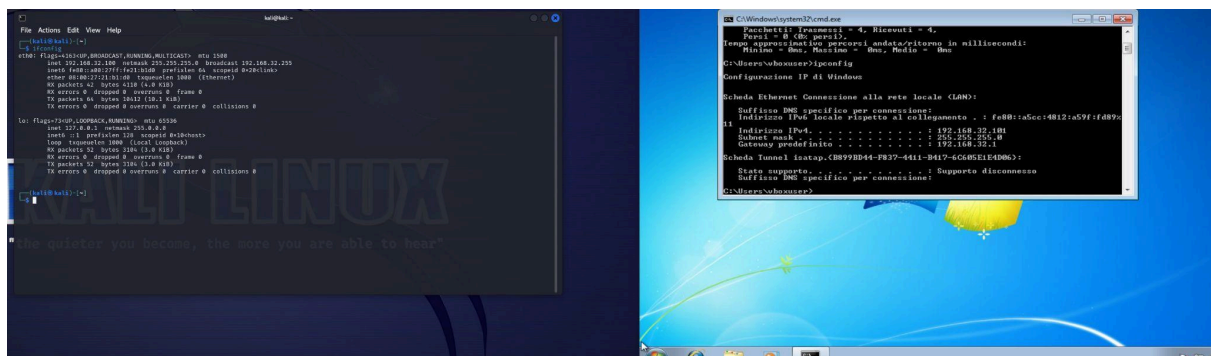


ESERICITAZIONE M1 W4 D4

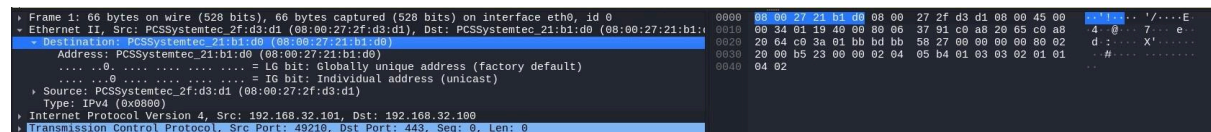
Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows 7) richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100 (Kali). Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS. Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

Creazione del laboratorio con le specifiche richieste:



Evidenziamento del MAC source (08:00:27:2f:d3:d1)

MAC destination (08:00:27:21:b1:d0)



Tramite inetsim andremo a usufruire dei servizi di HTTP,HTTPS e DNS impostando kali come server DNS assegnando lo stesso ip della VM di kali e attivando la fakemode_http. Successivamente andremo a settare il server DNS dicendo di risolvere epicode.internal in 192.168.35.100. Dopo si avvia inetsim e wireshark su kali.

Richiesta tramite Win7 <http://epicode.internal>.

```

1 0.000000000 PCSSystemtec_2f:d3:: Broadcast ARP 60 Who has 192.168.32.100? Tell 192.168.32.101
2 0.000010092 PCSSystemtec_21:b1:: PCSSystemtec_2f:d3:: ARP 42 192.168.32.100 is at 08:00:27:21:b1:d0
3 0.000113800 192.168.32.101 192.168.32.100 TCP 60 49262 - 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4 0.000431346 192.168.32.100 192.168.32.101 TCP 68 80 - 49262 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM WS=128
5 0.000718930 192.168.32.101 192.168.32.100 TCP 60 49262 - 80 [ACK] Seq=1 Ack=1 Win=65790 Len=0
6 0.000781800 192.168.32.101 192.168.32.100 HTTP 361 GET / HTTP/1.1
7 0.000793725 192.168.32.100 192.168.32.101 TCP 54 80 - 49262 [ACK] Seq=1 Ack=368 Win=64128 Len=0
8 0.012939803 192.168.32.100 192.168.32.101 TCP 204 80 - 49262 [FIN, ACK] Seq=1 Ack=368 Win=64128 Len=150 [TCP segment of a reassembled PDU]
9 0.014586955 192.168.32.100 192.168.32.101 HTTP 312 HTTP/1.1 200 OK (text/html)
10 0.015999339 192.168.32.101 192.168.32.100 TCP 60 49262 - 80 [ACK] Seq=368 Ack=410 Win=65292 Len=0
11 0.015999518 192.168.32.100 192.168.32.101 TCP 60 49262 - 80 [FIN, ACK] Seq=368 Ack=410 Win=65292 Len=0
12 0.015114811 192.168.32.100 192.168.32.101 TCP 54 80 - 49262 [ACK] Seq=410 Ack=369 Win=64128 Len=0
13 0.015802545 PCSSystemtec_21:b1:: PCSSystemtec_2f:d3:: ARP 42 Who has 192.168.32.101? Tell 192.168.32.100
14 0.016390909 PCSSystemtec_2f:d3:: PCSSystemtec_21:b1:: ARP 60 192.168.32.101 is at 08:00:27:2f:d3:d1

+ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
+ Ethernet II, Src: PCSSystemtec_2f:d3:d1 (08:00:27:2f:d3:d1), Dst: PCSSystemtec_21:b1:d0 (08:00:27:21:b1:d0)
+ Destination: PCSSystemtec_21:b1:d0 (08:00:27:21:b1:d0)
+ Source: PCSSystemtec_2f:d3:d1 (08:00:27:2f:d3:d1)
+ Type: IPv4 (0x0800)
+ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
+ Transmission Control Protocol, Src Port: 49262, Dst Port: 80, Seq: 0, Len: 0
+ Source Port: 49262
+ Destination Port: 80
+ [Stream index: 0]
+ [Conversation completeness: Complete, WITH_DATA (31)]
+ [TCP Segment Len: 0]
+ Sequence Number: 0 (relative sequence number)
+ Sequence Number (raw): 3040572016
+ [Next Sequence Number: 1 (relative sequence number)]
+ Acknowledgment Number: 0
+ Acknowledgment number (raw): 0
+ 1000 .... = Header Length: 32 bytes (8)
+ [Flags: 0x0002 (SYN)]
+ Window: 8192
+ [Calculated window size: 8192]
+ Checksum: 0xa491 [unverified]
+ [Checksum Status: Unverified]
+ Urgent Pointer: 0
+ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Op
+ [Timestamps:

```

Possiamo notare come avviene l'instaurazione del collegamento tra il fake server e il browser di win7.

Il primo protocollo troviamo l'ARP, collega un indirizzo IP in continua evoluzione a un indirizzo fisso del computer fisico chiamato MAC. Essendo la chiave dell'ARP ha necessità di sapere quale MAC corrisponde a quell'IP, così da non ripetere molto spesso questa procedura. Successivamente avviene l'instaurazione del TCP che va a chiedere una connessione nelle porte richieste. Si può notare l'avvio della richiesta della connessione alle porte e l'instaurazione del protocollo SYN SYN ACK. Dopo essere andato a buon fine inizia la trasmissione del HTTP e infine notiamo che il 192.168.32.101 richiede al TCP il FIN, ACK per la chiusura della connessione.

Richiesta tramite WIN7 <https://epicode.internal>

The screenshot displays a Wireshark network traffic analysis. The main focus is on a TLS Handshake packet (ClientHello) captured on the eth0 interface. The packet list on the left shows various network protocols, with the selected packet being a TLS Handshake. The packet details pane on the right provides a structured view of the ClientHello message, including the Magic Number, Version, Session ID, Cipher Suites, and Compression Methods. The raw packet data pane on the far right shows the hexadecimal and ASCII representation of the packet bytes.

Instaurazione del collegamento avviene come nel precedente caso dell'http fino alla conferma del canale di comunicazione con il protocollo TCP e l'avvio del SYN SYN ACK, ma successivamente notiamo l'instaurazione del TLS.

Il funzionamento del protocollo TLS può essere suddiviso in tre fasi principali

- Negoziante fra le parti dell' algoritmo da utilizzare
- Scambio delle chiavi di autenticazione
- Cifratura simmetrica e autenticazione dei messaggi

```
Type: IPv4 (0x0000)
+ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
+ Transmission Control Protocol, Src Port: 49210, Dst Port: 443, Seq: 1, Ack: 1, Len: 161
+ Transport Layer Security
- TLV Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 156
+ Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 152
Version: TLS 1.0 (0x0301)
Random: 65f4a1e4df8d8e96e5df6467de0b1b4b7d6e6d93f8ef646722083e31997cfc6
Session ID Length: 32
Session ID: 9ed9bc13cddc96183b958a91c6b87278a5248af5894a80c0cc568302e3adaaec
Cipher Suites Length: 24
Cipher Suites (12 suites)
Compression Methods Length: 1
Compression Methods (1 method)
Extensions Length: 55
Extension: renegotiation_info (len=1)
Extension: server_name (len=21) name=epicode.internal
Extension: status_request (len=5)
Extension: supported_groups (len=6)
Extension: ec_point_formats (len=2)
[J44: t10d120508_d94e65cd8899_c35b4a14be45]
[J44 r: t10d120508_0004_0005_000a_0013_002f_0032_0035_0038_c009_c00a_c013_c014_0005_000a_000b_ff0]
[J43 Fullstring: 709,47-53-5-10-49171-49172-49161-49162-50-50-19-4,05281-0-5-10-11,23-24,0]
[J43: 2201d8e086f8f005a0b415f61e077532]
```

Nella prima fase, il client e il server negoziano il protocollo di cifratura che sarà utilizzato nella comunicazione sicura, il protocollo per lo scambio delle chiavi e l'algoritmo di autenticazione nonché il Message authentication code (MAC). Si nota come tutto il layer applicativo sia sotto protocollo LSV e quindi crittografato. La chiusura avverrà sempre sotto richiesta del client (VM win7) che richiederà un FIN ACK tramite protocollo TCP e successiva risposta di ACK del server (kali).