

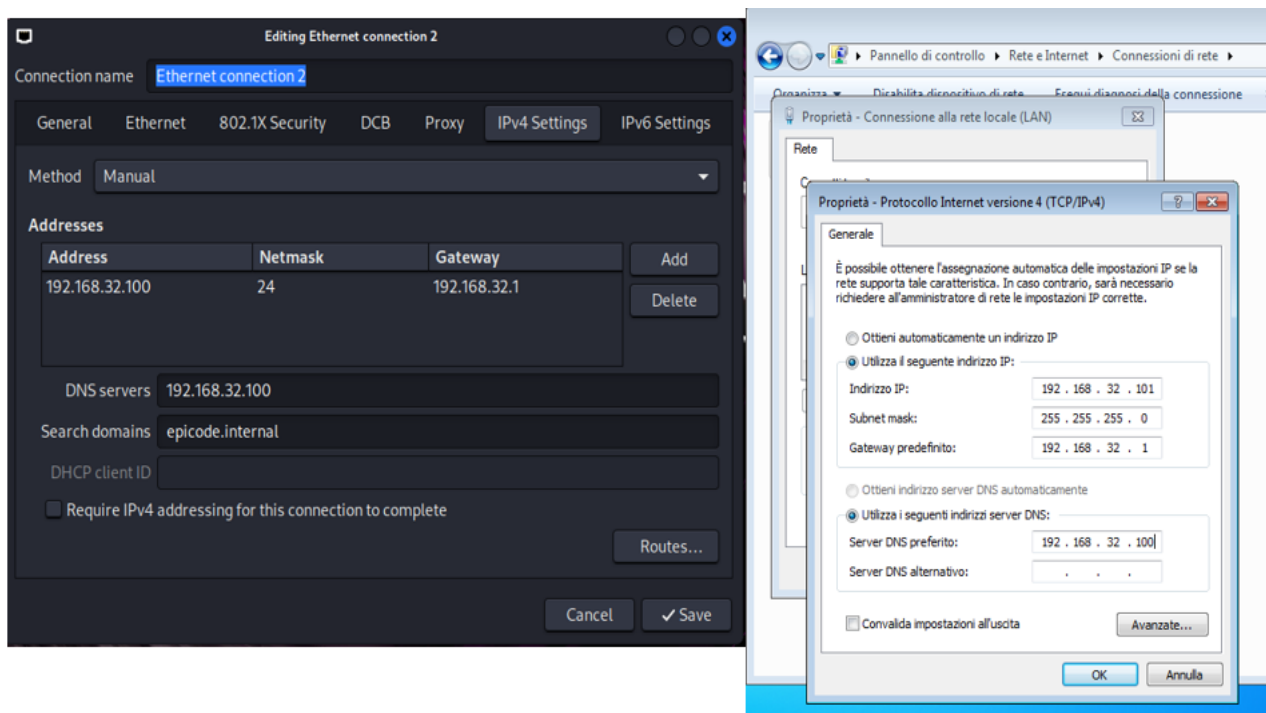
# Configurazione di una rete Client-Server e analisi dei pacchetti

Questo esercizio di simulazione di un'architettura client-server aiuta a comprendere fino in fondo le dinamiche delle reti, esplorando direttamente le interazioni tra client e server all'interno della rete, in questo caso creata per l'occasione, e mostrando l'importanza della sicurezza nelle trasmissioni web.

Gli obiettivi principali da porsi sono: simulare un'architettura client-server in un ambiente controllato, usare Wireshark per catturare e analizzare il traffico di rete tra un client (Windows 7) e un server (Kali) e, in conclusione, andare a confrontare sia il tipo di sicurezza e sia la composizione del traffico delle connessioni HTTPS e HTTP, andando ad evidenziarne le differenze.

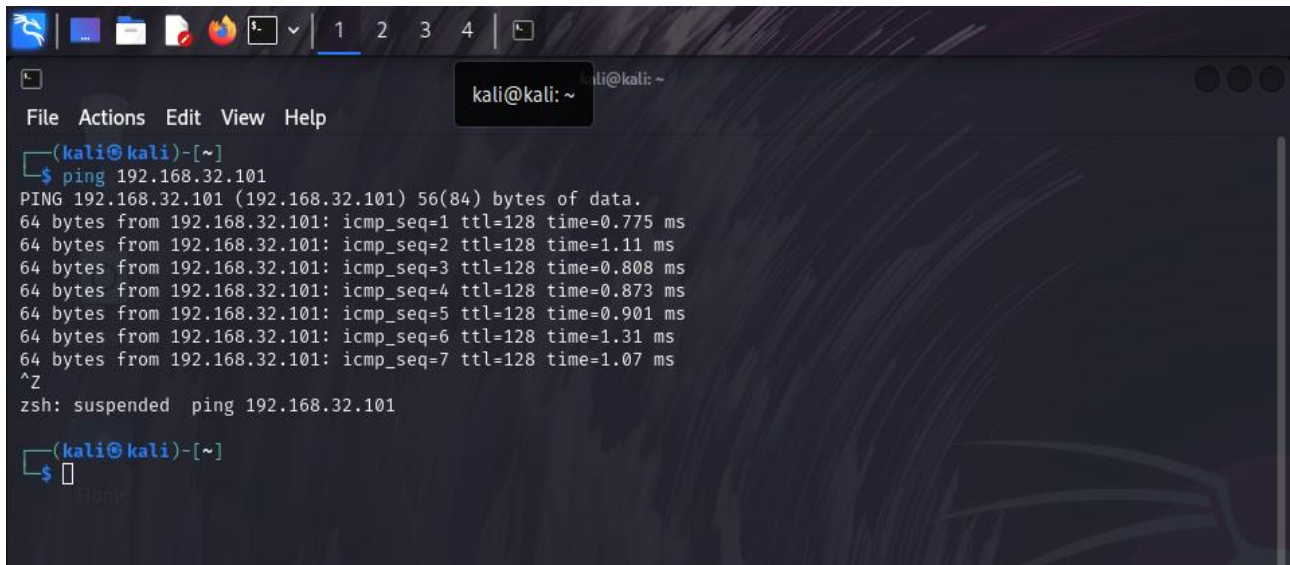
## PRIMO STEP: Configurazione dell'ambiente virtuale

L'esercizio richiedeva di impostare, in entrambe le macchine virtuali, un indirizzo IP statico. Gli indirizzi da inserire erano: Kali Linux IP 192.168.32.100, Windows 7 IP 192.168.32.101. Per comodità ho preferito creare ex novo una connessione sulla macchina Kali ed impostare l'indirizzo IP statico su Windows.



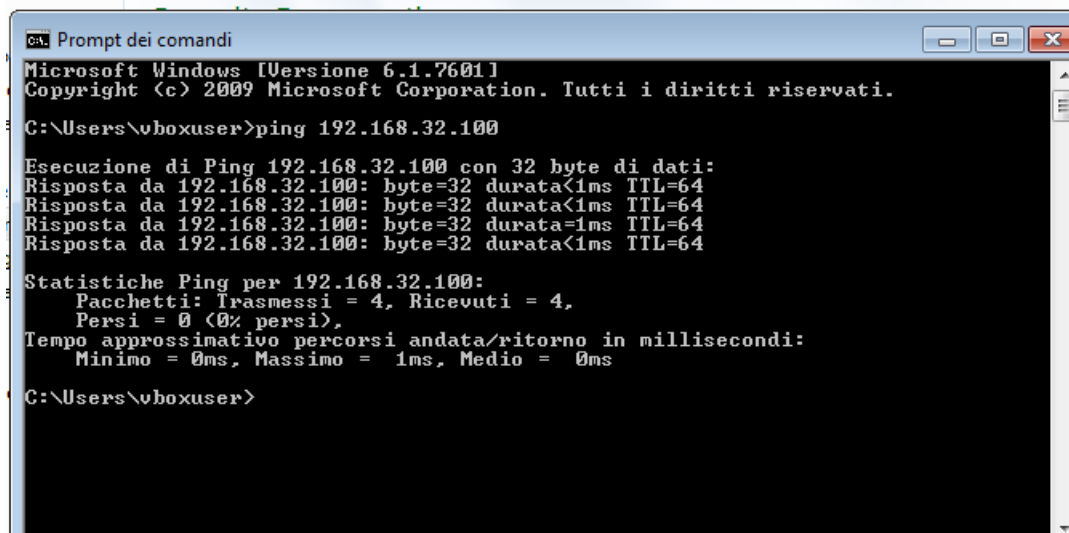
Mi assicuro anche che entrambe le macchine siano impostate su rete interna.

Adesso, dopo aver impostato entrambi gli indirizzi IP, verifico che comunichino tra di loro facendo il ping dalla macchina Kali a Windows e viceversa.



```
(kali@kali)-[~]
$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=0.775 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=1.11 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=0.808 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=0.873 ms
64 bytes from 192.168.32.101: icmp_seq=5 ttl=128 time=0.901 ms
64 bytes from 192.168.32.101: icmp_seq=6 ttl=128 time=1.31 ms
64 bytes from 192.168.32.101: icmp_seq=7 ttl=128 time=1.07 ms
^Z
zsh: suspended ping 192.168.32.101

(kali@kali)-[~]
$
```



```
CA Prompt dei comandi
Microsoft Windows [Versione 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\ vboxuser> 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\ vboxuser>
```

## SECONDO STEP: Configurazione di un DNS server su Kali

Decido di installare come DNS server dnsmasq, in quanto è adatto per ambienti di testing o comunque ad ambienti che dispongono di limitate risorse di sistema. È immediato da configurare e inoltre è adatto a network piccoli, come in questo caso, e che sono relativamente isolati, quindi dispongono di un limitato accesso ad internet.

Procedo con l'installazione di dnsmasq, usando la linea

```
sudo apt-get install dnsmasq
```

Configuro il dns server aprendo il file di configurazione con

```
sudo nano /etc/dnsmasq.conf
```

Nel file di configurazione cerco la linea dove è specificato l'address e sostituisco con ciò che vogliamo configurare.

In questo caso sarà `address=/epicode.internal/192.168.32.100`, ed inoltre modifico la linea riguardante il listen address, che verrà modificata con `listen-address=127.0.0.1, 192.168.32.100`.

```
# address→name queries for 192.168.3/24 to nameserver 10.1.2.3
#server=/3.168.192.in-addr.arpa/10.1.2.3

# Add local-only domains here, queries in these domains are answered
# from /etc/hosts or DHCP only.
#local=/localnet/

# Add domains which you want to force to an IP address here.
# The example below send any host in double-click.net to a local
# web-server.
address=/epicode.internal/192.168.32.100

# --address (and --server) work with IPv6 addresses too.
#address=/www.thekelleys.org.uk/fe80::20d:60ff:fe36:f83

# Add the IPs of all queries to yahoo.com, google.com, and their
# subdomains to the vpn and search ipsets:
#ipset=/yahoo.com/google.com/vpn,search

# Add the IPs of all queries to yahoo.com, google.com, and their
# subdomains to netfilters sets, which is equivalent to

#user=
#group=

# If you want dnsmasq to listen for DHCP and DNS requests only on
# specified interfaces (and the loopback) give the name of the
# interface (eg eth0) here.
# Repeat the line for more than one interface.
#interface=
# Or you can specify which interface _not_ to listen on
#except-interface=
# Or which to listen on by address (remember to include 127.0.0.1 if
# you use this.)
listen-address=127.0.0.1, 192.168.32.100
# If you want dnsmasq to provide only DNS service on an interface,
# configure it as shown above, and then use the following line to
# disable DHCP and TFTP on it.
#no-dhcp-interface=

# On systems which support it, dnsmasq binds the wildcard address,
# even when it is listening on only some interfaces. It then discards
```

Infine procedo a fare un restart di dnsmasq con `systemctl restart dnsmasq`.

Adesso faccio una prova per vedere se si è installato correttamente e risulta attivo, come si vede dall'immagine sottostante.

```
(kali@kali)-[~]
$ sudo nano /etc/dnsmasq.conf

(kali@kali)-[~]
$ sudo systemctl restart dnsmasq

(kali@kali)-[~]
$ systemctl status dnsmasq
● dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
   Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; disabled; preset: disabled)
   Active: active (running) since Fri 2024-05-31 22:24:32 EDT; 33s ago
     Process: 4107 ExecStartPre=/usr/share/dnsmasq/systemd-helper checkconfig (code=exited, status=0/SUCCESS)
     Process: 4112 ExecStart=/usr/share/dnsmasq/systemd-helper exec (code=exited, status=0/SUCCESS)
     Process: 4119 ExecStartPost=/usr/share/dnsmasq/systemd-helper start-resolvconf (code=exited, status=0/SUCCESS)
    Main PID: 4118 (dnsmasq)
      Tasks: 1 (limit: 2271)
     Memory: 2.3M (peak: 4.0M)
        CPU: 90ms
    CGroup: /system.slice/dnsmasq.service
            └─4118 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -7 /etc/dnsmasq.d,.dpkg-dist,.dpkg-old>

May 31 22:24:32 kali systemd[1]: Starting dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server ...
May 31 22:24:32 kali dnsmasq[4118]: started, version 2.90 cachesize 150
May 31 22:24:32 kali dnsmasq[4118]: DNS service limited to local subnets
May 31 22:24:32 kali dnsmasq[4118]: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua>
May 31 22:24:32 kali dnsmasq[4118]: reading /etc/resolv.conf
```

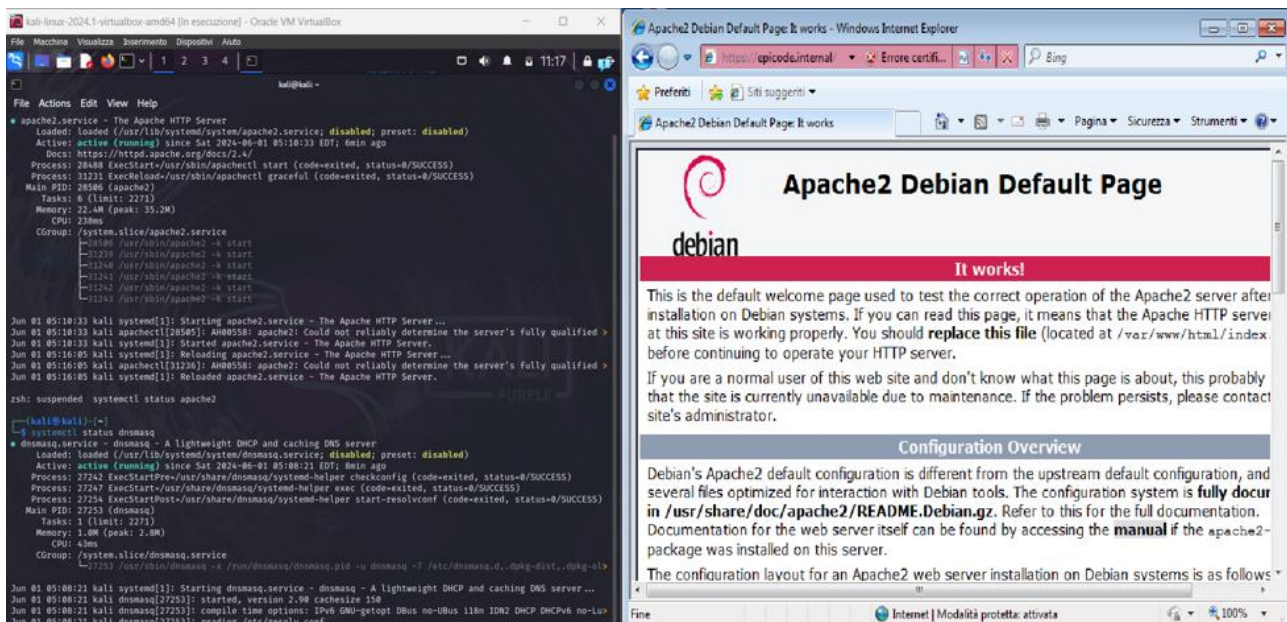
## TERZO STEP: Configurazione del server HTTPS

Installo Apache e attivo un SSL. Uso la linea `sudo apt-get install apache2` per installarlo, poi uso `sudo a2enmod ssl` per abilitare il modulo SSL (Secure Socket Layer) e utilizzare quindi HTTPS, ed inoltre, dando il comando `sudo a2ensite default-ssl`, mi abilito il sito specifico che ho configurato in precedenza sul server, ovvero `epicode.internal`. Da ultimo, procedo al reload di Apache con `sudo systemctl reload apache2`. Dopo di ciò, verifico lo stato di Apache.

```
(kali@kali)-[~]
└─$ systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Sat 2024-06-01 10:46:15 EDT; 12s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 75165 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 75191 (apache2)
    Tasks: 6 (limit: 2271)
   Memory: 21.4M (peak: 22.2M)
      CPU: 72ms
   CGroup: /system.slice/apache2.service
           └─75191 /usr/sbin/apache2 -k start
             └─75196 /usr/sbin/apache2 -k start
               └─75197 /usr/sbin/apache2 -k start
                 └─75198 /usr/sbin/apache2 -k start
                   └─75199 /usr/sbin/apache2 -k start
                     └─75200 /usr/sbin/apache2 -k start

Jun 01 10:46:15 kali systemd[1]: Starting apache2.service - The Apache HTTP Server...
Jun 01 10:46:15 kali apachectl[75182]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, us
Jun 01 10:46:15 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
```

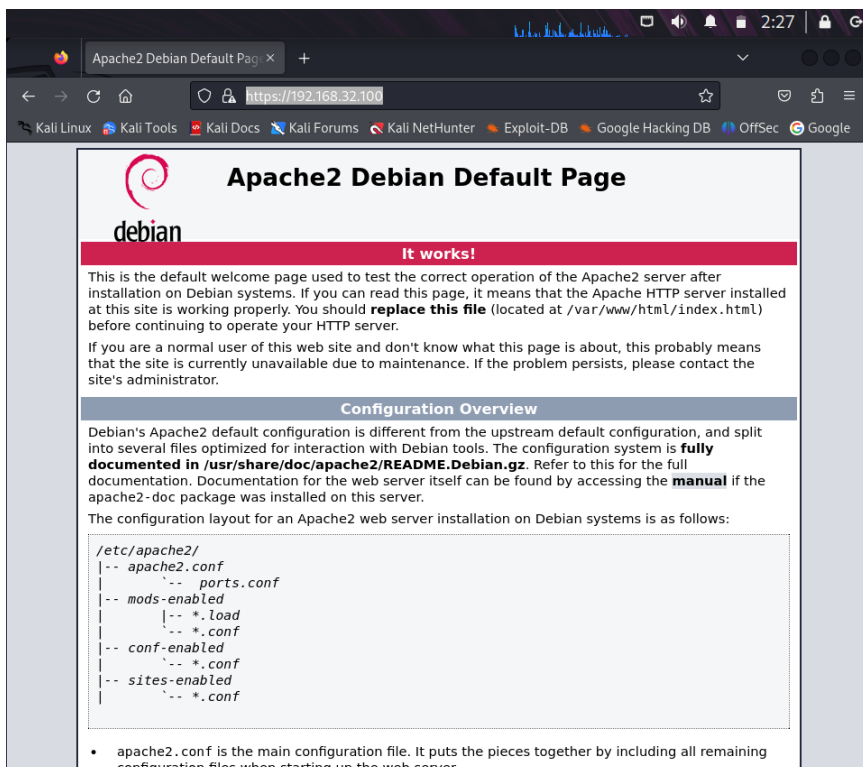
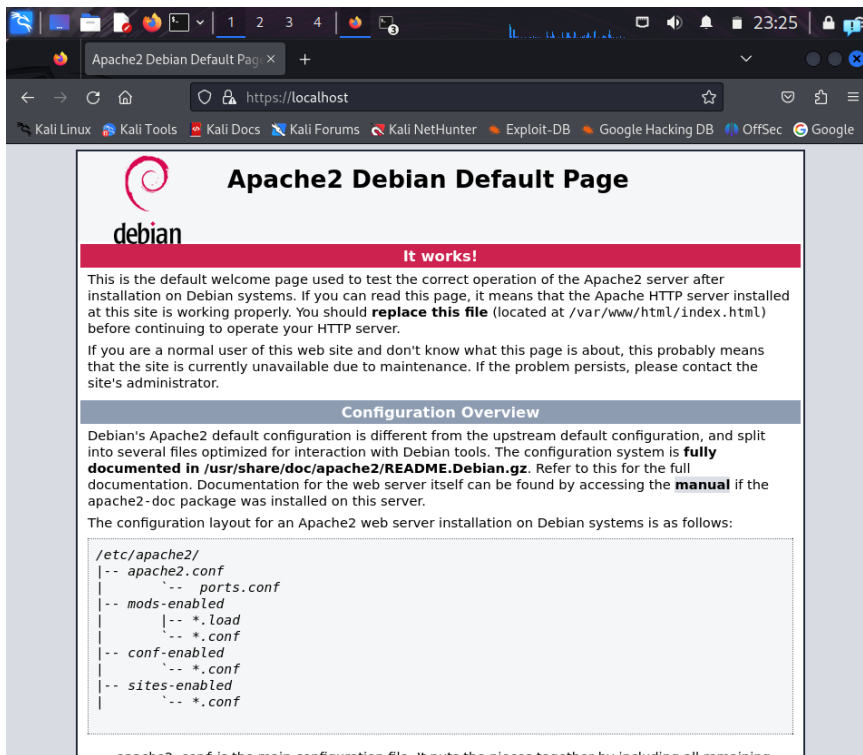
A questo punto, essendo il servizio attivo, posso fare una prima prova collegandomi tramite il browser della macchina Windows all'indirizzo <https://epicode.internal>

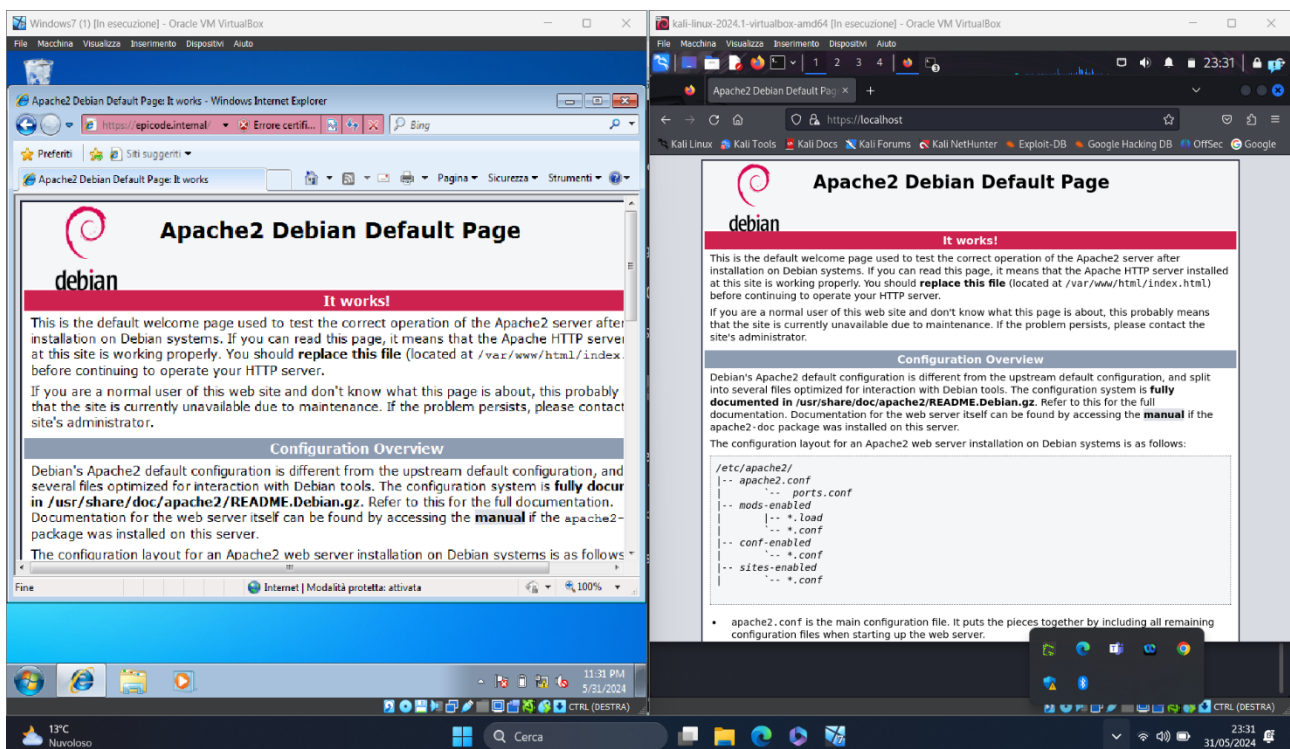
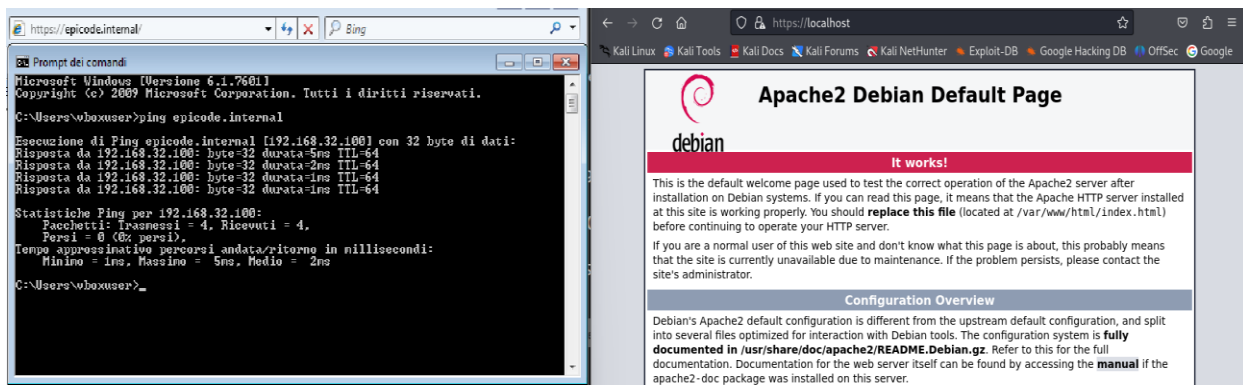




Faccio una ulteriore verifica, accedendo sul browser da Kali e collegandomi prima alla pagina <https://localhost> e in seguito a <https://192.168.32.100>, e provando che effettivamente il servizio è in esecuzione sulla macchina stessa.

Da macchina Windows posso inoltre verificare che la stessa comunica con l'hostname [epicode.internal](https://epicode.internal).





Collegandomi a <https://epicode.internal> da Windows riceverò un messaggio che mi avvisa che il certificato usato potrebbe non essere sicuro perché non verificato, in quanto si tratta di un certificato SSL autofirmato (**apache-selfsigned.crt**), ma, essendo in ambiente di test, si può bypassare senza alcun rischio e quindi visualizzare la pagina.

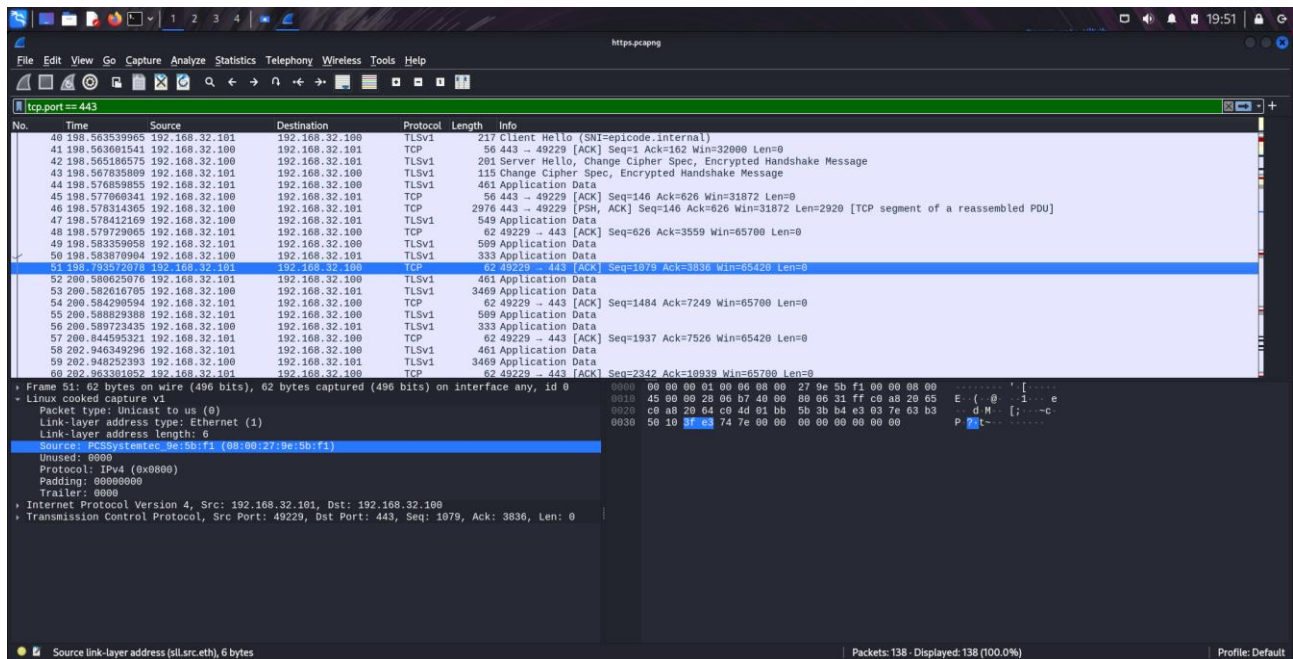
## QUARTO STEP: Catturamento traffico con Wireshark

### Pacchetti HTTPS

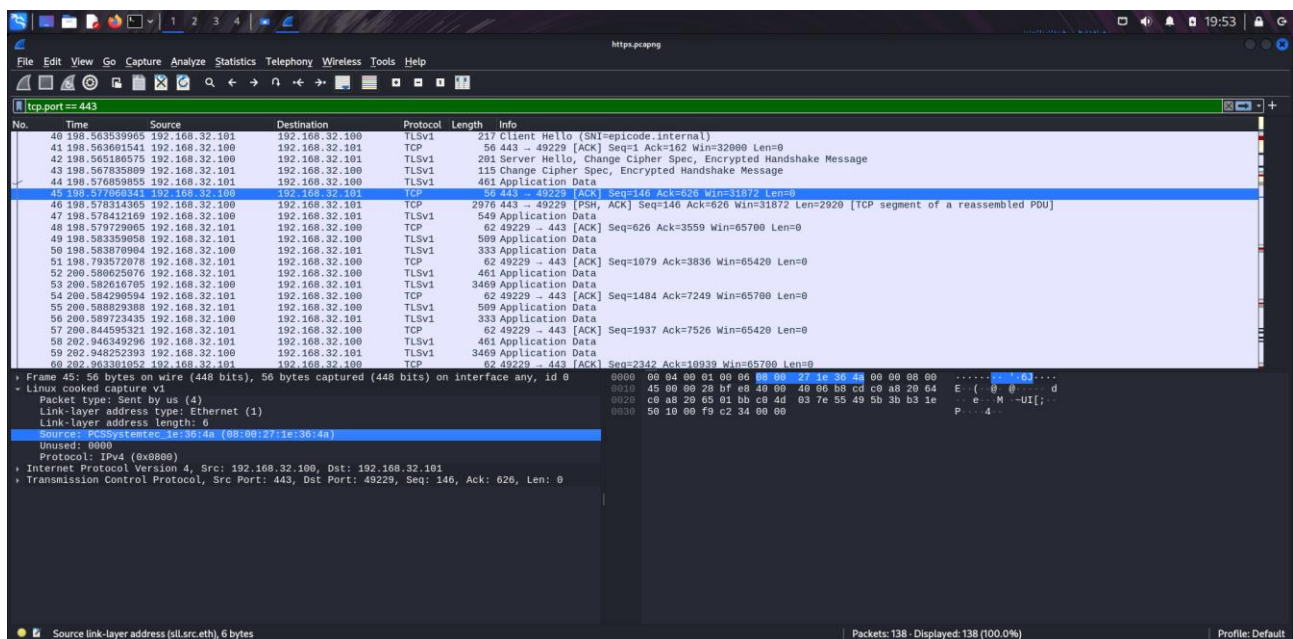
A questo punto, mantengo la pagina di browser su Windows 7 aperta su "https://epicode.internal" e intanto su Kali avvio Wireshark, per analizzare il traffico di

rete e procedere alla cattura dei pacchetti, che essendo pacchetti https non saranno immediatamente visualizzabili. Seleziono l'interfaccia di rete e imposto, quindi, un filtro di visualizzazione `tcp.port == 443`, in quanto è la porta standard utilizzata per il traffico HTTPS. Inoltre evidenzio il cambiamento dei MAC address sia in entrata che in uscita.

In entrata da Windows verso Kali

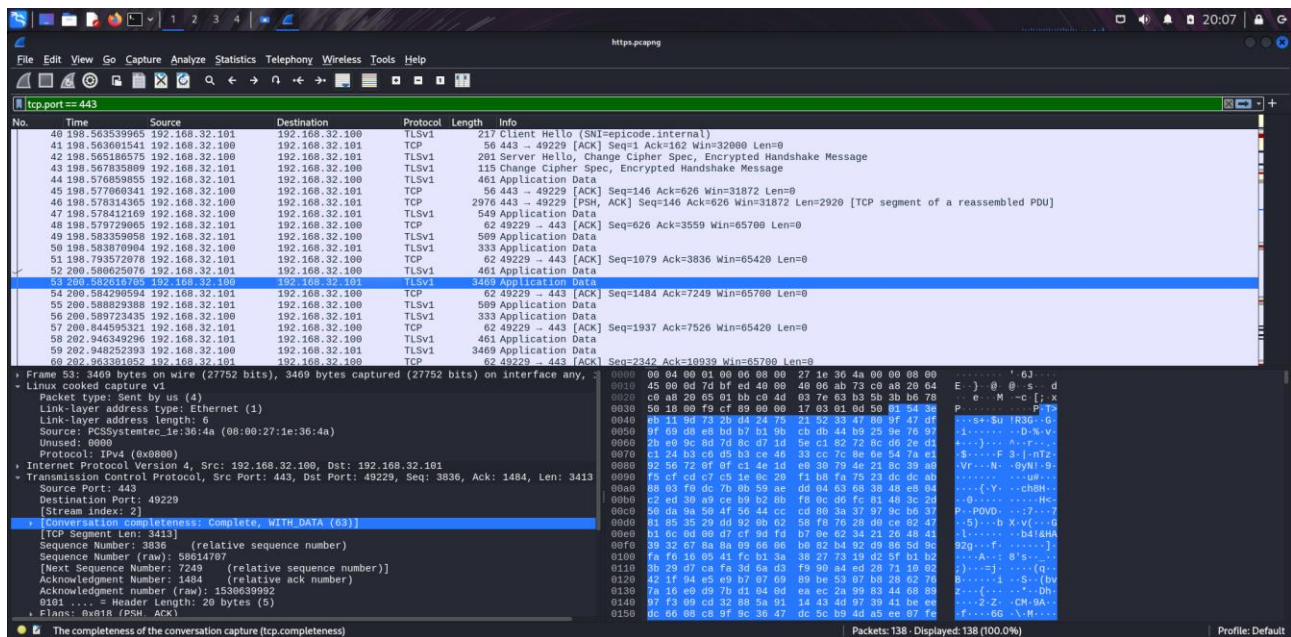
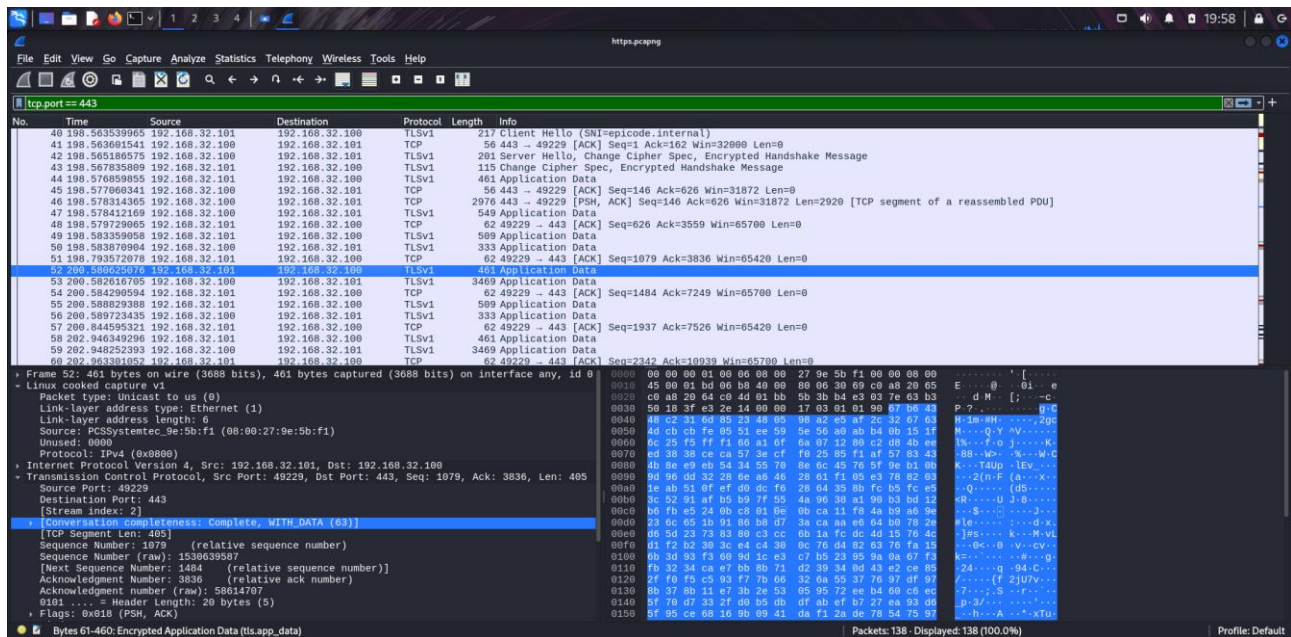


In uscita da Kali verso Windows 7





Notiamo che il MAC address cambia solo con il cambiamento della sorgente, e resta il medesimo se è il contenuto del pacchetto a cambiare. Il MAC address è, infatti, un identificatore unico assegnato all'hardware di rete e serve per identificare il dispositivo al livello data link (Modello OSI). È, quindi, utilizzato per indirizzare correttamente i pacchetti all'interno di una rete.



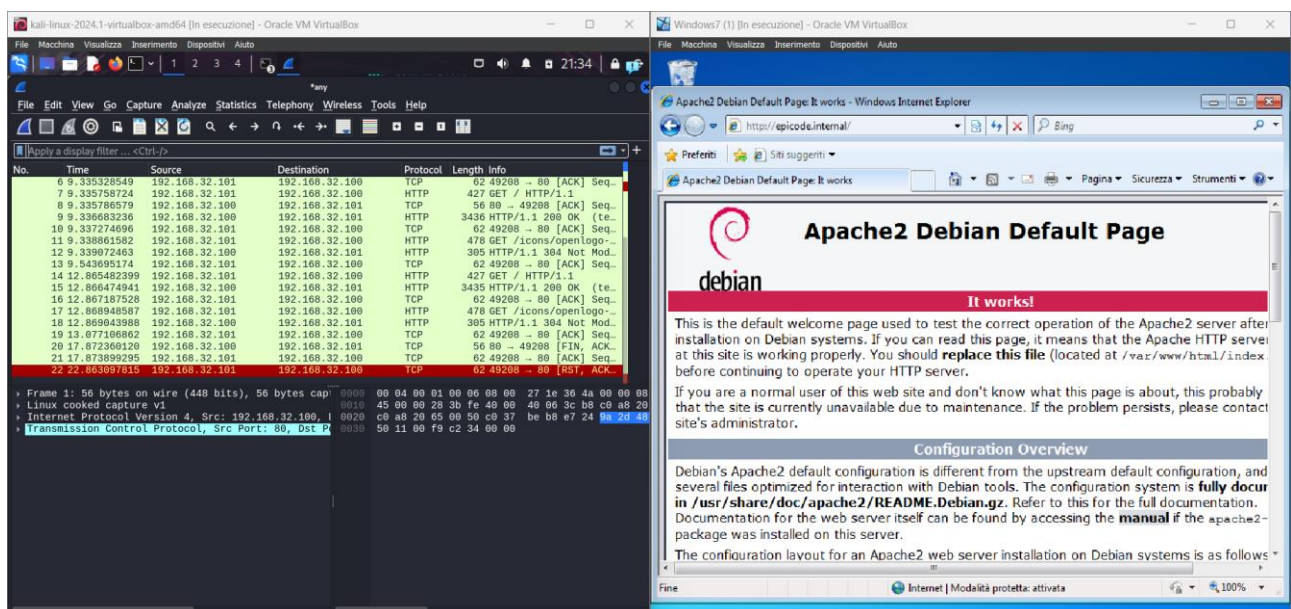


In queste immagini, il pacchetto nominato “Application Data” contiene i dati trasferiti dal protocollo TLS (SSL) e contiene i payload di HTTPS: in questo caso sono cifrati in quanto stiamo utilizzando dei protocolli di sicurezza e, quindi, il contenuto non sarà leggibile senza chiave di decrittazione. Anche in queste due immagini noto che il MAC address è rimasto uguale a quello presente nelle immagini precedenti perché la sorgente è uguale anche se il pacchetto è cambiato.

## Pacchetti HTTP

Prima di procedere con la cattura dei pacchetti HTTP disabilito il certificato SSL su Apache2 tramite la linea `sudo a2dissite default-ssl` e procedo con il reload di Apache2.

Adesso sul browser di Windows mi collegherò alla pagina <http://epicode.internal> e su Kali, sempre tramite Wireshark, monitorerò il traffico.



In questo caso, metto come filtro HTTP (si può usare anche la porta 80) e noto che i MAC address sono sempre gli stessi, ma soprattutto che il traffico è in chiaro, a differenza del caso precedente in cui era necessaria una chiave di decrittazione, e quindi, ad esempio, nel pacchetto text/html si potranno leggere, ed eventualmente modificare, le informazioni presenti, come il nome del browser, il nome dell'host che si sta raggiungendo ma anche il contenuto del messaggio che stiamo inviando.

## Traffico da Windows verso Kali

Wireshark capture of HTTP traffic from Windows to Kali. The packet list shows a GET request for /icons/openlogo-75.png. The packet details pane shows the HTTP request structure, including the User-Agent: Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC2; .NET CLR 2.0.50727; NET CLR 3.5.30729; .NET CLR 3.0.30729) and the request body.

## Traffico da Kali verso Windows

Wireshark capture of HTTP traffic from Kali to Windows. The packet list shows a GET request for /icons/openlogo-75.png. The packet details pane shows the HTTP request structure, including the User-Agent: Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC2; .NET CLR 2.0.50727; NET CLR 3.5.30729; .NET CLR 3.0.30729) and the request body.

MAC ADDRESS KALI 08:00:27:1e:36:4a

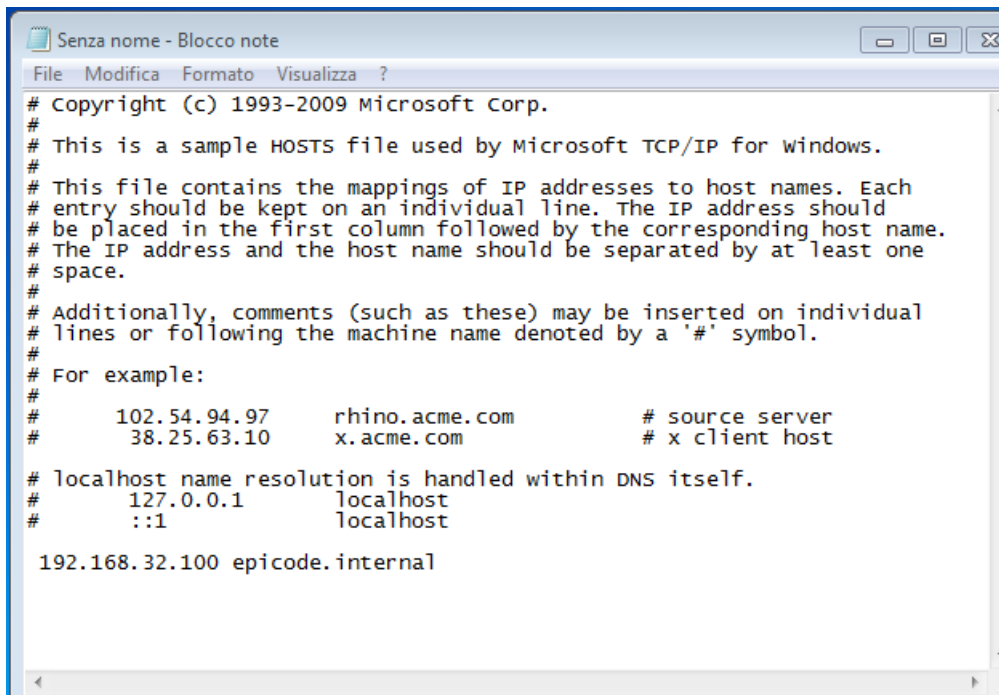
MAC ADDRESS WINDOWS 7 08:00:27:9e:5b:f1

## Metodo alternativo

Un metodo alternativo, per giungere comunque allo scopo finale di monitorare pacchetti di rete, consiste nel modificare il file host di Windows.

Questo serve a comunicare al sistema operativo di “risolvere” un particolare hostname, nello specifico epicode.internal, con un IP preciso, 192.168.32.100, invece di passare da un server DNS e quindi richiedere la risoluzione a quest’ultimo. Questo passaggio permette di bypassare alcune restrizioni ed è utile in ambiente di test anche se la risoluzione in questo caso risulta “forzata”. La parte che differisce sarà solo quella di configurazione iniziale, mentre la cattura con Wireshark, i MAC address e le differenze tra protocollo http e https rimarranno le stesse.

Il primo passaggio sarà quello di aprire, con privilegi di amministratore, il Blocco Note e da lì aprire il file host, tramite il percorso `C:\Windows\System32\drivers\etc\hosts`. Successivamente si aggiunge la linea `192.168.32.100 epicode.internal` in calce al testo, al fine di associare l’hostname al server.



```
# 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 epicode.internal
```

A questo punto si parte con l’installazione di Apache2 su Kali e, anche in questo caso, si verifica che sia in esecuzione, ad esempio, andando su <http://localhost> da browser.

Infine, si avvia Wireshark e si procede, come nel caso precedente, alla cattura dei pacchetti HTTP.

Per quanto riguarda la configurazione al fine di catturare pacchetti HTTPS, parto installando il certificato SSL auto-firmato.





```
GNU nano 8.0 /etc/apache2/sites-available/default-ssl.conf *
<VirtualHost *:443>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf

    # SSL Engine Switch:
    # Enable/Disable SSL for this virtual host.
    SSLEngine on

    # A self-signed (snakeoil) certificate can be created by installing
    # the ssl-cert package. See
    # /usr/share/doc/apache2/README.Debian.gz for more info.
    # If both key and certificate are stored in the same file, only the
    # SSLCertificateFile directive is needed.
    SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
    SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

    # Server Certificate Chain:
    # Point SSLCertificateChainFile at a file containing the
    # concatenation of PEM encoded CA certificates which form the
    # certificate chain for the server certificate. Alternatively
    # the referenced file can be the same as SSLCertificateFile
    # when the CA certificates are directly appended to the server
    # certificate for convinience.
    #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

    # Certificate Authority (CA):
    # Set the CA certificate verification path where to find CA
    # certificates for client authentication or alternatively one
    # huge file containing all of them (file must be PEM encoded)
    # Note: Inside SSLCACertificatePath you need hash symlinks to HTTP Server...
```

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute Serv ^C Location M-U Undo  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^\_ Go To Line M-E Redo

Procedo con l'abilitazione del modulo SSL e del sito specifico e finisco con il reload e il restart di Apache2 per attuare le modifiche.

```
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

(kali@kali)-[~]
└─$ sudo a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
    systemctl restart apache2

(kali@kali)-[~]
└─$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2

(kali@kali)-[~]
└─$ systemctl reload apache2 66 systemctl restart apache2

(kali@kali)-[~]
└─$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Fri 2024-05-31 17:16:42 EDT; 17s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 75831 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 75834 (apache2)
    Tasks: 6 (limit: 2271)
   Memory: 15.3M (peak: 15.7M)
      CPU: 86ms
   CGroup: /system.slice/apache2.service
           └─75834 /usr/sbin/apache2 -k start
             └─75839 /usr/sbin/apache2 -k start
               └─75840 /usr/sbin/apache2 -k start
                 └─75841 /usr/sbin/apache2 -k start
                   └─75842 /usr/sbin/apache2 -k start
                     └─75843 /usr/sbin/apache2 -k start

May 31 17:16:42 kali systemd[1]: Starting apache2.service - The Apache HTTP Server ...
May 31 17:16:42 kali apachectl[75833]: AH00558: apache2: Could not reliably determine the server's fully qualified
May 31 17:16:42 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-20/20 (END)
```

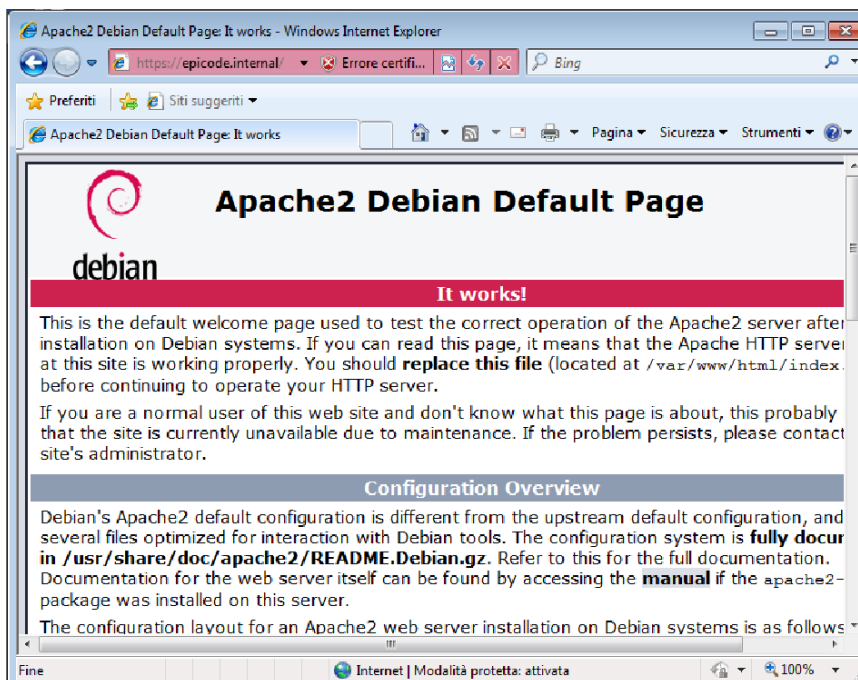
Ora posso collegarmi ad <https://epicode.internal> da Windows ed <https://localhost> da Kali.

Da entrambi i browser ci saranno degli avvisi di sicurezza per segnalare che il certificato utilizzato non è autenticato, quindi non firmato da una CA (Autorità di Certificazione) riconosciuta.

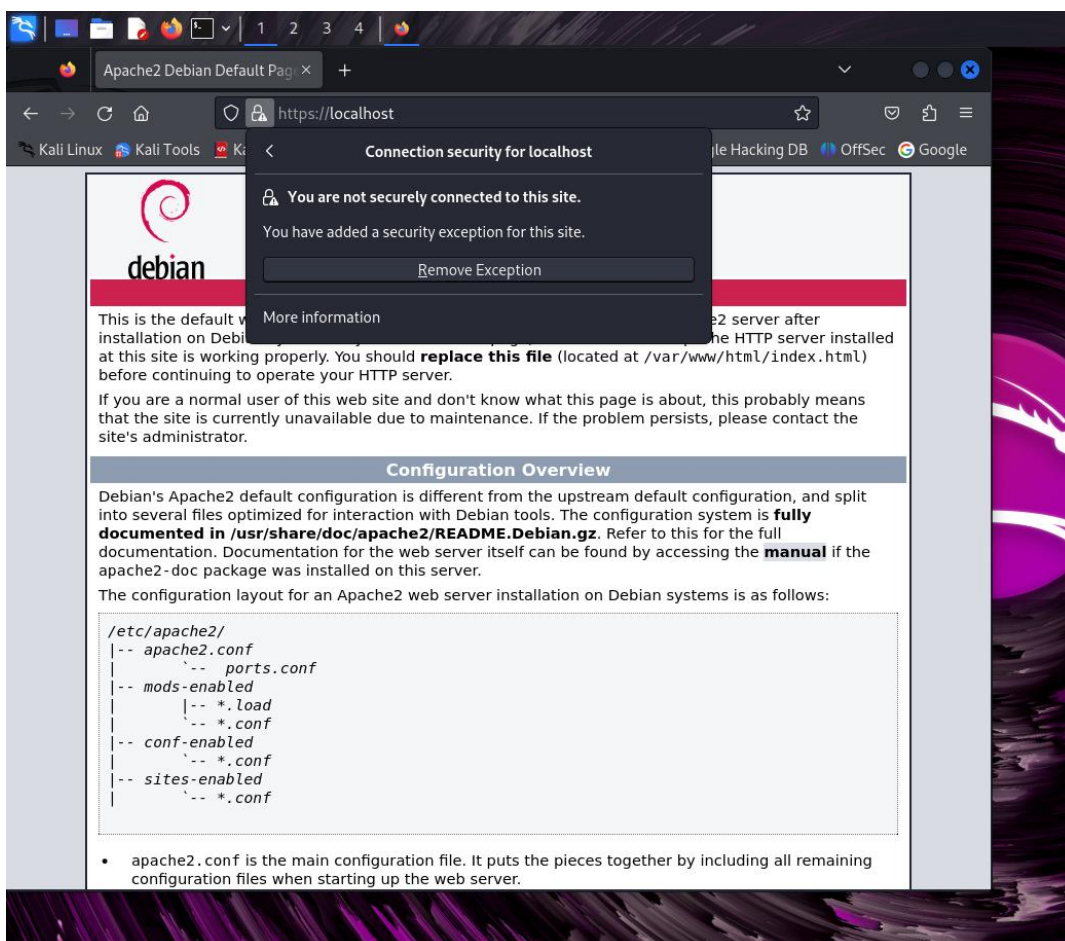
In tutti e due i casi il problema è facilmente bypassabile facendo un'eccezione.



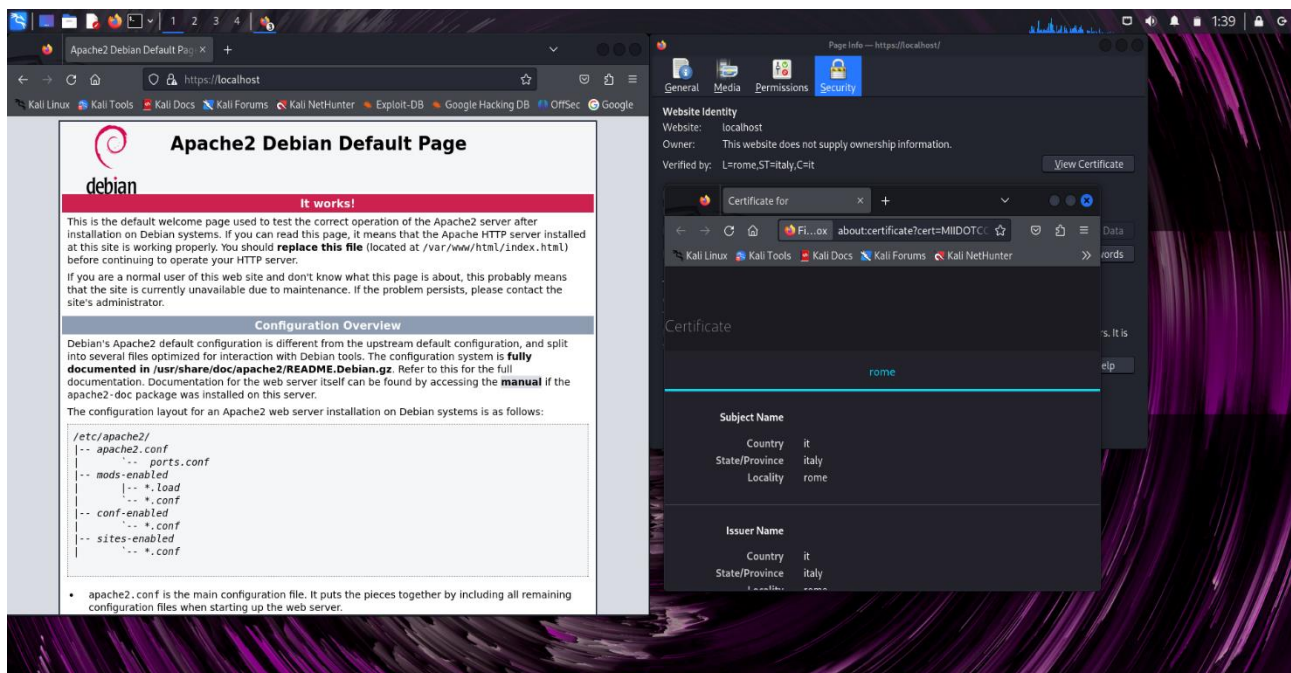
In Windows si segnala l'errore del certificato nella barra di ricerca



In Kali il certificato creerà un'eccezione per il sito specifico e la pagina verrà visualizzata



Questo è il certificato auto-firmato per https://localhost



La conseguente cattura di pacchetti con Wireshark darà risultati uguali al precedente metodo e non cambierà niente riguardo i MAC address e le differenze tra i pacchetti http e https.

