ESERCIZIO FINALE M1 GALLO COSIMO PIO

Traccia: Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 Windows) 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

Come primo passo avviare inetsim tramite il comando sudo inetsim:

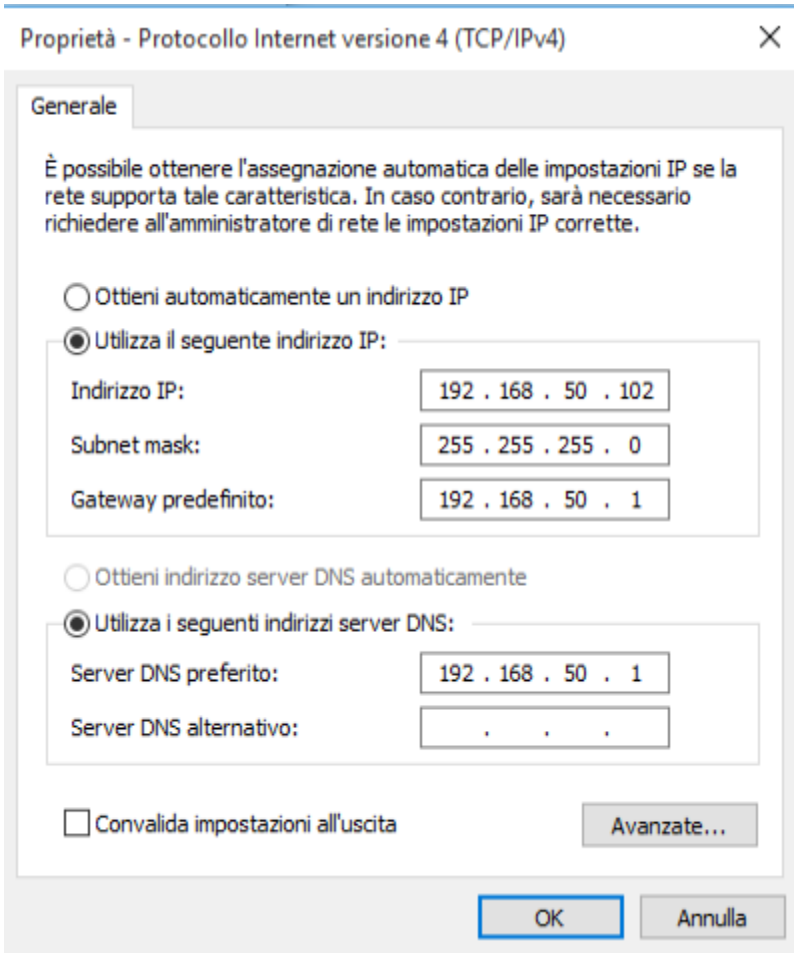
```
┌──(kali㉿kali)-[~]
└─$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 40738) ===
Session ID:     40738
Listening on:   0.0.0.0
Real Date/Time: 2025-07-18 14:40:14
Fake Date/Time: 2025-07-18 14:40:14 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 40742)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at /usr/share/perl5/INetSim/DNS.pm line 69.
  * https_443_tcp - started (PID 40744)
  * syslog_514_udp - started (PID 40764)
  * time_37_tcp - started (PID 40765)
  * http_80_tcp - started (PID 40743)
  * smtp_25_tcp - started (PID 40745)
  * tftp_69_udp - started (PID 40759)
  * pop3s_995_tcp - started (PID 40748)
  * irc_6667_tcp - started (PID 40760)
  * smtps_465_tcp - started (PID 40746)
  * finger_79_tcp - started (PID 40762)
  * time_37_udp - started (PID 40766)
  * ftp_21_tcp - started (PID 40749)
  * ident_113_tcp - started (PID 40763)
  * ntp_123_udp - started (PID 40761)
  * ftps_990_tcp - started (PID 40758)
  * daytime_13_udp - started (PID 40768)
  * pop3_110_tcp - started (PID 40747)
  * daytime_13_tcp - started (PID 40767)
  * echo_7_tcp - started (PID 40769)
  * discard_9_udp - started (PID 40772)
  * quotd_17_tcp - started (PID 40773)
  * echo_7_udp - started (PID 40770)
  * quotd_17_udp - started (PID 40774)
  * chargen_19_udp - started (PID 40776)
  * discard_9_tcp - started (PID 40771)
  * dummy_1_udp - started (PID 40778)
  * chargen_19_tcp - started (PID 40775)
  * dummy_1_tcp - started (PID 40777)
 done.
Simulation running.
```

N.B. in lezione abbiamo constatato che c'è un problema con il DNS. Come possiamo notare dà l'errore in quanto non trova main_loop. Abbiamo proseguito l'esercizio senza DNS.

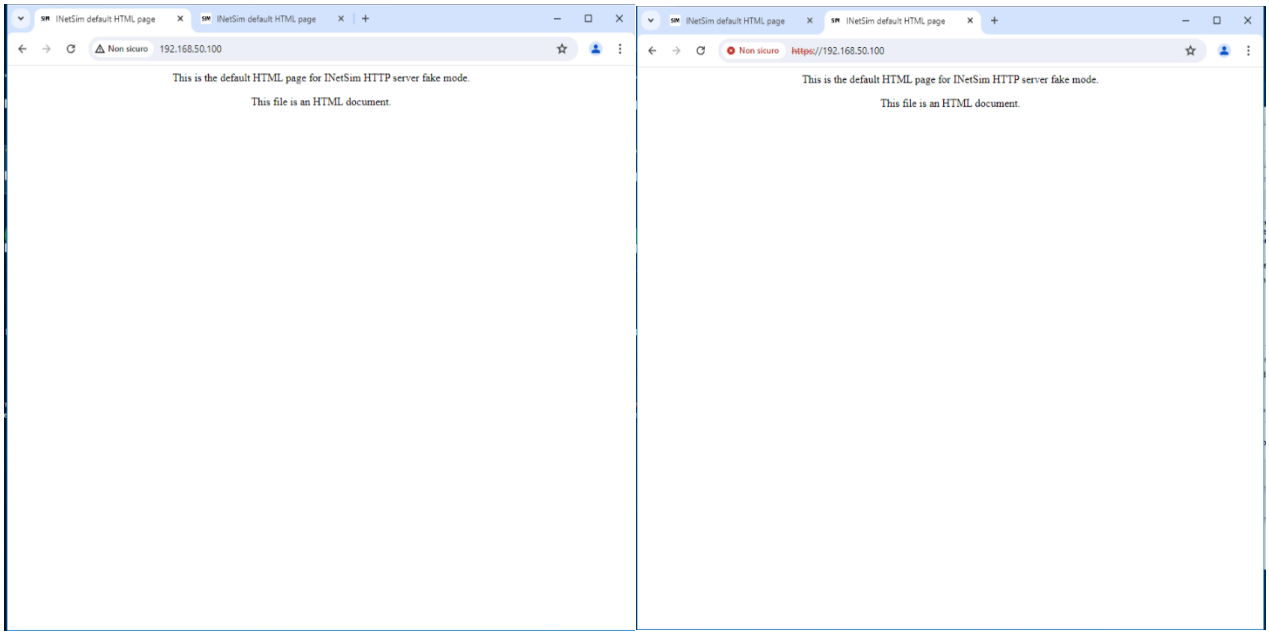
Impostiamo Inetsim in ascolto su tutte le interfacce:

```
##########################################
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 0.0.0.0
```

Procediamo con la configurazione del DNS di windows:

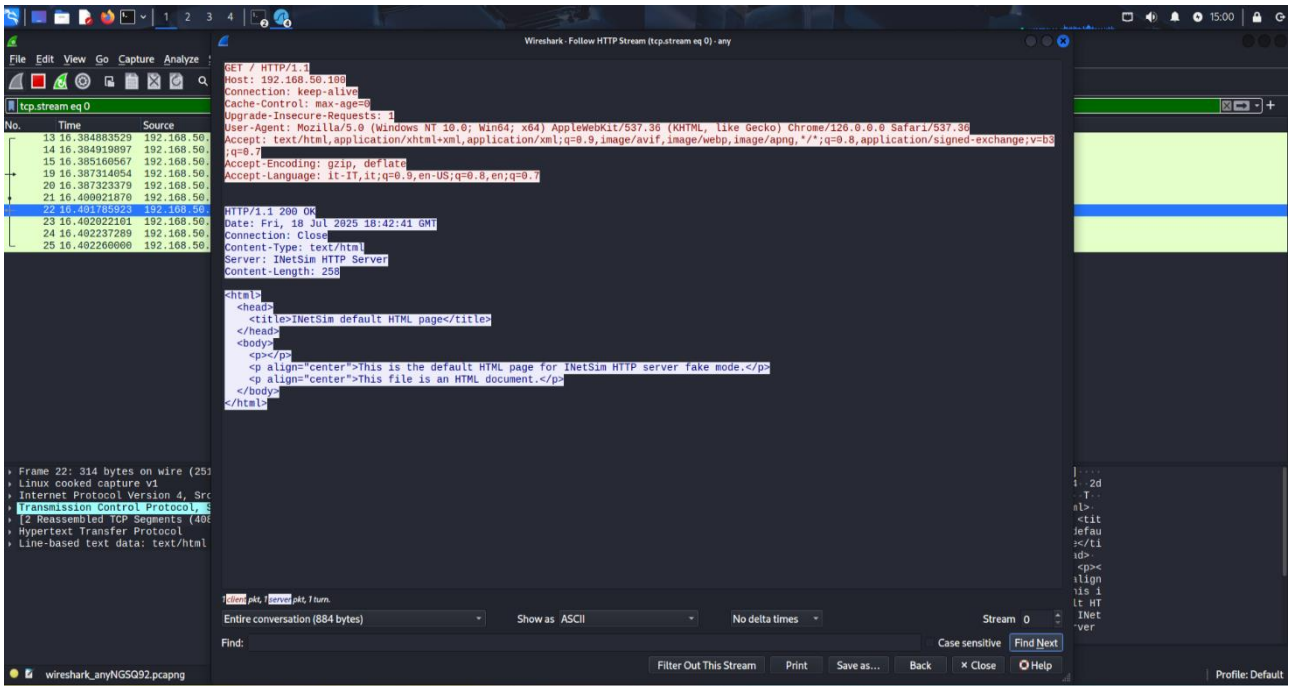


Infine, ricerchiamo tramite IP del DNS su Chrome prima in HTTP poi in HTTPS:

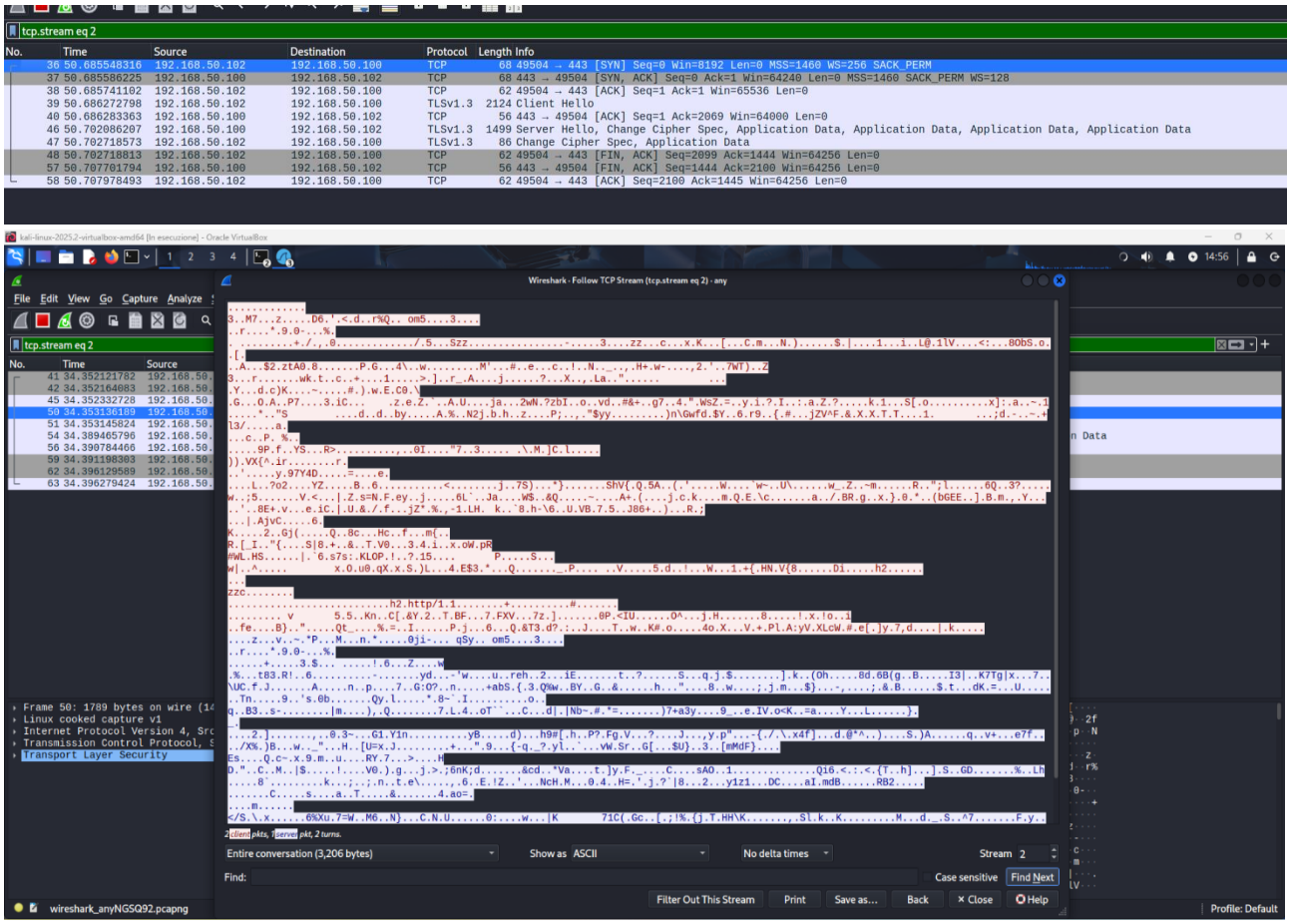


Vedendo che tutto funziona ci mettiamo in ascolto tramite Wireshark per intercettare i segnali in entrata ed uscita dalle varie macchine:

Segnale HTTP come possiamo notare è tutto in chiaro e riusciamo a vedere chi manda il messaggio (host 192.168.50.100 Kali) e chi riceve il segnale (user-agent: mozilla/5.0 (windows Nt 10.0; Win 64; x64)



Al contrario il segnale HTTPS è un segnale criptato per tanto non riusciremo a vedere il contenuto dei vari segnali:


tcp.stream eq 2
No. Time Source Destination Protocol Length Info
36 50.685548316 192.168.50.102 192.168.50.100 TCP 68 49504 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
37 50.685586225 192.168.50.100 192.168.50.102 TCP 68 443 → 49504 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
38 50.685741102 192.168.50.102 192.168.50.100 TCP 62 49504 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
39 50.686272798 192.168.50.102 192.168.50.100 TLSv1.3 2124 Client Hello
40 50.686283363 192.168.50.100 192.168.50.102 TCP 56 443 → 49504 [ACK] Seq=1 Ack=2069 Win=64000 Len=0
46 50.702086207 192.168.50.100 192.168.50.102 TLSv1.3 1499 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
47 50.702718573 192.168.50.102 192.168.50.100 TLSv1.3 86 Change Cipher Spec, Application Data
48 50.702718813 192.168.50.102 192.168.50.100 TCP 62 49504 → 443 [FIN, ACK] Seq=2099 Ack=1444 Win=64256 Len=0
57 50.707701794 192.168.50.100 192.168.50.102 TCP 56 443 → 49504 [FIN, ACK] Seq=1444 Ack=2100 Win=64256 Len=0
58 50.707978493 192.168.50.102 192.168.50.100 TCP 62 49504 → 443 [ACK] Seq=2100 Ack=1445 Win=64256 Len=0

kali-linux-2025.2-virtualbox-amd64 [In esecuzione] - Oracle VirtualBox
Wireshark · Follow TCP Stream (tcp.stream eq 2) · any
File Edit View Go Capture Analyze
tcp.stream eq 2
No. Time Source
Frame 50: 1789 bytes on wire (14
Linux cooked capture v1
Internet Protocol Version 4, Src
Transmission Control Protocol,
Transport Layer Security
Entire conversation (3,206 bytes)
Show as ASCII
No delta times
Stream 2
Find:
Case sensitive
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help
wireshark_anyNGSQ92.pcapng
Profile: Default