

# Esercitazione Fine modulo 1 - W4D4

Fabio Benevento - 17/11/2023

---

## Scopo

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 episode.internal che risponde all'indirizzo 192.168.32.100 (Kali).

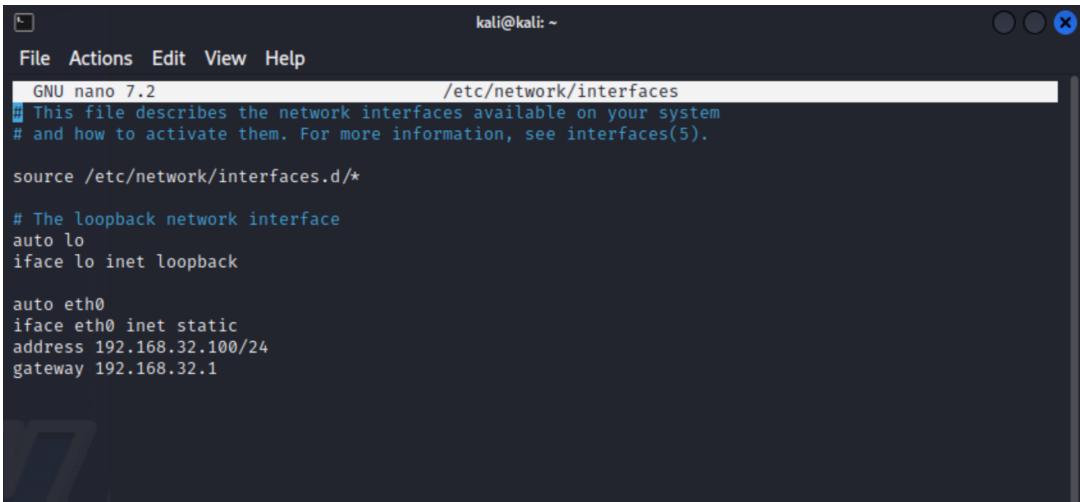
Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS. Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

## Implementazione

### Configurazione ambiente

#### Configurazione Kali Linux

Per la realizzazione dell'esercitazione, ho proceduto in primo luogo alla configurazione della macchina virtuale Kali Linux, assegnandole l'indirizzo ip statico 192.168.32.100 sull'interfaccia eth0 mediante modifica del file /etc/network/interfaces secondo quanto riportato in figura.



```
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).

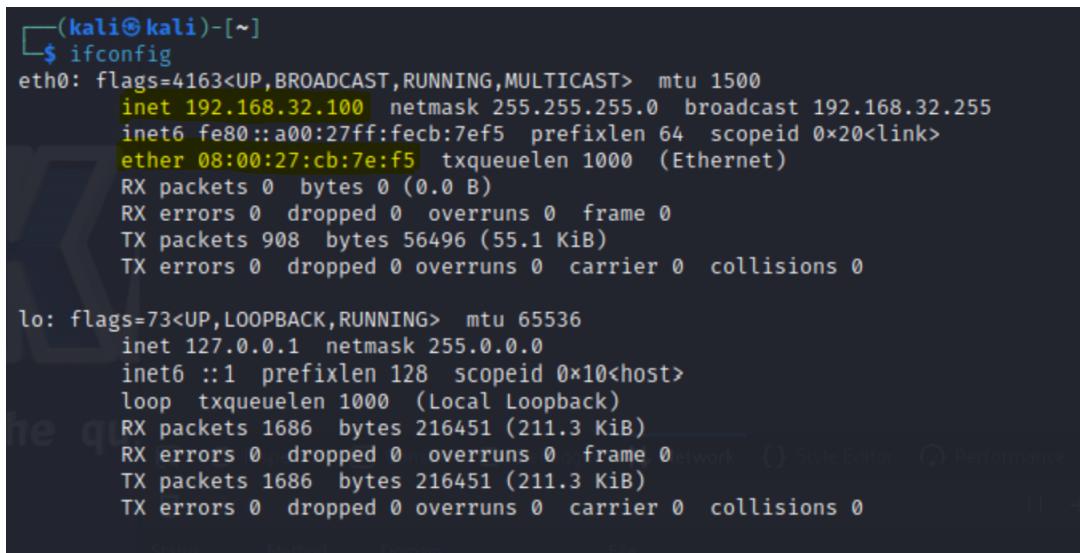
source /etc/network/interfaces.d/*

# 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
```

Fig. 1 - Configurazione di rete Kali Linux

Ho verificato che i parametri di rete fossero stati acquisiti correttamente dalla macchina tramite il comando `ifconfig` (parametro `inet`). Tramite questo comando è possibile anche individuare l'indirizzo fisico (MAC Address) dell'interfaccia `eth0` configurata corrispondente a `08:00:27:cb:7e:f5` (parametro `ether`)



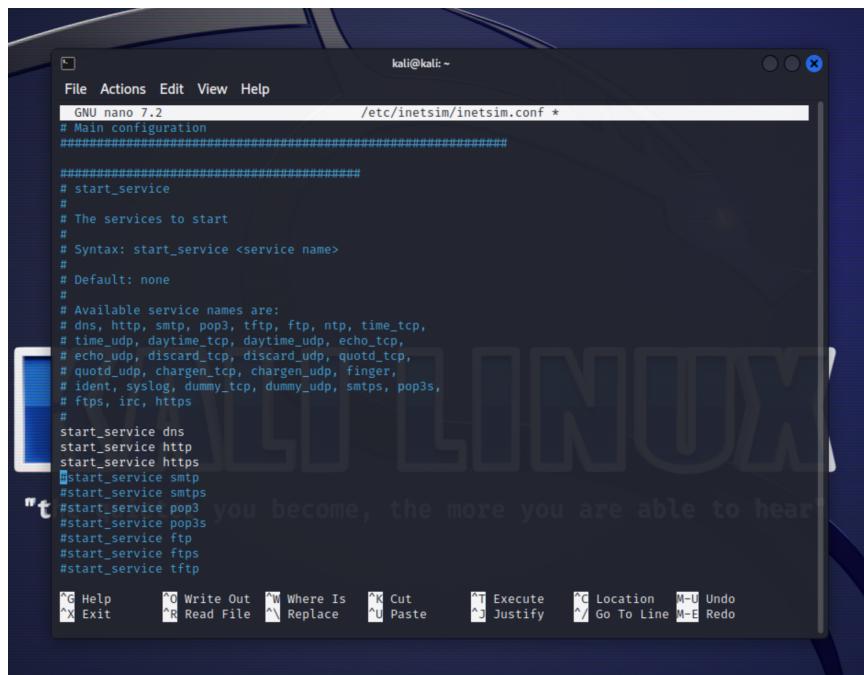
```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255
        inet6 fe80::a00:27ff:fe2b:7ef5 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
                RX packets 0 bytes 0 (0.0 B)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 908 bytes 56496 (55.1 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 1686 bytes 216451 (211.3 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1686 bytes 216451 (211.3 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig. 2 - Verifica configurazione di rete Kali Linux

Sono quindi passato all'attivazione dei servizi HTTP/HTTPS e DNS tramite `inetsim`, un tool già presente nella distribuzione Kali Linux che permette di simulare in maniera molto semplice svariati servizi, tra cui i servizi HTTP/HTTPS e DNS richiesti.

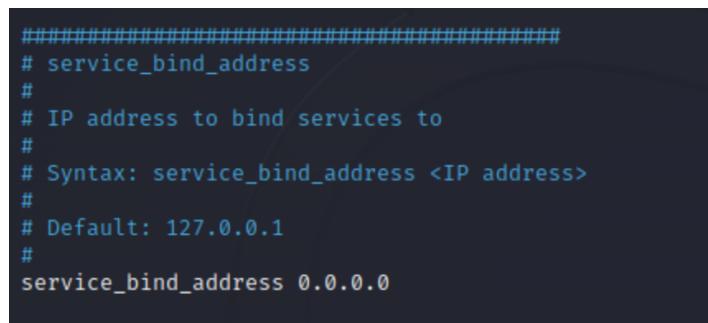
Per la sua configurazione ho editato il file di `inetsim.conf` presente sotto il percorso `/etc/inetsim/inetsim.conf`. Nello specifico ho innanzitutto abilitato i servizi richiesti (server HTTP/HTTPS e DNS), commentando i restanti per disabilitarli



```
File Actions Edit View Help
GNU nano 7.2                               /etc/inetsim/inetsim.conf *
#####
# Main configuration
#####
##### start_service #####
# 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
#
^G Help   ^W Write Out  ^N Where Is  ^K Cut
^X Exit   ^R Read File  ^A Replace  ^U Paste
          ^E Execute  ^J Justify  ^C Location  M-U Undo
          ^L Go To Line M-E Redo
```

Fig 3 - Abilitazione servizi http, https, dns su inetsim

e ho effettuato il bind per ricevere le richieste su tutti gli indirizzi IP impostando il parametro `service_bind_address` come `0.0.0.0`



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

Fig. 4 - Binding inetsim

Relativamente al servizio HTTP/HTTPS non sono necessarie ulteriori configurazioni. InetSim mette già a disposizione una serie di risorse fruibili tramite il server, come mostrato nella immagine seguente relativamente a HTTPS (quelle HTTP sono analoghe).

La risorsa di default, nel caso non specificata, è la pagina sample.html (parametri http default fakefile/https default fakefile)

```
#####
# https_fakefile 574 127.0.0.1          127.0.0.1    TCP      68 3 A
# 5 0.001944357 127.0.0.1          127.0.0.1    TLSv1.3   689 C
# Fake files returned in fake mode based on the file extension
# in the HTTPS request.
# The fake files must be placed in <data-dir>/http/fakefiles
#
# Syntax: https_fakefile <extension> <filename> <mime-type>
#
# Default: none
# 12 0.099389213 127.0.0.1          127.0.0.1    TCP      68 3 A
https_fakefile txt sample.txt      text/plain   127.0.0.1    TCP      68 3 A
https_fakefile htm sample.html    text/html     127.0.0.1    TLSv1.3   241 A
https_fakefile html sample.html   text/html     127.0.0.1    TCP      68 3 A
https_fakefile php sample.html   text/html     127.0.0.1    TLSv1.3   348 A
https_fakefile gif sample.gif    image/gif     127.0.0.1    TCP      68 3 A
https_fakefile jpg sample.jpg    image/jpeg    127.0.0.1    TCP      68 3 A
https_fakefile jpeg sample.jpg   image/jpeg    127.0.0.1    TLSv1.3   92 A
https_fakefile png sample.png   image/png     127.0.0.1    TCP      68 3 A
https_fakefile bmp sample.bmp   image/x-ms-bmp 127.0.0.1    TLSv1.3   92 A
https_fakefile ico favicon.ico  image/x-icon   127.0.0.1    TCP      56 3 A
https_fakefile exe sample_gui.exe x-msdos-program 127.0.0.1    TCP      76 46
https_fakefile com sample_gui.exe x-msdos-program 127.0.0.1    TCP      76 86

Frame 25: 499 bytes on wire (3992 bits), 499 bytes captured (3992 bits)
#####
# https_default_fakefile
# 1 0.001944357 127.0.0.1, Src: 127.0.0.1, Dst: 127.0.0.1
# [Transmission Control Protocol] Src Port: 4060 (4060), Dst Port: 80, Seq: 1, A
# [Stream index: 1]
# The default fake file returned in fake mode if the file extension
# in the HTTPS request does not match any of the extensions
# defined above.
# [Stream index: 1]
# The default fake file must be placed in <data-dir>/http/fakefiles
# [TCP Segment Len: 431]
# Syntax: https_default_fakefile <filename> <mime-type>
# Sequence Number (raw): 391379247
# Default: none
# LNEXT Sequence Number: 432    (relative sequence number)
# ACK Sequence Number: 1 (raw ack number)
# ACK Sequence Number: 2383023590
https_default_fakefile sample.html text/html
```

**Fig. 5 - Risorse server https inetsim**

Ho quindi provveduto alla configurazione specifica del servizio DNS impostando i parametri `dns_default_ip` con l'indirizzo stesso della macchina e associando quest'ultimo con l'indirizzo simbolico `epicode.internal` (record DNS), come illustrato in figura.

```
GNU nano 7.2          /etc/inetsim/inetsim.conf
#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 192.168.32.100

#####
# dns_default_hostname
#
# Default hostname to return with DNS replies
#
# Syntax: dns_default_hostname <hostname>
#
# Default: www
#
#dns_default_hostname www

#####
# dns_default_domainname
#
# Default domain name to return with DNS replies
#
# Syntax: dns_default_domainname <domain name>
#
# 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 episode.internal 192.168.32.100

^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line
```

Fig. 6 - Configurazione DNS

Ho infine avviato la simulazione dei servizi con il comando `sudo inetsim`. Come è possibile vedere dall'immagine seguente, i servizi http, https e dns sono tutti e tre attivi ed in ascolto rispettivamente sulle porte 80, 443 e 53 (porte di default).

```

kali㉿kali: ~
File Actions Edit View Help
└$ sudo nano /etc/inetsim/inetsim.conf
└(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 56137) ==
Session ID: 56137
Listening on: 0.0.0.0
Real Date/Time: 2023-11-17 14:44:23
Fake Date/Time: 2023-11-17 14:44:23 (Delta: 0 seconds)
Forking services ...
* https_443_tcp - started (PID 56141)
* dns_53_tcp_udp - started (PID 56139)
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.
* http_80_tcp - started (PID 56140)
done.
Simulation running.

```

Fig. 7 - Avvio Inetsim

## Configurazione Windows 7

Come per la macchina Kali Linux, ho provveduto alla configurazione dei parametri di rete impostando l'indirizzo di rete statico 192.168.32.101 come da specifiche e l'indirizzo del server DNS con quello della macchina Kali Linux (192.168.32.100) in maniera da utilizzare il servizio DNS configurato in precedenza.

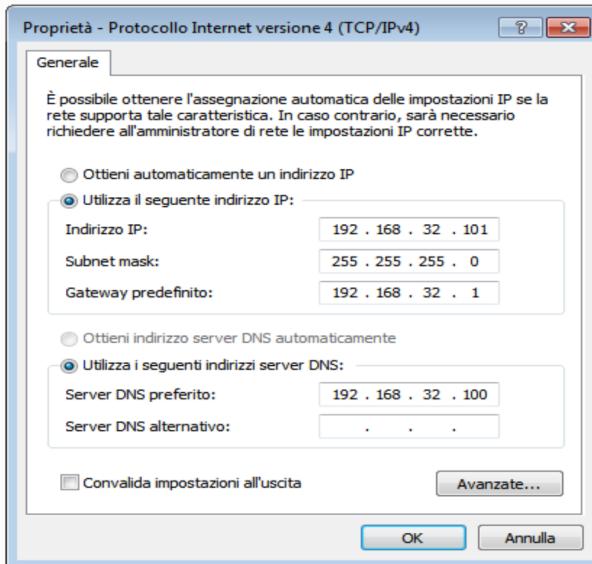
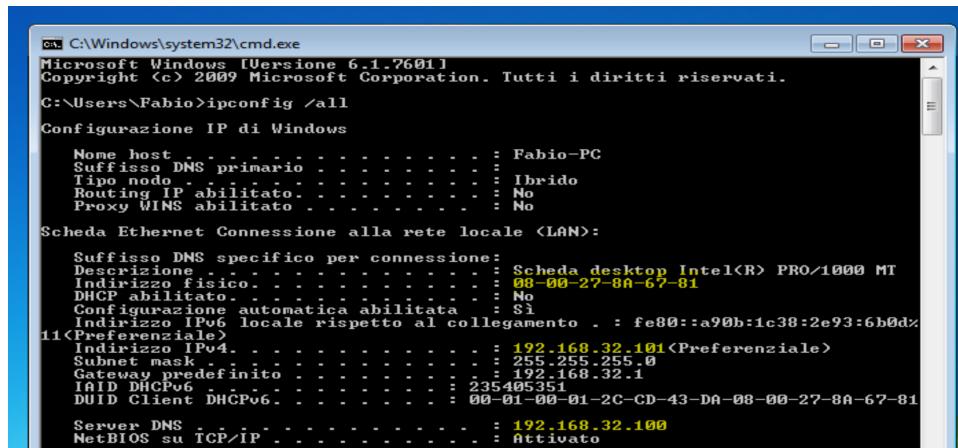


Fig. 8 - Configurazione di rete macchina Windows 7

Ho verificato che i parametri di rete fossero stati acquisiti correttamente dalla macchina tramite il comando ipconfig /all. Tramite questo comando è possibile inoltre individuare l'indirizzo fisico (MAC Address) dell'interfaccia Ethernet (LAN) configurata corrispondente a 08:00:27:8A:67:81 (parametro Indirizzo fisico)

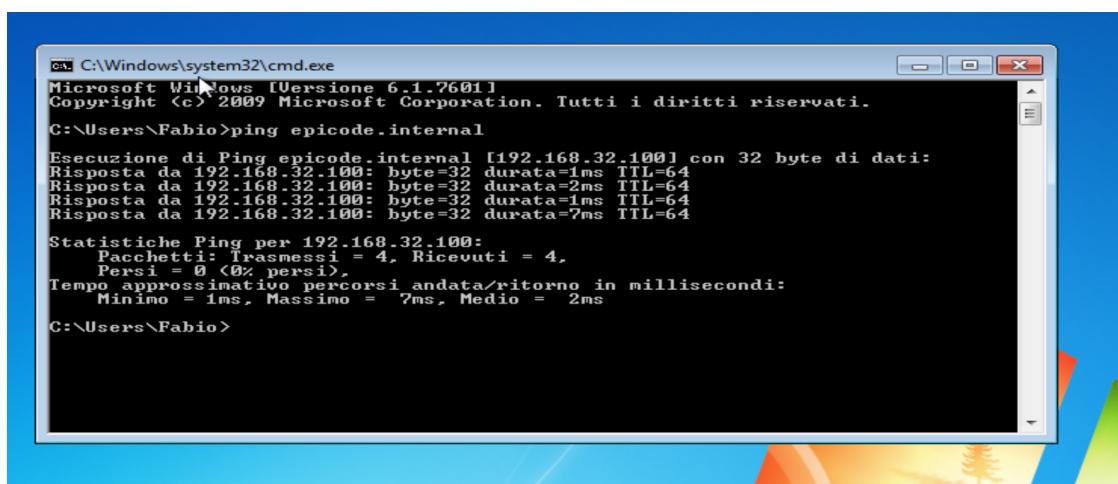


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versione 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. Tutti i diritti riservati.
C:\Users\Fabio>ipconfig /all
Configurazione IP di Windows
  Nome host . . . . . : Fabio-PC
  Suffixo DNS primario . . . . . :
  Tipo nodo . . . . . : Ubrido
  Routing IP abilitato . . . . . : No
  Proxy WINS abilitato . . . . . : No
Scheda Ethernet Connessione alla rete locale <LAN>:
  Suffixo DNS specifico per connessione:
  Descrizione . . . . . : Scheda desktop Intel(R) PRO/1000 MT
  Indirizzo fisico . . . . . : 08-00-27-8A-67-81
  DHCP abilitato . . . . . : No
  Configurazione automatica abilitata . . . . . : Si
  Utilizzo IPv6 locale rispetto al collegamento . . . : fe80::a90b:1c38:2e93:6b0d%11<Preferenziale>
  Indirizzo IPv4 . . . . . : 192.168.32.101<Preferenziale>
  Subnet mask . . . . . : 255.255.255.0
  Gateway predefinito . . . . . : 192.168.32.1
  IAID DHCPv6 . . . . . : 235405351
  DUID Client DHCPv6 . . . . . : 00-01-00-01-2C-CD-43-DA-08-00-27-8A-67-81
  Server DNS su TCP/IP . . . . . : 192.168.32.100
  NetBIOS su TCP/IP . . . . . : Attivato
```

Fig. 9 - Verifica configurazione di rete Windows 7

### Test configurazione

Per testare la configurazione ho provveduto in primo luogo ad eseguire il comando di ping verso l'indirizzo simbolico `epicode.internal` dalla macchina Windows 7 come nell'immagine seguente, ricevendo risposta dall'indirizzo IP corrispondente alla macchina Kali Linux (192.168.32.100). Ciò dimostra che le due macchine sono connesse tra di loro e che il DNS risolve correttamente l'indirizzo simbolico `epicode.internal` permettendo l'accesso alla macchina tramite esso.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versione 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. Tutti i diritti riservati.
C:\Users\Fabio>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=2ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=7ms TTL=64
Statistiche Ping per 192.168.32.100:
  Pacchetti: Trasmesse = 4, Ricevuti = 4,
  Persi = 0 <0% persi>
  Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 1ms, Massimo = 7ms, Medio = 2ms
C:\Users\Fabio>
```

Fig. 10 - Ping verso epicode.internal da Windows 7

Contestualmente ho catturato con Wireshark lo scambio di pacchetti DNS (vedi figura), in cui è evidenziato il pacchetto di risposta al cui interno è presente il record DNS (di tipo A) con l'associazione tra indirizzo simbolico `epicode.internal` e l'indirizzo fisico `192.168.32.100` (macchina Kali Linux)

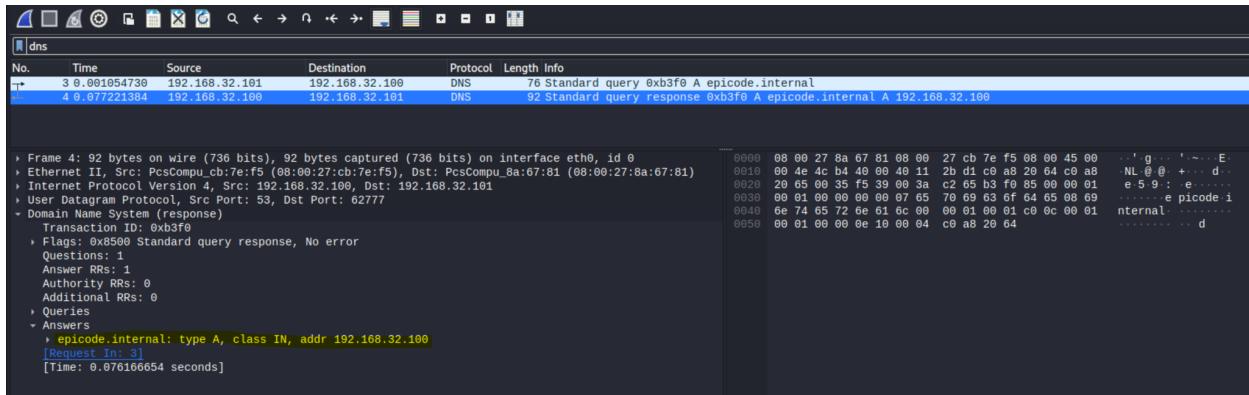


Fig. 11 - Cattura richiesta DNS tramite Wireshark

Ho quindi verificato il funzionamento del server HTTPS e del server HTTP effettuando una richiesta rispettivamente all'indirizzo <https://epicode.internal> e all'indirizzo <http://epicode.internal>. In entrambi i casi il server risponde con l'invio della risorsa di default ovvero la pagina sample.html come mostrato in figura.

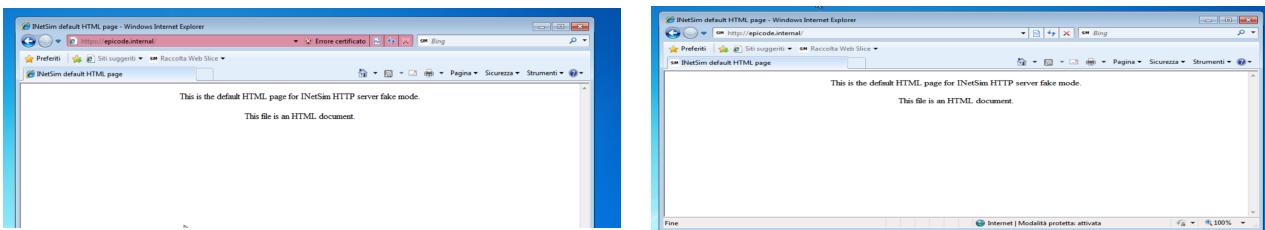


Fig. 12 - Richieste tramite browser al server `epicode.internal`

## Analisi comunicazione HTTP/HTTPS con Wireshark

### Test 1: richiesta HTTPS al server

Tramite il browser web da Windows 7 ho effettuato una richiesta HTTPS al server `epicode.internal` e intercettato la comunicazione con Wireshark in figura filtrando per indirizzo ip, protocollo (di tipo TCP) e porta (443, porta HTTPS di default). Essendo la chiamata rivolta alla radice del server, esso risponderà fornendo la risorsa di default, ovvero la pagina `sample.html`

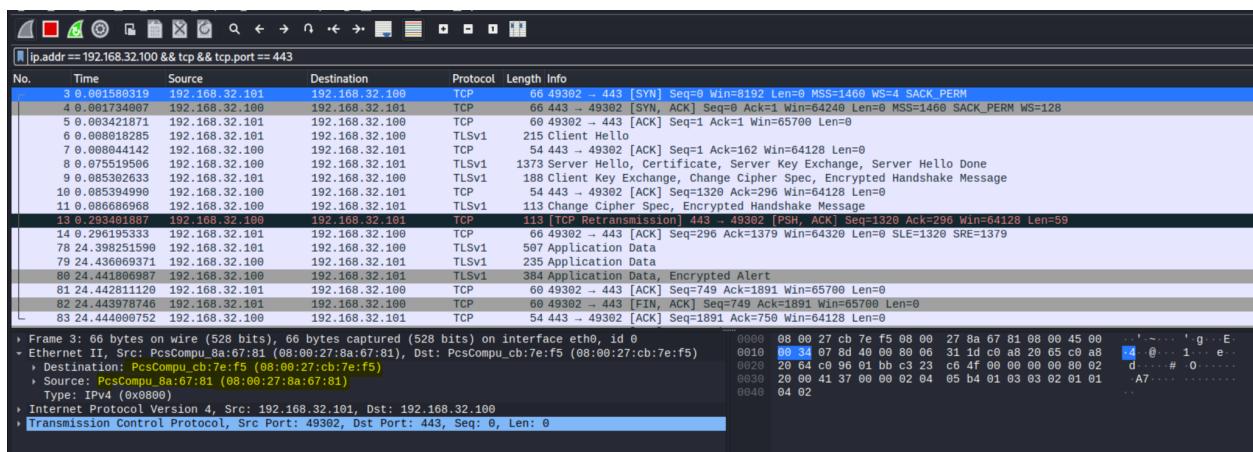


Fig. 13 - Cattura Wireshark richiesta HTTPS

Dato che HTTPS prevede che la comunicazione sia criptata tramite protocollo TLS, non è possibile vedere nel dettaglio la comunicazione. E' comunque possibile individuare il three-way handshake iniziale per stabilire la connessione (SYN, SYN-ACK, ACK, righe 3-5) e la richiesta di chiusura da parte del client (FIN-ACK, riga 82, l'handshake completo non è visibile in questo caso in quanto la connessione è per l'appunto criptata).

Inoltre analizzando ad esempio il dettaglio del primo pacchetto nell'apposita sezione in basso a sinistra di Wireshark, è possibile individuare l'indirizzo fisico (MAC Address) del computer sorgente (la macchina Windows 7) corrispondente a 08:00:27:8a:67:81 (campo Src, sezione Ethernet II) e quella del computer destinatario (la macchina Kali Linux) con MAC address 08:00:27:cb:7e:f5 (campo Dst, sezione Ethernet II) relativamente ad un pacchetto di richiesta (campi evidenziati). Essi corrispondono rispettivamente a quelli individuati tramite i comandi `ipconfig /all` (su Windows 7) e `ifconfig` (su Kali Linux) in precedenza (vedi fig. 2 e fig. 9).

## Test 2: richiesta HTTP

L'immagine seguente mostra la cattura dei pacchetti mediante Wireshark in seguito alla richiesta di accesso tramite browser all'indirizzo `http://epicode.internal`

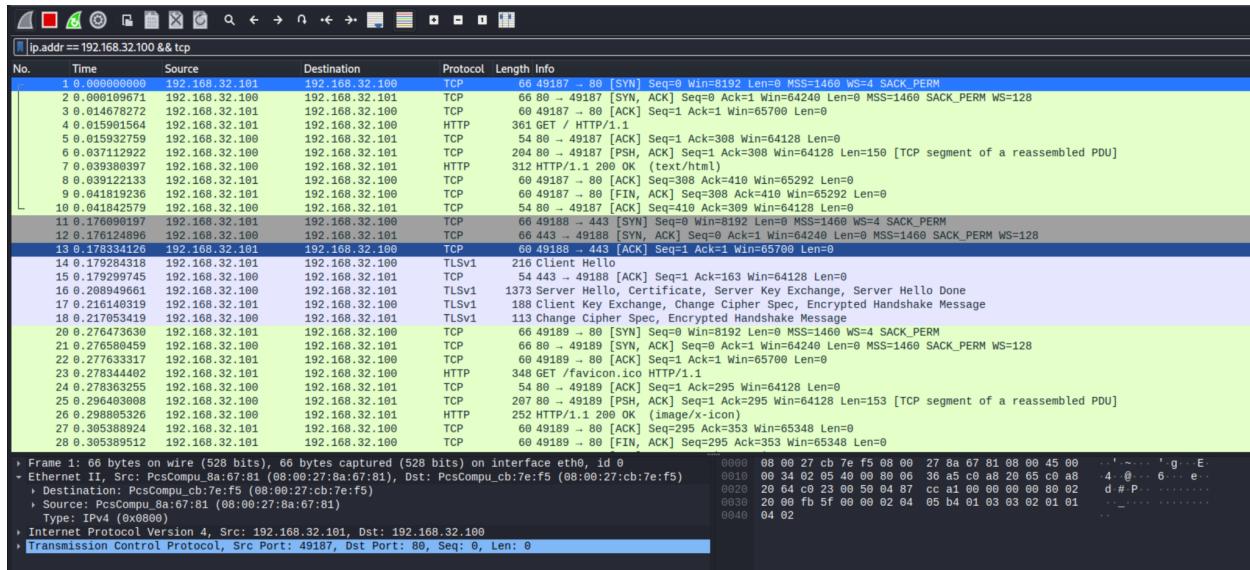


Fig. 14 - Cattura Wireshark richiesta HTTP

In questo caso, dato che il protocollo HTTP è in chiaro, è possibile analizzare con Wireshark l'intero scambio di pacchetti. In particolare è possibile individuare una prima richiesta HTTP GET alla radice del server tramite il quale il server restituisce la pagina principale (`sample.html`, riga 4) e poi, trattandosi di una pagina ipertestuale, delle risorse contestuali presenti nella pagina (immagini, video, etc.). Nel caso in esame viene richiesta tramite HTTP GET la risorsa `favicon.ico` (riga 23). Si tratta di una piccola immagine che rappresenta il sito e che viene visualizzata nella barra degli indirizzi del browser accanto all'URL e nella scheda del browser. In entrambi i casi il server risponde con il codice 200 OK (righe 7 e 26 rispettivamente) ad indicare che la richiesta è stata completata con successo e restituisce la risorsa nel corpo del medesimo pacchetto. Per ciascuna risorsa è inoltre possibile individuare l'apertura della connessione (SYN, SYN-ACK, ACK) e la chiusura (FIN, FIN-ACK, ACK) mediante three-way handshake. Gli indirizzi MAC evidenziati nella sezione di dettaglio sono naturalmente i medesimi, essendo coinvolte nella comunicazione le stesse macchine.