

## INDICE

INDICE .....	1
INTRODUZIONE ED OBIETTIVO.....	2
CONFIGURAZIONE INDIRIZZI IP DEGLI HOST.....	3
VERIFICA DELLA COMUNICAZIONE TRA HOST.....	4-5
CONFIGURAZIONE DELLA RISOLUZIONE LOCALE DEI NOMI.....	6-7
CONFIGURAZIONE E AVVIO DI INETSIM.....	8-9-10
VERIFICA DEL FUNZIONAMENTO DEI SERVIZI SIMULATI.....	11-12-13
VERIFICA DELL'ACCESSO AI SERVIZI DAL SERVER.....	14-15
CONCLUSIONE FINALE.....	16

## INTRODUZIONE ED OBIETTIVO

Il presente elaborato documenta un'attività di laboratorio , in ambiente isolato, volta a mettere in comunicazione due macchine virtuali ( **Windows** e **Kali**) configurate per comunicare direttamente tra di loro ,assegnando manualmente ad entrambi gli host gli indirizzi IP.

Kali svolge il ruolo di server ospitando **INetSim** per simulare i servizi *DNS*, *HTTP* e *HTTPS* e pubblicare localmente il dominio '**epicode.internal**'.

Windows opera come client, utilizzato per effettuare le verifiche di connettività, di risoluzione dei nomi e di accesso ai servizi applicativi.

L'obiettivo di questo laboratorio è comprendere e dimostrare il funzionamento dei principali meccanismi di comunicazione all'interno di una rete locale isolata.

Attraverso l'interazione tra un client e un server, l'esercitazione intende sviluppare la capacità di configurare correttamente gli indirizzi di rete, gestire la risoluzione dei nomi e analizzare il traffico generato dai protocolli.

## CONFIGURAZIONE INDIRIZZI IP DEGLI HOST

Nel primo passaggio è stata configurata la connettività di base tra i due sistemi, assegnando manualmente gli indirizzi IP alle rispettive interfacce di rete.

### WINDOWS

Indirizzo IP:	192 . 168 . 32 . 101
Subnet mask:	255 . 255 . 255 . 0
Gateway predefinito:	192 . 168 . 32 . 1

Sul sistema **Windows** è stato impostato l'indirizzo *192.168.32.101* avente subnet 255.255.255.0 e specificando come gateway predefinito l'indirizzo locale.

### KALI



Successivamente, sul sistema **Kali Linux** è stato configurato l'indirizzo *192.168.32.100* , netmsk 24 e come Gateway 192.168.32.1 .

Questa configurazione manuale assicura un controllo completo sull'indirizzamento di rete in quanto è una condizione necessaria per garantire stabilità e successo nelle prove successive di connettività.

## VERIFICA DELLA COMUNICAZIONE TRA HOST

Dopo la configurazione degli indirizzi IP, è stata verificata la connettività tra le due macchine virtuali al fine di confermare la corretta comunicazione sulla rete interna.

### WINDOWS

```
C:\> Prompt dei comandi

Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user>ping -n 4 192.168.32.100

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=8ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=2ms 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 = 8ms, Medio = 2ms

C:\Users\user>arp -a

Interfaccia: 192.168.32.101 --- 0x3
    Indirizzo Internet      Indirizzo fisico          Tipo
    192.168.32.100         08-00-27-9d-89-4b        dinamico
    192.168.32.255         ff-ff-ff-ff-ff-ff        statico
    224.0.0.9              01-00-5e-00-00-09        statico
    224.0.0.22             01-00-5e-00-00-16        statico
    224.0.0.252           01-00-5e-00-00-fc        statico

C:\Users\user>
```

Dal sistema **Windows** è stato eseguito un test di connettività mediante il comando *ping* verso l'indirizzo IP del server Kali, ottenendo risposta positiva e quindi confermando che i pacchetti ICMP raggiungono correttamente la destinazione.

## KALI

```
(kali㉿kali)-[~]  
$ ping -c 4 192.168.32.101  
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=0.938 ms  
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=1.81 ms  
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=1.16 ms  
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=0.650 ms  
  
— 192.168.32.101 ping statistics —  
4 packets transmitted, 4 received, 0% packet loss, time 3007ms  
rtt min/avg/max/mdev = 0.650/1.139/1.813/0.428 ms  
  
(kali㉿kali)-[~]  
$ ip neigh show  
  
192.168.32.101 dev eth0 lladdr 08:00:27:4e:73:95 STALE
```

Successivamente, la stessa operazione è stata effettuata in direzione opposta, eseguendo il *ping* dal sistema **Kali Linux** verso l'indirizzo IP assegnato al client Windows.

Anche in questo caso la risposta è risultata positiva, dimostrando la piena bidirezionalità della comunicazione e la corretta configurazione delle interfacce di rete su entrambi gli host.

## CONFIGURAZIONE DELLA RISOLUZIONE LOCALE DEI NOMI


Per consentire al client Windows di risolvere correttamente il nome del server è stata configurata manualmente una voce all'interno del file *hosts*, associando il dominio ***epicode.internal*** all'indirizzo IP del server.

Questa operazione permette che ogni richiesta verso il nome del dominio , venga instradata direttamente verso il corretto host di destinazione.

```
# This file contains the mappings of IP addresses to host
names. Each
# entry should be kept on an individual line. The IP address
should
# be placed in the first column followed by the corresponding
host name.
# The IP address and the host name should be separated by at
least one
# space.
#
# Additionally, comments (such as these) may be inserted on
individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97      rhino.acme.com          # source
server
#      38.25.63.10      x.acme.com              # x client
host

# localhost name resolution is handled within DNS itself.
#      127.0.0.1        localhost
#      ::1              localhost
192.168.32.100 epicode.internal
```

## WINDOWS

 Prompt dei comandi

```
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user>ping -n 3 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

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 3, Ricevuti = 3,
    Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 0ms, Massimo = 1ms, Medio = 0ms

C:\Users\user>
```

Completata la configurazione, è stato eseguito un test di connettività tramite il comando *ping* **epicode.internal** dal sistema **Windows**, che ha restituito risposta positiva con tempi di latenza minimi, confermando la corretta risoluzione del nome e la raggiungibilità del server.

## KALI

```
(kali㉿kali)-[~]
$ sudo nano /etc/hosts

(kali㉿kali)-[~]
$ ping -c 4 epicode.internal
PING epicode.internal (192.168.32.100) 56(84) bytes of data.
64 bytes from epicode.internal (192.168.32.100): icmp_seq=1 ttl=64 time=0.060 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=4 ttl=64 time=0.060 ms

— epicode.internal ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3056ms
rtt min/avg/max/mdev = 0.051/0.060/0.070/0.006 ms
```

In ugual modo, anche sul sistema **Kali Linux** è stata effettuata la modifica del file */etc/hosts*, inserendo l'associazione tra l'indirizzo IP del server e il nome di dominio **epicode.internal**, così da permettere al sistema di riconoscere il nome in modo diretto.

Dopo l'aggiornamento del file, è stato eseguito il comando *ping epicode.internal*, ottenendo risposta positiva con tempi di latenza molto ridotti a conferma della corretta risoluzione del nome e della piena raggiungibilità del servizio all'interno della rete isolata.

## CONFIGURAZIONE E AVVIO DI INETSIM

In questa fase è stato configurato e avviato **INetSim** sul sistema **Kali Linux**, con lo scopo di simulare i principali servizi di rete necessari ai test successivi.

```
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

#####
# 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
```

All'interno del file di configurazione `'/etc/inetsim/inetsim.conf'` sono stati abilitati i servizi *DNS*, *HTTP* e *HTTPS* impostando come `'service_bind_address'` quello del server Kali, in modo che i servizi risultassero raggiungibili dalla rete interna.



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

In seguito, è stato specificato nel medesimo file anche l'indirizzo di risposta predefinito per il servizio DNS affinché che le interrogazioni effettuate dai client venissero risolte correttamente verso l'indirizzo configurato.

```
.  
(kali@kali)-[~]  
$ perl -MNet::DNS -e 'print "$Net::DNS::VERSION\n"'  
1.37
```

Prima di procedere con l'avvio di *INetSim*, è stato necessario installare una versione compatibile del linguaggio *Perl* e del relativo modulo '*Net::DNS*' indispensabile per il corretto funzionamento del servizio DNS.

Per eseguire l'installazione, la macchina **Kali Linux** è stata temporaneamente impostata in modalità *NAT*, così da consentire l'accesso a Internet e il download dei pacchetti richiesti.

Una volta completata l'installazione, è stato verificato che la versione di *Perl* risultasse correttamente aggiornata e pienamente compatibile con *INetSim*, permettendo così l'avvio regolare del software e dei relativi servizi di rete.

```
(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 9629) ≡  
Session ID:      9629  
Listening on:    127.0.0.1  
Real Date/Time:  2025-10-19 08:17:48  
Fake Date/Time:  2025-10-19 08:17:48 (Delta: 0 seconds)  
Forking services ...  
  * dns_53_tcp_udp - started (PID 9631)  
  * https_443_tcp - started (PID 9633)  
  * http_80_tcp - started (PID 9632)  
done.  
Simulation running.  
█
```

Completata la configurazione, il servizio è stato avviato tramite il comando *sudo inetsim*, che ha confermato l'attivazione dei processi relativi alle porte 53 (*DNS*), 80 (*HTTP*) e 443 (*HTTPS*).

## VERIFICA DEL FUNZIONAMENTO DEI SERVIZI SIMULATI

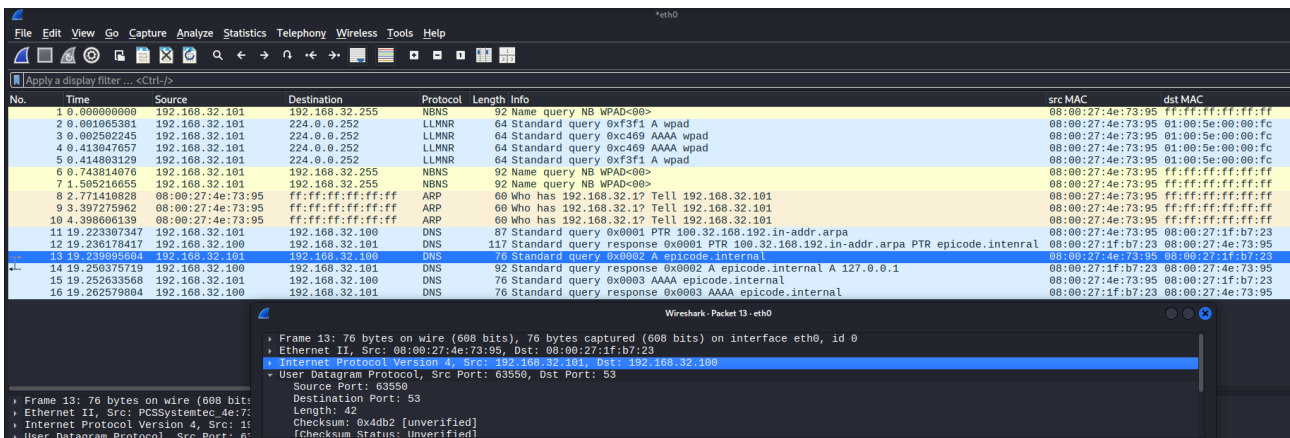
### SERVIZIO DI RETE DNS

```
C:\Users\user>nslookup epicode.intenal 192.168.32.100
Server: epicode.intenal
Address: 192.168.32.100

Nome: epicode.intenal
Address: 192.168.32.100
```

Per verificare il corretto funzionamento del servizio DNS simulato da *INetSim*, è stata eseguita una richiesta di risoluzione dal sistema Windows tramite il comando *'nslookup epicode.intenal'*.

Il risultato ha mostrato la corretta associazione del nome di dominio all'indirizzo IP del server, confermando che il servizio DNS era attivo e rispondeva alle interrogazioni provenienti dalla rete interna.

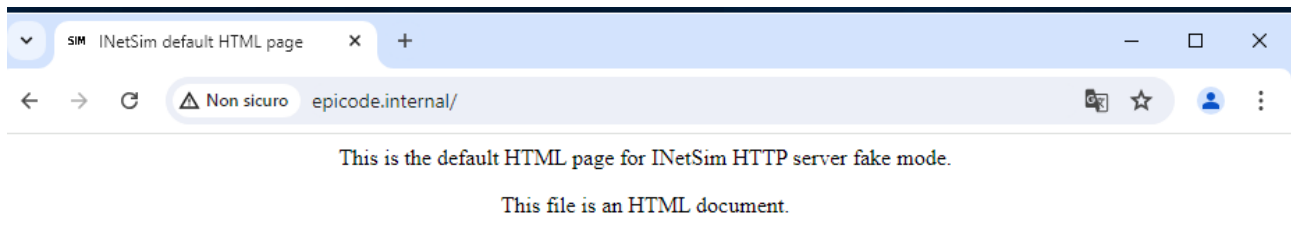


Per completare la verifica, è stata effettuata una cattura del traffico in *Wireshark*, filtrando i pacchetti DNS ; con lo scopo di osservare la query e la risposta relative al dominio *epicode.intenal*.

Durante l'analisi è stato evidenziato il campo *Internet Protocol Version 4 (IPv4)*, in cui risultano visibili gli indirizzi IP di origine e destinazione corrispondenti al client e al server, confermando la corretta comunicazione e il regolare funzionamento del servizio di risoluzione dei nomi.

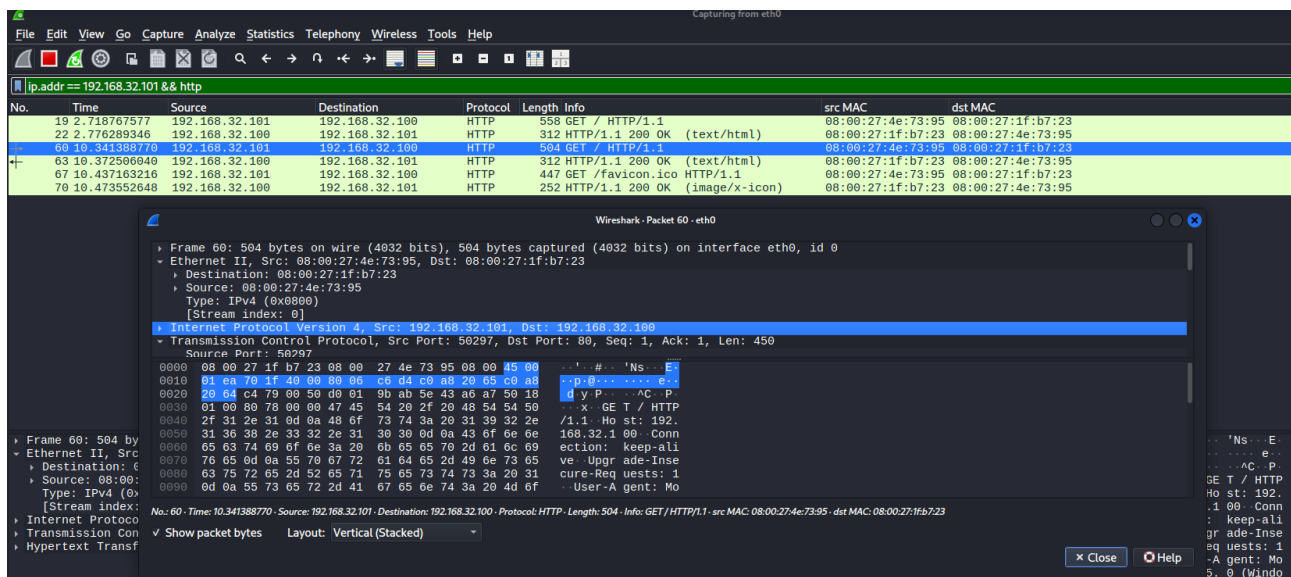
## PROTOCOLLO HTTP

Dopo aver confermato il corretto funzionamento del servizio DNS, è stata verificata la comunicazione tramite il protocollo HTTP simulato da *INetSim* sul server Kali.



Dal sistema Windows, configurato come client, è stato aperto il browser e digitato l'indirizzo '*http://epicode.internal*'.

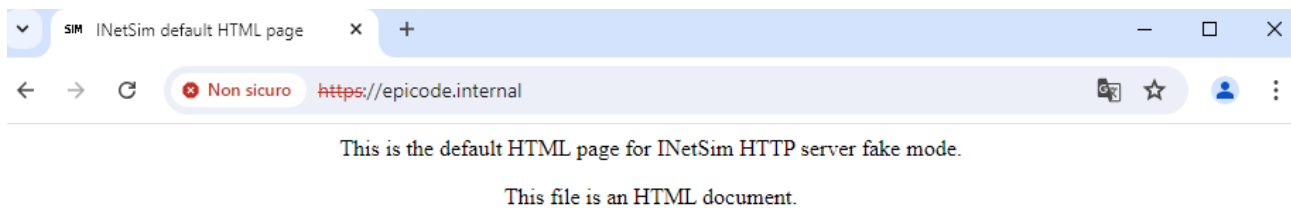
La richiesta è stata elaborata correttamente e il browser ha visualizzato la pagina predefinita generata da INetSim dimostrando che il servizio era attivo e raggiungibile.



A completamento della verifica, è stata eseguita una cattura del traffico con *Wireshark*, applicando un filtro HTTP per analizzare lo scambio tra client e server. L'analisi ha evidenziato le richieste e le risposte e, nel riquadro *IPv4*, la corretta corrispondenza degli indirizzi IP di origine e destinazione confermando il regolare funzionamento del protocollo.

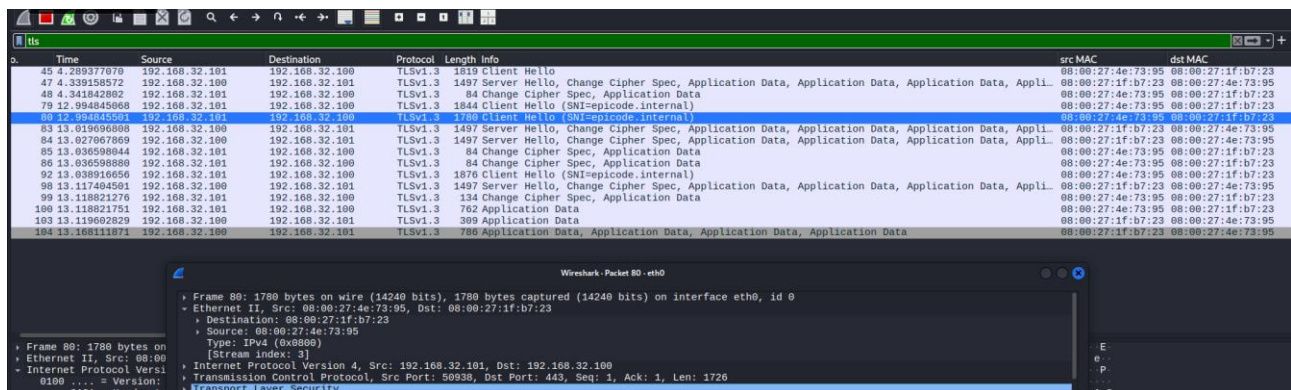
## PROTOCOLLO HTTPS

Successivamente è stata effettuata la verifica del protocollo **HTTPS**, anch'esso simulato da *INetSim* sul server Kali.



Dal sistema Windows è stato aperto il browser e digitato l'indirizzo <https://epicode.internal>, con l'obiettivo di verificare la comunicazione.

Il browser ha correttamente raggiunto il server, visualizzando la pagina generata da *INetSim*.



Anche in questo caso è stata avviata una cattura del traffico con Wireshark, applicando un filtro *TLS* per osservare il protocollo https e le informazioni relative alla sessione cifrata.

L'analisi ha confermato l'instaurazione della connessione sicura tra client e server e la corretta corrispondenza degli indirizzi IP nei pacchetti IPv4, validando il corretto funzionamento del servizio HTTPS.

Per i requisiti richiesti dalla traccia, in Wireshark sono state create colonne personalizzate dedicate alla visualizzazione degli indirizzi MAC di sorgente e di destinazione (*eth.src* e *eth.dst*) ; in maniera tale da evidenziare in modo chiaro l'indirizzamento di livello 2 durante l'analisi del traffico e rendere più immediata l'osservazione dei pacchetti scambiati tra i due host.

## VERIFICA DELL'ACCESSO AI SERVIZI DAL SERVER

Per completare l'attività di verifica, è stato eseguito un ulteriore test direttamente dal sistema Kali Linux, configurato come server.

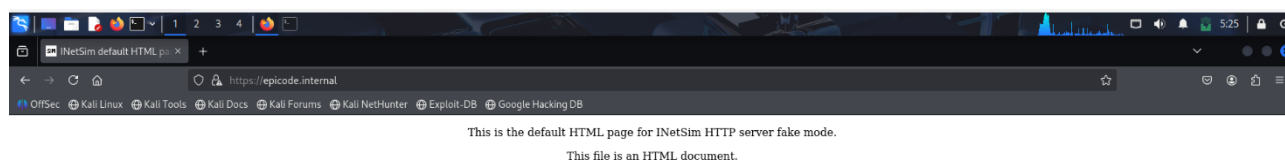
### DNS

Time	Source	Destination	Protocol	Length	Info
74 2.077312582	192.168.32.100	192.168.32.100	DNS	87	Standard query response 0x0065 A example.org A 127.0.0.1
75 2.077718996	192.168.32.100	192.168.32.100	DNS	84	Standard query 0xc0ae A detectportal.firefox.com
76 2.078053802	192.168.32.100	192.168.32.100	DNS	84	Standard query 0x785a A detectportal.firefox.com
77 2.078057114	192.168.32.100	192.168.32.100	DNS	84	Standard query 0x3259 AAAA detectportal.firefox.com
78 2.103587918	192.168.32.100	192.168.32.100	DNS	87	Standard query response 0x92d0 A example.org A 127.0.0.1
79 2.109436379	192.168.32.100	192.168.32.100	DNS	71	Standard query response 0x61d0 AAAA example.org
80 2.112936134	192.168.32.100	192.168.32.100	DNS	89	Standard query response 0x2fd0 A ipv4only.arpa A 127.0.0.1
81 2.115974021	192.168.32.100	192.168.32.100	DNS	73	Standard query response 0xb0d0 AAAA ipv4only.arpa
82 2.118890201	192.168.32.100	192.168.32.100	DNS	100	Standard query response 0xc0ae A detectportal.firefox.com A 127.0.0.1
85 2.123149708	192.168.32.100	192.168.32.100	DNS	100	Standard query response 0x785a A detectportal.firefox.com A 127.0.0.1
88 2.133137480	192.168.32.100	192.168.32.100	DNS	84	Standard query response 0x3259 AAAA detectportal.firefox.com
89 2.133893433	192.168.32.100	192.168.32.100	DNS	84	Standard query 0x68b3 A detectportal.firefox.com
90 2.133896724	192.168.32.100	192.168.32.100	DNS	84	Standard query 0x50b2 AAAA detectportal.firefox.com
91 2.143151092	192.168.32.100	192.168.32.100	DNS	100	Standard query response 0x68b3 A detectportal.firefox.com A 127.0.0.1
92 2.157615157	192.168.32.100	192.168.32.100	DNS	84	Standard query response 0x50b2 AAAA detectportal.firefox.com
115 57.115456824	192.168.32.100	192.168.32.100	DNS	87	Standard query 0xc590 A safebrowsing.googleapis.com
116 57.115481074	192.168.32.100	192.168.32.100	DNS	87	Standard query 0xab95 AAAA safebrowsing.googleapis.com
117 57.161996719	192.168.32.100	192.168.32.100	DNS	103	Standard query response 0xc590 A safebrowsing.googleapis.com A 127.0.0.1
118 57.171037630	192.168.32.100	192.168.32.100	DNS	87	Standard query 0xab95 AAAA safebrowsing.googleapis.com
121 78.252200953	192.168.32.100	192.168.32.100	DNS	85	Standard query 0x27a8 A push.services.mozilla.com
122 78.252212431	192.168.32.100	192.168.32.100	DNS	85	Standard query 0xfba6 AAAA push.services.mozilla.com
125 78.258390100	192.168.32.100	192.168.32.100	DNS	101	Standard query response 0x27a8 A push.services.mozilla.com A 127.0.0.1
126 78.260219519	192.168.32.100	192.168.32.100	DNS	85	Standard query response 0xfba6 AAAA push.services.mozilla.com

Per completare l'analisi, è stata eseguita una cattura del traffico DNS direttamente sul sistema Kali Linux, con l'obiettivo di osservare il comportamento del servizio dal lato server. La cattura ha mostrato numerose richieste e risposte DNS originate dal sistema stesso, confermando l'attività locale del servizio simulato da INetSim. Rispetto alla cattura effettuata su Windows, questa analisi non evidenzia traffico proveniente da un client remoto, ma mostra invece la gestione locale delle richieste da parte del server stesso.

Tale differenza è dovuta al fatto che le comunicazioni interne non attraversano il livello fisico, motivo per cui i **MAC address** non risultano presenti, a differenza della cattura eseguita dal lato client.

### PROTOCOLLO HTTPS

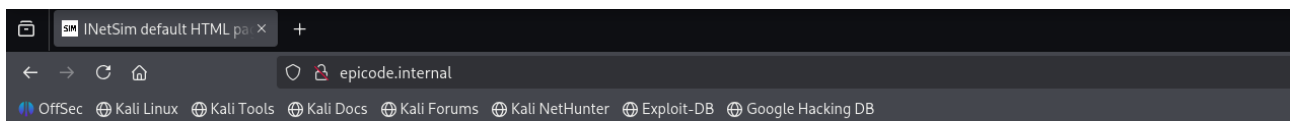


Aperto il browser e digitando l'indirizzo **https://epicode.internal** il sistema ha caricato la pagina predefinita di INetSim, identica a quella visualizzata dal client Windows, confermando che la simulazione dei servizi web era attiva e pienamente operativa anche dal lato server.

No.	Time	Source	Destination	Protocol	Length	Info
98	8.187305264	192.168.32.100	192.168.32.100	TLSv1.3	730	Client Hello (SNI=epicode.internal)
100	8.225091111	192.168.32.100	192.168.32.100	TLSv1.3	1509	Server Hello, Change Cipher Spec, Application Data, Application Data, Application ...
102	8.241456754	192.168.32.100	192.168.32.100	TLSv1.3	146	Change Cipher Spec, Application Data
104	8.241596983	192.168.32.100	192.168.32.100	TLSv1.3	321	Application Data
105	8.242657242	192.168.32.100	192.168.32.100	TLSv1.3	528	Application Data
106	8.242661553	192.168.32.100	192.168.32.100	TLSv1.3	321	Application Data
107	8.258175715	192.168.32.100	192.168.32.100	TLSv1.3	543	Application Data, Application Data, Application Data
109	8.260905854	192.168.32.100	192.168.32.100	TLSv1.3	90	Application Data

È stata inoltre eseguita una cattura del traffico TLS direttamente sul sistema Kali Linux, per analizzare il comportamento del protocollo HTTPS dal lato server. Anche qui gli indirizzi MAC sono assenti in quanto la comunicazione è avvenuta sul livello logico del protocollo di rete.

## PROTOCOLLO HTTP



This is the default HTML page for INetSim HTTP server fake mode.

This file is an HTML document.

Stesso procedimento effettuato digitando `http://epicode.internal` riscontrando il medesimo risultato.

No.	Time	Source	Destination	Protocol	Length	Info
98	7.569983762	192.168.32.100	192.168.32.100	HTTP	402	GET / HTTP/1.1
102	7.582140012	192.168.32.100	192.168.32.100	HTTP	324	HTTP/1.1 200 OK (text/html)

È stata infine eseguita una cattura del traffico HTTP sul sistema Kali Linux, con l'obiettivo di analizzare lo scambio dei pacchetti tra client e server in ambiente locale.

Anche in questo caso i pacchetti catturati riportano come indirizzo di origine e destinazione lo stesso host, confermando che il traffico è stato gestito internamente senza attraversare l'interfaccia fisica questo spiega l'assenza di indirizzi MAC.



## CONCLUSIONE FINALE

L'attività di laboratorio ha permesso di verificare in modo completo il funzionamento dei principali servizi di rete simulati da *INetSim* in un ambiente isolato, composto da due macchine virtuali configurate per la comunicazione diretta.

Attraverso la configurazione manuale degli indirizzi IP e la definizione delle associazioni nei file hosts è stata garantita la corretta risoluzione del dominio locale *epicode.internal* e la piena raggiungibilità tra client e server.

Le analisi condotte con *Wireshark* hanno evidenziato il corretto scambio dei pacchetti relativi ai protocolli *DNS*, *HTTP* e *HTTPS*, mostrando in particolare la coerenza tra gli indirizzi IP e i *MAC address* durante le comunicazioni tra i due host. Le catture interne al server Kali hanno inoltre confermato il comportamento locale dei servizi simulati, in cui l'assenza di indirizzi MAC è coerente con la natura logica del traffico interno.

Nel complesso, i risultati ottenuti dimostrano che l'ambiente è stato configurato correttamente e che i servizi di rete simulati da *INetSim* operano in modo stabile e conforme agli obiettivi del laboratorio, consentendo di osservare e comprendere in maniera diretta il funzionamento dei protocolli fondamentali alla base della comunicazione in rete.