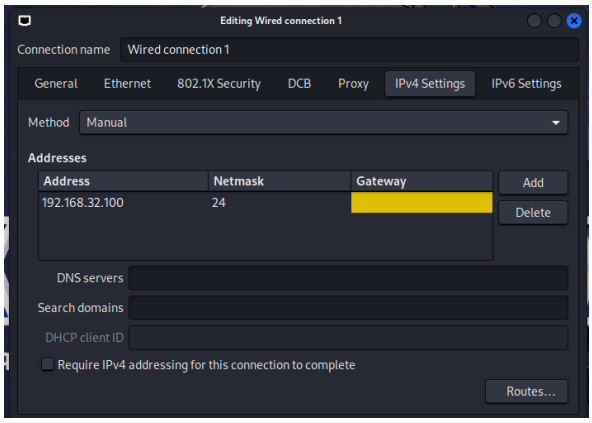
Report: simulazione rete complessa

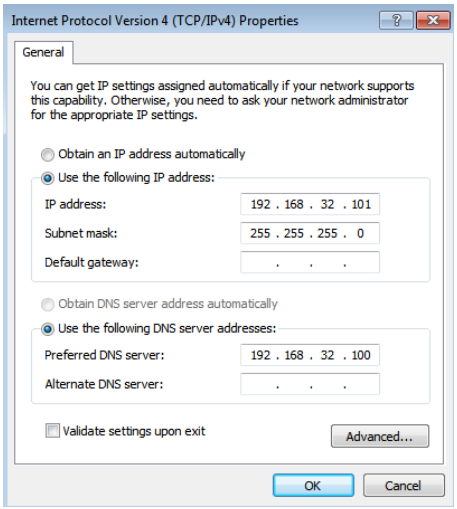
Unit 1 Week 1

Come prima operazione, dopo aver controllato sulla macchina Kali la presenza di InetSim (già preistallata), ho sistemato gli indirizzi Ip richiesti sulle due macchine e impostato su Windows, l'indirizzo del DNS server che configureremo a breve. Ecco i passaggi:

Kali linux: Edit connection > IPv4 settings> modifichiamo su manuale ed andiamo ad aggiungere l'indirizzo ip

Windows 7: Start > Control Panel > Network and Internet > Network and Sharing Center> change adapter settings > click destro su Local Area Connection > properties > IPv4 properties e configurazione di ip e DNS server sul quale mandare richieste.

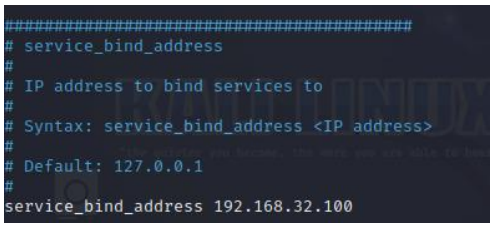


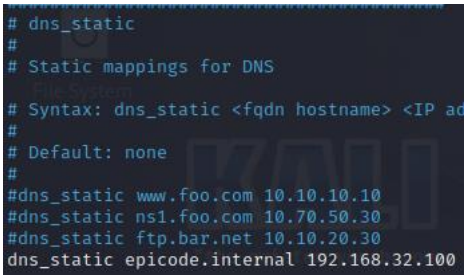


Successivamente ho acceso la macchina Linux, aperto la shell e sono andato a tirar su i server DNS e HTTPS con InetSim, utilizzando il seguendo comando per entrare nella configurazione dell'applicazione che funge da server:

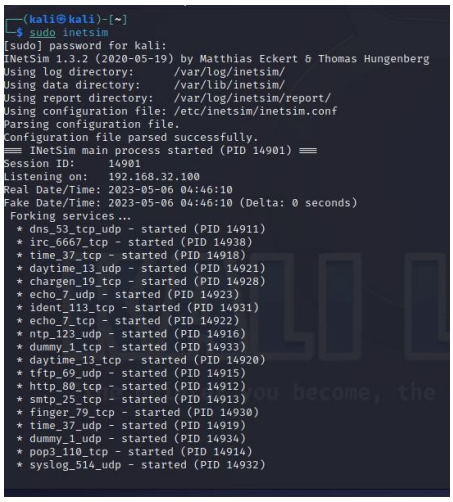
sudo nano /etc/inetsim/inetsim.conf

Scorrendo giù, ho settato l'indirizzo ip a cui associare il servizio, macchina kali dalla quale stiamo facendo l'operazione. Poi sono andato alla ricerca del server DNS sul quale impostare l' hostname collegato all'indirizzo del server http, che è lo stesso del DNS. Le porte sono aperte di default e il servizio http è già avviato.

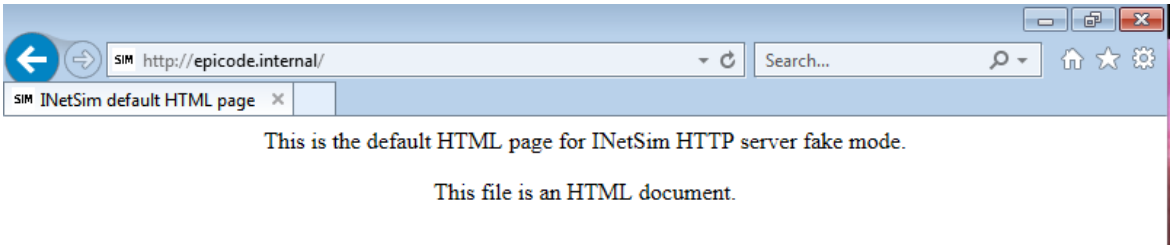




Così facendo, avviando InetSim su kali



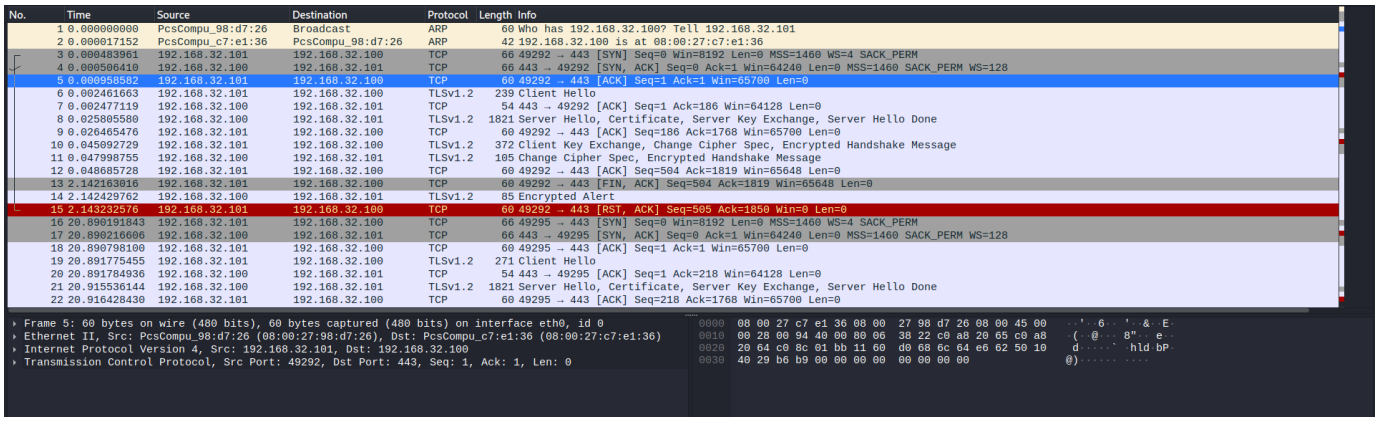
 e richiedendo dal Web browser della macchina windows “epicode.internal” , mi torna indietro ciò che ho richiesto, il servizio http, attraverso il server DNS che converte il dominio in indirizzo IP e invia la richiesta al server http.



Completati i primi step, grazie all’utillizzo di Wireshark, andremo ad analizzare qualsiasi pacchetto, flusso di traffico o connessione che passa attraverso alla scheda di rete. Su kali, Wireshark è già presente, non ci resta che avviarla e settare la scheda di rete su eth0.

Visto che il server è già funzionale, prima di cercare epicode.internal , avviamo la cattura di wireshark, così da poter analizzare qualsiasi pacchetto o flusso di traffico che passa attraverso la rete. La prima richiesta sarà fatta in HTTPS.

RICHIESTA IN HTTPS



Analizziamo i pacchetti intercettati. Come primi pacchetti, c’è il protocollo arp, che chiede in broadcast (tutti i dispositivi in rete) qual è il MAC associato all’indirizzo ip.

Chi ha indirizzo ip 192.168.32.100 (kali)? Chiede 192.168.32.101 (windows)

192.168.32.100 risponde e comunica il proprio indirizzo MAC, in questo caso 08:00:27:c7:e1:36, attraverso il protocollo ARP.

Successivamente TCP, che è un protocollo a livello di trasporto, cerca di instaurare un canale di comunicazione tra gli Host utilizzando le porte. (three-way-handshake)

In questo caso c'è una richiesta HTTPS, la porta di default è 443, mentre la porta del client è scelta casualmente dal sistema operativo, in questo caso 49292.

Il client invia un pacchetto TCP al server destinatario con il flag SYN abilitato ed un numero di sequenza casuale (pacchetto n 3)

Il server risponde inviando al client un pacchetto con i flag SYN e ACK abilitati e a sua volta il client risponde, instaurando così un canale di comunicazione. (pacchetto n 4 e 5).

Visto l'utilizzo dell'HTTPS, interviene il protocollo TLS, protocollo crittografico di presentazione, dove nello scambio di pacchetti n8, avviene lo scambio delle chiavi del server, dopo che il server risponde al "client hello" (pacchetto n6) con " server hello" e "server hello done" dopo lo scambio chiavi del server.

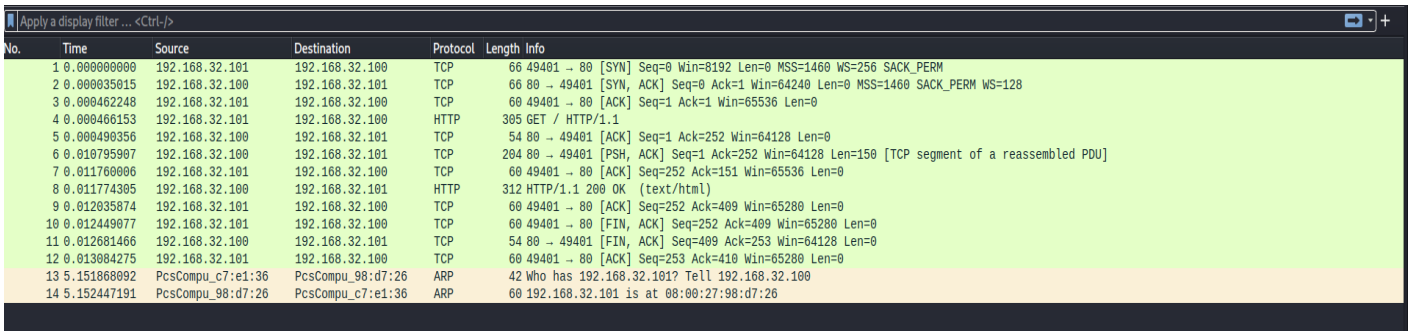
Successivamente (pacchetto n10 e 11), c'è lo scambio di chiavi del client, e con "change cipher spec" ed "Encrypted Handshake Message", si segnala il passaggio ad una nuova configurazione di cifratura e per scambiare informazioni cifrate con la nuova chiave di sessione condivisa.

Dopo questo scambio di informazioni, al pacchetto n14 c'è un messaggio di alert, che segnala un errore.

Come ultimo pacchetto(n15) infatti si interrompe la comunicazione TCP, e la linea di pacchetto diventa rossa. Successivamente le macchine ritentano la connessione con lo stesso risultato.

Infatti quando, su web browser effettuiamo la ricerca, c'è in rosso un "errore di certificato", in questo caso, l'https effettuando questi controlli, da una sicurezza che l'hhtp non può assicurare .

RICHIESTA HTTP

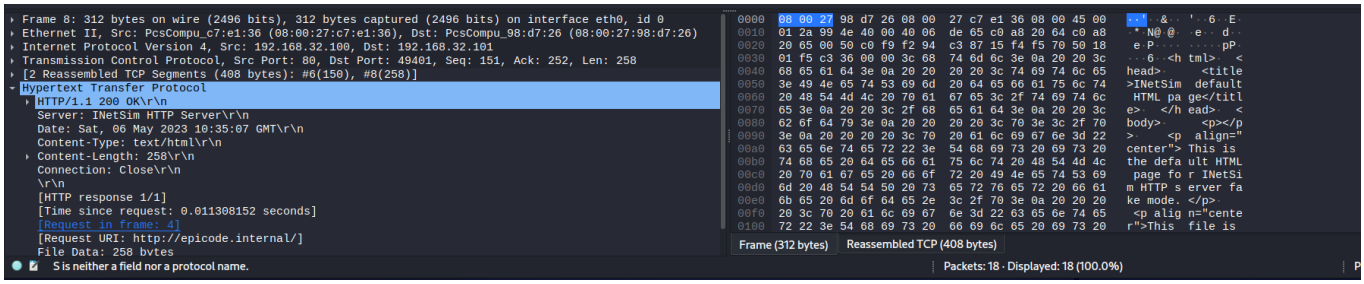

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	66	49401 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000035015	192.168.32.100	192.168.32.101	TCP	66	80 → 49401 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.000462248	192.168.32.101	192.168.32.100	TCP	60	49401 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
4	0.000466153	192.168.32.101	192.168.32.100	HTTP	305	GET / HTTP/1.1
5	0.000490356	192.168.32.100	192.168.32.101	TCP	54	80 → 49401 [ACK] Seq=1 Ack=252 Win=64128 Len=0
6	0.010795907	192.168.32.100	192.168.32.101	TCP	204	80 → 49401 [PSH, ACK] Seq=1 Ack=252 Win=64128 Len=150 [TCP segment of a reassembled PDU]
7	0.011760006	192.168.32.101	192.168.32.100	TCP	60	49401 → 80 [ACK] Seq=252 Ack=151 Win=65536 Len=0
8	0.011774305	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
9	0.012035874	192.168.32.101	192.168.32.100	TCP	60	49401 → 80 [ACK] Seq=252 Ack=409 Win=65280 Len=0
10	0.012449077	192.168.32.101	192.168.32.100	TCP	60	49401 → 80 [FIN, ACK] Seq=252 Ack=409 Win=65280 Len=0
11	0.012681466	192.168.32.100	192.168.32.101	TCP	54	80 → 49401 [FIN, ACK] Seq=409 Ack=253 Win=64128 Len=0
12	0.013084275	192.168.32.101	192.168.32.100	TCP	60	49401 → 80 [ACK] Seq=253 Ack=410 Win=65280 Len=0
13	5.151868092	PcsCompu_c7:e1:36	PcsCompu_98:d7:26	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
14	5.152447191	PcsCompu_98:d7:26	PcsCompu_c7:e1:36	ARP	60	192.168.32.101 is at 08:00:27:98:d7:26

In questa richiesta, dopo aver instaurato un canale di comunicazione con il protocollo TCP, con i vari scambi di Sync e ACK come visti nella richiesta HTTPS (pacchetti n 1,2,3) . In questo caso la comunicazione avviene col servizio della porta 80 (porta di default del protocollo http). Successivamente c'è la richiesta di GET al servizio http, dove il client (192.168.32.101) chiede al server la pagina richiesta (pacchetto n 4).

Al pacchetto n8, possiamo vedere (200 ok) che indica che la richiesta HTTP del client è stata elaborata con successo dal server e che il server sta restituendo il contenuto richiesto dal client.

Come vediamo nella prossima foto, il contenuto è senza crittografia, visto che il servizio http trasmette tutte le informazioni in chiaro senza criptazione.



DIFFERENZE HTTPS E http

La differenza di maggior rilevanza, sta nella crittografia, assente nel servizio http.

Effettuando ricerche in HTTPS, tutte le informazioni sono al sicuro e sono crittografate, in questo caso con TLS che provvede con lo scambio delle chiavi lato server e client. Come visto in http, ciò è assente e tutti i dati vengono scambiati in chiaro e un malintenzionato, seguendo procedure di hacking può intercettare le informazioni. Come visto in precedenza la porta di connessione di http è 80, https 443 (di default).

Per utilizzare HTTPS, il sito web deve avere un certificato SSL/TLS valido installato sul server. Nel nostro caso c'è un errore del certificato di sicurezza ed è per questo che segnala un errore, e per proseguire alla pagina richiesta, bisogna forzare l'ingresso anche se non raccomandato.

le prestazioni di HTTPS sono leggermente inferiori rispetto ad http, come visto anche nel time di invio pacchetti.

Quindi HTTPS offre una maggiore sicurezza rispetto ad HTTP grazie alla crittografia SSL/TLS, ma richiede un certificato valido e può essere leggermente più lento in termini di prestazioni.