


Progetto M1 Daniele Atzori 16/03/2024

**Esercizio**
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.

2

Per prima cosa imposto i due IP statici .100 per Kali e .101 per Windows 7. Su Windows 7 imposto anche il DNS con l'IP di Kali .100

```
Microsoft Windows [Versione 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\Daniele>ipconfig

Configurazione IP di Windows

Scheda Ethernet Connessione alla rete locale (LAN):

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::d0bb:ee81:e401:5a72%
11  Indirizzo IPv4 . . . . . : 192.168.32.101
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . : 192.168.32.1

Scheda Tunnel isatap.{BB8BB52C-DC4A-45AA-BE61-ECEAC685ACB6}:

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

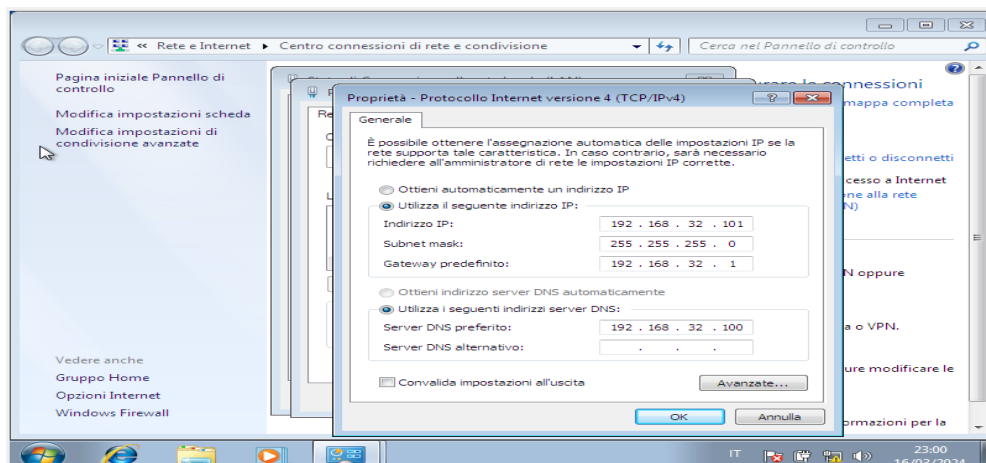
C:\Users\Daniele>
```

```
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).

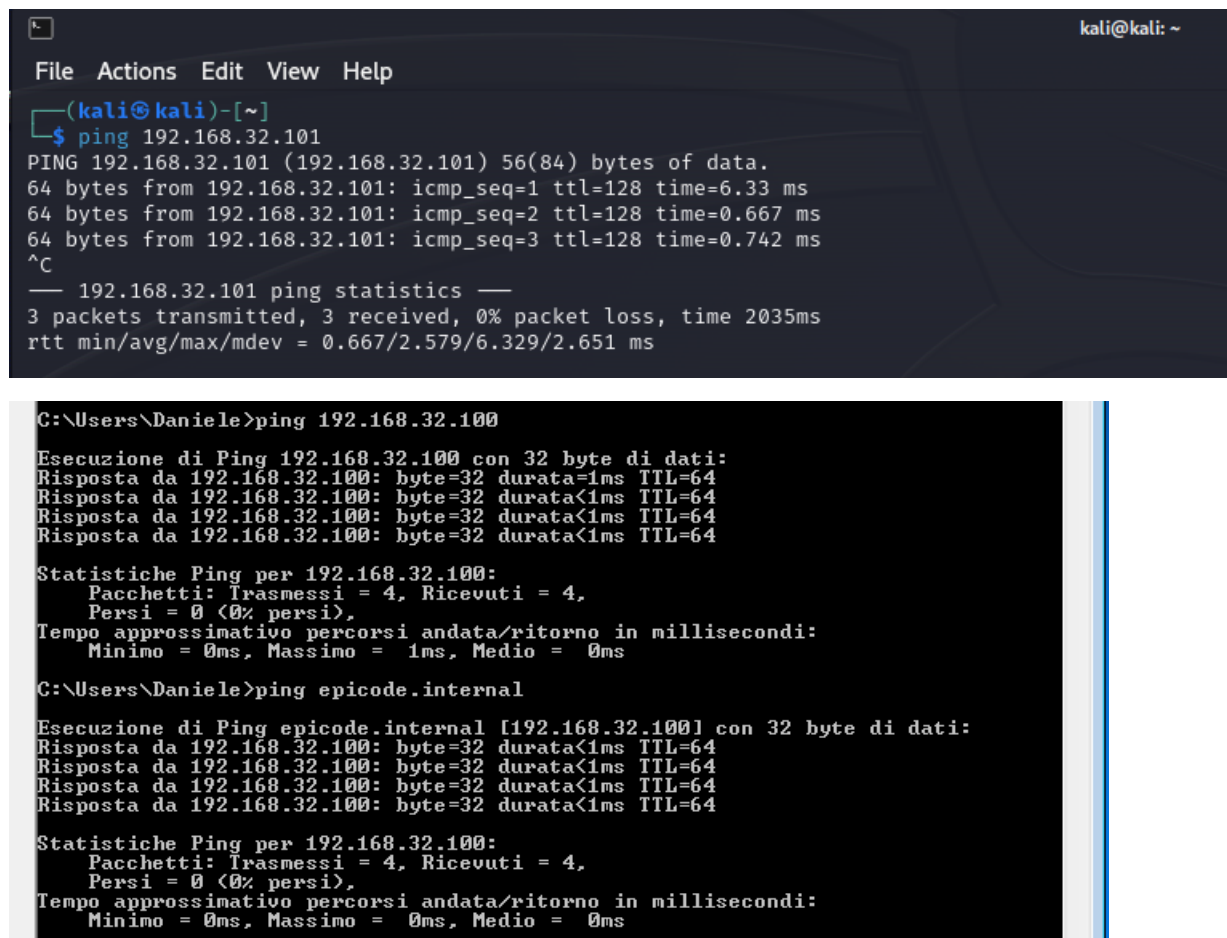
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.32.100
gateway 192.168.32.101
```



In seguito faccio pingare le due macchine:



```
(kali@kali)-[~]
$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=6.33 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=0.667 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=0.742 ms
^C
— 192.168.32.101 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.667/2.579/6.329/2.651 ms
```

```
C:\Users\Daniele>ping 192.168.32.100

Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
        Minimo = 0ms, Massimo = 1ms, Medio = 0ms

C:\Users\Daniele>ping epicode.internal

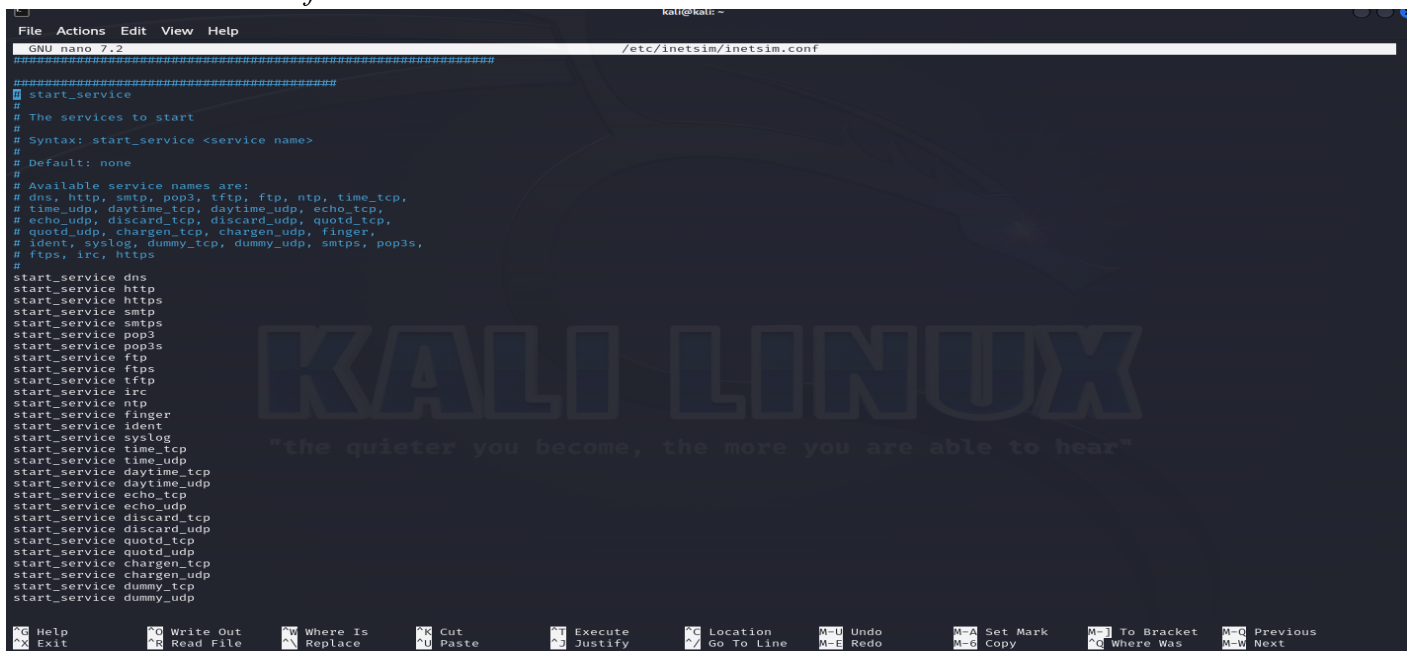
Esecuzione di Ping epicode.internal [192.168.32.100] con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
        Minimo = 0ms, Massimo = 0ms, Medio = 0ms
```

A Windows ho fatto pingare prima con IP e poi con DNS (dopo aver attivato inetsim) per confermare che inetsim fosse settato in maniera corretta. Nel caso inetsim fosse stato spento o settato in maniera errata, ci sarebbe stato comunque il ping con l'IP, ma non con l'hostname. Questo è un passaggio successivo ma gli screen e la spiegazione li sto scrivendo dopo aver già finito l'esercizio, quindi questo passaggio non è esattamente in ordine cronologico con il resto.

Settaggio Inetsim

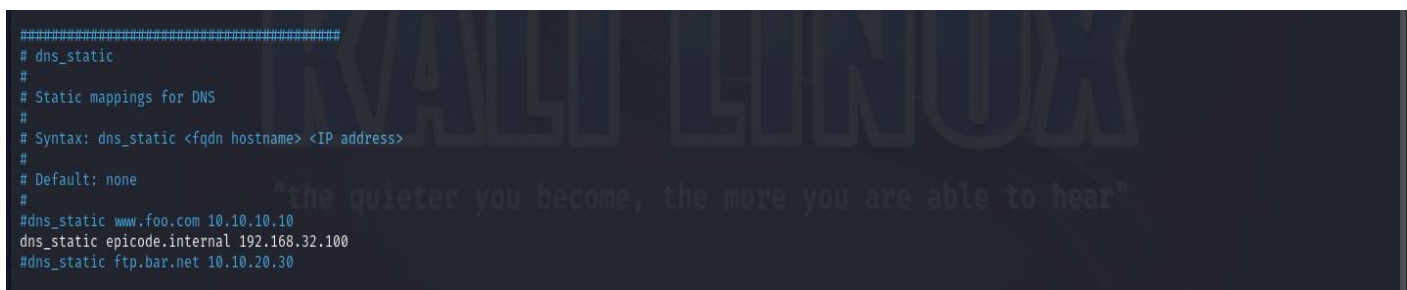
Riguardo il settaggio di inetsim, si accede eseguendo su terminale Kali il comando “*sudo nano /etc/inetsim/inetsim.conf*”



```
#####
# 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
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
start_service discard_udp
start_service quotd_tcp
start_service quotd_udp
start_service chargen_tcp
start_service chargen_udp
start_service dummy_tcp
start_service dummy_udp
```

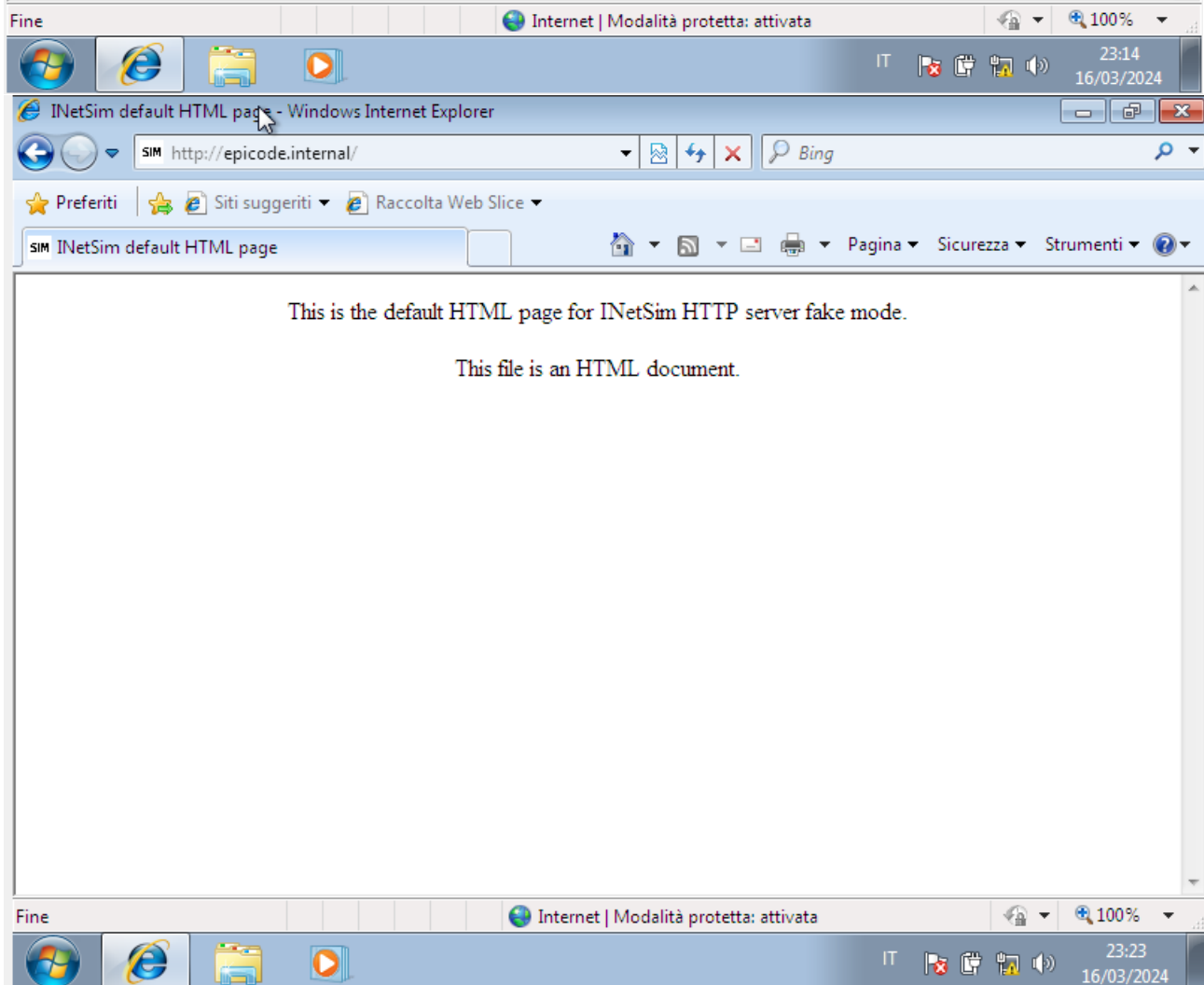
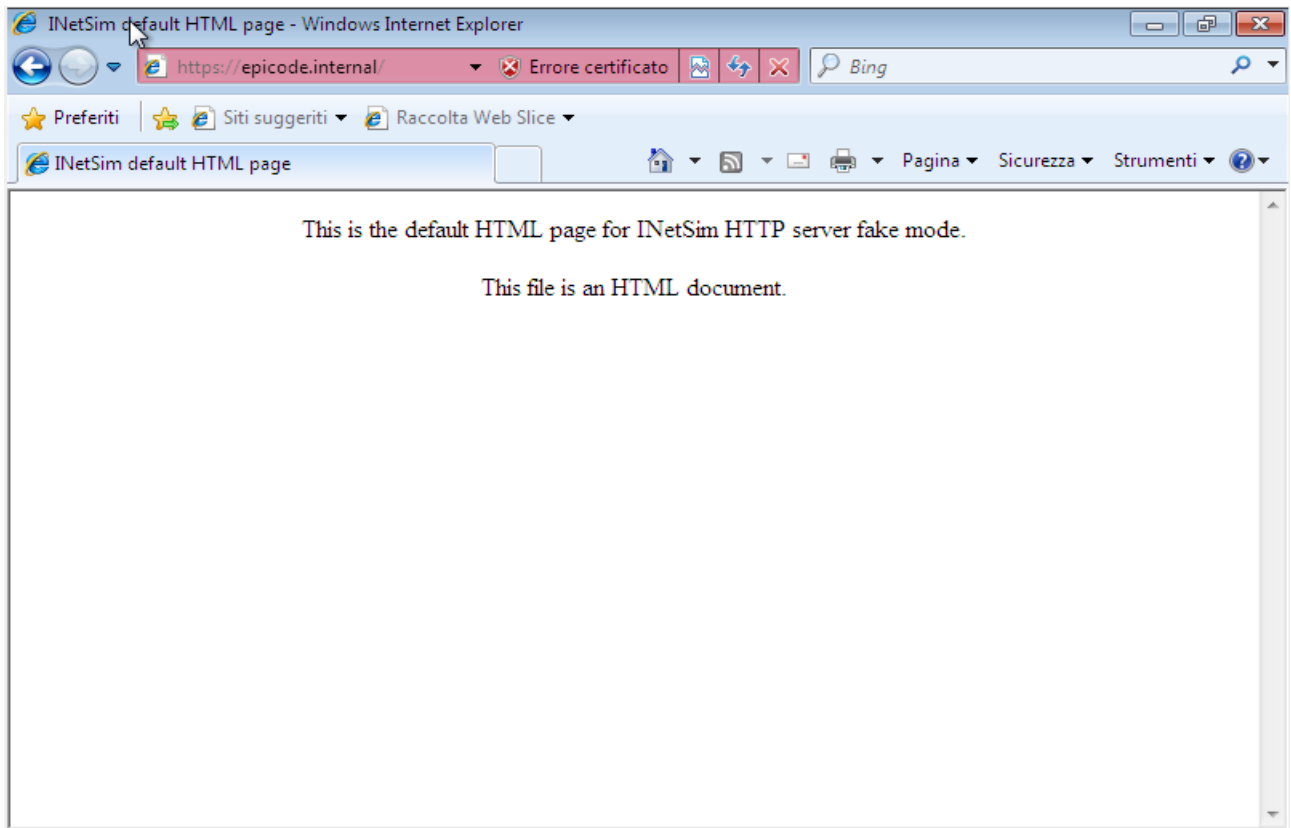
In questa parte abbiamo un elenco di tutti i vari protocolli che possono essere eseguiti da inetsim. In teoria per questo esercizio servirebbero solo i primi 3 (**evidenziati**): DNS, HTTP e HTTPS. Gli altri si potrebbero disattivare con un # a inizio riga ma io avendo trovato già questo setting a causa dell’esercizio delle settimane precedenti ho lasciato così, anche perché credo che per questo specifico esercizio non faccia differenza.

In seguito imposto DNS statico e IP associato, rispettivamente *epicode.internal* e 192.168.32.100 (Kali IP)

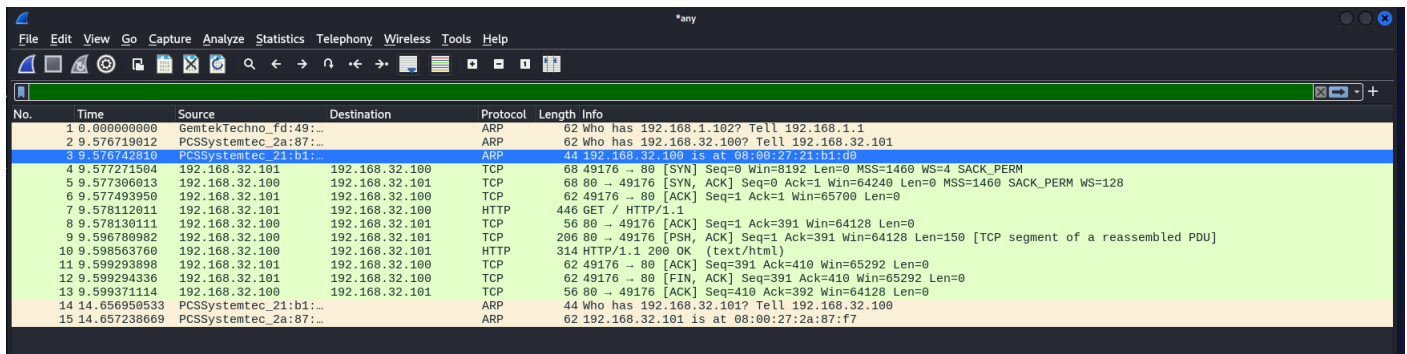


```
#####
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
#dns_static www.foo.com 10.10.10.10
dns_static epicode.internal 192.168.32.100
#dns_static ftp.bar.net 10.10.20.30
```

Dunque ora con il comando *sudo inetsim* su terminale Kali attivo la simulazione di connessione e provo, su Windows 7, a caricare la pagina *epicode.internal* sia con HTTP che con HTTPS.



Allo stesso tempo attivo Wireshark per intercettare il traffico tra Kali e Windows. Il primo screen mostra il traffico intercettato sul protocollo http. Si possono vedere nella riga 3 e nella riga 14, la coppia IP-MAC prima di Kali e poi di Windows. Durante questa trasmissione dati si utilizzano due protocolli ARP e TCP.



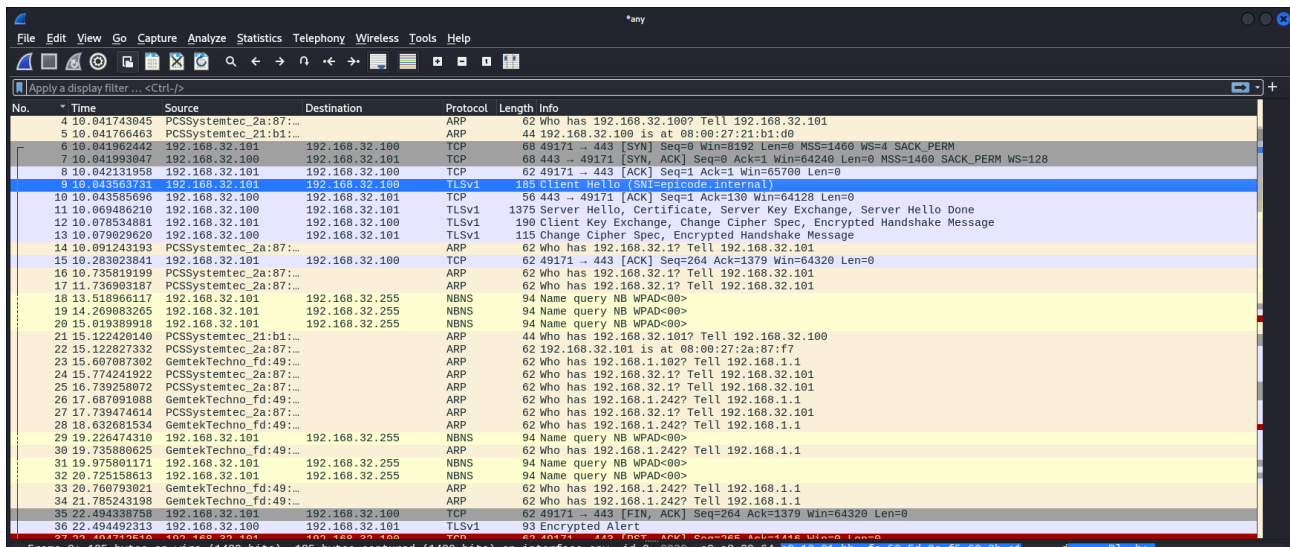
The image shows a Wireshark packet capture window. The packet list on the left shows 15 packets. The packet details pane on the right shows the selected packet (No. 3) as an ARP request from 192.168.32.101 to 192.168.32.100. The packet bytes pane on the right shows the raw data of the ARP request.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	GemtekTechno_fd:49:...		ARP	62	Who has 192.168.1.102? Tell 192.168.1.1
2	0.576719812	PCSSystemtec_2a:87:...		ARP	62	Who has 192.168.32.100? Tell 192.168.32.101
3	0.576721504	PCSSystemtec_21:b1:...	192.168.32.101	ARP	62	Who has 192.168.32.100? Tell 192.168.32.101
4	0.577215904	192.168.32.100	192.168.32.101	TCP	60	49176 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
5	0.577306013	192.168.32.100	192.168.32.101	TCP	60	80 → 49176 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
6	0.577493950	192.168.32.101	192.168.32.100	TCP	62	49176 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
7	0.578112011	192.168.32.101	192.168.32.100	HTTP	446	GET / HTTP/1.1
8	0.578130111	192.168.32.100	192.168.32.101	TCP	56	80 → 49176 [ACK] Seq=1 Ack=391 Win=64128 Len=0
9	0.596780982	192.168.32.100	192.168.32.101	TCP	206	80 → 49176 [PSH, ACK] Seq=1 Ack=391 Win=64128 Len=150 [TCP segment of a reassembled PDU]
10	0.598563760	192.168.32.100	192.168.32.101	HTTP	314	HTTP/1.1 200 OK (text/html)
11	0.599293898	192.168.32.101	192.168.32.100	TCP	62	49176 → 80 [ACK] Seq=391 Ack=410 Win=65292 Len=0
12	0.599294336	192.168.32.101	192.168.32.100	TCP	62	49176 → 80 [FIN, ACK] Seq=391 Ack=410 Win=65292 Len=0
13	0.599371114	192.168.32.100	192.168.32.101	TCP	56	80 → 49176 [ACK] Seq=410 Ack=392 Win=64128 Len=0
14	14.656950533	PCSSystemtec_21:b1:...		ARP	44	Who has 192.168.32.101? Tell 192.168.32.100
15	14.657238669	PCSSystemtec_2a:87:...		ARP	62	192.168.32.101 is at 08:00:27:2a:87:f7

Il protocollo ARP: (Address Resolution Protocol) è un protocollo di rete utilizzato per mappare gli indirizzi IP (Internet Protocol) degli host su una rete locale agli indirizzi MAC (Media Access Control) corrispondenti utilizzati dai dispositivi di rete per l'instradamento dei pacchetti. Il suo obiettivo principale è quello di consentire ai dispositivi di comunicare tra loro all'interno della stessa rete locale. Il punto debole di questo protocollo è che, in seguito alla fase di richiesta/risposta di due dispositivi che vogliono comunicare, salva temporaneamente nella sua cache (specie quando ci sono più richieste dagli stessi dispositivi in modo da velocizzare il processo di comunicazione) la coppia IP-MAC dei dispositivi dando dunque a un man in the middle la possibilità inserirsi nella connessione http (che non è criptata) e entrare in possesso di questi dati sensibili.

Il protocollo TCP: (Transmission Control Protocol) è uno dei principali protocolli utilizzati in reti sia domestiche che aziendali. La principale caratteristica di questo protocollo è l'affidabilità del flusso di dati, è infatti garantita la loro trasmissione, molto utile in particolarmente in ambito aziendale dove la perdita di informazioni potrebbe complicare il lavoro di dipendenti e azienda.

Questo secondo screen mostra il traffico HTTPS intercettato da Wireshark. La differenza rispetto al protocollo HTTP visto prima è la presenza del protocollo TLS.



No.	Time	Source	Destination	Protocol	Length	Info
4	10.041743045	PCSSystemtec_2a:87...	192.168.32.101	ARP	62	Who has 192.168.32.100? Tell 192.168.32.101
5	10.041766463	PCSSystemtec_21:b1...	192.168.32.100	ARP	44	192.168.32.100 is at 08:00:27:21:b1:d0
6	10.041962442	192.168.32.101	192.168.32.100	TCP	68	49171 → 443 [SYN] Seq=0 Win=6192 Len=0 MSS=1460 WS=4 SACK_PERM
7	10.041993047	192.168.32.100	192.168.32.101	TCP	68	443 → 49171 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
8	10.042131958	192.168.32.101	192.168.32.100	TCP	62	49171 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
9	10.042553721	192.168.32.101	192.168.32.100	TLSv1	1375	Client Hello (len=1375) (len=1375) (len=1375)
10	10.043585696	192.168.32.100	192.168.32.101	TCP	56	443 → 49171 [ACK] Seq=1 Ack=139 Win=64128 Len=0
11	10.069486210	192.168.32.100	192.168.32.101	TLSv1	1375	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	10.078534881	192.168.32.101	192.168.32.100	TLSv1	190	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	10.079929620	192.168.32.100	192.168.32.101	TLSv1	115	Change Cipher Spec, Encrypted Handshake Message
14	10.091243193	PCSSystemtec_2a:87...	192.168.32.100	ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
15	10.283023841	192.168.32.101	192.168.32.100	TCP	62	49171 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
16	10.735819199	PCSSystemtec_2a:87...	192.168.32.100	ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
17	11.736903187	PCSSystemtec_2a:87...	192.168.32.100	ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
18	13.518966117	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
19	14.269083205	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
20	15.019389918	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
21	15.122420140	PCSSystemtec_21:b1...	192.168.32.101	ARP	44	Who has 192.168.32.101? Tell 192.168.32.100
22	15.122827332	PCSSystemtec_2a:87...	192.168.32.101	ARP	62	192.168.32.101 is at 08:00:27:2a:87:f7
23	15.697087302	GemtekTechno_Fd:49...	192.168.32.101	ARP	62	Who has 192.168.1.102? Tell 192.168.1.1
24	15.774241922	PCSSystemtec_2a:87...	192.168.32.101	ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
25	16.739258072	PCSSystemtec_2a:87...	192.168.32.101	ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
26	17.687091088	GemtekTechno_Fd:49...	192.168.1.1	ARP	62	Who has 192.168.1.242? Tell 192.168.1.1
27	17.739474614	PCSSystemtec_2a:87...	192.168.32.101	ARP	62	Who has 192.168.32.1? Tell 192.168.32.101
28	18.632681534	GemtekTechno_Fd:49...	192.168.1.1	ARP	62	Who has 192.168.1.242? Tell 192.168.1.1
29	19.226474310	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
30	19.735808625	GemtekTechno_Fd:49...	192.168.1.1	ARP	62	Who has 192.168.1.242? Tell 192.168.1.1
31	19.975801171	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
32	20.725158613	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD<00>
33	20.760793021	GemtekTechno_Fd:49...	192.168.1.1	ARP	62	Who has 192.168.1.242? Tell 192.168.1.1
34	21.785243198	GemtekTechno_Fd:49...	192.168.1.1	ARP	62	Who has 192.168.1.242? Tell 192.168.1.1
35	22.494338758	192.168.32.101	192.168.32.100	TCP	62	49171 → 443 [FIN, ACK] Seq=264 Ack=1379 Win=64320 Len=0
36	22.494492313	192.168.32.100	192.168.32.101	TLSv1	93	Encrypted Alert

Il protocollo TLS: (Transport Layer Security) è un protocollo di crittografia che ha per obbiettivo garantire la sicurezza delle comunicazioni che vengono cifrate e non trasmesse in chiaro. Nella riga 9 si vede come il processo di connessione inizi con un handshake durante il quale il client invia il messaggio “Client Hello” che contiene le tipologie di crittografia supportate dal protocollo. In questo momento client e server stabiliscono algoritmo di crittografia e chiavi per la comunicazione. Questo protocollo evita la possibilità che una persona terza si inserisca tra noi e il server e rubi informazioni sensibili come la coppia IP-MAC. La cifrazione di IP e MAC è utilizzata per esempio dalle VPN o dalle app di messaggistica istantanea che impediscono l’intercettazione dei messaggio e delle richieste nel momento dell’invio.

In conclusione, il protocollo HTTPS cripta l’intero canale di comunicazione rendendo praticamente impossibile, anche a causa della velocità di esecuzione, la decrittazione e l’inserimento di un man in the middle nella comunicazione rendendo la trasmissione della richiesta molto più sicura.