

W4D4 - Progetto Finale

Obiettivo dell'esercitazione

In questo esercizio, abbiamo simulato un'architettura client-server in un ambiente di laboratorio virtuale. L'obiettivo era configurare un server web su Kali Linux (con indirizzo IP 192.168.32.100) e far sì che un client Windows (con indirizzo IP 192.168.32.101) potesse accedere a una risorsa tramite un browser web, utilizzando l'hostname `epicode.internal`.

Abbiamo configurato due versioni del server:

1. Un server HTTPS, che utilizza un certificato SSL per cifrare le comunicazioni;
2. Un server HTTP, che trasmette i dati in chiaro.

Successivamente, abbiamo utilizzato Wireshark, uno strumento di analisi del traffico di rete, per intercettare e analizzare le comunicazioni tra il client e il server. In particolare, abbiamo evidenziato:

- I MAC address di sorgente e destinazione;
- Il contenuto delle richieste e delle risposte, osservando le differenze tra il traffico HTTPS (cifrato) e HTTP (in chiaro).

Ho scelto un linguaggio semplice e diretto, evitando tecnicismi non necessari. L'obiettivo è rendere il report comprensibile a un pubblico ampio, pur mantenendo un livello di dettaglio sufficiente per chi è interessato ad approfondire.

Configurazione delle macchine

Per far comunicare correttamente il client (Windows) e il server (Kali), abbiamo configurato gli indirizzi IP statici e la risoluzione DNS locale.

Configurazione Kali Linux:

1. Abbiamo assegnato a Kali l'indirizzo IP 192.168.32.100;
2. Per farlo, abbiamo modificato il file di configurazione di rete `/etc/network/interfaces`.



```
(kali@kali)-[~]  
$ sudo nano /etc/network/interfaces
```

Configurazione dell'interfaccia di rete

```

GNU nano 8.3 /etc/network/interfaces
# This file describes the network interfaces
# and how to activate them. For more information, see
# the file /usr/share/doc/network-base/README.interfaces

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1

```

Modifica del file di configurazione di rete

3. Dopo aver salvato le modifiche, abbiamo riavviato il servizio di rete con il comando "sudo systemctl restart networking".

Configurazione del DNS su Kali Linux:

1. Modifica del file /etc/hosts con il comando: "sudo nano /etc/hosts"

```

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

```

Apertura file /etc/hosts

2. Abbiamo aggiunto la seguente riga: 192.168.32.100 epicode.internal:

```

GNU nano 8.3 /etc/hosts
127.0.0.1 localhost
127.0.1.1 kali
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.32.100 epicode.internal

```

Modifica del file /etc/hosts

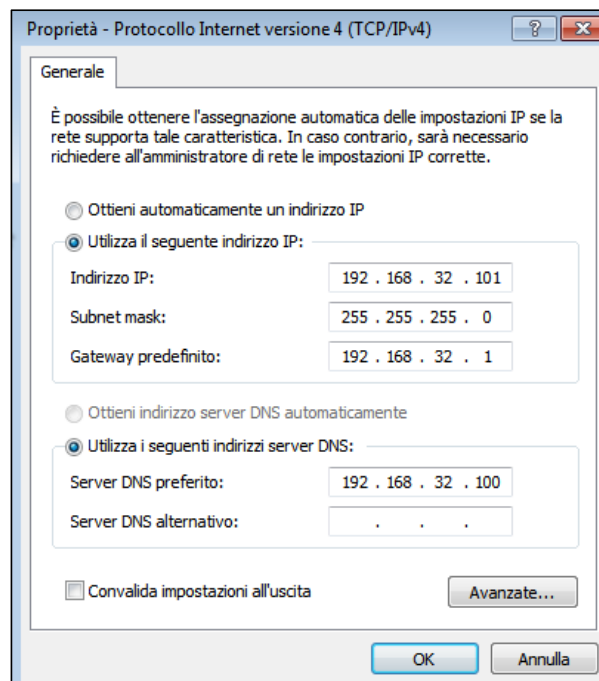
3. Questo associa l'hostname epicode.internal all'indirizzo IP 192.168.32.100;
4. Infine abbiamo usato il comando ping per verificare che l'hostname fosse risolto correttamente;
5. Se il comando restituisce risposte da 192.168.32.100, la configurazione è corretta.

```
(kali@kali)-[~]
$ ping 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=0.021 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=3 ttl=64 time=0.025 ms
^C
— epicode.internal ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.021/0.028/0.038/0.007 ms
```

ping epicode.internol

Configurazione di Windows:

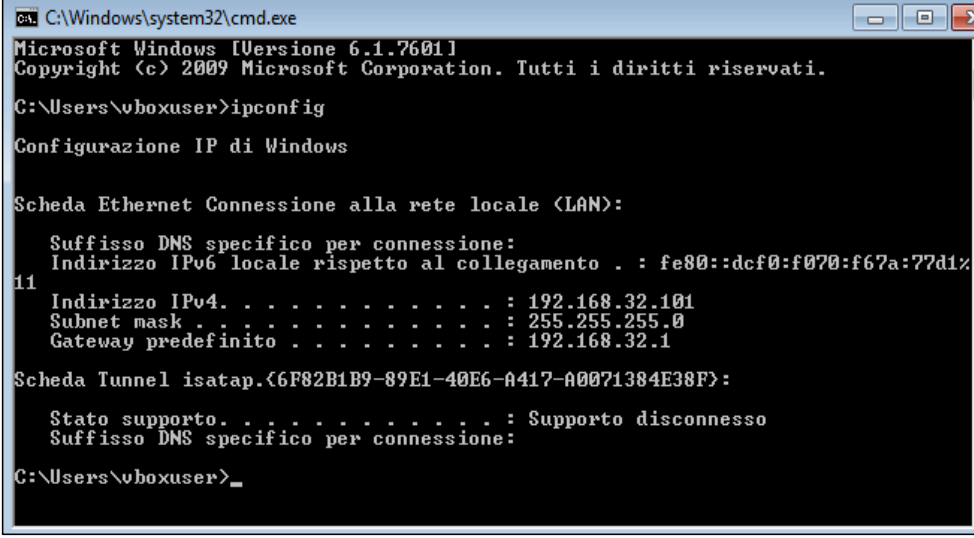
1. Abbiamo assegnato a Windows l'indirizzo IP 192.168.32.101;
2. Subnet mask: 255.255.255.0;
3. Gateway predefinito: 192.168.32.1



Configurazione TCP/IPv4

Dopo aver configurato l'indirizzo IP statico su Windows, abbiamo usato il comando "ipconfig" per verificare le impostazioni:

1. Aprire il prompt dei comandi (cercare "cmd" nel menu Start);
2. Eseguire ipconfig e verificare l'output;
3. Se tutto corrisponde, la configurazione è corretta.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versione 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\ vboxuser>ipconfig

Configurazione IP di Windows

Scheda Ethernet Connessione alla rete locale (LAN):

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::dcf0:f070:f67a:77d1%11
    Indirizzo IPv4. . . . . : 192.168.32.101
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . : 192.168.32.1

Scheda Tunnel isatap.{6F82B1B9-89E1-40E6-A417-A0071384E38F}:

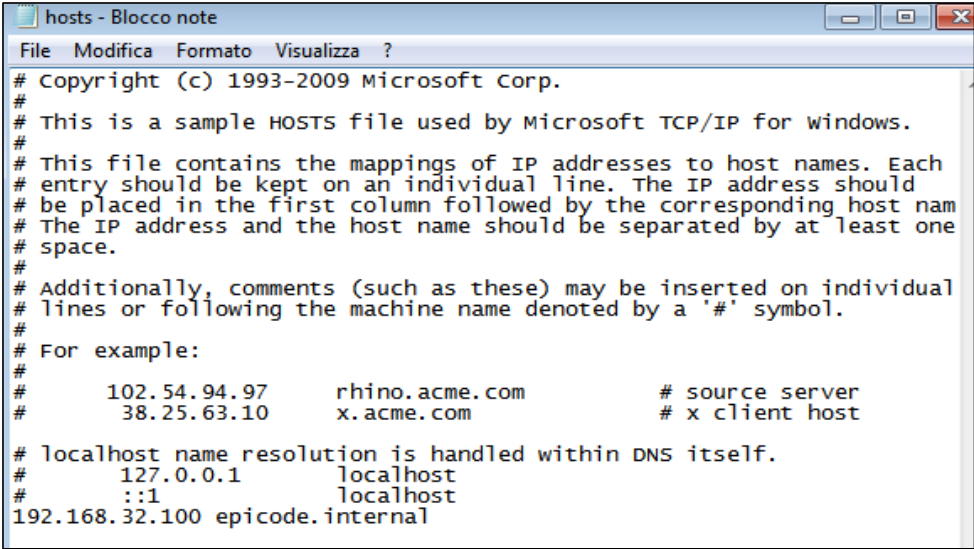
    Stato supporto. . . . . : Supporto disconnesso
    Suffisso DNS specifico per connessione:

C:\Users\ vboxuser>
```

Esecuzione del comando ipconfig

Configurazione del file hosts:

1. Per risolvere l'hostname epicode.internal all'indirizzo IP del server (192.168.32.100), abbiamo modificato il file hosts di Windows utilizzando il blocco note con permessi da amministratore.
2. Ecco la riga aggiunta: 192.168.32.100 epicode.internal



```
hosts - Blocco note
File  Modifica  Formato  Visualizza  ?

# 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 nam
# 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
```

Configurazione del file hosts

Verifica della risoluzione DNS:

1. Per assicurarci che la configurazione funzionasse, abbiamo eseguito un ping a epicode.internal da Windows: ping epicode.internal
2. Il comando ha restituito risposte dall'indirizzo IP 192.168.32.100, confermando che la risoluzione del DNS era corretta

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

C:\Users\vboxuser>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
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 = 0ms, Medio = 0ms

C:\Users\vboxuser> _
```

ping epicode.internal

Perchè questa configurazione è importante?

Modificare il file hosts su entrambe le macchine ci ha permesso di risolvere l'hostname senza utilizzare un server DNS esterno.

Configurazione del sito HTTPS su Kali Linux

Dopo aver configurato gli indirizzi IP e il DNS, abbiamo preparato Kali Linux per ospitare un sito web sicuro utilizzando il protocollo HTTPS. Ecco i passaggi chiave:

Generazione del Certificato SSL (Per abilitare HTTPS, abbiamo generato un certificato SSL autofirmato utilizzando openssl):

1. Creazione della directory per i certificati

```
(kali@kali)-[~]
$ sudo mkdir -p /etc/apache2/ssl
```

Directory per i certificati

2. Per la generazione del certificato abbiamo eseguito il seguente comando:

```
(kali@kali)-[~]
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/epicode.internal.key -out /etc/apache2/ssl/epicode.internal.crt
```

3. Durante il processo, abbiamo inserito informazioni come il nome di dominio (epicode.internal):

```
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Campania
Locality Name (eg, city) []:Napoli
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:epicode.internal
Email Address []:
```

Compilazione certificato

Configurazione di Apache per HTTPS:

Abbiamo configurato Apache per utilizzare il certificato SSL e abilitare HTTPS

1. Creazione del file di configurazione:

```
(kali@kali)-[~]
$ sudo nano /etc/apache2/sites-available/epicode.internal-ssl.conf
```

2. Ecco il contenuto del file:

```
GNU nano 8.3 /etc/apache2/sites-available/epicode.internal
<VirtualHost *:443>
    ServerAdmin webmaster@epicode.internal
    ServerName epicode.internal
    DocumentRoot /var/www/epicode.internal

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/epicode.internal.crt
    SSLCertificateKeyFile /etc/apache2/ssl/epicode.internal.key

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

3. Abilitazione del modulo SSL:

```
(kali@kali)-[~]
$ sudo a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
```

4. Abilitazione del sito HTTPS:

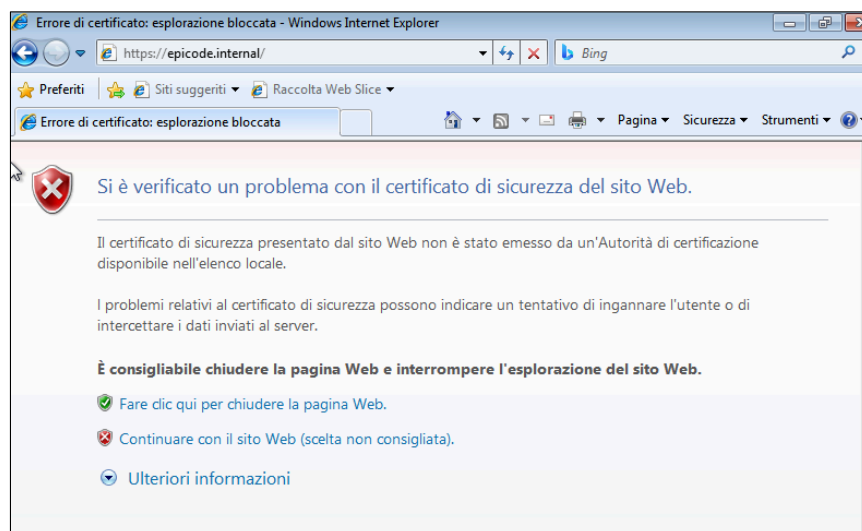
```
(kali@kali)-[~]  
$ sudo a2ensite epicode.internal-ssl.conf  
Site epicode.internal-ssl already enabled
```

Abbiamo abilitato il sito HTTPS e riavviato Apache.

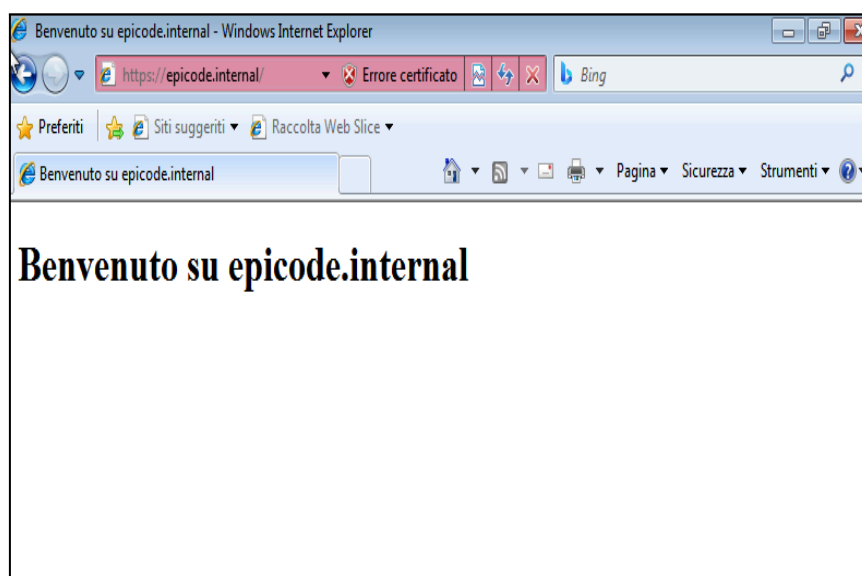
Verifica del Sito HTTPS:

Per assicurarci che tutto funzionasse correttamente, abbiamo eseguito i seguenti test:

1. Abbiamo aperto un browser su Windows e visitato "<https://epicode.internal/>"
2. Poiché il certificato è autofirmato, il browser ha mostrato un avviso di sicurezza. Abbiamo ignorato l'avviso per accedere al sito:



Avviso di sicurezza



Analisi del traffico HTTPS con Wireshark

In questa sezione, abbiamo utilizzato Wireshark, uno strumento di analisi del traffico di rete, per intercettare e analizzare le comunicazioni tra il client (Windows) e il server (Kali) quando questi comunicano tramite il protocollo HTTPS. L'obiettivo è stato:

1. Catturare il traffico HTTPS tra le due macchine;
2. Evidenziare i MAC address di sorgente e destinazione;
3. Analizzare il contenuto cifrato delle richieste e delle risposte;
4. Confrontare il traffico HTTPS con quello HTTP (nella prossima sezione).

Grazie a Wireshark, abbiamo potuto osservare come il traffico HTTPS sia cifrato, rendendo impossibile leggere il contenuto delle comunicazioni senza la chiave di decifratura. Questo dimostra l'importanza di HTTPS per proteggere i dati sensibili durante la trasmissione.

ip.addr == 192.168.32.100 && ip.addr == 192.168.32.101 && tcp.port == 443

Filtro applicato per visualizzare solo il traffico HTTPS tra il client 192.168.32.101 e il server 192.168.32.100 sulla porta 443

No.	Time	Source	Destination	Protocol	Length	Info
54	38.595521924	192.168.32.101	192.168.32.100	TCP	60	49165 → 443 [FIN, ACK] Seq=221 Ack=146 Win=65552 Len=0
55	38.595605717	192.168.32.100	192.168.32.101	TLSv1	91	Encrypted Alert
56	38.595809917	192.168.32.101	192.168.32.100	TCP	66	49166 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
57	38.595822066	192.168.32.100	192.168.32.101	TCP	66	443 → 49166 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
58	38.596038977	192.168.32.101	192.168.32.100	TCP	60	49165 → 443 [RST, ACK] Seq=222 Ack=183 Win=0 Len=0
59	38.596317525	192.168.32.101	192.168.32.100	TCP	60	49166 → 443 [ACK] Seq=1 Ack=1 Win=65780 Len=0
60	38.59648469	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello (SNI=epicode.internal)
61	38.596661490	192.168.32.100	192.168.32.101	TCP	54	443 → 49166 [ACK] Seq=1 Ack=162 Win=64128 Len=0
62	38.597024130	192.168.32.100	192.168.32.101	TLSv1	199	Server Hello, Change Cipher Spec, Encrypted Handshake Message
63	38.597664475	192.168.32.101	192.168.32.100	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
64	38.600709093	192.168.32.101	192.168.32.100	TLSv1	507	Application Data
65	38.600776755	192.168.32.100	192.168.32.101	TCP	54	443 → 49166 [ACK] Seq=146 Ack=674 Win=64128 Len=0
66	38.601217523	192.168.32.100	192.168.32.101	TLSv1	539	Application Data
67	38.806068379	192.168.32.101	192.168.32.100	TCP	60	49166 → 443 [ACK] Seq=674 Ack=631 Win=65068 Len=0
68	43.602429075	192.168.32.100	192.168.32.101	TLSv1	91	Encrypted Alert
69	43.602534406	192.168.32.100	192.168.32.101	TCP	54	443 → 49166 [FIN, ACK] Seq=668 Ack=674 Win=64128 Len=0
70	43.602879365	192.168.32.101	192.168.32.100	TCP	60	49166 → 443 [ACK] Seq=674 Ack=669 Win=65032 Len=0

Frame 44: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_99:2d:aa (08:00:27:99:2d:aa), Dst: PCSSystemtec_04:42:0f (08:00:27:04:42:0f)
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 49164, Dst Port: 443, Seq: 456, Ack: 1137, Len: 0

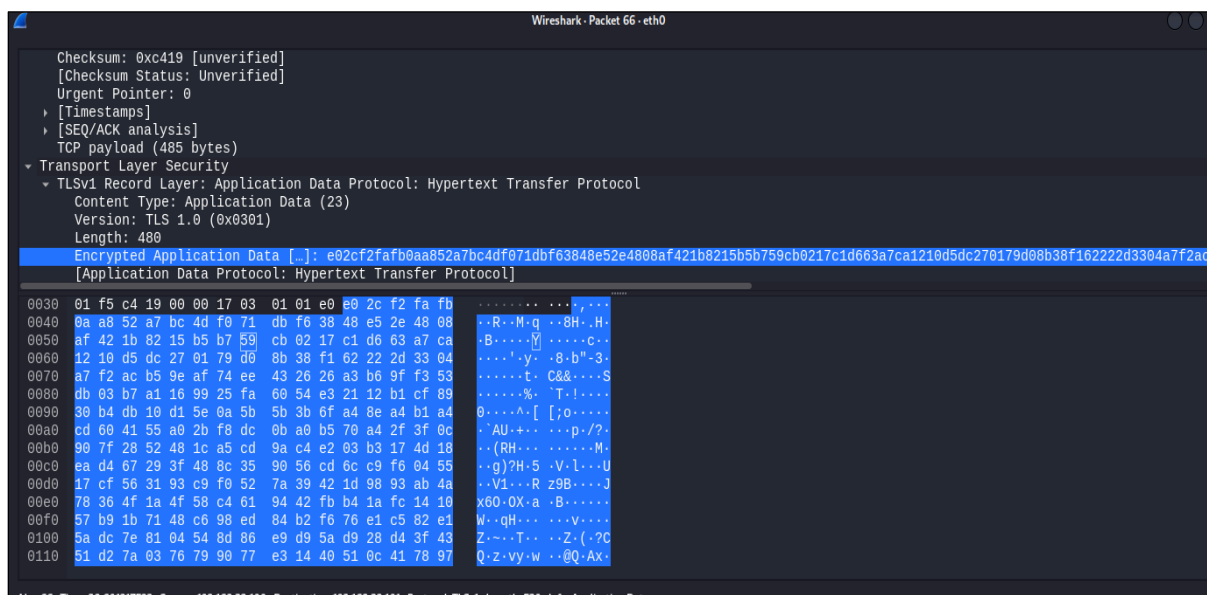
Pacchetti HTTPS catturati: il traffico è cifrato e non è possibile leggere il contenuto delle richieste e delle risposte

Wireshark - Packet 60 - eth0
Frame 60: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_99:2d:aa (08:00:27:99:2d:aa), Dst: PCSSystemtec_04:42:0f (08:00:27:04:42:0f)
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 49166, Dst Port: 443, Seq: 1, Ack: 1, Len: 161
Transport Layer Security

Sono visibili i MAC address di sorgente e destinazione, oltre alle informazioni sul protocollo TLS

	Protocol	Length	Info
	TCP	66	49164 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
	TCP	66	443 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
	TCP	60	49164 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
	TLSv1	183	Client Hello (SNI=epicode.internal)
	TCP	54	443 → 49164 [ACK] Seq=1 Ack=130 Win=64128 Len=0
	TLSv1	1131	Server Hello, Certificate, Server Hello Done

Fase di handshake TLS, in cui il client e il server negoziano la cifratura. Sono visibili i messaggi "Client Hello" e "Server Hello"



Esempio di traffico cifrato (Application data) in un pacchetto HTTPS. Il contenuto è illeggibile a causa della cifratura TLS

Configurazione del sito HTTP su Kali Linux

Disabilitazione del sito HTTPS

Per evitare conflitti, abbiamo disabilitato il sito HTTPS:

```
(kali㉿kali)-[~]
└─$ sudo a2dissite epicode.internal-ssl.conf
[sudo] password for kali:
Site epicode.internal-ssl disabled.
To activate the new configuration, you need to run:
    systemctl reload apache2
```

Disabilitazione del sito HTTPS utilizzando il comando "sudo a2dissite epicode.internal-ssl.conf"

Configurazione del sito HTTP

Abbiamo configurato Apache per ospitare il sito HTTP:

1. Creazione del file di configurazione:

```
(kali@kali)-[~]  
$ sudo nano /etc/apache2/sites-available/epicode.internal.conf
```

```
GNU nano 8.3 /etc/apache2/sites-available/epic  
<VirtualHost *:80>  
    ServerAdmin webmaster@epicode.internal  
    ServerName epicode.internal  
    DocumentRoot /var/www/epicode.internal  
  
    ErrorLog ${APACHE_LOG_DIR}/error.log  
    CustomLog ${APACHE_LOG_DIR}/access.log combined  
</VirtualHost>
```

Creazione e configurazione del file epicode.internal.conf per il sito HTTP

2. Abilitazione del sito HTTP e riavvio di Apache

```
(kali@kali)-[~]  
$ sudo a2ensite epicode.internal.conf  
Site epicode.internal already enabled
```

Abilitazione del sito HTTP utilizzando il comando "sudo a2ensite epicode.internal.conf"

Intercettazione del traffico HTTP con Wireshark

1. Abbiamo applicato un filtro per visualizzare solo il traffico HTTP tra il client 192.168.32.101 e il server 192.168.32.100 sulla porta 80:

```
ip.addr == 192.168.32.100 && ip.addr == 192.168.32.101 && tcp.port == 80
```

Filtro per visualizzare solo il traffico HTTP

2. Generazione del traffico HTTP (sul sistema Windows, abbiamo aperto un browser e visitato <http://epicode.internal/>)



Accesso al sito HTTP da Windows utilizzando il browser

3. Analisi dei pacchetti HTTP (abbiamo fermato la cattura e analizzato i pacchetti HTTP)

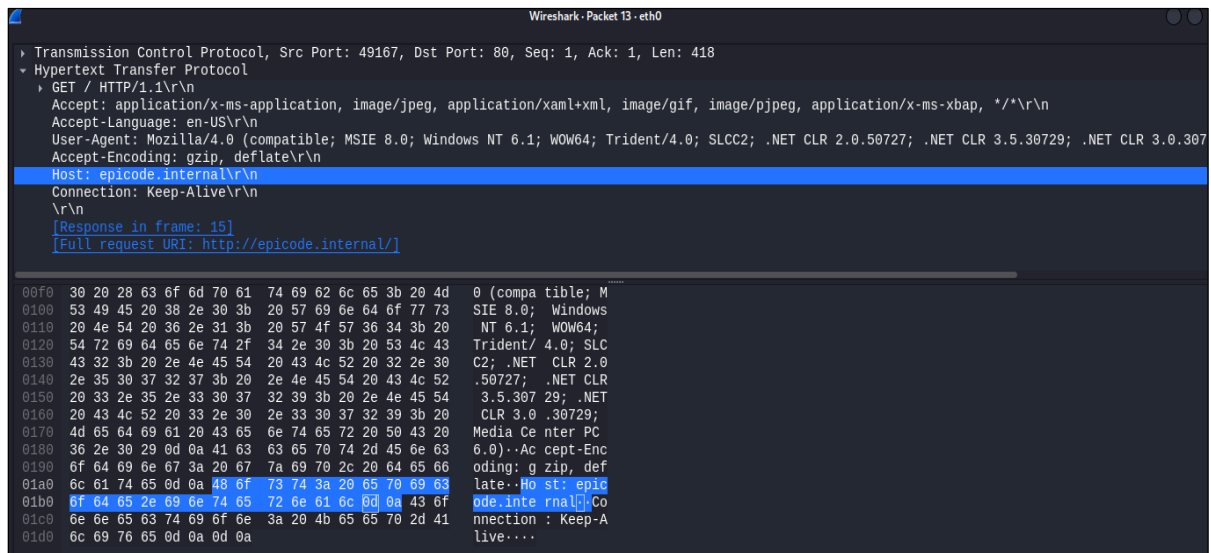
A screenshot of a network packet capture analysis tool. The top pane shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The bottom pane shows a detailed view of a selected packet (Frame 10), including its raw data in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length	Info
10	14.324979942	192.168.32.101	192.168.32.100	TCP	66	49167 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
11	14.325005133	192.168.32.100	192.168.32.101	TCP	66	80 → 49167 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM
12	14.325368224	192.168.32.101	192.168.32.100	TCP	60	49167 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	14.326226053	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
14	14.326267944	192.168.32.100	192.168.32.101	TCP	54	80 → 49167 [ACK] Seq=1 Ack=419 Win=64128 Len=0
15	14.326893955	192.168.32.100	192.168.32.101	HTTP	502	HTTP/1.1 200 OK (text/html)
16	14.356589766	192.168.32.101	192.168.32.100	TCP	66	49168 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
17	14.356614876	192.168.32.100	192.168.32.101	TCP	66	80 → 49168 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM
18	14.357270987	192.168.32.101	192.168.32.100	TCP	60	49168 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
19	14.357271147	192.168.32.101	192.168.32.100	HTTP	348	GET /favicon.ico HTTP/1.1
20	14.357293100	192.168.32.100	192.168.32.101	TCP	54	80 → 49168 [ACK] Seq=1 Ack=295 Win=64128 Len=0
21	14.357832906	192.168.32.100	192.168.32.101	HTTP	549	HTTP/1.1 404 Not Found (text/html)
22	14.540302986	192.168.32.101	192.168.32.100	TCP	60	49167 → 80 [ACK] Seq=419 Ack=449 Win=65252 Len=0
23	14.562884797	192.168.32.101	192.168.32.100	TCP	60	49168 → 80 [ACK] Seq=295 Ack=496 Win=65204 Len=0
24	19.329562475	192.168.32.100	192.168.32.101	TCP	54	80 → 49167 [FIN, ACK] Seq=449 Ack=419 Win=64128 Len=0
25	19.330026392	192.168.32.101	192.168.32.100	TCP	60	49167 → 80 [ACK] Seq=419 Ack=450 Win=65252 Len=0
26	19.363047793	192.168.32.100	192.168.32.101	TCP	54	80 → 49168 [FIN, ACK] Seq=496 Ack=295 Win=64128 Len=0

Frame 10: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: PCSSystemtec_99:2d:aa (08:00:27:99:2d:aa), Dst: PCSSyste
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 49167, Dst Port: 80, Seq: 0, Len:
0000 08 00 27 04 42 0f 08 00 27 99 2d aa 08 00 45 00
0010 00 34 00 c9 40 00 80 06 37 e1 c0 a8 20 65 c0 a8
0020 20 64 c0 0f 00 50 7c f0 a0 c9 00 00 00 00 80 02
0030 20 00 ae e2 00 00 02 04 05 b4 01 03 03 02 01 01
0040 04 02

Il traffico è in chiaro e può essere letto direttamente

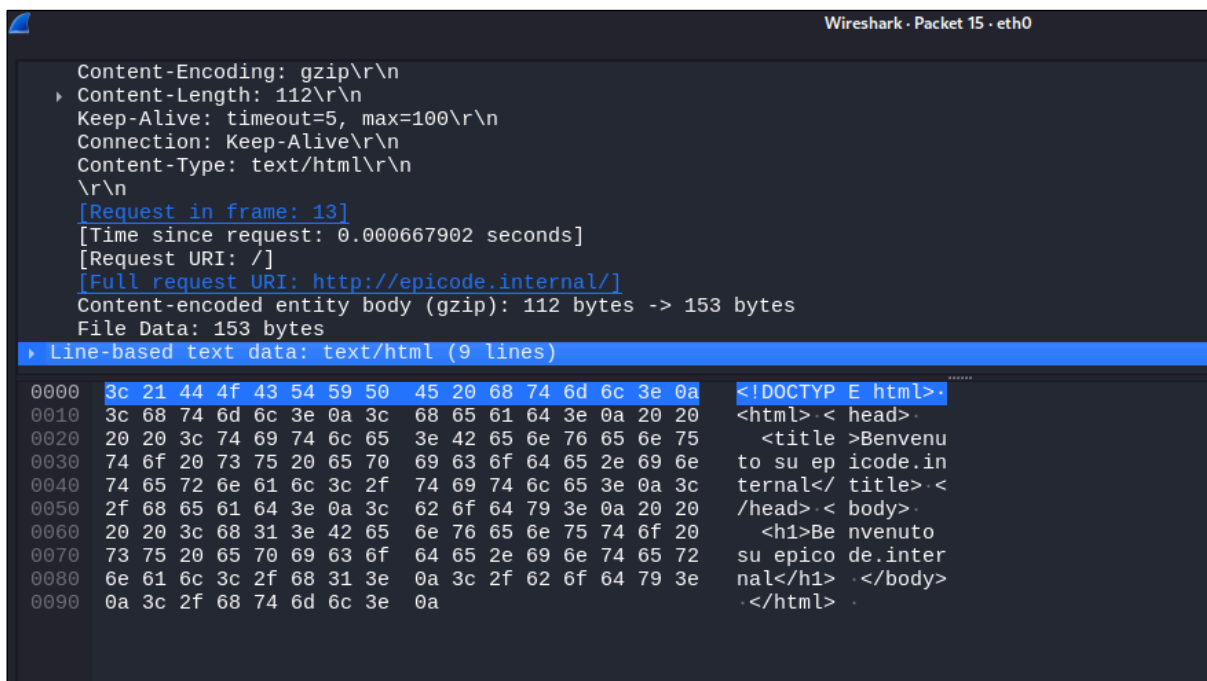
- Contenuto in chiaro delle richieste e risposte HTTP (abbiamo esaminato una richiesta HTTP in chiaro, come una richiesta GET /HTTP/1.1)



The image shows a Wireshark packet capture of an HTTP GET request. The packet list on the left shows 'Transmission Control Protocol, Src Port: 49167, Dst Port: 80, Seq: 1, Ack: 1, Len: 418' and 'Hypertext Transfer Protocol'. The packet details pane shows the following fields: GET / HTTP/1.1, Accept: application/x-ms-application, image/jpeg, application/xml+xml, image/gif, image/pjpeg, application/x-ms-xbap, */*, Accept-Language: en-US, User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729), Accept-Encoding: gzip, deflate, Host: epicode.internal, Connection: Keep-Alive. The packet bytes pane shows the raw data of the request, including the status bar at the bottom: 'Line-based text data: text/html (9 lines)'.

Esempio di una richiesta HTTP in chiaro, con il metodo GET e l'hostname epicode.internal

- Abbiamo esaminato una risposta HTTP in chiaro, come una risposta HTTP/1.1 200 OK con il contenuto della pagina



The image shows a Wireshark packet capture of an HTTP 200 OK response. The packet list on the left shows 'Content-Encoding: gzip' and 'Content-Length: 112'. The packet details pane shows the following fields: Content-Encoding: gzip, Content-Length: 112, Keep-Alive: timeout=5, max=100, Connection: Keep-Alive, Content-Type: text/html, Request in frame: 13, Time since request: 0.000667902 seconds, Request URI: /, Full request URI: http://epicode.internal/, Content-encoded entity body (gzip): 112 bytes -> 153 bytes, File Data: 153 bytes. The packet bytes pane shows the raw data of the response, including the status bar at the bottom: 'Line-based text data: text/html (9 lines)'.

Esempio di una risposta HTTP in chiaro, con codice di stato 200 OK e il contenuto HTML della pagina

Differenze principali tra HTTPS e HTTP

Cifratura:

- HTTPS: il traffico è cifrato, proteggendo i dati da intercettazioni.
- HTTP: il traffico è in chiaro, rendendo i dati vulnerabili.

Porte:

- HTTPS: Utilizza la porta 443.
- HTTP: Utilizza la porta 80.

Sicurezza:

- HTTPS: fornisce autenticazione e integrità dei dati.
- HTTP: non offre protezione contro attacchi man-in-the-middle.

Conclusione

L'analisi del traffico HTTP ha evidenziato l'importanza di utilizzare HTTPS per proteggere le comunicazioni web. Mentre HTTP è semplice e veloce, HTTPS offre sicurezza e privacy, essenziali per proteggere i dati sensibili.