

PROJECT MODULE 1

FATIMA NARDA NICOLETT RODRIGUEZ ORTEGA

1) TRACCIA E REQUISITI

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows 7) 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.

- Kali Linux □ IP 192.168.32.100
- Windows 7 □ IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

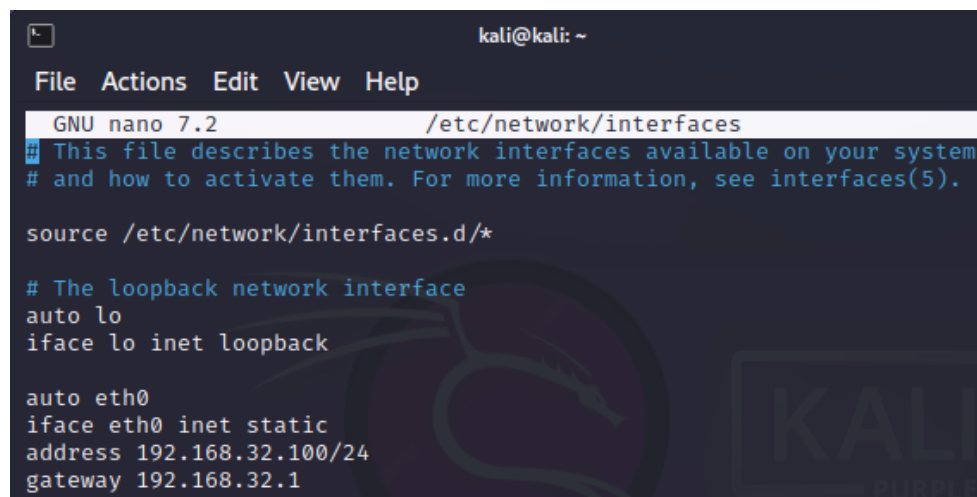
2) RISOLUZIONE

• IMPOSTAZIONE IP

KALI LINUX

Aperto il terminale e lanciando il comando:

"`sudo nano /etc/network/interfaces`" possiamo modificare il file e impostare l'indirizzo IP 192.168.32.100.



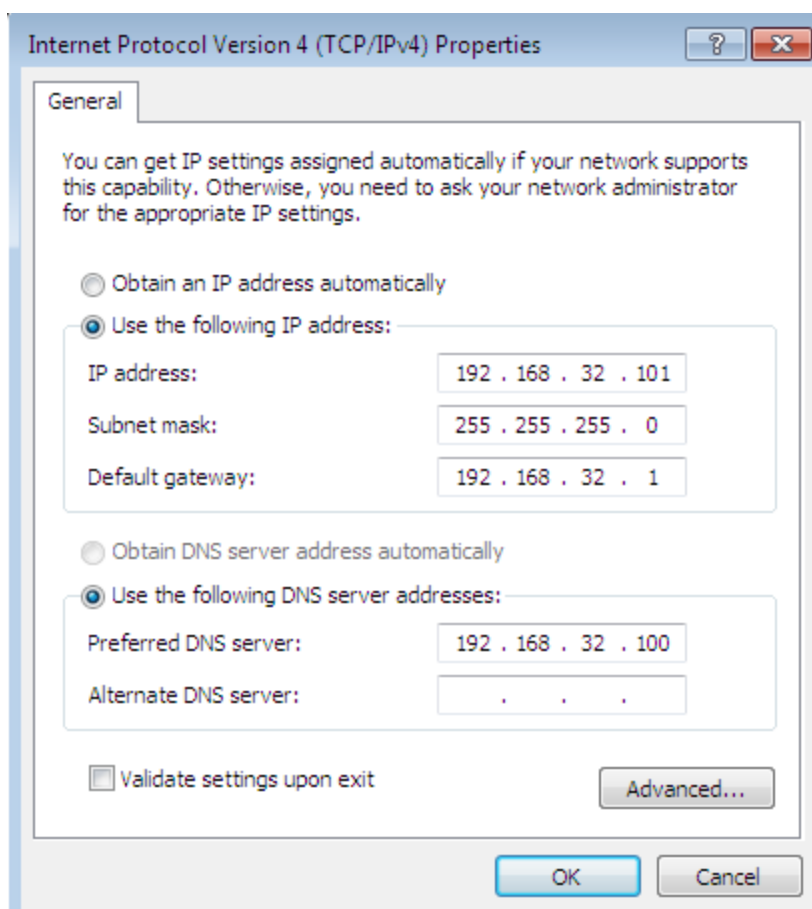
```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/network/interfaces  
# This file describes the network interfaces available on your system  
# and how to activate them. For more information, see interfaces(5).  
  
source /etc/network/interfaces.d/*  
  
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
auto eth0  
iface eth0 inet static  
address 192.168.32.100/24  
gateway 192.168.32.1
```

Dopo aver salvato con *ctrl+o* ed esserci esciti dal file con *ctrl+x*, lanciamo il comando: “ *ifconfig* ” per assicurarci che le modifiche siano state salvate con successo.

```
(kali㉿kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255  
    inet6 fe80::a00:27ff:fe64:a07e prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:64:a0:7e txqueuelen 1000 (Ethernet)  
    RX packets 144 bytes 11636 (11.3 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 87 bytes 9490 (9.2 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

WINDOWS 7

Accedendo alle impostazioni di rete dal Control Panel, impostiamo l'IP *192.168.32.101* e l'IP del DNS server, che è quello di Kali Linux, nelle proprietà d'Internet Protocol Version 4 (TCP/IPv4).



Poi nel Command Prompt, lanciamo il comando “ *ipconfig /all* ” per confermare che le modifiche siano state salvate correttamente.

```
Command Prompt

Connection-specific DNS Suffix . : 
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Physical Address. . . . . : 08-00-27-11-A7-30
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::a44b:e15e:9538:ce55%11(Preferred)
IPv4 Address. . . . . : 192.168.32.101(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.32.1
DHCPv6 IAID . . . . . : 235405351
DHCPv6 Client DUID. . . . . : 00-01-00-01-2C-CF-39-D9-08-00-27-11-A7-30

DNS Servers . . . . . : 192.168.32.100
NetBIOS over Tcpip. . . . . : Enabled
```

• CONFIGURAZIONE DNS, HTTPS E HTTP SERVER

Tornando a Kali Linux, nel terminale lanciamo il comando:
“*sudo nano /etc/inetsim/inetsim.conf*” e commentiamo tutti i servizi tranne
DNS, HTTP e HTTPS.

```
kali@kali: ~
File Actions Edit View Help
GNU nano 7.2 /etc/inetsim/inetsim.conf *
#
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
#start_service ntp
#start_service finger
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

Impostiamo come IP del service bind address 0.0.0.0

```
#####  
# 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
```

Poi impostiamo il *DNS default IP* con l'IP di Kali Linux e creiamo un DNS statico che collega l'IP di Kali Linux con l'hostname epicode.internal.

```
#####  
# dns_default_ip  
#  
# Default IP address to return with DNS replies  
#  
# Syntax: dns_default_ip <IP address>  
#  
# Default: 127.0.0.1  
#  
dns_default_ip 192.168.32.100
```

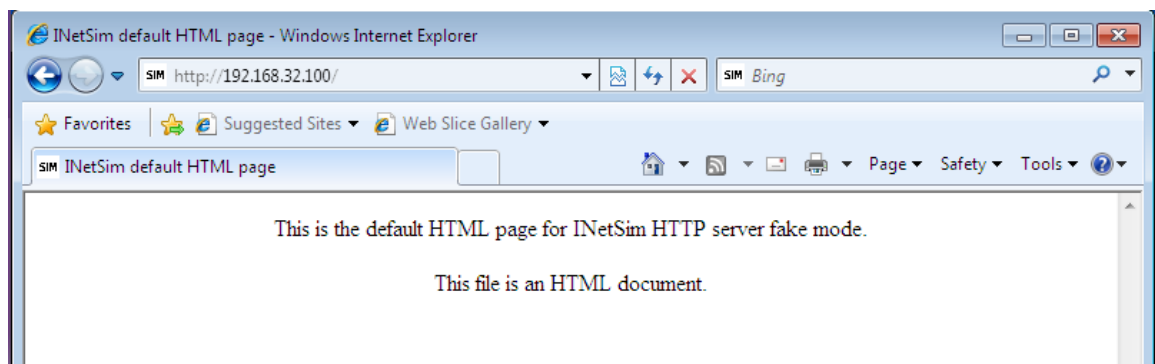
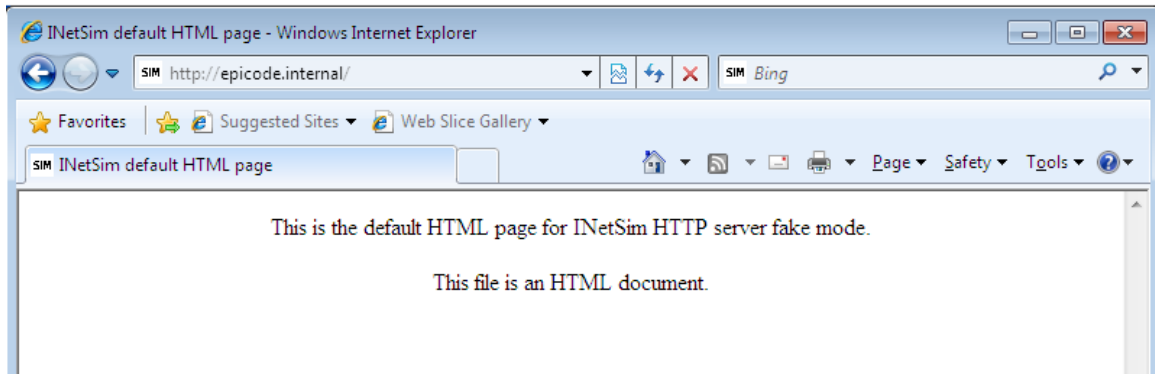
```
#####  
# dns_static  
#  
# Static mappings for DNS  
#  
# Syntax: dns_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
#dns_static www.foo.com 10.10.10.10  
#dns_static ns1.foo.com 10.70.50.30  
#dns_static ftp.bar.net 10.10.20.30  
dns_static epicode.internal 192.168.32.100
```

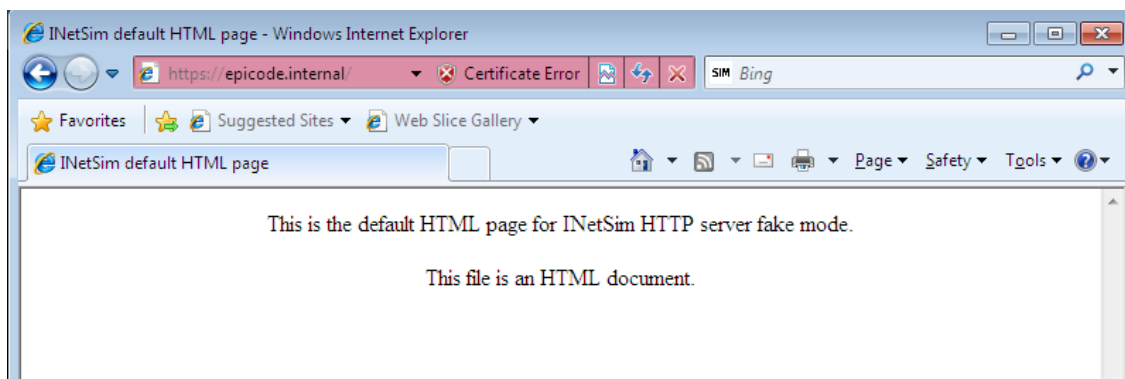
Salviamo le modifiche apportate nel file con ctrl+o e chiudiamolo con ctrl+x, poi lanciamo il comando "sudo inetsim" per simulare un servizio di rete e controllare che i servizi funzionino correttamente.

```
(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 5449) ==
Session ID: 5449
Listening on: 0.0.0.0
Real Date/Time: 2023-11-19 23:18:59
Fake Date/Time: 2023-11-19 23:18:59 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 5463)
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l
ine 399.
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l
ine 399.
* http_80_tcp - started (PID 5464)
* https_443_tcp - started (PID 5465)
done.
Simulation running.
```

● RICHIESTA HTTP E HTTPS

Apriamo Internet Explorer da Windows 7 per mandare richieste GET ai HTTP e HTTPS servers con l'IP address di Kali Linux e con l'hostname *epicode.internal*.





● CATTURA PACCHETTI CON WIRESHARK

Utilizzando Wireshark su Kali Linux, intercettiamo il *traffico* di HTTP e HTTPS relativi a *epicode.internal*. Sull'interfaccia di rete *eth0* si può visualizzare il traffico dei *pacchetti*.

HTTPS TRAFFIC

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	PcsCompu_11:a7:30	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2 0.000019331	PcsCompu_64:a0:7e	PcsCompu_11:a7:30	ARP	42	192.168.32.100 is at 08:00:27:64:a0:7e
3 0.000286385	192.168.32.101	192.168.32.100	TCP	66	49171 → 443 [SYN] Seq=0 Win=6192 Len=0 MSS=1460 WS=4 SACK_PERM
4 0.000318641	192.168.32.100	192.168.32.101	TCP	66	443 → 49171 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PER
5 0.000563869	192.168.32.101	192.168.32.100	TCP	60	49171 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6 0.001289570	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
7 0.001330239	192.168.32.100	192.168.32.101	TCP	54	443 → 49171 [ACK] Seq=1 Ack=162 Win=64128 Len=0
8 0.263196281	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9 0.295174351	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10 0.295279590	192.168.32.100	192.168.32.101	TCP	54	443 → 49171 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
11 0.298044282	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12 0.317547075	PcsCompu_11:a7:30	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101

Si può notare la presenza del *TLS (Transport Layer Security) protocol*, che ha lo scopo di garantire la protezione di dati sensibili nella comunicazione tra server e client. Infatti dopo la *three-hand-shake*, che stabilisce una connessione affidabile, il client invia un pacchetto *TLSv1 "Client Hello"* e il server risponde con un pacchetto *TLSv1 "Server Hello"*, senza rendere visibile il tipo di richiesta inviato dal client nè la risposta del server perchè i dati sono criptati.

```

▶ Frame 6: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on interface eth0, id 0
▼ Ethernet II, Src: PcsCompu_11:a7:30 (08:00:27:11:a7:30), Dst: PcsCompu_64:a0:7e (08:00:27:64:a0:7e)
  ▶ Destination: PcsCompu_64:a0:7e (08:00:27:64:a0:7e)
  ▶ Source: PcsCompu_11:a7:30 (08:00:27:11:a7:30)
    Type: IPv4 (0x0800)
  ▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
  ▶ Transmission Control Protocol, Src Port: 49171, Dst Port: 443, Seq: 1, Ack: 1, Len: 161
  ▶ Transport Layer Security

```

Nonostante ciò, è possibile visualizzare gli IP e MAC address di Windows 7 e Kali Linux: si trovano nell'header, i primi del livello 3 e i secondi nel livello 2 (più in basso si possono visualizzare i MAC address delle due macchine per confronto e accertare l'esattezza dei dati intercettati).

HTTP TRAFFIC

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	PcsCompu_11:a7:30	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2 0.000027123	PcsCompu_64:a0:7e	PcsCompu_11:a7:30	ARP	42	192.168.32.100 is at 08:00:27:64:a0:7e
3 0.000226835	192.168.32.101	192.168.32.100	TCP	66	49202 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SA=
4 0.000257405	192.168.32.100	192.168.32.101	TCP	66	80 → 49202 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=
5 0.000440173	192.168.32.101	192.168.32.100	TCP	60	49202 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6 0.000722739	192.168.32.101	192.168.32.100	HTTP	333	GET / HTTP/1.1
7 0.000737187	192.168.32.100	192.168.32.101	TCP	54	80 → 49202 [ACK] Seq=1 Ack=280 Win=64128 Len=0
8 0.056174289	192.168.32.100	192.168.32.101	TCP	204	80 → 49202 [PSH, ACK] Seq=1 Ack=280 Win=64128 Len=150
9 0.065104503	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
10 0.065520369	192.168.32.101	192.168.32.100	TCP	60	49202 → 80 [ACK] Seq=280 Ack=410 Win=65292 Len=0
11 0.065681714	192.168.32.101	192.168.32.100	TCP	60	49202 → 80 [FIN, ACK] Seq=280 Ack=410 Win=65292 Len=0
12 0.065698850	192.168.32.100	192.168.32.101	TCP	54	80 → 49202 [ACK] Seq=410 Ack=281 Win=64128 Len=0

A differenza del traffico di HTTPS, subito dopo la three-hand-shake si può vedere il tipo di richiesta che ha inviato il client al server, che è GET, perchè non è presente il TLS protocol a criptare i dati.

```
▶ Frame 9: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface eth0, id 0
▼ Ethernet II, Src: PcsCompu_64:a0:7e (08:00:27:64:a0:7e), Dst: PcsCompu_11:a7:30 (08:00:27:11:a7:30)
  ▶ Destination: PcsCompu_11:a7:30 (08:00:27:11:a7:30)
  ▶ Source: PcsCompu_64:a0:7e (08:00:27:64:a0:7e)
  Type: IPv4 (0x0800)
```

```
▶ Hypertext Transfer Protocol
  ▶ Line-based text data: text/html (10 lines)
    <html>\n
    <head>\n
      <title>INetSim default HTML page</title>\n
    </head>\n
    <body>\n
      <p></p>\n
      <p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>\n
      <p align="center">This file is an HTML document.</p>\n
    </body>\n
  </html>\n
```

Infatti, nel pacchetto HTTP “HTTP/1.1 200 OK (text/html)” si può vedere il contenuto della pagina html.

MAC ADDRESS

```
Description . . . . . : Intel(R) PRO/1000
Physical Address . . . . . : 08-00-27-11-A7-30
DHCIP Enabled . . . . . : No
```

Su Windows 7, si trova lanciando il comando “*ipconfig /all*” nel command prompt.

```
inet6 fe80::a00:27ff:fe64:a07e
```

Su Kali Linux, si trova lanciando il comando “*ifconfig*” nel terminale.