

PROGETTO M1 - W4D4

Introduzione all'Hacking

Configurazione IP

Kali: 192.168.50.100

Windows: 192.168.50.102

Per attivare i servizi **HTTPS** e **DNS** basta lanciare il comando “*sudo nano /etc/inetsim/inetsim.conf*”, così aprendo, come si vede sotto in figura, la configurazione di **InetSim**, dopodiché basta levare l’hashtag per far sì che il comando venga eseguito, perché lasciando l’hashtag la linea diventa un commento e non sarà eseguibile.

```
# ~~~~~ /esercizio/tmp ~~~~~
# INetSim configuration file
# command 'ls' not found, did you mean:
#####
# command 'lsd' from deb lsd
# command 'les' from deb atm-tools
#####
# Main configuration deb binutils
#####
# command 'ls' from deb coreutils
#####
# start_service install <deb name>
#
# The services to start(cizio/tmp)
#
# Syntax: start_service <service name>
#
# Default: none (~./esercizio/tmp)
#
# 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 ~./esercizio/tmp
start_service http
start_service https
#start_service smtp
#start_service smtps(esercizio)
#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
```

per configurare il servizio DNS, come nelle figure sotto, bisogna aggiungere la linea “dns_static epicode.internal 192.168.50.100” per configurare l’indirizzo IP (kali) con il nome epicode.internal, così da configurare il DNS, ovvero il servizio farà sì che se cerchiamo epicode.internal si reindirizzerà al IP assegnato e viceversa, ricordiamo che il servizio DNS traduce i nomi di dominio in indirizzi IP per facilitare agli utenti gli accessi ai siti web. Inoltre si è configurato la parte dns_default_ip con 0.0.0.0 (tutti gli indirizzi).

```
#####
# dns_static 39463
# Listening on: 0.0.0.0
# Static mappings for DNS: 14:35:59
# Fake Date/Time: 2025-07-18 14:35:59 (Delta: 0 seconds)
# Syntax: dns_static <fqdn hostname> <IP address>
# Can't locate object method "listen" via package "Net::DNS::Nameserver" at /usr/share/perl5/Net/Sim/DNS.pm line 69.
# Default: none - started (PID 39474)
# * ftp_21_tcp - started (PID 39476)
# dns_static www.foo.com 10.10.10.10 (75)
# dns_static ns1.foo.com 10.70.50.30
# dns_static ftp.bar.net 10.10.20.30
# dns_static epicode.internal 192.168.50.100
# * http_80_tcp - stopped (PID 39474)
```

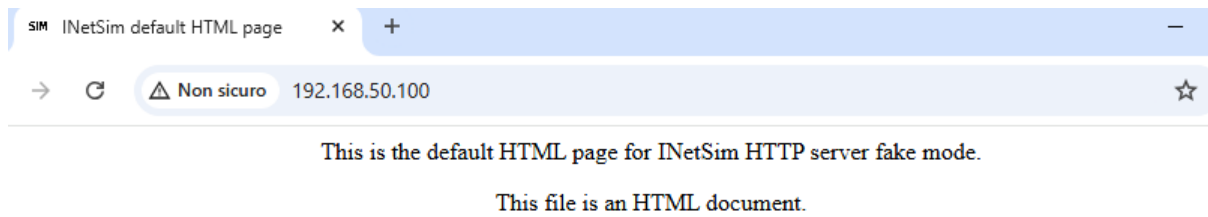
```
#####
# dns_default_ip
# Using log directory: /var/lib/inetsim/
# Using report directory: /var/log/inetsim/report/
# Default IP address to return with DNS replies
# Parsing configuration file.
# Syntax: dns_default_ip <IP address>
# InetSim main process started (PID 39463)
# Default: 127.0.0.1
# Listening on: 0.0.0.0
# dns_default_ip 0.0.0.0 -18 14:35:59
# Fake Date/Time: 2025-07-18 14:35:59 (Delta: 0 seconds)
```

infine si può lanciare il comando “*sudo inetsim*” per avviarlo

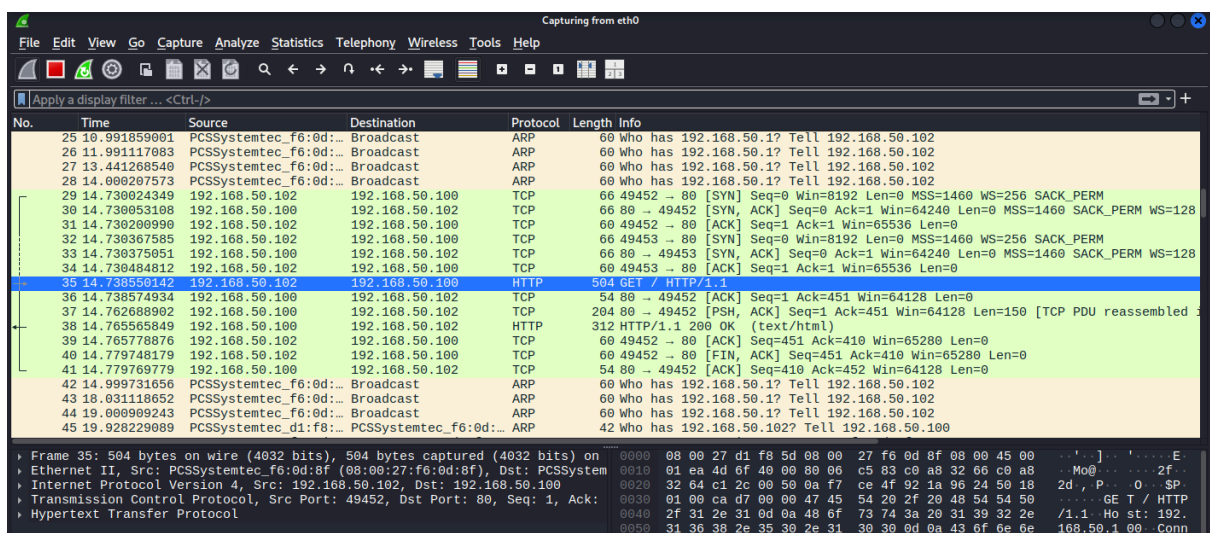
```
(kali@kali)-[~]
$ sudo inetsim
[sudo] password for kali:
InetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== InetSim main process started (PID 22346) ==
Session ID: 22346
Listening on: 0.0.0.0
Real Date/Time: 2025-07-18 14:00:58
Fake Date/Time: 2025-07-18 14:00:58 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 22356)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at /usr/share/perl5/Net/Sim/DNS.pm line 69.
* ftp_21_tcp - started (PID 22359)
* http_80_tcp - started (PID 22357)
* https_443_tcp - started (PID 22358)
done.
Simulation running.
```

come si può vedere c’è stato un errore nel DNS (visto insieme a lezione)
nel mio caso ho lasciato attivo i servizi FTP, HTTP e HTTPS

Adesso se facciamo la prova su Windows, avviamo Chrome e cerchiamo <http://192.168.50.100> ci da come risultato:



quindi funziona, se il DNS era in funzione bastava cercare <http://epicode.internal> e dava lo stesso risultato. Dava stesso risultato anche se si cercava <https://192.168.50.100>. Sotto si può vedere l'analisi da parte di Wireshark e si può notare il three-way-handshake (No.29 - 31) tra client (Windows) e server (Kali), che si tratta di una procedura per stabilire una connessione sicura tra i due dispositivi e viene utilizzata nel protocollo TCP



sempre nella figura precedente, si può vedere come cambiano gli indirizzi MAC source e destination

MAC address Source	MAC address Destination
Windows	Kali
Kali	Windows
Windows	Kali

(esempio. three way handshake, lo scambio dei pacchetti tra i due dispositivi Kali e Windows)

Ecco il contenuto della richiesta HTTP (rosso client e blu server). Si può vedere la richiesta e la risposta da parte del server con lo status code 200 "OK", ovvero la richiesta è stata ricevuta, elaborata e completata con successo.



```
Wireshark · Follow HTTP Stream (tcp.stream eq 0) · eth0

GET / HTTP/1.1
Host: 192.168.50.100
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7

HTTP/1.1 200 OK
Connection: Close
Content-Type: text/html
Date: Fri, 18 Jul 2025 18:43:25 GMT
Server: INetSim HTTP Server
Content-Length: 258

<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>
```

A differenza del servizio HTTP (porta 80), Hyper Text Transfer Protocol, che trasferisce i dati in chiaro rendendoli vulnerabili e con facile accesso da terze parti, il servizio HTTPS (porta 443), HTTP Secure, protegge i dati usando la crittografia TLS/SSL così rendendo la trasmissione dei dati sicura e non facile da accedere.

Wireshark non è riuscito analizzare i pacchetti del servizio HTTPS perché appunto erano crittografati, se nel caso si potesse, la differenza con il contenuto HTTP (come in figura precedente) dove i dati sono leggibili, nel contenuto HTTPS non si riesce a leggere dato che sono cifrati