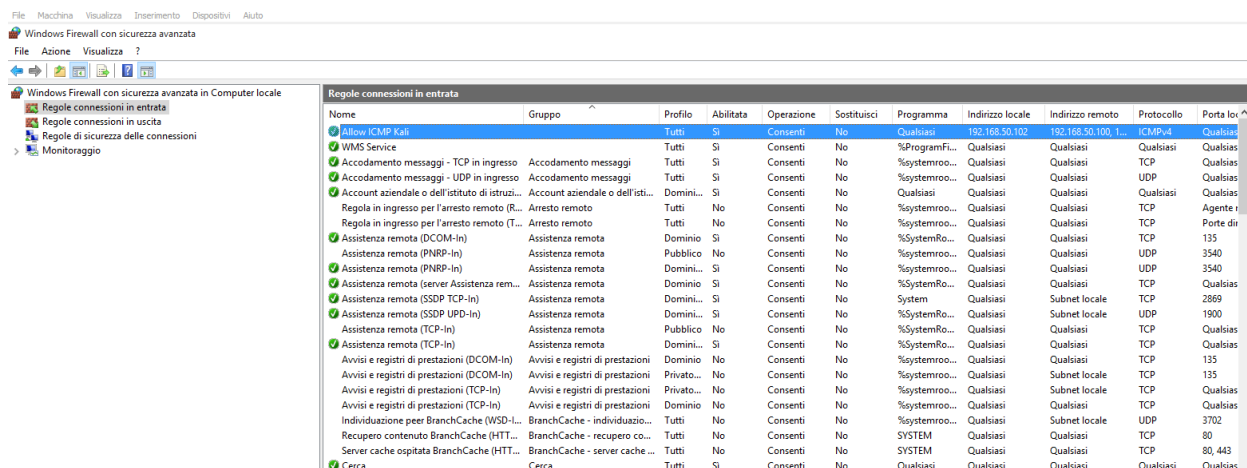
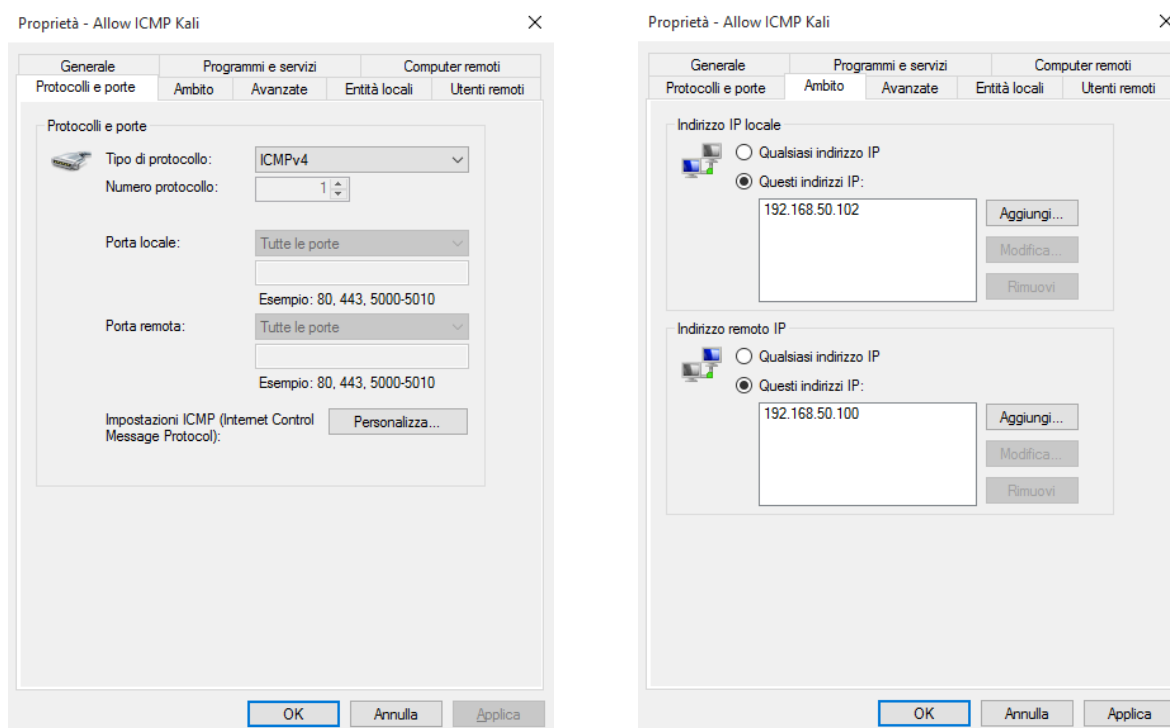


Impostare una regola con Windows Firewall

In questo esercizio vedremo come impostare una regola sul firewall nativo di Windows per consentire il ping da **Kali (192.168.50.100)** verso la **VM Windows 10** e, allo stesso tempo, bloccare ogni tipo di richiesta **ICMP** proveniente dalle altre macchine presenti nella sottorete. La regola dovrà essere creata e messa in alto. Di default, il firewall di Windows blocca ogni richiesta ICMP in entrata, quindi la nostra regola dovrà esplicitamente consentire il ping proveniente dall'indirizzo IP della nostra macchina **Kali**.



Ho chiamato la regola **Allow ICMP Kali**, specificando come protocollo **ICMPv4**, come indirizzo IP locale **192.168.50.102** appartenente alla macchina Windows e come IP remoto **192.168.50.100** (Kali).



Adesso possiamo procedere a “pingare” la macchina Windows sia da **Kali** che da **Metasploitable**. Da **Kali** possiamo vedere che la richiesta viene inviata con successo e riceviamo risposta dalla macchina Windows.

```
(kali㉿kali)-[~]  
$ ping 192.168.50.102  
PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data.  
64 bytes from 192.168.50.102: icmp_seq=1 ttl=128 time=0.570 ms  
64 bytes from 192.168.50.102: icmp_seq=2 ttl=128 time=0.777 ms  
64 bytes from 192.168.50.102: icmp_seq=3 ttl=128 time=0.558 ms  
64 bytes from 192.168.50.102: icmp_seq=4 ttl=128 time=0.508 ms  
^C  
— 192.168.50.102 ping statistics —  
4 packets transmitted, 4 received, 0% packet loss, time 3053ms  
rtt min/avg/max/mdev = 0.508/0.603/0.777/0.102 ms  
  
(kali㉿kali)-[~]  
$
```

Da **Metasploitable** invece il ping non ha successo. Questo conferma che la nostra regola funziona perfettamente.

```
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
msfadmin@metasploitable:~$ ping 192.168.50.102  
PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data.
```

Catturare traffico di rete con Wireshark

In questa seconda metà del compito invece vedremo come simulare dei servizi di rete (**HTTP/HTTPS**) sulla nostra macchina **Kali** utilizzando il tool **InetSim** e successivamente eseguiremo una serie di catture utilizzando Wireshark per analizzare il traffico ed il flusso di dati tra una macchina e l'altra e tra i vari livelli della suite **TCP/IP** (5 layers version).

Per poter utilizzare **InetSim** al meglio dobbiamo modificare il file di configurazione **/etc/inetsim/inetsim.conf**. Qui commenteremo, dalla lista dei servizi che il tool è in grado di simulare, tutti i servizi che non ci interessano. Lascieremo quindi attivi i servizi web **HTTP** e **HTTPS**.

```
#####
# start_service control Protocol, Src Port: 80, Dst Port: 49
# Content-Type: text/html
# Content-Length: 409 (bytes) #17(150), #13(2
# Hypertext Transfer Protocol
# The services to start
# Date: Mon, 13 Oct 2025 15:32:54 GMT\r\n
# Syntax: start_service <service name>
# Content-Type: text/html
# Connection: Close\r\n
# 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
```

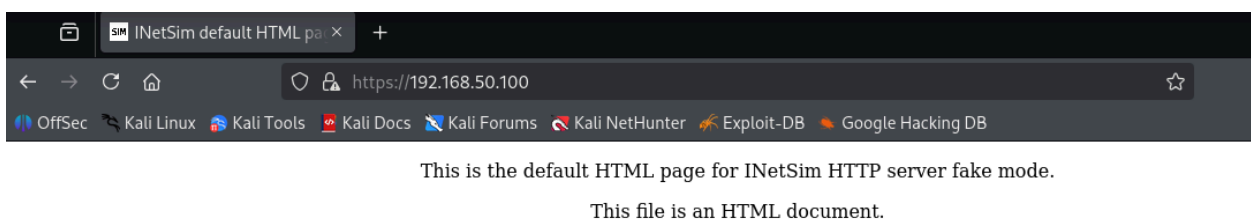
Dopo di che dovremo impostare l'indirizzo IP della macchina che verrà utilizzata per servire la pagina web.

```
#####
# service_bind_address
#
# IP address to bind services to
# [Syntax: service_bind_address <IP address>]
# Syntax: service_bind_address <IP address>
# File Data: 256 bytes
# Default: 127.0.0.1 [text/html (10 lines)]
#
service_bind_address 192.168.50.100
```

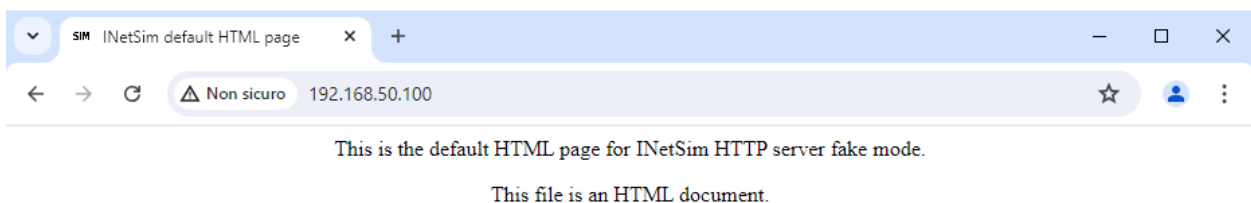
Adesso possiamo salvare e avviare **InetSim** con privilegi di root.

```
(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 6043) ===
Session ID:      6043
Listening on:    192.168.50.100
Real Date/Time:  2025-10-13 11:23:16
Fake Date/Time:  2025-10-13 11:23:16 (Delta: 0 seconds)
Forking services...
  * http_80_tcp - started (PID 6045)
  * https_443_tcp - started (PID 6046)
done.
Simulation running.
```

A questo punto, se apriamo un browser dalla nostra macchina **Kali**, riusciremo a visualizzare con successo la pagina web gestita da **InetSim**; sia in **HTTP** che in **HTTPS**.



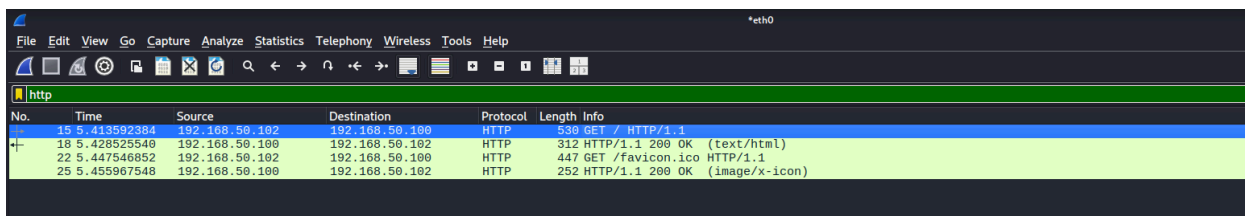
Possiamo anche provare ad accedere alla pagina via browser da **Windows**...



...e utilizzando **cURL** da **Metasploitable**.

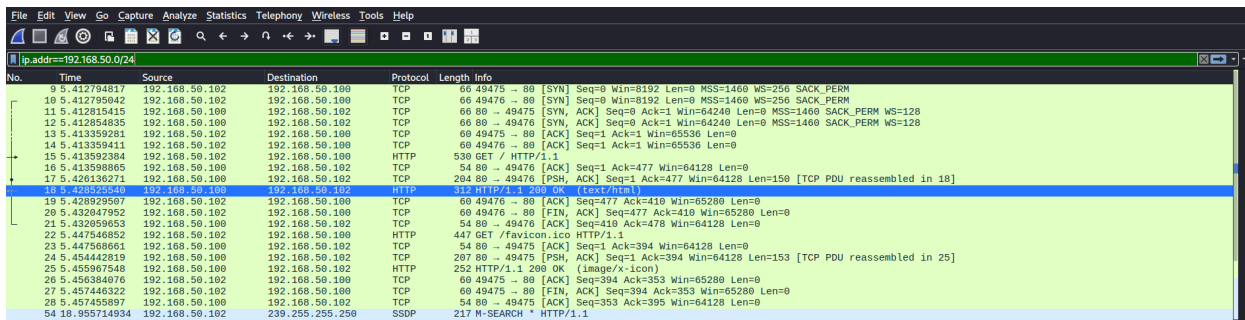
```
msfadmin@metasploitable:~$ curl http://192.168.50.100
<html>
  <head>
    <title>INetSim default HTML page</title>
  </head>
  <body>
    <p></p>
    <p align="center">This is the default HTML page for INetSim HTTP server fake
mode.</p>
    <p align="center">This file is an HTML document.</p>
  </body>
</html>
msfadmin@metasploitable:~$ _
```

Ora che i servizi sono attivi possiamo lanciare Wireshark per fargli catturare una richiesta **GET** originata dalla macchina Windows verso il web server (**Kali - InetSim**).



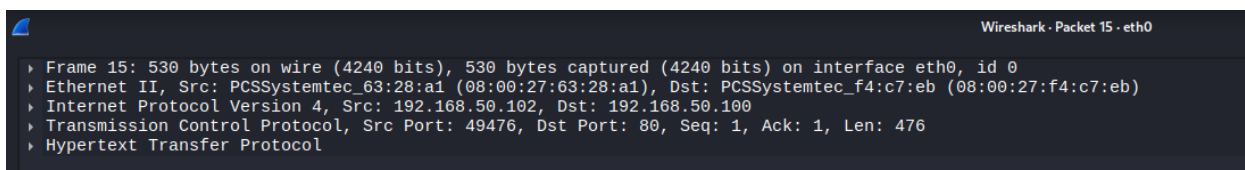
No.	Time	Source	Destination	Protocol	Length	Info
15	5.413592384	192.168.50.102	192.168.50.100	HTTP	530	GET / HTTP/1.1
18	5.428525540	192.168.50.100	192.168.50.102	HTTP	312	HTTP/1.1 200 OK (text/html)
22	5.447546852	192.168.50.102	192.168.50.100	HTTP	447	GET /favicon.ico HTTP/1.1
25	5.455967548	192.168.50.100	192.168.50.102	HTTP	252	HTTP/1.1 200 OK (image/x-icon)

Possiamo anche filtrare in modo da poter vedere tutte le comunicazioni avvenute all'interno della rete **192.168.50.0/24**.



No.	Time	Source	Destination	Protocol	Length	Info
9	5.412794817	192.168.50.102	192.168.50.100	TCP	66	49475 -> 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
10	5.412795042	192.168.50.102	192.168.50.100	TCP	66	49476 -> 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
11	5.412815415	192.168.50.100	192.168.50.102	TCP	66	80 -> 49475 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
12	5.412854835	192.168.50.100	192.168.50.102	TCP	66	80 -> 49476 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
13	5.413359281	192.168.50.102	192.168.50.100	TCP	60	49475 -> 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
14	5.413359411	192.168.50.102	192.168.50.100	TCP	60	49476 -> 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
15	5.413592384	192.168.50.102	192.168.50.100	HTTP	530	GET / HTTP/1.1
16	5.413598865	192.168.50.100	192.168.50.102	TCP	54	80 -> 49476 [ACK] Seq=1 Ack=477 Win=64128 Len=0
17	5.426136271	192.168.50.100	192.168.50.102	TCP	204	80 -> 49476 [PSH, ACK] Seq=1 Ack=477 Win=150 [TCP PDU reassembled in 18]
18	5.428525540	192.168.50.100	192.168.50.102	HTTP	312	HTTP/1.1 200 OK (text/html)
19	5.428929507	192.168.50.102	192.168.50.100	TCP	60	49476 -> 80 [ACK] Seq=477 Ack=410 Win=65280 Len=0
20	5.432047952	192.168.50.102	192.168.50.100	TCP	60	49476 -> 80 [FIN, ACK] Seq=477 Ack=410 Win=65280 Len=0
21	5.432059653	192.168.50.100	192.168.50.102	TCP	54	80 -> 49476 [ACK] Seq=410 Ack=478 Win=64128 Len=0
22	5.447546852	192.168.50.102	192.168.50.100	HTTP	447	GET /favicon.ico HTTP/1.1
23	5.447558651	192.168.50.100	192.168.50.102	TCP	54	80 -> 49475 [ACK] Seq=1 Ack=394 Win=64128 Len=0
24	5.454442819	192.168.50.100	192.168.50.102	TCP	207	80 -> 49475 [PSH, ACK] Seq=1 Ack=394 Win=64128 Len=153 [TCP PDU reassembled in 25]
25	5.455967548	192.168.50.100	192.168.50.102	HTTP	252	HTTP/1.1 200 OK (image/x-icon)
26	5.456384876	192.168.50.102	192.168.50.100	TCP	60	49475 -> 80 [ACK] Seq=394 Ack=353 Win=65280 Len=0
27	5.457446322	192.168.50.102	192.168.50.100	TCP	60	49475 -> 80 [FIN, ACK] Seq=394 Ack=353 Win=65280 Len=0
28	5.457455897	192.168.50.100	192.168.50.102	TCP	54	80 -> 49475 [ACK] Seq=353 Ack=395 Win=64128 Len=0
54	18.955714934	192.168.50.102	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1

Nell'immagine sopra possiamo vedere che la comunicazione ha inizio con il consueto **Three-Way Handshake**, in seguito al quale, a connessione avvenuta, avviene lo scambio **HTTP**. Se prendiamo la richiesta HTTP e ne analizziamo i dettagli possiamo vedere che Wireshark ci mostra tutti i dettagli sui cinque livelli che compongono lo stack TCP/IP.



Frame 15: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_63:28:a1 (08:00:27:63:28:a1), Dst: PCSSystemtec_f4:c7:eb (08:00:27:f4:c7:eb)
Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100
Transmission Control Protocol, Src Port: 49476, Dst Port: 80, Seq: 1, Ack: 1, Len: 476
Hypertext Transfer Protocol

Il primo livello, dall'alto verso il basso, è il layer fisico. Dopo di esso troviamo il layer **Data-Link**, dove ci vengono mostrati gli indirizzi **MAC** di origine e di destinazione dei due dispositivi.

```
Wireshark · Packet 15 · eth0
▶ Frame 15: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
▶ Ethernet II, Src: PCSSystemtec_63:28:a1 (08:00:27:63:28:a1), Dst: PCSSystemtec_f4:c7:eb (08:00:27:f4:c7:eb)
  ▶ Destination: PCSSystemtec_f4:c7:eb (08:00:27:f4:c7:eb)
  ▶ Source: PCSSystemtec_63:28:a1 (08:00:27:63:28:a1)
  Type: IPv4 (0x0800)
  [Stream index: 2]
▶ Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100
▶ Transmission Control Protocol, Src Port: 49476, Dst Port: 80, Seq: 1, Ack: 1, Len: 476
▶ Hypertext Transfer Protocol
```

Nel successivo troviamo tutte le informazioni relative al livello **Internet/Network**, con gli indirizzi IP delle due parti comunicanti.

```
Wireshark · Packet 15 · eth0
▶ Frame 15: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
▶ Ethernet II, Src: PCSSystemtec_63:28:a1 (08:00:27:63:28:a1), Dst: PCSSystemtec_f4:c7:eb (08:00:27:f4:c7:eb)
▶ Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 516
    Identification: 0x586d (22637)
  ▶ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0xba6b [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.50.102
    Destination Address: 192.168.50.100
    [Stream index: 0]
  ▶ Transmission Control Protocol, Src Port: 49476, Dst Port: 80, Seq: 1, Ack: 1, Len: 476
  ▶ Hypertext Transfer Protocol
```

Nel layer quattro troviamo i dettagli del protocollo di trasporto **TCP** con indicate le porte di origine e destinazione.

```
Wireshark · Packet 15 · eth0
▶ Frame 15: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
▶ Ethernet II, Src: PCSSystemtec_63:28:a1 (08:00:27:63:28:a1), Dst: PCSSystemtec_f4:c7:eb (08:00:27:f4:c7:eb)
▶ Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100
▶ Transmission Control Protocol, Src Port: 49476, Dst Port: 80, Seq: 1, Ack: 1, Len: 476
  Source Port: 49476
  Destination Port: 80
  [Stream index: 1]
  [Stream Packet Number: 4]
  ▶ [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 476]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 2275732178
  [Next Sequence Number: 477 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 888146684
  0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x018 (PSH, ACK)
  Window: 256
  [Calculated window size: 65536]
  [Window size scaling factor: 256]
  Checksum: 0x1e5e [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  ▶ [Timestamps]
  ▶ [SEQ/ACK analysis]
  TCP payload (476 bytes)
  ▶ Hypertext Transfer Protocol
```

Mentre nel livello applicativo risiede il contenuto della richiesta **GET HTTP**.

```
Wireshark - Packet 15 - eth0
> Frame 15: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_63:28:a1 (08:00:27:63:28:a1), Dst: PCSSystemtec_f4:c7:eb (08:00:27:f4:c7:eb)
> Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100
> Transmission Control Protocol, Src Port: 49476, Dst Port: 80, Seq: 1, Ack: 1, Len: 476
> Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: 192.168.50.100\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7\r\n
  \r\n
[Response in frame: 18]
[Full request URI: http://192.168.50.100/]
```

Possiamo inoltre notare che, utilizzando il protocollo **HTTP** piuttosto che **HTTPS**, il contenuto della richiesta e della risposta sono in chiaro e leggibili da chiunque stesse “sniffando” la rete.

0000	08 00 27 63 28 a1 08 00 27 f4 c7 eb 08 00 45 00	.. 'c(... 'E.
0010	01 2a bc 7f 40 00 40 06 97 33 c0 a8 32 64 c0 a8	. * . . @ . @ . . 3 . . 2d . .
0020	32 66 00 50 c1 44 34 f0 0b 92 87 a4 ec ae 50 19	2f . P . D4 P .
0030	01 f5 e7 37 00 00 3c 68 74 6d 6c 3e 0a 20 20 3c	... 7 ... <h tml> . <
0040	68 65 61 64 3e 0a 20 20 20 20 3c 74 69 74 6c 65	head> . <title
0050	3e 49 4e 65 74 53 69 6d 20 64 65 66 61 75 6c 74	>INetSim default
0060	20 48 54 4d 4c 20 70 61 67 65 3c 2f 74 69 74 6c	HTML pa ge</titl
0070	65 3e 0a 20 20 3c 2f 68 65 61 64 3e 0a 20 20 3c	e> . </h ead> . <
0080	62 6f 64 79 3e 0a 20 20 20 20 3c 70 3e 3c 2f 70	body> . <p></p
0090	3e 0a 20 20 20 20 3c 70 20 61 6c 69 67 6e 3d 22	> . <p align="
00a0	63 65 6e 74 65 72 22 3e 54 68 69 73 20 69 73 20	center"> This is
00b0	74 68 65 20 64 65 66 61 75 6c 74 20 48 54 4d 4c	the defa ult HTML
00c0	20 70 61 67 65 20 66 6f 72 20 49 4e 65 74 53 69	page fo r INetSi
00d0	6d 20 48 54 54 50 20 73 65 72 76 65 72 20 66 61	m HTTP s erver fa
00e0	6b 65 20 6d 6f 64 65 2e 3c 2f 70 3e 0a 20 20 20	ke mode. </p> .
00f0	20 3c 70 20 61 6c 69 67 6e 3d 22 63 65 6e 74 65	<p align ="cente
0100	72 22 3e 54 68 69 73 20 66 69 6c 65 20 69 73 20	r">This file is
0110	61 6e 20 48 54 4d 4c 20 64 6f 63 75 6d 65 6e 74	an HTML document
0120	2e 3c 2f 70 3e 0a 20 20 3c 2f 62 6f 64 79 3e 0a	.</p> . </body> .
0130	3c 2f 68 74 6d 6c 3e 0a	</html> .