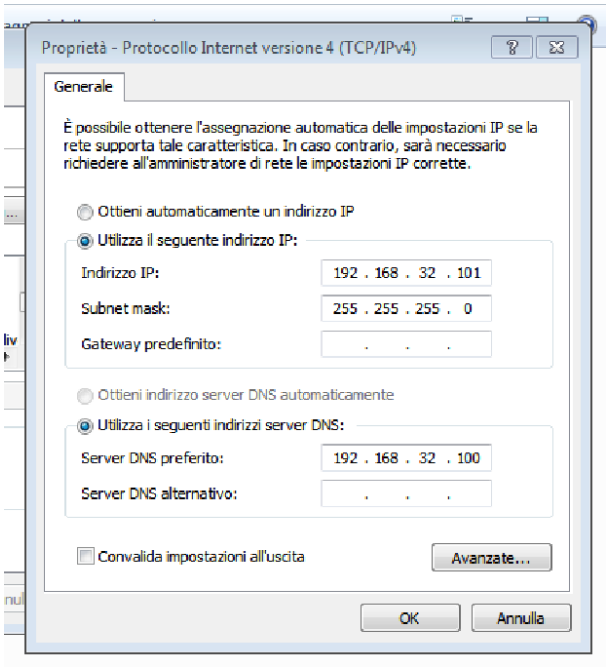
EDOARDO CERIANI

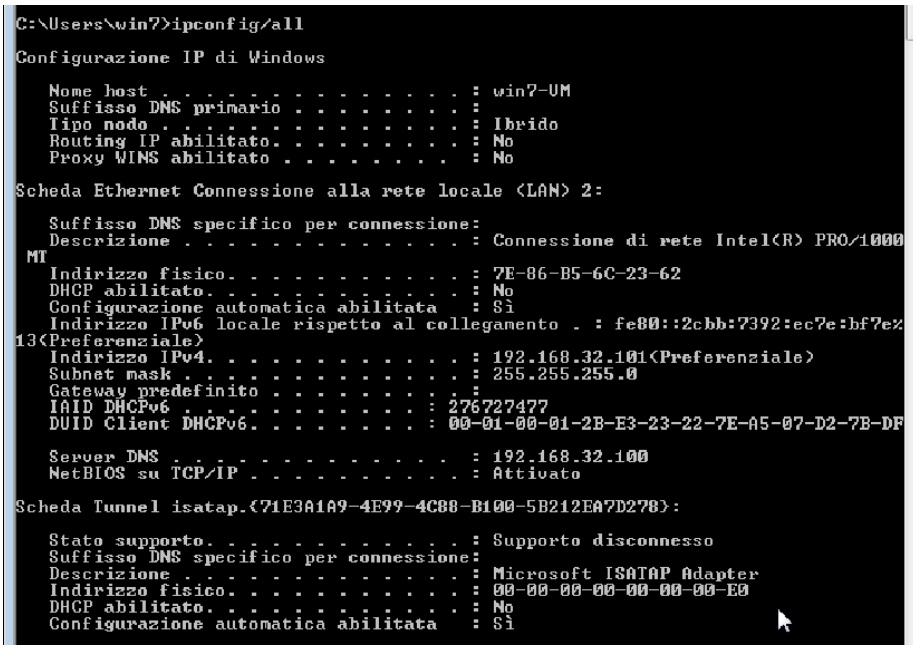
SIMULAZIONE RETE COMPLESSA

La prima cosa da fare per poter svolgere l'esercizio è cambiare gli indirizzi IP delle due macchine, impostandoli come richiesto.

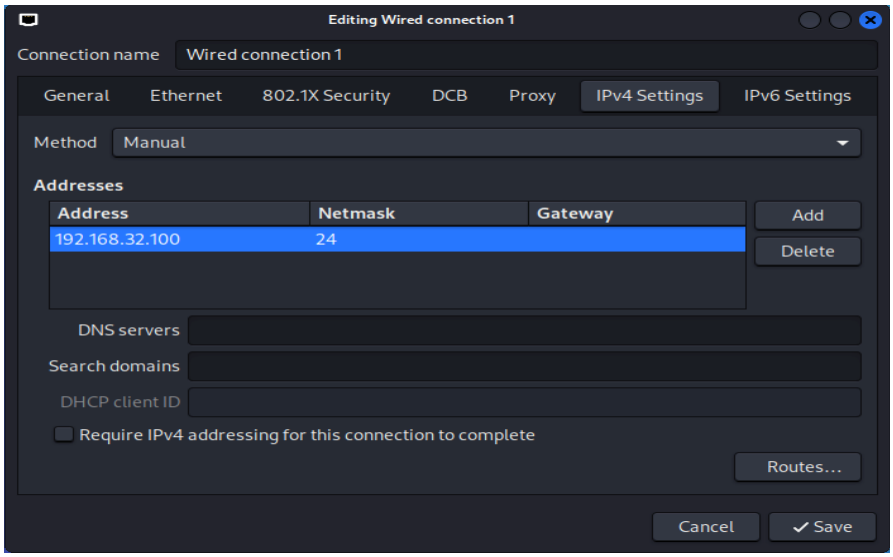
Per fare ciò, avvio la macchina client (Windows 7), mi reco nelle impostazioni della scheda di rete ed imposto l'indirizzo IP del protocollo IPv4 con quello assegnatomi dalla consegna.
Imposto già anche il server DNS con l'indirizzo IP assegnato da consegna alla macchina Kali.



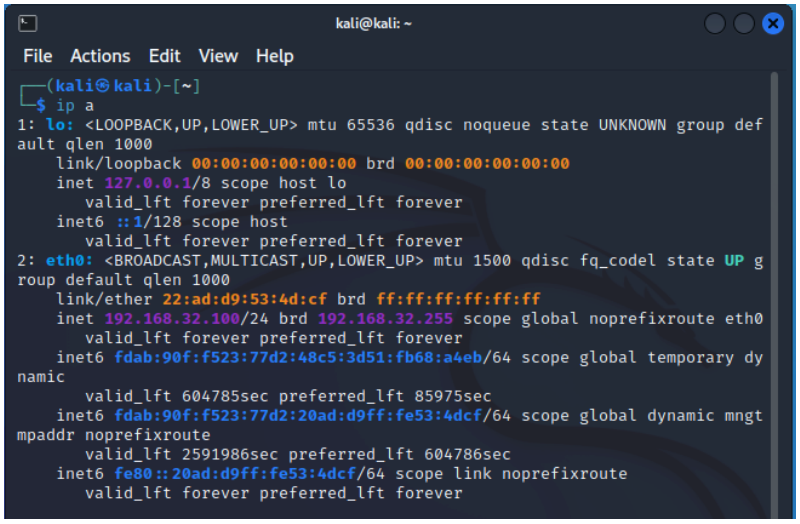
Eseguo anche il comando "ipconfig/all" per verificare l'effettivo cambiamento dell'indirizzo IP



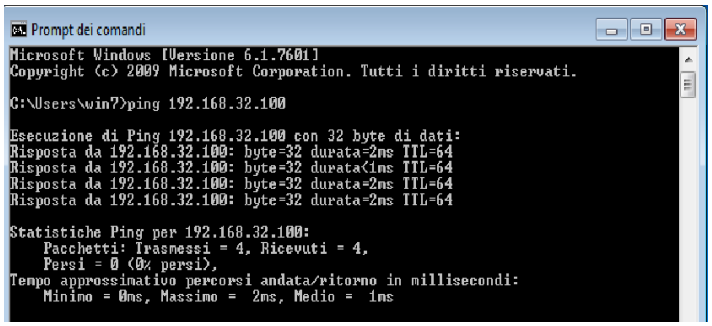
Successivamente, avvio la macchina che servirà da server (Kali Linux) e tramite GUI, nelle impostazioni di rete, imposto l'indirizzo IP come da consegna

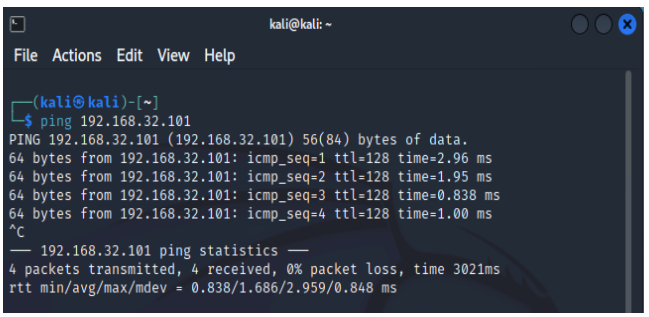


Anche in questo caso, per controllare l'effettivo cambiamento, riavvio la macchina ed eseguo da terminale il comando "ip a"



Per eseguire un ulteriore controllo che entrambi gli indirizzi IP siano stati configurati in modo corretto, eseguo da entrambe le macchine il comando "ping indirizzo_IP_altra_macchina" e vedo che comunicano tra di loro.

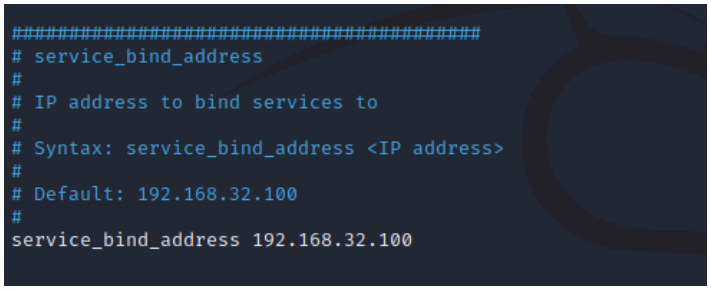


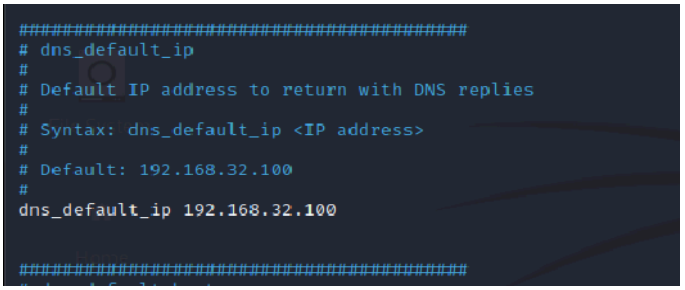


A questo punto, lavorando su Kali, controllo che inetsim sia installato, lancio quindi il comando “sudo inetsim” e vedendo che il server inizia a lavorare, posso passare alla configurazione dei due server necessari per la consegna: quello DNS e quello HTTPS.

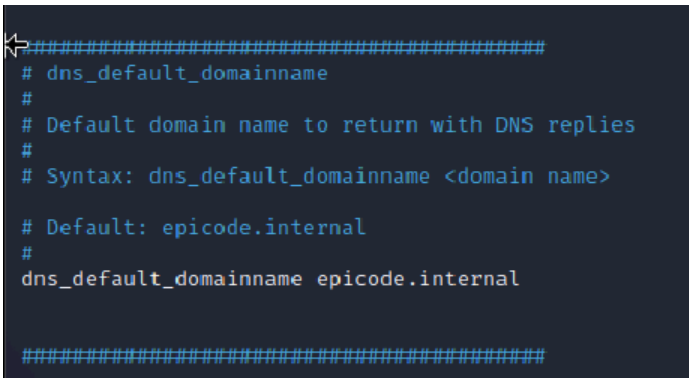
Per configurare il server DNS, eseguo da terminale il comando “sudo nano etc/inetsim/inetsim.conf” così da poter accedere al file di configurazione e poter inserire i parametri richiesti.

Imposto quindi l’indirizzo IP della macchina come indirizzo legato al servizio richiesto e come indirizzo per la risposta del DNS

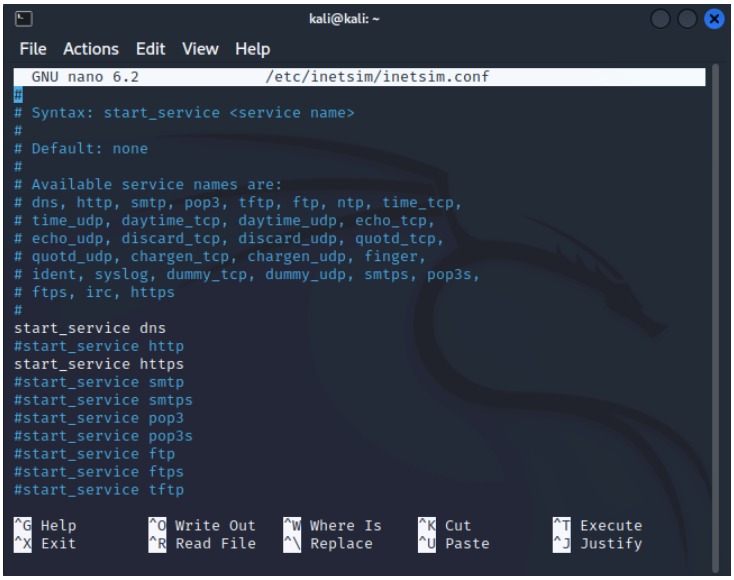




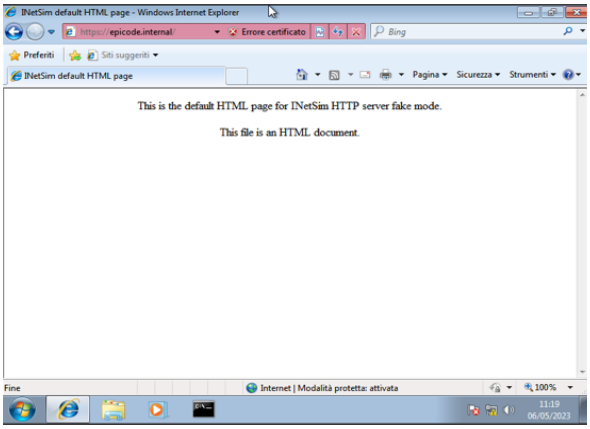
Sempre nelle impostazioni del servizio DNS, imposto il nome del dominio come richiestomi

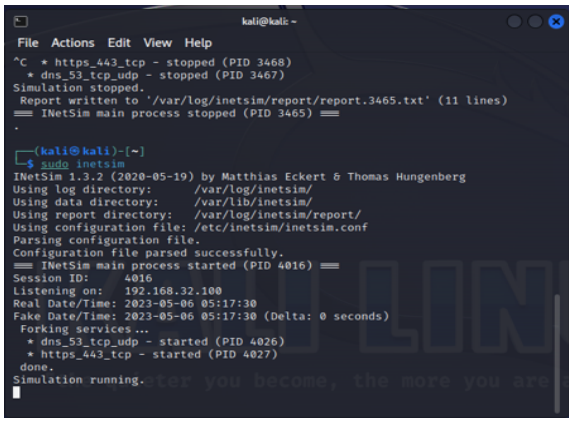


Infine, attivo solo i servizi DNS e HTTPS aggiungendo il simbolo “#” prima di ogni riga riguardante gli altri servizi.



Eseguo quindi su Kali il comando “sudo inetsim” per avviare i due server e contemporaneamente sulla macchina Win7 apro Internet Explorer per cercare la pagina https://epicode.internal.





A questo punto avvio Wireshark per intercettare la comunicazione, premo il pulsante a forma di pinna blu in alto a sinistra per avviare il programma e dopo aver ottenuto quanto richiesto, fermo e salvo i risultati per poterli analizzare.

Dal file ottenuto si può innanzitutto notare che all’inizio del traffico (frame 4 e 5) viene creata una richiesta ARP da parte della macchina client per sapere quale indirizzo MAC ha la macchina server.

```
   4 2.798265251  7e:86:b5:6c:23:62  Broadcast          ARP      42 Who has 192.168.32.100? Tell 192.168.32.101
   5 2.798278210  22:ad:d9:53:4d:cf  7e:86:b5:6c:23:62  ARP      42 192.168.32.100 is at 22:ad:d9:53:4d:cf
```

Si può notare anche che gli indirizzi MAC di sorgente e destinazione sono visibili nelle corrispettive colonne solo in questi due frame di ARP, negli altri invece bisogna analizzare ogni singolo frame per individuarli (esempio in foto).

```
▾ Ethernet II, Src: 7e:86:b5:6c:23:62 (7e:86:b5:6c:23:62), Dst: 22:ad:d9:53:4d:cf (22:ad:d9:53:4d:cf)
  ▸ Destination: 22:ad:d9:53:4d:cf (22:ad:d9:53:4d:cf)
  ▸ Source: 7e:86:b5:6c:23:62 (7e:86:b5:6c:23:62)
```

Per analizzare la richiesta HTTPS imposto come filtro “tcp.port==443” sapendo che la porta 443 corrisponde al servizio HTTPS.

tcp.port==443

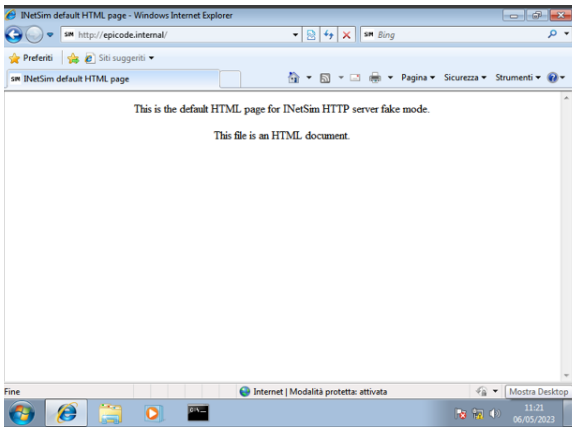
No.	Time	Source	Destination	Protocol	Length	Info
6	2.799512710	192.168.32.101	192.168.32.100	TCP	66	49175 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
7	2.799529626	192.168.32.100	192.168.32.101	TCP	66	443 → 49175 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PE
8	2.799967293	192.168.32.101	192.168.32.100	TCP	54	49175 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
9	2.801005835	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
10	2.801013835	192.168.32.100	192.168.32.101	TCP	54	443 → 49175 [ACK] Seq=1 Ack=162 Win=64128 Len=0
11	2.803635918	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	2.831287043	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Messag
13	2.831303043	192.168.32.100	192.168.32.101	TCP	54	443 → 49175 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
14	2.831533293	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
19	3.035812460	192.168.32.100	192.168.32.101	TCP	113	[TCP Retransmission] 443 → 49175 [PSH, ACK] Seq=1320 Ack=296 Win=64
20	3.037024835	192.168.32.101	192.168.32.100	TCP	66	49175 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len=0 SLE=1320 SRE=137
45	9.201287505	192.168.32.101	192.168.32.100	TLSv1	395	Application Data
46	9.201360338	192.168.32.100	192.168.32.101	TCP	54	443 → 49175 [ACK] Seq=1379 Ack=637 Win=64128 Len=0
47	9.210202255	192.168.32.100	192.168.32.101	TLSv1	235	Application Data
48	9.211450880	192.168.32.100	192.168.32.101	TLSv1	384	Application Data, Encrypted Alert
49	9.212070713	192.168.32.101	192.168.32.100	TCP	54	49175 → 443 [ACK] Seq=637 Ack=1891 Win=65700 Len=0
51	9.218170088	192.168.32.101	192.168.32.100	TCP	54	49175 → 443 [FIN, ACK] Seq=637 Ack=1891 Win=65700 Len=0
52	9.218181421	192.168.32.100	192.168.32.101	TCP	54	443 → 49175 [ACK] Seq=1891 Ack=638 Win=64128 Len=0

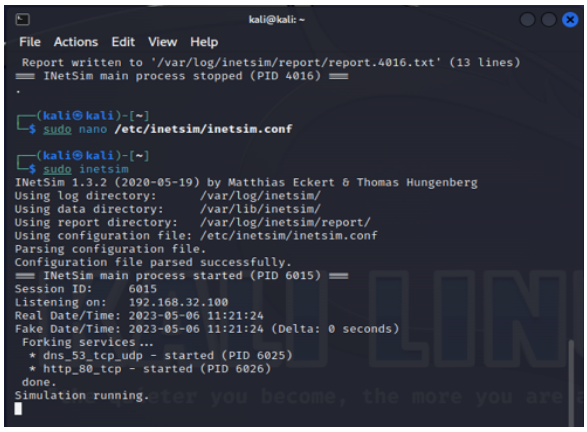
Il frame 19 penso corrisponda alla richiesta di sicurezza del web browser sulla macchina client, in quanto riscontrava un errore di certificato e necessitava dell’approvazione dell’utente per raggiungere la pagina web desiderata.

Aprendo per esempio il frame 46 della richiesta HTTPS, si possono notare i MAC address di sorgente e destinazione e le informazioni contenute nell'header del TCP come i numeri della porta di sorgente e destinazione, l'ACK e la window size.

```
▸ Frame 46: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface eth0, id 0
▸ Ethernet II, Src: 22:ad:d9:53:4d:cf (22:ad:d9:53:4d:cf), Dst: 7e:86:b5:6c:23:62 (7e:86:b5:6c:23:62)
▸ Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 49175, Seq: 1379, Ack: 637, Len: 0
    Source Port: 443
    Destination Port: 49175
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1379    (relative sequence number)
    Sequence Number (raw): 3523895366
    [Next Sequence Number: 1379    (relative sequence number)]
    Acknowledgment Number: 637    (relative ack number)
    Acknowledgment number (raw): 2433610858
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window: 501
    [Calculated window size: 64128]
    [Window size scaling factor: 128]
    Checksum: 0x7229 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
```

Per eseguire invece la richiesta su server HTTP, torno nel file di configurazione di inetsim, rendo attivo il server HTTP e inattivo quello HTTPS, faccio partire nuovamente inetsim e sulla macchina Win7 apro Internet Explorer per cercare la pagina http://epicode.internal.





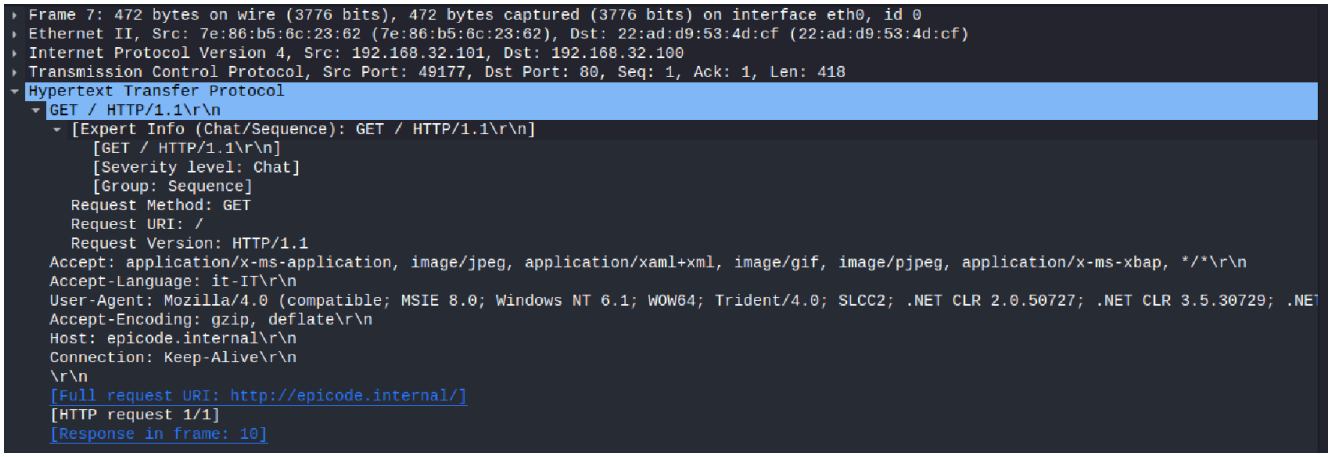
Avvio nuovamente Wireshark per intercettare la comunicazione ed ottenere i risultati per poter analizzare quanto ottenuto e cercare eventuali differenze con i risultati catturati in precedenza con il server HTTPS.

No.	Time	Source	Destination	Protocol	Leng	Info
1	0.000000000	fe80::20ad:d9ff:fe53:4dcf	ff02::1:ffd9:8364	ICMPv6	86	Neighbor Solicitation for fe80::5ce9:1eff:fed9:8364 from 22:ad:d9
2	1.023112292	fe80::20ad:d9ff:fe53:4dcf	ff02::1:ffd9:8364	ICMPv6	86	Neighbor Solicitation for fe80::5ce9:1eff:fed9:8364 from 22:ad:d9
3	2.051197501	fe80::20ad:d9ff:fe53:4dcf	ff02::1:ffd9:8364	ICMPv6	86	Neighbor Solicitation for fe80::5ce9:1eff:fed9:8364 from 22:ad:d9
4	2.152221668	192.168.32.101	192.168.32.100	TCP	66	49177 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
5	2.152245709	192.168.32.100	192.168.32.101	TCP	66	80 → 49177 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PE
6	2.152768001	192.168.32.101	192.168.32.100	TCP	54	49177 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
7	2.161390293	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
8	2.161427293	192.168.32.100	192.168.32.101	TCP	54	80 → 49177 [ACK] Seq=1 Ack=419 Win=64128 Len=0
9	2.166530834	192.168.32.100	192.168.32.101	TCP	204	80 → 49177 [PSH, ACK] Seq=1 Ack=419 Win=64128 Len=150 [TCP segment
10	2.167415209	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
11	2.167890334	192.168.32.101	192.168.32.100	TCP	54	49177 → 80 [ACK] Seq=419 Ack=410 Win=65292 Len=0
12	2.172468918	192.168.32.101	192.168.32.100	TCP	54	49177 → 80 [FIN, ACK] Seq=419 Ack=410 Win=65292 Len=0
13	2.172475751	192.168.32.100	192.168.32.101	TCP	54	80 → 49177 [ACK] Seq=410 Ack=420 Win=64128 Len=0
14	3.074890626	fe80::20ad:d9ff:fe53:4dcf	ff02::1:ffd9:8364	ICMPv6	86	Neighbor Solicitation for fe80::5ce9:1eff:fed9:8364 from 22:ad:d9
15	4.098430252	fe80::20ad:d9ff:fe53:4dcf	ff02::1:ffd9:8364	ICMPv6	86	Neighbor Solicitation for fe80::5ce9:1eff:fed9:8364 from 22:ad:d9
16	5.122531419	fe80::20ad:d9ff:fe53:4dcf	ff02::1:ffd9:8364	ICMPv6	86	Neighbor Solicitation for fe80::5ce9:1eff:fed9:8364 from 22:ad:d9

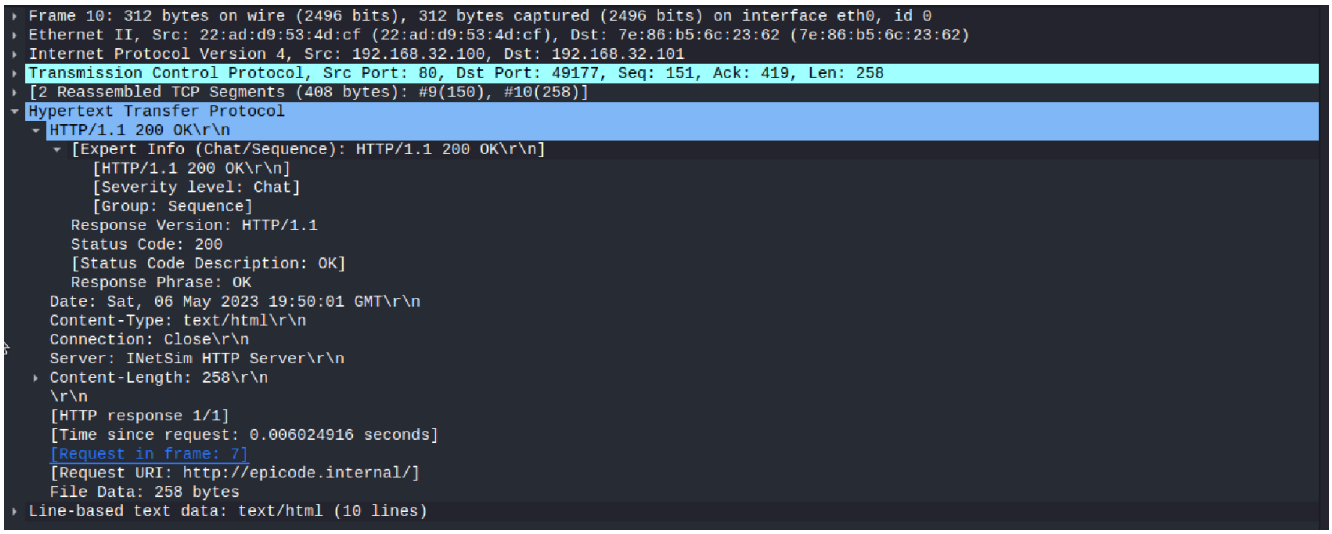
La prima cosa che si può notare è che il numero di frame è molto minore rispetto alla richiesta HTTPS.

Anche in questo tipo di richiesta, MAC address di sorgente e destinazione sono visibili solo all'interno di ogni singolo frame.

In questo caso il protocollo HTTP compare nella colonna specifica, aprendo infatti il frame 7 si può riscontrare la presenza dell'Hypertext Transef Protocol che riporta informazioni riguardanti la richiesta da parte del client.
Qui si possono notare, per esempio, il metodo (GET), l'host (epicode.internal), l'user-agent e il tipo di connessione.



Aprendo invece il frame 10, è riportata la risposta da parte del server, in cui si può evidenziare, tra le altre cose, il codice di stato 200 che rappresenta uno stato di successo (OK).



Queste ultime sono tutte informazioni che nel protocollo HTTPS non sono visibili in quanto è un protocollo cifrato, e quindi più sicuro, che nasconde queste informazioni così da evitare che un potenziale attaccante possa intercettare la comunicazione e quindi avere accesso ad informazioni sensibili.