



# **EPICODE**

**Progetto Alessandro Moscetti 29/09**



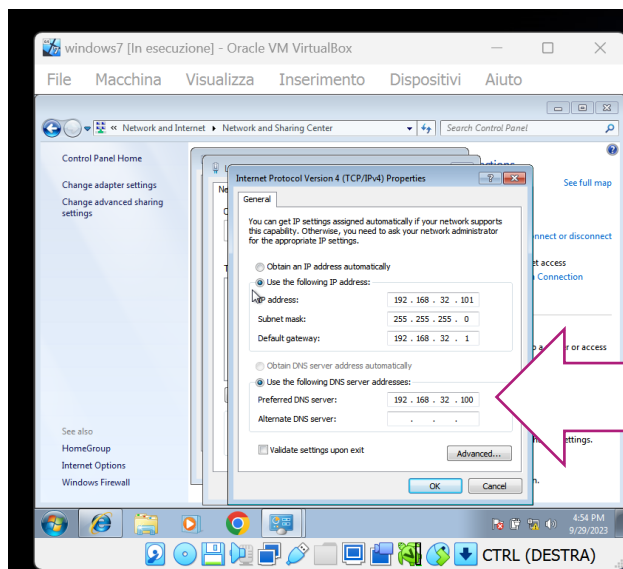
```
File Actions Edit View Help
GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1
```

Come primo procedimento sono andato a sostituire gli indirizzi IP delle due macchine virtuali come veniva richiesto dalla traccia.



Ho inserito nel DNS server l'indirizzo dato a kali perché diventerà il server DNS

A seguire entrando nel codice di inetsim per la simulazione dei server, sono andato ad attivare i servizi richiesti (fig. 1).

Sono andato anche a modificare il service bind address (fig. 2), per farlo combaciare con l'IP richiesto, in questo caso l'IP della macchina kali.

Infine siamo andati a configurare la regola del DNS (fig. 3) con cui il nome epicode.internal viene associato all'IP della nostra macchina.

```
#  
# The services to start  
#  
# Syntax: start_service <service name>  
#  
# Default: none  
#  
# Available service names are:  
# dns, http, smtp, pop3, tftp, ftp, ntp, time_t  
# 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, p  
# 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
```

1

```
# Default: 127.0.0.1  
#  
service_bind_address 192.168.32.100
```

2

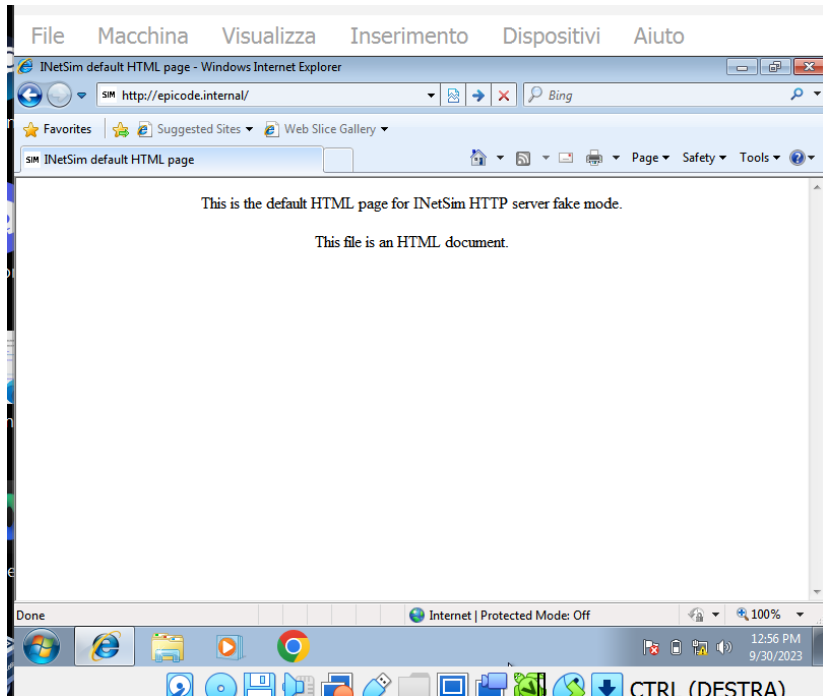
```
#  
# Syntax: dns_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
dns_static epicode.internal 192.168.32.100  
#dns_static ns1.foo.com 10.70.50.30  
#dns_static ftp.bar.net 10.10.20.30  
#####  
# dns version
```

3

```
(jonny@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 3064) ==  
Session ID: 3064  
Listening on: 192.168.32.100  
Real Date/Time: 2023-09-30 12:54:53  
Fake Date/Time: 2023-09-30 12:54:53 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 3074)  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.  
* http_80_tcp - started (PID 3075)  
done.  
Simulation running.  
█
```

1

Andando ad attivare la simulazione con il comando inetsim (fig. 1), la nostra seconda macchina trova sotto il nome 'epicode.internal' la nostra pagina http finta (fig. 2).



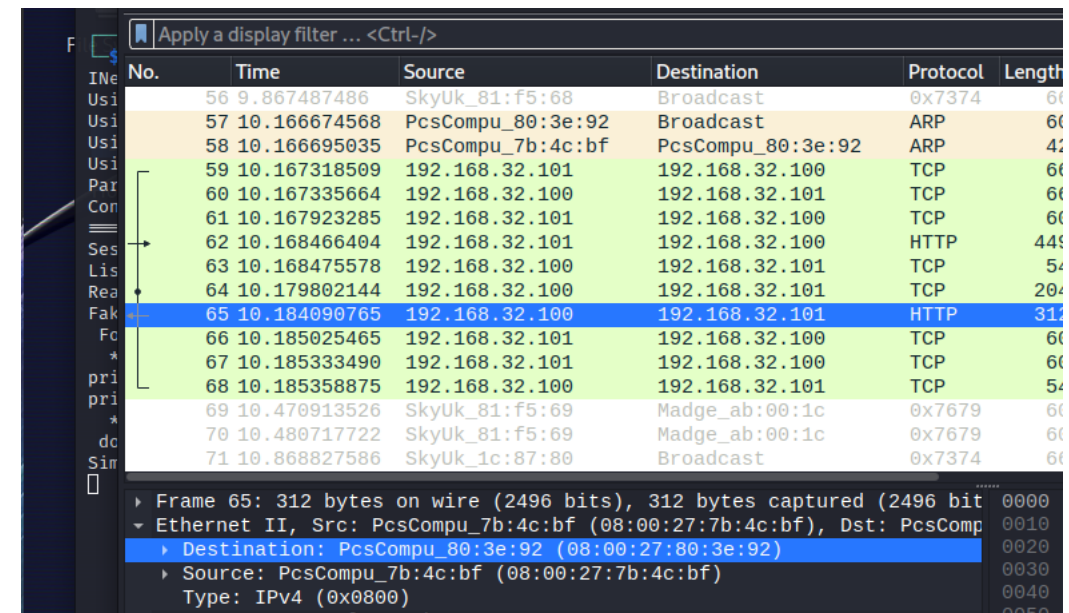
2

Nel mentre utilizzando Wireshark, sono andato a sniffare i pacchetti.

Possiamo notare nella fig. 1 il protocollo utilizzato in questo caso HTTP, che a sua volta utilizza il protocollo TCP.

Sempre nella fig. 1 possiamo notare gli IP di sorgente e destinazione e i loro MAC address (in blu).

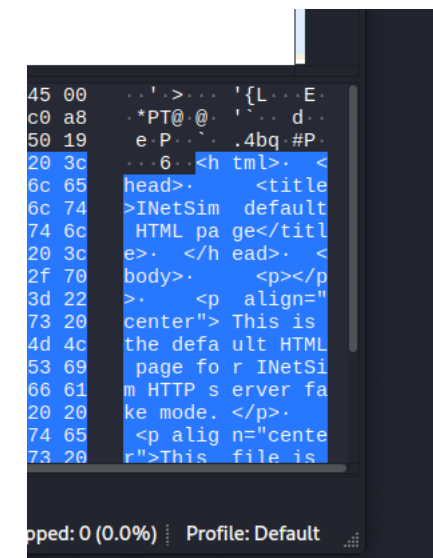
Possiamo notare nella fig. 2 come i pacchetti non siano criptati.



No.	Time	Source	Destination	Protocol	Length
56	9.867487486	SkyUk_81:f5:68	Broadcast	0x7374	60
57	10.166674568	PcsCompu_80:3e:92	Broadcast	ARP	60
58	10.166695035	PcsCompu_7b:4c:bf	PcsCompu_80:3e:92	ARP	42
59	10.167318509	192.168.32.101	192.168.32.100	TCP	60
60	10.167335664	192.168.32.100	192.168.32.101	TCP	60
61	10.167923285	192.168.32.101	192.168.32.100	TCP	60
62	10.168466404	192.168.32.101	192.168.32.100	HTTP	449
63	10.168475578	192.168.32.100	192.168.32.101	TCP	54
64	10.179802144	192.168.32.100	192.168.32.101	TCP	204
65	10.184090765	192.168.32.101	192.168.32.101	HTTP	312
66	10.185025465	192.168.32.101	192.168.32.100	TCP	60
67	10.185333490	192.168.32.101	192.168.32.100	TCP	60
68	10.185358875	192.168.32.100	192.168.32.101	TCP	54
69	10.470913526	SkyUk_81:f5:69	Madge_ab:00:1c	0x7679	60
70	10.480717722	SkyUk_81:f5:69	Madge_ab:00:1c	0x7679	60
71	10.868827586	SkyUk_1c:87:80	Broadcast	0x7374	60

Frame 65: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface 0  
Ethernet II, Src: PcsCompu\_7b:4c:bf (08:00:27:7b:4c:bf), Dst: PcsCompu\_80:3e:92 (08:00:27:80:3e:92)  
Destination: PcsCompu\_80:3e:92 (08:00:27:80:3e:92)  
Source: PcsCompu\_7b:4c:bf (08:00:27:7b:4c:bf)  
Type: IPv4 (0x0800)

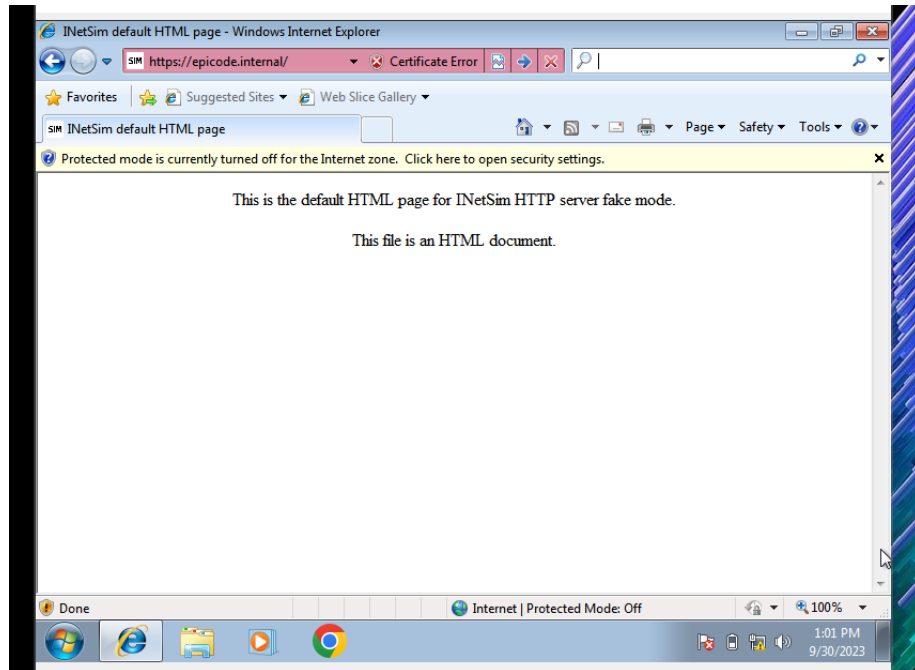
1



2

```
# echo_udp, discard_tcp, discard_udp,  
# quotd_udp, chargen_tcp, chargen_udp,  
# ident, syslog, dummy_tcp, dummy_udp,  
# ftps, irc, https  
#  
start_service dns  
#start_service http  
start_service https  
#start_service smtp  
#start_service smtps  
#start_service pop3
```

1



2

Dopo sono andato a disattivare nella configurazione il server HTTP, attivando il server HTTPS (fig. 1).  
Startando di nuovo la simulazione, si può vedere di nuova la pagina precedente dalla nostra macchina Windows che risponde al nome epicode.intern, questa volta però utilizzando il protocollo HTTPS (fig. 2).

Andando a sniffare nuovamente i pacchetti si nota come in precedenza l'utilizzo del protocollo TCP ma a differenza del protocollo HTTP i pacchetti utilizzano il protocollo TLSv1 (fig. 1).

Come si può vedere dalla fig. 2 utilizzando questo protocollo i dati del pacchetto sono criptati.

Gli indirizzi IP di sorgente e destinazione con i loro rispettivi MAC address rimangono invariati (fig. 1)

Come si può vedere dalla fig. 2 utilizzando questo protocollo i dati del pacchetto sono criptati.  
Gli indirizzi IP di sorgente e destinazione con i loro rispettivi MAC address rimangono invariati (fig. 1)

Gli indirizzi IP di sorgente e destinazione con i loro rispettivi MAC address rimangono invariati (fig. 1)

65	12.093067706	SkyUk_1c:87:80	Broadcast	0x7374	0
66	12.117105064	SkyUk_09:86:aa	Madge_ab:00:1c	0x7679	0
67	12.673848358	PcsCompu_80:3e:92	Broadcast	ARP	0
68	12.673865268	PcsCompu_7b:4c:bf	PcsCompu_80:3e:92	ARP	4
69	12.674932689	192.168.32.101	192.168.32.100	TCP	0
70	12.674978772	192.168.32.100	192.168.32.101	TCP	0
71	12.676003798	192.168.32.101	192.168.32.100	TCP	0
72	12.677225266	192.168.32.101	192.168.32.100	TLSv1	13
73	12.677239648	192.168.32.100	192.168.32.101	TCP	5
74	12.701153984	192.168.32.100	192.168.32.101	TLSv1	13
75	12.705266855	192.168.32.101	192.168.32.100	TLSv1	13
76	12.706596967	192.168.32.100	192.168.32.101	TLSv1	13
77	12.718652443	PcsCompu_80:3e:92	Broadcast	ARP	0
78	12.902337669	192.168.32.101	192.168.32.100	TCP	0
79	12.907352148	SkyUk_09:86:aa	Madge_ab:00:1c	0x7679	0
80	13.056064684	SkyUk_81:f5:69	Broadcast	0x7380	0

```

> Frame 72: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bi
Ethernet II, Src: PcsCompu_80:3e:92 (08:00:27:80:3e:92), Dst: PcsCom
  > Destination: PcsCompu_7b:4c:bf (08:00:27:7b:4c:bf)
    > Source: PcsCompu_80:3e:92 (08:00:27:80:3e:92)
      Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.10
> Transmission Control Protocol, Src Port: 49200, Dst Port: 443, Seq:
> Transport Layer Security

```

1

```
32.101
Win=64320 Len=0

45 00      ..'[L... '...>...E.
c0 a8      ...=@... 7... e..
50 18      d.0...Q wy.qi.P.
5f 03      @).b... ..C...
2f d5      .e..3].* N..."./
f3 16      .....>.. }..TiV..
c0 14      ...../5
00 1e      .....2.8 .....
00 0a      .....
          .....
```

2

**Grazie per l'attenzione.**

