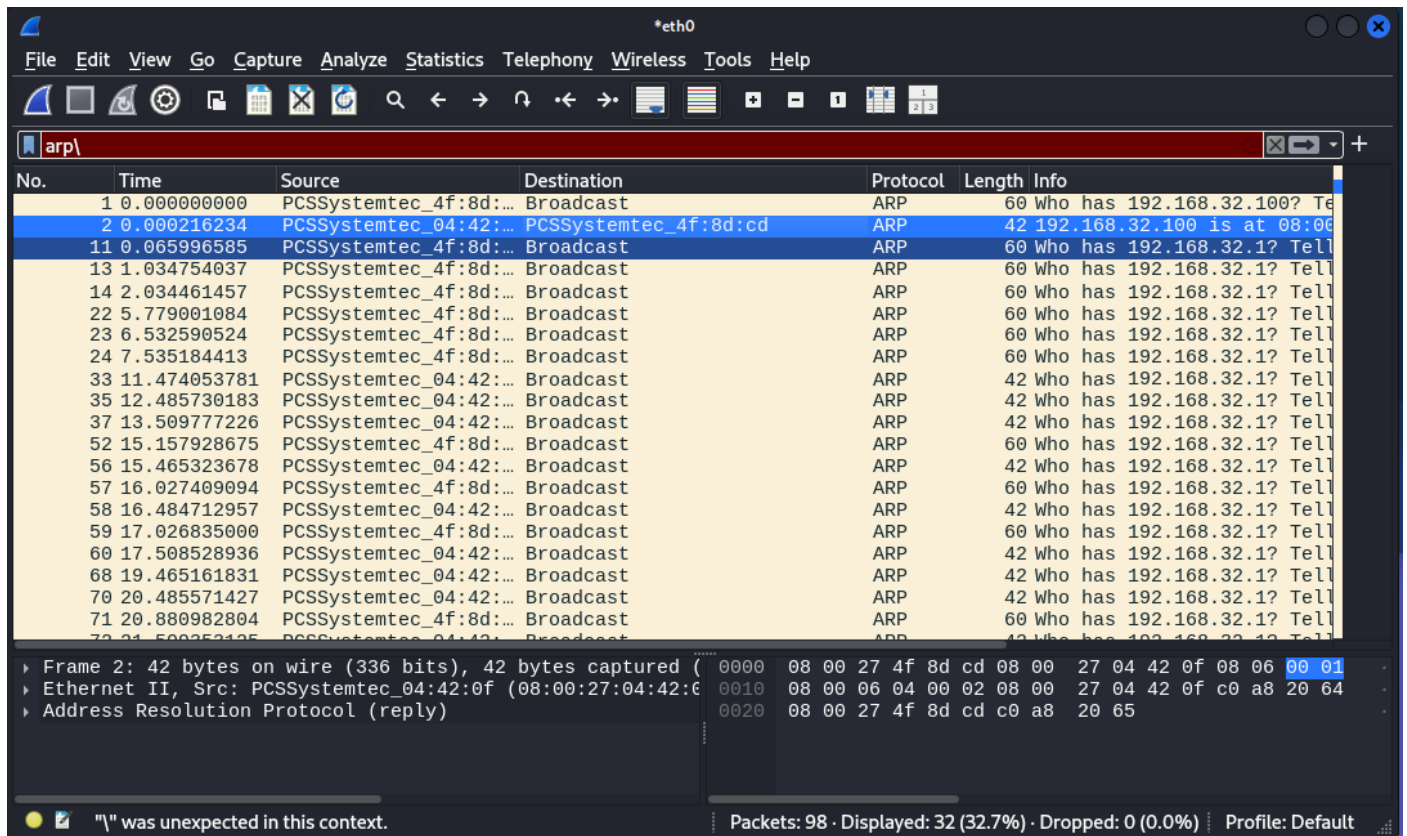
wireshark_eth0JFFX32.pcapng Packets: 98 · Dropped: 0 (0.0%) Profile: Default

All'apertura di wireshark ci verrà mostrata una schermata simile, con la lista di tutti i pacchetti intercettati, per analizzare i pacchetti che ci interessano applicheremo dei filtri nella barra di ricerca del programma, in modo da poter visualizzare in modo più veloce i dati che ci interessa analizzare.

MAC ADDRESS

Partiamo analizzando i pacchetti ARP, per visualizzarli filtriamoli scrivendo *"arp"* nella barra di ricerca

Il pacchetto ARP (Address Resolution Protocol) è utilizzato per risolvere gli indirizzi IP in indirizzi MAC all'interno di una rete. Quando un dispositivo deve inviare dati a un altro dispositivo sulla stessa rete locale, utilizza ARP per scoprire l'indirizzo MAC corrispondente all'indirizzo IP di destinazione.



The image shows a Wireshark packet capture on interface eth0. A filter 'arp' is applied. The packet list shows several ARP requests (broadcast) and one ARP reply. The packet details pane shows the structure of the selected ARP reply packet.

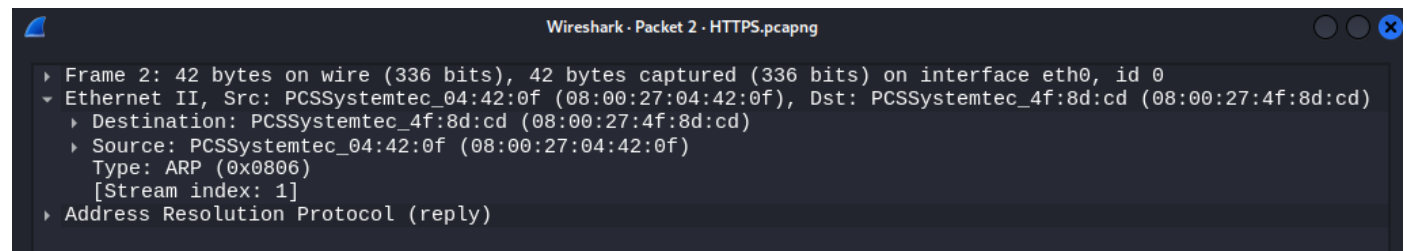
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.100? Te
2	0.000216234	PCSSystemtec_04:42:...	PCSSystemtec_4f:8d:cd	ARP	42	192.168.32.100 is at 08:00
11	0.065996585	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
13	1.034754037	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
14	2.034461457	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
22	5.779001084	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
23	6.532590524	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
24	7.535184413	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
33	11.474053781	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
35	12.485730183	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
37	13.509777226	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
52	15.157928675	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
56	15.465323678	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
57	16.027409094	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
58	16.484712957	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
59	17.026835000	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell
60	17.508528936	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
68	19.465161831	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
70	20.485571427	PCSSystemtec_04:42:...	Broadcast	ARP	42	Who has 192.168.32.1? Tell
71	20.880982804	PCSSystemtec_4f:8d:...	Broadcast	ARP	60	Who has 192.168.32.1? Tell

Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_04:42:0f (08:00:27:04:42:0f), Dst: PCSSystemtec_4f:8d:cd (08:00:27:4f:8d:cd)
Address Resolution Protocol (reply)

Vengono visualizzati una serie di risultati, la maggior parte con destinazione *Broadcast*.

I pacchetti ARP con destinazione broadcast vengono quando un dispositivo non conosce l'indirizzo MAC corrispondente a un IP, invia quindi una richiesta ARP in broadcast a tutti i dispositivi. Solo il dispositivo con l'indirizzo IP specificato risponde, fornendo il proprio MAC.

Per visualizzare invece i *MAC address* dei dispositivi in comunicazione andremo ad analizzare il pacchetto che ha come destinazione il sistema windows (PCSSystemtec_4f:8d:cd), facendo doppio click entriamo nei dettagli del pacchetto.



The image shows the packet details pane for the selected ARP reply packet. It displays the Ethernet II header, the ARP type, and the source and destination MAC addresses.

```
Wireshark - Packet 2 - HTTPS.pcapng
Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
  Ethernet II, Src: PCSSystemtec_04:42:0f (08:00:27:04:42:0f), Dst: PCSSystemtec_4f:8d:cd (08:00:27:4f:8d:cd)
    Destination: PCSSystemtec_4f:8d:cd (08:00:27:4f:8d:cd)
    Source: PCSSystemtec_04:42:0f (08:00:27:04:42:0f)
      Type: ARP (0x0806)
      [Stream index: 1]
    Address Resolution Protocol (reply)
```

Da qui possiamo visualizzare il MAC address sorgente (Kali)
e il MAC destinazione (Windows)

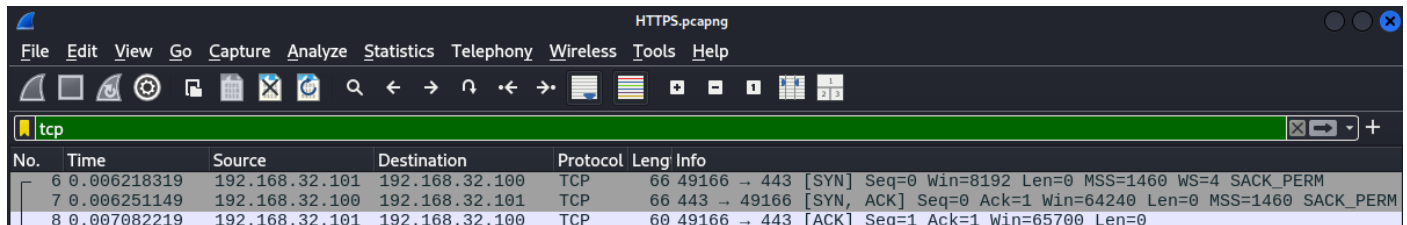
Src: 08:00:27:04:42:0f
Dst: 08:00:27:4f:8d:cd

3-WAY HANDSHAKE

Un'altra funzione interessante da analizzare è quella del *3-way handshake*.

Il *3-way handshake* è un processo di connessione TCP che stabilisce una comunicazione tra un client e un server. Consiste in tre passaggi:

- il client invia un pacchetto SYN per iniziare la connessione,
- il server risponde con un pacchetto SYN-ACK per confermare,
- infine il client invia un pacchetto ACK per completare l'handshake e stabilire la connessione



The image shows a Wireshark capture of a 3-way TCP handshake. The filter is set to 'tcp'. The packet list shows three packets: a SYN packet from 192.168.32.101 to 192.168.32.100, a SYN-ACK packet from 192.168.32.100 to 192.168.32.101, and an ACK packet from 192.168.32.101 to 192.168.32.100.

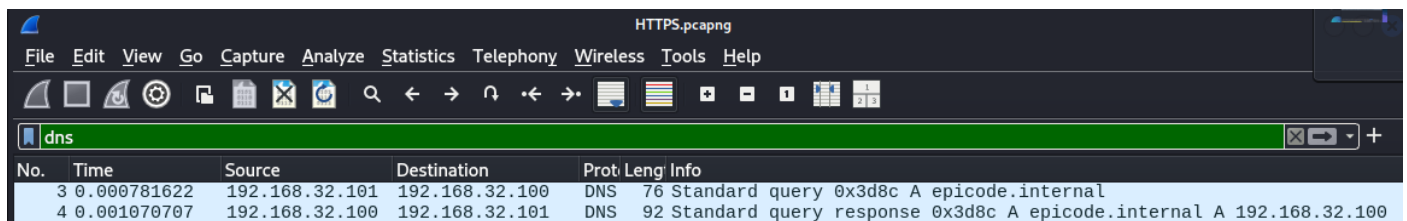
No.	Time	Source	Destination	Protocol	Length	Info
6	0.006218319	192.168.32.101	192.168.32.100	TCP	66	49166 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
7	0.006251149	192.168.32.100	192.168.32.101	TCP	66	443 → 49166 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM
8	0.007082219	192.168.32.101	192.168.32.100	TCP	60	49166 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0

Per visualizzare questo processo filtriamo i risultati in wireshark cercando *"TCP"*.

Nell'immagine possiamo visualizzare l'esecuzione del processo, e notare gli indirizzi del client e del server della nostra simulazione

DNS

Utilizziamo ora il filtro *"DNS"* per visualizzare e analizzare i pacchetti DNS



The image shows a Wireshark capture of DNS packets. The filter is set to 'dns'. The packet list shows two packets: a DNS query from 192.168.32.101 to 192.168.32.100 and a DNS response from 192.168.32.100 to 192.168.32.101.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000781622	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x3d8c A epicode.internal
4	0.001070707	192.168.32.100	192.168.32.101	DNS	92	Standard query response 0x3d8c A epicode.internal A 192.168.32.100

Come si vede dall'immagine il primo pacchetto è la richiesta del client verso il DNS per la risoluzione del dominio *epicode.internal*.

Il secondo pacchetto viaggia dal DNS al client con la risposta della risoluzione e l'IP associato al dominio.

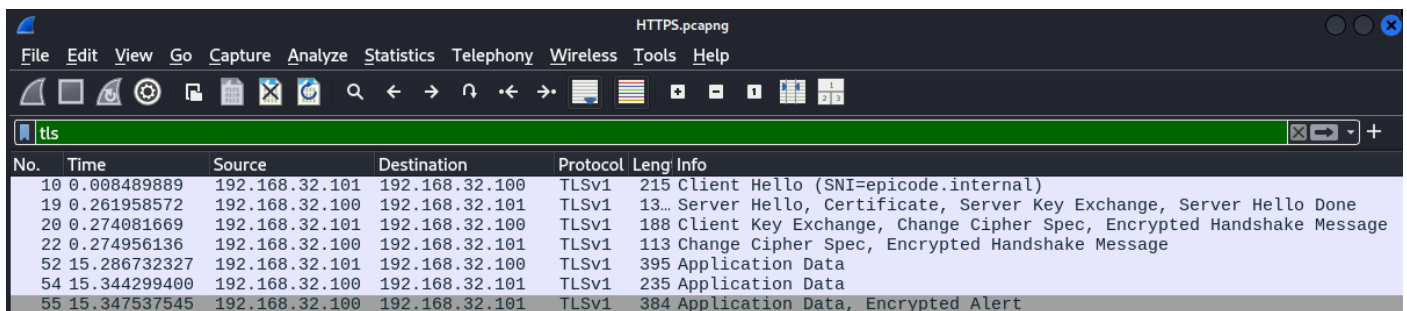
HTTPS

Per analizzare il servizio HTTPS filtriamo i risultati digitando "TLS"

TLS sta per Transport Layer Security. È un protocollo crittografico progettato per garantire la sicurezza delle comunicazioni su una rete

HTTPS utilizza TLS per crittografare i dati scambiati tra il browser (client) e il server. Questo significa che anche se i dati vengono intercettati, non possono essere letti senza la chiave di crittografia.

La terminologia HTTPS riflette il fatto che HTTP è stato esteso con la sicurezza fornita da TLS. Quindi, quando si parla di HTTPS, in realtà stai parlando di HTTP che utilizza TLS per la sicurezza.



No.	Time	Source	Destination	Protocol	Length	Info
10	0.008489889	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello (SNI=epicode.internal)
19	0.261958572	192.168.32.100	192.168.32.101	TLSv1	13...	Server Hello, Certificate, Server Key Exchange, Server Hello Done
20	0.274081669	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
22	0.274956136	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
52	15.286732327	192.168.32.101	192.168.32.100	TLSv1	395	Application Data
54	15.344299400	192.168.32.100	192.168.32.101	TLSv1	235	Application Data
55	15.347537545	192.168.32.100	192.168.32.101	TLSv1	384	Application Data, Encrypted Alert

Analizzando i pacchetti filtrati possiamo fare le seguenti osservazioni:

Connessione TLS:

I pacchetti mostrano una connessione TLS stabilita tra il client e il server

Fasi della Connessione TLS:

Il client invia un "Client Hello" per iniziare la negoziazione TLS.

In questo primo messaggio Il client propone una serie di opzioni crittografiche (cipher suites) che può supportare.

Segue lo scambio del "Server Hello", del certificato del server, dello "Server Key Exchange" e del "Server Hello Done".

Server Hello:

Il server risponde selezionando una delle cipher suite proposte dal client. Questa selezione determina gli algoritmi crittografici che verranno utilizzati per la connessione.

Server Certificate:

Il server invia il suo certificato digitale, che contiene la sua chiave pubblica, Questo certificato viene utilizzato dal client per autenticare l'identità del server.

Server Key Exchange:

Se necessario, il server invia parametri aggiuntivi per lo scambio delle chiavi.

Server Hello Done:

Questo messaggio indica che il server ha completato la fase di saluto iniziale.

Il client invia quindi il "Client Key Exchange", il "Change Cipher Spec" e il "Encrypted Handshake Message" per completare la negoziazione.

Client Key Exchange:

Il client genera una chiave di sessione e la invia al server, crittografata con la chiave pubblica del server.

Change Cipher Spec:

Questo messaggio indica che d'ora in poi tutti i dati saranno crittografati utilizzando le chiavi negoziate.

Encrypted Handshake Message:

Il client invia un messaggio crittografato per confermare la conclusione della negoziazione TLS.

Questa sequenza di messaggi completa la fase di handshake TLS, stabilendo una connessione sicura tra client e server prima che inizi lo scambio di dati applicativi.

Traffico Crittografato:

Dopo la negoziazione TLS, i pacchetti mostrano "Application Data" e "Encrypted Alert", indicando che il traffico è crittografato.

Questo significa che il contenuto dei dati trasmessi non è leggibile senza decrittare la connessione TLS.

Analisi del Traffico:

Per analizzare il contenuto di questa connessione HTTPS, sarebbe necessario disporre della chiave di crittografia TLS, senza la chiave, il contenuto dei dati crittografati non può essere visualizzato

ANALISI DELLE DIFFERENZE TRA HTTP E HTTPS

Andremo adesso ad inviare una richiesta al server utilizzando però il protocollo HTTP.

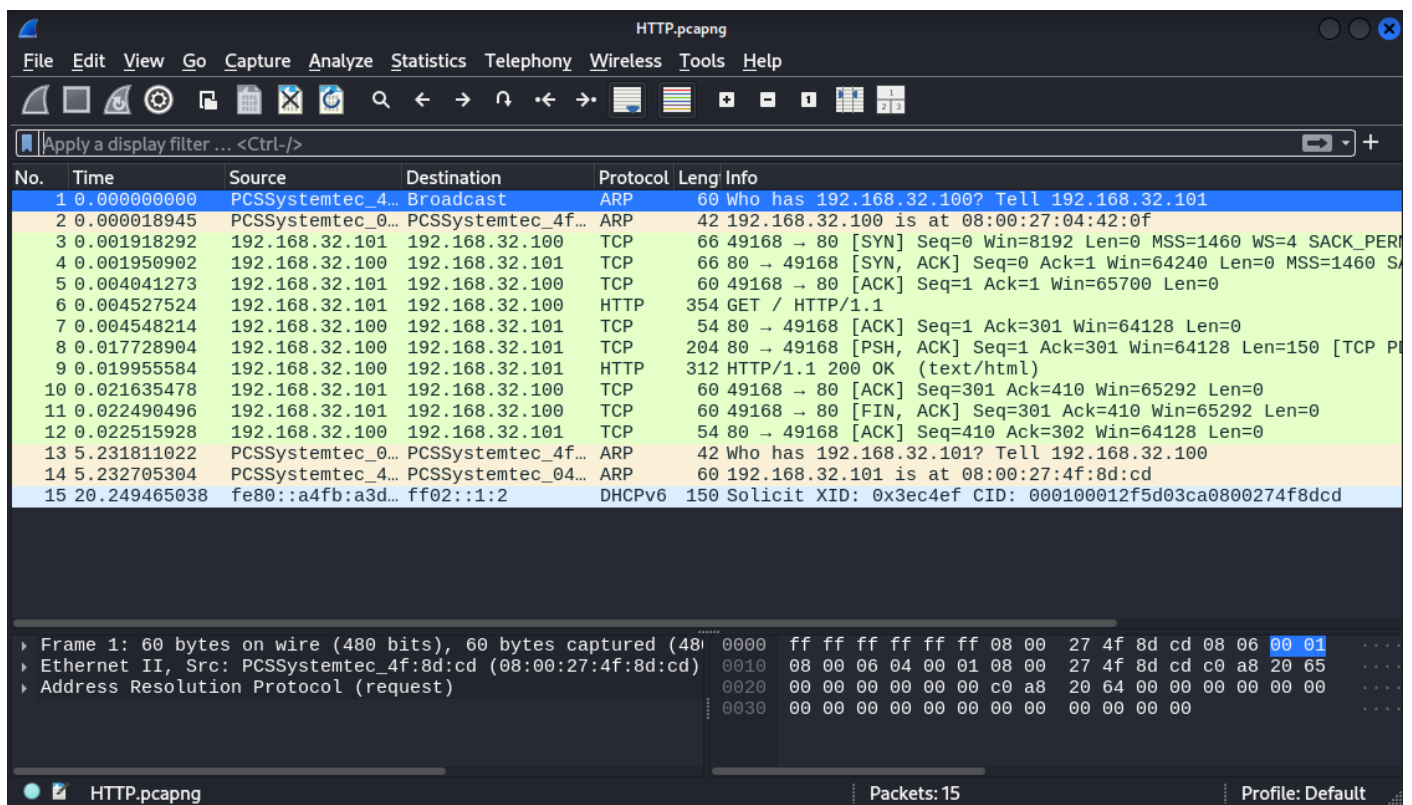
Andremo ad intercettare ancora la comunicazione con wireshark e analizzeremo le principali differenze tra i due protocolli utilizzati.

Apriamo wireshark e impostiamo ancora la rete eth0, e lasciamolo in attesa.

Torniamo quindi al browser del client window e digitiamo sulla barra di ricerca:

<http://epicode.internal>

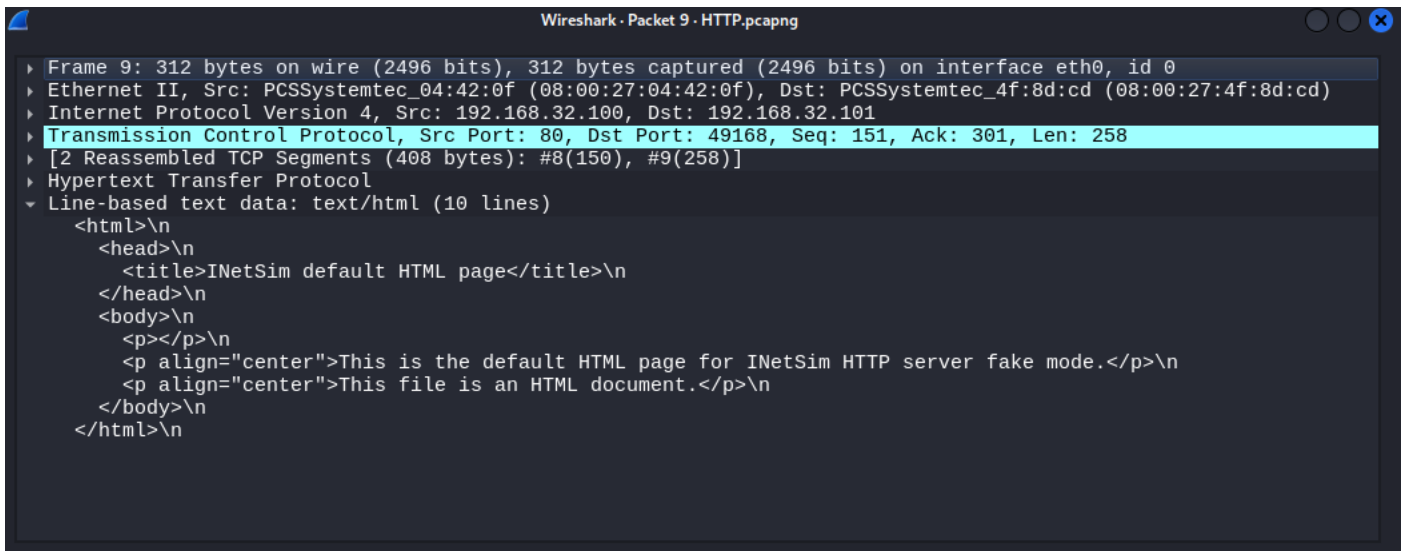
La pagina che ci si presenterà sarà la stessa della connessione precedente, dato che i file di sample di inetsim sono sempre gli stessi, per analizzare le differenze torniamo su wireshark e controlliamo i pacchetti che sono stati catturati in questa sessione.



A prima occhiata possiamo subito notare come, anche in questo caso, gli indirizzi MAC siano stati risolti con il protocollo ARP, e che anche questa connessione sia stata stabilita utilizzando il *3-way handshake*, di cui abbiamo parlato in precedenza.

Le prime differenze si notano nell'assenza della negoziazione TLS ,in quanto questo servizio non è presente nel protocollo HTTP

Entrando in dettaglio nell'analisi dei pacchetti HTTP, notiamo che cliccando 2 volte nel pacchetto della riga 9, dove il server invia i dati del sito richiesto al client, abbiamo la possibilità di intercettare tutti i dati che il server ha trasferito al client.

A screenshot of the Wireshark network protocol analyzer interface. The title bar reads 'Wireshark - Packet 9 - HTTP.pcapng'. The packet list on the left shows several layers: Frame 9, Ethernet II, Internet Protocol Version 4, Transmission Control Protocol (highlighted in blue), [2 Reassembled TCP Segments], Hypertext Transfer Protocol, and Line-based text data. The packet details pane on the right shows the raw HTML content of the captured packet, which is a default page from INetSim. The HTML code includes a title 'INetSim default HTML page' and two paragraphs of text, one centered.

```
Wireshark - Packet 9 - HTTP.pcapng
> Frame 9: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_04:42:0f (08:00:27:04:42:0f), Dst: PCSSystemtec_4f:8d:cd (08:00:27:4f:8d:cd)
> Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
> Transmission Control Protocol, Src Port: 80, Dst Port: 49168, Seq: 151, Ack: 301, Len: 258
> [2 Reassembled TCP Segments (408 bytes): #8(150), #9(258)]
> Hypertext Transfer Protocol
> Line-based text data: text/html (10 lines)
  <html>\n
    <head>\n
      <title>INetSim default HTML page</title>\n
    </head>\n
    <body>\n
      <p></p>\n
      <p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>\n
      <p align="center">This file is an HTML document.</p>\n
    </body>\n
  </html>\n
```

Nell'immagine, per esempio, abbiamo la possibilità di leggere tutto il codice HTML del sito senza nessuna restrizione.

Questo tipo di intrusione non è possibile con il protocollo HTTPS, in quanto i dati trasmessi tra client e server sono criptati con le chiavi di sicurezza e leggibili sono da chi le detiene.

CONCLUSIONI

Riassumendo, la principale differenza tra HTTP e HTTPS è che HTTPS fornisce un livello aggiuntivo di sicurezza attraverso la crittografia, l'autenticazione del server e l'integrità dei dati, mentre HTTP non offre alcuna di queste protezioni, rendendo l'utilizzo di questo protocollo vulnerabile da attacchi esterni.