

PROGETTO FINALE MODULO 1

Requisiti e servizi:

Kali Linux: IP 192.168.32.100

Windows: IP 192.168.32.101

HTTPS server: attivo

Servizio DNS per risoluzione nomi di dominio: attivo

In questo esercizio metteremo insieme le competenze acquisite finora. Traccia: Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows) 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.

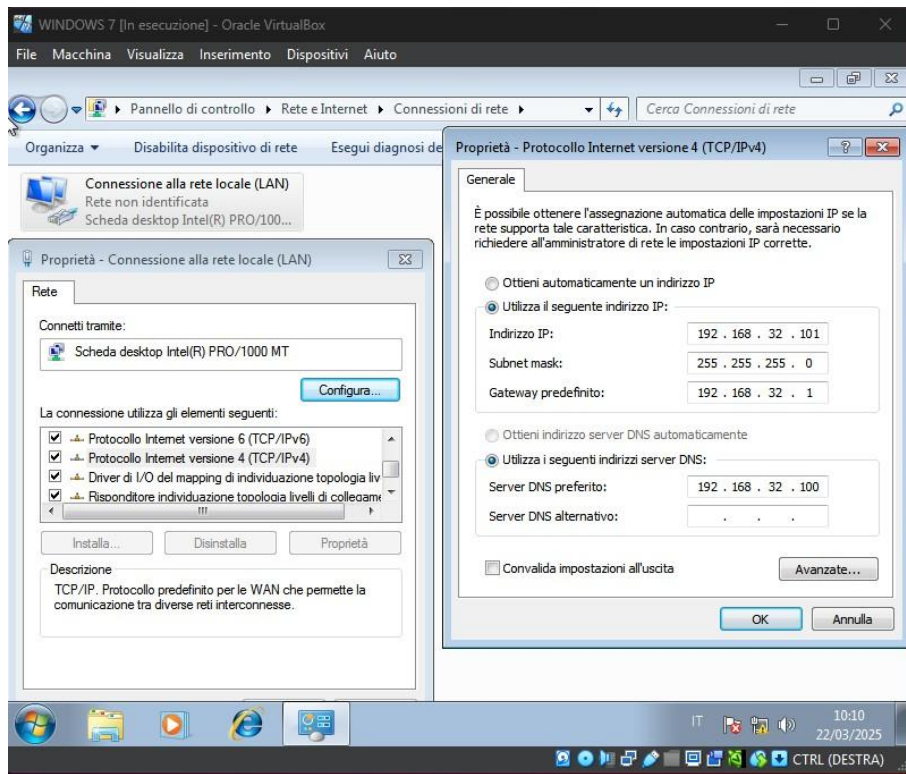
Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS.

Spiegare, motivandole, le principali differenze se presenti.

SVOLGIMENTO

Per lo svolgimento dell'esercizio, come prima cosa, andiamo ad avviare le due macchine virtuali (Windows e Kali) andando poi ad impostare gli indirizzi IP corretti su entrambe.

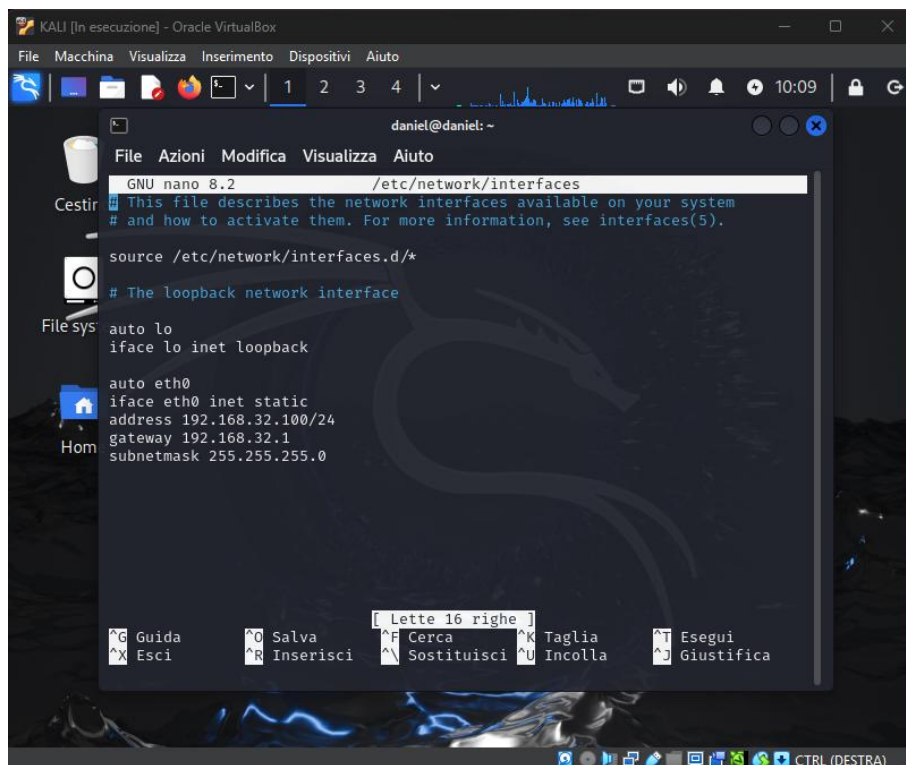
Su Windows la sequenza corretta è: Pannello di controllo – Rete e Internet – Connessione di Rete – Proprietà della scheda – Protocollo IPV4, inseriamo l'indirizzo IP richiesto 192.168.32.101 facendo attenzione ad inserire il DNS con l'IP della macchina Kali, ovvero 192.168.32.100



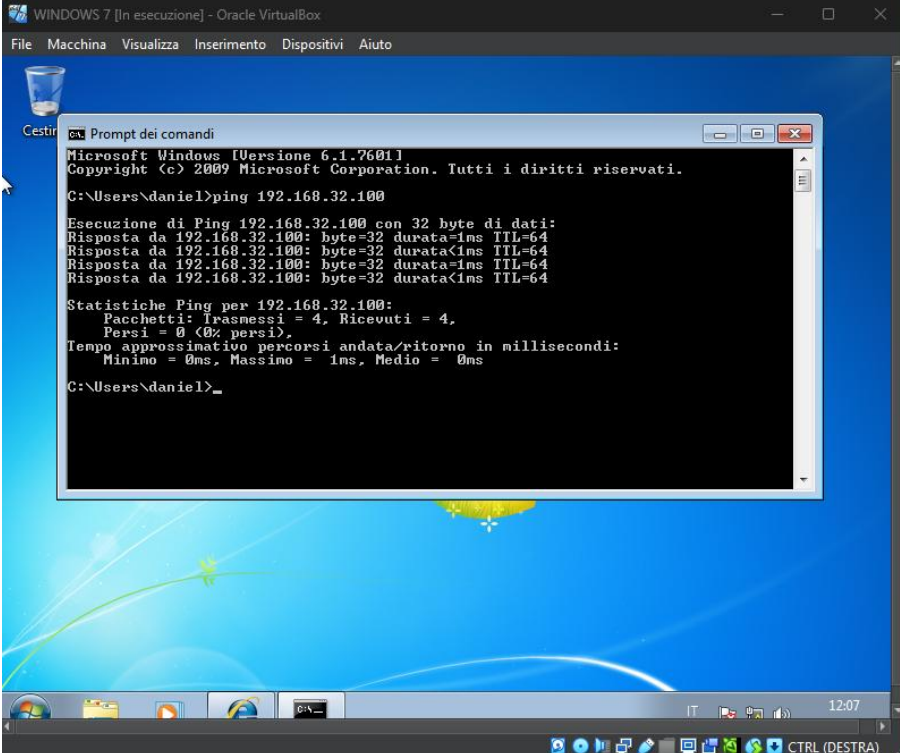
Su Kali apriamo l'emulatore di terminale ed andiamo ad inserire il comando:

sudo nano /etc/network/interfaces

il quale, ci permetterà di modificare l'indirizzo IP ed impostarlo su 192.168.32.100/24



Fatto questo, dobbiamo controllare che le macchine virtuali possano comunicare tra loro, andiamo ad eseguire un ping da Windows a Kali e come possiamo vedere dall'immagine, ci viene data risposta.



```
WINDOWS 7 [In esecuzione] - Oracle VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto

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

C:\Users\daniel>ping 192.168.32.100

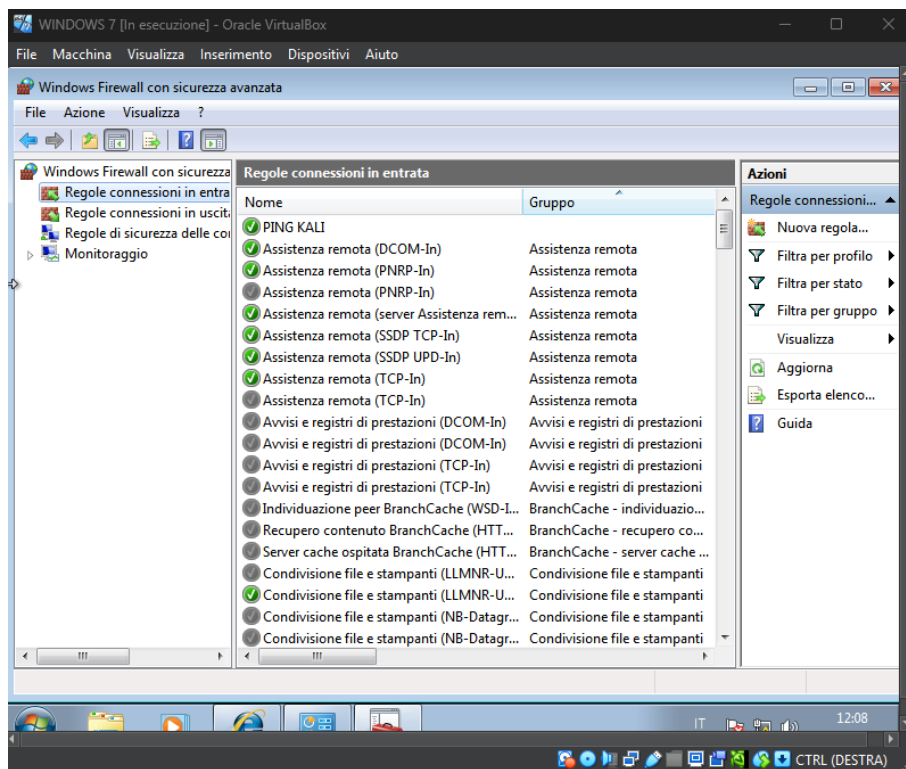
Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

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\daniel>
```

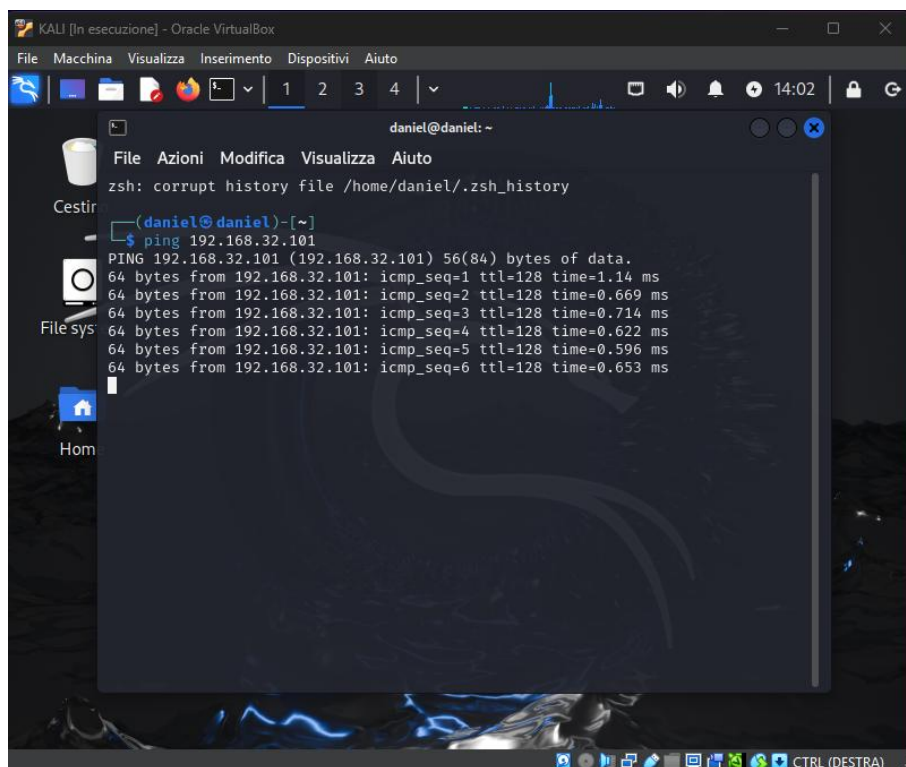
Ora proviamo la comunicazione tra Kali e Windows, in questo caso, non funzionerà, perché il Firewall di Windows bloccherà questa richiesta.

Dobbiamo modificare le regole di Firewall in modo da consentire a Kali (IP 192.168.32.100) di poter comunicare.



Il Firewall non viene disabilitato, creiamo una regola, la quale ci permette di mettere in comunicazione le due macchine, nella slide si può notare che ho creato PING KALI.

In 'regole connessione entrata' aggiungiamo l'indirizzo IP di Kali (192.168.32.100) così da permettere il ping da solo quella macchina.



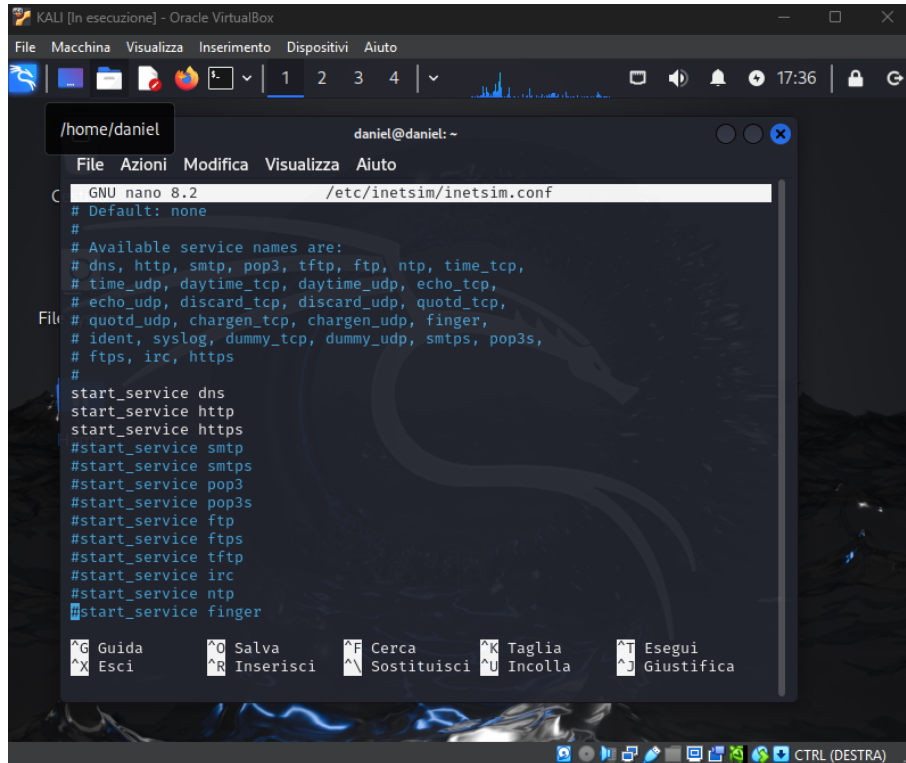
Ripetiamo la procedura di ping e come si può vedere, Kali comunica con windows.

Ora, passiamo ad impostare il DNS ed attivare i protocolli che a noi interessano.

Sempre sulla macchina Kali, digitiamo il codice: `sudo nano /etc/inetsim/inetsim.conf`

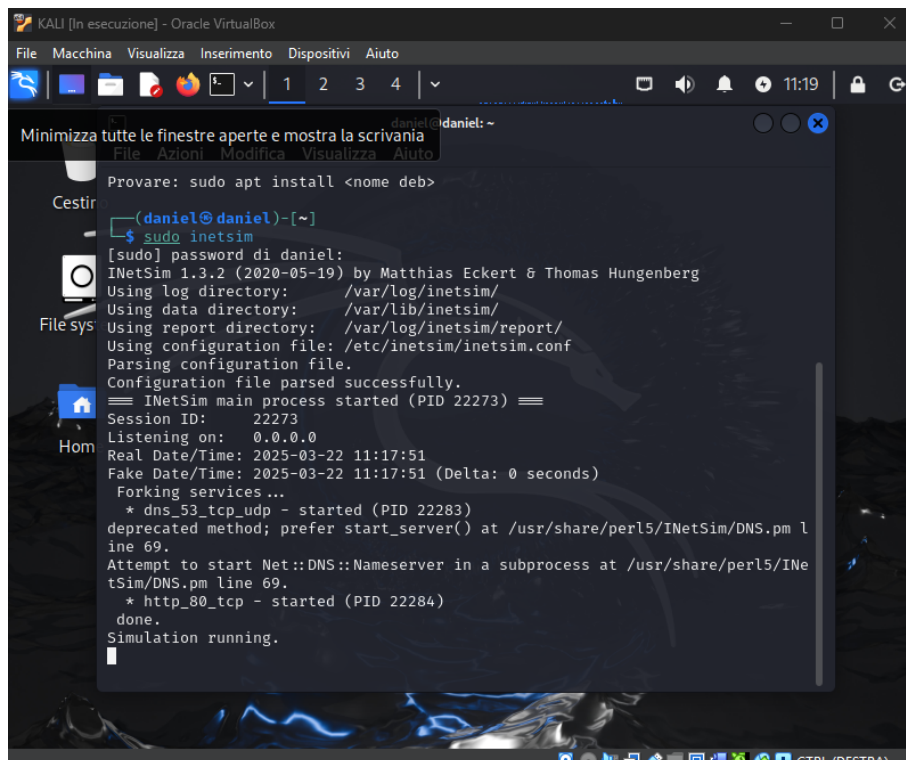
Il quale, ci permette di poter configurare diversi servizi virtuali.

Selezioniamo solo quelli che ci interessano, quindi DNS HTTP e HTTPS togliendo la spunta `#` così facendo attiviamo solo questi.



```
GNU nano 8.2 /etc/inetsim/inetsim.conf
# 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
```

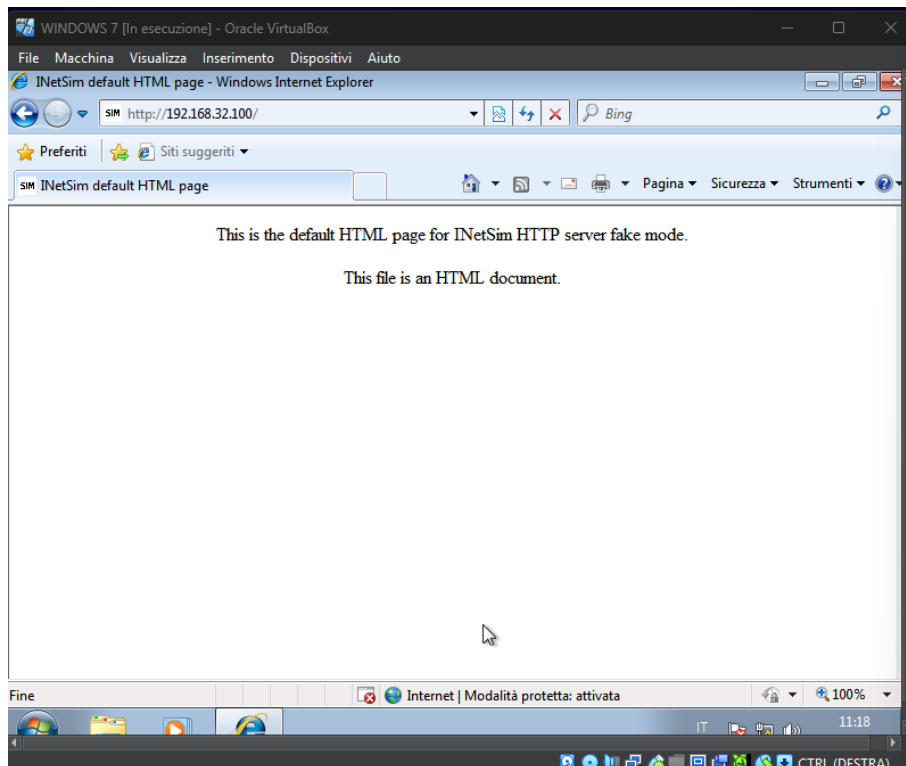
A questo punto, salviamo con CTRL+X poi selezionando Y e chiudiamo. Avviamo Inetsim.



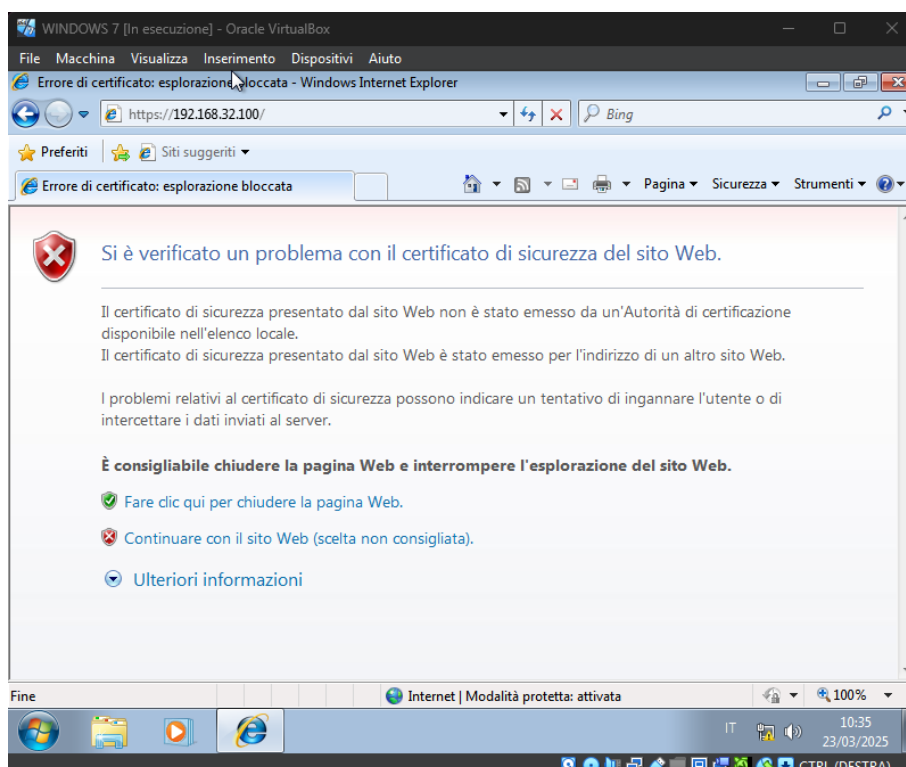
```
(daniel@daniel)-[~]
$ sudo inetsim
[sudo] password di daniel:
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 22273) ==
Session ID: 22273
Listening on: 0.0.0.0
Real Date/Time: 2025-03-22 11:17:51
Fake Date/Time: 2025-03-22 11:17:51 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 22283)
deprecated method; prefer start_server() at /usr/share/perl5/INetSim/DNS.pm l
ine 69.
Attempt to start Net::DNS::Nameserver in a subprocess at /usr/share/perl5/INE
tSim/DNS.pm line 69.
* http_80_tcp - started (PID 22284)
done.
Simulation running.
```

Ora che il programma in simulazione su Kali è attivo, passiamo alla macchina virtuale Windows ed apriamo il browser.

Inserendo l'indirizzo IP 192.168.32.100 il quale richiama la macchina Kali, ci darà come risposta questa pagina web. (Non sono riuscito ad impostare il DNS con epicode.internal)



Ora proviamo con il dominio HTTPS, come potete vedere, il firewall mi blocca la connessione, dicendo che la pagina web non è sicura, ha un certificato non valido e di conseguenza mi chiede se voglio proseguire comunque. Proseguiamo così da avere la connessione e il relativo scambio di dati.



Ora che abbiamo provato con entrambi i protocolli, HTTP e HTTPS apriamo WireShark e controlliamo i pacchetti e le relative differenze.

HTTP

The screenshot shows the Wireshark interface with a capture on the eth0 interface. The packet list shows several packets, with packet 6 selected, which is an HTTP GET request. The packet details pane shows the structure of the HTTP packet, including the Ethernet II header, Internet Protocol Version 4 header, and the User Datagram Protocol header. The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	ISAKMP	378	Unknown
2	0.000024239	192.168.32.100	192.168.32.101	ICMP	406	Destination Unreachable
3	2.964062873	192.168.32.101	192.168.32.100	TCP	66	4916 → 80
4	2.964089697	192.168.32.100	192.168.32.101	TCP	66	80 → 4916
5	2.964715346	192.168.32.101	192.168.32.100	TCP	60	4916 → 80
6	2.965267580	192.168.32.101	192.168.32.100	HTTP	359	GET / HTTP/1.1
7	2.965276020	192.168.32.100	192.168.32.101	TCP	54	80 → 4916
8	2.976855824	192.168.32.100	192.168.32.101	TCP	204	80 → 4916
9	2.978805788	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK
10	2.979226384	192.168.32.101	192.168.32.100	TCP	60	4916 → 80

HTTPS

The screenshot shows the Wireshark interface with a capture on the eth0 interface. The packet list shows several packets, with packet 11 selected, which is an ARP request. The packet details pane shows the structure of the ARP packet, including the Ethernet II header, Internet Protocol Version 4 header, and the Address Resolution Protocol header. The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000013722	PCSSystemtec_b8:94:...	PCSSystemtec_c5:59:...	ARP	42	Request
3	5.139508523	192.168.32.101	192.168.32.100	TCP	66	4916 → 80
4	5.139539007	192.168.32.100	192.168.32.101	TCP	66	80 → 4916
5	5.140885767	192.168.32.101	192.168.32.100	TCP	60	4916 → 80
6	5.140758519	192.168.32.101	192.168.32.100	TLSv1	190	Client Hello
7	5.140769726	192.168.32.100	192.168.32.101	TCP	54	80 → 4916
8	5.178886043	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello
9	5.183977124	192.168.32.101	192.168.32.100	TLSv1	188	Client Hello
10	5.184979536	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec
11	5.190408496	PCSSystemtec_c5:59:...	Broadcast	ARP	60	Who has 192.168.32.101?

Come possiamo notare, le differenze sono nel protocollo, HTTP ha una connessione non crittografata e quindi non sicura, mentre HTTPS è crittografata, lo possiamo notare dalla colonna

del protocollo, la quale ci evidenzia una dicitura TLS (transport layer security) che garantisce l'integrità dei dati.