

REPORT SIMULAZIONE RETE COMPLESSA WEEK 1

Per prima cosa abbiamo impostato i nuovi ip sulle macchine virtuali;

Kali(immagine 1)

Windows (immagine 2)

immagine 1

```
(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:fec7:e136 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:c7:e1:36 txqueuelen 1000 (Ethernet)
    RX packets 85 bytes 17427 (17.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 10476 (10.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali) ~$
```

immagine 2

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versione 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\chris>ipconfig

Configurazione IP di Windows

Scheda Ethernet Connessione alla rete locale (LAN):

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::2455:94c2:3998:8d14%11
    Indirizzo IPv4. . . . . : 192.168.32.101
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . : 192.168.32.1

Scheda Tunnel isatap.{1D2D397C-C6D8-4A02-ACB0-59DA1843EACB}:

    Stato supporto. . . . . : Supporto disconnesso
    Suffisso DNS specifico per connessione:

C:\Users\chris>S_
```

Successivamente mediante l'aiuto di InetSim (che ci permetterà di analizzare i comportamenti della rete), andiamo ad impostare sul terminale linux di kali andiamo a creare un server DNS e un server HTTPS, impostando:

indirizzo di collegamento al servizio

```
61 # service_bind_address
62 #
63 # IP address to bind services to
64 #
65 # Syntax: service_bind_address <IP address>
66 #
67 # Default: 127.0.0.1
68 #
69 #service_bind_address 10.10.10.1
70 service_bind_address 192.168.32.100
71
```

assegnamo un indirizzo ip al Server DNS

```
198 #####
199 # dns_default_ip
200 #
201 # Default IP address to return with DNS replies
202 #
203 # Syntax: dns_default_ip <IP address>
204 #
205 # Default: 127.0.0.1
206 #
207 #dns_default_ip 10.10.10.1
208 dns_default_ip 192.168.32.100
209
```

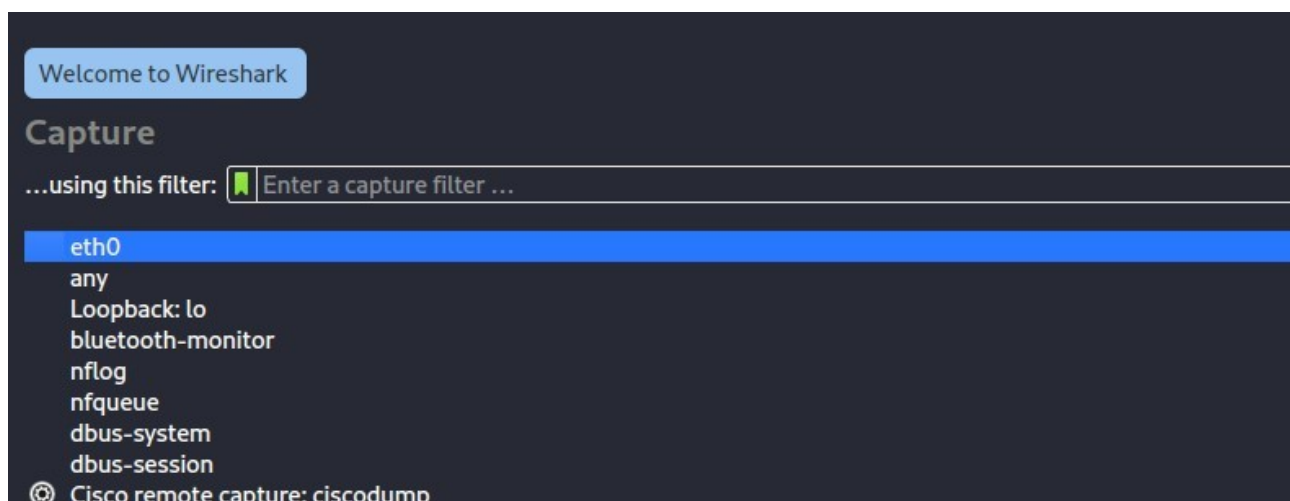
infine assegnamo il nome “epicode.internal” al nostro server

```
235 # dns_static
236 #
237 # Static mappings for DNS
238 #
239 # Syntax: dns_static <fqdn hostname> <IP address>
240 #
241 # Default: none
242 #
243 #dns_static www.foo.com 10.10.10.10
244 #dns_static ns1.foo.com 10.70.50.30
245 #dns_static ftp.bar.net 10.10.20.30
246 dns_static epicode.internal 192.168.32.100
```

Dopo aver impostato tutti i parametri , sul terminale di kali eseguiamo la simulazione che ci permetterà di il nostro ambiente di analisi

```
(kali@kali)-[~]
$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & T
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.co
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 14953) ==
Session ID:      14953
Listening on:    192.168.32.100
Real Date/Time:  2023-05-06 12:08:34
Fake Date/Time: 2023-05-06 12:08:34 (Delta: 0 sec
Forking services ...
* dns_53_tcp_udp - started (PID 14963)
* irc_6667_tcp - started (PID 14973)
* finger_79_tcp - started (PID 14975)
* discard_9_udp - started (PID 14985)
* time_37_udp - started (PID 14979)
* daytime_13_tcp - started (PID 14980)
* echo_7_tcp - started (PID 14982)
* smtp_25_tcp - started (PID 14966)
* smtps_465_tcp - started (PID 14967)
* ntp_123_udp - started (PID 14974)
* ident_113_tcp - started (PID 14976)
* daytime_13_udp - started (PID 14981)
* time_37_tcp - started (PID 14978)
* echo_7_udp - started (PID 14983)
* quotd_17_udp - started (PID 14987)
* discard_9_tcp - started (PID 14984)
* quotd_17_tcp - started (PID 14986)
* chargen_19_udp - started (PID 14989)
* tftp_69_udp - started (PID 14972)
* syslog_514_udp - started (PID 14977)
* dummy_1_tcp - started (PID 14990)
* dummy_1_udp - started (PID 14991)
* http_80_tcp - started (PID 14964)
* chargen_19_tcp - started (PID 14988)
* pop3_110_tcp - started (PID 14968)
* pop3s_995_tcp - started (PID 14969)
* ftp_21_tcp - started (PID 14970)
* https_443_tcp - started (PID 14965)
* ftps_990_tcp - started (PID 14971)
done.
Simulation running.
```

per analizzare il traffico di pacchetti utilizzeremo wireshark sulla sorgente eth0 che identifica il nostro ambiente virtuale



Avviata l'analisi sulla macchina relativa a windows 7, sul web browser nello spazio del codice URL "<https://epicode.internal>" dandoci questa risposta:



mentre su wireshark partirà un analisi di pacchetti di invio e risposta, e se andiamo ad analizzare un pacchetto che segue il protocollo tcp vedremo che il nostro dns sta usando un protocollo sicuro con la porta 443 (vedi immagine) che garantisce una connessione crittografata.

| | | | | | |
|----|--------------|----------------|----------------|-----|--|
| 1 | 0.000000000 | 192.168.32.101 | 192.168.32.100 | TCP | 66 49305 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM |
| 2 | 0.000062456 | 192.168.32.100 | 192.168.32.101 | TCP | 66 443 → 49305 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 3 | 0.000196270 | 192.168.32.101 | 192.168.32.100 | TCP | 60 49305 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0 |
| 5 | 0.000881412 | 192.168.32.100 | 192.168.32.101 | TCP | 54 443 → 49305 [ACK] Seq=1 Ack=130 Win=64128 Len=0 |
| 10 | 0.247687747 | 192.168.32.100 | 192.168.32.101 | TCP | 113 [TCP Retransmission] 443 → 49305 [PSH, ACK] Seq=1320 Ack=264 Win=64128 Len=59 |
| 11 | 0.247685799 | 192.168.32.101 | 192.168.32.100 | TCP | 60 49305 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0 |
| 12 | 0.247841553 | 192.168.32.101 | 192.168.32.100 | TCP | 66 [TCP Dup ACK 11#1] 49305 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0 SLE=1320 SRE=1379 |
| 34 | 11.435532159 | 192.168.32.101 | 192.168.32.100 | TCP | 66 49306 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM |
| 35 | 11.435572294 | 192.168.32.100 | 192.168.32.101 | TCP | 66 80 → 49306 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 36 | 11.435684979 | 192.168.32.101 | 192.168.32.100 | TCP | 60 49306 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| 38 | 11.435769055 | 192.168.32.100 | 192.168.32.101 | TCP | 54 80 → 49306 [ACK] Seq=1 Ack=218 Win=64128 Len=0 |
| 39 | 11.446739179 | 192.168.32.100 | 192.168.32.101 | TCP | 204 80 → 49306 [PSH, ACK] Seq=1 Ack=218 Win=64128 Len=150 [TCP segment of a reassembled PDU] |
| 41 | 11.448249216 | 192.168.32.101 | 192.168.32.100 | TCP | 60 49306 → 80 [ACK] Seq=218 Ack=410 Win=65280 Len=0 |

- › Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
- › Ethernet II, Src: PcsCompu_9b:70:86 (08:00:27:9b:70:86), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)
- › Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
- › Transmission Control Protocol, Src Port: 49305, Dst Port: 443, Seq: 0, Len: 0

Utilizzando il protocollo http sempre sul web browser di windows 7 sempre cercando il nostro server “<http://epicode.internal>”,apparirà una pagina così:

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

Mentre su Wireshark vedremo un analisi di pacchetti dove al contrario della precedente analisi usava la porta 443 ,il protocollo http userà la porta 80:

| | | | |
|----------------|----------------|-----|--|
| 192.168.32.101 | 192.168.32.100 | TCP | 66 49309 → 80 [SYN, Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM |
| 192.168.32.100 | 192.168.32.101 | TCP | 66 80 → 49309 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 192.168.32.101 | 192.168.32.100 | TCP | 60 49309 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0 |
| 192.168.32.100 | 192.168.32.101 | TCP | 54 80 → 49309 [ACK] Seq=1 Ack=412 Win=64128 Len=0 |
| 192.168.32.100 | 192.168.32.101 | TCP | 204 80 → 49309 [PSH, ACK] Seq=1 Ack=412 Win=64128 Len=150 [TCP segment of a reassembled PDU] |
| 192.168.32.101 | 192.168.32.100 | TCP | 60 49309 → 80 [ACK] Seq=412 Ack=410 Win=65292 Len=0 |
| 192.168.32.101 | 192.168.32.100 | TCP | 60 49309 → 80 [FIN, ACK] Seq=412 Ack=410 Win=65292 Len=0 |
| 192.168.32.100 | 192.168.32.101 | TCP | 54 80 → 49309 [ACK] Seq=410 Ack=413 Win=64128 Len=0 |
| 192.168.32.101 | 192.168.32.100 | TCP | 66 49310 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM |
| 192.168.32.100 | 192.168.32.101 | TCP | 66 443 → 49310 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 192.168.32.101 | 192.168.32.100 | TCP | 66 49311 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM |
| 192.168.32.100 | 192.168.32.101 | TCP | 66 443 → 49311 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |

```
▶ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
▶ Ethernet II, Src: PcsCompu_9b:70:86 (08:00:27:9b:70:86), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)
▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
▶ Transmission Control Protocol, Src Port: 49309, Dst Port: 80, Seq: 0, Len: 0
```

come visto nell'immagine del web browser di windows 7 viene mostrata la risposta di inetsim del nostro DNS server impostato precedentemente crittografata che non necessita certificazioni col protocollo http mentre a differenza del protocollo https che stabilisce connessioni crittografate.