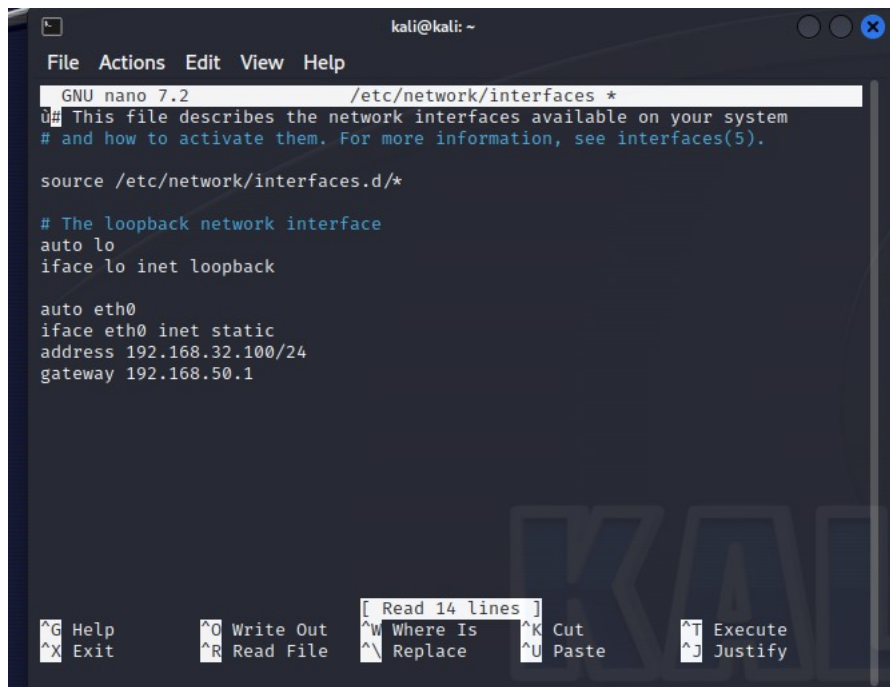
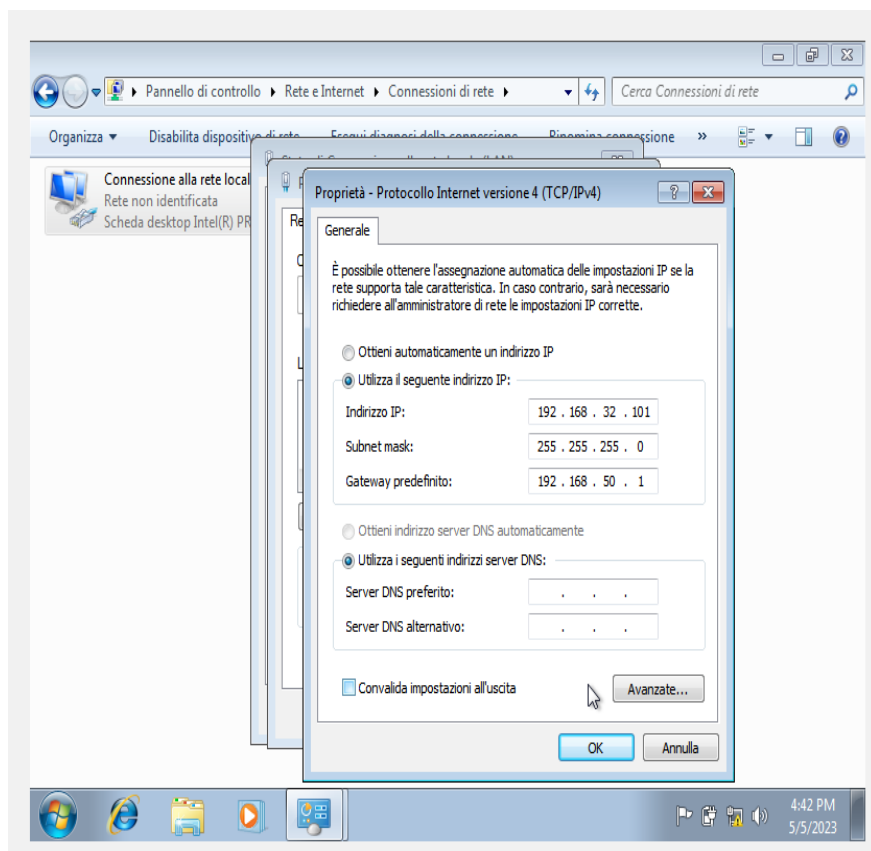


Simulazione rete complessa

Il seguente report ha lo scopo di descrivere come ho effettuato l'esercizio assegnato nel corso di Epicode in Cybersecurity. L'esercizio prevedeva l'assegnazione di un nuovo indirizzo IP alle macchine virtuali Kali e Windows utilizzate durante la sessione di laboratorio.



```
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 192.168.32.100/24  
gateway 192.168.50.1
```

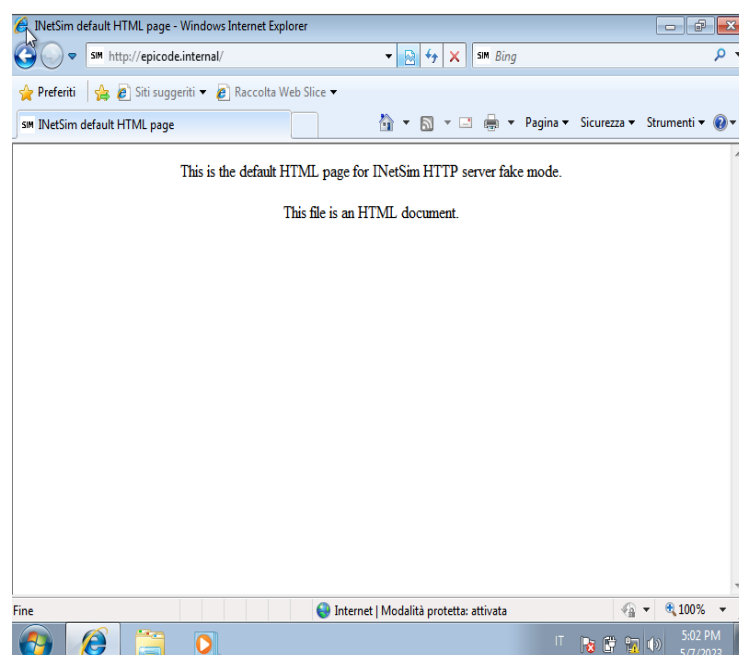
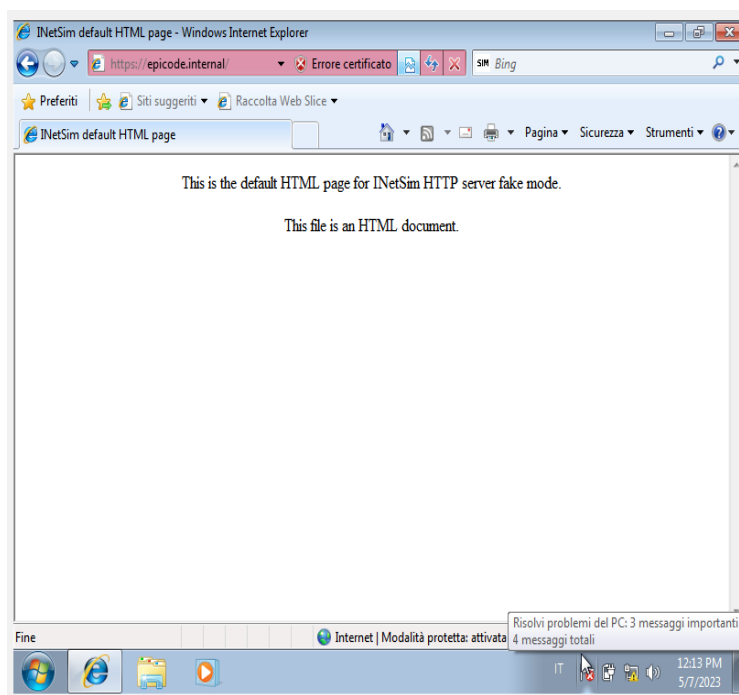


Successivamente con Inetsim uno strumento che consente di creare un ambiente di test per vari servizi di rete, ho creato un'architettura client-server simulando un server DNS e HTTP/HTTPS. In questo modo il client con indirizzo 192.168.32.101 può richiedere una risorsa tramite browser web all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100

```
#####
# service_bind_address
#
# IP address to bind services to (336 bits), 42
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
#service_bind_address 10.10.10.1
service_bind_address 192.168.32.100
```

```
#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address> (bits), 42
# Syntax: dns_default_ip <IP address> (bits), 42
# Default: 127.0.0.1
#
#dns_default_ip 10.10.10.1
dns_default_ip 192.168.32.100
```

```
(kali㉿kali)-[~]
└─$ sudo inetsim
InetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenb
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 75742) ==
Session ID: 75742
Listening on: 192.168.32.100
Real Date/Time: 2023-05-07 06:12:28
Fake Date/Time: 2023-05-07 06:12:28 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 75744)
* ntp_123_udp - started (PID 75754)
* irc_6667_tcp - started (PID 75753)
* ftp_21_tcp - started (PID 75750)
* discard_9_udp - started (PID 75766)
* ident_113_tcp - started (PID 75756)
* pop3s_995_tcp - started (PID 75749)
* finger_79_tcp - started (PID 75755)
* smtps_465_tcp - started (PID 75747)
* syslog_514_udp - started (PID 75757)
* daytime_13_tcp - started (PID 75761)
* daytime_13_udp - started (PID 75762)
* smtp_25_tcp - started (PID 75746)
* echo_7_tcp - started (PID 75763)
* tftp_69_udp - started (PID 75752)
* echo_7_udp - started (PID 75764)
* time_37_tcp - started (PID 75758)
* time_37_udp - started (PID 75760)
* https_443_tcp - started (PID 75745)
* chargen_19_tcp - started (PID 75769)
* discard_9_tcp - started (PID 75765)
* dummy_1_tcp - started (PID 75771)
* pop3_110_tcp - started (PID 75748)
* dummy_1_udp - started (PID 75772)
* ftps_990_tcp - started (PID 75751)
* quotd_17_udp - started (PID 75768)
* chargen_19_udp - started (PID 75770)
* quotd_17_tcp - started (PID 75767)
done.
Simulation running.
```



Ho utilizzato Wireshark per catturare il traffico HTTPS e HTTP

The screenshot shows a Wireshark capture of network traffic on interface eth0. The display filter is set to 'Apply a display filter ... <Ctrl-F>'. The packet list shows a series of packets, with packet 379 selected. The packet details pane shows the following structure:

- Frame 40: 379 bytes on wire (3032 bits), 379 bytes captured (3032 bits) on interface eth0, id 0
- Ethernet II, Src: PcsCompu_71:0a:02 (08:00:27:71:0a:02), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)
- Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
- Transmission Control Protocol, Src Port: 49199, Dst Port: 443, Seq: 296, Ack: 1379, Len: 325
- Transport Layer Security
 - TLSv1 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
 - Content Type: Application Data (23)
 - Version: TLS 1.0 (0x0301)
 - Length: 320
 - Encrypted Application Data: 5b85640de6cb1d29abaf8adaa0d7363e6deac807354a349f8a7a6ae1293c9501aaac
 - [Application Data Protocol: Hypertext Transfer Protocol]

The packet bytes pane shows the raw data of the selected packet, including the TLS record structure and the encrypted application data.

The screenshot shows a Wireshark capture of network traffic on interface eth0. The display filter is set to 'Apply a display filter ... <Ctrl-F>'. The packet list shows a series of packets, with packet 6 selected. The packet details pane shows the following structure:

- Frame 6: 340 bytes on wire (2720 bits), 340 bytes captured (2720 bits) on interface eth0, id 0
- Ethernet II, Src: PcsCompu_71:0a:02 (08:00:27:71:0a:02), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)
- Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
- Transmission Control Protocol, Src Port: 49198, Dst Port: 80, Seq: 1, Ack: 1, Len: 286
- Hypertext Transfer Protocol
 - GET / HTTP/1.1
 - Accept: */*
 - Accept-Language: en-US
 - User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR ...)
 - Accept-Encoding: gzip, deflate
 - Host: epicode.internal
 - Connection: Keep-Alive
 - Full request URI: http://epicode.internal/
 - [HTTP request 1/1]
 - [Response in frame: 9]

The packet bytes pane shows the raw data of the selected packet, including the HTTP request structure and the response data.

Si può notare che esiste una differenza tra i pacchetti HTTP e HTTPS mentre HTTPS utilizza una connessione sicura SSL/TLS (Secure Sockets Layer/Transport Layer Security) per crittografare i dati trasmessi tra il client e il server, HTTP non lo fa. Per questo motivo mentre i dati trasmessi tramite HTTP sono inviati in chiaro e possono essere intercettati e letti da terze parti, i dati trasmessi tramite HTTPS sono crittografati e quindi più sicuri. Ciò rende HTTPS più adatto per la trasmissione di dati sensibili, come informazioni di login, dati di pagamento, informazioni personali. Per garantire la sicurezza della connessione HTTPS, il server e il client devono autenticarsi a vicenda e stabilire una chiave di crittografia condivisa utilizzando il protocollo SSL/TLS. Questa chiave viene utilizzata per crittografare e decrittografare i dati trasmessi tra il client e il server.