PROGETTO FINALE M1

1. Introduzione

Questo report documenta l'analisi del traffico di rete in un ambiente di laboratorio virtuale, simulando un'architettura client-server. Un client **Windows** con indirizzo IP 192.168.32.101 richiede l'accesso a una risorsa web ospitata su un server **Kali Linux** all'indirizzo IP 192.168.32.100, risolvibile tramite il nome di dominio epicode.internal

L'obiettivo dell'esperimento è intercettare il traffico di rete utilizzando **Wireshark**, evidenziando i protocolli di origine e destinazione in server HTTPS. Successivamente, si ripeterà la stessa analisi sostituendo il server HTTPS con uno HTTP, confrontando le differenze tra i due tipi di traffico.

Attraverso questa simulazione, si valuteranno le implicazioni di sicurezza legate all'uso del protocollo HTTP rispetto a HTTPS, evidenziando le differenze nella trasmissione dei dati e la loro potenziale esposizione a intercettazioni di rete.

2. Requisiti

- Client: Windows (192.168.32.101)
- Server: Kali Linux (192.168.32.100)
- Hostname: epicode.internal
- Servizi attivi: DNS per la risoluzione dei nomi, Inetsim per la simulazione dei server HTTPS e HTTP

3. Configurazione di Kali linux

Innanzitutto cominciamo dalla configurazione dell'indirizzo IP, attraverso il comando:
sudo nano /etc/network/interfaces
sotto **auto eth0** troviamo l'indirizzo IP e il **gateway** nel nostro caso scriviamo address 192.168.32.100/24 e **gateway** 192.168.32.1

```
auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1
```

.
Netsim è un programma default di **Kali linux** che serve per emulare servizi, nel nostro caso lo utilizzeremo per emulare i servizi HTTP e HTTPS
Quindi per impostarlo facciamo il comando
sudo nano /net/inetsim/inetsim.conf
Andiamo su **start_service_http** togliamo il **#** all'inizio, e poi lo rifacciamo con il servizio HTTPS e modifichiamo anche il **service_bind_address** con quello della nostra **Kali** quindi 192.168.32.100 il bind address serve per specificare che indirizzo IP vogliamo che prenda la simulazione e infine utilizzeremo **Wireshark** per la cattura dei pacchetti

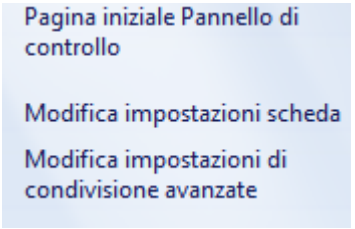
Poi per la simulazione del server DNS ho utilizza **DNSchef** che è preinstallato su **Kali** e ho scritto il comando:

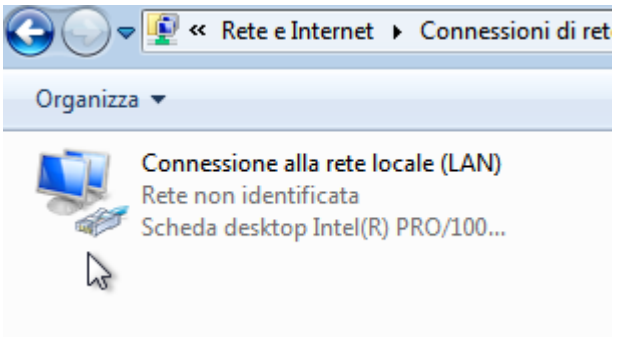
dnschef –fakeip=192.168.32.100 –fakedomains=epicode.internal –nameserver=192.168.32.100 –interface=192.168.32.100

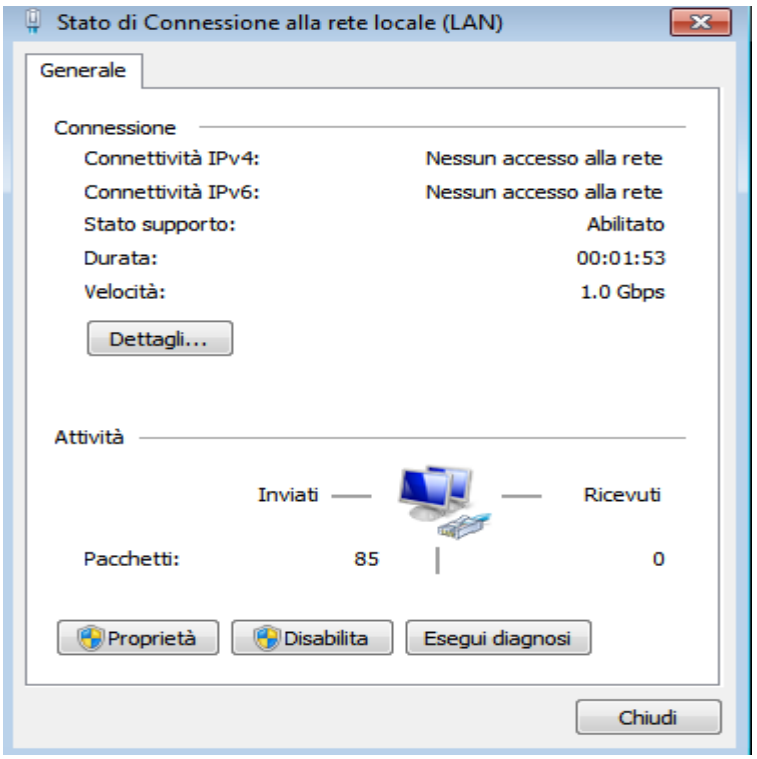
```
#start_service dns
start_service http
#start_service https
#start_service smtp
#start_service smtps
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service ftps
#start_service tftp
#start_service irc
#start_service ntp
#start_service finger
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#start_service dummy_udp


###########################################
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 192.168.32.100
#
 service_bind_address 192.168.32.100
```

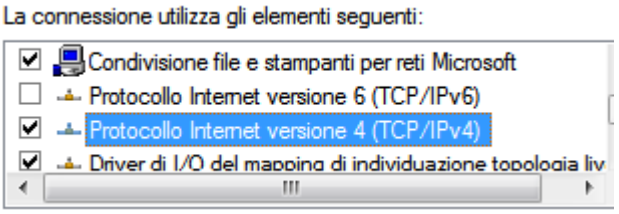
```
┌──(kali㉿kali)-[~]
└─$ dnschef --fakeip=192.168.32.100 --fakedomains=epicode.internal --nameservers 192.168.32.100 --interface 192.168.32.100
/usr/bin/dnschef:453: SyntaxWarning: invalid escape sequence '\/'
  header += "      / _` | '_ \/ __|/ __| '_ \ / _ \  _|\n"
/usr/bin/dnschef:454: SyntaxWarning: invalid escape sequence '\_'
  header += "     | (_| | | | \__ \ (__| | | |  __/ |  \n"
/usr/bin/dnschef:455: SyntaxWarning: invalid escape sequence '\_'
  header += "      \__,_|_| |_|___/\___|_| |_|\___|_|  \n"
                                    _                 __
        | | version 0.4  | |       / _|
      __| |_ __  ___ ___| |__   ___| |_
     / _` | '_ \/ __|/ __| '_ \ / _ \  _|
    | (_| | | | \__ \ (__| | | |  __/ |
     \__,_|_| |_|___/\___|_| |_|\___|_|
                iphelix@thesprawl.org

(16:25:51) [*] DNSChef started on interface: 192.168.32.100
(16:25:51) [*] Using the following nameservers: 192.168.32.100
(16:25:51) [*] Cooking A replies to point to 192.168.32.100 matching: epicode.internal
```

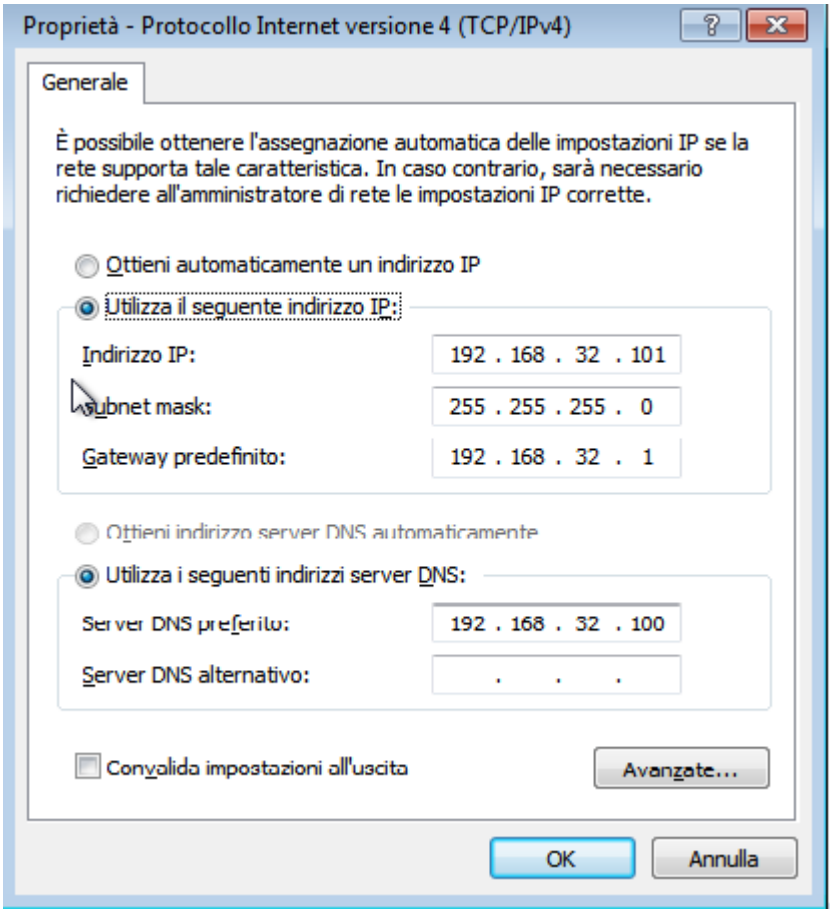
4. **Configurazione windows 7**
   Per la configurazione dell'indirizzo IP di **Windows 7** dobbiamo andare:
   Pannello di controllo >Centro connessioni di rete e condivisione > modifica impostazione scheda > Connessione alla rete loca > Proprietà > Protocollo internet versione 4 > utilizza il seguente indirizzo IP
   - ➔ Indirizzo IP: 192.168.32.101
   - ➔ Subnetmask: 255.255.255.0
   - ➔ Gateway predefinito: 192.168.32.1
   - ➔ Server DNS preferito: 192.168.32.1





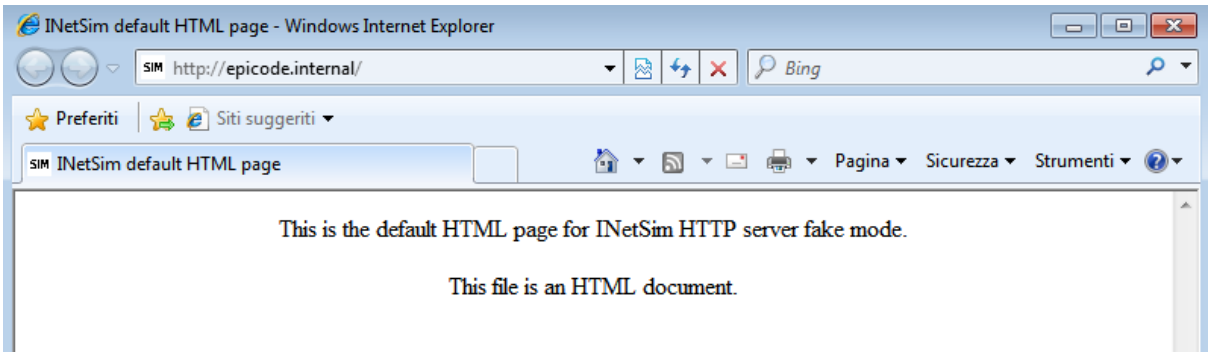

Stato di Connessione alla rete locale (LAN)
Generale
Connessione
Connettività IPv4: Nessun accesso alla rete
Connettività IPv6: Nessun accesso alla rete
Stato supporto: Abilitato
Durata: 00:01:53
Velocità: 1.0 Gbps
Dettagli...
Attività
Inviati
Ricevuti
Pacchetti: 85 0
Proprietà
Disabilita
Esegui diagnosi
Chiudi

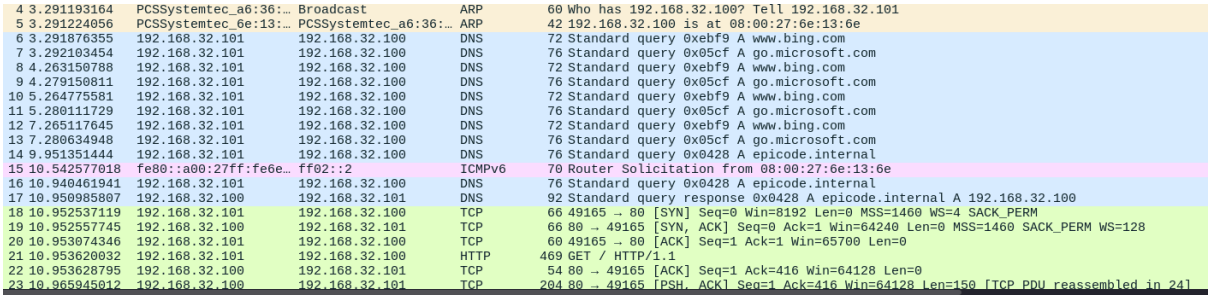

La connessione utilizza gli elementi seguenti:
Condivisione file e stampanti per reti Microsoft
Protocollo Internet versione 6 (TCP/IPv6)
Protocollo Internet versione 4 (TCP/IPv4)
Driver di I/O del mapping di individuazione topologia liv

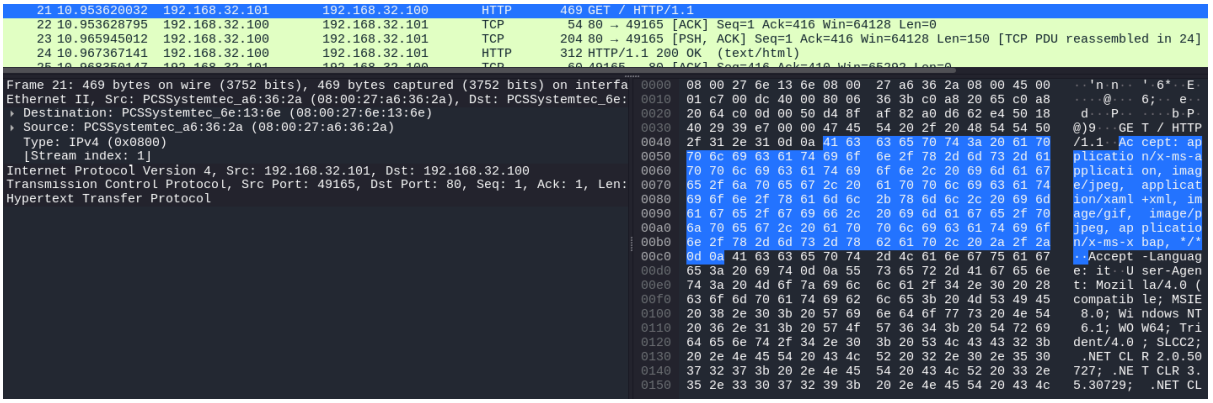


5. Analisi HTTP

Per prima cosa su kali eseguiamo **DNSchef** poi **Inetsim** in questo ordine perché se eseguiamo prima **Inetsim** e poi **DNSchef** non funzionerebbe, e per ultimo apriamo **Wireshark**, poi su **Windows** apriamo http://epicode.internal e adesso andiamo ad analizzare i pacchetti con **Wireshark** notiamo che dal protocollo HTTP possiamo trovare l'indirizzo MAC sorgente e destinatario, anche che il payload è in **plaintext** quindi l'informazione potrebbe essere a rischio di un attacco **man-in the-middle**

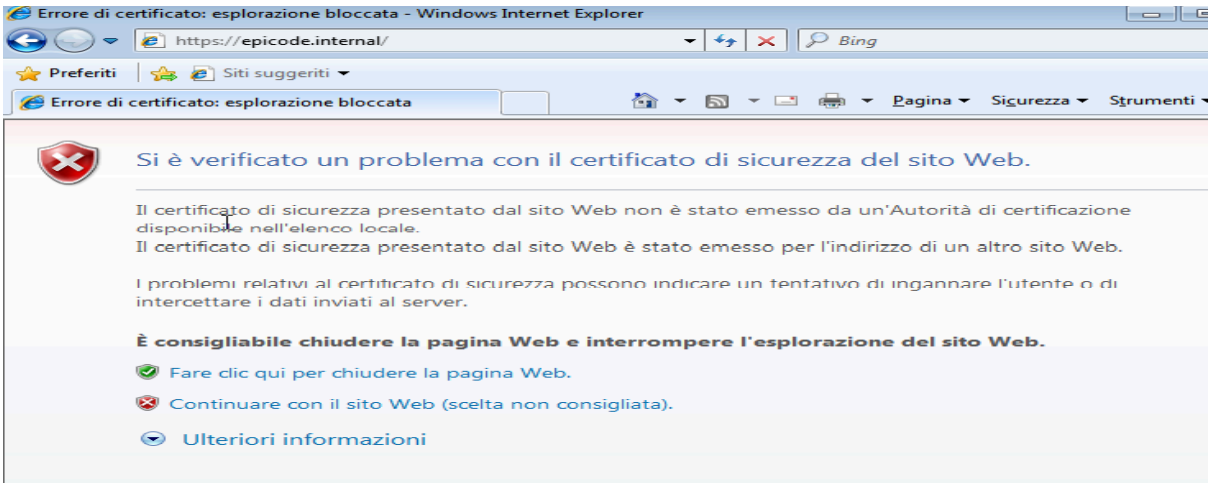


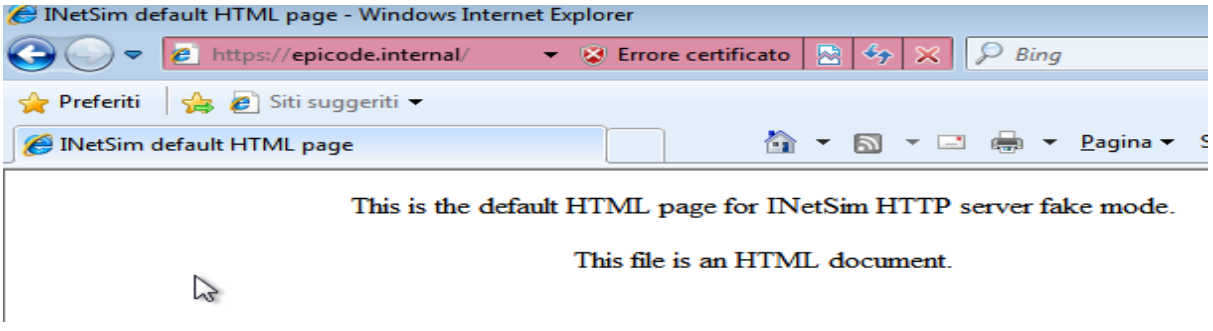


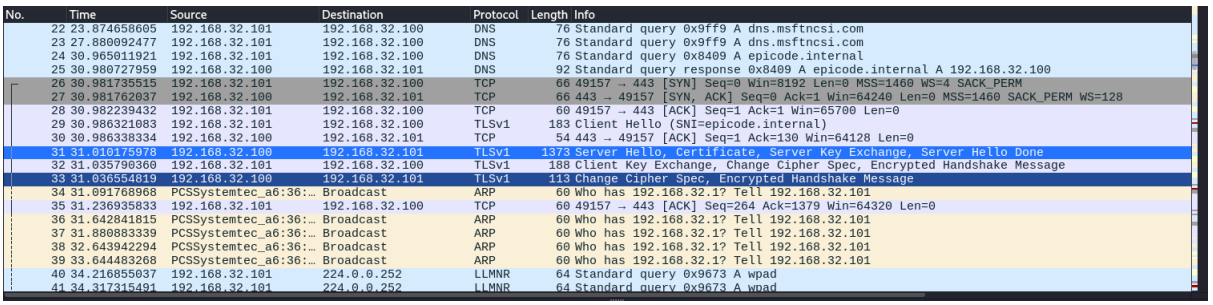


6. Analisi HTTPS

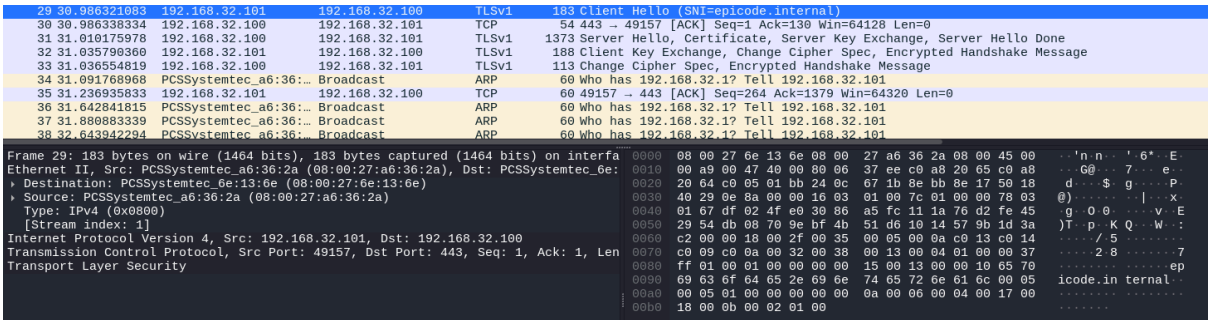
Adesso facciamo lo stesso ma con il server HTTPS, al momento di andare su https://epicode.internal/ ci apparirà un messaggio di sicurezza nel **browser** il quale dovremmo ignorare, e poi andiamo ad analizzare i pacchetti con **Wireshark**, e invece con il servizio HTTPS notiamo che comunque riusciamo lo stesso a vedere l'indirizzo MAC sorgente e destinatario ma che tutto il payload è in **ciphertext**







No.	Time	Source	Destination	Protocol	Length	Info
22	23.874658605	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x9ff9 A dns.msftncsi.com
23	27.880092477	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x9ff9 A dns.msftncsi.com
24	30.965011921	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x8409 A epicode.internal
25	30.980727959	192.168.32.100	192.168.32.101	DNS	92	Standard query response 0x8409 A epicode.internal A 192.168.32.100
26	30.981735515	192.168.32.101	192.168.32.100	TCP	66	49157 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
27	30.981762037	192.168.32.100	192.168.32.101	TCP	66	443 → 49157 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
28	30.982239432	192.168.32.101	192.168.32.100	TCP	60	49157 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
29	30.986321083	192.168.32.101	192.168.32.100	TLSv1	183	Client Hello (SNI=epicode.internal)
30	30.986338334	192.168.32.100	192.168.32.101	TCP	54	443 → 49157 [ACK] Seq=1 Ack=130 Win=64128 Len=0
31	31.010175978	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
32	31.035790360	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
33	31.036554819	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
34	31.091768968	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
35	31.236935833	192.168.32.101	192.168.32.100	TCP	60	49157 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
36	31.642841815	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
37	31.880883339	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
38	32.643942294	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
39	33.644483268	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
40	34.216855037	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x9673 A wpad
41	34.317315491	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x9673 A wpad



29	30.986321083	192.168.32.101	192.168.32.100	TLSv1	183	Client Hello (SNI=epicode.internal)
30	30.986338334	192.168.32.100	192.168.32.101	TCP	54	443 → 49157 [ACK] Seq=1 Ack=130 Win=64128 Len=0
31	31.010175978	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
32	31.035790360	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
33	31.036554819	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
34	31.091768968	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
35	31.236935833	192.168.32.101	192.168.32.100	TCP	60	49157 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
36	31.642841815	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
37	31.880883339	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
38	32.643942294	PCSSystemtec_a6:36:…	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101

```
Frame 29: 183 bytes on wire (1464 bits), 183 bytes captured (1464 bits) on interfa
Ethernet II, Src: PCSSystemtec_a6:36:2a (08:00:27:a6:36:2a), Dst: PCSSystemtec_6e:
▸ Destination: PCSSystemtec_6e:13:6e (08:00:27:6e:13:6e)
▸ Source: PCSSystemtec_a6:36:2a (08:00:27:a6:36:2a)
  Type: IPv4 (0x0800)
  [Stream index: 1]
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 49157, Dst Port: 443, Seq: 1, Ack: 1, Len
Transport Layer Security
```

```
0000  08 00 27 6e 13 6e 08 00  27 a6 36 2a 08 00 45 00   ··'n·n·· '·6*··E·
0010  00 a9 00 47 40 00 80 06  37 ee c0 a8 20 65 c0 a8   ···G@··· 7··· e··
0020  20 64 c0 05 01 bb 24 0c  67 1b 8e bb 8e 17 50 18    d····$· g·····P·
0030  40 29 0e 8a 00 00 16 03  01 00 7c 01 00 00 78 03   @)······ ··|···x·
0040  01 67 df 02 4f e0 30 86  a5 fc 11 1a 76 d2 fe 45   ·g··O·0· ····v··E
0050  29 54 db 08 70 9e bf 4b  51 d6 10 14 57 9b 1d 3a   )T··p··K Q···W··:
0060  c2 00 00 18 00 2f 00 35  00 05 00 0a c0 13 c0 14   ·····/·5 ········
0070  c0 09 c0 0a 00 32 00 38  00 13 00 04 01 00 00 37   ·····2·8 ·······7
0080  ff 01 00 01 00 00 00 00  15 00 13 00 00 10 65 70   ········ ······ep
0090  69 63 6f 64 65 2e 69 6e  74 65 72 6e 61 6c 00 05   icode.in ternal··
00a0  00 05 01 00 00 00 00 00  0a 00 06 00 04 00 17 00   ········ ········
00b0  18 00 0b 00 02 01 00                               ·······
```

7. **Confronto**
    - Gli indirizzi MAC sorgente e destinatario sono visibili
    - Il payload è cifrato (ciphertext) quindi i dati sono protetti

Protocolli	**HTTP**	**HTTPS**
Visibilità MAC	Sì	Sì
Payload leggibile	Sì(plaintext)	No(ciphertext)
Sicurezza	Vulnerabile a MITM	Protetto da crittografia
Protocolli utilizzati	ARP, DNS, TCP	ARP, DNS, TCP + TLS

Entrambi I protocolli utilizzano ARP (Address Resolution Protocol) per associare un indirizzo IP a un indirizzo MAC e DNS per la risoluzione dei nomi. Il protocollo TCP è usato per la trasmissione, ma HTTPS aggiunge la crittografia con TLS, proteggendo i dati dopo il 3-way-handshake

8. **Conclusioni**

Dall'analisi emerge chiaramente che l'uso di HTTPS fornisce una protezione significativa rispetto a HTTP. L'uso del protocollo HTTP espone i dati in chiaro, rendendoli vulnerabili ad attacchi di intercettazione. HTTPS, invece, garantisce la cifratura dei dati, impedendo a un attaccante di leggere le informazioni trasmesse sulla rete.

Pertanto, si raccomanda di evitare HTTP per la trasmissione di dati sensibili e di adottare sempre HTTPS per garantire la sicurezza delle comunicazioni online.