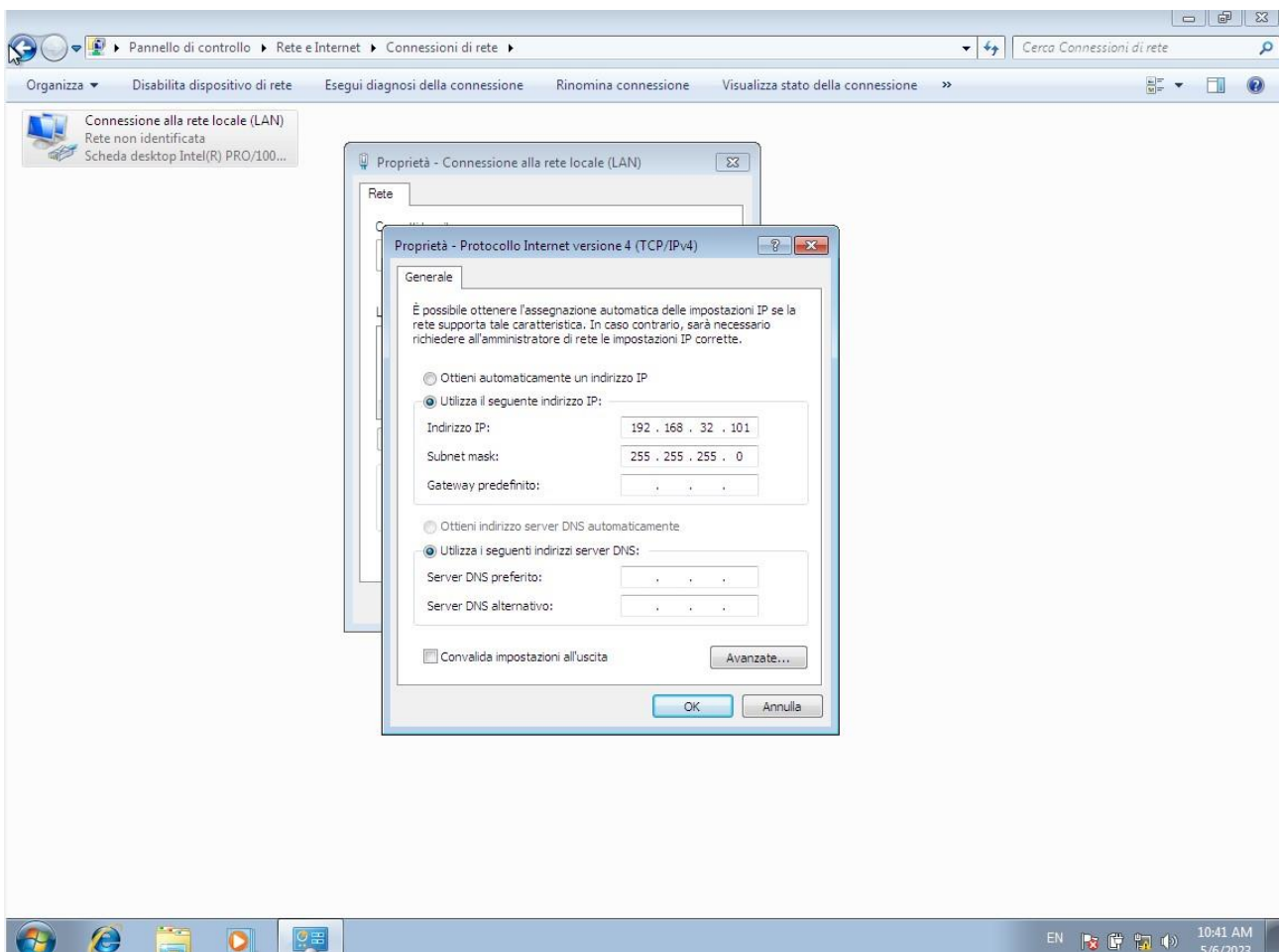


# ESERCIZIO WEEK 1

## 1. IMPOSTAZIONE DI NUOVI INDIRIZZI IP KALI/WINDOWS

Per prima cosa, da WINDOWS ho modificato l'indirizzo ip entrando nelle proprietà della scheda di rete, impostando i seguenti valori:

- Indirizzo IP 192.168.132.100
- Subnet mask 255.255.255.0



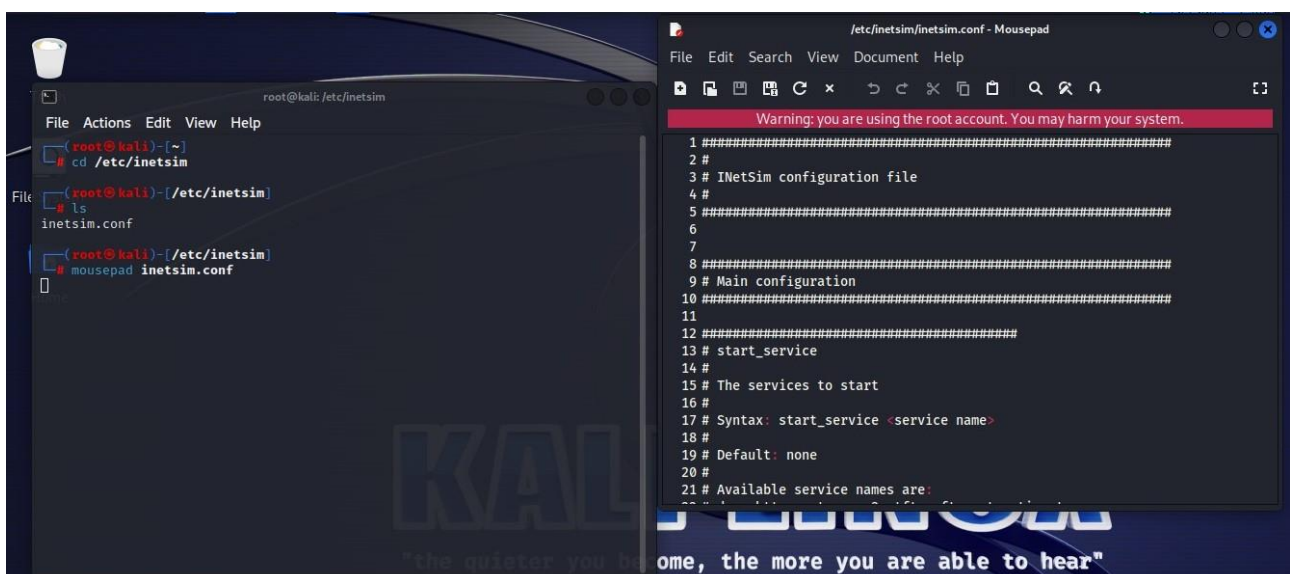
Successivamente, da KALI utilizzando nel terminale la riga di comando “nano /etc/network/interfaces”, ho modificato l'indirizzo ip in 192.168.32.101 e il gateway in 192.168.32.1



```
root@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  
  
[ Read 13 lines ]  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
```

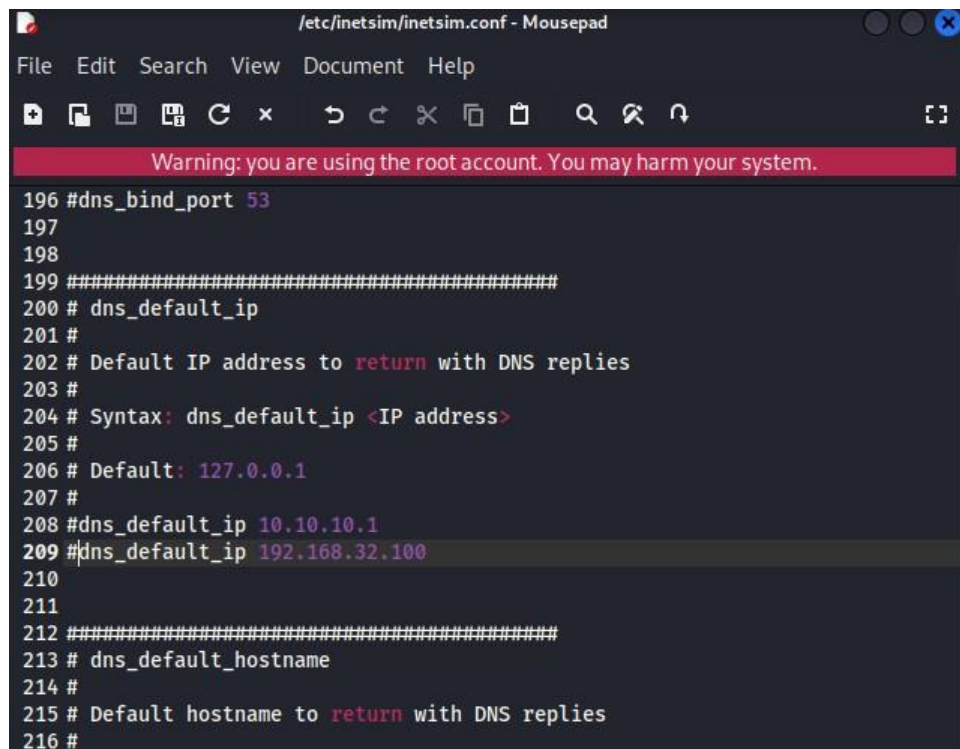
## 2. Avvio server DNS e HTTPS su Kali e configurazione DNS

Per poter eseguire questo passaggio mi serve “inetsim”. L'applicativo è già installato sulla macchina virtuale, quindi ho digitato nel terminale direttamente inetsim. Questo comando, farà partire la simulazione avviando i server DNS e HTTPS, prima ho bisogno però di modificare le impostazioni del file di configurazione. Sempre dal terminale ho raggiunto il file “inetsim.conf” dalla cartella “/etc/inetsim” e l’ho eseguito digitando “mousepad inetsim.conf”



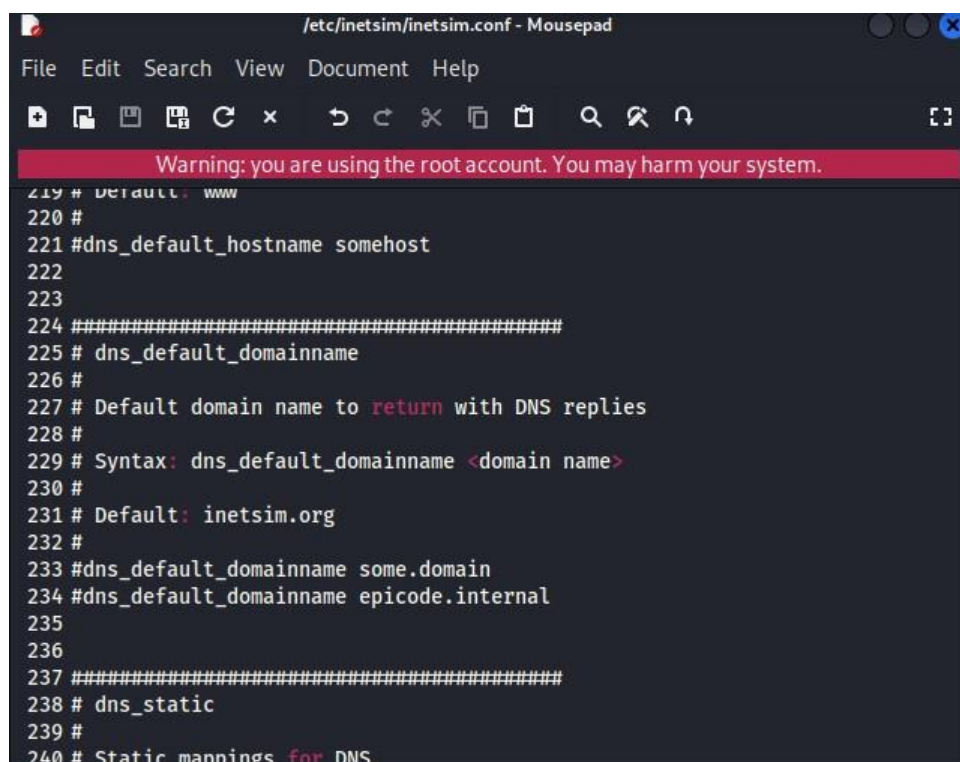
```
root@kali: /etc/inetsim  
File Actions Edit View Help  
(root@kali) ~  
# cd /etc/inetsim  
(root@kali) ~  
ls  
inetsim.conf  
(root@kali) ~  
# mousepad inetsim.conf  
  
Warning: you are using the root account. You may harm your system.  
1 #####  
2 #  
3 # InetSim configuration file  
4 #  
5 #####  
6 #  
7 #  
8 #####  
9 # Main configuration  
10 #####  
11 #  
12 #####  
13 # start_service  
14 #  
15 # The services to start  
16 #  
17 # Syntax: start_service <service name>  
18 #  
19 # Default: none  
20 #  
21 # Available service names are:  
22 #
```

Dalla schermata di configurazione appena aperta, lasciando tutto intoccato ho cercato la riga “dns\_default\_ip” e ho aggiunto una seconda riga con l’indirizzo ip di KALI appena impostato



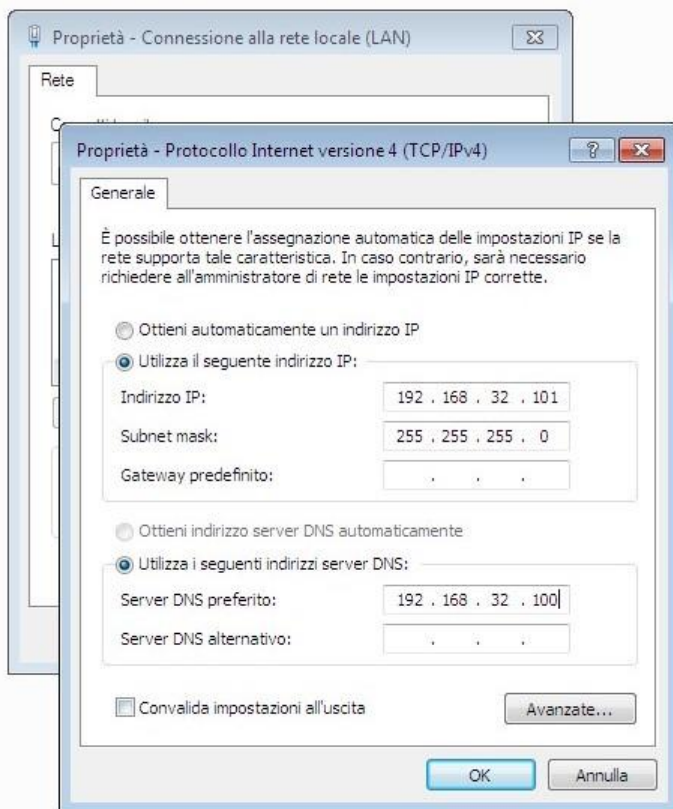
```
/etc/inetsim/inetsim.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
196 #dns_bind_port 53
197
198
199 #####
200 # dns_default_ip
201 #
202 # Default IP address to return with DNS replies
203 #
204 # Syntax: dns_default_ip <IP address>
205 #
206 # Default: 127.0.0.1
207 #
208 #dns_default_ip 10.10.10.1
209 #dns_default_ip 192.168.32.100
210
211
212 #####
213 # dns_default_hostname
214 #
215 # Default hostname to return with DNS replies
216 #
```

Poco più giù sono andato ad impostare anche il domain name su epicode.internal aggiungendo l’apposita riga



```
/etc/inetsim/inetsim.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
219 # Default: www
220 #
221 #dns_default_hostname somehost
222
223
224 #####
225 # dns_default_domainname
226 #
227 # Default domain name to return with DNS replies
228 #
229 # Syntax: dns_default_domainname <domain name>
230 #
231 # Default: inetsim.org
232 #
233 #dns_default_domainname some.domain
234 #dns_default_domainname epicode.internal
235
236
237 #####
238 # dns_static
239 #
240 # Static mappings for DNS
```

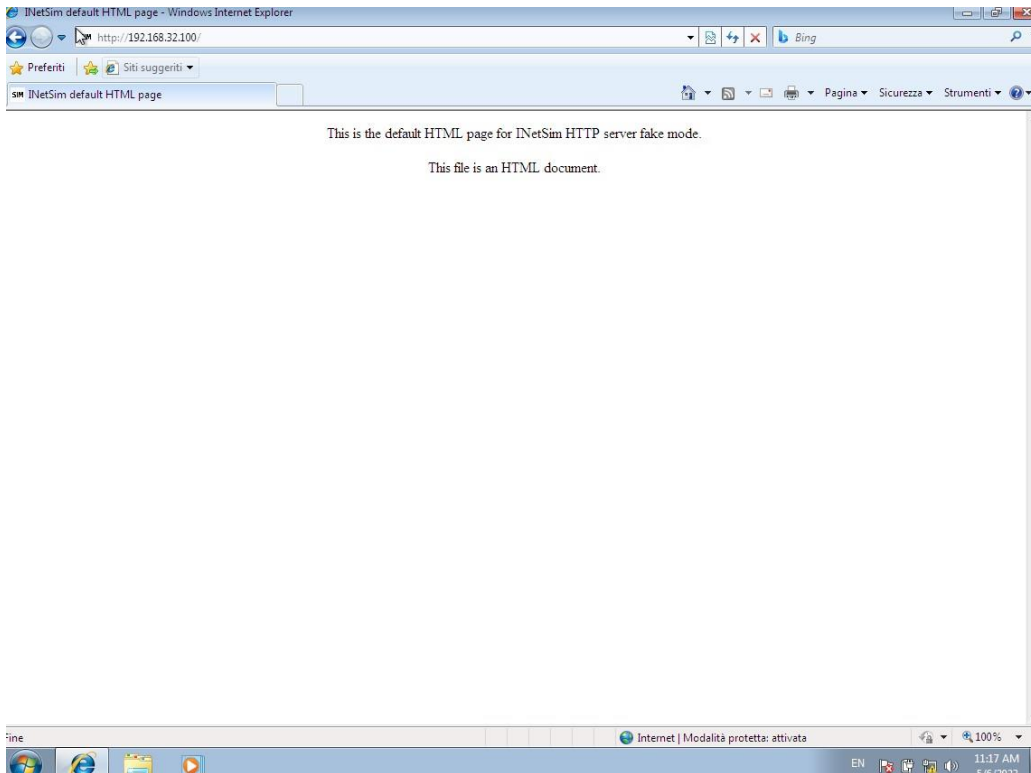
Fatta questa operazione mi serve che WINDOWS utilizzi questo DNS per risolvere il nome dell'host, quindi sempre dalla finestra delle impostazioni della scheda, ho aggiunto l'ip di KALI sul server DNS



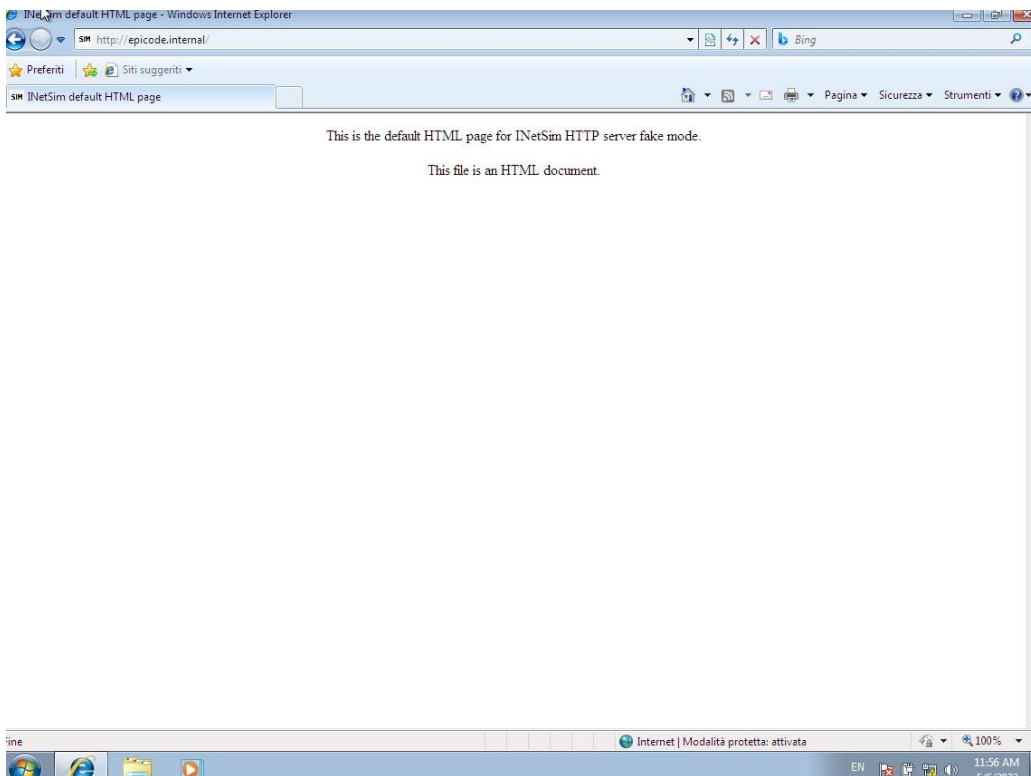
A questo punto faccio partire la simulazione dal terminale KALI semplicemente digitando "inetsim" nel terminale, in modo che vengano avviati tutti i servizi.

```
root@kali: ~  
File Actions Edit View Help  
root@kali) ~  
# 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 165018) ==  
Session ID: 165018  
Listening on: 0.0.0.0  
Real Date/Time: 2023-05-06 05:14:39  
Fake Date/Time: 2023-05-06 05:14:39 (Delta: 0 seconds)  
Forking services...  
* dns_53_tcp_udp - started (PID 165038)  
* irc_6667_tcp - started (PID 165048)  
* syslog_514_udp - started (PID 165052)  
* ident_113_tcp - started (PID 165051)  
* time_37_udp - started (PID 165054)  
* ntp_123_udp - started (PID 165049)  
* finger_79_tcp - started (PID 165050)  
* time_37_tcp - started (PID 165053)  
* daytime_13_udp - started (PID 165056)  
* daytime_13_tcp - started (PID 165055)  
* echo_7_tcp - started (PID 165057)  
* discard_9_tcp - started (PID 165059)  
* discard_9_udp - started (PID 165060)  
* echo_7_udp - started (PID 165058)  
* tftp_69_udp - started (PID 165047)  
* smtp_25_tcp - started (PID 165041)  
* quotd_17_udp - started (PID 165062)  
* pop3s_995_tcp - started (PID 165044)  
* chargen_19_tcp - started (PID 165063)  
* smtps_465_tcp - started (PID 165042)  
* ftp_21_tcp - started (PID 165045)  
* dummy_1_tcp - started (PID 165065)  
* quotd_17_tcp - started (PID 165061)  
* chargen_19_udp - started (PID 165064)  
* dummy_1_udp - started (PID 165066)  
* ftps_990_tcp - started (PID 165046)  
* https_443_tcp - started (PID 165040)  
* pop3_110_tcp - started (PID 165043)  
* http_80_tcp - started (PID 165039)  
done.  
Simulation running.
```

Per verificare che il server http/https sia funzionante, da WINDOWS utilizzando il web browser internet explorer raggiungo il sito fake inetsim digitando nella barra degli indirizzi l'indirizzo IP.

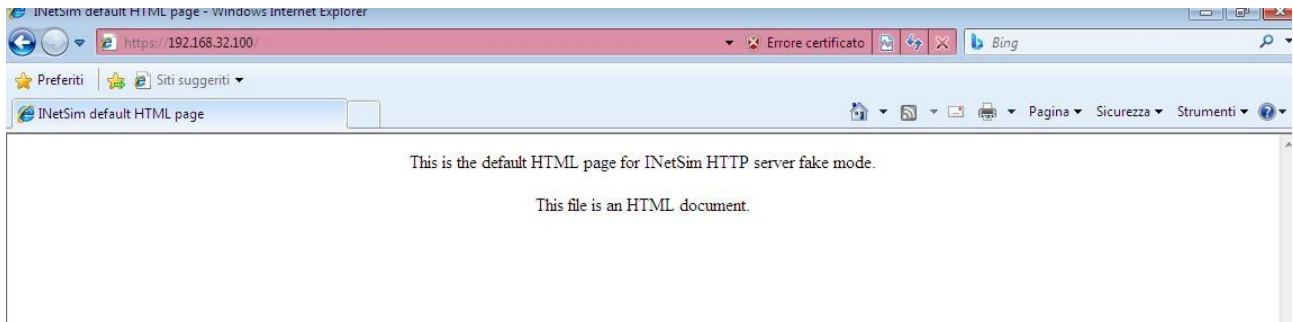


Verifico anche che il server risolva il DNS digitando epicode.internal nella barra di ricerca. Il risultato è positivo



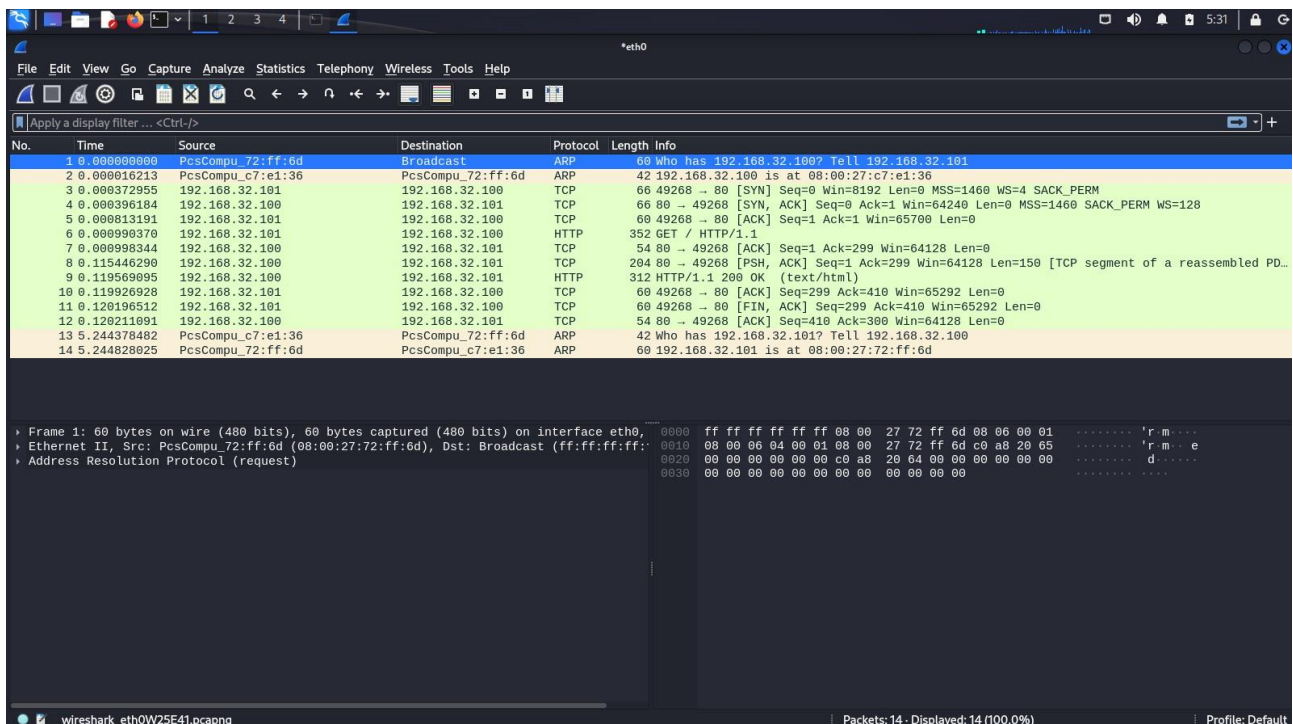


L'ultima verifica da fare è controllare se il sito HTTPS sia raggiungibile. Internet explorer restituisce un errore di attendibilità sui certificati ed è necessario continuare ignorando l'allert: il sito risulta comunque raggiungibile.



### 3. CATTURA DI PACCHETTI CON WIRESHARK

A questo punto per catturare i pacchetti di dati scambiati tra client e server, ho bisogno di Wireshark. Lo avvio direttamente dagli strumenti di sniffing & spoofing di KALI, faccio partire la cattura e da WINDOWS mi collego al sito fake. I risultati sono i seguenti:



Noto innanzitutto nella riga selezionata che tramite protocollo ARP il client, noto l'IP, sta effettuando una chiamata per conoscere l'indirizzo MAC di destinazione. Il messaggio di broadcast ff:ff:ff:ff:ff:ff interroga il nodo e restituisce il MAC richiesto alla sorgente (who has 192.168.32.100? tell 192.168.32.101).

Nelle righe in verde è possibile vedere lo scambio di risposte tra client e server che instaurano una comunicazione attraverso il protocollo di trasporto TCP con il "three-way handshake": il server invia il SEQ 0,

il client risponde con SEQ 0 e ACK 1, e di nuovo il server restituisce SEQ 1 e ACK 1. Il punto saliente successivo è il metodo GET in risposta dal server con protocollo http e la risposta positiva 200 OK come risultato del successo della scambio di dati.

Adesso voglio avere evidenza degli indirizzi MAC: per farlo evidenzio la prima riga, e nel riquadro sottostante nella riga **blu** visualizzo gli indirizzi di origine e destinazione al livello di trasporto delle comunicazioni di rete. Se evidenzio la seconda riga, noto gli stessi indirizzi, ovviamente variati come origine e destinazione, in seguito allo scambio di risposte e la richiesta broadcast al nodo tramite ARP.

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_72:ff:6d	Broadcast	ARP	60	Who has 192.168
2	0.000016213	PcsCompu_c7:e1:36	PcsCompu_72:ff:6d	ARP	42	192.168.32.100
3	0.000372955	192.168.32.101	192.168.32.100	TCP	66	49268 → 80 [SYN
4	0.000396184	192.168.32.100	192.168.32.101	TCP	66	80 → 49268 [SYN
5	0.000813191	192.168.32.101	192.168.32.100	TCP	60	49268 → 80 [ACK
6	0.000990370	192.168.32.101	192.168.32.100	HTTP	352	GET / HTTP/1.1
7	0.000998344	192.168.32.100	192.168.32.101	TCP	54	80 → 49268 [ACK
8	0.115446290	192.168.32.100	192.168.32.101	TCP	204	80 → 49268 [PSH
9	0.119569095	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK
10	0.119926928	192.168.32.101	192.168.32.100	TCP	60	49268 → 80 [ACK
11	0.120196512	192.168.32.101	192.168.32.100	TCP	60	49268 → 80 [FIN
12	0.120211091	192.168.32.100	192.168.32.101	TCP	54	80 → 49268 [ACK
13	5.244378482	PcsCompu_c7:e1:36	PcsCompu_72:ff:6d	ARP	42	Who has 192.168
14	5.244828025	PcsCompu_72:ff:6d	PcsCompu_c7:e1:36	ARP	60	192.168.32.101

▶ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0  
 ▶ Ethernet II, Src: PcsCompu\_72:ff:6d (08:00:27:72:ff:6d), Dst: PcsCompu\_c7:e1:36 (08:00:27:c7:e1:36)  
   ▶ Destination: PcsCompu\_c7:e1:36 (08:00:27:c7:e1:36)  
   ▶ Source: PcsCompu\_72:ff:6d (08:00:27:72:ff:6d)  
   Type: IPv4 (0x0800)  
 ▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100  
 ▶ Transmission Control Protocol, Src Port: 49268, Dst Port: 80, Seq: 0, Len: 0

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_72:ff:6d	Broadcast	ARP	60	Who has 192.168
2	0.000016213	PcsCompu_c7:e1:36	PcsCompu_72:ff:6d	ARP	42	192.168.32.100
3	0.000372955	192.168.32.101	192.168.32.100	TCP	66	49268 → 80 [SYN
4	0.000396184	192.168.32.100	192.168.32.101	TCP	66	80 → 49268 [SYN
5	0.000813191	192.168.32.101	192.168.32.100	TCP	60	49268 → 80 [ACK
6	0.000990370	192.168.32.101	192.168.32.100	HTTP	352	GET / HTTP/1.1
7	0.000998344	192.168.32.100	192.168.32.101	TCP	54	80 → 49268 [ACK
8	0.115446290	192.168.32.100	192.168.32.101	TCP	204	80 → 49268 [PSH
9	0.119569095	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK
10	0.119926928	192.168.32.101	192.168.32.100	TCP	60	49268 → 80 [ACK
11	0.120196512	192.168.32.101	192.168.32.100	TCP	60	49268 → 80 [FIN
12	0.120211091	192.168.32.100	192.168.32.101	TCP	54	80 → 49268 [ACK
13	5.244378482	PcsCompu_c7:e1:36	PcsCompu_72:ff:6d	ARP	42	Who has 192.168
14	5.244828025	PcsCompu_72:ff:6d	PcsCompu_c7:e1:36	ARP	60	192.168.32.101

▶ Frame 4: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0  
 ▶ Ethernet II, Src: PcsCompu\_c7:e1:36 (08:00:27:c7:e1:36), Dst: PcsCompu\_72:ff:6d (08:00:27:72:ff:6d)  
   ▶ Destination: PcsCompu\_72:ff:6d (08:00:27:72:ff:6d)  
   ▶ Source: PcsCompu\_c7:e1:36 (08:00:27:c7:e1:36)  
   Type: IPv4 (0x0800)  
 ▶ Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101  
 ▶ Transmission Control Protocol, Src Port: 80, Dst Port: 49268, Seq: 0, Ack: 1, Len: 0



In ultimo, voglio vedere cosa cambia ripetendo la procedura di cattura, ma cercando questa volta da WINDOWS di raggiungere il sito HTTPS: internet explorer restituirà un errore riguardante il certificato non sicuro, ignoro l'alert e proseguo. Torno su KALI e verifico Wireshark: la cattura mi evidenzia il successo della connessione (il sito è effettivamente raggiungibile anche se explorer indica mancanza certificati attendibili) passata sempre per SYN/ACK, ma a seguito dell'instaurazione del TCP, noto delle righe aggiuntive che riportano lo scambio di risposte con il protocollo di sicurezza TLSv1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	66	49164 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
2	0.000040446	192.168.32.100	192.168.32.101	TCP	66	443 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=1
3	0.000388040	192.168.32.101	192.168.32.100	TCP	60	49164 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.000699786	192.168.32.101	192.168.32.100	TLSv1	185	Client Hello
5	0.000710882	192.168.32.100	192.168.32.101	TCP	54	443 → 49164 [ACK] Seq=1 Ack=132 Win=64128 Len=0
6	0.109874277	192.168.32.100	192.168.32.101	TLSv1	1368	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.126602675	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
8	0.126662608	192.168.32.100	192.168.32.101	TCP	54	443 → 49164 [ACK] Seq=1315 Ack=266 Win=64128 Len=0
9	0.127887581	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
10	0.146794099	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x5c2e A wpad
11	0.255920500	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x5c2e A wpad
12	0.331632951	192.168.32.101	192.168.32.100	TCP	60	49164 → 443 [ACK] Seq=266 Ack=1374 Win=64324 Len=0
13	0.457757283	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
14	1.207897971	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
15	1.958037720	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
16	2.718109763	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x30c5 A wpad
17	2.818415440	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x30c5 A wpad
18	3.021743408	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>

▶ Frame 4: 185 bytes on wire (1480 bits), 185 bytes captured (1480 bits) on interface eth0, id 0  
▶ Ethernet II, Src: PcsCompu\_72:ff:6d (08:00:27:72:ff:6d), 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: 49164, Dst Port: 443, Seq: 1, Ack: 1, Len: 131  
▶ Transport Layer Security  
▶ TLSv1 Record Layer: Handshake Protocol: Client Hello

Record Layer (tls.record), 131 bytes

Packets: 33 - Displayed: 33 (100.0%)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	66	49164 → 443 [SYN]
2	0.000040446	192.168.32.100	192.168.32.101	TCP	66	443 → 49164 [SYN]
3	0.000388040	192.168.32.101	192.168.32.100	TCP	60	49164 → 443 [ACK]
4	0.000699786	192.168.32.101	192.168.32.100	TLSv1	185	Client Hello
5	0.000710882	192.168.32.100	192.168.32.101	TCP	54	443 → 49164 [ACK]
6	0.109874277	192.168.32.100	192.168.32.101	TLSv1	1368	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.126602675	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
8	0.126662608	192.168.32.100	192.168.32.101	TCP	54	443 → 49164 [ACK]
9	0.127887581	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
10	0.146794099	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x5c2e A wpad
11	0.255920500	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x5c2e A wpad
12	0.331632951	192.168.32.101	192.168.32.100	TCP	60	49164 → 443 [ACK]
13	0.457757283	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
14	1.207897971	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
15	1.958037720	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
16	2.718109763	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x30c5 A wpad
17	2.818415440	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x30c5 A wpad
18	3.021743408	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>

▶ Frame 6: 1368 bytes on wire (10944 bits), 1368 bytes captured (10944 bits) on interface eth0, id 0  
▶ Ethernet II, Src: PcsCompu\_c7:e1:36 (08:00:27:c7:e1:36), Dst: PcsCompu\_72:ff:6d (08:00:27:72:ff:6d)  
▶ Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101  
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 49164, Seq: 1, Ack: 132, Len: 1314  
▶ Transport Layer Security  
▶ TLSv1 Record Layer: Handshake Protocol: Server Hello  
▶ TLSv1 Record Layer: Handshake Protocol: Certificate  
▶ TLSv1 Record Layer: Handshake Protocol: Server Key Exchange  
▶ TLSv1 Record Layer: Handshake Protocol: Server Hello Done



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	66	49164 → 443 [SYN]
2	0.000040446	192.168.32.100	192.168.32.101	TCP	66	443 → 49164 [SYN]
3	0.000388040	192.168.32.101	192.168.32.100	TCP	60	49164 → 443 [ACK]
4	0.000699786	192.168.32.101	192.168.32.100	TLSv1	185	Client Hello
5	0.000710882	192.168.32.100	192.168.32.101	TCP	54	443 → 49164 [ACK]
6	0.109874277	192.168.32.100	192.168.32.101	TLSv1	1368	Server Hello, C
7	0.126602675	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exch
8	0.126662608	192.168.32.100	192.168.32.101	TCP	54	443 → 49164 [ACK]
9	0.127887581	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher S
10	0.146794099	192.168.32.101	224.0.0.252	LLMNR	64	Standard query
11	0.255920500	192.168.32.101	224.0.0.252	LLMNR	64	Standard query
12	0.331632951	192.168.32.101	192.168.32.100	TCP	60	49164 → 443 [ACK]
13	0.457757283	192.168.32.101	192.168.32.255	NBNS	92	Name query NB v
14	1.207897971	192.168.32.101	192.168.32.255	NBNS	92	Name query NB v
15	1.958037720	192.168.32.101	192.168.32.255	NBNS	92	Name query NB v
16	2.718109763	192.168.32.101	224.0.0.252	LLMNR	64	Standard query
17	2.818415440	192.168.32.101	224.0.0.252	LLMNR	64	Standard query
18	3.021743408	192.168.32.101	192.168.32.255	NBNS	92	Name query NB v
▶ Frame 7: 188 bytes on wire (1504 bits), 188 bytes captured (1504 bits) on interface eth0, id 0 ▶ Ethernet II, Src: PcsCompu_72:ff:6d (08:00:27:72:ff:6d), 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: 49164, Dst Port: 443, Seq: 132, Ack: 1315, Len: 134 ▶ Transport Layer Security ▶ TLSv1 Record Layer: Handshake Protocol: Client Key Exchange ▶ TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec ▶ TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message						

## 4. CONCLUSIONI

In sostanza, la differenza visibile con wireshark nello scambio dei pacchetti di dati tra protocollo HTTP e HTTPS è lo step aggiuntivo dello scambio di pacchetti dopo aver instaurato un protocollo di sicurezza TLS con lo scambio delle chiavi di crittografia.