



W4D4

PROVA DI FINE MODULO

Traccia e requisiti

Nell'esercizio di oggi metteremo insieme le competenze acquisite finora.
Lo studente verrà valutato sulla base della risoluzione al problema seguente.

Requisiti e servizi:

- 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

Traccia:

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).

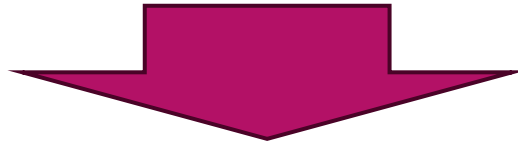
Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS.

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.

Setting del laboratorio (Kali)

Innanzitutto prepariamo il laboratorio in modo da essere in linea con i requisiti richiesti, per cui nel file di configurazione «interfaces» di Kali impostiamo l'indirizzo IP di eth0 e il gateway (come da requisiti indicati) e verifichiamo la configurazione con il comando «ifconfig».

Per Kali intenderemo la macchina virtuale dove è installato l'OS Kali Linux e per Win7 la macchina virtuale dove è installato l'OS Windows 7.



```
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
```

```
(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:feeb:7ef5 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)  
    RX packets 238 bytes 21809 (21.2 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 212 bytes 21514 (21.0 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 24 bytes 1240 (1.2 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 24 bytes 1240 (1.2 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

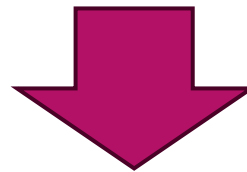
Setting del laboratorio (Kali)

Successivamente modifichiamo il file di configurazione di InetSim su Kali che ci permetterà di attivare i servizi HTTP, HTTPS e DNS su tale server simulato.

Inoltre imposteremo il bind address, cioè collegheremo i servizi del server simulato all'indirizzo IP di Kali facendo sì che quest'ultimo agisca da Server.

I protocolli HTTP e HTTPS permetteranno al server di restituire delle risorse, il DNS di risolvere un hostname e collegarlo ad un indirizzo IP, infatti sul file di configurazione associo all'hostname «epicode.internal» l'indirizzo IP di Kali.

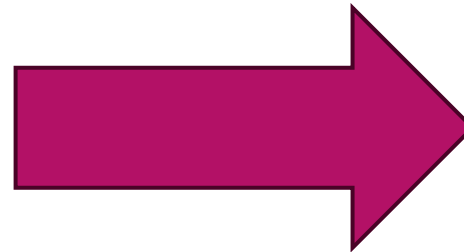
```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetSim/inetSim.conf  
  
#####  
# Main configuration  
#####  
  
#####  
# start_service  
#  
# The services to start  
#  
# Syntax: start_service <service name>  
#  
# Default: none  
#  
# Available service names are:  
# dns, 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
```



```
#####  
# dns_static  
#  
# Static mappings for DNS  
#  
# 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  
#  
#####  
# service_bind_address  
#  
# IP address to bind services to  
#  
# Syntax: service_bind_address <IP address>  
#  
# Default: 127.0.0.1  
#  
service_bind_address 192.168.32.100
```

Setting del laboratorio (Windows 7)

Configuro anche la macchina virtuale di Windows 7 reimpostando l'indirizzo IP come da requisiti e indicando l'indirizzo del DNS (prima configurato su Kali) che mi permetterà di risolvere il nome del dominio utilizzando appunto il servizio DNS attivato su Kali tramite InetSim.
Win7 agirà da Client.

The screenshot shows the 'Internet Protocol Version 4 (TCP/IPv4) Properties' dialog box. The 'General' tab is selected. It contains instructions about automatic IP assignment and manual configuration options. The 'Use the following IP address' option is selected, with the IP address set to 192.168.32.101, subnet mask 255.255.255.0, and default gateway 192.168.32.1. The 'Use the following DNS server addresses' option is also selected, with the preferred DNS server set to 192.168.32.100 and the alternate DNS server field empty. There are checkboxes for 'Obtain an IP address automatically', 'Obtain DNS server address automatically', and 'Validate settings upon exit'. At the bottom, there are 'OK', 'Cancel', and 'Advanced...' buttons.

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

☐ Obtain an IP address automatically

☒ Use the following IP address:

IP address: 192 . 168 . 32 . 101

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 32 . 1

☐ Obtain DNS server address automatically

☒ Use the following DNS server addresses:

Preferred DNS server: 192 . 168 . 32 . 100

Alternate DNS server: . . .

☐ Validate settings upon exit

Advanced...

OK Cancel

Simulazione richiesta Http e Https

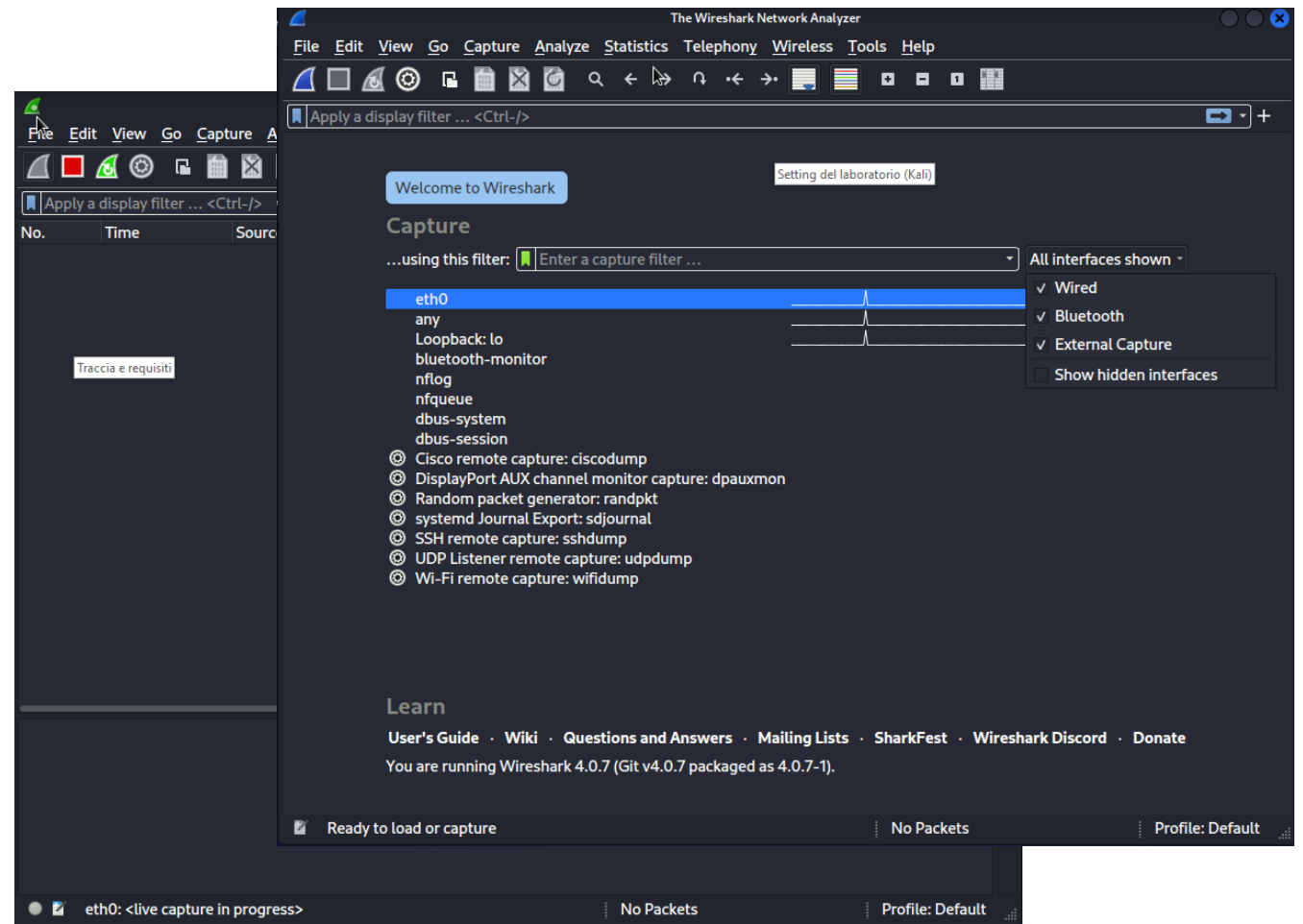
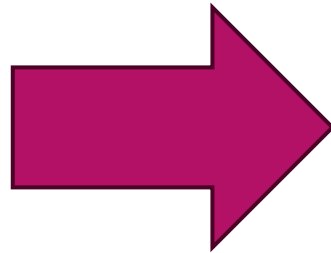
Avendo attivato e collegato i servizi Server di InetSim all'indirizzo IP di Kali possiamo avviare InetSim da terminale in modo da permettere al Server di «ascoltare» le richieste da parte del Client.



```
kali@kali: ~  
File Actions Edit View Help  
(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 2245) ≡  
Session ID: 2245  
Listening on: 192.168.32.100  
Real Date/Time: 2023-11-17 03:09:56  
Fake Date/Time: 2023-11-17 03:09:56 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 2255)  
* http_80_tcp - started (PID 2256)  
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.  
* https_443_tcp - started (PID 2257)  
done.  
Simulation running.  
□
```

Simulazione richiesta Http e Https

I pacchetti partiranno da Win7 e arriveranno a Kali(e viceversa), ciò vuol dire che passeranno per la eth0 di Kali per cui se su Wireshark settiamo eth0 vedremo il traffico che passerà per tale interfaccia sia in entrata che in uscita.

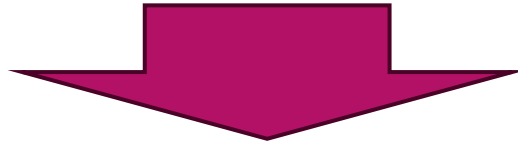


Simulazione richiesta Http e Https

Il primo obiettivo è quello di evidenziare i MAC address di sorgente e destinazione sulle rispettive VM per poi verificarne la corrispondenza nei pacchetti intercettati con Wireshark.

Su Kali per vedere il Mac digiteremo «ifconfig» mentre su Win7 digiteremo «ipconfig».

Il MAC address di Kali è 08:00:27:cb:7e:f5 , quello di Win7 è 08:00:27:72:3d:ab .



```
C:\Windows\system32\cmd.exe

Host Name . . . . . : Win7
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : 
   Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . : 08-00-27-72-3D-A5
   DHCP Enabled. . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::eced:20b1:bdc7:8c9c%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-CD-8C-9E-08-00-27-72-3D-A5

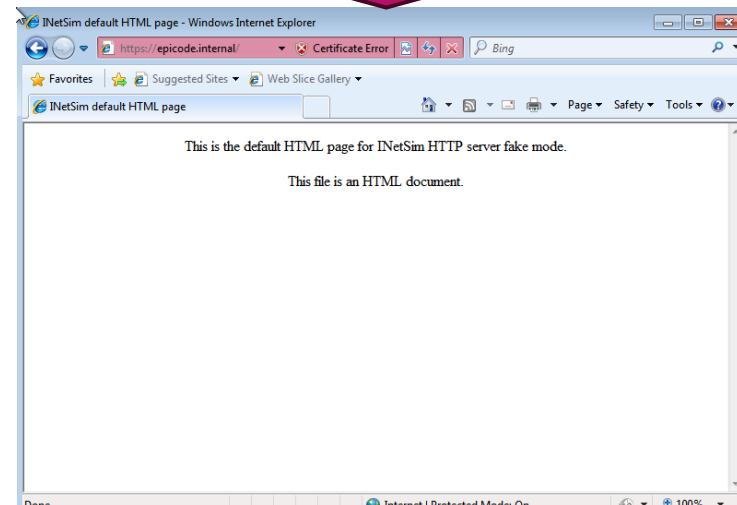
   DNS Servers . . . . . : 192.168.32.100
   NetBIOS over Tcpip. . . . . : Enabled

Tunnel adapter isatap.{A9D2D267-E774-4BDD-8BFB-67D343211161}:
```

```
kali@kali: ~
File Actions Edit View Help
(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:feeb:7ef5 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 1 bytes 247 (247.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 19 bytes 2634 (2.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```


Simulazione richiesta Http e Https

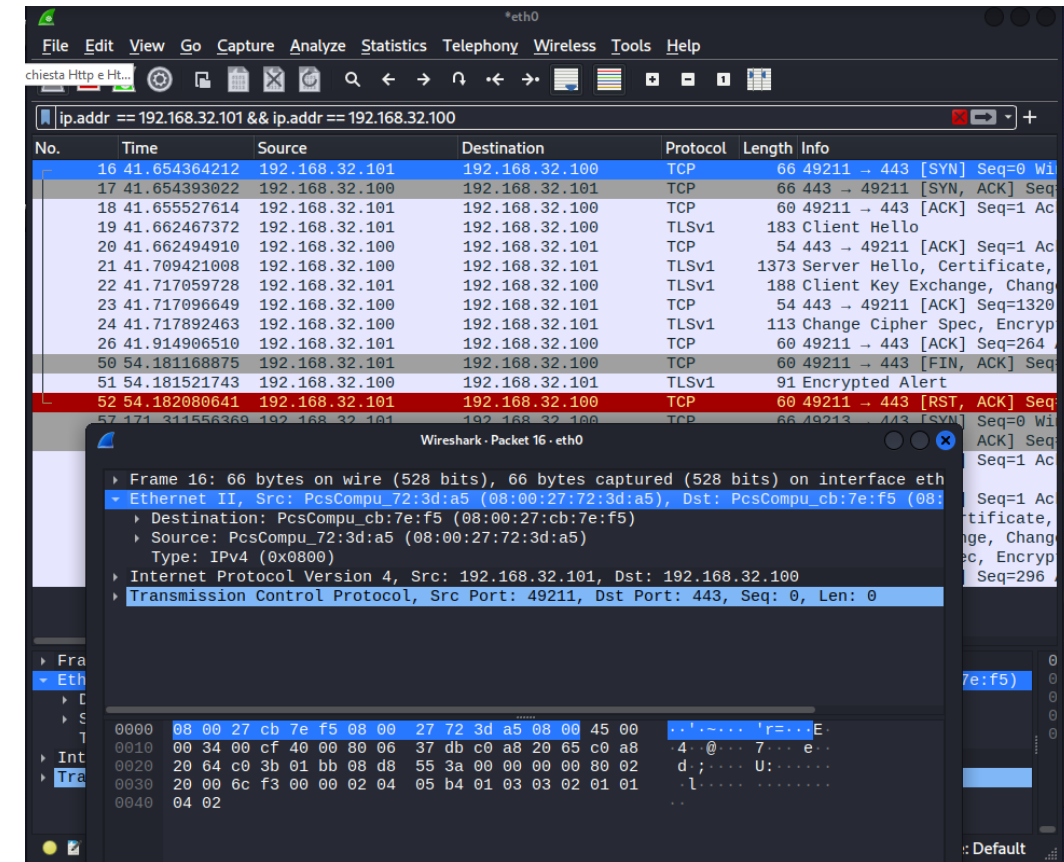
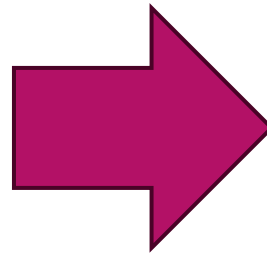
Apro il browser «Internet Explorer» e nella barra degli indirizzi prima digito l'hostname «epicode.internal» usando il protocollo HTTPs e verifico se funziona la risoluzione dell'hostname «epicode.internal» grazie al servizio DNS attivato sul Server. Nel frattempo verifico il traffico generato su Wireshark isolando i risultati relativi agli Ip delle due VM. La risorsa presente sul server viene visualizzata correttamente, questo vuol dire la risoluzione dell'hostname è avvenuta correttamente è che grazie al servizio HTTPs è stata fornita la risorsa.



Simulazione richiesta Http e Https

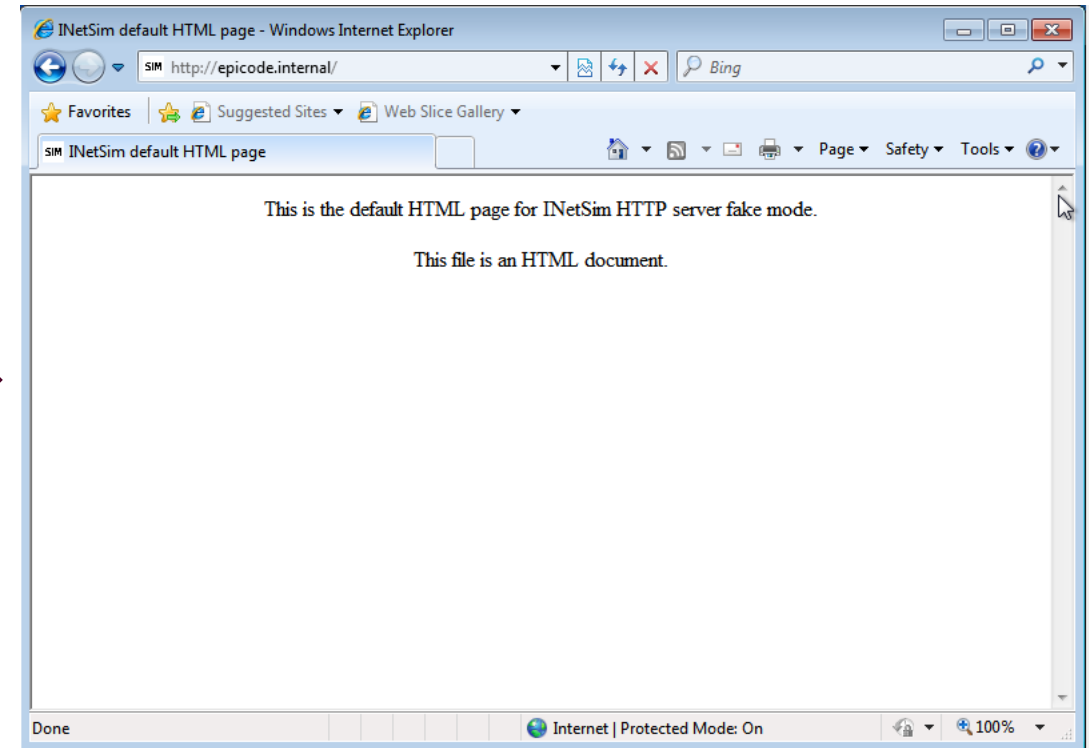
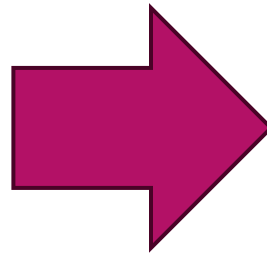
Digito il filtro «ip.addr == 192.168.32.101 && ip.addr == 192.168.32.100» su Wireshark per isolare la visione del traffico tra le sole due VM(Kali e Win7) interessate. Tramite browser su Win7 mi connetto a «epicode.internal» osservo la generazione di pacchetti per stabilire la connessione tra le due VM secondo il processo di «three-way-handshake».

Nella prima delle tre fasi di quest'ultima osserviamo la richiesta SYN da parte di Win7 (evidenziata nella figura accanto): abbiamo detto che è Win7 a fare la richiesta per cui il source MAC dovrebbe corrispondere a quello della scheda di rete di tale macchina, infatti è 08:00:27:72:3d:ab . Il destination MAC address deve corrispondere a quello di Kali e infatti è 08:00:27:cb:7e:f5.



Simulazione richiesta Http e Https

Ritorno su Win7 e questa volta la richiesta verrà effettuata tramite il protocollo HTTP.
La risoluzione dell'hostname avviene con successo.
Vediamo cosa succede su Wireshark.



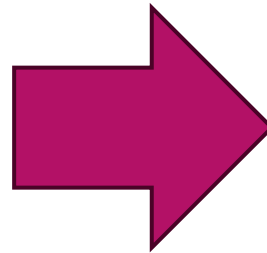
Simulazione richiesta Http e Https

Anche qui c'è il processo di three-way-handshake.

Nulla varia per quanto riguarda la corrispondenza dei MAC address.

Qui possiamo visualizzare anche il protocollo DNS in azione, il Server leggerà l'hostname e grazie a questo servizio fornirà il corrispondente indirizzo IP.

La differenza più importante però è che qui a differenza dell'HTTPs abbiamo la possibilità di vedere il contenuto della richiesta e della risorsa trasmessa in chiaro.



The image shows a Wireshark packet capture on the interface eth0. The filter is set to 'ip.addr == 192.168.32.101 && ip.addr == 192.168.32.100'. The packet list shows various protocols including TCP, HTTP, and DNS. The packet details pane is expanded for packet 100, showing the Ethernet II header, Internet Protocol Version 4 header, and the first part of the Transmission Control Protocol header.

No.	Time	Source	Destination	Protocol	Length	Info
103	183.895392547	192.168.32.101	192.168.32.100	TCP	60	49215 → 443 [FIN, ACK] Seq=189
104	183.895417642	192.168.32.100	192.168.32.101	TCP	54	443 → 49215 [ACK] Seq=189
129	1322.6050358...	192.168.32.101	192.168.32.100	TCP	66	49216 → 80 [SYN] Seq=0 Win
130	1322.6050650...	192.168.32.100	192.168.32.101	TCP	66	80 → 49216 [SYN, ACK] Seq=
131	1322.6056894...	192.168.32.101	192.168.32.100	TCP	60	49216 → 80 [ACK] Seq=1 Ac
132	1322.6060181...	192.168.32.101	192.168.32.100	HTTP	451	GET / HTTP/1.1
133	1322.6060276...	192.168.32.100	192.168.32.101	TCP	54	80 → 49216 [ACK] Seq=1 Ac
134	1322.6202783...	192.168.32.100	192.168.32.101	TCP	204	80 → 49216 [PSH, ACK] Seq=
135	1322.6228935...	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/htm
136	1322.6245349...	192.168.32.101	192.168.32.100	TCP	60	49216 → 80 [ACK] Seq=398
137	1322.6248435...	192.168.32.101	192.168.32.100	TCP	60	49216 → 80 [FIN, ACK] Seq=
138	1322.6248665...	192.168.32.100	192.168.32.101	TCP	54	80 → 49216 [ACK] Seq=410
139	1322.6711849...	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x1444 A e
140	1322.6929434...	192.168.32.100	192.168.32.101	DNS	92	Standard query response 0
141	1322.6987960...	192.168.32.101	192.168.32.100	TCP	66	49217 → 80 [SYN] Seq=0 Win
142	1322.6988368...	192.168.32.100	192.168.32.101	TCP	66	80 → 49217 [SYN, ACK] Seq=
143	1322.6999467...	192.168.32.101	192.168.32.100	TCP	60	49217 → 80 [ACK] Seq=1 Ac
144	1322.6999468...	192.168.32.101	192.168.32.100	HTTP	327	GET /favicon.ico HTTP/1.1
145	1322.6999780...	192.168.32.100	192.168.32.101	TCP	54	80 → 49217 [ACK] Seq=1 Ac
146	1322.7152425...	192.168.32.100	192.168.32.101	TCP	207	80 → 49217 [PSH, ACK] Seq=
147	1322.7173690...	192.168.32.100	192.168.32.101	HTTP	252	HTTP/1.1 200 OK (image/x
148	1322.7180846...	192.168.32.101	192.168.32.100	TCP	60	49217 → 80 [ACK] Seq=274
149	1322.7187738...	192.168.32.101	192.168.32.100	TCP	60	49217 → 80 [FIN, ACK] Seq=
150	1322.7188017...	192.168.32.100	192.168.32.101	TCP	54	80 → 49217 [ACK] Seq=353

Frame 100: 235 bytes on wire (1880 bits), 235 bytes captured (1880 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: PcsCompu_72:3d:a5 (08:00:27:72:3d:a5)
Destination: PcsCompu_72:3d:a5 (08:00:27:72:3d:a5)
Source: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
Transmission Control Protocol, Src Port: 443, Dst Port: 49215, Seq: 1379, Ack: 733, Len: 181
Transport Layer Security

Ethernet (eth), 14 bytes Packets: 171 · Displayed: 61 (35.7%) Profile: Default

Simulazione richiesta Http e Https: differenze tra i due protocolli e conclusioni

Nell'immagine di fianco possiamo infatti osservare che cliccando sul pacchetto possiamo visionare il contenuto della pagina restituita.

Nel protocollo HTTPs infatti interviene nel processo di handshake il protocollo TLS che prevede lo scambio della chiave tra Client e Server per cifrare i contenuti, nella richiesta HTTP ciò non avviene.

Altra differenza è che l'HTTP usa la porta 80 mentre l'HTTPs usa la 443.

In sostanza col protocollo HTTPs su Wireshark vedremo solo connessione stabilita tramite il protocollo TCP e TLS mentre in quella HTTP anche il protocollo DNS e soprattutto il contenuto della richiesta.

