

M1_W4D4



Esercizio

Traccia e requisiti

Nell'esercizio di oggi metteremo insieme le competenze acquisite finora.
Lo studente verrà valutato sulla base della risoluzione al problema seguente.

Requisiti e servizi:

- Kali Linux ☐ IP 192.168.32.100
- Windows 7 ☐ IP 192.168.32.101
- HTTPS 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 (Windows 7) richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100 (Kali).

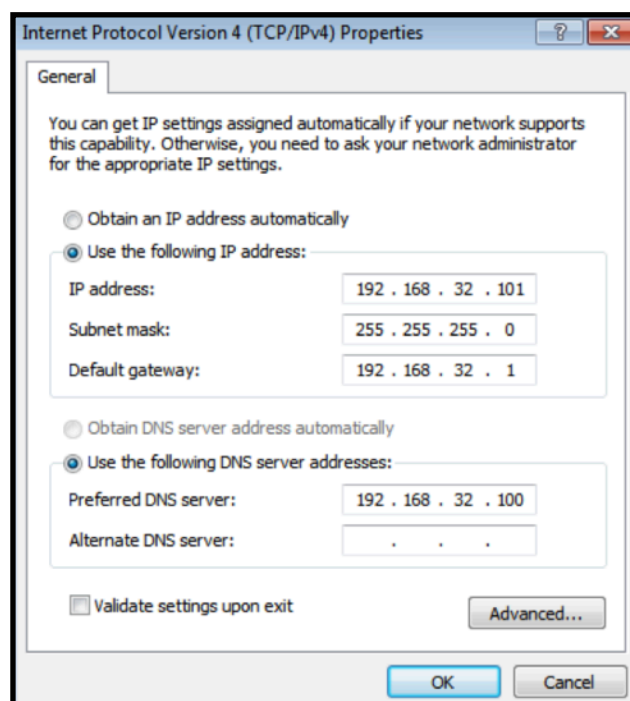
Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS.

Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

Configurazione Indirizzo IP su Kali Linux

```
(salvatoremorale@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255
    inet6 fe80::6c04:b7ff:fee8:7f17 prefixlen 64 scopeid 0x20<link>
    ether 6e:04:b7:e8:7f:17 txqueuelen 1000 (Ethernet)
    RX packets 8 bytes 340 (340.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 13 bytes 2208 (2.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Configurazione Indirizzo IP su Windows 7



Configurazione servizi **HTTPS** e **DNS** su InetSim.

Per il **DNS** è richiesta l'associazione tra "epicode.internal" e l'IP di Kali Linux (192.168.32.100)

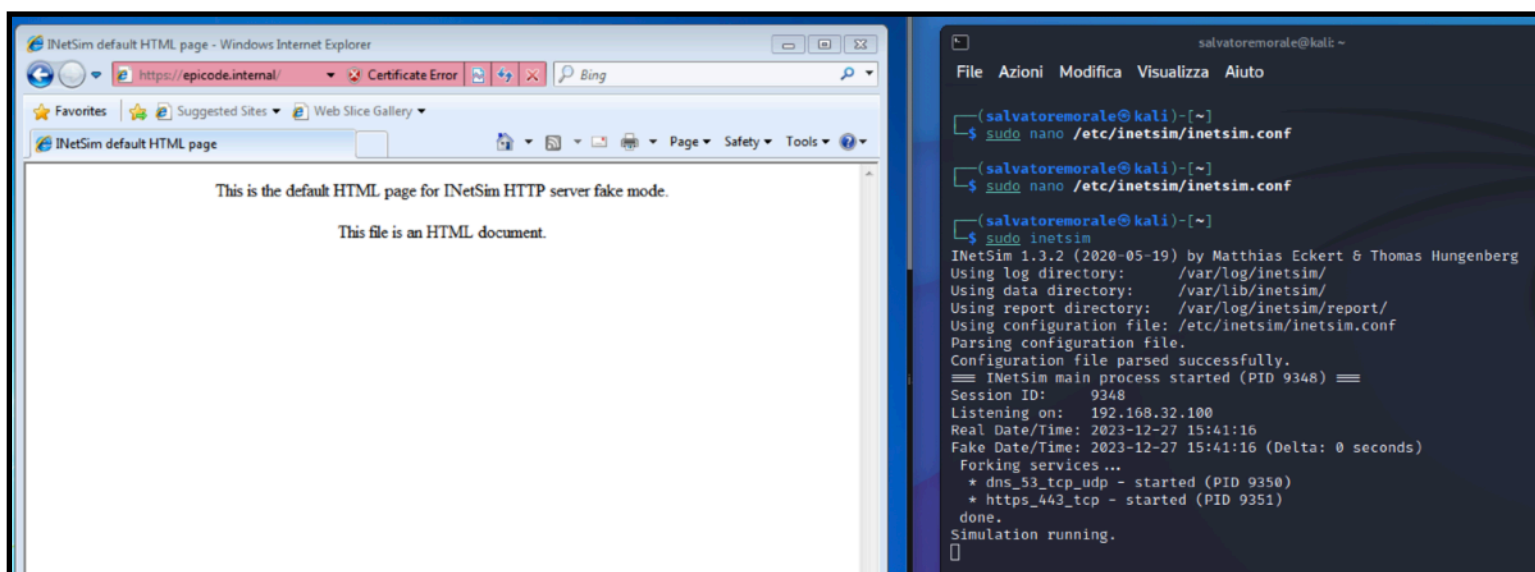
```
start_service dns
#start_service http
start_service https
start_service smtp
# start_service smtps
# start_service pop3
# start_service pop3s
# start_service ftp
# start_service ftps
# start_service tftp
```

```
GNU nano 6.2 /etc/inetsim/inetsim.conf *
#
# Default domain name to return with DNS replies
#
# Syntax: dns_default_domainname <domain name>
#
# Default: inetsim.org
#
dns_default_domainname epicode.internal

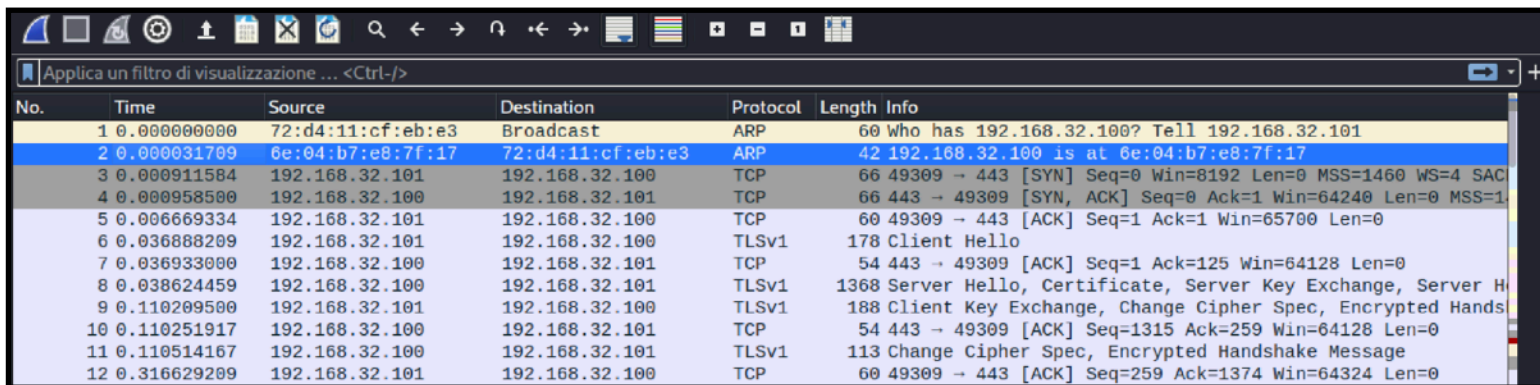
#####
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
dns_static epicode.internal 192.168.32.100
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
```

Abilitazione servizi tramite InetSim e test connessione con il client Windows 7:

Test connettività al servizio **HTTPS** verso https://epicode.internal



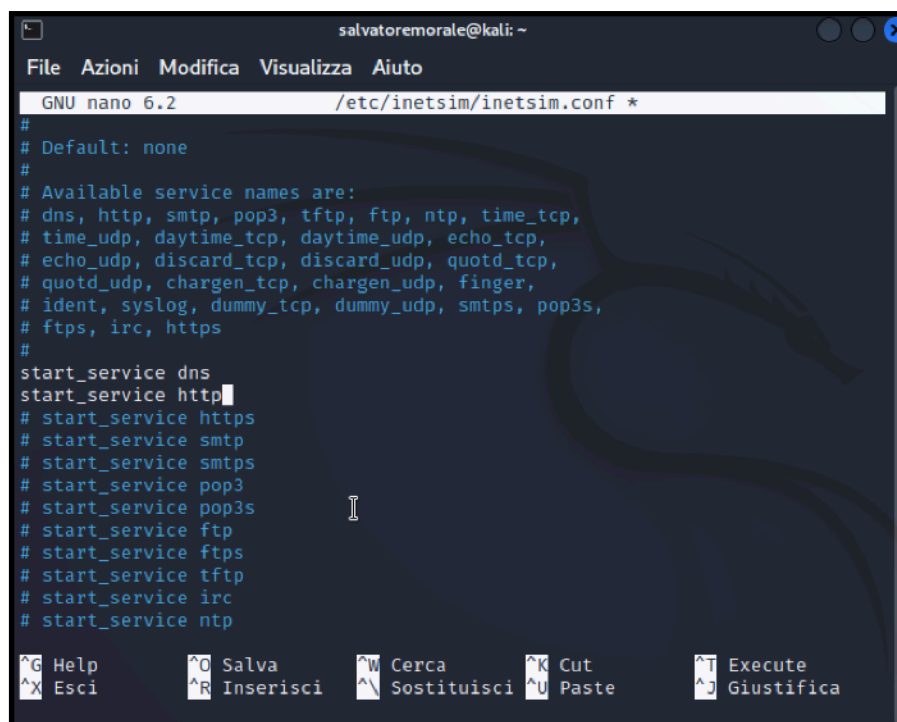
Prima cattura con Wireshark (servizio HTTPS)



Applica un filtro di visualizzazione ... <Ctrl-/>

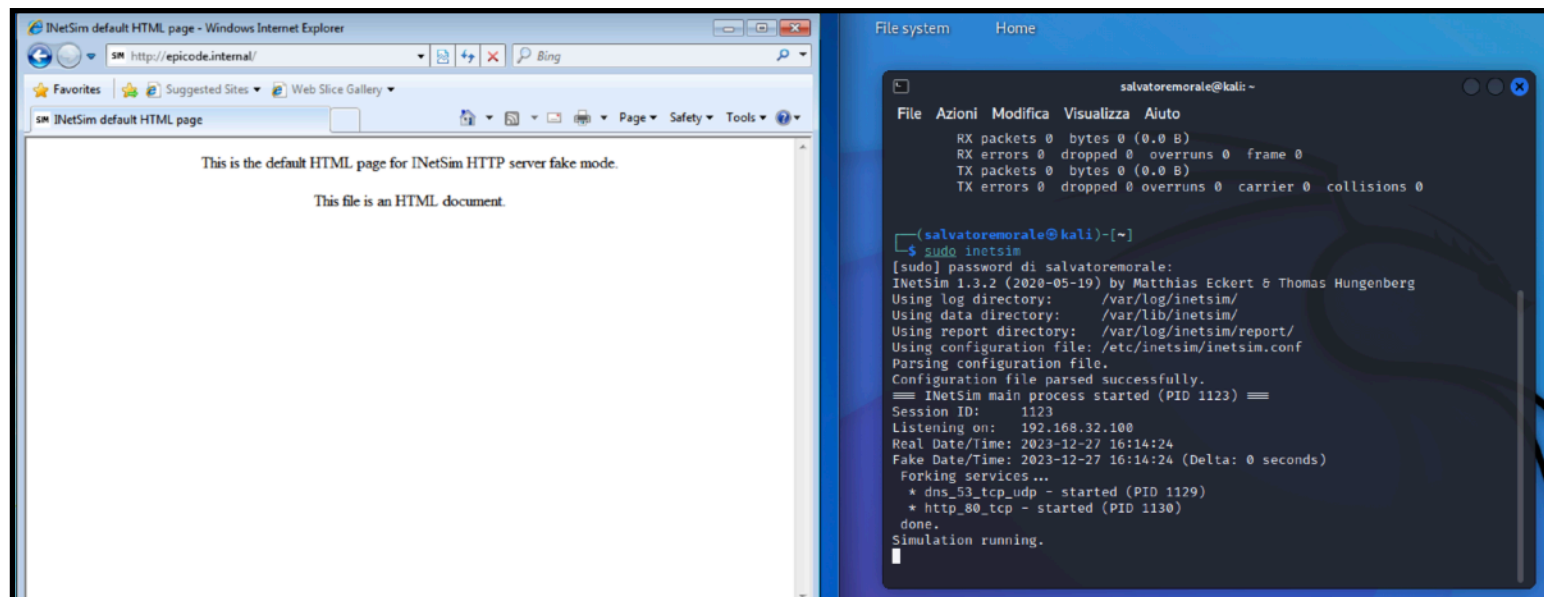
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	72:d4:11:cf:eb:e3	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000031709	6e:04:b7:e8:7f:17	72:d4:11:cf:eb:e3	ARP	42	192.168.32.100 is at 6e:04:b7:e8:7f:17
3	0.000911584	192.168.32.101	192.168.32.100	TCP	66	49309 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SAC
4	0.000958500	192.168.32.100	192.168.32.101	TCP	66	443 → 49309 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
5	0.006669334	192.168.32.101	192.168.32.100	TCP	60	49309 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.036888209	192.168.32.101	192.168.32.100	TLSv1	178	Client Hello
7	0.036933000	192.168.32.100	192.168.32.101	TCP	54	443 → 49309 [ACK] Seq=1 Ack=125 Win=64128 Len=0
8	0.038624459	192.168.32.100	192.168.32.101	TLSv1	1368	Server Hello, Certificate, Server Key Exchange, Server H
9	0.110209500	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Hands
10	0.110251917	192.168.32.100	192.168.32.101	TCP	54	443 → 49309 [ACK] Seq=1315 Ack=259 Win=64128 Len=0
11	0.110514167	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12	0.316629209	192.168.32.101	192.168.32.100	TCP	60	49309 → 443 [ACK] Seq=259 Ack=1374 Win=64324 Len=0

Disabilitazione servizio HTTPS e configurazione del servizio HTTP su InetSim



```
salvatoremorale@kali: ~  
File Azioni Modifica Visualizza Aiuto  
GNU nano 6.2 /etc/inetSim/inetSim.conf *  
#  
# 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  
# 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  
  
^G Help      ^O Salva    ^W Cerca    ^K Cut      ^T Execute  
^X Esci      ^R Inserisci ^\ Sostituisci ^U Paste    ^J Giustifica
```

Test connettività al servizio **HTTP** verso <http://epicode.internal>



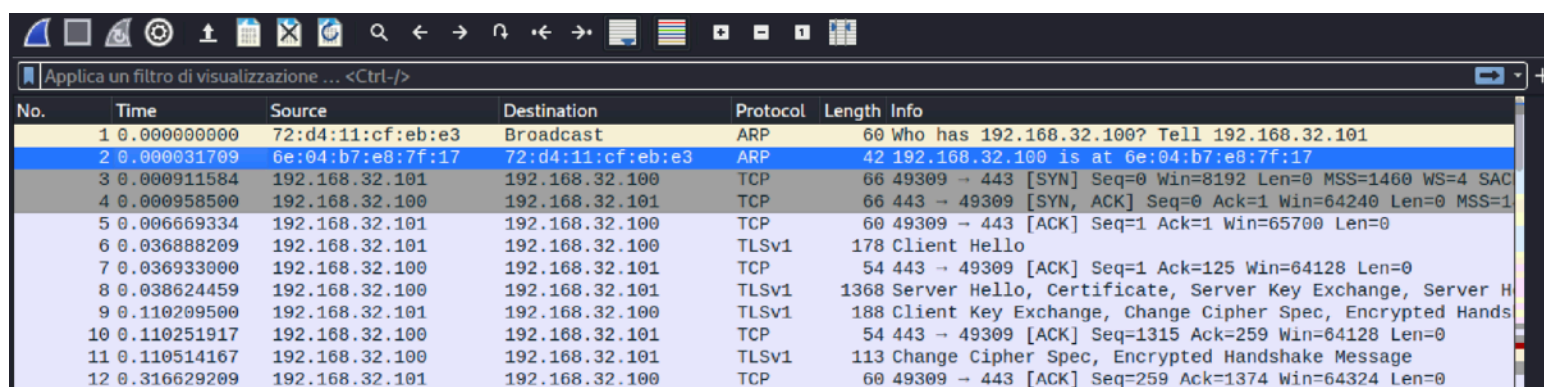
Seconda cattura con Wireshark (servizio **HTTP**)

The screenshot shows a Wireshark network traffic capture. The top toolbar includes buttons for applying filters, saving, and other functions. The main display area shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets are filtered by 'Applica un filtro di visualizzazione ... <Ctrl-/>'. The first packet is ignored. The second packet is a TCP SYN packet from 192.168.32.101 to 192.168.32.100. The third packet is a TCP SYN, ACK packet from 192.168.32.100 to 192.168.32.101. The fourth packet is a TCP ACK packet from 192.168.32.101 to 192.168.32.100. The fifth packet is an HTTP GET request from 192.168.32.101 to 192.168.32.100. The sixth packet is a TCP ACK packet from 192.168.32.100 to 192.168.32.101. The seventh packet is a TCP PSH, ACK packet from 192.168.32.100 to 192.168.32.101. The eighth packet is an HTTP 200 OK response from 192.168.32.100 to 192.168.32.101. The ninth packet is a TCP ACK packet from 192.168.32.101 to 192.168.32.100. The tenth packet is a TCP FIN, ACK packet from 192.168.32.101 to 192.168.32.100. The eleventh packet is a TCP ACK packet from 192.168.32.100 to 192.168.32.101.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000				50	<Ignored>
2	1.925462125	192.168.32.101	192.168.32.100	TCP	66	49299 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK
3	1.925495292	192.168.32.100	192.168.32.101	TCP	66	80 → 49299 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=14
4	1.926178417	192.168.32.101	192.168.32.100	TCP	60	49299 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
5	1.927691542	192.168.32.101	192.168.32.100	HTTP	361	GET / HTTP/1.1
6	1.927710834	192.168.32.100	192.168.32.101	TCP	54	80 → 49299 [ACK] Seq=1 Ack=308 Win=64128 Len=0
7	1.936166084	192.168.32.100	192.168.32.101	TCP	204	80 → 49299 [PSH, ACK] Seq=1 Ack=308 Win=64128 Len=150 [T
8	1.937421084	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
9	1.938336792	192.168.32.101	192.168.32.100	TCP	60	49299 → 80 [ACK] Seq=308 Ack=410 Win=65292 Len=0
10	1.944301667	192.168.32.101	192.168.32.100	TCP	60	49299 → 80 [FIN, ACK] Seq=308 Ack=410 Win=65292 Len=0
11	1.944320959	192.168.32.100	192.168.32.101	TCP	54	80 → 49299 [ACK] Seq=410 Ack=309 Win=64128 Len=0

Macro differenze tra le due catture:

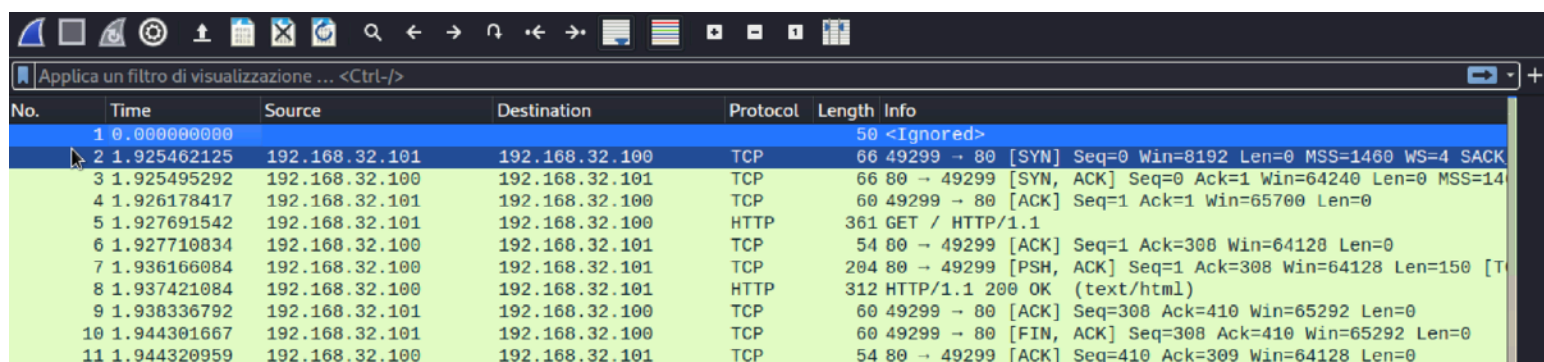
HTTPS



Applica un filtro di visualizzazione ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	72:d4:11:cf:eb:e3	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000031709	6e:04:b7:e8:7f:17	72:d4:11:cf:eb:e3	ARP	42	192.168.32.100 is at 6e:04:b7:e8:7f:17
3	0.000911584	192.168.32.101	192.168.32.100	TCP	66	49309 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK
4	0.000958500	192.168.32.100	192.168.32.101	TCP	66	443 → 49309 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
5	0.006669334	192.168.32.101	192.168.32.100	TCP	60	49309 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.036888209	192.168.32.101	192.168.32.100	TLSv1	178	Client Hello
7	0.036933000	192.168.32.100	192.168.32.101	TCP	54	443 → 49309 [ACK] Seq=1 Ack=125 Win=64128 Len=0
8	0.038624459	192.168.32.100	192.168.32.101	TLSv1	1368	Server Hello, Certificate, Server Key Exchange, Server H
9	0.110209500	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Hands
10	0.110251917	192.168.32.100	192.168.32.101	TCP	54	443 → 49309 [ACK] Seq=1315 Ack=259 Win=64128 Len=0
11	0.110514167	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12	0.316629209	192.168.32.101	192.168.32.100	TCP	60	49309 → 443 [ACK] Seq=259 Ack=1374 Win=64324 Len=0

HTTP



Applica un filtro di visualizzazione ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000				50	<Ignored>
2	1.925462125	192.168.32.101	192.168.32.100	TCP	66	49299 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK
3	1.925495292	192.168.32.100	192.168.32.101	TCP	66	80 → 49299 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=14
4	1.926178417	192.168.32.101	192.168.32.100	TCP	60	49299 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
5	1.927691542	192.168.32.101	192.168.32.100	HTTP	361	GET / HTTP/1.1
6	1.927710834	192.168.32.100	192.168.32.101	TCP	54	80 → 49299 [ACK] Seq=1 Ack=308 Win=64128 Len=0
7	1.936166084	192.168.32.100	192.168.32.101	TCP	204	80 → 49299 [PSH, ACK] Seq=1 Ack=308 Win=64128 Len=150 [T
8	1.937421084	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
9	1.938336792	192.168.32.101	192.168.32.100	TCP	60	49299 → 80 [ACK] Seq=308 Ack=410 Win=65292 Len=0
10	1.944301667	192.168.32.101	192.168.32.100	TCP	60	49299 → 80 [FIN, ACK] Seq=308 Ack=410 Win=65292 Len=0
11	1.944320959	192.168.32.100	192.168.32.101	TCP	54	80 → 49299 [ACK] Seq=410 Ack=309 Win=64128 Len=0

La disparità più significativa tra le due acquisizioni risiede nell'effettuato tentativo di istituire un canale cifrato su **HTTPS**. È possibile esaminare la cattura Wireshark focalizzandosi sul protocollo TLS (Transport Layer Security) per monitorare il susseguirsi dei pacchetti.

TLS fornisce un canale sicuro, crittografando i dati durante la trasmissione e impedendo a terze parti di intercettare o manipolare le informazioni.

Questo protocollo non è invece riscontrabile nella cattura **HTTP**, la quale, come noto, opera senza cifrare i dati. Nel caso del pacchetto dati **HTTP**, il contenuto della richiesta è visibile in maniera non crittografata, mentre nel contesto di **HTTPS**, il protocollo TLS istituisce un tunnel cifrato prima della trasmissione dei flussi di dati.