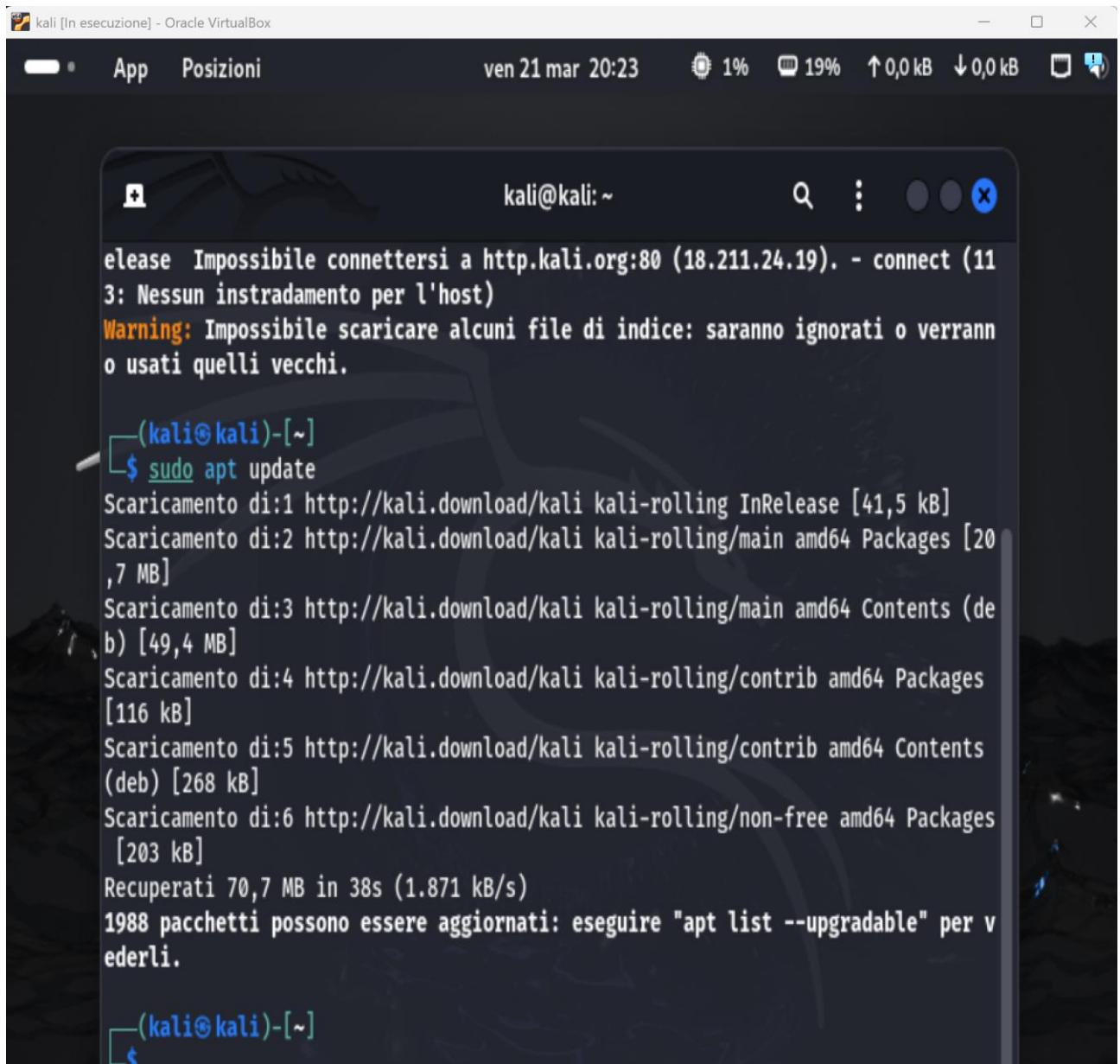


Fase 1: Partenza configurazione dell'ambiente di rete

-impostare la VM di Kali Linux inizialmente con un server HTTPS con indirizzo IP  
**192.168.32.100**

-Impostare windows come Client con il seguente indirizzo IP **192.168.32.101**

-Prima di iniziare assicurarsi di avere un server HTTPS attivo su Kali ed un servizio DNS attivo all'indirizzo 192.168.132.10 per utilizzare come nome di dominio "**epicode.internal**".



The screenshot shows a terminal window titled "kali [In esecuzione] - Oracle VirtualBox". The terminal prompt is "kali@kali:~". The output of the command "sudo apt update" is displayed:

```
kali@kali:~$ sudo apt update
Scaricamento di:1 http://kali.download/kali kali-rolling InRelease [41,5 kB]
Scaricamento di:2 http://kali.download/kali kali-rolling/main amd64 Packages [20,7 MB]
Scaricamento di:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [49,4 MB]
Scaricamento di:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [116 kB]
Scaricamento di:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [268 kB]
Scaricamento di:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [203 kB]
Recuperati 70,7 MB in 38s (1.871 kB/s)
1988 pacchetti possono essere aggiornati: eseguire "apt list --upgradable" per vederli.

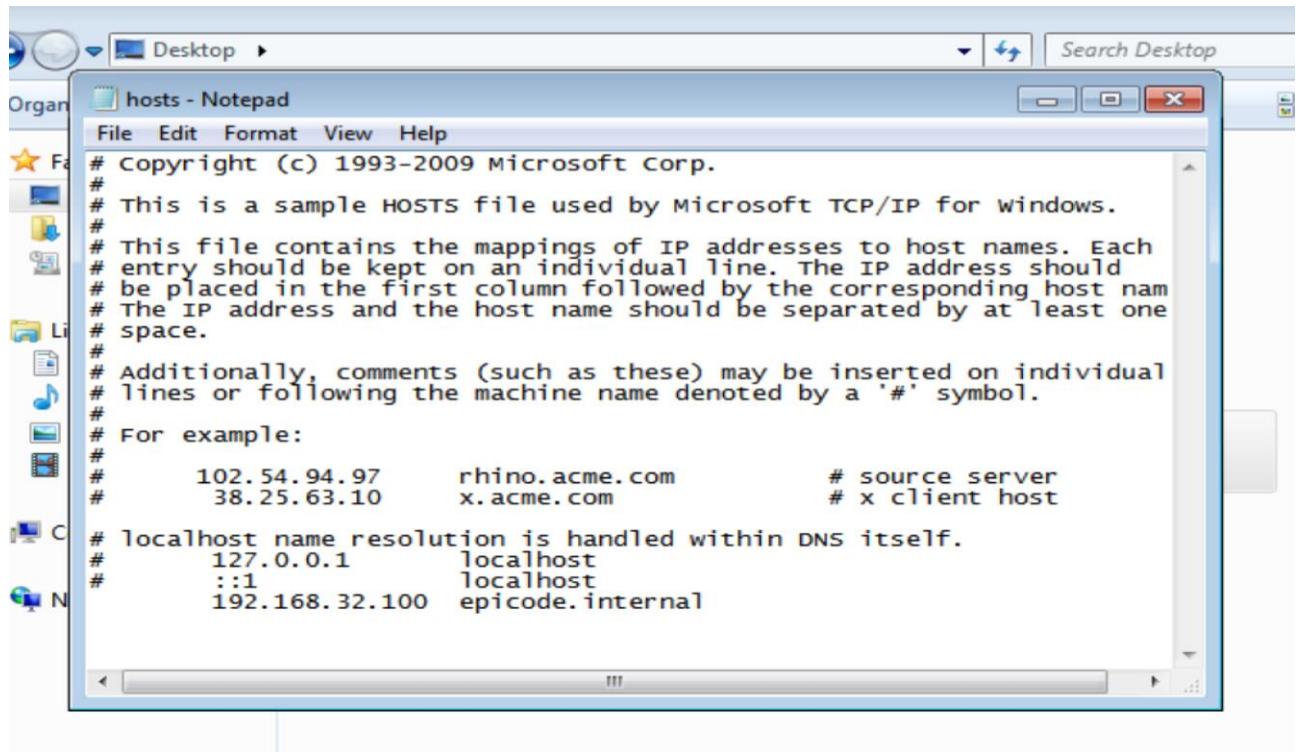
kali@kali:~$
```

Effettuando un sudo apt update per aggiornare la macchina kali

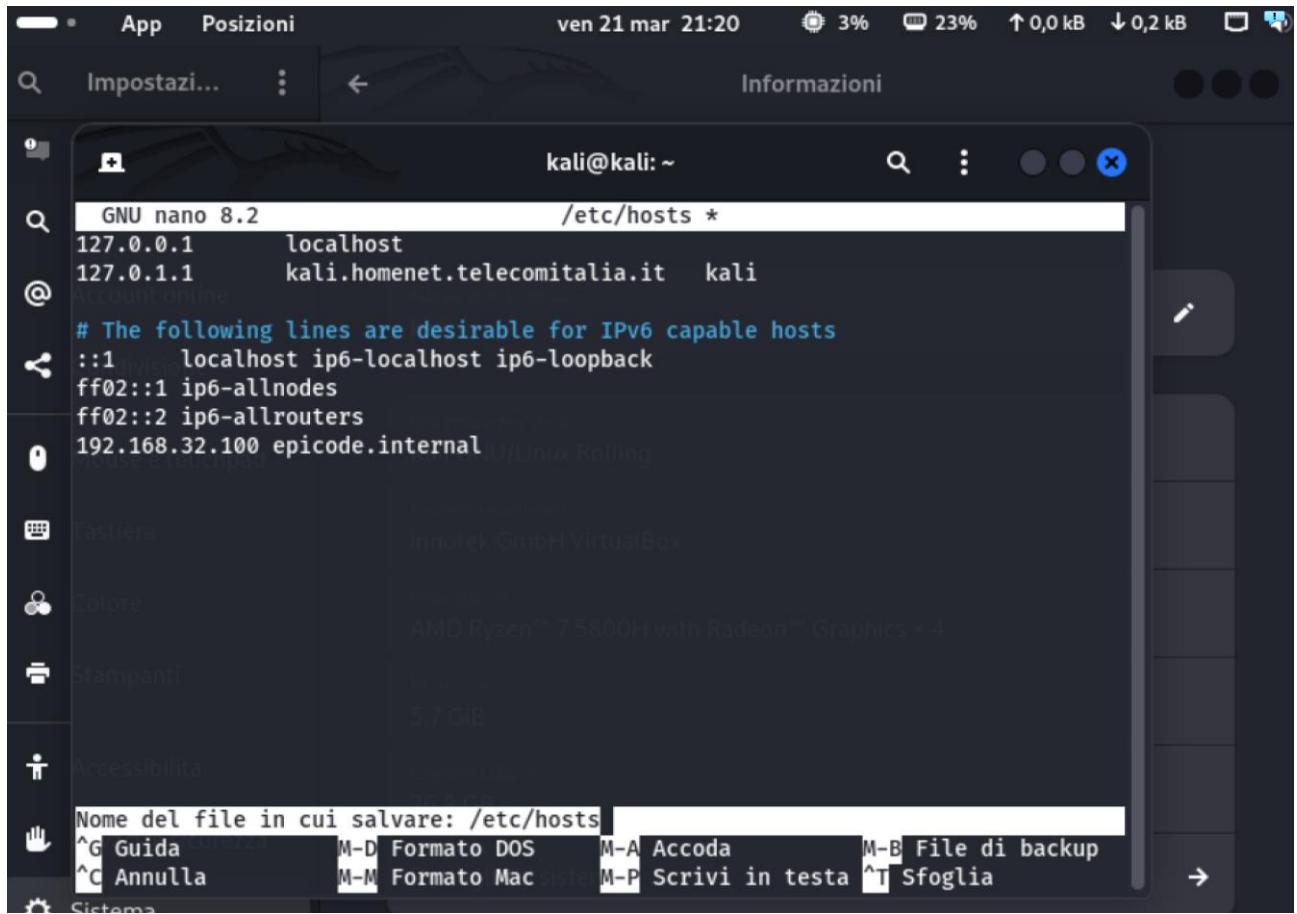
ora procedero con l'impostazione dei Dns su entrambe le macchine

(purtroppo nelle ultime versioni kali le procedure di configurazione dei dns con "InetSim" da problemi di compatibilità e si è ancora in attesa di un update che corregga questo problema che purtroppo abbiamo riscontrato in molti)

ora ci possono essere due modalità operative una che prevede di andare a toccare il codice sorgente di "InetSim" verificarlo e continuare oppure un impostazione manuale all'interno dei file hosts sia di kali che di windows per aggiungere l'ip 192.168.32.100 episode.internal per il proseguimento dell'esercizio. Inoltre aprirò la porta 443 su kali senno potrei avere problemi di ascolto con HTTPS.



```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# 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 episode.internal
```



queste due operazioni manuali mi permettono di far di associare epicode.internals come 192.168.32.100 in Windows e di far riconoscere epicode.internal come 192.168.32.100.

#### Fase 2:Avvio Server analisi witheshrk

Inizio la prima parte dell'esercizio dove viene chiesto di connettermi ad un server HTTPS da Windows.

La Procedura in questo caso prevede che apra il browser su Windows 7 ed inserisca l'indirizzo <https://epicode.internal>.

Così facendo la VM di Windows tenta una comunicazione con il server Kali usando https,in caso di attivita di server e di collegamento andato a buon fine il server mi dovrebbe rispondere.

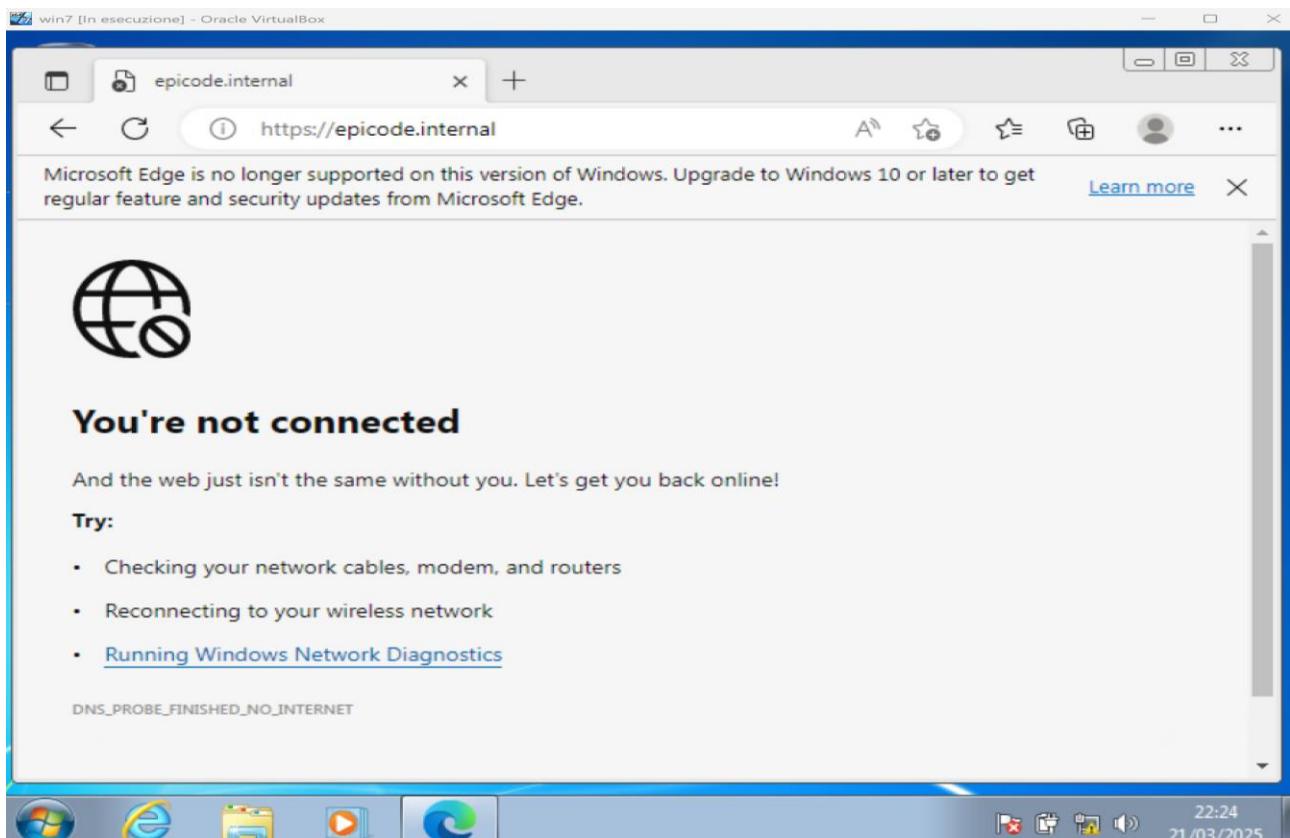
Prima di fare tutto ciò ho ricontrollato tutti i parametri di connessione rete in ipv4.

\*nota il flag di problema di connessione di Windows è dato dal fatto che nella VM di Oracle la connettività sia di Windows che di Kali è impostata su "Rete Interna" il che non permette di navigare su internet(ecco il perchè del flag di errore).

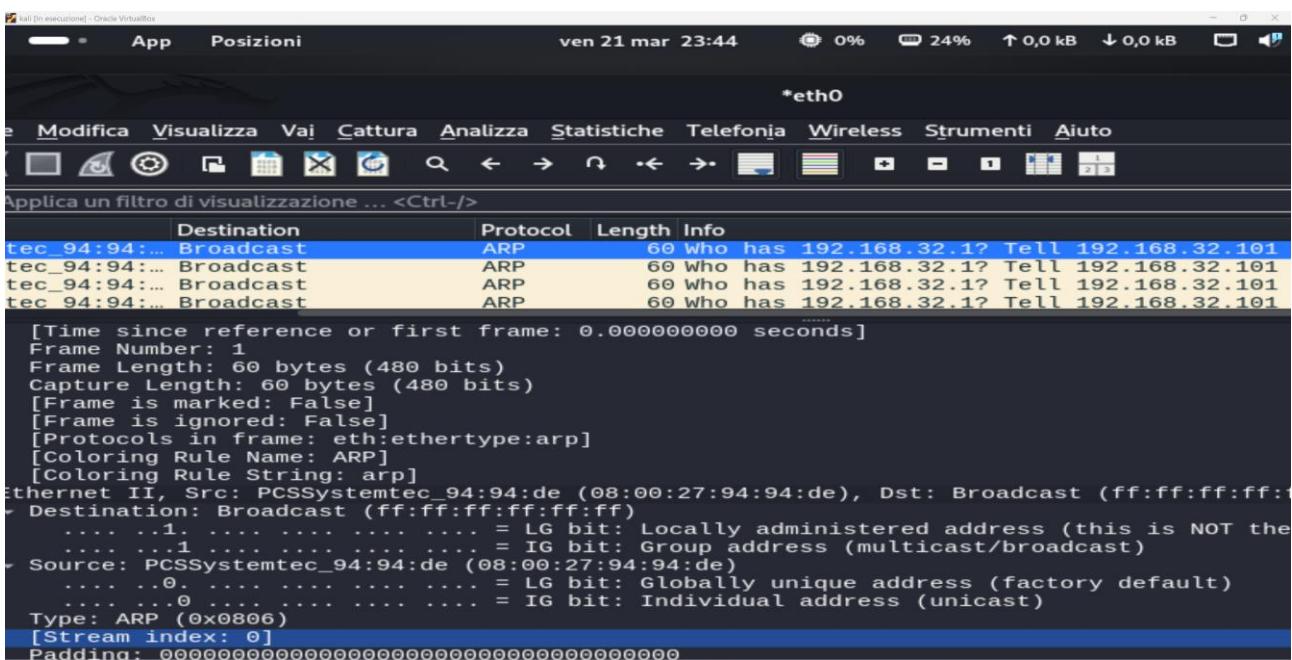
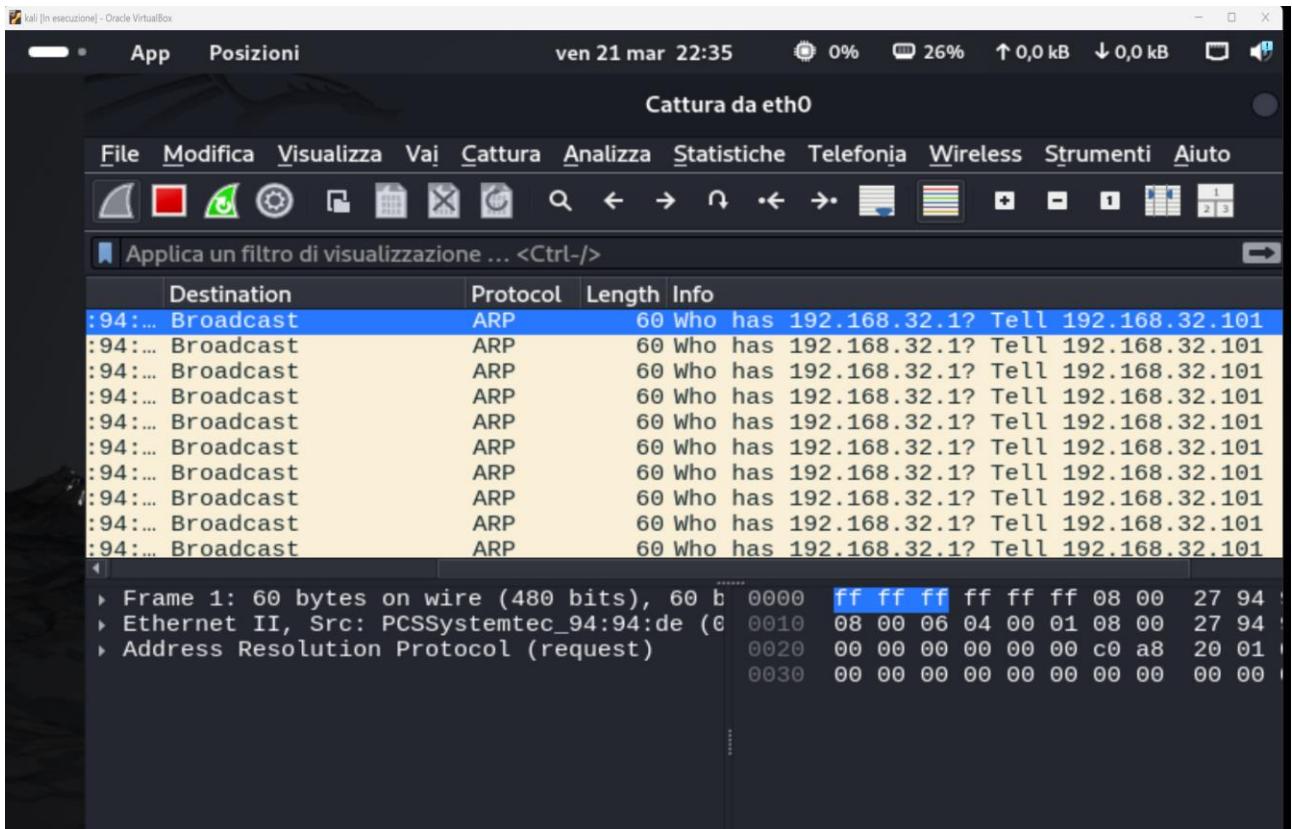
Questo procedimento è consigliato anche per altri tipi di operazioni che richiedano l'isolamento delle VM e di una maggiore sicurezza rimanendo senza accesso alla rete internet che viene richiesto per determinati tipi di test.

Mentre nella VM di Kali ho avviato **Wireshark** per intercettare il traffico di rete.

- Ho selezionato l'interfaccia di rete eth0 che corrisponde alla scheda di rete di Kali.



- Ora eseguendo il comando di invio dell'indirizzo **<https://epicode.internal>** nel browser di Windows Wireshark inizierà la sua cattura come mostrato nella prossima immagine. Questo significa che le due macchine comunicano e possiamo procedere alla nostra analisi evidenziando i mac address di sorgente e destinazione ed il contenuto della richiesta HTTPS come indicato nella prima parte dell'esercizio.



Da un primo lancio emerge solamente un protocollo arp il che significa che le macchine comunicano ma manca la comunicazione https, infatti se lancio un filtraggio "tls" in wireshark non trovo corrispondenze anche se ho analizzato tutte le connessioni.

Premessa la mia versione di virtualbox per problemi di compatibilità amd Vm da diversi problemi ma onestamente ora proverò una procedura diversa un po più complicata ma che dovrebbe risolvere tutto.

# **Esperimento 1 analisi di whiresnark**

Se le mie ipotesi sono giuste il problema di connessione https può essere dato da qualche certificato mancante o qualche malfunzionamento sulla porta 443, ma prima di intervenire come penso voglio impostare la connessione delle due macchine su bridge o nat e vedere se filtrando "tls" (transport layer security riesco ad avere un Server Hello oppure Client Hello in whiresnark andando sulla rete, se così fosse il problema dipende da una delle due configurazioni interne delle vm in ambiente offline.

Per questa prova in bridge userò il dhcp in modo tale da vedere se avrà un indirizzamento automatico.

Spero di trovare dei riscontri nel filtraggio whiresnark sia per TLS e sia per la porta 443

kali [in esecuzione] - Oracle VirtualBox

sab 22 mar 00:02 1% 42% ↑ 0,0 kB ↓ 0,5 kB \*eth0

File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonja Wireless Strumenti Aiuto

tcp.port == 443

No.	Time	Source	Destination	Protocol	Length	Info
1403...	18.432503645	192.168.1.13	2.22.248.171	TCP	60	49211 → 44
1526...	20.108005497	192.168.1.13	192.168.32.100	TCP	66	49214 → 44
1527...	20.129327548	192.168.1.13	192.168.32.100	TCP	66	49215 → 44
1542...	20.384164961	192.168.1.13	192.168.32.100	TCP	66	49216 → 44
1749...	23.106011066	192.168.1.13	192.168.32.100	TCP	66	[TCP Retra
1754...	23.187380974	192.168.1.13	192.168.32.100	TCP	66	[TCP Retra
1782...	23.391755627	192.168.1.13	192.168.32.100	TCP	66	[TCP Retra
2079...	27.375650336	192.168.1.13	172.205.25.163	TCP	60	49195 → 44
2087...	27.428832135	172.205.25.163	192.168.1.13	TCP	66	443 → 4919
2096...	27.543045071	192.168.1.13	4.231.66.184	TCP	60	49198 → 44
2097...	27.575986254	4.231.66.184	192.168.1.13	TCP	66	443 → 4919

Frame 174997: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0  
Ethernet II, Src: PCSSystemtec\_94:94:de (08:00:27:94:94:de), Dst: zte\_d1:1f:4c (c8:98:  
Destination: zte\_d1:1f:4c (c8:98:28:d1:1f:4c)  
App Posizioni sab 22 mar 00:04 0% 43% ↑ 0,0 kB ↓ 0,5 kB

Wireshark · Pacchetto 174990 · eth0

```

Frame 174990: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface eth0
Ethernet II, Src: PCSSystemtec_94:94:de (08:00:27:94:94:de), Dst: zte_d1:1f:4c (c8:98:28:d1:1f:4c)
  Destination: zte_d1:1f:4c (c8:98:28:d1:1f:4c)
    ... .0. .... . . . . . = LG bit: Globally unique address (1)
    ... .0. .... . . . . . = IG bit: Individual address (1)
  Source: PCSSystemtec_94:94:de (08:00:27:94:94:de)
    ... .0. .... . . . . . = LG bit: Globally unique address (1)
    ... .0. .... . . . . . = IG bit: Individual address (1)
  Type: IPv4 (0x0800)
  [Stream index: 8]
Internet Protocol Version 4, Src: 192.168.1.13, Dst: 2.22.248.171
  0100 . . . = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1278
    Identification: 0x141d (5149)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x255b [validation disabled]
    [Header checksum status: Unverified]

```

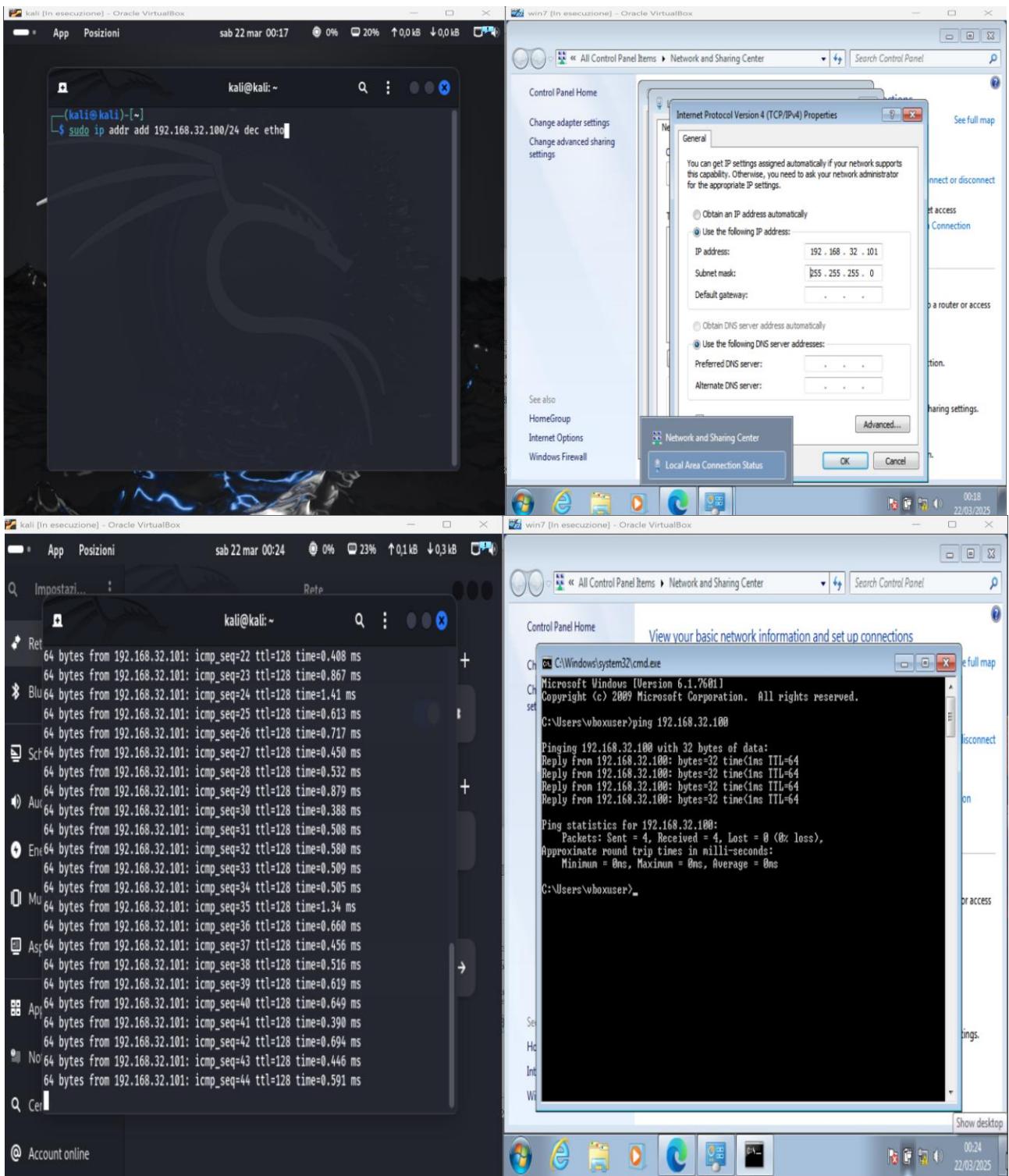
Frame (1292 bytes) Decrypted QUIC (1215 bytes) Reassembled QUIC CRYPTO (29)

No.: 174990 · Time: 23.104559811 · Source: 192.1...ADDING, CRYPTO, CRYPTO, PADDING, CRYPTO, PADDING

✓ Mostra byte del pacchetto Disposizione: Vertical (Stacked)

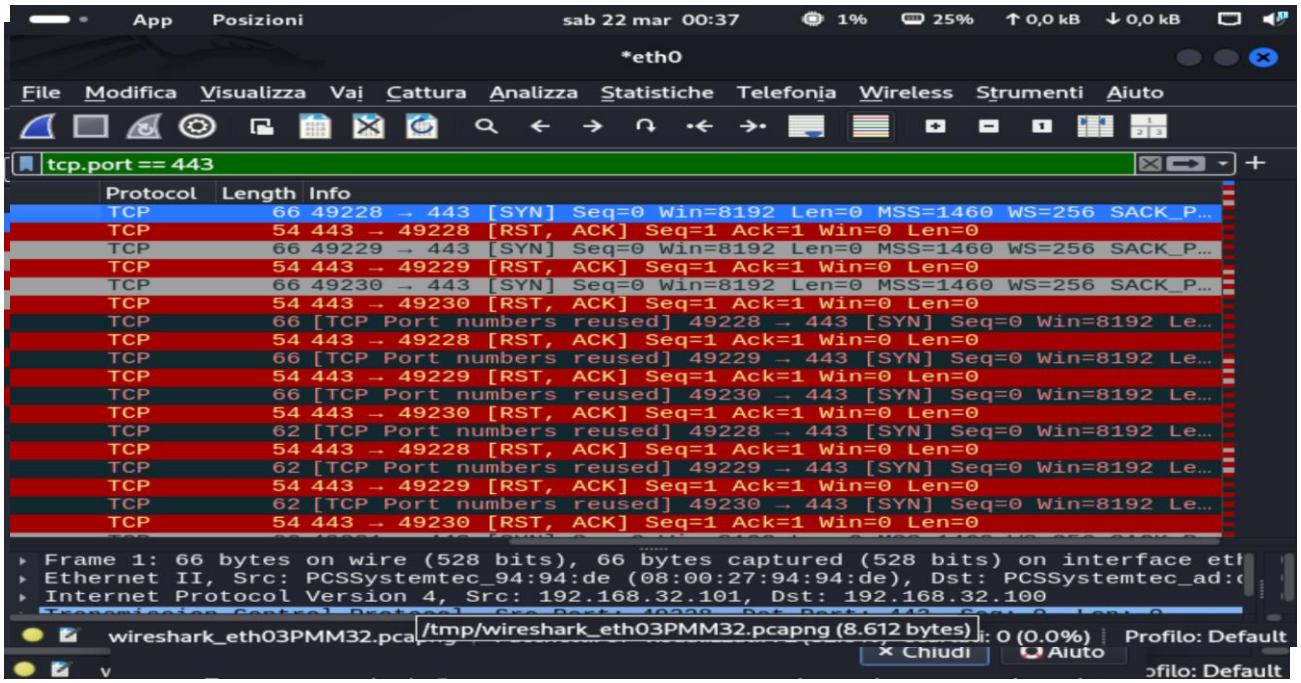
**analisi esperimento 1:** dal test da me condotto sono emersi dei riscontri sia in TLS che in tcp.port == 443 il che vuol che il protocollo https da parte della macchina windows parte(che in questo caso per via del dhcp ha un indirizzo IP 192.168.1.13) ma nel traffico vengono visualizzati diversi marker contenenti messaggi di errore, questo test mi è servito per comprendere se ci fossero problemi in wireshark, ma a quanto pare l'ennesima battaglia tra il sottoscritto la Virtual Machine e i chipset AMD (non proprio compatibili con questo software della Oracle) sta per avere inizio.

## Prima prova di problem solving



ora visto che la compatibilità tra me e la Oracle è minima e sono le ore 00:30 decido di usare le cattive (concedetemi un po' di libertà di narrazione in questo report senno veramente se questo esercizio non riesce entro le 3 di notte costruirò una vera e propria Bare Metal che sono due mesi che non riescono a rilasciare un aggiornamento per risolvere questi problemi e ho dovuto impostare anche i source code per far funzionare la VM della Oracle su un notebook di ultima generazione che è eccezionale in tutto ma purtroppo conosciamo bene il razzismo tra Virtual Machine e AMD).

Tornando a noi ora le macchine grazie alla configurazione ipv4 manuale su entrambe senza immettere DNS poiché stiamo usando host pingano ed è già buon segno.



Ora vediamo se parlano la stessa lingua:

Il client Windows (192.168.32.101) sta inviando pacchetti SYN a Kali (192.168.32.100) sulla porta 443 (ed è già qualcosa) ma purtroppo i server di Kali mi mandano una marea di reset (RST) a questo punto credo che la comunicazione avvenga ma che probabilmente (e non capisco come anche perché a inizio esercizio ho aperto la 443) la porta 443 è chiusa e credo sia questo il motivo per il quale https non mi ascolta...

riapro la porta 443 con i seguenti comandi

```
sudo netstat -tulnp | grep 443 (ma visto che netstat oggi è proprio felice di esistere non funzionerà)
e
sudo ss -tulnp | grep 443
```

```

.101  TCP  54 443 → 49252 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
.100  TCP  66 [TCP Port numbers reused] 49253 → 443 [SYN] Seq=0 Win=8192 Le...
.101  TCP  54 443 → 49253 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
.100  TCP  66 [TCP Port numbers reused] 49254 → 443 [SYN] Seq=0 Win=8192 Le...
.101  TCP  54 443 → 49254 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
.100  TCP  62 [TCP Port numbers reused] 49252 → 443 [SYN] Seq=0 Win=8192 Le...
.101  TCP  54 443 → 49252 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
.100  TCP  62 [TCP Port numbers reused] 49253 → 443 [SYN] Seq=0 Win=8192 Le...
.101  TCP  54 443 → 49253 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
.100  TCP  62 [TCP Port numbers reused] 49254 → 443 [SYN] Seq=0 Win=8192 Le...
.101  TCP  54 443 → 49254 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
.255  NBNS  92 Name query NB WWW.BING.COM<00>
.255  NBNS  92 Name query NB WWW.BING.COM<00>
.255  NBNS  92 Name query NB WWW.BING.COM<00>
ec_ad:de:... ARP  60 Who has 192.168.32.100? Tell 192.168.32.101
ec_94:94:... ARP  42 192.168.32.100 is at 08:00:27:ad:de:26
5.250  SSDP  217 M-SEARCH * HTTP/1.1
5.250  SSDP  217 M-SEARCH * HTTP/1.1
Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface eth0
Ethernet II, Src: PCSSystemtec_94:94:de (08:00:27:94:94:de), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.255
User Datagram Protocol, Src Port: 127, Dst Port: 127

```

eth0: <live capture in progress> | Pacchetti: 43 | Profilo: Default

e niente sempre lo stesso problema adesso ho due soluzioni ma sarò sincero è la prima volta che le utilizzerò ma essendo il mio un caso veramente strano non ho altra scelta, devo avviare un server https su kali ed ho due soluzioni o utilizzare python o apache e li sto iniziando a studiare ora... questa sta diventando una vera e propria questione di principio anche perchè non si capisce perchè non mi apra quella porta sinceramente.

Premessa kali non dovrebbe avere firewall ho controllato anche con sudo ufw status per vedere se kali avesse deciso di installarlo ma non ho risposta dal comando ne se ufw sia installato e attivo o viceversa ed anche con iptables non ci sarebbero problemi.

```
(kali㉿kali)-[~]
└─$ sudo ufw
sudo: ufw: comando non trovato
```

```
(kali㉿kali)-[~]
└─$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source          destination
Chain FORWARD (policy ACCEPT)
target     prot opt source          destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source          destination
```

Ora procedo alla creazione del server sperando che vada tutto bene

```
[kali㉿kali)-[~]
```

```
$ sudo iptables -L
```

```
Chain INPUT (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

```
Chain FORWARD (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

```
Chain OUTPUT (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

```
[kali㉿kali)-[~]
```

```
$ openssl s_server -accept 443 -www
```

```
Could not open file or uri for loading server certificate private key from serve  
r.pem: No such file or directory
```

E niente a quanto pare kali a quanto pare non mi ha installato il certificato SSL per creare la connessione (ore 01:15 di notte e la soluzione di fare fisicamente questo esercizio installando i software su macchina fisica è sempre più vicina) ora faccio pausa sigaretta e cerco una maniera per generare questo file perchè tanto lo so che sto antipatico al drago di kali ma come ben sappiamo i draghi possono essere sconfitti anche se ti sputano fiamme addosso!!! Riuscirà il prode Andrea ad ascoltare la soave voce della principessa HTTPS e a salvarla dal drago Kali?

.....

....

....

ps io lo sapevo che dovevo farmi un full chipset intel ed i prof me l'hanno mandata... apparte gli scherzi sono quasi totalmente sicuro che nei vari problemi che ho avuto questo file non sia stato proprio creato o installato ed ora veramente provo a vedere se vi è la possibilità da qualche parte di ricrearlo senza reinstallare kali tanto sarebbe lo stesso. Sono passate 6 ore da quando sto provando l'esercizio e credo che veramente la Oracle debba fare qualche intervento serio per le macchine amd.

01:15

22/03/2025

Mentre ero fuori a fumare ho incontrato un vecchio signore anziano con la barba lunga e bianca che fumava erba pipa io gli ho chiesto se fosse in ritardo data l'ora e lui mi rispose "Uno Stregone Ssl non è mai in ritardo, arriva esattamente

nel momento in cui deve arrivare" io gli chiedo cosa lui stesse trasportando e lui mi risponde "manuali di SSL ma dovrebbe essere un segreto". A Quel punto lo sventurato Andrea parla della principessa https e del drago allo stregone e lo stregone per aiutarlo gli rivela due incantesimi per aprire la stanza 443 dove è stata imprigionata la principessa.

```
xport OPENSSL_CONF=/etc/ssl/openssl.cnf  
openssl req -x509 -newkey rsa:2048 -keyout server.pem -out server.pem -days 365  
-nodes
```

che di solito funzionava quando il drago kali era ancora giovane  
e l'ultimo rivelatosi fondamentale

```
OPENSSL_CONF=/etc/ssl/openssl.cnf openssl req -x509 -newkey rsa:2048 -keyout server.pem -out server.pem -days 365 -nodes
```

entrambi con il potere di conferire la creazione di certificati SSL per salvare

```
(kali㉿kali)-[~]
$ openssl req -x509 -newkey rsa2048 -keyout server.pem -out server.pem
Error allocating keygen context
40C70381FC7E0000:error:0308010C:digital envelope routines:inner_evp_generic_fetch:unsupported:../crypto/evp/evp_fetch.c:355:Global default library context, Algorithm (rsa2048 : 0), Properties (<null>)
```

## la principessa dal drago

```
-----  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [AU]:IT  
State or Province Name (full name) [Some-State]:Italy  
Locality Name (eg, city) []:Avezzano  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ADR Studio  
Organizational Unit Name (eg, section) []:ADR  
Common Name (e.g. server FQDN or YOUR name) []:ADRRRRR  
Email Address []:adrrecording@gmail.com
```

Finita la breve parentesi fantasy autoironica (ho cercato ovunque delle righe di comando che funzionassero per creare la libreria contenente il certificato).

Tutto questo problema purtroppo è stato dato dai problemi che ci sono tra l'unica virtual machine funzionante sulla mia macchina Oracle VM 7.1.4, e kali versione linux 6.11.2 amd64 che sono le uniche funzionanti ma purtropo come ha dimostrato da infiniti problemi e perdite di file su notebook chipset AMD di ultima generazione.

Ora dovrei essere riuscito a creare il certificato ssl che mi permette di far comunicare in https senza avere piu problemi sulla porta 443 creando il file server.pem contenente sia la chiave privata che il certificato pubblico.

Ora procedo con il lancio del protocollo https://epicode.internal e la visualizzazione della comunicazione attraverso wireshark.  
Ora dovrei avere un server https attivo sulla porta 443 pronto a rispondere vado ad eseguire i test.

Wireshark · Pacchetto 39 · eth0

```
Frame 39: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface
  Section number: 1
  Interface id: 0 (eth0)
  Encapsulation type: Ethernet (1)
  Arrival Time: Mar 22, 2025 02:03:13.596580270 CET
  UTC Arrival Time: Mar 22, 2025 01:03:13.596580270 UTC
  Epoch Arrival Time: 1742605393.596580270
  [Time shift for this packet: 0.000000000 seconds]
  [Time delta from previous captured frame: 0.000601651 seconds]
  [Time delta from previous displayed frame: 0.002240386 seconds]
  [Time since reference or first frame: 1.487834191 seconds]
  Frame Number: 39
  Frame Length: 571 bytes (4568 bits)
  Capture Length: 571 bytes (4568 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp:tls]
  [Coloring Rule Name: TCP]
  [Coloring Rule String: tcp]
Ethernet II, Src: PCSSystemtec_94:94:de (08:00:27:94:94:de), Dst: PCSSystemtec_ad:(...)(factory default)
  Destination: PCSSystemtec_ad:de:26 (08:00:27:ad:de:26)
  .... ..0. .... .... .... = LG bit: Globally unique address (factory default)
No.: 39 · Time: 1.487834191 · Source: 192.168.32.101 · Destination: 192.... · tocol: TLSv1.3 · Length: 571 · Info: Client Hello (SNI=epicode.internal)
```

App Posizioni sab 22 mar 02:06 0% 26% ↑ 0,0 kB ↓ 0,0 kB \*eth0

**File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonia Wireless Strumenti Aiuto**

**tls**

Destination	Protocol	Length	Info
192.168.32.100	TLSv1.3	609	Client Hello (SNI=epicode.internal)
192.168.32.101	TLSv1.3	279	Server Hello, Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	84	Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	571	Client Hello (SNI=epicode.internal)
192.168.32.100	TLSv1.3	571	Client Hello (SNI=epicode.internal)
192.168.32.101	TLSv1.3	1617	Server Hello, Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	84	Change Cipher Spec, Application Data
192.168.32.101	TLSv1.3	1617	Server Hello, Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	84	Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	571	Client Hello (SNI=epicode.internal)
192.168.32.101	TLSv1.3	1617	Server Hello, Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	118	Change Cipher Spec, Application Data
192.168.32.101	TLSv1.3	293	Application Data
192.168.32.100	TLSv1.3	769	Application Data
192.168.32.101	TLSv1.3	293	Application Data
192.168.32.101	TLSv1.3	440	Application Data
192.168.32.101	TLSv1.3	78	Application Data
192.168.32.100	TLSv1.3	609	Client Hello (SNI=epicode.internal)
192.168.32.101	TLSv1.3	279	Server Hello, Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	84	Change Cipher Spec, Application Data
192.168.32.100	TLSv1.3	609	Client Hello (SNI=epicode.internal)

Frame 59: 279 bytes on wire (2232 bits), 279 bytes captured (2232 bits) on interface

- [Conversation completeness: complete, WITH\_DATA (SI)]
- [TCP Segment Len: 517]
- Sequence Number: 1 (relative sequence number)
- Sequence Number (raw): 4293681348
- [Next Sequence Number: 518 (relative sequence number)]
- Acknowledgment Number: 1 (relative ack number)
- Acknowledgment number (raw): 907677210
- 0101 .... = Header Length: 20 bytes (5)
- Flags: 0x018 (PSH, ACK)
- Window: 256
- [Calculated window size: 65536]
- [Window size scaling factor: 256]
- Checksum: 0x37b9 [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- [Timestamps]
- [SEQ/ACK analysis]

No.: 39 - Time: 1.487834191 - Source: 192.168.32.101 - Destination: 192...tocol: TLSv1.3 - Length: 571 - Info: Client Hello (SNI=epicode.internal)

✓ Mostra byte del pacchetto Disposizione: Vertical (Stacked)

Checksum: 0x37b9 [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
[Timestamps]  
[SEQ/ACK analysis]  
TCP payload (517 bytes)

Transport Layer Security

- TLSv1.3 Record Layer: Handshake Protocol: Client Hello
  - Content Type: Handshake (22)
  - Version: TLS 1.0 (0x0301)
  - Length: 512
  - Handshake Protocol: Client Hello

No.: 39 - Time: 1.487834191 - Source: 192.168.32.101 - Destination: 192...tocol: TLSv1.3 - Length: 571 - Info: Client Hello (SNI=epicode.internal)

✓ Mostra byte del pacchetto Disposizione: Vertical (Stacked)

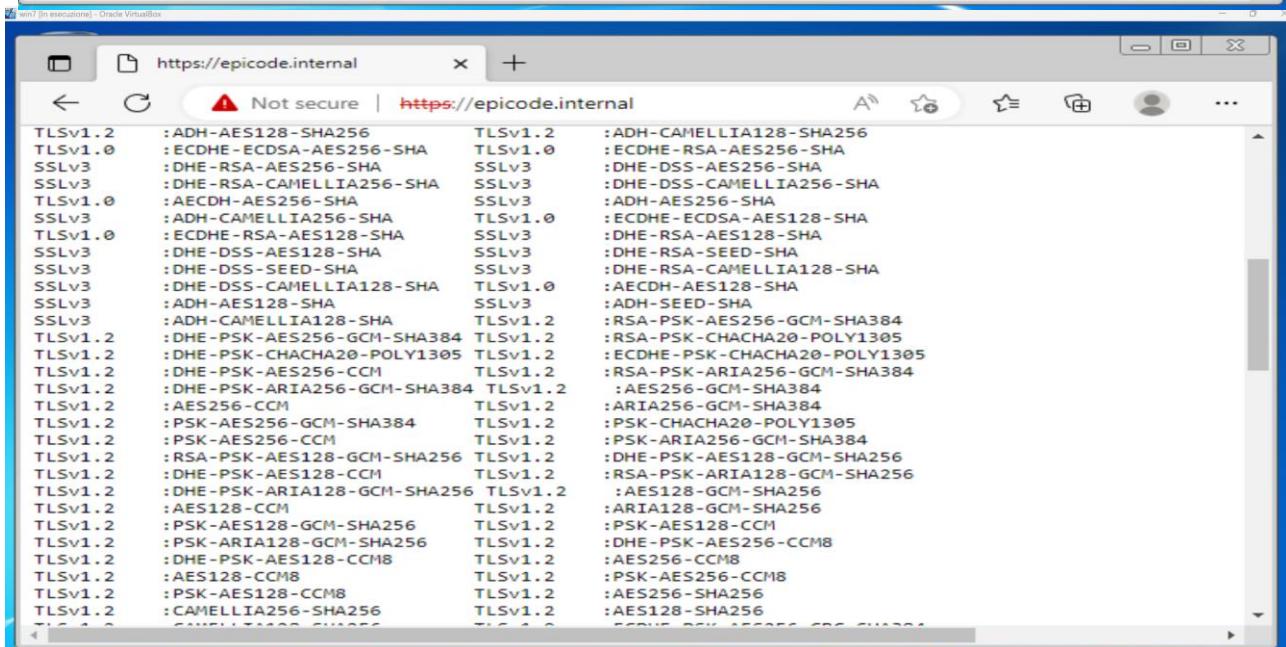
s\_server -accept 443 -cert server.pem -key server.pem -www  
This TLS version forbids renegotiation.  
Ciphers supported in s\_server binary

TLSv1.3	TLSv1.3	TLSv1.3	TLSv1.3
:TLS_AES_256_GCM_SHA384	TLSv1.3	:TLS_CHACHA20_POLY1305_SHA256	
:TLS_AES_128_GCM_SHA256	TLSv1.2	:ECDHE-ECDSA-AES256-GCM-SHA384	
:ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	:DHE-DSS-AES256-GCM-SHA384	
:DHE-RSA-AES256-GCM-SHA384	TLSv1.2	:ECDHE-ECDSA-CHACHA20-POLY1305	
:ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	:DHE-RSA-CHACHA20-POLY1305	
:ECDHE-ECDSA-AES256-CCM	TLSv1.2	:DHE-RSA-AES256-CCM	
:ECDHE-ECDSA-ARIA256-GCM-SHA384	TLSv1.2	:ECDHE-ARIA256-GCM-SHA384	
:DHE-DSS-ARIA256-GCM-SHA384	TLSv1.2	:DHE-RSA-ARIA256-GCM-SHA384	
:ADH-AES256-GCM-SHA384	TLSv1.2	:ECDHE-ECDSA-AES128-GCM-SHA256	
:ECDHE-RSA-AES128-GCM-SHA256	TLSv1.2	:DHE-DSS-AES128-GCM-SHA256	
:DHE-RSA-AES128-GCM-SHA256	TLSv1.2	:ECDHE-ECDSA-AES128-CCM	
:DHE-RSA-AES128-CCM	TLSv1.2	:ECDHE-ECDSA-ARIA128-GCM-SHA256	
:ECDHE-ARIA128-GCM-SHA256	TLSv1.2	:DHE-DSS-ARIA128-GCM-SHA256	
:DHE-RSA-ARIA128-GCM-SHA256	TLSv1.2	:ADH-AES128-GCM-SHA256	
:ECDHE-ECDSA-AES128-GCM-SHA256	TLSv1.2	:ECDHE-ECDSA-AES128-CCM8	
:DHE-RSA-AES128-CCM	TLSv1.2	:DHE-RSA-AES128-CCM8	
:ECDHE-ECDSA-AES256-SHA384	TLSv1.2	:ECDHE-RSA-AES256-SHA384	
:DHE-RSA-AES256-SHA256	TLSv1.2	:DHE-DSS-AES256-SHA256	
:ECDHE-ECDSA-CAMELLIA256-SHA384	TLSv1.2	:ECDHE-RSA-CAMELLIA256-SHA384	
:DHE-RSA-CAMELLIA256-SHA256	TLSv1.2	:DHE-DSS-CAMELLIA256-SHA256	
:ADH-AES256-SHA256	TLSv1.2	:ADH-CAMELLIA256-SHA256	
:ECDHE-ECDSA-AES128-SHA256	TLSv1.2	:ECDHE-RSA-AES128-SHA256	
:DHE-RSA-AES128-SHA256	TLSv1.2	:DHE-DSS-AES128-SHA256	
:ECDHE-ECDSA-CAMELLIA128-SHA256	TLSv1.2	:ECDHE-RSA-CAMELLIA128-SHA256	
:DHF-RSA-CAMELLIA128-SHA256	TLSv1.2	:DHF-DSS-CAMELLIA128-SHA256	

```

TLSv1.2 : CAMELLIA128-SHA256      TLSv1.0      : ECDHE-PSK-AES256-CBC-SHA384
TLSv1.0 : ECDHE-PSK-AES256-CBC-SHA  SSLv3       : SRP-DSS-AES-256-CBC-SHA
SSLv3   : SRP-RSA-AES-256-CBC-SHA  SSLv3       : SRP-AES-256-CBC-SHA
TLSv1.0 : RSA-PSK-AES256-CBC-SHA384 TLSv1.0      : DHE-PSK-AES256-CBC-SHA384
SSLv3   : RSA-PSK-AES256-CBC-SHA  SSLv3       : DHE-PSK-AES256-CBC-SHA
TLSv1.0 : ECDHE-PSK-CAMELLIA256-SHA384 TLSv1.0      : RSA-PSK-CAMELLIA256-SHA384
TLSv1.0 : DHE-PSK-CAMELLIA256-SHA384 SSLv3       : AES256-SHA
SSLv3   : CAMELLIA256-SHA          TLSv1.0      : PSK-AES256-CBC-SHA384
SSLv3   : PSK-AES256-CBC-SHA       TLSv1.0      : PSK-CAMELLIA256-SHA384
TLSv1.0 : ECDHE-PSK-AES128-CBC-SHA256 TLSv1.0      : ECDHE-PSK-AES128-CBC-SHA
SSLv3   : SRP-DSS-AES-128-CBC-SHA  SSLv3       : SRP-RSA-AES-128-CBC-SHA
SSLv3   : SRP-AES-128-CBC-SHA       TLSv1.0      : RSA-PSK-AES128-CBC-SHA256
TLSv1.0 : DHE-PSK-AES128-CBC-SHA256 SSLv3       : RSA-PSK-AES128-CBC-SHA
SSLv3   : DHE-PSK-AES128-CBC-SHA  TLSv1.0      : ECDHE-PSK-CAMELLIA128-SHA256
TLSv1.0 : RSA-PSK-CAMELLIA128-SHA256 TLSv1.0      : DHE-PSK-CAMELLIA128-SHA256
SSLv3   : AES128-SHA              SSLv3       : SEED-SHA
SSLv3   : CAMELLIA128-SHA          TLSv1.0      : PSK-AES128-CBC-SHA256
SSLv3   : PSK-AES128-CBC-SHA       TLSv1.0      : PSK-CAMELLIA128-SHA256
---
Ciphers common between both SSL end points:
TLS_AES_128_GCM_SHA256    TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256 ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256
ECDHE-ECDSA-AES256-GCM-SHA384 ECDHE-RSA-AES256-GCM-SHA384 ECDHE-ECDSA-CHACHA20-POLY1305
ECDHE-RSA-CHACHA20-POLY1305 ECDHE-RSA-AES128-SHA     ECDHE-RSA-AES256-SHA
AES128-GCM-SHA256          AES256-GCM-SHA384        AES128-SHA
AES256-SHA
Signature Algorithms: ECDSA+SHA256:RSA-PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA-PSS+SHA384:RSA+SHA384:RSA-F
Shared Signature Algorithms: ECDSA+SHA256:RSA-PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA-PSS+SHA384:RSA+SHA384
Supported groups: :x25519:secp256r1:secp384r1

```



```

New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
SSL-Session:
Protocol : TLSv1.3
Cipher   : TLS_AES_128_GCM_SHA256
Session-ID: BDE30C8045ED8526FA358DB6BB32EDCBE6D505EE05BC6569128847B34BA876BB
Session-ID-ctx: 01000000
Resumption PSK: F447BE6A6FDA59C624E0A2EF852F8B172CAEF5346F18B975298CF23B19485C29
PSK identity: None
PSK identity hint: None
SRP username: None
Start Time: 1742605393
Timeout   : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
-
0 items in the session cache
0 client connects (SSL_connect())
0 client renegotiates (SSL_connect())
0 client connects that finished
24 server accepts (SSL_accept())
0 server renegotiates (SSL_accept())
7 server accepts that finished
9 session cache hits
0 session cache misses
0 session cache timeouts
0 callback cache hits
0 cache full overflows (128 allowed)

```

Alla fine il Paladino Andrea assieme al Mago SSL sono riusciti a salvare la principessa HTTPS dal Drago Kali che l'aveva rinchiusa nel castello di Linux precisamente nella segreta 443 buttando la chiave SSL nel fossato del non recovery, ma L'intervento del mago SSL ha fatto sì che questa chiave venisse rigenerata assieme a tutto il mazzo di chiavi delle porte del castello di linux e finalmente siamo riusciti ad ascoltare la soave voce della principessa HTTPS.  
E vissero per sempre felici ed in attesa di patch!!!

ok apparte tutti gli scherzi ed un po di fantasy (mi metto nei panni dei professori a leggere 20 e più pagine di report) ma finalmente credo di aver sistemato il problema con la VM di Kali non so come sia possibile che l'installazione avesse perso il file server.pem e la sua chiave, addirittura neanche il comando che permettesse di utilizzare la configurazione classica di OpenSSL ha funzionato ed è grave, ho dovuto proprio ricostruire tutto e dopo il fallimento del primo comando mi ha salvato un forum dove per fortuna un user ha avuto il mio stesso problema.

**Comando kali: OPENSSL CONF=/etc/ssl/openssl.cnf openssl req -x509 -newkey rsa:2048 -keyout server.pem -out server.pem -days 365 -nodes**

Credo di non aver avuto altre alternative anche perchè tanti dei comandi di questa stringa li devo ancora capire ma credo di aver creato una chiave ed il file server con una scadenza annuale ed ho dovuto inserire anche i miei dati come riportato nelle immagini precedenti.

Una cosa particolare che ho notato è stata che appena creato questo file subito nella finestra del browser di Windows mi ha dato l'avviso di sicurezza che tanto aspettavo e al suo interno tutti i dati di comunicazione.

Ora come si può notare dalla immagine ho una comunicazione ottimale tra le due macchine e finalmente sono finiti i messaggi di reset e sono apparsi i vari "Hello" che indicano una corretta comunicazione dei pacchetti attraverso le porte.

Procedo con la parte conclusiva del primo esercizio

## Conclusione Esercizio Parte 1 HTTPS

nei pacchetti intercettati come si può vedere dalle ultime immagini  
ho trovato i seguenti mac address:

MAC address di sorgente Windows: 08:00:27:94:94:de

MAC address di destinazione Kali server: 08:00:27:ad:de:26

- da notare che utilizzando l'HTTPS il contenuto della richiesta è cifrato proprio grazie all'utilizzo dello stesso e Wireshark stesso non è in grado di leggere queste cifrature.

Riassumendo una comunicazione in HTTPS è avvenuta con successo dopo tante difficoltà dovute alla mancanza del file server e della chiave SSL, sono riuscito ad identificare entrambi i Mac sia di sorgente che di destinazione ma soprattutto ho notato e credo che questo sia lo scopo della prima parte dell'esercizio come tutto il contenuto della comunicazione sia protetto con una cifratura TLS (Transport Layer Security) rendendo i dati impossibili da leggere ma allo stesso tempo sono verificati dal server.

Indirizzi MAC Sorgente-Destinazione

```
Ethernet II, Src: PCSSystemtec_94:94:de (08:00:27:94:94:de),  
  Destination: PCSSystemtec_ad:de:26 (08:00:27:ad:de:26)
```

contenuto della richiesta https

```
Transport Layer Security  
  TLSv1.3 Record Layer: Handshake Protocol: Client Hello  
    Content Type: Handshake (22)  
    Version: TLS 1.0 (0x0301)  
    Length: 512  
      Handshake Protocol: Client Hello  
  
0000  08 00 27 ad de 26 08 00  27 94 94 de 08 00 45 00  ...'...&...  .... E.  
Io.: 39 · Time: 1.487834191 · Source: 192.168.32.101 · Destination: 192....tocol: TLSv1.3 · Length: 571 · Info: Client Hello (SNI=epicode.internal)
```

**ESERCIZIO PARTE 2 RIPETERE LA STESSA PROCEDURA SOSTITUENDO HTTPS CON HTTP, EVIDENZIARNE LE DIFFERENZE ED INFINE SPIEGAZIONE DELLE PRINCIPALI DIFFERENZE SE PRESENTI:**

*Per iniziare visti i problemi riscontrati nella prima parte dell'esercizio parto subito con una programmazione nel terminale Kali dove apro un server HTTP sulla porta 80 .*

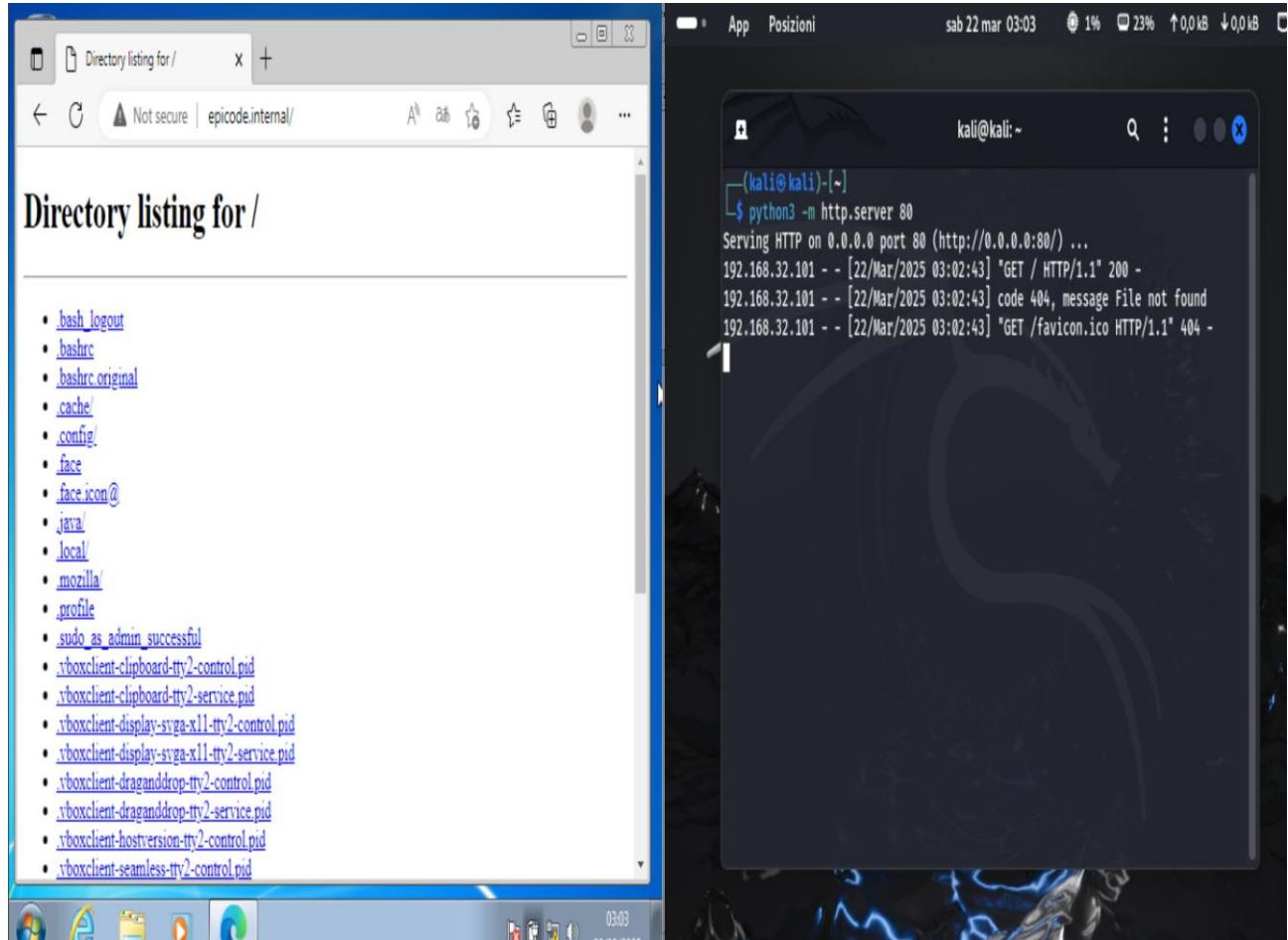
userò questo tipo di comando python che ho trovato nelle

precedenti ricerche per risolvere il problema del server poiché non vorrei che utilizzando il metodo manuale (che è molto simile al metodo dell'host di prima) mi dia dei problemi...

il comando è il seguente:

```
python3 -m http.server 80
```

dove nel terminale indico prima il linguaggio di scrittura poi il modulo ad esso collegato con relativa esecuzione(in questo caso http) ed in fine il numero della porta.



Ora Procedo ad inviare il comando da browser windows e ad intercettare i pacchetti, dopodichè filtrerò gli http e li analizzerò come richiesto.

```
  * [Conversation completeness: Complete, WITH_DATA (31)]
    ..0. .... = RST: Absent
    ...1 .... = FIN: Present
    .... 1... = Data: Present
    .... .1.. = ACK: Present
    .... ..1. = SYN-ACK: Present
    .... ..1 = SYN: Present
    [Completeness Flags: ·FDASS]
    [TCP Segment Len: 463]
    Sequence Number: 1      (relative sequence number)
    Sequence Number (raw): 2813239273
    [Next Sequence Number: 464      (relative sequence number)]
    Acknowledgment Number: 1      (relative ack number)
    Acknowledgment number (raw): 3801873160
    0101 .... = Header Length: 20 bytes (5)
  * Flags: 0x018 (PSH, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ..1. .... = Acknowledgment: Set
    .... ..1. .... = Push: Set
0000  08 00 27 ad de 26 08 00 27 94 94 de 08 00 45 00  ····&... '··· E·

No.: 22 · Time: 15.142980798 · Source: 192.168.32.101 · Destination: 192.168.32.100 · Protocol: HTTP · Length: 517 · Info: GET /HTTP/1.1
  * Frame 22: 517 bytes on wire (4136 bits), 517 bytes captured (4136 bits) on interface
    Section number: 1
  * Interface id: 0 (eth0)
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 22, 2025 03:04:16.358788534 CET
    UTC Arrival Time: Mar 22, 2025 02:04:16.358788534 UTC
    Epoch Arrival Time: 1742609056.358788534
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.007199174 seconds]
    Wireshark · Pacchetto 22 · eth0

    Type: IPv4 (0x0800)
    [Stream index: 3]
  * Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  * Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 503
    Identification: 0x1aab (6827)
  * 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x1c3c [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.32.101
    Destination Address: 192.168.32.100
    [Stream index: 3]
  * Transmission Control Protocol, Src Port: 49287, Dst Port: 80, Seq: 1, Ack: 1, Len: 517
    Source Port: 49287
    Destination Port: 80
    [Stream index: 0]
    [Stream Packet Number: 4]
    [Conversation completeness: Complete, WITH_DATA (31)]
    0000  08 00 27 ad de 26 08 00 27 94 94 de 08 00 45 00  ····&... '··· E·

No.: 22 · Time: 15.142980798 · Source: 192.168.32.101 · Destination: 192.168.32.100 · Protocol: HTTP · Length: 517 · Info: GET /HTTP/1.1
```

**Nella prima immagine superiore si può vedere** la richiesta **HTTP GET** dal client 192.168.32.101 al server 192.168.32.100 e questo conferma che la connessione in HTTP ha dato esito positivo(e decisamente in modo molto più rapido rispetto all'HTTPS grazie anche al lavoro che ho dovuto fare per ricostruire il file server prima).

Si può notare anche come nella risposta del server sia scritto HTTP/1.0 200 OK che è la prova essenziale che il server ha risposto ottimamente all'HTML.

Nella stessa posizione di dove ho trovato i MAC per HTTPS ci sono anche i mac per HTTP:

MAC sorgente: **08:00:27:94:94:de** MAC destinazione: **08:00:27:ad:de:26**

```
Window: 256
[Calculated window size: 65536]
[Window size scaling factor: 256]
Checksum: 0xb3e0 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]
TCP payload (463 bytes)
- Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: epicode.internal\r\n
    Connection: keep-alive\r\n
    Cache-Control: max-age=0\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/*\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
\r\n
  [Response in frame: 26]
  [Full request URI: http://epicode.internal/]
```

la stessa cosa vale per gli IP:

**IP sorgente: 192.168.32.101 Windows -IP destinazione: 192.168.32.100 Kali**

ed infine anche il contenuto http è ben visibile

## Conclusioni Finali

In queste due esercitazione di intercettazione di traffico HTTPS e HTTP generate dall'VM Windows ed analizzate dalla VM Kali con Wireshark noto le relative differenze tra le due.

In alcune parti ho un po romanizzato il report e chiedo scusa se può sembrare di poco gusto essendo un esame ma onestamente l'ho fatto un pò per sdrammatizzare e un po' per far sorridere i professori nella lettura di un report di 20 pagine poichè mi sono ritrovato ad affrontare un gravissimo errore di sistema ed onestamente stavo quasi pensando di lasciar stare e chiudere l'esercizio con un errore quasi irreversibile da correggere per le mie capacità di programmazione linux ed ssl attuali.

Tornando all'esercizio attraverso l'analisi dei comportamenti dell'HTTPS e HTTP analizzati con Wireshark si possono notare tante similitudini come ad esempio dove trovare MAC

**address,IP e contenuti delle ricerche ma le differenze che ri guardano questi due protocolli sono molteplici proverò a fare uno schema qui sotto per spiegare meglio il tutto:**

	HTTPS	HTTP
protezione	Utilizza end-to-end quindi alta	niente
cifrature	SI attraverso tls	no
Visualizzazione su wireshark	Solamente i vari Hello meglio conosciuti come handshake	È visibile tutto quanto
Porta di comunicazione	443	80 o 8080
Per che cosa utilizzarli	Per tutti quei siti dove ci sono e vengono richiesti dati sensibili e personali grazie ai protocolli e all'architettura più sicura	Siti dove non sono presenti dati sensibili,test e siti più semplici dove non ci sono dati di rilievo ricordiamoci che questo protocollo non ha protezioni
Velocità teorica di analisi	Meno veloce per via delle cifrature	Più veloce per la mancanza di sicurezza e cifrature da decifrare
Lucchetto browser	si	no
Certificato digitale	si	no
Difficoltà di configurazione	bassa	Alta

**Un altra piccola conclusione of topic visto quello che è successo ed ho riportato in questo esercizio:Ho 38 anni e lavoro con i computer sin da quando ero bambino ed onestamente mai come oggi in un esame di Cybersecurity mi sono reso conto dell'utilità di forum piattaforme ed ebook ed AI dove trovare soluzioni che ti salvassero la vita soprattutto durante un esame.**

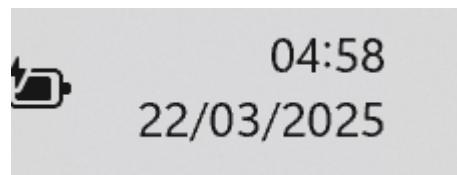
So che alcuni potrebbero pensare “hai usato stringhe che non conosci copiandole su internet”,ma la verità è che senza una ricerca web accurata e vari tentativi(ho iniziato il test alle 7 e mezza di oggi pomeriggio ed ora sono le 4 di notte quindi fate voi)ho dovuto rimediare ad un problema enorme con un sistema operativo basato su linux dove non inserivo un “sudo” da almeno 10 anni,per risolvere questo problema...25 anni fa avrei spento il computer e forse continuato a provarci quando ti sarebbero arrivati dopo un mese dei libri di programmazione SSI e linux dall'americana in lingua originale.

**E' da inizio corso che ho problemi di compatibilità tra Oracle e Kali (ed il prof mi ha sempre aiutato ma questa configurazione ha veramente vita propria) eppure il mio MSI alpha 15 va alla grande ci gestisco concerti da migliaia di persone ci uso programmi di audio editing delicatissimi e che ti portano all'estremo la macchina.**

**Ma oggi mi sono reso conto soprattutto di una cosa...noi parliamo di informatica moderna studiamo cybersecurity e siamo tutti un po' smanettoni ma la verità è che se di fronte ad un problema enorme che come in questo caso rischiava di farmi bocciare all'esame per mancata consegna...se non utilizzi tutti i mezzi che hai a disposizione fallisci!ho usato comandi che non ho mai usato ma prima di immetterli ho studiato ogni singolo carattere perchè la macchina come riportato nella prima parte del report non aveva proprio il file base per svolgere l'esercizio ma alla fine sarà anche grazie alla mia cocciutaggine Abruzzese/Informatica sono riuscito a risolvere il problema e anche se l'esercizio è importantissimo sono strafelice di essere riuscito a risolvere un problema di software che mi sembrava impossibile da sistemare**

e aldilà del voto che prenderò in questo esame sono contento perchè sono riuscito in un qualcosa che ritenevo impossibile per me questa è la vera informatica.

Ps personalmente credo che se non esca la famosa patch kali tra almeno 2 settimane per il programma che abbiamo io credo di buildarmi proprio un VM ba UBUNTU son i software della Kali dopo l'esperienza di questa notte.



A  
n  
d  
r  
e  
a

D  
i

R  
o  
c  
c  
o

e  
s  
a  
m  
e

d  
i

p  
r  
a  
t  
i  
c  
a

M  
a  
r  
z  
o

2  
0  
2  
5