Informe Trabajo Práctico Manejo y Análisis de trama :

Agustín Samperi

Para resolver los problemas que se nos plantearon en el documento tuve que llevar acabo varios algoritmos, primero que nada el algoritmo principal que nos ayuda a dividir cada trama, este algoritmo funciona buscando las banderas que en nuestro caso fueron "7E" pero teníamos que tener en cuenta las secuencia de escape que tuvieran "7D" antes que nuestro "7E", lo cual tenían que estar juntos para que nos diera a entender que el siguiente byte era dato y no una bandera, entonces primero que nada buscaba con la función .find de python los "7E" para entender donde empezaba y donde terminaba la trama, pero tenia que tener en cuenta lo anterior dicho de las secuencia de escape, entonces cuando queríamos saber donde terminaba (o también se puede decir empezaba la siguiente trama) teníamos que tener en cuenta que no hubiera un '7D' al lado de nuestro "7E" si no tenia que pasar al siguiente "7E" para que sea el comienzo la de siguiente trama.

A partir de resolver esto los otros algoritmos fueron un poco mas fácil, ya que en la longitud teníamos que tener en cuenta el valor decimal de los siguiente 2 byte después de la bandera, y cortar en "fragmentos" nuestro string,(se corta dejando solo la carga útil, es decir, bandera, checksum, longitud, son "eliminados" para poder hacer esto) analizar si no tienen secuencia de escape porque en la longitud las secuencias de escape como "7D7E" solo se tiene que contar el "7E" quedaría como que "7D" es 0, entonces al tener esto en mente podíamos hacer un calculo aritmético donde el tamaño que nos dicen los dos primeros byte tiene que conseguir con la longitud de nuestros caracteres divido por 2 ya que cada 2 caracteres teníamos 1 byte, por lo tanto quedaba de la manera : longitud que nos dice los dos byte == (longitud de los caracteres / 2) gracias a esta condición lógica, podíamos ver cuales tramas tenían longitud correcta y cuales no.

Después de esto nos pedían analizar el checksum de los que tenían la longitud correcta, para esto emplee un algoritmo el cual también cortaba el string en partes, o "fragmentos" y analizaba la carga útil, cuando fui a analizar la carga útil tuve un problema porque tenia que ir juntando dos caracteres, lo resolví empleando un algoritmo (que no se si esta empleado de una forma correcta pero que es funcional), al hacer la suma de todos los caracteres hacia la diferencia AND bit a bit entre 0xFF (que es 256) y el checksum que yo saque de la suma de mis caracteres, después hacia la dif de la suma con el carácter 0xFF que esto me daba el checksum de la trama, y lo que resultaba de este algoritmo lo usaba en una una comparación lógica entre el checksum en decimal y la checksum que saque de la trama que saque transformado a decimal, si eran iguales entonces estaba correcto el checksum.
Y el último algoritmo fue el de la secuencias de escape, lo cual lo único que analizaba era en las tramas correctas ver cuales tenían "7D7E" en su interior, ya que encontrado esto sabemos que es una secuencia de escape.

Instrucciones para usar el programa

Primero que nada tenes que tener descargado python3, si no lo tenes instalado podes usar uno de los siguientes comandos dependiendo tu SO

Linux : (python-pip no es necesario pero si lo recomiendo)

- Arch : `sudo pacman -S python python-pip`
- Ubuntu : `sudo apt install python3 python-pip`
- Fedora : `sudo dnf install python3 python-pip`

Windows :

- Para windows hay que descargar el .exe desde https://www.python.org/downloads/windows/ , y a partir de ahí hacer la instalación,
  Una vez tenemos instalado python en nuestra computadora, tenemos que descargar el frame_analyzer.py , .log que contiene la trama, y guardarlos en el mismo directorio (carpeta/archivo), entonces desde nuestra terminal nos vamos hacia ese directorio y solamente con teclear el siguiente comando "" python frame_analyzer.py"" ya tendría que estar en funcionamiento nuestro programa que analiza las tramas

SALIDA :

python frame_analyzer.py
==========
Salida de función que cuenta la longitud, en la salida muestra el indice y la trama que es incorrecta
==========
594 7E0011920013A200403A3BF8063441010000001020983 1358 7E0013920013A200403A3BF806344101000003020A020777
1828 7E0013920013A200403A3BF806344101000003020A020F6F
2369 7E0015920013A200403A3BF8063441010000030209020 37C
Tramas Totales: 3237
Tramas con longitud correcta: 3233
Tramas con longitud incorrecta: 4
==========

Salida de función que cuenta los checksum, la salida muestra los incorrectos:

==========

0 7E001117010013A200403A3BF8FFFE0244303032B1

486 7E0012920013A200403A3BF806344101000001020B82

755 7E0012920013A200403A3BF80634410100000102107B

1202 7E0012920013A200403A3BF8063441010000001020982

1600 7E0014920013A200403A3BF80634410100000030212020096C

1923 7E0014920013A200403A3BF8063441010000003020B020874

2238 7E0014920013A200403A3BF80634410100000030209020047A

2484 7E0014920013A200403A3BF80634410100000030200C020972

3109 7E0014920013A200403A3BF80634410100000030205020087A

CheckSum que son correctos : 3224

CheckSum que son incorrectos : 9

==========

Salida función que cuenta las tramas con Secuencia de Escape:

==========

3 7E001117010013A200403A3BF8FFFE024430307D7E64

132 7E001197010013A200403A3BF87D7D7D7E6D443003A9

515 7E001017010013A200403A3BF8FFFE027D7E3032A6

1767 7E0012920013A200403A3BF80634410100007D7E020708

2701 7E0012920013A200403A7D7EF806344101000001020A3F

3054 7E001117010013A200403A3BF8FFFE0244307D7E3262

Total de tramas con secuencia de escape : 6

==========

Actividad 2

Trama 3 archivo : captura1.cap

Capa de Enlace de datos
PPP - CHAP
Capa Física
Ethernet

a. Ninguna trama de datos fue enviada, por lo tanto no se pierde ni llegan datos. Esto pasa porque la conexión nunca se estableció.

b. Esta pregunta no se puede responder ya que el checksum se calcula para las tramas de datos que fueron enviadas, y como la conexión no se estableció no hay ningún checksum que se pueda verificar

Trama 2 archivo : captura3.cap

Capa de Aplicación
HTTP
Capa de Transporte
TCP
Capa de Red
IP
Capa Física
Ethernet II

a. Ninguna trama se pierde ya que tenemos una capa con el protocolo TCP el cual si una trama o paquete no llego al destino, gracias al mecanismo de ACK, al emisor no recibir el ACK de nuestro destino, vuelve a reenviar el paquete, por un hecho de que si no recibió el ACK entonces el paquete se perdió o no llego

b. Todas las tramas tendrá checksum correcto porque Ethernet descarta tramas con checksum incorrecto, y también gracias al protocolo TCP el cual descarta los checksum que son incorrectos, y no responder con un ACK al emisor conseguimos que el emisor vuelva a enviar el paquete y que llegue en buen estado

Trama 24 archivo : captura4.pcanpg

Capa de Aplicación
HTTPS/TLS
Capa de Transporte
TCP
Capa de Red
IP
Capa Física
Ethernet II

a. Ninguna trama se pierde ya que tenemos una capa con el protocolo TCP el cual si una trama o paquete no llego al destino, gracias al mecanismo de ACK, al emisor no recibir el ACK de nuestro destino, vuelve a reenviar el paquete, por un hecho de que si no recibió el ACK entonces el paquete se perdió o no llego

b. Todas las tramas tendrá checksum correcto porque Ethernet descarta tramas con checksum incorrecto, y también gracias al protocolo TCP el cual descarta los checksum que son incorrectos, y no responder con un ACK al emisor conseguimos que el emisor vuelva a enviar el paquete y que llegue en buen estado

Código :

```
 ## Corta las tramas por su bandera '7E'
def framer_analyzer(frame) :
    frame_list = []
    index = 0

    while index != -1  :
        start = frame.find("7E", index)

        if start == -1 :
            break

        end = frame.find("7E", start + 1)

        while end != -1 and frame[end - 2: end] == "7D":
            end = frame.find("7E", end + 1)

        subframe = ''
        if end == -1 :
            subframe = frame[start:]
            index = end
        elif end != -1:
            subframe = frame[start:end]
            index = end
        frame_list.append(subframe)
    return frame_list

## Analiza si la longitud del frame es correcta o no
def is_ok_frame(frame):
    if frame.find("7D7E") != -1 :
       frame = frame.replace("7D7E", "7E")
    long = frame[2:6]
    long_dec = int(long, 16)
    long_frame = frame[6: -2]
    checksum = frame[-2:]

    if long_dec == len(long_frame)/2 :
        return True
    else :
        return False
## Cuenta Las longitudes correctas y las que no, esta las agrega a otra lista nueva para poder ser utilizada mas ade
```

```
def counter_frame_ok(listf):
    tramas_ok = []
    tramas_not_ok = []
    imprimir_con_decoracion("Salida de funcion que cuenta la longitud, en la salida muestra el indice y la trama que
    for ind, frame in enumerate(listf) :
        if is_ok_frame(frame):
            tramas_ok.append(frame)
        else :
            print(ind, frame)
            tramas_not_ok.append(frame)
    return tramas_ok, tramas_not_ok
## Un utils
def imprimir_con_decoracion(mensaje):
    print("==========================================")
    print(mensaje)
    print("==========================================")
##Esto es para ver si tiene un salto de linea, una forma de verificar si hay algun detalle que se escapo cuando hice
def tiene_salto_de_linea(list_frame):
    counter = 0
    for frame in list_frame:
        if frame.find(" ") != -1 :
            print("upsi")
        find_replace(list_frame, frame, "\n", "")
        ## Parecido a otro
        counter += 1
    return counter
## Utils
def find_replace(list_frame, frame, stringBusc, cambiar, mostrar = False):
    if frame.find(stringBusc) != -1 :
        framex = frame.replace(stringBusc, cambiar)
        ind = list_frame.index(frame)
        if mostrar :
            print(ind, frame)
        list_frame.pop(ind)
        list_frame.insert(ind, framex)

## Suma el frame para al retornarlo podamos compararlo con el checksum
def sumfr(frame)  :
    suma = 0
    for i in range(0, len(frame), 2):
        hexSum = frame[i] + frame[i+1]
        suma += int(hexSum, 16)
    return suma
## Funcion que analiza los checksum
def checksum_is_ok(list_frame):
    counter = 0
    counter_isnotok = 0
    imprimir_con_decoracion("Salida de funcion que cuenta los checksum, la salida muestra los incorrectos : " )
    for i,fr in enumerate(list_frame):
        if fr.find("7D7E") != -1 :
            fr = fr.replace("7D7E", "7E")
        checksum_frame = fr[-2:]
        long_frame = fr[6: -2]
        sumCheck = sumfr(long_frame)
        difSum = 0xFF & sumCheck
        checkSumByte = 0xFF - difSum
        if int(checksum_frame, 16) == int(checkSumByte):
            counter += 1
        else :
            print(i, fr)
            counter_isnotok +=1
    return counter, counter_isnotok
```

```
## Funcion que cuenta las tramas con secuencia de escape
def con_sec_escape(list_frame):
    counter = 0
    imprimir_con_decoracion("Salida funcion que cuenta las tramas con Secuencia de Escape:")
    for i,frm in enumerate(list_frame) :
        if frm.find("7D7E") != -1 :
            counter += 1
            print(i, frm)
            frame = frm.replace("7D7E", "7E")
            list_frame.pop(i)
            list_frame.insert(i, frame)
    return counter


if __name__ == "__main__":
    try:
        with open('Tramas_802-15-4.log', 'r') as frame_log :
            frame = frame_log.read()

        list_anal = framer_analyzer(frame)
        tiene_salto_de_linea(list_anal)

        tramasOk, tramasnotOk = counter_frame_ok(list_anal)
        print("Tramas Totales:", len(list_anal))
        print("Tramas con longitud correcta:", len(tramasOk), "\nTramas con longitud incorrecta:", len(tramasnotOk))

        checkSumOk, checkSumNotOK = checksum_is_ok(tramasOk)
        print("CheckSum que son correctos :", checkSumOk, "\nCheckSum que son incorrectos: ", checkSumNotOK)

        print("Total de tramas con secuencia de escape : ", con_sec_escape(tramasOk))

    except FileNotFoundError:
        print("Error: El archivo 'Tramas_802-15-4.log' no se encuentra.")
    except Exception as e:
        print(f"Ocurrió un error inesperado: {e}")
```