

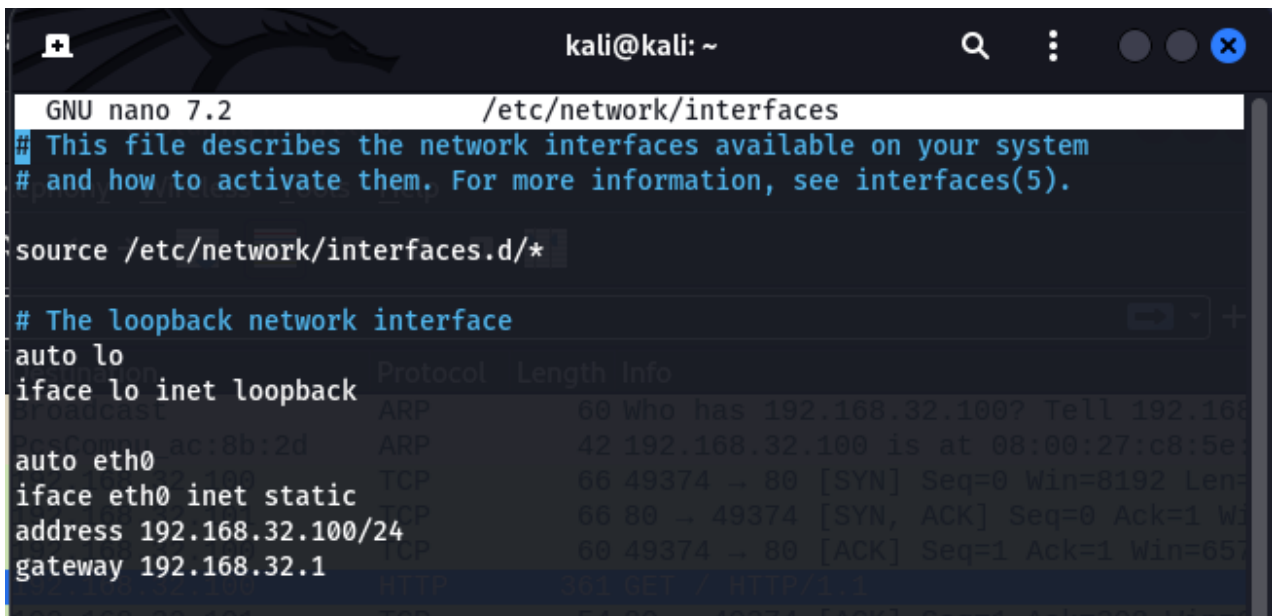
# Simulazione rete complessa

Requisiti e servizi:

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

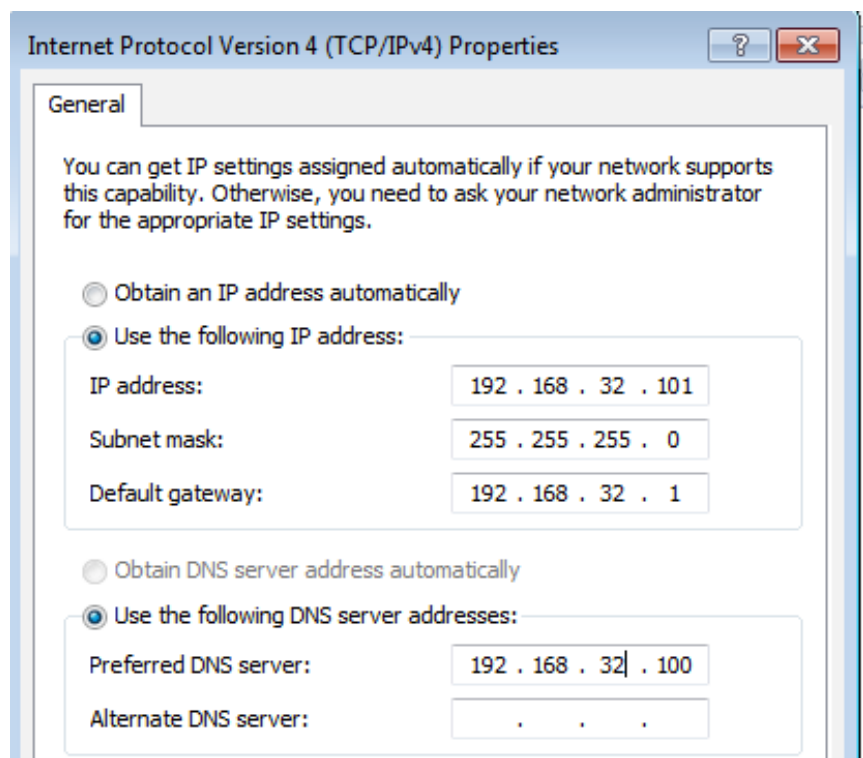
Traccia: Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100. Si intercetti poi la comunicazione con Wireshark, evidenziando 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 differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze.

Assegno IP statico su Kali : 192.168.32.100



```
kali@kali: ~  
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 192.168.32.1
```

Assegno IP statico su Win:  
192.168.32.101  
con inserimento dell'IP di Kali  
come server DNS  
preferito



Tramite inetsim verifico che siano attivi i servizi HTTPS, HTTP e DNS

```
GNU nano 7.2 /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
```

Configuro il server DNS per fare l'associazione tra epicode.internal e IP di Kali

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

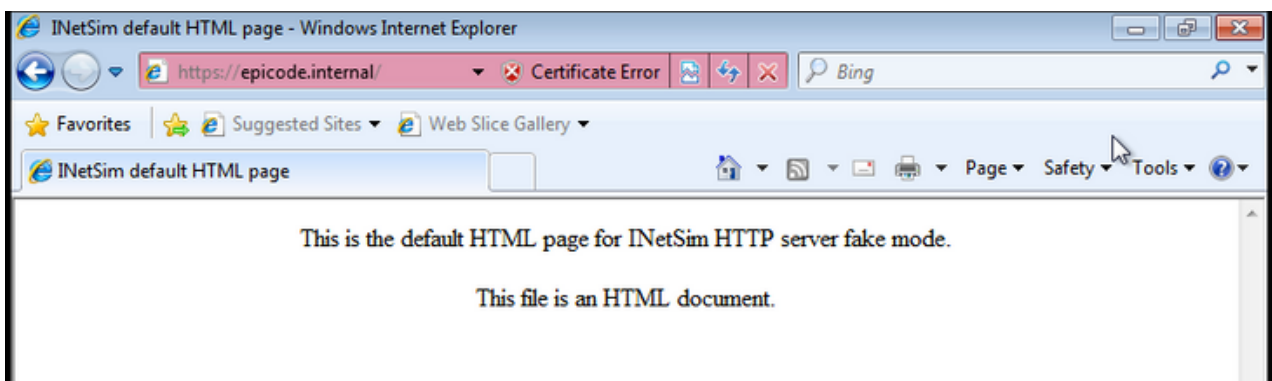
Utilizzo l'indirizzo IP di Kali 192.168.32.100 per il binding dei servizi per renderli disponibili esternamente alla macchina

```
GNU nano 7.2 /etc/inetsim/inetsim.conf
#####
# service_bind_address Help
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100
```

Abilito servizi tramite inetsim lanciando il comando "sudo inetsim" per non rischiare di abusare dei privilegi di root per compiti di routine che non ne richiedono l'uso. Avendo riscontrato dei problemi ho eseguito il comando "inetsim" accedendo prima come root.

```
└─$ sudo su
(root@kali)-[/home/kali]
# inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Main logfile '/var/log/inetsim/main.log' does not exist. Trying to create it...
Main logfile '/var/log/inetsim/main.log' successfully created.
Sub logfile '/var/log/inetsim/service.log' does not exist. Trying to create it..
.
Sub logfile '/var/log/inetsim/service.log' successfully created.
Debug logfile '/var/log/inetsim/debug.log' does not exist. Trying to create it..
.
Debug logfile '/var/log/inetsim/debug.log' successfully created.
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 2565) ===
Session ID:      2565
Listening on:    192.168.32.100
Real Date/Time:  2023-11-25 18:31:21
Fake Date/Time:  2023-11-25 18:31:21 (Delta: 0 seconds)
Forking services...
```

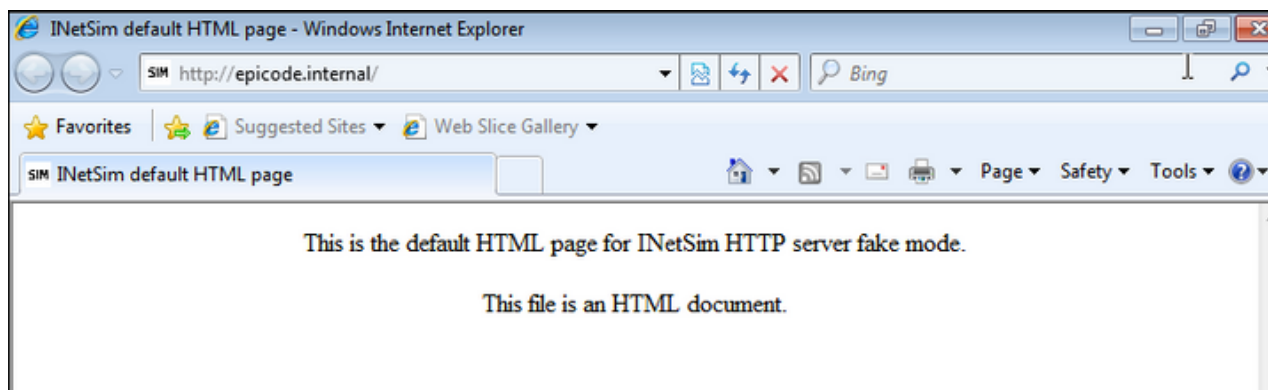
Da macchina Win testo la connettività al servizio HTTPS



Con Wireshark intercetto e catturo il traffico di rete

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_ac:8b:2d	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000054820	PcsCompu_c8:5e:3f	PcsCompu_ac:8b:2d	ARP	42	192.168.32.100 is at 08:00:27:c8:5e:3f
3	0.001523657	192.168.32.101	192.168.32.100	TCP	66	49292 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.001555999	192.168.32.100	192.168.32.101	TCP	66	443 → 49292 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.002246452	192.168.32.101	192.168.32.100	TCP	60	49292 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.002723762	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
7	0.002747377	192.168.32.100	192.168.32.101	TCP	54	443 → 49292 [ACK] Seq=1 Ack=162 Win=64128 Len=0
8	0.008373012	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9	0.002184095	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.002218402	192.168.32.100	192.168.32.101	TCP	54	443 → 49292 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
11	0.007111833	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12	0.006969605	192.168.32.101	192.168.32.100	TLSv1	395	Application Data
13	0.117299387	192.168.32.100	192.168.32.101	TLSv1	235	Application Data
14	0.121481017	192.168.32.100	192.168.32.101	TLSv1	384	Application Data, Encrypted Alert
15	0.122149417	192.168.32.101	192.168.32.100	TCP	60	49292 → 443 [ACK] Seq=637 Ack=1891 Win=65700 Len=0
16	0.122741521	192.168.32.101	192.168.32.100	TCP	60	49292 → 443 [FIN, ACK] Seq=637 Ack=1891 Win=65700 Len=0
17	0.122760870	192.168.32.100	192.168.32.101	TCP	54	443 → 49292 [ACK] Seq=1891 Ack=638 Win=64128 Len=0

Da macchina Win testo la connettività al servizio HTTP



Con Wireshark intercetto e catturo il traffico di rete

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_ac:8b:2d	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000019883	PcsCompu_c8:5e:3f	PcsCompu_ac:8b:2d	ARP	42	192.168.32.100 is at 08:00:27:c8:5e:3f
6	0.001700446	192.168.32.101	192.168.32.100	HTTP	301	GET / HTTP/1.1
9	0.033895139	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
16	1.071987792	192.168.32.101	192.168.32.100	HTTP	301	GET / HTTP/1.1
19	1.115739749	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
3	0.000499998	192.168.32.101	192.168.32.100	TCP	66	49319 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.000521809	192.168.32.100	192.168.32.101	TCP	66	80 → 49319 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.001397418	192.168.32.101	192.168.32.100	TCP	66	49319 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
7	0.001781337	192.168.32.100	192.168.32.101	TCP	54	80 → 49319 [ACK] Seq=1 Ack=308 Win=64128 Len=0
8	0.030599429	192.168.32.100	192.168.32.101	TCP	204	80 → 49319 [PSH, ACK] Seq=1 Ack=308 Win=64128 Len=150 [TCP segment of a reassembled PDU]
10	0.036974168	192.168.32.101	192.168.32.100	TCP	66	49319 → 80 [ACK] Seq=308 Ack=410 Win=65292 Len=0
11	0.037759347	192.168.32.101	192.168.32.100	TCP	66	49319 → 80 [FIN, ACK] Seq=308 Ack=410 Win=65292 Len=0
12	0.037785399	192.168.32.100	192.168.32.101	TCP	54	80 → 49319 [ACK] Seq=410 Ack=309 Win=64128 Len=0
13	1.070166129	192.168.32.101	192.168.32.100	TCP	66	49322 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
14	1.070267324	192.168.32.100	192.168.32.101	TCP	66	80 → 49322 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
15	1.071295599	192.168.32.101	192.168.32.100	TCP	66	49322 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
17	1.072029549	192.168.32.100	192.168.32.101	TCP	54	80 → 49322 [ACK] Seq=1 Ack=308 Win=64128 Len=0
18	1.107673869	192.168.32.100	192.168.32.101	TCP	204	80 → 49322 [PSH, ACK] Seq=1 Ack=308 Win=64128 Len=150 [TCP segment of a reassembled PDU]
20	1.117830429	192.168.32.101	192.168.32.100	TCP	66	49322 → 80 [ACK] Seq=308 Ack=410 Win=65292 Len=0
21	1.118338874	192.168.32.101	192.168.32.100	TCP	66	49322 → 80 [FIN, ACK] Seq=308 Ack=410 Win=65292 Len=0
22	1.118358433	192.168.32.100	192.168.32.101	TCP	54	80 → 49322 [ACK] Seq=410 Ack=309 Win=64128 Len=0

Confronto catture:

Analizzando le due catture di rete si evidenzia una distinzione fondamentale legata alla sicurezza dei dati trasmessi. Nella cattura relativa al protocollo HTTP, è possibile osservare direttamente il contenuto delle richieste poiché HTTP trasmette i dati in forma non cifrata. La cattura HTTPS mostra invece l'implementazione del protocollo TLS, che svolge un ruolo cruciale nella creazione di un canale cifrato.

Quindi mentre HTTP espone i dati in chiaro, HTTPS utilizza il TLS per cifrare i dati prima della loro trasmissione, assicurando così una maggiore sicurezza.