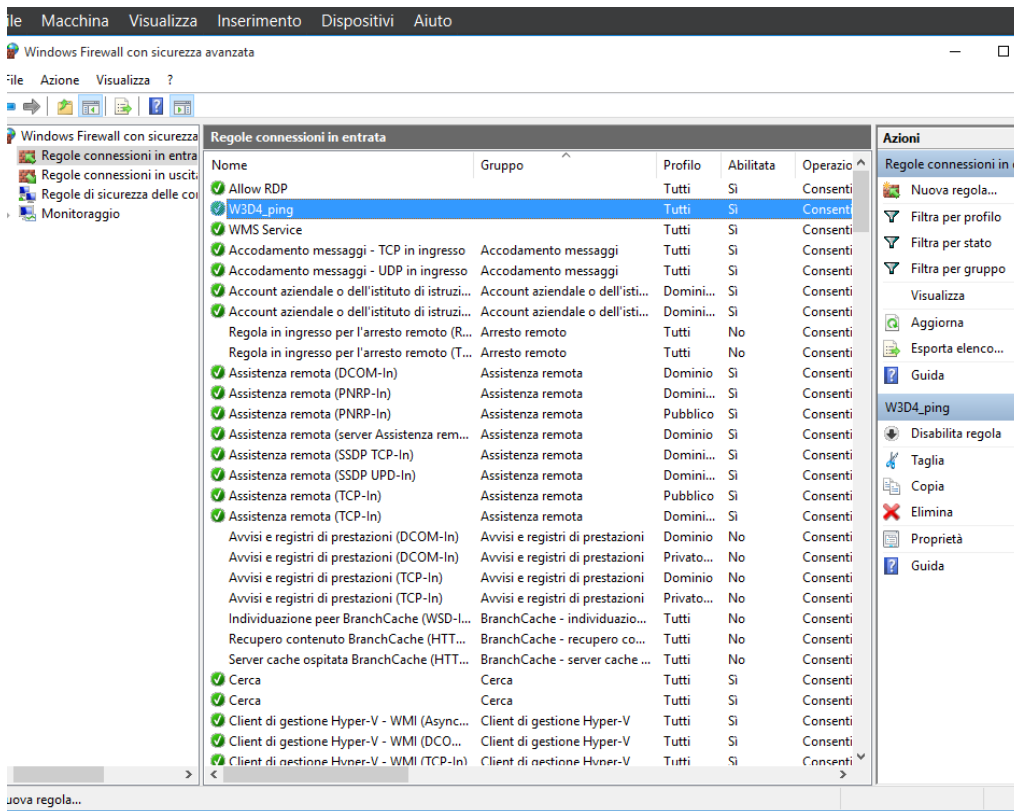


CONFIGURARE IL FIREWALL PER IL PING

Dovevo far funzionare il ping dalla mia macchina Kali Linux (IP 192.168.50.100) verso la macchina Windows (IP 192.168.50.102). Ho aperto il terminale su Kali e ho provato a lanciare ping 192.168.50.102, ma non arrivava nulla: tutti i pacchetti persi! Sapevo che il firewall di Windows blocca i ping di default, quindi sono passato alla macchina Windows per sistemare la cosa.

Sul Pannello di controllo di Windows, ho cercato “Windows Firewall” e ho aperto le impostazioni. Sono andato su “Inbound Rules” (le regole per il traffico in entrata) e ho cliccato su “Nuova regola”. Ho scelto “Personalizzata” per avere pieno controllo. Non mi interessava un programma specifico, quindi ho selezionato “Tutti i programmi”. Poi, ho impostato il protocollo su ICMP, che è quello del ping. Ho lasciato “Qualsiasi indirizzo IP” per sorgente e destinazione, così non limitavo nulla. Ho scelto “Consenti la connessione” e ho applicato la regola a tutti i profili (dominio, privato, pubblico). Per trovarla facilmente, l’ho chiamata “W3D4_ping” con una breve descrizione. Ho salvato tutto e sono tornato su Kali.



Ho rilanciato il comando ping 192.168.50.102 e tutti i pacchetti sono arrivati. Il ping funzionava.

```
File Actions Edit View Help
sh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ ping 192.168.50.102

PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data:
4 bytes from 192.168.50.102: icmp_seq=1 ttl=128 time=0.865 ms
4 bytes from 192.168.50.102: icmp_seq=2 ttl=128 time=0.454 ms
4 bytes from 192.168.50.102: icmp_seq=3 ttl=128 time=0.566 ms
4 bytes from 192.168.50.102: icmp_seq=4 ttl=128 time=0.595 ms
4 bytes from 192.168.50.102: icmp_seq=5 ttl=128 time=0.551 ms
4 bytes from 192.168.50.102: icmp_seq=6 ttl=128 time=0.586 ms
C
— 192.168.50.102 ping statistics —
6 packets transmitted, 6 received, 0% packet loss, time 5567ms
rtt min/avg/max/mdev = 0.454/0.602/0.865/0.126 ms
```

CONFIGURARE INTESIM PER HTTPS E USARE WIRESHARK

Ho aperto un terminale su Kali e ho digitato `sudo nano /etc/inetsim/inetsim.conf` per modificare il file di configurazione di InetSim. Quando ho aperto il file, c'era una lista infinita di servizi attivi: DNS, HTTP, FTP, ecc... Ma a me serviva solo HTTPS.

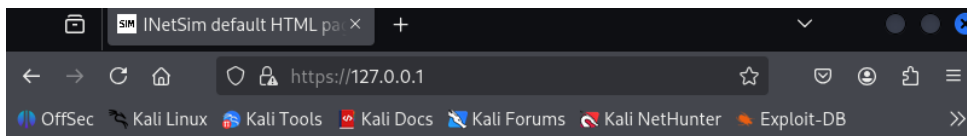
Le impostazioni di HTTPS erano già ok: il servizio ascoltava su localhost (127.0.0.1) e sulla porta 443, quella standard.

```
File Actions Edit View Help
GNU nano 8.4
#dns_server 127.0.0.1
#http_server 127.0.0.1
https_server 127.0.0.1
#ftp_server 127.0.0.1
#ftps_server 127.0.0.1
#smtp_server 127.0.0.1#####
#
# InetSim configuration file
#
```

Ho salvato il file e ho avviato InetSim con `sudo inetsim`. L'output mi ha confermato che HTTPS era attivo sulla porta 443.

```
File Actions Edit View Help
L$ sudo inetsim
[sudo] password for kali:
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.
Warning: Unknown option 'https_server' in configuration file '/etc/inetsim/inetsim.conf' line 3
Warning: Unknown option 'service_bind_address' in configuration file '/etc/inetsim/inetsim.conf' line 385
Configuration file parsed successfully.
== INetSim main process started (PID 5676) ==
Session ID: 5676
Listening on: 0.0.0.0
Real Date/Time: 2025-07-12 03:34:48
Fake Date/Time: 2025-07-12 03:34:48 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 5686)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at /usr/share/perl5/INetSim/DNS.pm line 69.
* irc_6667_tcp - started (PID 5696)
* time_37_tcp - started (PID 5701)
* ident_113_tcp - started (PID 5699)
* smtp_25_tcp - started (PID 5689)
* tftp_69_udp - started (PID 5695)
* discard_9_udp - started (PID 5708)
* smtps_465_tcp - started (PID 5690)
* daytime_13_udp - started (PID 5704)
* ftp_21_tcp - started (PID 5693)
* https_443_tcp - started (PID 5688)
* http_80_tcp - started (PID 5687)
* echo_7_udp - started (PID 5706)
* chargen_19_udp - started (PID 5712)
* quotd_17_udp - started (PID 5710)
* discard_9_tcp - started (PID 5707)
* finger_79_tcp - started (PID 5698)
* time_37_udp - started (PID 5702)
* syslog_514_udp - started (PID 5700)
* daytime_13_tcp - started (PID 5703)
* ftps_990_tcp - started (PID 5694)
* pop3s_995_tcp - started (PID 5692)
* ntp_123_udp - started (PID 5697)
* echo_7_tcp - started (PID 5705)
* chargen_19_tcp - started (PID 5711)
* pop3_110_tcp - started (PID 5691)
* quotd_17_tcp - started (PID 5709)
* dummy_1_udp - started (PID 5714)
* dummy_1_tcp - started (PID 5713)
done.
Simulation running.
```

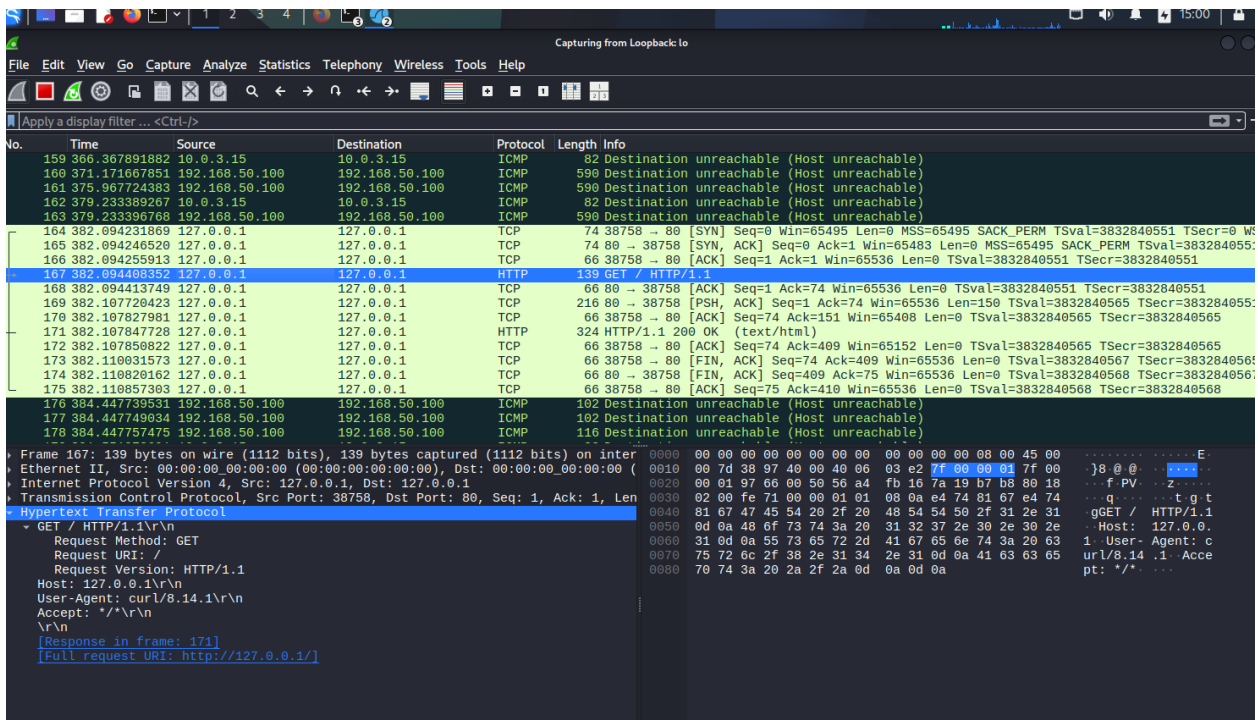
Per sicurezza, ho aperto un browser su Kali e ho digitato <https://127.0.0.1>



This is the default HTML page for INetSim HTTP server fake mode.

This file is an HTML document.

Poi, ho aperto Wireshark e ho messo in ascolto l'interfaccia di loopback (localhost). Ho rifatto la connessione a https://127.0.0.1 dal browser e ho catturato il traffico. In Wireshark, ho visto e analizzato i dettagli dei pacchetti.



ESERCIZIO FACOLTATIVO

SIMULARE ALTRI SERVIZI CON INTESIM E ANALIZZARE IL TRAFFICO

Ho provato a simulare un altro servizio con InetSim, catturare il traffico e analizzarlo.

Ho scelto di configurare il servizio FTP, perché mi sembrava un buon modo per vedere qualcosa di diverso da HTTPS. Sono tornato al file di configurazione di InetSim con `sudo nano /etc/inetsim/inetsim.conf`. Ho cercato la sezione dedicata a FTP e ho rimosso il “#” dalla riga che abilita il servizio FTP, lasciando commentati gli altri servizi (tranne HTTPS, che avevo già configurato).

Le impostazioni di default per FTP erano: indirizzo localhost (127.0.0.1) e porta 21, che è quella standard.

Non ho cambiato nulla, ho salvato il file e ho riavviato InetSim con `sudo inetsim`. L’output mi ha confermato che FTP era attivo sulla porta 21.

Per testarlo, ho aperto un terminale su Kali e ho provato a connettermi con `ftp 127.0.0.1`. Ho usato le credenziali di default di InetSim (utente: “anonymous”, password: password). La connessione è riuscita, e ho visto la directory fittizia di InetSim.

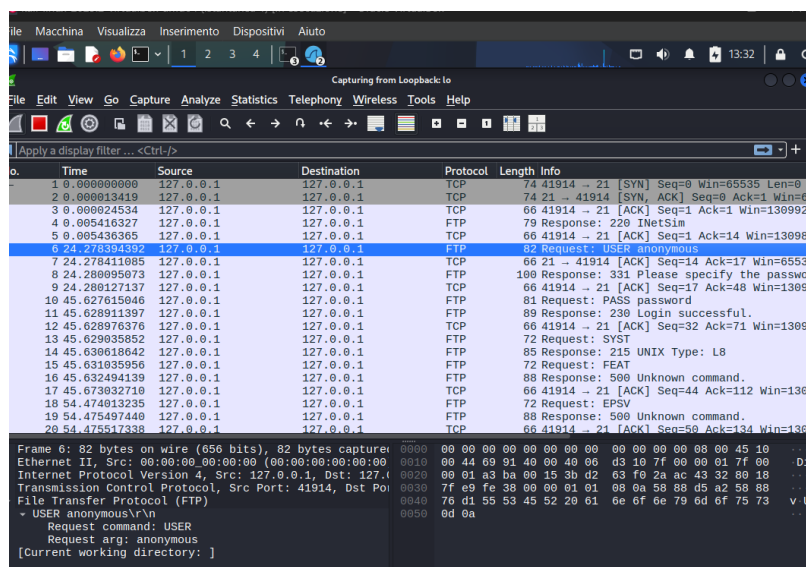
```
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ ftp 127.0.0.1
Connected to 127.0.0.1.
220 INetSim FTP Service ready.
Name (127.0.0.1:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

```
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
-rw-rw-rw-  1 1021 1021      28 Jul 12 04:21 sample.txt
226 Transfer complete.
```

Poi, ho riavviato Wireshark, sempre sull’interfaccia di loopback, e ho rifatto la connessione FTP

```
(kali@kali)-[~]
$ ftp 127.0.0.1
Connected to 127.0.0.1.
220 INetSim FTP Service ready.
Name (127.0.0.1:kali):
```

Ho catturato il traffico e ho notato i pacchetti TCP per la connessione iniziale, ma anche i comandi FTP come “USER” e “PASS” per l’autenticazione. Nel pannello di Wireshark, ho analizzato il contenuto: potevo vedere chiaramente i comandi FTP inviati e le risposte del server simulato.



È stato utile per capire come funziona il protocollo FTP e come i pacchetti si muovono tra client e server