

# W4D4 - Progetto

Simulazione protocolli DNS, HTTP e HTTPS su INetSim

# Impostazione indirizzo IP VM Kali Linux

Editing 192.168.32.100

Connection name 192.168.32.100

General Ethernet 802.1X Security DCB Proxy **IPv4 Settings** IPv6 Settings

Method Manual

**Addresses**

Address	Netmask	Gateway	
192.168.32.100	24	192.168.32.1	<div>Add</div> <div>Delete</div>

DNS servers

Search domains

DHCP client ID

☒ Require IPv4 addressing for this connection to complete

Routes...

Cancel

✓ Save

L'indirizzo IP della macchina Kali è stato impostato tramite il NetworkManager.

Indirizzo IPv4: 192.168.32.100/24

La macchina virtuale Kali Linux, grazie alla simulazione dei servizi con il tool INetSim, fungerà da server che risponderà alla richiesta di un client di accedere alla risorsa 'epicode.internal'.

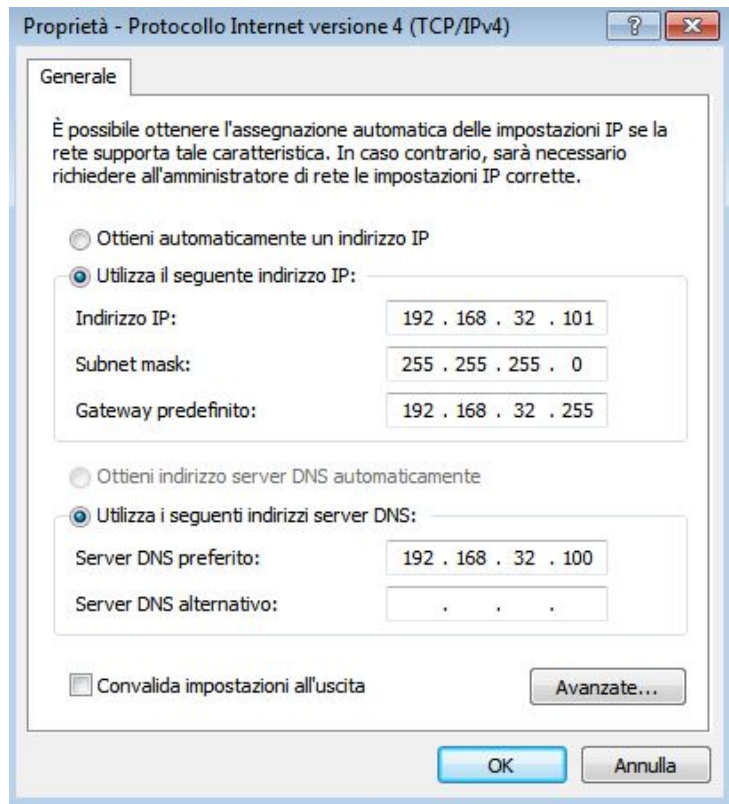
# Impostazione indirizzo IP VM Kali Linux

```
(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::9f55:75c3:951a:6b53 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:43:73:bc txqueuelen 1000 (Ethernet)  
    RX packets 383 bytes 32233 (31.4 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 401 bytes 38716 (37.8 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 1504 (1.4 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 24 bytes 1504 (1.4 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Verifica corretta configurazione di rete con comando 'ifconfig' su macchina Kali Linux. Questo comando permette di visualizzare sia lo stato della rete che la sua configurazione, evidenziando indirizzo IP, subnet mask, indirizzo di broadcast e MAC address della macchina virtuale.

Il MAC address di Kali Linux, come da screenshot al lato è: **08:00:27:43:73:bc**

# Impostazione indirizzo IP VM Windows 7



```
C:\Users\utente>ipconfig

Configurazione IP di Windows

Scheda Ethernet Connessione alla rete locale (LAN):

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::d971:5b4b:8bc6:c8a3%11
    Indirizzo IPv4. . . . . : 192.168.32.101
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . : 192.168.32.255

Scheda Tunnel isatap.{3471EE44-5A46-42A3-BB36-A869097FC1E8}:

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

Indirizzo IPv4: 192.168.32.101/24

Server DNS: Indirizzo IPv4 Kali Linux  
192.168.32.100

# Impostazione indirizzo IP VM Windows 7

```
C:\Users\utente>ipconfig /all

Configurazione IP di Windows

Nome host . . . . . : utente-PC
Suffisso DNS primario . . . . . : 
Tipo nodo . . . . . : Ibrido
Routing IP abilitato . . . . . : No
Proxy WINS abilitato . . . . . : No

Scheda Ethernet Connessione alla rete locale (LAN):

Suffisso DNS specifico per connessione:
Descrizione . . . . . : Scheda desktop Intel(R) PRO/1000 MT
Indirizzo fisico . . . . . : 08-00-27-69-C4-C9
DHCP abilitato . . . . . : No
Configurazione automatica abilitata : Si
Indirizzo IPv6 locale rispetto al collegamento . : fe80::d971:5b4b:8bc6:c8a3%11(Preferenziale)
Indirizzo IPv4 . . . . . : 192.168.32.101(Preferenziale)
Subnet mask . . . . . : 255.255.255.0
Gateway predefinito . . . . . : 192.168.32.255
ID DHCPv6 . . . . . : 235405351
DUID Client DHCPv6 . . . . . : 00-01-00-01-2D-ED-E8-F6-08-00-27-69-C4-C9

Server DNS . . . . . : 192.168.32.100
NetBIOS su TCP/IP . . . . . : Attivato

Scheda Tunnel isatap.{3471EE44-5A46-42A3-BB36-A869097FC1E8}:

Stato supporto . . . . . : Supporto disconnesso
Suffisso DNS specifico per connessione:
Descrizione . . . . . : Microsoft ISA/ATP Adapter
Indirizzo fisico . . . . . : 00-00-00-00-00-00-E0
DHCP abilitato . . . . . : No
Configurazione automatica abilitata : Si
```

Per verificare la corretta configurazione di rete della macchina virtuale (client) Windows 7, è utile il comando 'ipconfig /all', che mostrerà sia l'indirizzo fisico MAC associato alla scheda di rete virtuale della macchina Windows, che l'indirizzo IPv4, la subnet mask e il DNS.

Nel caso della macchina Windows 7, il MAC address corrisponde a: **08-00-27-69-C4-C9**.

Questo tornerà utile nel successivo tracciamento dei pacchetti e comunicazioni tra client e server con Wireshark.

# Verifica comunicazione tra le due macchine

```
Microsoft Windows [Versione 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.  
  
C:\Users\utente>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
```

Ping da Client Windows 7 - Indirizzo IPv4:  
192.168.32.101/24 al Server Kali Linux - Indirizzo  
IPv4: 192.168.32.100/24

```
(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=0.527 ms  
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=0.985 ms  
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=1.13 ms  
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=1.05 ms  
64 bytes from 192.168.32.101: icmp_seq=5 ttl=128 time=0.557 ms  
^C  
--- 192.168.32.101 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4016ms  
rtt min/avg/max/mdev = 0.527/0.851/1.134/0.257 ms
```

Ping da Server Kali Linux - Indirizzo IPv4:  
192.168.32.100 a Client Windows 7 - Indirizzo  
IPv4: 192.168.32.101/24

# Configurazione tool INetSim - attivazione servizi

```
GNU nano 5.4
#####
#
# INetSim configuration file
#
#####
# 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
#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
```

Il file di configurazione del tool INetSim è salvato nella directory `/etc/inetsim/inetsim.conf`. Da terminale, si possono apportare modifiche al file di configurazione con privilegi amministratore accedendo al file tramite editor di testo nano.

Nello screenshot al lato si può osservare che i servizi utili allo svolgimento della traccia sono attivi.



# Configurazione tool INetSim - ascolto su porte Kali

```
#####  
# service_bind_address  
#  
# IP address to bind services to  
#  
# Syntax: service_bind_address <IP address>  
#  
# Default: 127.0.0.1  
#  
#service_bind_address 10.10.10.1  
service_bind_address 192.168.32.100
```

INetSim è stato impostato per ascoltare sulle porte della macchina Kali.

In alternativa, si può lanciare il tool INetSim da terminale con il comando `--bind-address=<IP address>`, impostando sempre il tool per ascoltare sull'indirizzo IPv4 di Kali.

```
#####  
# service_bind_address  
#  
# IP address to bind services to  
#  
# Syntax: service_bind_address <IP address>  
#  
# Default: 127.0.0.1  
#  
#service_bind_address 10.10.10.1  
service_bind_address 0.0.0.0
```

Se impostato su indirizzo IPv4 0.0.0.0, il tool ascolterà su tutti gli indirizzi IPv4 dei client che possono stabilire una connessione con il server.





# Configurazione tool INetSim - protocollo DNS

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

Il protocollo DNS risolverà la richiesta di una risorsa all'hostname 'epicode.internal' da parte del client in un indirizzo IPv4 - in questo caso quello della macchina virtuale Kali (server).

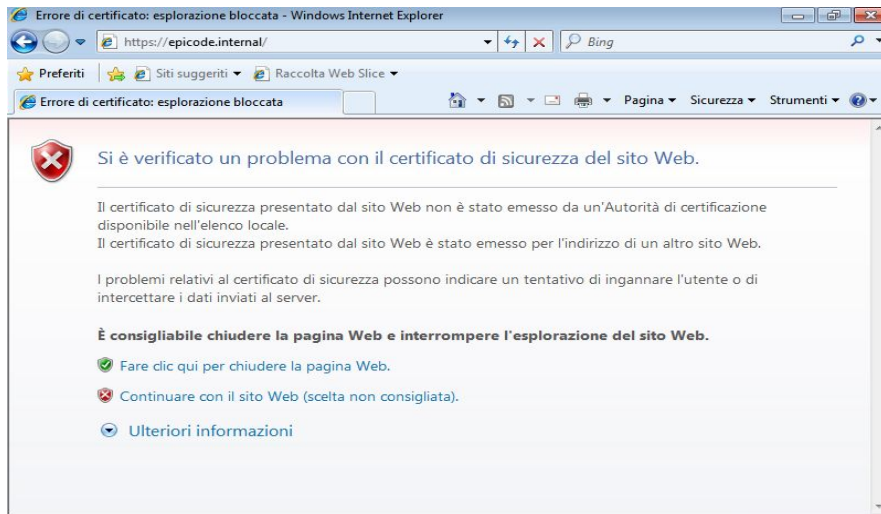


# Lancio tool INetSim

```
(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 1563) ≡  
Session ID:    1563  
Listening on:  0.0.0.0  
Real Date/Time: 2024-06-03 05:00:38  
Fake Date/Time: 2024-06-03 05:00:38 (Delta: 0 seconds)  
Forking services...  
  * dns_53_tcp_udp - started (PID 1567)  
  * https_443_tcp - started (PID 1569)  
  * http_80_tcp - started (PID 1568)  
done.  
Simulation running.  
█
```

Per simulare i servizi di rete è innanzitutto necessario lanciare il tool INetSim tramite comando 'sudo inetsim'.

# Richiesta risorsa web da client - HTTPS

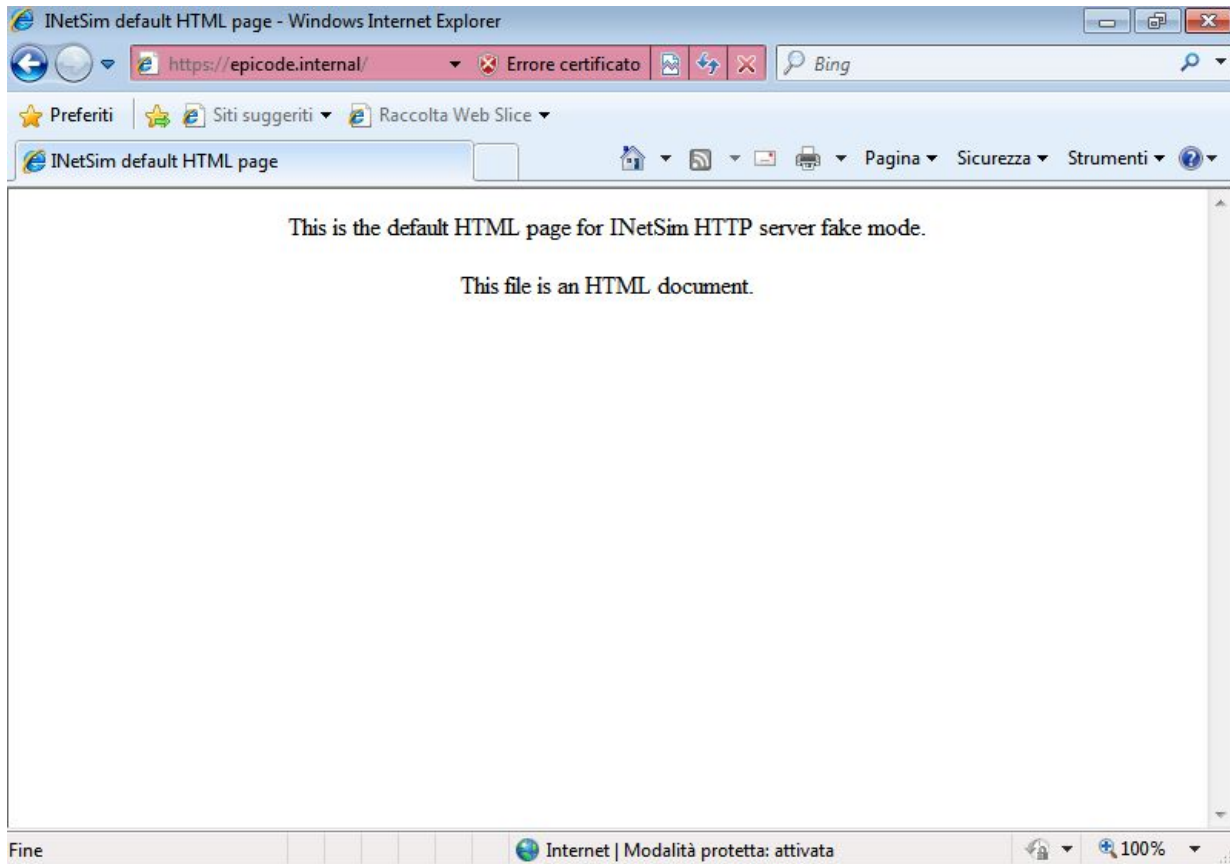


Una volta lanciato INetSim, si può procedere a richiedere una risorsa web da client Windows 7 tramite servizio HTTPS. Da web browser, digitando '<https://epicode.internal/>', Windows Explorer ci avvisa di un problema con il certificato sicurezza del sito web richiesto, che non risulta emesso da un'Autorità di certificazione riconosciuta per quella particolare risorsa web.

Facendo click sulla voce 'Sicurezza' > 'Rapporto sulla sicurezza', si potranno visualizzare i dettagli del certificato.

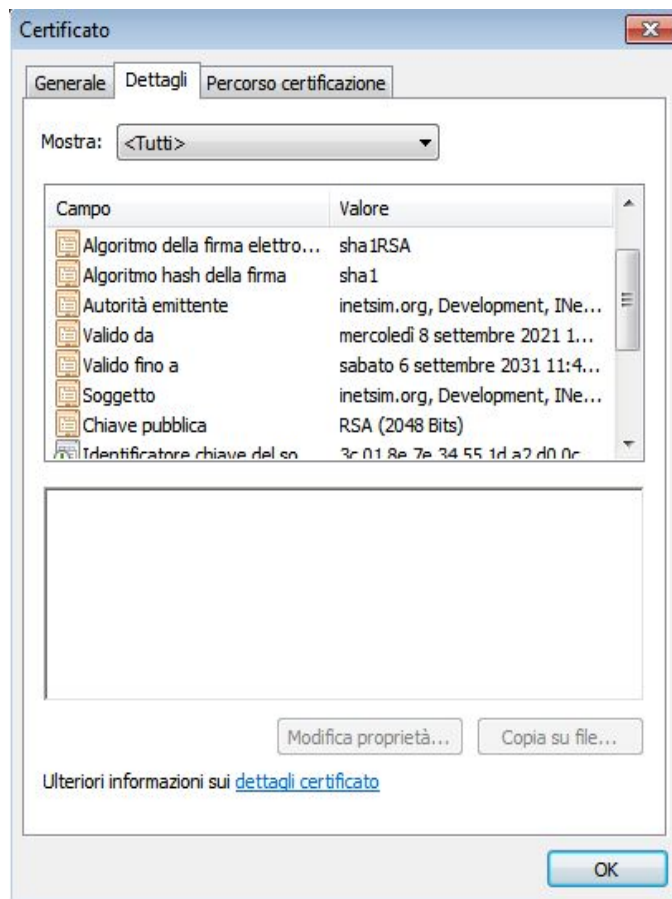
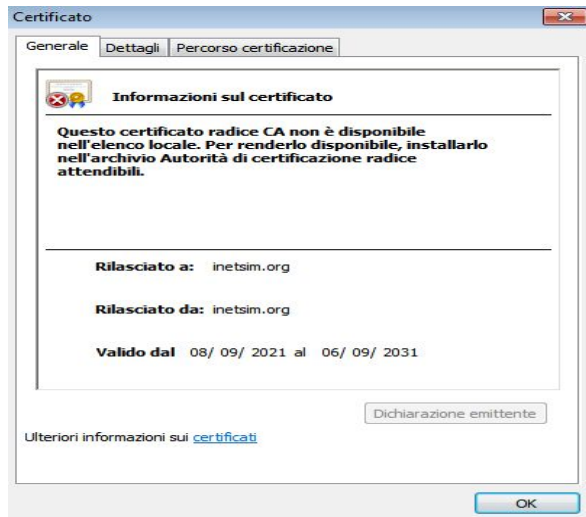
Facendo clic su 'Continuare con il sito Web' e ignorando l'avviso di sicurezza di Windows, si potrà invece accedere alla risorsa.

# Intercettazione comunicazione con Wireshark - HTTPS



Una volta cliccato su 'Continuare con il sito Web', è possibile ignorare l'alert e accedere alla risorsa.

# Richiesta risorsa web da client - HTTPS



I dettagli del certificato per '<https://epicode.internal/>' sono come in figura al lato. Il certificato è stato rilasciato a inetsim.org da inetsim.org.

Alla voce 'Dettagli' è possibile visualizzare la versione del certificato, il numero di serie, nonché l'algoritmo della firma elettronica e l'algoritmo hash della firma, oltre alla chiave pubblica del certificato digitale.

Dunque l'alert di Windows 7 è dovuto probabilmente al fatto che il tool INetSim include certificati SSL generici che vengono utilizzati per simulare i servizi e il certificato per la particolare risorsa web accessibile sull'hostname epicode.internal non è stato generato per quel dominio e non è stato generato da un'Autorità riconosciuta per il rilascio di certificati digitali (come ad esempio Amazon.com).

Ciò significa che non è possibile verificare l'autenticità del sito web epicode.internal.

# Intercettazione comunicazione con WireShark - HTTPS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
2	2.998155338	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
3	6.999712441	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
4	9.999331014	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
5	12.998488729	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
6	14.726429196	PcsCompu_69:c4:c9	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
7	14.726483954	PcsCompu_43:73:bc	PcsCompu_69:c4:c9	ARP	42	192.168.32.100 is at 08:00:27:69:c4:c9
8	14.726758319	192.168.32.101	192.168.32.100	TCP	60	49194 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=4 SACK_PERM=1
9	14.726771622	192.168.32.100	192.168.32.101	TCP	66	443 → 49194 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
10	14.727075371	192.168.32.101	192.168.32.100	TCP	60	49194 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
11	14.727388972	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
12	14.727399705	192.168.32.100	192.168.32.101	TCP	54	443 → 49194 [ACK] Seq=1 Ack=162 Win=64128 Len=0
13	14.737854572	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
14	14.747575962	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
15	14.747646911	192.168.32.100	192.168.32.101	TCP	54	443 → 49194 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
16	14.748018958	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
17	14.765543095	fe80::d971:5b4b:8bc...	ff02::1:3	LLMNR	84	Standard query 0xaa32 A wpad
18	14.765543145	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xaa32 A wpad
19	14.872784009	fe80::d971:5b4b:8bc...	ff02::1:3	LLMNR	84	Standard query 0xaa32 A wpad
20	14.872784147	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xaa32 A wpad
21	14.950937434	192.168.32.101	192.168.32.100	TCP	60	49194 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len=0
22	15.077401123	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
23	15.825883248	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
Protocol type: IPv4 (0x0800)						
Hardware size: 6						
Protocol size: 4						
Opcode: request (1)						
Sender MAC address: PcsCompu_69:c4:c9 (08:00:27:69:c4:c9)						
Sender IP address: 192.168.32.101						
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)						
Target IP address: 192.168.32.100						

Il client e il server sono sulla stessa rete. Il client deve richiedere una risorsa web al server 192.168.32.100, ma ne conosce solo l'indirizzo IP e non il MAC address. Dunque, per associare l'indirizzo IP al MAC invia una richiesta ARP (Address Resolution Protocol) a tutti i dispositivi sullo stesso dominio di broadcast, chiedendo: 'Chi ha l'indirizzo IP 192.168.32.100?'. Come si può vedere dal dettaglio mostrato in basso a sinistra, il MAC address del client corrisponde a quello della macchina Windows 7.

# Intercettazione comunicazione con WireShark - HTTPS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
2	2.998155338	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
3	6.999712441	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
4	9.999331014	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
5	12.998488729	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
6	14.726429196	PcsCompu_69:c4:c9	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
7	14.726483954	PcsCompu_43:73:bc	PcsCompu_69:c4:c9	ARP	42	192.168.32.100 is at 08:00:27:43:73:bc
8	14.726759310	192.168.32.101	192.168.32.100	TCP	66	49194 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
9	14.726771622	192.168.32.100	192.168.32.101	TCP	66	443 → 49194 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
10	14.727075371	192.168.32.101	192.168.32.100	TCP	60	49194 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
11	14.727388972	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
12	14.727399705	192.168.32.100	192.168.32.101	TCP	54	443 → 49194 [ACK] Seq=1 Ack=162 Win=64128 Len=0
13	14.737854572	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
14	14.747575962	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
15	14.747646911	192.168.32.100	192.168.32.101	TCP	54	443 → 49194 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
16	14.748018958	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
17	14.765543095	fe80::d971:5b4b:8bc...	ff02::1:3	LLMNR	84	Standard query 0xaa32 A wpad
18	14.765543145	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xaa32 A wpad
19	14.872784009	fe80::d971:5b4b:8bc...	ff02::1:3	LLMNR	84	Standard query 0xaa32 A wpad
20	14.872784147	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xaa32 A wpad
21	14.950937434	192.168.32.101	192.168.32.100	TCP	60	49194 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len=0
22	15.077401123	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
23	15.825883248	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reply (2)

Sender MAC address: PcsCompu\_43:73:bc (08:00:27:43:73:bc)

Sender IP address: 192.168.32.100

Target MAC address: PcsCompu\_69:c4:c9 (08:00:27:69:c4:c9)

Target IP address: 192.168.32.101

Il server Kali, che ha l'indirizzo IP incluso nella richiesta ARP, risponderà inviando il proprio MAC address al MAC address della macchina Windows 7.



# Intercettazione comunicazione con WireShark - HTTPS

No.	Time	Source	Destination	Protocol	Length	Info
13	6.717001695	PcsCompu. 69:c4:c9		ARP	62	192.168.32.101 is at 08:00:27:69:c4:c9
14	9.080604395	192.168.32.101	192.168.32.100	DNS	78	Standard query 0x953d A epicode.internal
15	9.089926074	192.168.32.100	192.168.32.101	DNS	94	Standard query response 0x953d A epicode.internal A 192.168.32.100
16	9.090890642	192.168.32.101	192.168.32.100	TCP	68	49200 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
17	9.090927994	192.168.32.100	192.168.32.101	TCP	68	443 → 49200 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
18	9.091305496	192.168.32.101	192.168.32.100	TCP	62	49200 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
19	9.094288205	192.168.32.101	192.168.32.100	TLSv1	185	Client Hello
20	9.094308223	192.168.32.100	192.168.32.101	TCP	56	443 → 49200 [ACK] Seq=1 Ack=130 Win=64128 Len=0
21	9.098121282	192.168.32.100	192.168.32.101	TLSv1	1375	Server Hello, Certificate, Server Key Exchange, Server Hello Done
22	9.106693864	192.168.32.101	192.168.32.100	TLSv1	190	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
23	9.106730557	192.168.32.100	192.168.32.101	TCP	56	443 → 49200 [ACK] Seq=1320 Ack=264 Win=64128 Len=0
24	9.107482406	192.168.32.100	192.168.32.101	TLSv1	115	Change Cipher Spec, Encrypted Handshake Message
25	9.138102903	fe80::d971:5b4b:8bc...	ff02::1:3	LLMNR	86	Standard query 0xa19f A wpad

```
Frame 16: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 49200, Dst Port: 443, Seq: 0, Len: 0
  Source Port: 49200
  Destination Port: 443
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 1884041388
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1000 .... = Header Length: 32 bytes (8)
```

WireShark permette di avere una visuale dettagliata del 'Three-way handshake' tipico del protocollo TCP, che inizia a scambiare pacchetti di dati solo dopo aver stabilito una connessione tra hosts o client/server. Come da screenshot, la comunicazione viene iniziata dal client, che invia al server un pacchetto con la flag 'SYN' attiva e un numero di sequenza casuale. Il server risponde, inviando al client un pacchetto con i flag SYN e ACK abilitati, e un altro numero di sequenza casuale. L'ACK sarà uguale al precedente numero Seq +1. Il client completa la sincronizzazione inviando un pacchetto ACK e inviando i numeri Seq, e ACK come fatto dal server. La richiesta HTTPS avviene sulla porta 443, che è quella riservata a questo protocollo.

Una volta finito il processo di stretta di mano in tre fasi, inizia la comunicazione con il TLSv1 (Transport Layer Security version 1), protocollo di trasporto che permette una comunicazione sicura end-to-end e fornisce autenticazione, integrità dei dati e confidenzialità.

In questo frangente avviene lo scambio di chiavi tra client e server con cifratura asimmetrica. Nel protocollo HTTPS, un client può accedere alla chiave pubblica tramite il certificato digitale e possiede una chiave privata univoca. I messaggi nel protocollo HTTPS sono criptati, come è possibile verificare alla riga 22 e 24.

# Intercettazione comunicazione con WireShark - HTTPS

44	15.269838915	192.168.32.101	192.168.32.100	TCP	62 49200 → 443 [FIN, ACK] Seq=264 Ack=1379 Win=64320 Len=0
45	15.274205390	192.168.32.100	192.168.32.101	TLSv1	93 Encrypted Alert
46	15.274472743	192.168.32.101	192.168.32.100	TCP	62 49200 → 443 [RST, ACK] Seq=265 Ack=1416 Win=0 Len=0
47	16.406865433	192.168.32.101	239.255.255.250	SSDP	546 NOTIFY * HTTP/1.1
48	16.406865574	fe80::d971:5b4b:8bc...	ff02::c	SSDP	573 NOTIFY * HTTP/1.1
49	16.407149030	192.168.32.101	239.255.255.250	SSDP	560 NOTIFY * HTTP/1.1
50	16.407305919	fe80::d971:5b4b:8bc...	ff02::c	SSDP	587 NOTIFY * HTTP/1.1
51	16.407638871	192.168.32.101	239.255.255.250	SSDP	480 NOTIFY * HTTP/1.1
52	16.407639005	fe80::d971:5b4b:8bc...	ff02::c	SSDP	507 NOTIFY * HTTP/1.1

```
[Calculated window size: 64128]
[Window size scaling factor: 128]
Checksum: 0xc259 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [SEQ/ACK analysis]
> [Timestamps]
TCP payload (37 bytes)
Transport Layer Security
- TLSv1 Record Layer: Encrypted Alert
  Content Type: Alert (21)
  Version: TLS 1.0 (0x0301)
  Length: 32
  Alert Message: Encrypted Alert
```

Come è possibile vedere alla riga 45, WireShark tiene traccia anche dell'alert ricevuto riguardo il certificato di 'epicode.internal'. L'alert 21 è un alert del protocollo TLS e, come si può vedere, anche questo è criptato.

La flag FIN alla riga 44 indica la fine della connessione tra client e server.

La flag RST denota invece un 'abortion' della connessione, solitamente viene inviata quando uno dei due peer (hosts o client/server) ha ragione di credere che la connessione non dovrebbe esistere.

Ad esempio, in caso di attacchi, errori, o eventi sospetti.

In questo caso, può essere dovuto all'errore nel certificato di 'epicode.internal'.

# Intercettazione comunicazione con WireShark - HTTP

No.	Time	Source	Destination	Protocol	Length	Info
7	5.999112560	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
8	6.899884135	PcsCompu_43:73:bc	PcsCompu_69:c4:c9	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
9	6.900292317	PcsCompu_69:c4:c9	PcsCompu_43:73:bc	ARP	60	192.168.32.101 is at 08:00:27:69:c4:c9
10	6.916283337	192.168.32.101	192.168.32.100	TCP	66	49207 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
11	6.916348350	192.168.32.100	192.168.32.101	TCP	66	80 → 49207 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
12	6.916670647	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	6.917348346	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
14	6.917363272	192.168.32.100	192.168.32.101	TCP	54	80 → 49207 [ACK] Seq=1 Ack=419 Win=64128 Len=0
15	6.937349288	192.168.32.100	192.168.32.101	TCP	204	80 → 49207 [PSH, ACK] Seq=1 Ack=419 Win=64128 Len=150 [TCP segment of a reassembled PDU]
16	6.941250464	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
17	6.941629075	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [ACK] Seq=419 Ack=410 Win=65292 Len=0
18	6.941770787	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [FIN, ACK] Seq=419 Ack=410 Win=65292 Len=0
19	6.941782896	192.168.32.100	192.168.32.101	TCP	54	80 → 49207 [ACK] Seq=410 Ack=420 Win=64128 Len=0

▶ Frame 8: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0

▶ Ethernet II, Src: PcsCompu\_43:73:bc (08:00:27:43:73:bc), Dst: PcsCompu\_69:c4:c9 (08:00:27:69:c4:c9)

▼ Address Resolution Protocol (request)

- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: request (1)
- Sender MAC address: PcsCompu\_43:73:bc (08:00:27:43:73:bc)
- Sender IP address: 192.168.32.100
- Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
- Target IP address: 192.168.32.101

Effettuando una richiesta http per 'epicode.internal' da web browser di Windows 7, si può osservare quali siano le differenze tra HTTP e HTTPS e come avviene la comunicazione.

Innanzitutto, anche nel caso del servizio HTTP viene lanciata una richiesta ARP dal client per conoscere il MAC address dell'indirizzo IP del server.

# Intercettazione comunicazione con WireShark - HTTP

No.	Time	Source	Destination	Protocol	Length	Info
7	5.999112560	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
8	6.899884135	PcsCompu_43:73:bc	PcsCompu_69:c4:c9	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
9	6.900292317	PcsCompu_69:c4:c9	PcsCompu_43:73:bc	ARP	60	192.168.32.101 is at 08:00:27:69:c4:c9
10	6.916283337	192.168.32.101	192.168.32.100	TCP	66	49207 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
11	6.916348350	192.168.32.100	192.168.32.101	TCP	66	80 → 49207 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
12	6.916670647	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	6.917348346	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
14	6.917363272	192.168.32.100	192.168.32.101	TCP	54	80 → 49207 [ACK] Seq=1 Ack=419 Win=64128 Len=0
15	6.937349288	192.168.32.100	192.168.32.101	TCP	204	80 → 49207 [PSH, ACK] Seq=1 Ack=419 Win=64128 Len=150 [TCP segment of a reassembled PDU]
16	6.941250464	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
17	6.941629075	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [ACK] Seq=419 Ack=410 Win=65292 Len=0
18	6.941770787	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [FIN, ACK] Seq=419 Ack=410 Win=65292 Len=0
19	6.941782896	192.168.32.100	192.168.32.101	TCP	54	80 → 49207 [ACK] Seq=410 Ack=420 Win=64128 Len=0
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100						
Transmission Control Protocol, Src Port: 49207, Dst Port: 80, Seq: 1, Ack: 1, Len: 418						
Hypertext Transfer Protocol						
GET / HTTP/1.1\r\n						
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap, */*\r\n						
Accept-Language: it-IT\r\n						
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)\r\n						
Accept-Encoding: gzip, deflate\r\n						
Host: epicode.internal\r\n						
Connection: Keep-Alive\r\n						
\r\n						
[Full request URI: http://epicode.internal/]						
[HTTP request 1/1]						
[Response in frame: 16]						

Come si può vedere alla riga 10, la richiesta HTTP avviene sulla porta 80, che è quella riservata a questo protocollo. Viene stabilita una connessione tra client e server con il three-way-handshake. Oltre alle 'Synchronization' e 'Acknowledgement' flag, alla riga 15 si osserva la flag 'PSH', che comunica al server di inviare l'intero contenuto immediatamente.

Alla riga 13, si può vedere una richiesta con metodo 'GET', che è quello utilizzato quando si richiede una risorsa web. Il campo host specifica la risorsa richiesta, in questo caso epicode.internal. Il campo user-agent dà indicazione al server del programma lato client dal quale sta partendo la richiesta.

Il campo Accept indica che tipo di documento ci si aspetta come risposta. Il parametro 'Keep-Alive' alla voce Connection farà in modo che la connessione potrà essere riutilizzata nelle successive interazioni client-server.

Come si può osservare, le richieste HTTP a differenza di quelle HTTPS non sono criptate.

# Intercettazione comunicazione con WireShark - HTTP

No.	Time	Source	Destination	Protocol	Length	Info
7	5.999112560	fe80::d971:5b4b:8bc...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
8	6.899884135	PcsCompu_43:73:bc	PcsCompu_69:c4:c9	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
9	6.900292317	PcsCompu_69:c4:c9	PcsCompu_43:73:bc	ARP	60	192.168.32.101 is at 08:00:27:69:c4:c9
10	6.916283337	192.168.32.101	192.168.32.100	TCP	66	49207 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
11	6.916348350	192.168.32.100	192.168.32.101	TCP	66	80 → 49207 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
12	6.916670647	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	6.917348346	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
14	6.917363272	192.168.32.100	192.168.32.101	TCP	54	80 → 49207 [ACK] Seq=1 Ack=419 Win=64128 Len=0
15	6.937349288	192.168.32.100	192.168.32.101	TCP	204	80 → 49207 [PSH, ACK] Seq=1 Ack=419 Win=64128 Len=150 [TCP segment of a reassembled PDU]
16	6.941250464	192.168.32.101	192.168.32.100	HTTP	312	HTTP/1.1 200 OK (text/html)
17	6.941629075	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [ACK] Seq=419 Ack=410 Win=65292 Len=0
18	6.941770787	192.168.32.101	192.168.32.100	TCP	60	49207 → 80 [FIN, ACK] Seq=419 Ack=410 Win=65292 Len=0
19	6.941782896	192.168.32.100	192.168.32.101	TCP	54	80 → 49207 [ACK] Seq=410 Ack=420 Win=64128 Len=0
Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101						
Transmission Control Protocol, Src Port: 80, Dst Port: 49207, Seq: 151, Ack: 419, Len: 258						
[2 Reassembled TCP Segments (408 bytes): #15(150), #16(258)]						
Hypertext Transfer Protocol						
HTTP/1.1 200 OK\r\n						
Server: INetSim HTTP Server\r\n						
Connection: Close\r\n						
Content-Type: text/html\r\n						
Date: Mon, 03 Jun 2024 10:40:34 GMT\r\n						
Content-Length: 258\r\n						
\r\n						
[HTTP response 1/1]						
[Time since request: 0.023902118 seconds]						
[Request in frame: 13]						

Quando un server riceve la chiamata, la elabora e poi spedisce la risposta al client.

Alla riga 16 si leggono i dettagli della risposta del server. La riga di stato '200 OK' indica che la risorsa è stata trovata e i dettagli della risposta e della risorsa trovata sono leggibili nel pannello in basso. Ad esempio, la data in cui è stato generato il messaggio, le informazioni sul server che ha generato la richiesta (in questo caso INetSim HTTP Server) e il formato del contenuto della risorsa, ossia html.

I caratteri \r (ritorno a capo) e \n (nuova linea) sono utilizzati per terminare le righe HTTP.



# Conclusioni

Le differenze principali tra il traffico catturato in HTTP e quello catturato in HTTPS sono le seguenti:

1. La comunicazione nelle richieste HTTPS è criptata. In HTTP, client e server si scambiano dati in chiaro.
2. Nelle risposte HTTP, il server espone anche informazioni sul tipo di versione utilizzata (disclosure, può essere sfruttata per attacchi).
3. Nelle richieste HTTP non c'è l'extra layer del certificato di sicurezza SSL/TLS che consente la verifica dell'autenticità di un sito web.

