

## Progetto 1 – Epicode

18/11/2023

Studente: Maria Ludovica Tartaglia

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 eventuali differenze, se presenti.

Requisiti e servizi:

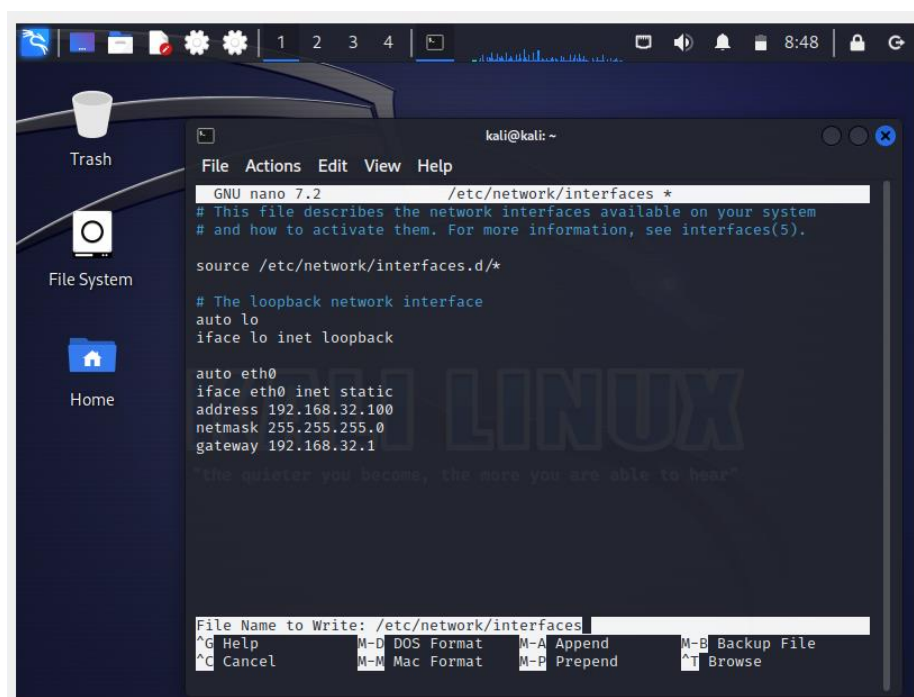
- Kali Linux: IP 192.168.32.100
- Windows 7: IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

## SVOLGIMENTO

Per lo svolgimento della seguente esercitazione, utilizzeremo le macchine virtuali Kali Linux e Windows7 configurate in Oracle Virtual Box. Le due macchine dovranno comunicare tra loro, pertanto come prima cosa andranno configurati gli indirizzi IP di ciascuna macchina così come indicatoci nella traccia.

### Configurazione IP statico nella macchina Kali Linux:

Per prima cosa andiamo a dare da terminale il comando ***sudo nano /etc/network/interfaces***, di modo che possa essere modificato l'indirizzo IP inserendo ***192.168.32.100***, come mostrato nella slide che segue:

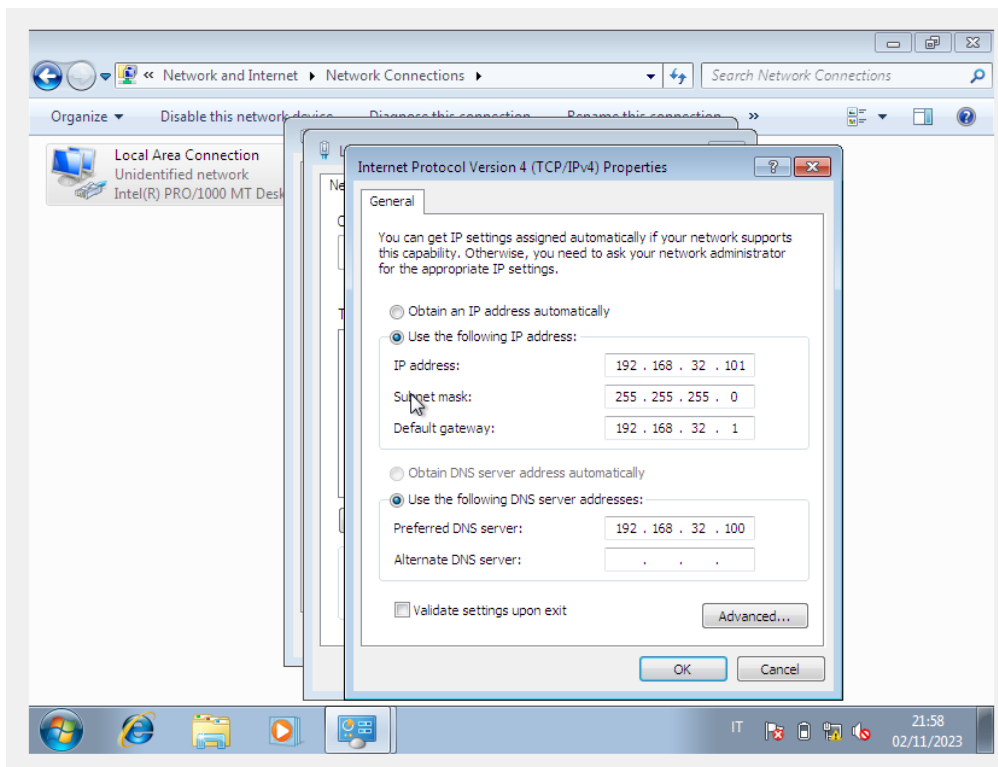


Salviamo l'impostazione appena data alla macchina e la riavviamo per far sì che rimanga configurata come appena fatto. A riprova lanceremo nel terminale il comando **ifconfig** e vedremo che l'impostazione dell'IP statico previsto è andata a buon fine.

```
(kali㉿kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> m  
      inet 192.168.32.100  netmask 255.255.255.0  
5
```

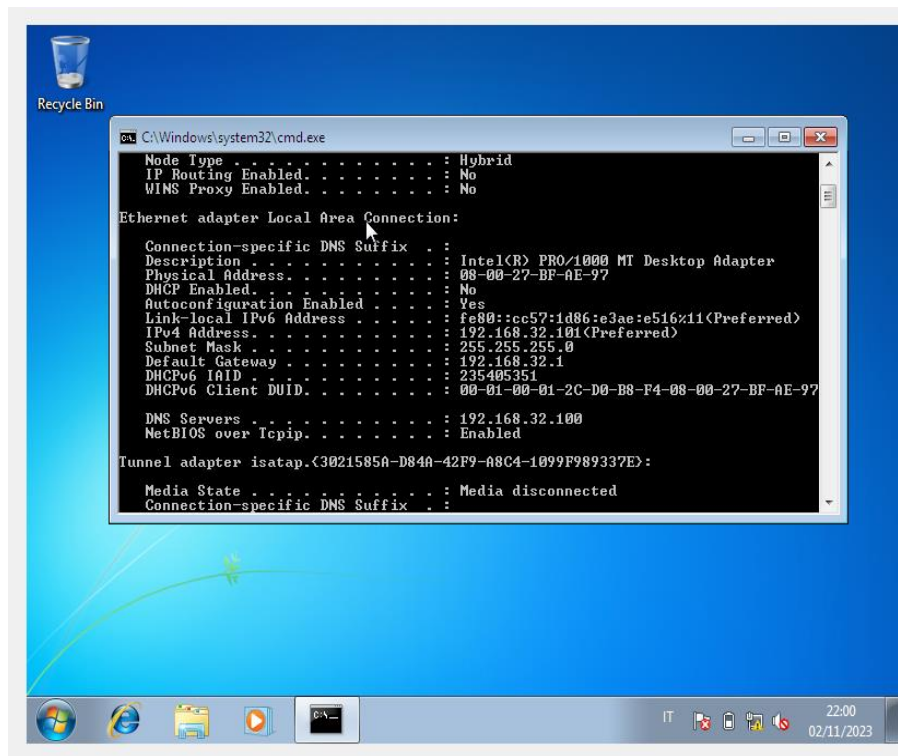
### Configurazione IP statico e DNS nella macchina Windows7:

Andremo ad effettuare la stessa configurazione nella seconda macchina virtuale in uso. Tramite le impostazioni di rete inseriamo l'indirizzo IP ed il DNS come nella seguente slide:

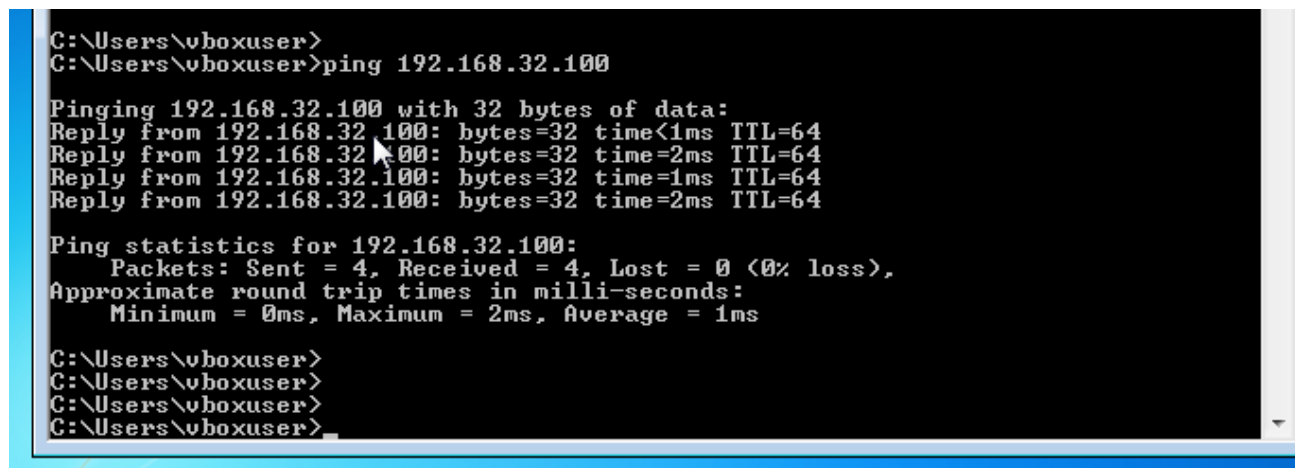


Notiamo come l'impostazione del server DNS sia lo stesso indirizzo IP della macchina Kali Linux, in quanto essa stessa funzionerà da server http e https e da server DNS stesso.

Verifichiamo che anche nella macchina Windows7 le impostazioni appena date siano state eseguite e salvate...



...E procediamo a questo punto a connettere le macchine virtuali tra loro, di modo che possano comunicare. Per fare ciò, attiveremo su Windows7 il comando **ping 192.168.32.100**



A questo punto le macchine avranno configurato l'indirizzo IP come richiesto dalla traccia e comunicano ufficialmente tra loro.

### Configurazione server DNS e http/https:

La traccia richiede che Windows7 possa effettuare una richiesta ad http e https al dominio Epicode.internal. Dovremo pertanto configurare i tre server attraverso l'utilizzo di **Inetsim**, che permetterà di avviare una simulazione tra i server stessi.

Andremo quindi sul terminale di Kali Linux andando a configurare Inetsim attraverso il comando **sudo nano /etc/inetsim/inetsim.conf**, come nelle slides che seguono:

```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf  
# 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  
#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  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf *  
#  
# Default: inetsim.org  
#  
#dns_default_domainname some.domain  
#  
#####  
# 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  
#  
#####  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

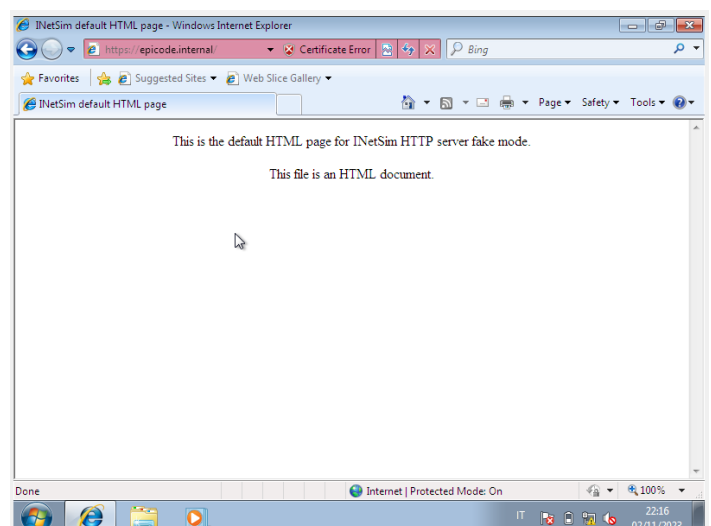
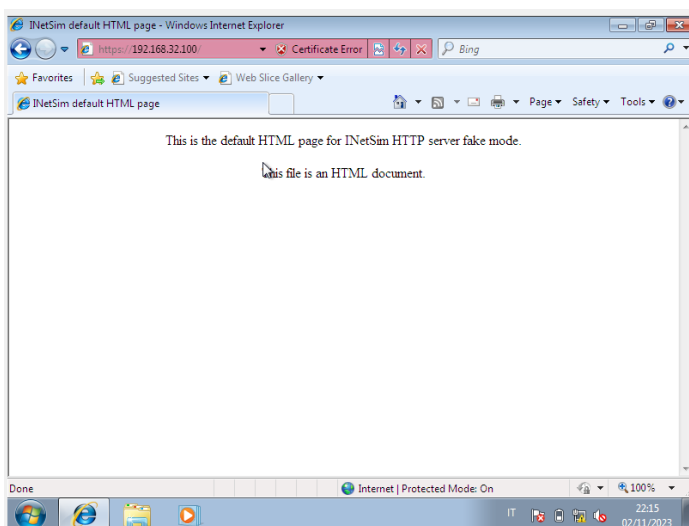
```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf *  
#  
# Syntax: service_bind_address <IP address>  
#  
# Default: 127.0.0.1  
#  
service_bind_address 0.0.0.0  
#  
#####  
# service_run_as_user  
#  
# User to run services  
#  
# Syntax: service_run_as_user <username>  
#  
# Default: inetsim  
#service_run_as_user nobody  
#  
#####  
# service_max_childs  
#  
# Maximum number of child processes (parallel connections)  
#  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Nella prima slide abbiamo levato il commento a tutti i servizi che andremo ad utilizzare, nella seconda andiamo a rinominare il DNS statico con il dominio **Epicode.internal** e l'indirizzo IP della macchina Kali Linux, ed infine impostiamo un **bind address** di tipo **0.0.0.0**, che permetterà la ricezione da qualsiasi altro dispositivo.

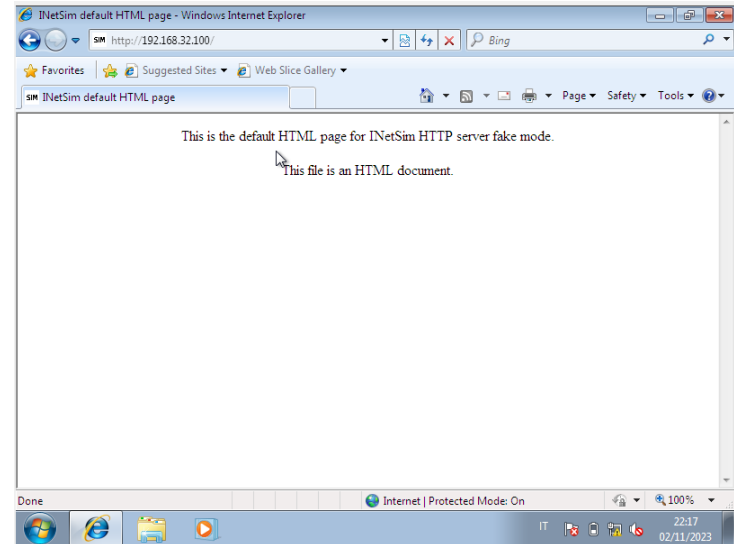
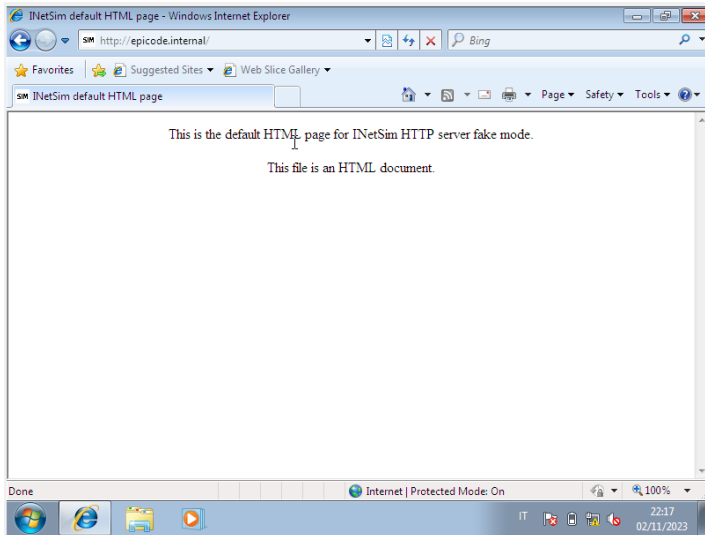
Dopo aver salvato le impostazioni date e riavviato la macchina, col comando **sudo inetsim** mandiamo in svolgimento la simulazione che ci confermi che tutti i sistemi sono operativi e funzionanti.

```
kali@kali: ~  
File Actions Edit View Help  
$ service inetsim stop  
  
(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 66003) ==  
Session ID: 66003  
Listening on: 192.168.32.100  
Real Date/Time: 2023-11-18 09:07:54  
Fake Date/Time: 2023-11-18 09:07:54 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 66013)  
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 66014)  
* https_443_tcp - started (PID 66015)  
done.  
Simulation running.  
█
```

A questo punto verifichiamo la corretta comunicazione tra le macchine ed il funzionamento della simulazione avviata con Inetsim. Apriamo il browser di Windows7, Internet Explorer, ed inseriamo la nostra richiesta, prima con https e poi con http, utilizzando come controprova sia l'hostname Epicode.internal che l'indirizzo IP 192.168.32.100:







## Intercettare il traffico tramite Wireshark:

Per intercettare il traffico nel browser di Windows7, utilizzeremo lo strumento Wireshark sulla macchina Kali Linux. Internet Explorer effettuerà una richiesta sia ad http che ad https, che risponderanno con una schermata HTML come nelle slides precedenti.

Sul dispositivo Wireshark, andremo a cercare la trasmissione dei dati nella sezione eth0, la quale ci darà i seguenti risultati nell'intercettazione della richiesta https:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_bf:ae:97	Broadcast	ARP	60	Who has 1
2	0.000037224	PcsCompu_cb:7e:f5	PcsCompu_bf:ae:97	ARP	42	192.168.3
3	0.000822235	192.168.32.101	192.168.32.100	TCP	66	49188 → 4
4	0.000890418	192.168.32.100	192.168.32.101	TCP	66	443 → 491
5	0.001800803	192.168.32.101	192.168.32.100	TCP	60	49188 → 4
6	0.002860137	192.168.32.101	192.168.32.100	TLSv1	215	Client He
7	0.002939116	192.168.32.100	192.168.32.101	TCP	54	443 → 491
8	0.200488562	192.168.32.100	192.168.32.101	TLSv1	1373	Server He
9	0.216101783	192.168.32.101	192.168.32.100	TLSv1	188	Client Ke
10	0.216136477	192.168.32.100	192.168.32.101	TCP	54	443 → 491

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured  
 Ethernet II, Src: PcsCompu\_bf:ae:97 (08:00:27:bf:ae:97)  
 Address Resolution Protocol (request)

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_bf:ae:97	Broadcast	ARP	60	Who has 1
2	0.000015725	PcsCompu_cb:7e:f5	PcsCompu_bf:ae:97	ARP	42	192.168.3
3	0.000344066	192.168.32.101	192.168.32.100	TCP	66	49192 → 4
4	0.000367572	192.168.32.100	192.168.32.101	TCP	66	443 → 491
5	0.000713515	192.168.32.101	192.168.32.100	TCP	60	49192 → 4
6	0.001520065	192.168.32.101	192.168.32.100	TLSv1	215	Client He
7	0.001545026	192.168.32.100	192.168.32.101	TCP	54	443 → 491
8	0.128643495	192.168.32.100	192.168.32.101	TLSv1	1373	Server He
9	0.144244275	192.168.32.101	192.168.32.100	TLSv1	188	Client Ke
10	0.144351787	192.168.32.100	192.168.32.101	TCP	54	443 → 491

Frame 3: 66 bytes on wire (528 bits), 66 bytes captured  
 Ethernet II, Src: PcsCompu\_bf:ae:97 (08:00:27:bf:ae:97)  
 Destination: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5)  
 Source: PcsCompu\_bf:ae:97 (08:00:27:bf:ae:97)  
 Type: IPv4 (0x0800)  
 Internet Protocol Version 4, Src: 192.168.32.101, Dst:  
 Transmission Control Protocol, Src Port: 49192, Dst Port:

	Destination	Protocol	Length	Info
e:97	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
e:f5	PcsCompu_bf:ae:97	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
1	192.168.32.100	TCP	66	49192 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=14
0	192.168.32.101	TCP	66	443 → 49192 [SYN, ACK] Seq=0 Ack=1 Win=64240
1	192.168.32.100	TCP	60	49192 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
1	192.168.32.100	TLSv1	215	Client Hello
0	192.168.32.101	TCP	54	443 → 49192 [ACK] Seq=1 Ack=162 Win=64128 Len=
0	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange
1	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encry
0	192.168.32.101	TCP	54	443 → 49192 [ACK] Seq=1320 Ack=296 Win=64128

```

C:\Windows\system32\cmd.exe

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : 
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Physical Address. . . . . : 08-00-27-BF-AE-97
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::cc57:1d86:e3ae:e516%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-D0-B8-F4-08-00-27-BF-AE-97

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

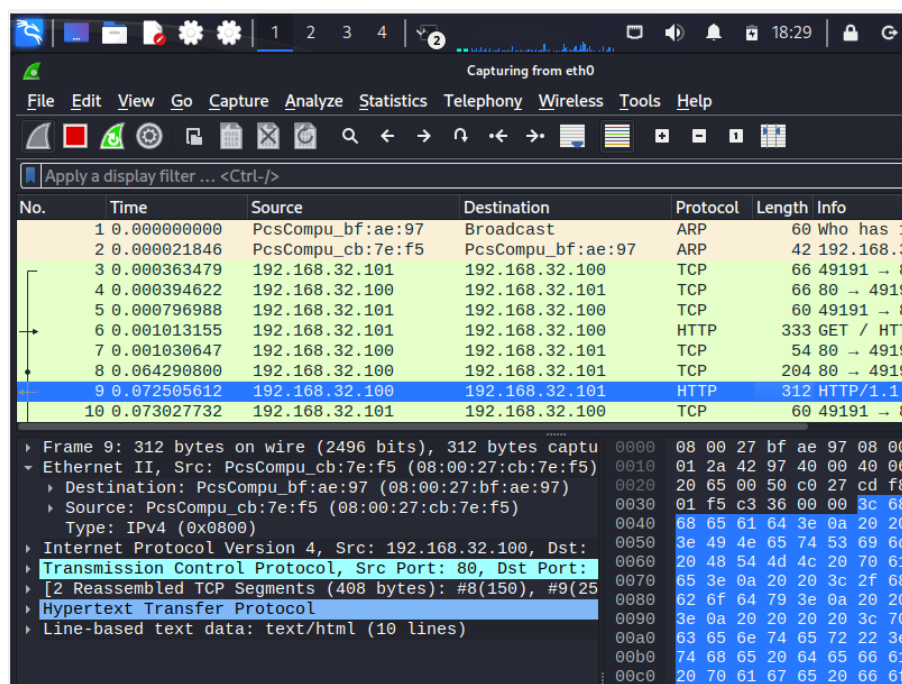
Tunnel adapter {3021585A-D84A-42F9-A8C4-1099F989337E}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft ISATAP Adapter
Physical Address. . . . . : 00-00-00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No

```

Nella sequenza di slides di cui sopra possiamo notare la cattura di pacchetti TCP che rappresentino lo scambio di pacchetti tra Source e Destination rispettivamente tra Windows7 e Kali Linux (in questo caso utilizzato come nostro server) e viceversa, con porta 443 considerata di default per il server https; vediamo anche i MAC address di entrambe le macchine, rappresentati da Destination (08: 00: 27: cb: 7e: f5) e Source (08: 00: 27: bf: ae: 97); da ultimo, possiamo osservare come ci sia uno scambio a tre nei pacchetti TCP che inviano richiesta di sincronizzazione (SYN), accettazione (SYN, ACK) e da ultimo si prende atto della Sincronizzazione avvenuta (ACK), nonché lo scambio di chiavi rappresentato dal Server Key Exchange. Il protocollo TLS configura una richiesta tra Server e Client, che simpaticamente sarà accordata con un “Hello Client, Hello Server”. Ricordiamo infine che essendo la cattura proveniente dal server https, le informazioni saranno criptate.

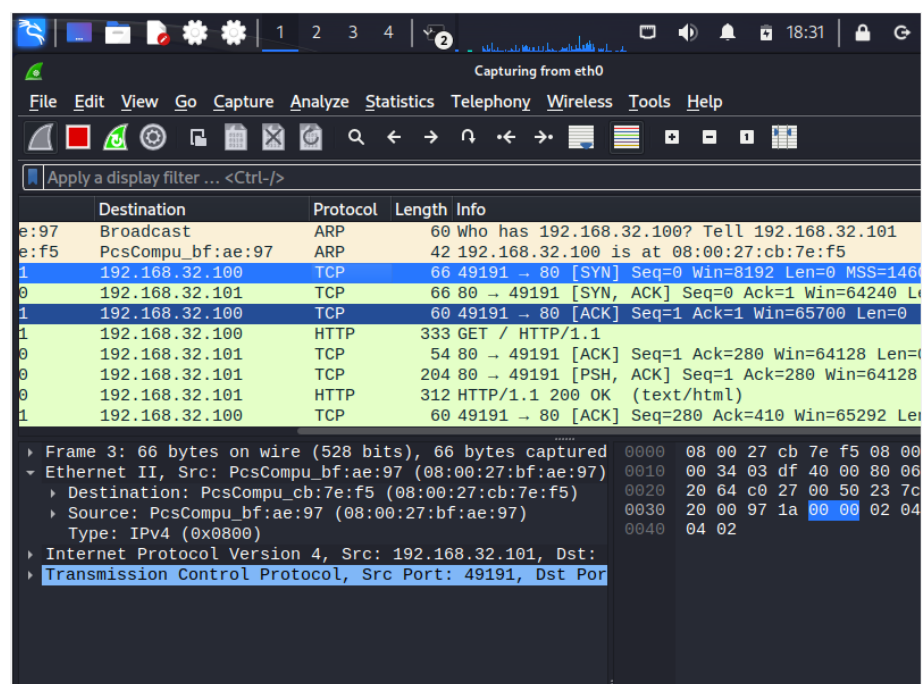
Procediamo alla stessa cattura tramite server http:



Wireshark interface showing a packet capture on eth0. The packet list shows a GET request from 192.168.32.101 to 192.168.32.100 on port 80. The packet details pane shows the structure of the HTTP request, including the GET method and the path /1.1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_bf:ae:97	Broadcast	ARP	60	Who has 1
2	0.000021846	PcsCompu_cb:7e:f5	PcsCompu_bf:ae:97	ARP	42	192.168.3
3	0.000363479	192.168.32.101	192.168.32.100	TCP	66	49191 → 8
4	0.000394622	192.168.32.100	192.168.32.101	TCP	66	80 → 4919
5	0.000796988	192.168.32.101	192.168.32.100	TCP	60	49191 → 8
6	0.001013155	192.168.32.101	192.168.32.100	HTTP	333	GET / HTTP
7	0.001030647	192.168.32.100	192.168.32.101	TCP	54	80 → 4919
8	0.064290800	192.168.32.100	192.168.32.101	TCP	204	80 → 4919
9	0.072505612	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1
10	0.073027732	192.168.32.101	192.168.32.100	TCP	60	49191 → 8

Frame 9: 312 bytes on wire (2496 bits), 312 bytes captured on interface eth0, 312 bytes from 192.168.32.100 to 192.168.32.101 on interface eth0  
 Ethernet II, Src: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: PcsCompu\_bf:ae:97 (08:00:27:bf:ae:97)  
 Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101  
 Transmission Control Protocol, Src Port: 80, Dst Port: 49191  
 [2 Reassembled TCP Segments (408 bytes): #8(150), #9(258)]  
 Hypertext Transfer Protocol  
 Line-based text data: text/html (10 lines)



Wireshark interface showing a packet capture on eth0. The packet list shows a SYN-ACK exchange and an HTTP GET request. The packet details pane shows the structure of the HTTP request, including the GET method and the path /1.1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_bf:ae:97	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000021846	PcsCompu_cb:7e:f5	PcsCompu_bf:ae:97	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.000363479	192.168.32.101	192.168.32.100	TCP	66	49191 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
4	0.000394622	192.168.32.100	192.168.32.101	TCP	66	80 → 49191 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
5	0.000796988	192.168.32.101	192.168.32.100	TCP	60	49191 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.001013155	192.168.32.101	192.168.32.100	HTTP	333	GET / HTTP/1.1
7	0.001030647	192.168.32.100	192.168.32.101	TCP	54	80 → 49191 [ACK] Seq=1 Ack=280 Win=64128 Len=0
8	0.064290800	192.168.32.100	192.168.32.101	TCP	204	80 → 49191 [PSH, ACK] Seq=1 Ack=280 Win=64128 Len=0
9	0.072505612	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
10	0.073027732	192.168.32.101	192.168.32.100	TCP	60	49191 → 80 [ACK] Seq=280 Ack=410 Win=65292 Len=0

Frame 3: 66 bytes on wire (528 bits), 66 bytes captured on interface eth0, 66 bytes from 192.168.32.101 to 192.168.32.100 on interface eth0  
 Ethernet II, Src: PcsCompu\_bf:ae:97 (08:00:27:bf:ae:97), Dst: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5)  
 Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100  
 Transmission Control Protocol, Src Port: 49191, Dst Port: 80  
 [Sequence 1, Window 65292, Len 0, Win 65292, Len 0]



Qui possiamo notare che la richiesta GET avviene in automatico, senza richiesta di “connessione” tra i due device a differenza del server https. La porta utilizzata è la porta 80, di default per il server http. A differenza del server https, con la richiesta effettuata su http non abbiamo informazioni criptate a causa della mancanza del protocollo TLS per la negoziazione di uno scambio di connessioni criptato. Sarà quindi possibile, per chi intercetterà i pacchetti trasmessi, vedere ogni tipologia di informazione della pagina HTML che è stata aperta.