

W4 - D5 - PRATICA - 18 07 205 - Giuseppe Gigliotti- PROGETTO M1 - INTRODUZIONE ALL'HACKING

Traccia:

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows) richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100 (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.

Per semplicità usiamo direttamente gli indirizzi già utilizzati durante le configurazioni iniziali. Su Windows l'indirizzo sarà 192.168.50.102, mentre per kali l'indirizzo sarà 192.168.50.100

- Verifichiamo che la nostra macchina kali sia connessa all'indirizzo IP richiesto

```

(kali㉿kali)-[~]
$ ip a
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:d1:f8:5d brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.100/24 brd 192.168.50.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::416:9a41:8519:128/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(kali㉿kali)-[~]
$ sudo nano /etc/inetsim/inetsim.conf

```

Fatto ciò utilizzando in comando apposito ed utilizzando un editor di testo ,
passiamo sul file di configurazione di inetsim

```

GNU nano 8.4 /etc/inetsim/inetsim.conf *
# INetSim configuration file
#
#####

#####
# Main configuration
#####
#
# start_service
#
# The services to start
#
# Syntax: start_service <service name>
#
# Default: none
#
# Available service names are:
# 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

```

Nel nostro primo caso avremo bisogno di abilitare i servizi https e del dns.

Questa configurazione ci dice che, inetsim esporrà i servizi sull'IP che diremo noi (o precisamente su tutto) , in questo caso l'indirizzo visualizzato tramite il comando ip a 192.168.50.100.

```
GNU nano 8.4 /etc/inetsim/inetsim.conf
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 0.0.0.0
#####
```

Configuriamo il dns

- Modifichiamo l'IP 10.10.10.1 con il nostro IP da utilizzare

```
GNU nano 8.4 /etc/inetsim/inetsim.conf
#####
# Service DNS
#####

#####
# dns_bind_port
#
# Port number to bind DNS service to
#
# Syntax: dns_bind_port <port number>
#
# Default: 53
#
#dns_bind_port 53

#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 192.168.50.100
```

- Scendiamo in basso verso i record dns statici e aggiungiamo il nostro come richiesto nella traccia (epicode.internal)

```
#####  
# dns_static  
#  
# Static mappings for DNS  
#  
# Syntax: dns_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
#dns_static www.foo.com 10.10.10.10  
#dns_static ns1.foo.com 10.70.50.30  
#dns_static ftp.bar.net 10.10.20.30  
dns_static epicode.internal 192.168.50.100  
#####
```

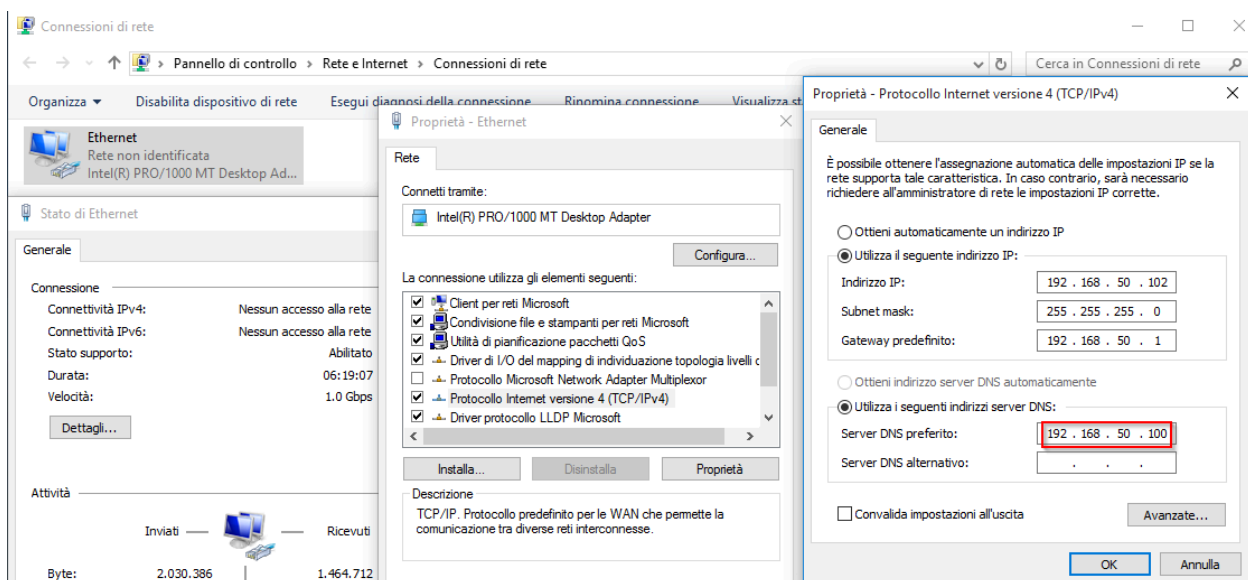
Per quanto riguarda sia il servizio https che http non facciamo modifiche.

Salvando e chiudendo il file configurazione, avviamo inetsim tramite l'apposito comando

sudo inetsim

```
(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 196980) ===
Session ID:      196980
Listening on:    0.0.0.0
Real Date/Time:  2025-07-24 15:20:23
Fake Date/Time:  2025-07-24 15:20:23 (Delta: 0 seconds)
Forking services ...
  * dns_53_tcp_udp - started (PID 196984)
  * https_443_tcp - started (PID 196985)
done.
Simulation running.
```

Lasciamo la simulazione attiva sul terminale di kali e passiamo alla macchina windows 10 dove impostiamo l'IP del nostro record dns statico aggiunto dalle opzioni della scheda di rete



- Convalidiamo all'uscita e un'ulteriore verifica se tutto procede bene, apriamo un prompt dei comandi e utilizziamo il comando nslookup con il nome del record dns aggiunto epicode.internal.

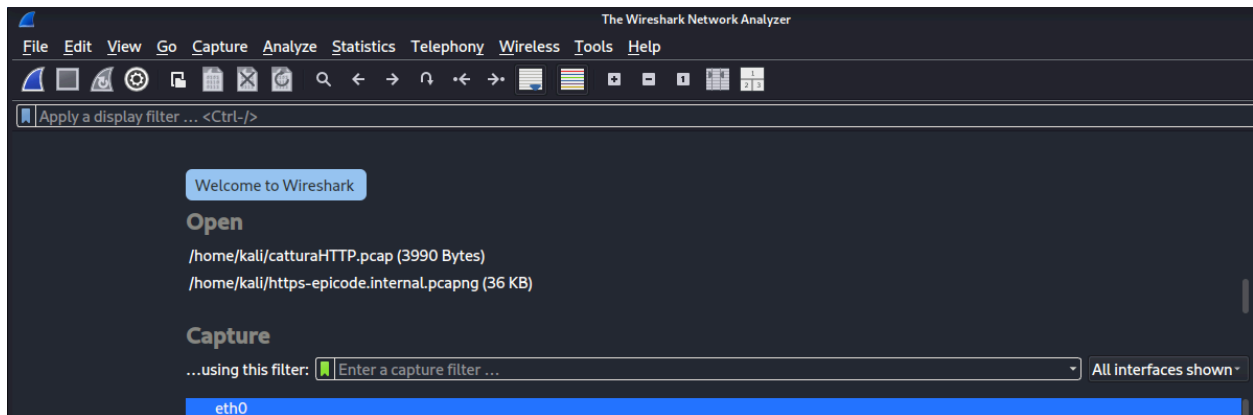
```
C:\Users\user>nslookup epicode.internal
Server: epicode.internal
Address: 192.168.50.100

Nome: epicode.internal
Address: 192.168.50.100
```

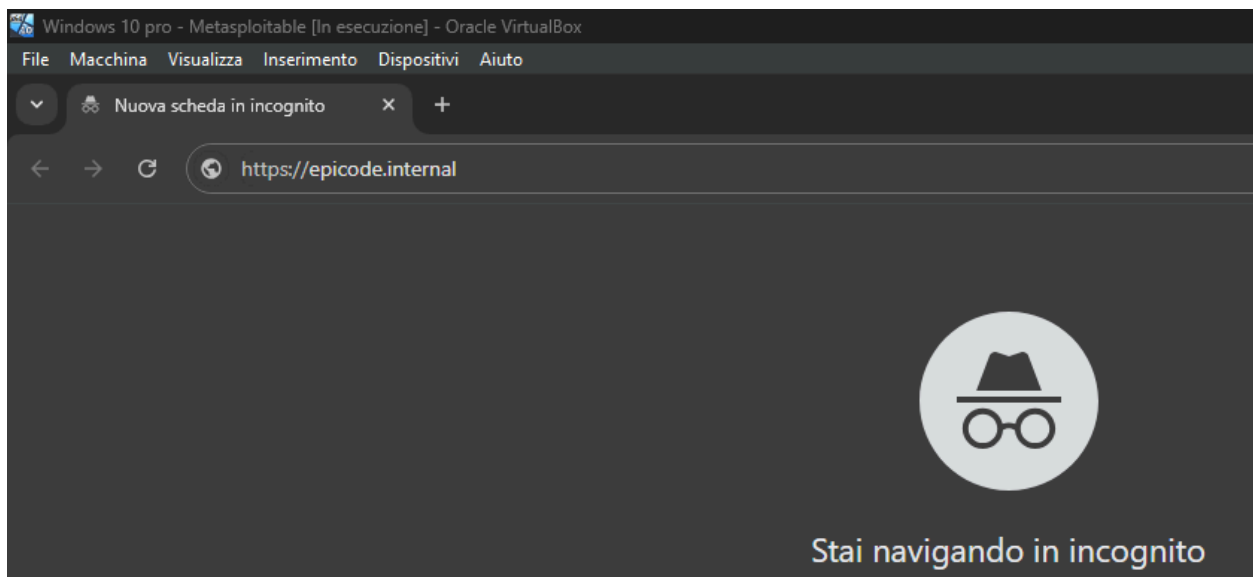
Possiamo procedere agli altri punti

1. Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS.

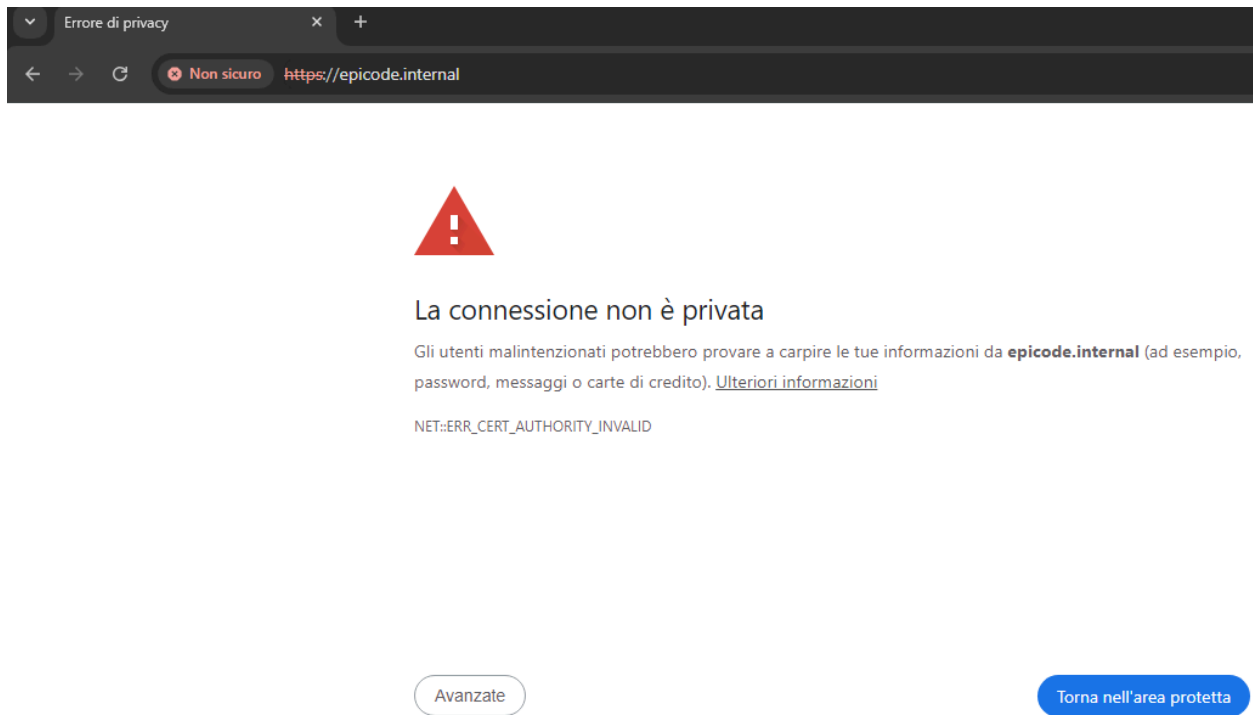
- Avviamo wireshark sulla macchina kali sull'interfaccia eth0 e torniamo sulla macchina windows.



- Apriamo un web browser e sulla barra di ricerca scriviamo il seguente url:
`https://epicode.internal`



- Cliccando invio la nostra richiesta dovrebbe farci visualizzare la pagina di default di inetsim, ma ricordando che siamo su https e il record dns aggiunto non ha un certificato di sicurezza valido la pagina visualizzata sarà:



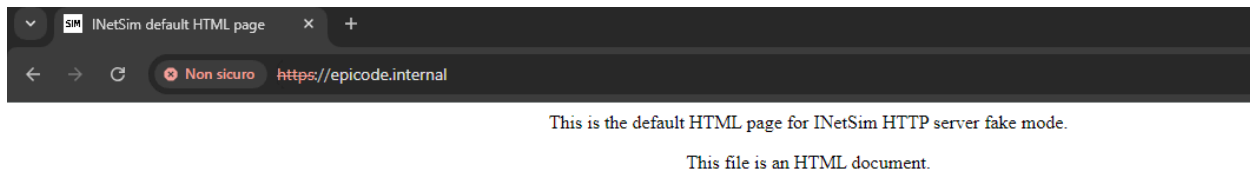
- Clicchiamo sul pulsante in basso avanzate ed mandiamo di nuovo la richiesta cliccando

Procedi su epicode.internal (non sicuro)

Questo server non è riuscito a dimostrare che si tratta di **epicode.internal**; il relativo certificato di sicurezza non è considerato attendibile dal sistema operativo del computer. Il problema potrebbe essere dovuto a un'errata configurazione o a un malintenzionato che intercetta la connessione.

[Procedi su epicode.internal \(non sicuro\)](https://epicode.internal)

- Non appena la nostra richiesta viene inviata visualizzeremo la pagina sottostante



- Torniamo su wireshark e vediamo cosa abbiamo intercettato

No.	Time	Source	Destination	Protocol	Length	Info
49	4.365364	192.168.50.102	192.168.50.100	DNS	80	Standard query 0x1e50 HTTPS beacons.gcp.gvt2.com
50	4.370100	192.168.50.100	192.168.50.102	DNS	96	Standard query response 0xba0c A beacons.gcp.gvt2.com A 192.168.50.102
51	4.373669	192.168.50.100	192.168.50.102	DNS	80	Standard query response 0x1e50 Not implemented HTTPS beacons.gcp.gvt2.com
52	4.374896	192.168.50.102	192.168.50.100	DNS	80	Standard query 0xc983 HTTPS beacons.gcp.gvt2.com
53	4.379575	192.168.50.100	192.168.50.102	DNS	80	Standard query response 0xc983 Not implemented HTTPS beacons.gcp.gvt2.com
54	4.380249	192.168.50.102	192.168.50.100	ICMP	108	Destination unreachable (Port unreachable)
55	4.380484	192.168.50.102	192.168.50.100	TCP	66	53977 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
56	4.380497	192.168.50.100	192.168.50.102	TCP	66	443 → 53977 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1
57	4.381164	192.168.50.102	192.168.50.100	TCP	60	53977 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
58	4.382085	192.168.50.102	192.168.50.100	TLSv1.3	1784	Client Hello (SNI=beacons.gcp.gvt2.com)
59	4.382093	192.168.50.100	192.168.50.102	TCP	54	443 → 53977 [ACK] Seq=1 Ack=1731 Win=62592 Len=0
60	4.388239	192.168.50.102	192.168.50.100	TCP	66	53978 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
61	4.388267	192.168.50.100	192.168.50.102	TCP	66	443 → 53978 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1
62	4.388724	192.168.50.102	192.168.50.100	TCP	60	53978 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
63	4.391548	192.168.50.102	192.168.50.100	TLSv1.3	1844	Client Hello (SNI=epicode.internal)
64	4.391561	192.168.50.100	192.168.50.102	TCP	54	443 → 53978 [ACK] Seq=1 Ack=1791 Win=62464 Len=0
65	4.394551	192.168.50.100	192.168.50.102	TLSv1.3	1497	Server Hello, Change Cipher Spec, Application Data, Application Data
66	4.401895	192.168.50.100	192.168.50.102	TLSv1.3	1497	Server Hello, Change Cipher Spec, Application Data, Application Data
67	4.403927	192.168.50.102	192.168.50.100	TLSv1.3	84	Change Cipher Spec, Application Data
68	4.404675	192.168.50.102	192.168.50.100	TCP	60	53978 → 443 [FIN, ACK] Seq=1821 Ack=1444 Win=64256 Len=0
69	4.405578	192.168.50.102	192.168.50.100	TLSv1.3	84	Change Cipher Spec, Application Data
70	4.405863	192.168.50.102	192.168.50.100	TCP	60	53977 → 443 [FIN, ACK] Seq=1761 Ack=1444 Win=64256 Len=0
71	4.408394	192.168.50.100	192.168.50.102	TCP	54	443 → 53978 [FIN, ACK] Seq=1444 Ack=1822 Win=62464 Len=0
72	4.409119	192.168.50.102	192.168.50.100	TCP	60	53978 → 443 [ACK] Seq=1822 Ack=1445 Win=64256 Len=0
73	4.411538	192.168.50.100	192.168.50.102	TCP	54	443 → 53977 [FIN, ACK] Seq=1444 Ack=1762 Win=62592 Len=0
74	4.412096	192.168.50.102	192.168.50.100	TCP	60	53977 → 443 [ACK] Seq=1762 Ack=1445 Win=64256 Len=0

- Mettiamo in evidenza i MAC address di sorgente e destinazione

```

63 4.391548 192.168.50.102 192.168.50.100 TLSv1.3 1844 Client Hello (SNI=epicode.internal)
64 4.391561 192.168.50.100 192.168.50.102 TCP 54 443 → 53978 [ACK] Seq=1 Ack=1791 Win=62464 Len=0
65 4.394551 192.168.50.100 192.168.50.102 TLSv1.3 1497 Server Hello, Change Cipher Spec, Application Data, Application Data
66 4.401895 192.168.50.100 192.168.50.102 TLSv1.3 1497 Server Hello, Change Cipher Spec, Application Data, Application Data

Destination: PCSSystemtec_d1:f8:5d (08:00:27:d1:f8:5d)
.....
0000 08 00 27 d1 f8 5d 08 00 27 67 de 22 08 00
0010 07 26 72 c1 40 00 80 06 9a f5 c0 a8 32 60
0020 32 64 d2 da 01 bb f0 46 45 2c 34 13 00 c0
0030 01 00 ed 33 00 00 16 03 01 06 f9 01 00 00
0040 03 75 f2 f4 57 bd 7d 03 dd d0 7c 2d e8 00
0050 fd 7a 4f 48 93 f1 77 c7 2b 2e 0f 57 46 90
0060 d5 20 c8 90 18 5f 3f 54 d2 1a ad 89 48 20
0070 8f 66 f9 85 3c 59 c4 d6 5a 5a 44 af 0c 10

Source: PCSSystemtec_07:de:22 (08:00:27:67:de:22)
.....
0000 08 00 27 d1 f8 5d 08 00 27 67 de 22 08 00
0010 07 26 72 c1 40 00 80 06 9a f5 c0 a8 32 60
0020 32 64 d2 da 01 bb f0 46 45 2c 34 13 00 c0
0030 01 00 ed 33 00 00 16 03 01 06 f9 01 00 00
0040 03 75 f2 f4 57 bd 7d 03 dd d0 7c 2d e8 00
0050 fd 7a 4f 48 93 f1 77 c7 2b 2e 0f 57 46 90
0060 d5 20 c8 90 18 5f 3f 54 d2 1a ad 89 48 20
0070 8f 66 f9 85 3c 59 c4 d6 5a 5a 44 af 0c 10

Type: IPv4 (0x0000)
[Stream index: 0]

```

- Cerchiamo di controllare il contenuto, ma come possiamo vedere è crittografato

```

69 4.405578 192.168.50.102 192.168.50.100 TLSv1.3 84 Change Cipher Spec, Application Data
70 4.405863 192.168.50.102 192.168.50.100 TCP 60 53977 → 443 [FIN, ACK] Seq=1761 Ack=1444 Win=64256 Len=0
71 4.408394 192.168.50.100 192.168.50.102 TCP 54 443 → 53978 [FIN, ACK] Seq=1444 Ack=1822 Win=62464 Len=0
72 4.409119 192.168.50.102 192.168.50.100 TCP 60 53978 → 443 [ACK] Seq=1822 Ack=1445 Win=64256 Len=0
73 4.411538 192.168.50.100 192.168.50.102 TCP 54 443 → 53977 [FIN, ACK] Seq=1444 Ack=1762 Win=62592 Len=0
74 4.412096 192.168.50.102 192.168.50.100 TCP 60 53977 → 443 [ACK] Seq=1762 Ack=1445 Win=64256 Len=0
75 4.418571 192.168.50.102 192.168.50.100 TCP 66 53979 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
76 4.418587 192.168.50.100 192.168.50.102 TCP 54 80 → 53979 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Transmission Control Protocol, Src Port: 53977, Dst Port: 443, Seq: 1731, Ack: 1444, Len: 30
Transport Layer Security
  Version: TLS 1.2 (0x0303)
  Content Type: Change Cipher Spec (20)
  Length: 1
  Change Cipher Spec Message
  TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
  Opaque Type: Application Data (23)
  Version: TLS 1.2 (0x0303)
  Length: 19
  Encrypted Application Data: 7fe0247ba843f277a494fc410aff6898b38058
  [Application Data Protocol: Hypertext Transfer Protocol]

```

2. Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico.

- Torniamo sul file di configurazione di inetsim e commentiamo il servizio https e mettiamo in esecuzione il servizio http.

```
GNU nano 8.4 /etc/inetsim/inetsim.conf

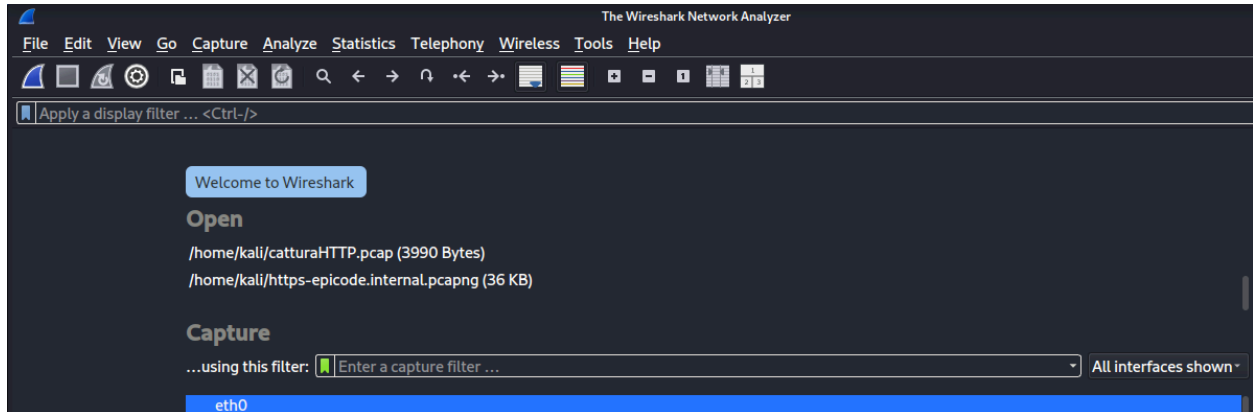
#####
# Main configuration
#####

#####
# start_service
#
# The services to start
#
# Syntax: start_service <service name>
#
# Default: none
#
# Available service names are:
# 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
```

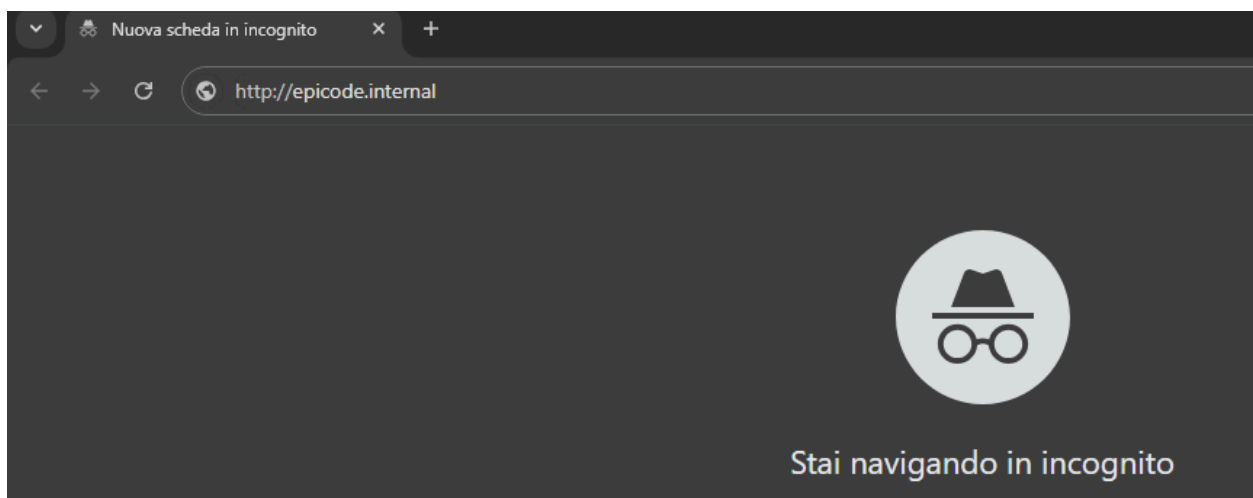
- Dopodiché mettiamo in esecuzione inetsim con l'apposito comando usato precedentemente

```
(kali㉿kali)-[~] /home/kali/curlHTTPS.pcap (11 KB)
$ 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 253228) ==
Session ID: 253228
Listening on: 0.0.0.0
Real Date/Time: 2025-07-24 17:17:17
Fake Date/Time: 2025-07-24 17:17:17 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 253232)
* http_80_tcp - started (PID 253233)
done.
Simulation running.
```

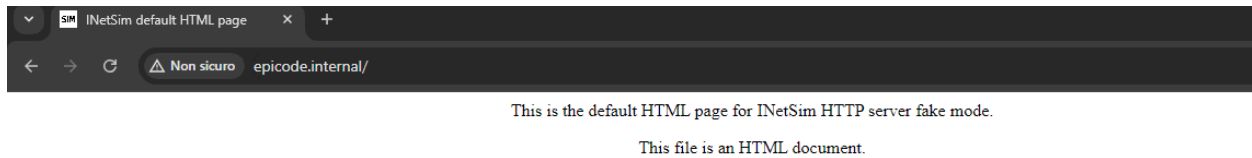
- Avviamo wireshark sulla macchina kali sull'interfaccia eth0 e torniamo sulla macchina windows.



- Apriamo un web browser e sulla barra di ricerca scriviamo il seguente url:
<http://epicode.internal>



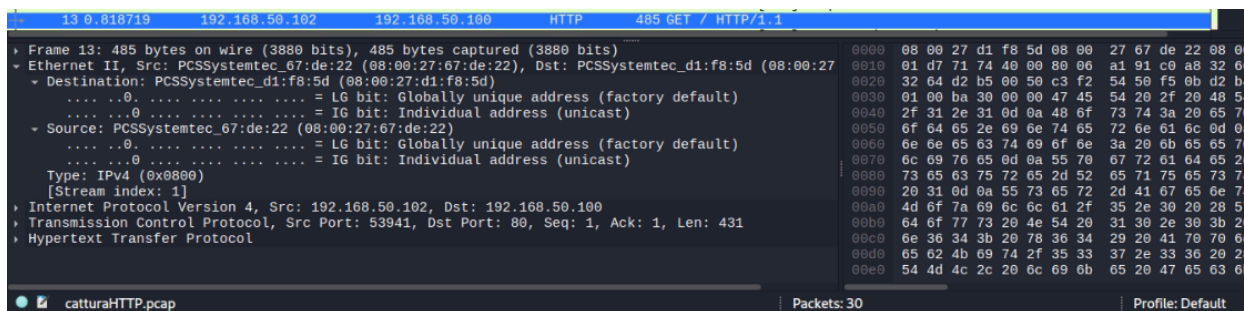
- Cliccando invio la nostra richiesta dovrebbe farci visualizzare la pagina di default di inetsim



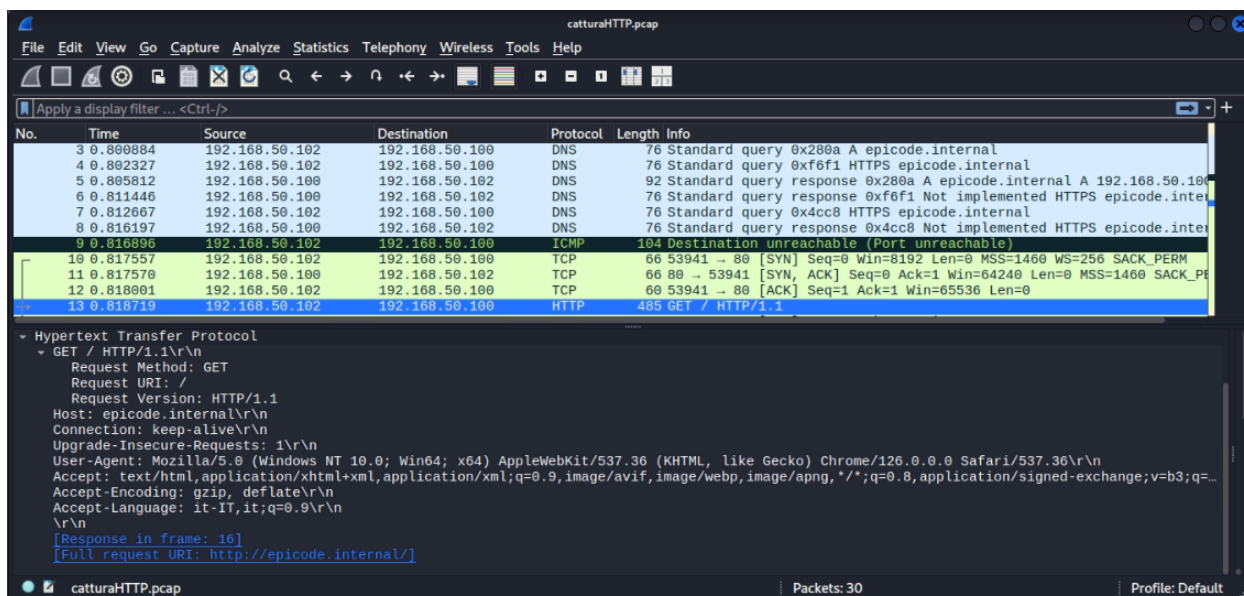
- Torniamo su wireshark e vediamo cosa abbiamo intercettato

No.	Time	Source	Destination	Protocol	Length	Info
3	0.800884	192.168.50.102	192.168.50.100	DNS	76	Standard query 0x280a A epicode.internal
4	0.802327	192.168.50.102	192.168.50.100	DNS	76	Standard query 0xf6f1 HTTPS epicode.internal
5	0.805812	192.168.50.100	192.168.50.102	DNS	92	Standard query response 0x280a A epicode.internal A 192.168.50.100
6	0.811446	192.168.50.100	192.168.50.102	DNS	76	Standard query response 0xf6f1 Not implemented HTTPS epicode.inte
7	0.812667	192.168.50.102	192.168.50.100	DNS	76	Standard query 0x4cc8 HTTPS epicode.internal
8	0.816197	192.168.50.100	192.168.50.102	DNS	76	Standard query response 0x4cc8 Not implemented HTTPS epicode.inte
9	0.816896	192.168.50.102	192.168.50.100	ICMP	104	Destination unreachable (Port unreachable)
10	0.817557	192.168.50.102	192.168.50.100	TCP	60	53941 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
11	0.817570	192.168.50.100	192.168.50.102	TCP	60	80 → 53941 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_P
12	0.818001	192.168.50.102	192.168.50.100	TCP	60	53941 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
13	0.818719	192.168.50.102	192.168.50.100	HTTP	485	GET / HTTP/1.1
14	0.818730	192.168.50.100	192.168.50.102	TCP	54	80 → 53941 [ACK] Seq=1 Ack=432 Win=64128 Len=0
15	0.829863	192.168.50.100	192.168.50.102	TCP	204	80 → 53941 [PSH, ACK] Seq=1 Ack=432 Win=64128 Len=150 [TCP PDU re
16	0.831444	192.168.50.100	192.168.50.102	HTTP	312	HTTP/1.1 200 OK (text/html)
17	0.831882	192.168.50.102	192.168.50.100	TCP	60	53941 → 80 [ACK] Seq=432 Ack=410 Win=65280 Len=0
18	0.834911	192.168.50.102	192.168.50.100	TCP	60	53941 → 80 [FIN, ACK] Seq=432 Ack=410 Win=65280 Len=0
19	0.834938	192.168.50.100	192.168.50.102	TCP	54	80 → 53941 [ACK] Seq=410 Ack=433 Win=64128 Len=0
20	0.884226	192.168.50.102	192.168.50.100	TCP	60	53942 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
21	0.884245	192.168.50.100	192.168.50.102	TCP	60	80 → 53942 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_P
22	0.884810	192.168.50.102	192.168.50.100	TCP	60	53942 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
23	0.885482	192.168.50.102	192.168.50.100	HTTP	439	GET /favicon.ico HTTP/1.1
24	0.885500	192.168.50.100	192.168.50.102	TCP	54	80 → 53942 [ACK] Seq=1 Ack=377 Win=64128 Len=0
25	0.897598	192.168.50.100	192.168.50.102	TCP	207	80 → 53942 [PSH, ACK] Seq=1 Ack=377 Win=64128 Len=153 [TCP PDU re
26	0.899431	192.168.50.100	192.168.50.102	HTTP	252	HTTP/1.1 200 OK (image/x-icon)
27	0.900080	192.168.50.102	192.168.50.100	TCP	60	53942 → 80 [ACK] Seq=377 Ack=353 Win=65280 Len=0
28	0.900298	192.168.50.102	192.168.50.100	TCP	60	53942 → 80 [FIN, ACK] Seq=377 Ack=353 Win=65280 Len=0

- Mettiamo in evidenza i MAC address



- Controlliamo il contenuto della richiesta



3. Evidenzia le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS.

Come possiamo ben vedere rispetto a http, https prevede una comunicazione tramite TLS, quindi cifrando il contenuto, rispetto che lasciarlo in chiaro.

4. Spiegare, motivandole, le principali differenze se presenti

La differenza maggiore è sullo scambio tramite i 3-way handshake, mentre per l'http vengono svolti solo 3 scambi (SYN, SYN ACK e ACK), inviando subito dopo il contenuto, per l'https utilizzando TLS (è una negoziazione di sicurezza che avviene prima di trasmettere qualsiasi dato. Il client e il server si accordano su quale algoritmo di crittografia usare, come scambiare le chiavi segrete, come verificare l'identità del server) dopo lo scambio di chiavi il contenuto della richiesta sarà cifrato

N.B. Durante lo svolgimento dell'esercizio abbiamo riscontrato dei problemi con inetsim, risolti tramite la sequenza di questi quattro comandi

```
(kali㉿kali)-[~]  
$ sudo apt upgrade inetsim  
inetsim is already the newest version (1.3.2+dfsg.1-1).  
The following packages were automatically installed and are no longer required:  
  python3-packaging-whl python3-pyinstaller-hooks-contrib python3-wheel-whl  
Use 'sudo apt autoremove' to remove them.  
  
Summary:  
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```



```

(kali@kali)-[~]
$ sudo cpan -f -u Net::DNS
Loading internal logger. Log::Log4perl recommended for better logging
Reading '/root/.cpan/Metadata'
Warning: Found only 0 objects in /root/.cpan/Metadata
Fetching with HTTP::Tiny:
https://cpan.org/authors/01mailrc.txt.gz
Reading '/root/.cpan/sources/authors/01mailrc.txt.gz'
.....DONE
Fetching with HTTP::Tiny:
https://cpan.org/modules/02packages.details.txt.gz
Reading '/root/.cpan/sources/modules/02packages.details.txt.gz'
Database was generated on Thu, 24 Jul 2025 06:52:43 GMT
.....
New CPAN.pm version (v2.38) available.
[Currently running version is v2.36]
You might want to try
  install CPAN
  reload cpan
to both upgrade CPAN.pm and run the new version without leaving
the current session.
.....DONE

```

```

(kali@kali)-[~]
$ sudo apt install cpanminus
The following packages were automatically installed and are no longer required:
  python3-packaging-whl python3-pyinstaller-hooks-contrib python3-wheel-whl
Use 'sudo apt autoremove' to remove them.

Installing:
  cpanminus

```

```

(kali@kali)-[~]
$ sudo cpanm --notest Net::DNS@1.22
→ Working on Net::DNS
Fetching http://backpan.perl.org/authors/id/N/NL/NLNETLABS/Net-DNS-1.22.tar.gz ... OK
Configuring Net-DNS-1.22 ... OK
Building Net-DNS-1.22 ... OK
Successfully installed Net-DNS-1.22 (downgraded from 1.50)
1 distribution installed

```