

w4d4 pratica

Giacomo di Giacinto

Nell'esercizio di oggi metteremo insieme le competenze acquisite finora.
Lo studente verrà valutato sulla base della risoluzione al problema seguente.

Requisiti e servizi:

- Kali Linux □ IP 192.168.32.100
- Windows 7 □ IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

Traccia:

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows 7) richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100 (Kali).

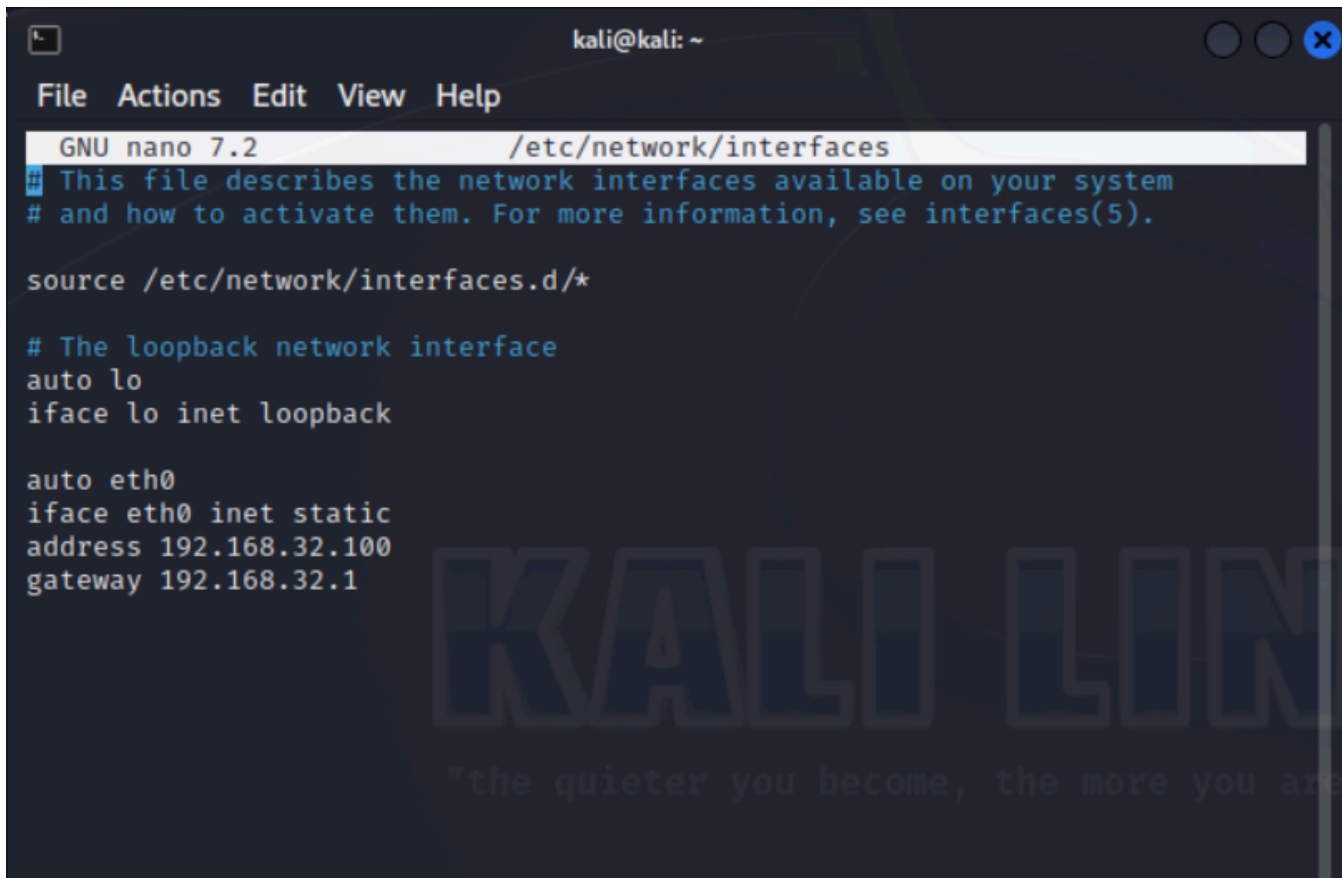
Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS.

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.

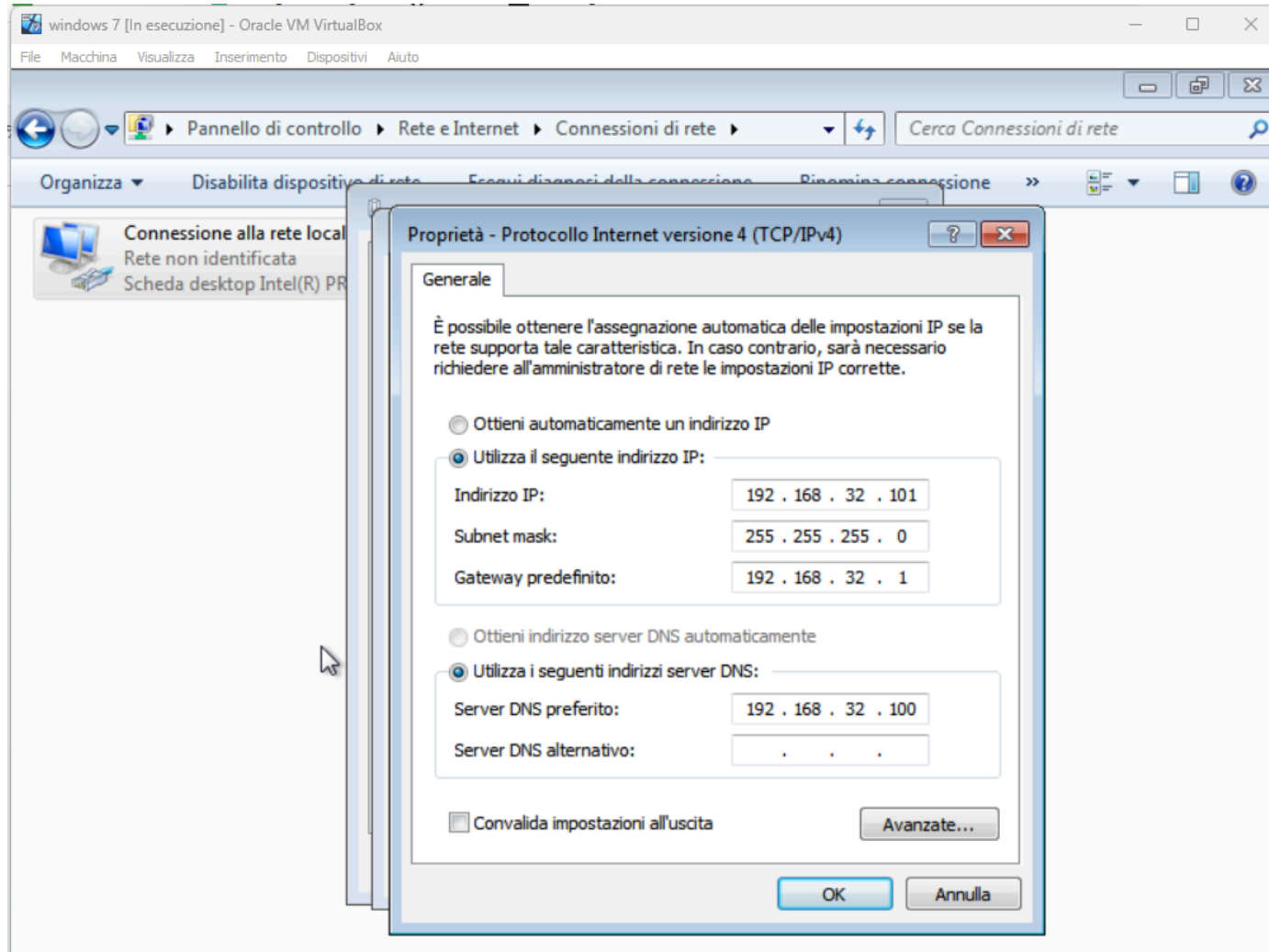
Fase 1

- Configurazione IP statici di client e server

in questa fase andremo a configurare gli indirizzi IP di kali linux e windows 7 manualmente come da istruzioni.



```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /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  
gateway 192.168.32.1  
  
KALI LIN  
"the quieter you become, the more you are"
```

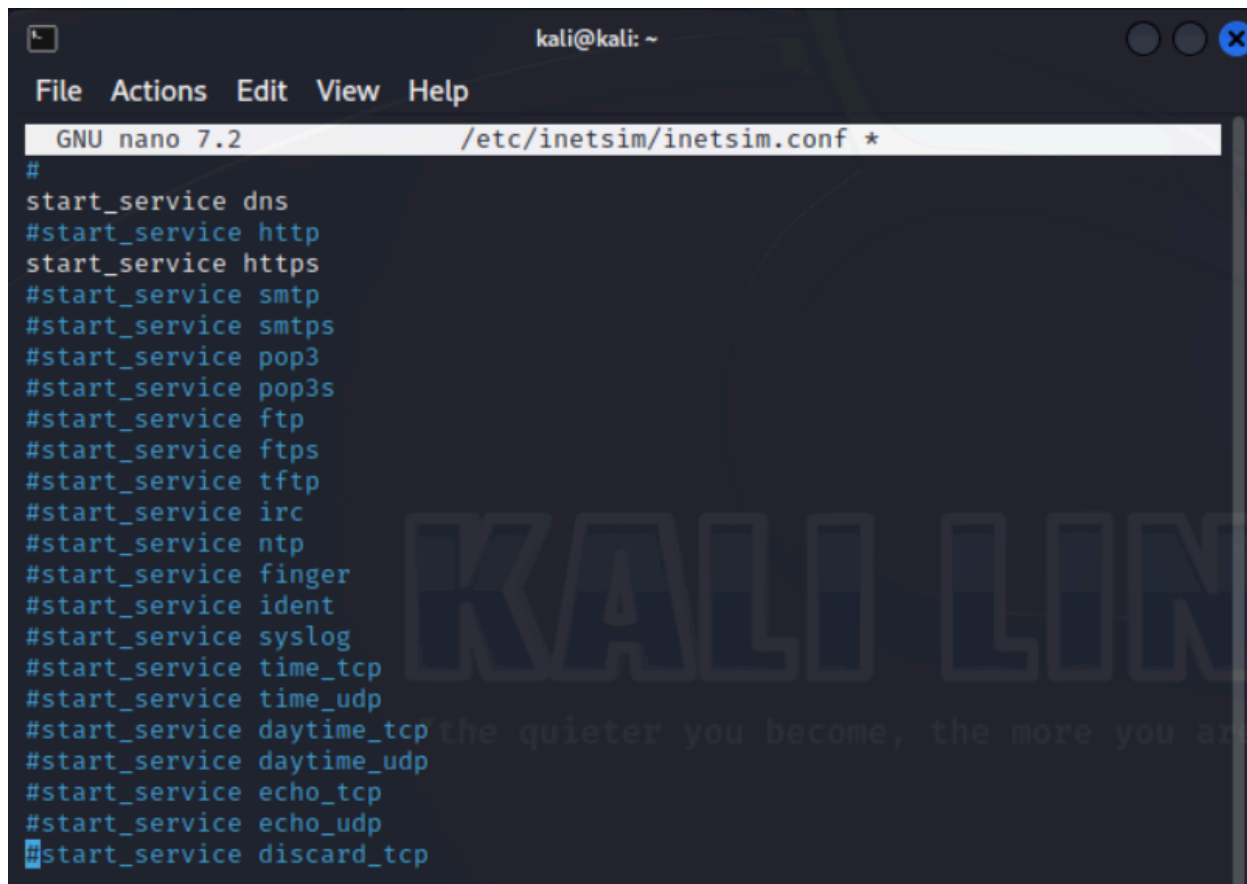


Fase 2

- Configurazione servizi HTTPS e DNS su kali linux

di seguito incollo tutte le impostazioni che ho modificato nel servizio inetsim con il comando `sudo nano /etc/inetsim/inetsim.conf`

attivo i servizi https e dns



The screenshot shows a terminal window with the nano text editor open. The title bar indicates the user is 'kali@kali' in the home directory. The editor's status bar shows 'GNU nano 7.2' and the file path '/etc/inetsim/inetsim.conf *'. The file content lists various services to be started, with 'dns' and 'https' being the focus of the configuration. A large, semi-transparent 'KALI LINUX' watermark is visible in the background of the terminal.

```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf *  
#  
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  
#start_service tftp  
#start_service irc  
#start_service ntp  
#start_service finger  
#start_service ident  
#start_service syslog  
#start_service time_tcp  
#start_service time_udp  
#start_service daytime_tcp the quieter you become, the more you are  
#start_service daytime_udp  
#start_service echo_tcp  
#start_service echo_udp  
#start_service discard_tcp
```

collego l'IP di kali linux al server dns, assegno il dominio `epicode.internal` al server dns,

Home

```
#####  
# 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
```

Home

```
#####  
# 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 192.168.32.100
```

```
#####  
# dns_default_domainname  
#  
# Default domain name to return with DNS replies  
#  
# Syntax: dns_default_domainname <domain name>  
#  
# Default: inetsim.org  
#  
dns_default_domainname epicode.internal
```

```
#####  
# dns_static  
#  
# Static mappings for DNS  
#  
# Syntax: dns_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
dns_static epicode.internal 192.168.32.100  
#dns_static ns1.foo.com 10.70.50.30  
#dns_static ftp.bar.net 10.10.20.30
```

configuro il fake file nei servizi http e https

```
#####  
# http_default_fakefile  
#  
# The default fake file returned in fake mode if the file extension  
# in the HTTP request does not match any of the extensions  
# defined above.  
#  
# The default fake file must be placed in <data-dir>/http/fakefiles  
#  
# Syntax: http_default_fakefile <filename> <mime-type>  
#  
# Default: none  
#  
http_default_fakefile sample.html text/html
```

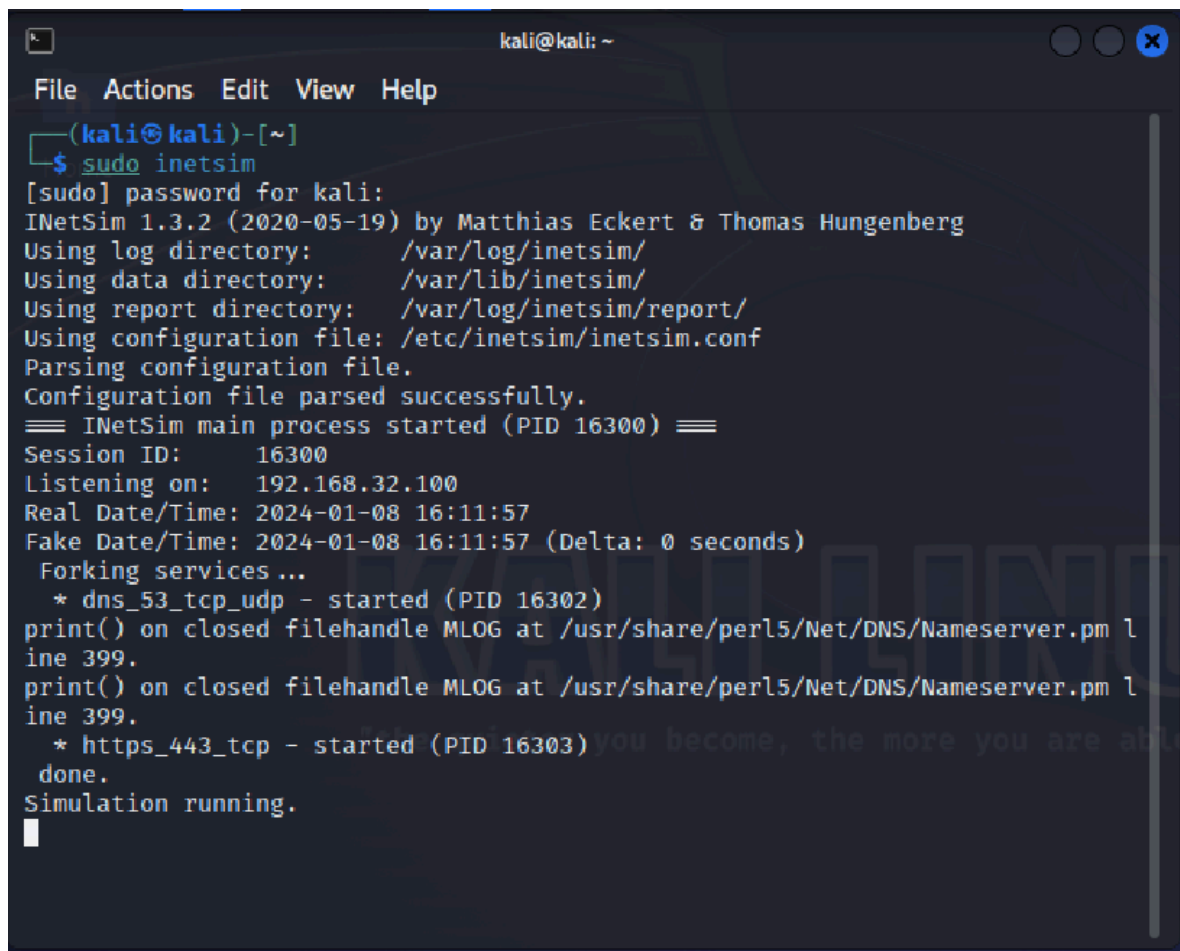
```
#####  
# https_default_fakefile  
#  
# The default fake file returned in fake mode if the file extension  
# in the HTTPS request does not match any of the extensions  
# defined above.  
#  
# The default fake file must be placed in <data-dir>/http/fakefiles  
#  
# Syntax: https_default_fakefile <filename> <mime-type>  
#  
# Default: none  
#  
https_default_fakefile sample.html text/html
```

Fase 4

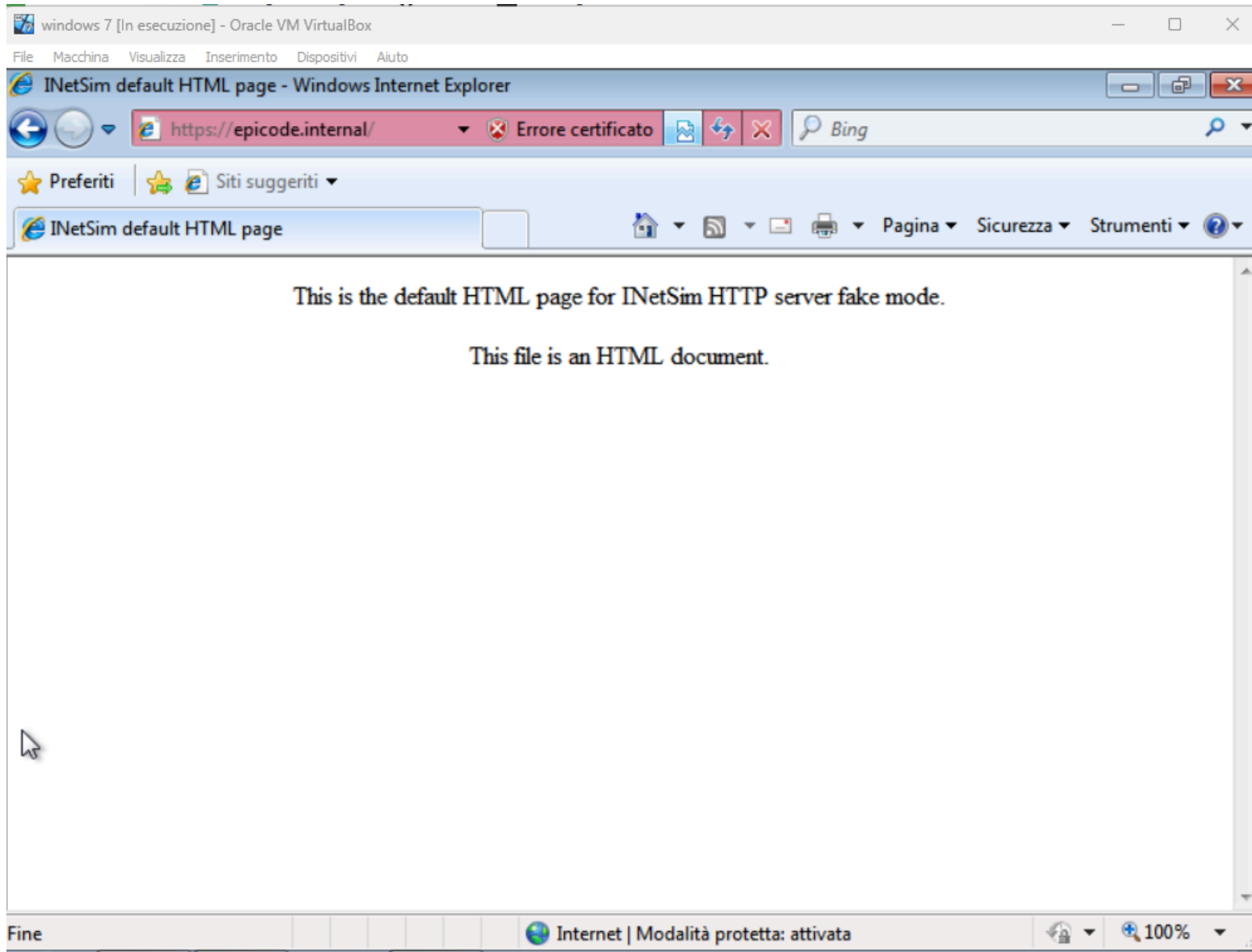
- Attivazione servizio inetsim e test connessione su client W7

In questa fase attivo il servizio inetsim su kali linux dopo averlo configurato con il comando `sudo inetsim`

Dopodichè in ambiente W7, aprendo internet explorer verifico la connessione dei servizi digitando `https://epicode.internal`



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sudo inetsim  
[sudo] password for kali:  
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 16300) ==  
Session ID: 16300  
Listening on: 192.168.32.100  
Real Date/Time: 2024-01-08 16:11:57  
Fake Date/Time: 2024-01-08 16:11:57 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 16302)  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.  
* https_443_tcp - started (PID 16303)  
done.  
Simulation running.  
█
```



Fase 5

- cattura dei pacchetti con wireshark HTTPS

in questa fase faremo la prima cattura dei pacchetti con il servizio https

The image shows a Wireshark network traffic capture. 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 packet capture and analysis. The main display area shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets are color-coded: blue for ARP, yellow for TCP, and green for TLSv1 and NBNS. The first 25 packets are listed, showing an ARP request, a TCP SYN exchange, a TLS handshake (Client Hello, Server Hello, Key Exchange, Change Cipher Spec), and several NBNS queries. The bottom status bar indicates 'any: <live capture in progress>' and 'Packets: 68 · Displayed: 68 (100.0%)'.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000014994	PcsCompu_cb:7e:f5		ARP	44	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.000010309	192.168.32.101	192.168.32.100	TCP	68	49165 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.0000895110	192.168.32.100	192.168.32.101	TCP	68	443 → 49165 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.001767064	192.168.32.101	192.168.32.100	TCP	62	49165 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.003214105	192.168.32.101	192.168.32.100	TLSv1	217	Client Hello
7	0.003228794	192.168.32.100	192.168.32.101	TCP	56	443 → 49165 [ACK] Seq=1 Ack=162 Win=64128 Len=0
8	0.050573041	192.168.32.100	192.168.32.101	TLSv1	1375	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9	0.055733351	192.168.32.101	192.168.32.100	TLSv1	190	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.055756676	192.168.32.100	192.168.32.101	TCP	56	443 → 49165 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
11	0.056677906	192.168.32.100	192.168.32.101	TLSv1	115	Change Cipher Spec, Encrypted Handshake Message
12	0.064912868	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
13	0.261999827	192.168.32.101	192.168.32.100	TCP	62	49165 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len=0
14	0.872143441	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
15	1.872547054	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
16	3.514127434	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
17	4.264589419	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
18	5.014706552	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
19	5.127933697	PcsCompu_cb:7e:f5		ARP	44	Who has 192.168.32.101? Tell 192.168.32.100
20	5.128925174	PcsCompu_0e:c4:ea		ARP	62	192.168.32.101 is at 08:00:27:0e:c4:ea
21	5.767976340	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
22	6.374927326	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
23	7.376407371	PcsCompu_0e:c4:ea		ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
24	9.220789862	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
25	9.970562893	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>

Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface any, id 0
Linux cooked capture v1
Address Resolution Protocol (request)

any: <live capture in progress> Packets: 68 · Displayed: 68 (100.0%) Profile: Default

Fase 6

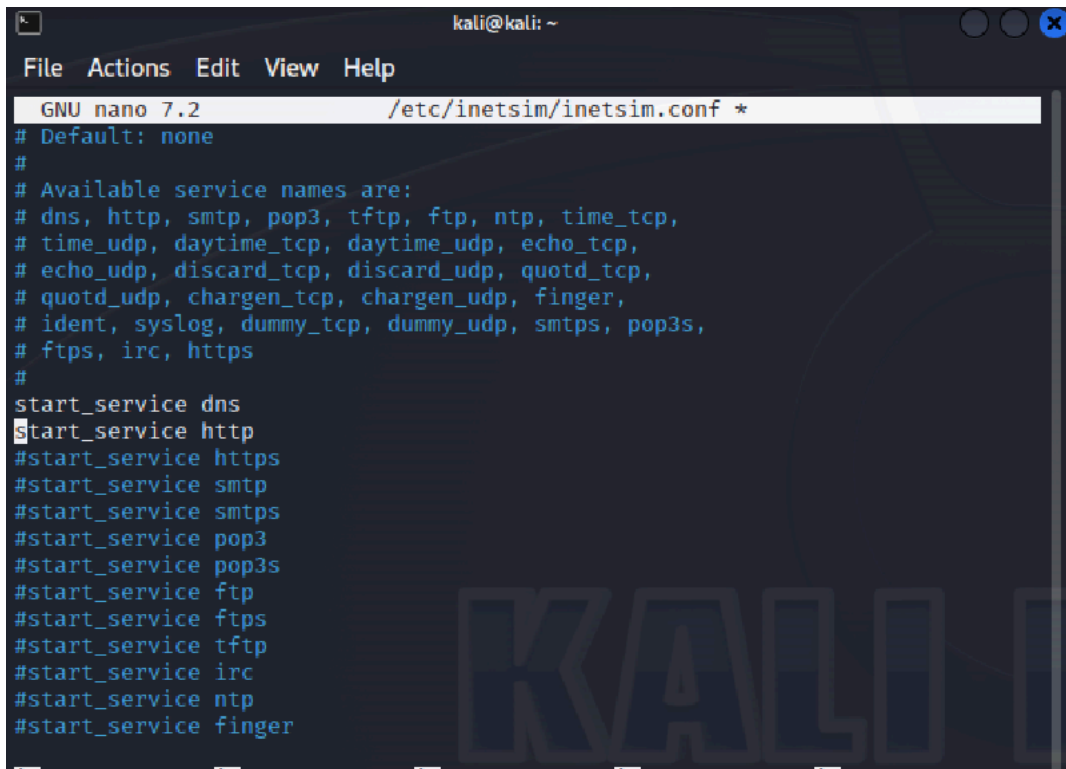
- cattura di pacchetti con wireshark HTTP

in questa fase procediamo alla cattura dei pacchetti con il servizio wireshark; dovrò disattivare il servizio inetsim, modificarlo attivando solo l'HTTP e rilanciare di nuovo il servizio inetsim.

A questo punto andrò su W7 ed in internet explorer testerò il servizio HTTP digitando `http://epicode.internal` e, se il fake file si aprirà correttamente procederò alla seconda cattura di pacchetti con il servizio wireshark.

di seguito incollo tutti i passaggi:

attivazione servizio HTTP e DNS

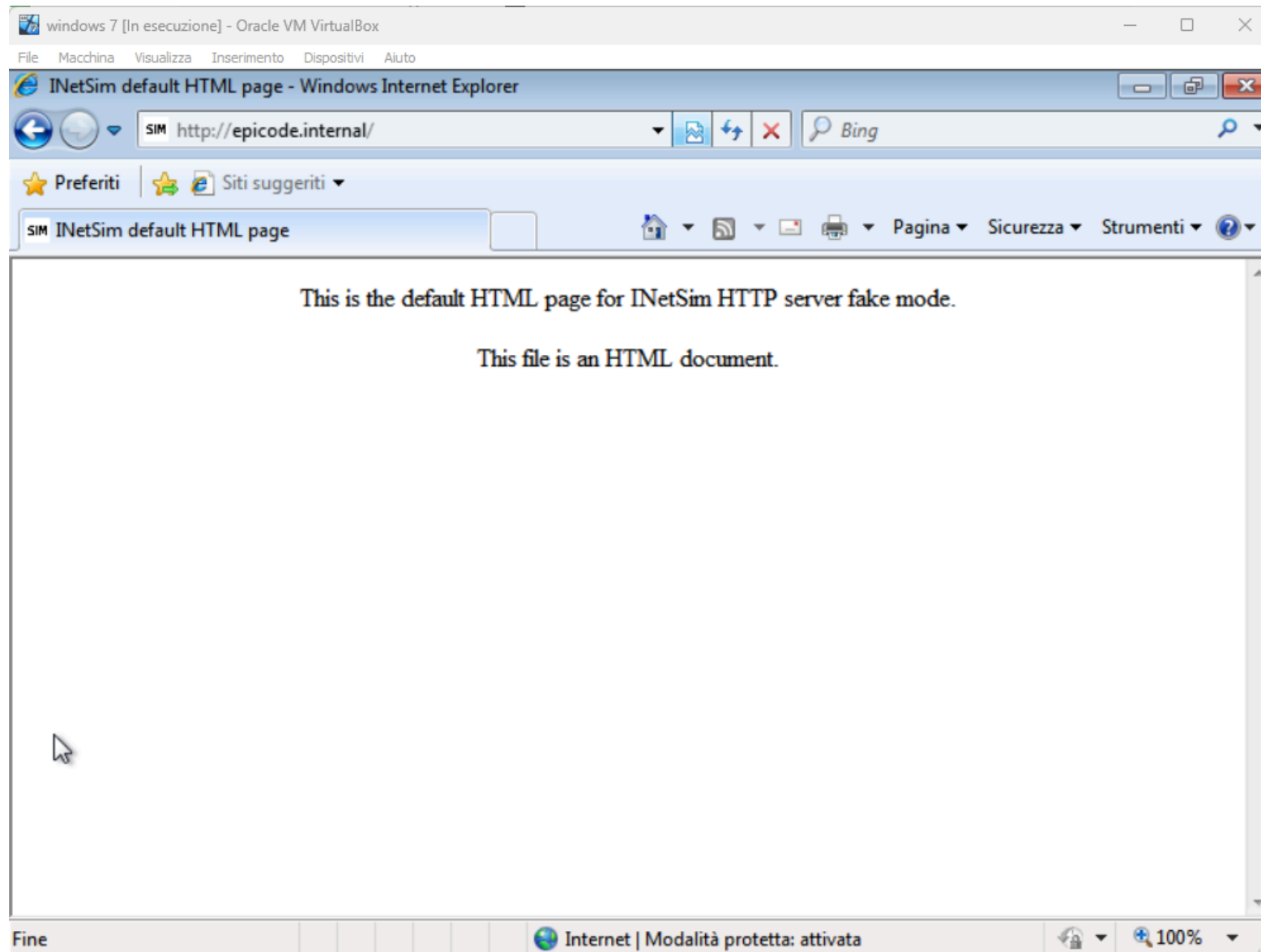


```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf *  
# Default: none  
#  
# Available service names are:  
# dns, http, smtp, pop3, tftp, ftp, ntp, time_tcp,  
# time_udp, daytime_tcp, daytime_udp, echo_tcp,  
# echo_udp, discard_tcp, discard_udp, quotd_tcp,  
# 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  
#start_service smtps  
#start_service pop3  
#start_service pop3s  
#start_service ftp  
#start_service ftps  
#start_service tftp  
#start_service irc  
#start_service ntp  
#start_service finger
```

attivazione servizio inetsim

```
kali@kali: ~  
File Actions Edit View Help  
└─$ sudo nano /etc/inetsim/inetsim.conf  
[sudo] password for kali:  
  
└─(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 23525) ≡  
Session ID:      23525  
Listening on:    192.168.32.100  
Real Date/Time:  2024-01-08 16:26:36  
Fake Date/Time: 2024-01-08 16:26:36 (Delta: 0 seconds)  
Forking services...  
  * dns_53_tcp_udp - started (PID 23527)  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l  
ine 399.  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l  
ine 399.  
  * http_80_tcp - started (PID 23528)  
done.  
Simulation running.  
█
```

test funzionamento servizio



cattura pacchetti HTTP

The image shows a Wireshark network traffic capture. The main packet list displays 12 packets. Packet 4 is selected, showing an HTTP GET request from 192.168.32.101 to 192.168.32.100. The packet details pane shows the Hypertext Transfer Protocol section with various headers. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	68	49171 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
2	0.000119042	192.168.32.100	192.168.32.101	TCP	68	80 → 49171 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.002868331	192.168.32.101	192.168.32.100	TCP	62	49171 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.002868569	192.168.32.101	192.168.32.100	HTTP	363	GET / HTTP/1.1
5	0.002910656	192.168.32.100	192.168.32.101	TCP	56	80 → 49171 [ACK] Seq=1 Ack=308 Win=64128 Len=0
6	0.014677451	192.168.32.100	192.168.32.101	TCP	206	80 → 49171 [PSH, ACK] Seq=1 Ack=308 Win=64128 Len=150 [TCP segment of a reass...
7	0.016003563	192.168.32.100	192.168.32.101	HTTP	314	HTTP/1.1 200 OK (text/html)
8	0.016589648	192.168.32.101	192.168.32.100	TCP	62	49171 → 80 [ACK] Seq=308 Ack=410 Win=65292 Len=0
9	0.017058620	192.168.32.101	192.168.32.100	TCP	62	49171 → 80 [FIN, ACK] Seq=308 Ack=410 Win=65292 Len=0
10	0.017076180	192.168.32.100	192.168.32.101	TCP	56	80 → 49171 [ACK] Seq=410 Ack=309 Win=64128 Len=0
11	5.022832462	PcsCompu_cb:7e:f5		ARP	44	Who has 192.168.32.101? Tell 192.168.32.100
12	5.023737741	PcsCompu_0e:c4:ea		ARP	62	192.168.32.101 is at 08:00:27:0e:c4:ea

Hypertext Transfer Protocol

- GET / HTTP/1.1\r\n
- Accept: */*\r\n
- Accept-Language: it-IT\r\n
- User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0
- Accept-Encoding: gzip, deflate\r\n
- Host: epicode.internal\r\n
- Connection: Keep-Alive\r\n
- \r\n
- [Full request URI: <http://epicode.internal/>]
- [HTTP request 1/1]

0000 00 00 00 01 00 06 08 00 27 0e c4 ea 00 00 08 00 ..
0010 45 00 01 5b 00 be 40 00 80 06 36 c5 c0 a8 20 65 E
0020 c0 a8 20 64 c0 13 00 50 7d 9c 3d 2c 68 f8 ca f9 ..
0030 50 18 40 29 d9 22 00 00 47 45 54 20 2f 20 48 54 P
0040 54 50 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 TP
0050 2a 2f 2a 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 */
0060 75 61 67 65 3a 20 69 74 2d 49 54 0d 0a 55 73 65 ua
0070 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 r-
0080 2f 34 2e 30 20 28 63 6f 6d 70 61 74 69 62 6c 65 /4
0090 3b 20 4d 53 49 45 20 37 2e 30 3b 20 57 69 6e 64 ;
00a0 6f 77 73 20 4e 54 20 36 2e 31 3b 20 57 4f 57 36 ow
00b0 34 3b 20 54 72 69 64 65 6e 74 2f 34 2e 30 3b 20 4;

wireshark_anyC860G2.pcapng Packets: 12 · Displayed: 12 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Fase 7

- valutazione delle differenze tra HTTP e HTTPS

di seguito incollo a titolo esplicativo i due screenshot di cattura.

Il primo è relativo all'HTTP mentre il secondo è relativo all'HTTPS;

la principale differenza tra i due protocolli sta nel fatto che l'HTTP è in chiaro e, attraverso il servizio wireshark, possiamo analizzare il dettaglio di ogni pagina visitata e ricerca fatta;

mentre il protocollo HTTPS utilizza il sistema di crittografia TLS che impedisce ad un soggetto intermedio di conoscere il dettaglio delle richieste eseguite sul browser.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	68	49171 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
2	0.000119042	192.168.32.100	192.168.32.101	TCP	68	80 → 49171 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.002868331	192.168.32.101	192.168.32.100	TCP	62	49171 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.002868569	192.168.32.101	192.168.32.100	HTTP	363	GET / HTTP/1.1
5	0.002910656	192.168.32.100	192.168.32.101	TCP	56	80 → 49171 [ACK] Seq=1 Ack=308 Win=64128 Len=0
6	0.014677451	192.168.32.100	192.168.32.101	TCP	206	80 → 49171 [PSH, ACK] Seq=1 Ack=308 Win=64128 Len=150 [TCP segment of a reass...
7	0.016003563	192.168.32.100	192.168.32.101	HTTP	314	HTTP/1.1 200 OK (text/html)
8	0.016589648	192.168.32.101	192.168.32.100	TCP	62	49171 → 80 [ACK] Seq=308 Ack=410 Win=65292 Len=0
9	0.017058620	192.168.32.101	192.168.32.100	TCP	62	49171 → 80 [FIN, ACK] Seq=308 Ack=410 Win=65292 Len=0
10	0.017076180	192.168.32.100	192.168.32.101	TCP	56	80 → 49171 [ACK] Seq=410 Ack=309 Win=64128 Len=0
11	5.022832462	PcsCompu_cb:7e:f5		ARP	44	Who has 192.168.32.101? Tell 192.168.32.100
12	5.023737741	PcsCompu_0e:c4:ea		ARP	62	192.168.32.101 is at 08:00:27:0e:c4:ea

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	68	49166 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
2	0.000033889	192.168.32.100	192.168.32.101	TCP	68	443 → 49166 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.000785209	192.168.32.101	192.168.32.100	TCP	62	49166 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.000785305	192.168.32.101	192.168.32.100	TLSv1	217	Client Hello
5	0.000813285	192.168.32.100	192.168.32.101	TCP	56	443 → 49166 [ACK] Seq=1 Ack=162 Win=64128 Len=0
6	0.030407800	192.168.32.100	192.168.32.101	TLSv1	1375	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.034504037	192.168.32.101	192.168.32.100	TLSv1	190	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
8	0.035092529	192.168.32.100	192.168.32.101	TLSv1	115	Change Cipher Spec, Encrypted Handshake Message
9	0.038389179	192.168.32.101	192.168.32.100	TLSv1	397	Application Data
10	0.044817961	192.168.32.100	192.168.32.101	TLSv1	237	Application Data
11	0.046347829	192.168.32.100	192.168.32.101	TLSv1	386	Application Data, Encrypted Alert
12	0.047815342	192.168.32.101	192.168.32.100	TCP	62	49166 → 443 [ACK] Seq=637 Ack=1891 Win=65700 Len=0
13	0.048317791	192.168.32.101	192.168.32.100	TCP	62	49166 → 443 [FIN, ACK] Seq=637 Ack=1891 Win=65700 Len=0
14	0.048332741	192.168.32.100	192.168.32.101	TCP	56	443 → 49166 [ACK] Seq=1891 Ack=638 Win=64128 Len=0