

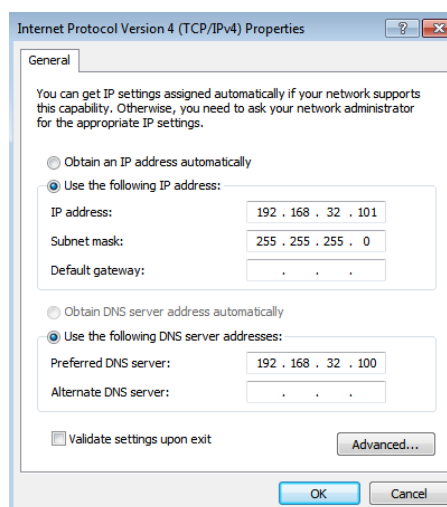
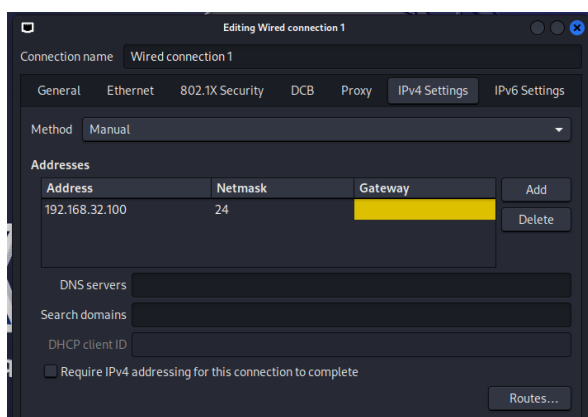
Report: simulazione rete complessa

Unit 1 Week 1

Come prima operazione, dopo aver controllato sulla macchina Kali la presenza di InetSim (già preinstallata), 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 seguente 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.

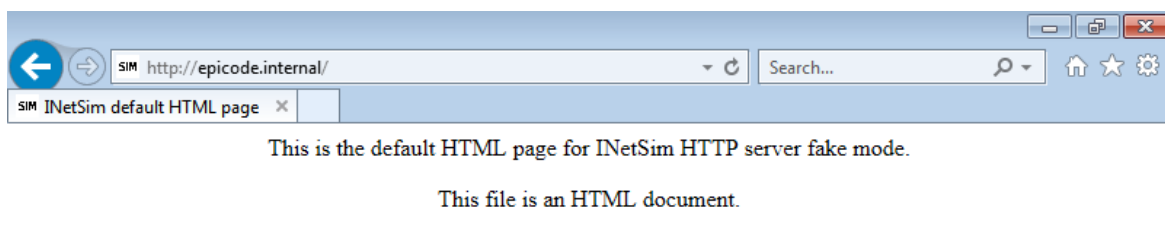
```
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100
```

```
#####
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
#dns_static www.foo.com 10.10.10.10
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
#dns_static epicode.internal 192.168.32.100
```

Così facendo, avviando InetSim su kali

```
(kali@kali)-[~]
└─$ sudo inetSim
[sudo] password for kali:
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetSim/
Using data directory: /var/lib/inetSim/
Using report directory: /var/log/inetSim/report/
Using configuration file: /etc/inetSim/inetSim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 14901) ==
Session ID: 14901
Listening on: 192.168.32.100
Real Date/Time: 2023-05-06 04:46:10
Fake Date/Time: 2023-05-06 04:46:10 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 14911)
* irc_6667_tcp - started (PID 14938)
* time_37_tcp - started (PID 14918)
* daytime_13_udp - started (PID 14921)
* chargen_19_tcp - started (PID 14928)
* echo_7_udp - started (PID 14923)
* ident_113_tcp - started (PID 14931)
* echo_7_tcp - started (PID 14922)
* ntp_123_udp - started (PID 14916)
* dummy_1_tcp - started (PID 14933)
* daytime_13_tcp - started (PID 14920)
* tftp_69_udp - started (PID 14915)
* http_80_tcp - started (PID 14912)
* smtp_25_tcp - started (PID 14913)
* finger_79_tcp - started (PID 14930)
* time_37_udp - started (PID 14919)
* dummy_1_udp - started (PID 14934)
* pop3_110_tcp - started (PID 14914)
* syslog_514_udp - started (PID 14932)
```

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’utilizzo 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

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-------------------|-------------------|----------|--------|--|
| 1 | 0.000000000 | PcsCompu_98:d7:26 | Broadcast | ARP | 60 | Who has 192.168.32.100? Tell 192.168.32.101 |
| 2 | 0.000017152 | PcsCompu_c7:e1:36 | PcsCompu_98:d7:26 | ARP | 42 | 192.168.32.100 is at 08:00:27:c7:e1:36 |
| 3 | 0.000483961 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49292 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM |
| 4 | 0.000596418 | 192.168.32.100 | 192.168.32.101 | TCP | 66 | 443 → 49292 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 5 | 0.000605532 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49292 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0 |
| 6 | 0.002461663 | 192.168.32.101 | 192.168.32.100 | TLSv1.2 | 239 | Client Hello |
| 7 | 0.002477119 | 192.168.32.100 | 192.168.32.101 | TCP | 54 | 443 → 49292 [ACK] Seq=1 Ack=186 Win=64128 Len=0 |
| 8 | 0.025805580 | 192.168.32.100 | 192.168.32.101 | TLSv1.2 | 1821 | Server Hello, Certificate, Server Key Exchange, Server Hello Done |
| 9 | 0.026465476 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49292 → 443 [ACK] Seq=186 Ack=1768 Win=65700 Len=0 |
| 10 | 0.045992729 | 192.168.32.101 | 192.168.32.100 | TLSv1.2 | 372 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message |
| 11 | 0.047998755 | 192.168.32.100 | 192.168.32.101 | TLSv1.2 | 105 | Change Cipher Spec, Encrypted Handshake Message |
| 12 | 0.048685728 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49292 → 443 [ACK] Seq=504 Ack=1819 Win=65648 Len=0 |
| 13 | 2.142163816 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49292 → 443 [FIN, ACK] Seq=504 Ack=1819 Win=65648 Len=0 |
| 14 | 2.142429762 | 192.168.32.100 | 192.168.32.101 | TLSv1.2 | 85 | Encrypted Alert |
| 15 | 2.142432276 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49292 → 443 [RST, ACK] Seq=505 Ack=1850 Win=0 Len=0 |
| 16 | 20.890191843 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49295 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM |
| 17 | 20.890216606 | 192.168.32.100 | 192.168.32.101 | TCP | 66 | 443 → 49295 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 18 | 20.890798100 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49295 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0 |
| 19 | 20.891775455 | 192.168.32.101 | 192.168.32.100 | TLSv1.2 | 271 | Client Hello |
| 20 | 20.891784936 | 192.168.32.100 | 192.168.32.101 | TCP | 54 | 443 → 49295 [ACK] Seq=1 Ack=218 Win=64128 Len=0 |
| 21 | 20.915536144 | 192.168.32.100 | 192.168.32.101 | TLSv1.2 | 1821 | Server Hello, Certificate, Server Key Exchange, Server Hello Done |
| 22 | 20.916428430 | 192.168.32.101 | 192.168.32.100 | TCP | 66 | 49295 → 443 [ACK] Seq=218 Ack=1768 Win=65700 Len=0 |

Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_98:d7:26 (08:00:27:98:d7:26), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
Transmission Control Protocol, Src Port: 49292, Dst Port: 443, Seq: 1, Ack: 1, Len: 0

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'http non può assicurare .

RICHIESTA HTTP

| 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 | 0.012035074 | 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_c7: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 crittazione.

```
Frame 8: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36), Dst: PcsCompu_98:d7:26 (08:00:27:98:d7:26)
Internet Protocol Version 4, Src: 192.168.32.180, Dst: 192.168.32.191
Transmission Control Protocol, Src Port: 80, Dst Port: 49481, Seq: 151, Ack: 252, Len: 258
[2 Reassembled TCP Segments (408 bytes): #6(150), #8(258)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Server: INetSim HTTP Server\r\n
    Date: Sat, 06 May 2023 10:35:07 GMT\r\n
    Content-Type: text/html\r\n
    Content-Length: 258\r\n
    Connection: Close\r\n
  \r\n
[HTTP response 1/1]
[Time since request: 0.011308152 seconds]
[Request in frame: 4]
[Request URI: http://epicode.internal/]
File data: 258 bytes
S is neither a field nor a protocol name.
0000  08 00 27 98 d7 26 08 00 27 c7 e1 36 08 00 45 00  &... 6 E:
0010  01 2a 99 4e 40 00 40 06 de 65 c0 a8 20 64 c0 a8  *N@.@... d...
0020  20 65 00 50 c0 f9 f2 94 c3 87 15 f4 f5 70 50 18  e P... .. pP
0030  01 f5 c3 30 00 00 3c 69 74 6d 6c 3e 0a 20 20 3c  6...<html> <
0040  68 65 61 64 3e 0a 20 20 20 20 3c 74 69 74 6c 65  head> <title
0050  3e 49 4e 65 74 53 69 6d 20 64 65 66 61 75 6c 74  >INetSim default
0060  20 48 54 4d 4c 20 79 61 67 65 3c 2f 74 69 74 6c  HTML page<titl
0070  65 3e 0a 20 20 3c 2f 68 65 61 64 3e 0a 20 20 3c  e> </head> <
0080  62 6f 64 79 3e 0a 20 20 20 20 3c 70 3e 3c 2f 70  body> <p><p
0090  3e 0a 20 20 20 20 3c 70 20 61 6c 69 67 6e 3d 22  > <p align="
00a0  63 65 6e 74 65 72 22 3e 54 68 69 73 20 69 73 20  center"> This is
00b0  74 68 65 20 64 65 66 61 75 6c 74 20 48 54 4d 4c  the default HTML
00c0  20 70 61 67 65 20 66 6f 72 20 49 4e 65 74 53 69  page for INetSi
00d0  6d 20 48 54 54 50 20 73 65 72 76 65 72 20 66 61  m HTTP server fa
00e0  6b 65 20 6d 6f 64 65 2e 3c 2f 70 3e 0a 20 20 20  ke mode. </p>
00f0  20 3c 70 20 61 6c 69 67 6e 3d 22 63 65 6e 74 65  <p align="cente
0100  72 22 3e 54 68 69 73 20 66 69 6c 65 20 69 73 20  r">This file is
Frame (312 bytes) Reassembled TCP (408 bytes)
Packets: 18 - Displayed: 18 (100.0%)
```

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.