

Di seguito viene illustrato come è stata svolta la richiesta di instaurare un laboratorio virtuale con lo scopo di simulare un server con i servizi DNS e HTTP/HTTPS attivi.

Per la realizzazione è stato utilizzato, come richiesto una macchina Kali linux e una macchina Windows 10.

invece per l'attivazione dei servizi è stato scelto **Apache** per gestire HTTP e HTTPS, mentre **dnsmasq** per la risoluzione dei domini locali.

Il primo passo è stato assegnare gli indirizzi IP richiesti:

```
(kali@kali)-[~]
$ sudo ip addr add 192.168.32.100/24 dev eth0
[sudo] password for kali:

(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.32.100  netmask 255.255.255.0  broadcast 0.0.0.0
    inet6 fe80::1c34:7eb6:c3ec:4fca  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:1f:b7:23  txqueuelen 1000  (Ethernet)
    RX packets 1249  bytes 94917 (92.6 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 488  bytes 40580 (39.6 KiB)
    TX errors 0  dropped 36 overruns 0  carrier 0  collisions 0
```

Kali Linux: 192.168.32.100

```
Amministratore: Prompt dei comandi
netsh
netsh>interface ip
netsh interface ipv4>set address "ethernet" static 192.168.32.101

netsh interface ipv4>set dns "ethernet" static 192.168.32.100

netsh interface ipv4>quit

ipconfig

Configurazione IP di Windows

Scheda Ethernet Ethernet:

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::cc05:68e3:b68:9e84%4
    Indirizzo IPv4. . . . . : 192.168.32.101
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . :
```

Windows 10:
192.168.32.101

(DNS: 192.168.32.100)

HTTP e HTTPS

Essendo apache già presente in Kali, è stato sufficiente avviarlo

sudo systemctl start apache2 (il 2 si riferisce alla sua versione)

Successivamente è stato eseguito un test di raggiungimento IP dalla macchina windows tramite browser per verificare il corretto funzionamento del servizio:



Per il servizio HTTPS sono stati necessari più passaggi:

- l'abilitazione del modulo SSL per permettere l'utilizzo di HTTPS
- la configurazione del certificato ssl di default (auto firmato)
- il riavvio del servizio per applicare le modifiche.

sudo a2enmod ssl

sudo a2ensite default-ssl

sudo systemctl restart apache2

Anche qui, successivamente ai comandi è stato eseguito un test di verifica dal browser, sempre dalla macchina windows:



DNS

Per il servizio DNS, come detto all'inizio, è stato utilizzato **dnsmasq**, un server DNS leggero per domini locali

sudo apt install -y dnsmasq

Dopo l'installazione è stato necessario creare un file di configurazione in modo che il servizio potesse associare il nome di dominio al suo IP specifico (**epicode.internal = 192.168.32.100**)

sudo nano /etc/dnsmasq.d/epicode.conf

All'interno del file sono state inserite le seguenti righe di associazione:

address=/epicode.internal/192.168.32.100

listen-address=192.168.32.100

Infine è stato riavviato il servizio per applicare le modifiche:

sudo systemctl restart dnsmasq

Le richieste fatte con HTTP e HTTPS dalla macchina windows hanno dato esito positivo:



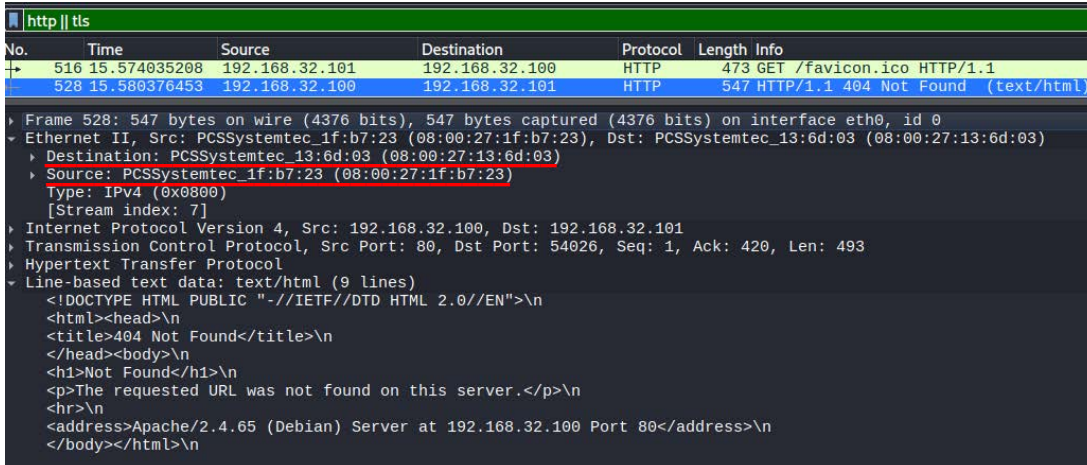
```
C:\Users\admin>ping epicode.internal

Esecuzione di Ping epicode.internal [192.168.32.100] con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
```

Sniffing di pacchetti

Per la cattura e l'analisi dei pacchetti HTTP e TLS (https) è stato utilizzato Wireshark.

L'esecuzione di un http request dalla macchina windows ha generato evidente traffico visibile nel programma in ascolto:

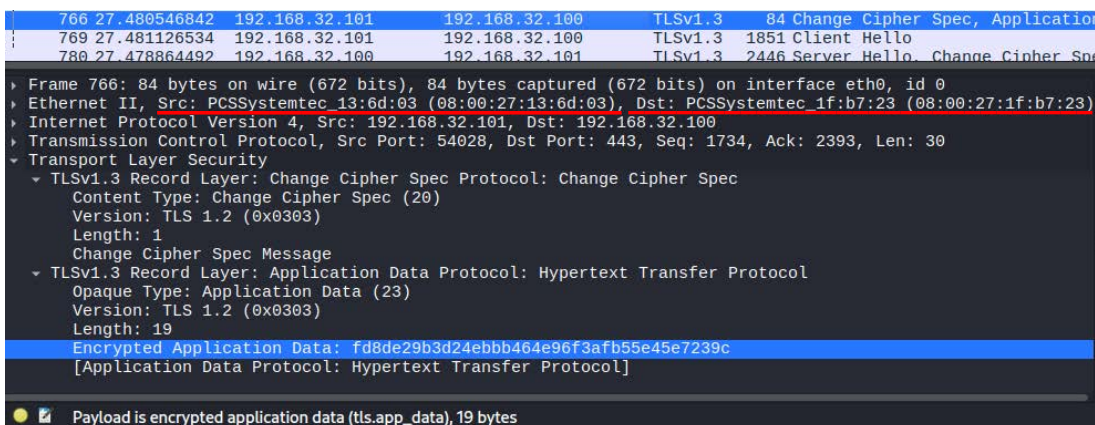


The image shows a Wireshark packet capture of an HTTP transaction. The top pane shows the packet list with two packets: a GET request for /favicon.ico and a 404 Not Found response. The bottom pane shows the details of the selected packet (the 404 response), including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTML body of the response is visible, showing a 404 error message and server information (Apache/2.4.65 (Debian) Server at 192.168.32.100 Port 80).

```
http | tls
No.    Time           Source            Destination        Protocol  Length  Info
+-----+-----+-----+-----+-----+-----+
516    15.574035208    192.168.32.101    192.168.32.100    HTTP      473     GET /favicon.ico HTTP/1.1
528    15.580376453    192.168.32.100    192.168.32.101    HTTP      547     HTTP/1.1 404 Not Found (text/html)

Frame 528: 547 bytes on wire (4376 bits), 547 bytes captured (4376 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_1f:b7:23 (08:00:27:1f:b7:23), Dst: PCSSystemtec_13:6d:03 (08:00:27:13:6d:03)
  Destination: PCSSystemtec_13:6d:03 (08:00:27:13:6d:03)
  Source: PCSSystemtec_1f:b7:23 (08:00:27:1f:b7:23)
  Type: IPv4 (0x0800)
  [Stream index: 7]
Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
Transmission Control Protocol, Src Port: 80, Dst Port: 54026, Seq: 1, Ack: 420, Len: 493
Hypertext Transfer Protocol
  Line-based text data: text/html (9 lines)
    <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n
    <html><head>\n
    <title>404 Not Found</title>\n
    </head><body>\n
    <h1>Not Found</h1>\n
    <p>The requested URL was not found on this server.</p>\n
    <hr>\n
    <address>Apache/2.4.65 (Debian) Server at 192.168.32.100 Port 80</address>\n
    </body></html>\n
```

**Il pacchetto HTTP in ricezione mostra evidente trasparenza dei dati :
L'HTML è in chiaro, e trapelano anche le info del server (Apache/2.4.65).**



The image shows a Wireshark packet capture of a TLS handshake and an encrypted HTTP response. The top pane shows three packets: a Change Cipher Spec, a Client Hello, and a Server Hello. The bottom pane shows the details of the selected packet (the Server Hello), including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Transport Layer Security. The TLS record layer shows the Change Cipher Spec and the Application Data Protocol (Hypertext Transfer Protocol). The encrypted application data is visible, but the payload is encrypted.

```
766 27.480546842 192.168.32.101 192.168.32.100 TLSv1.3 84 Change Cipher Spec, Application
769 27.481126534 192.168.32.101 192.168.32.100 TLSv1.3 1851 Client Hello
780 27.478864492 192.168.32.100 192.168.32.101 TLSv1.3 2446 Server Hello, Change Cipher Spec

Frame 766: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_13:6d:03 (08:00:27:13:6d:03), Dst: PCSSystemtec_1f:b7:23 (08:00:27:1f:b7:23)
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 54028, Dst Port: 443, Seq: 1734, Ack: 2393, Len: 30
Transport Layer Security
  TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 19
    Encrypted Application Data: fd8de29b3d24ebbb464e96f3afb55e45e7239c
    [Application Data Protocol: Hypertext Transfer Protocol]

Payload is encrypted application data (tls.app_data), 19 bytes
```

Un pacchetto TLS (HTTPS) invece non mostra dati in chiaro, il contenuto HTTP è cifrato e illeggibile grazie alla crittografia fornita dal protocollo TLS.

Tuttavia alcune informazioni meta (meta-dati) rimangono visibili (come l'indirizzo IP e MAC del server e del client che ha fatto richiesta).

L'utilizzo di HTTP espone il traffico a potenziali intercettazioni da parte di terzi, consentendo la lettura di dati sensibili (ad esempio credenziali e informazioni personali).

HTTPS, tramite il protocollo TLS, cifra la comunicazione e mitiga in modo significativo questi rischi.