Módulo 3 - Sesión Laboratorio

Nombre: Univ. Mamani Chavez Carla Vanesa	**CI:** 9124602 LP **Paralelo:** Martes
Docente: Lic. Gallardo Portanda Franz Ramiro	**Fecha :** 28/06/2020

1. **Analisis de trafico TCP**

   a) **En su máquina active un servidor Web, podemos usar Python para este propósito ejecutando el siguiente comando en la consola: python3 –m http.server 9000**

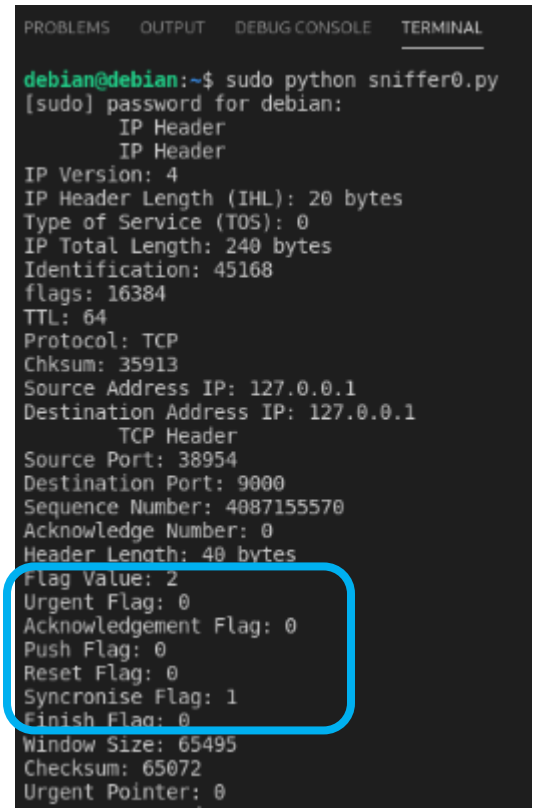
   Una vez iniciado ejecutamos el comando

   python3 –m http.server 9000

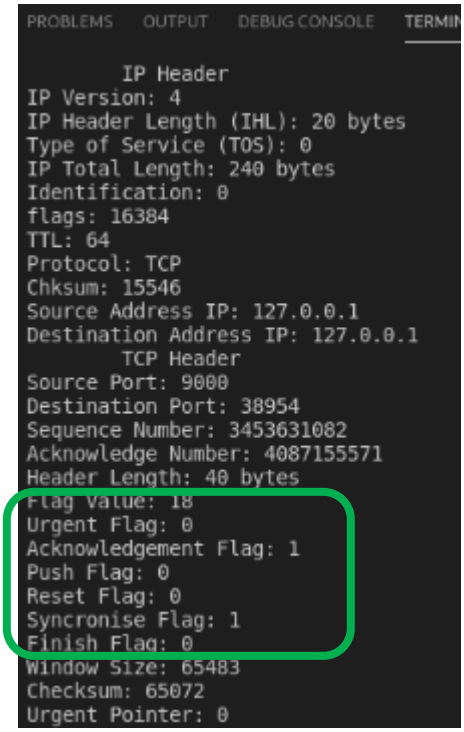
   El comando iniciara un servidor que escuche peticiones en el puerto 9000, a continuación, ejecutemos en el navegador y solicite un archivo al servidor. Con la página o el archivo desplegado en su navegador, habremos capturado suficiente tráfico para el análisis.

   b) **Analice la información que arroja su programa e identifique los paquetes intercambiados para el establecimiento de la conexión TCP entre el servidor y el navegador. Recuerde que en TCP, el establecimiento de la conexión suele denominarse: proceso de acuerdo en tres, y que hace referencia al intercambio de tres paquetes específicos para establecer los parámetros de la conexión. Para estos tres paquetes proporcione toda la información de capa 3 y capa 4 que proporciona su programa.**

1er camino, Flag Syn activado (Origen: PC – Destino: 9000)



2do camino, Flags Ack - Syn activado (Origen: 9000 – Destino: PC)



3er camino, Flag Ack activado (Origen: PC – Destino: 9000)

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERM

Checksum: 65072
Urgent Pointer: 0
        IP Header
IP Version: 4
IP Header Length (IHL): 20 bytes
Type of Service (TOS): 0
IP Total Length: 208 bytes
Identification: 45169
flags: 16384
TTL: 64
Protocol: TCP
Chksum: 35920
Source Address IP: 127.0.0.1
Destination Address IP: 127.0.0.1
        TCP Header
Source Port: 38954
Destination Port: 9000
Sequence Number: 4087155571
Acknowledge Number: 3453631083
Header Length: 32 bytes
Flag Value: 16
Urgent Flag: 0
Acknowledgement Flag: 1
Push Flag: 0
Reset Flag: 0
Syncronise Flag: 0
Finish Flag: 0
Window Size: 512
Checksum: 65064
Urgent Pointer: 0
```

Donde:

```
159     #Entonces, tenemos que multiplicar el IHL por 4 para obtener el tamaño del encabezado en bytes
160     iph_length = IHL * 4
161     tcp_header = data[iph_length:iph_length+20]
162
163     #ahora descomprimirlos
164     tcph = struct.unpack('!HHLLBBHHH' , tcp_header)
165
166     source_port = tcph[0]   # uint16_t
167     dest_port = 9000     # uint16_t
168     sequence = tcph[2]      # uint32_t
169     acknowledgement = tcph[3]   # uint32_t
170     doff_reserved = tcph[4]     # uint8_t
171     tcph_length = doff_reserved >> 4
172
173     tcph_flags = tcph[5]            #uint8_t
174     tcph_window_size = tcph[6]      #uint16_t
175     tcph_checksum = tcph[7]         #uint16_t
176     tcph_urgent_pointer = tcph[8]   #uint16_t
177
178     print("\tTCP Header")
179
180     print("Source Port: ",source_port)
181     print("Destination Port: ",dest_port)
182     print("Sequence Number: ",sequence)
183     print("Acknowledge Number: ",acknowledgement)
184     print("Header Length: ",tcph_length,'DWORDS or ',str(tcph_length*32//8) ,'bytes')
185     print("Flag Value: " + str(tcph_flags))
186     banderas(tcph_flags)
187     print("Urgent Flag: ",tcph_urgent_pointer)
188     print("Window Size:",tcph_window_size)
189     print("Checksum:",tcph_checksum)
```

Source Port: Identifica el número de puerto de un programa de aplicación de origen.

Destination Port: Identifica el número de puerto de un programa de aplicación de destino.

Sequence Number: Especifica el número de secuencia del primer byte de datos de este segmento.

Acknowledgment Number: Identifica la posición del byte más alto recibido.
Doff reserved: Reservado para uso futuro.
Windows Size: Especifica la cantidad de datos que el destino está dispuesto a aceptar.
Checksum: Verifica la integridad de la cabecera y los datos de segmento.
Urgent Pointer: Indica datos que se deben entregar lo más rápidamente posible. Este puntero especifica la posición donde finalizan los datos urgentes.

Code Bits de control para identificar la finalidad del segmento:	
URG	El campo de puntero urgente es válido.
ACK	El campo de reconocimiento es válido.
PSH	El segmento solicita un PUSH.
RTS	Restablece la conexión.
SYN	Sincroniza los números de secuencia.
FIN	El remitente ha alcanzado el final de la corriente de bytes.

2. Conclusión

Construimos un sniffer de la primera tarea que interpretaba los campos de la cabecera IP asi generando el trafico local o conectado a la red,
Ahora extendiendo las capacidades analizaremos el trafico TCP. Donde adquirimos e identificamos los paquetes establecidos por la conexión TCP todo esto con la ayuda de BaseSniffer que contuviese los mecanismos en común entre los tipos de captura para procesar los paquetes que se capturen, cada uno con diferentes características.
Así mismo las banderas (Flags) son las encargadas de especificar los diferentes estados de la comunicación. También validan los valores de los distintos campos de la cabecera de control. Puede haber simultáneamente varios flags activados.