Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de la Universidad

Tarea 3

Profesor: José M. Piquer
Auxiliar: David Miranda
Estudiante: Samuel Chavez F.

Fecha de realización: 25 de noviembre de 2023
Fecha de entrega: 27 de noviembre de 2023
Santiago de Chile

1. Metodologia

Este experimento se realizo utilizando en Windows 11 22H2 usando Windows Subsystem for Linux 2 (WSL2) corriendo la distribucion de ubuntu 22.04.2 LTS.

La tarea detallaba usar el cliente y servidor de la T1 no obstante debido a que mi desarrollo en la T2 lograba globalizar todos estos aspectos configurando la ventan a con tamaño 1 (véase mi programa acepta la flag especifica °—-window_sz=n°) simplemente testee todo lo pedido con esta mejor versión de cliente, que también acepta el servidor de la T1 en el puerto 1819 de anakena y basicamente se corrio de la siguiente forma para los tests mas basicos.

Código 1: Comando que se lanzo

```
# ./H4CK5.py anakena.dcc.uchile.cl 1819 IP_DESTINO PUERTO_DESTINO
```

notar que se uso la convencion de comandos para linux, donde '#' denota un usuario root, por lo que si se corre en otro usuario debe llamarse con 'sudo'.

1.0.1. Uso correcto del archivo

El archivo tiene 4 argumentos que requiere para funcionar por un lado 'origin' y 'origin_port' correspondientes a el emisor a imitar, falsificando los paquetes y 'destination' y 'destination_port' correspondientes al destinatario a enviar. Esto es normal de lo que solicita la tarea, tambien se incluyen otras 2 flags opcionales

'-t, –timeout' por un lado, especifica el tiempo que estará enviando los paquetes al destinatario antes de terminar, por defecto 60 segundos, '-f, –flood' es una flag experimental que lo que hace es sobrecargar el uso normal de scapy, proveyéndole un socket de antemano a la función send, lo que hace que el envió de paquetes básicamente no tenga costo de tiempo (al no cerrar y abrir un socket todo el rato) lo que como explicaremos mas adelante termina secuestrando fuertemente el protocolo, esta flag implementa también por lo tanto, un envió extra de 100 paquetes con el label 'E' para que pueda terminar de forma regular el ciente.

2. P1

En lo personal conjeturo que seria mas fácil, pero antes de dejarlo únicamente como una hipótesis al respecto, podemos hacer la prueba concreta por como realize mi setup.

Código 2: Comando experimental del cliente y programa malicioso

```
# ./bwc-sr.py 50 900000 100 0 out anakena.dcc.uchile.cl 1818
# ./H4CK5.py anakena.dcc.uchile.cl 1818 IP_DESTINO PUERTO_DESTINO
```

Efectivamente funciona y en general tiende a ser inclusive mas efectivo que normalmente lo es con solo stop and wait, esto ultimo razonablemente se le puede atribuir a que como esta esperando a una ventana de paquetes en lugar de a un único paquete, hay mas rango en el que el programa aleatoriamente puede por mera chance, dar con algo que el protocolo acepte.

3. P2

Una forma que es estándar para poder asegurar un camino de comunicación es el protocolo TLS el cual básicamente hace un intercambio de llaves que asegura la comunicación servidor cliente, haciendo que directamente lo que envié un atacante sea a ojos del cliente sea basura, ni siquiera podria contener el numero de paquete secuencial, sin hacer un ataque especializado en el que pueda conseguir alguna de las llaves del servidor privadas, o intentar craquearlo tomándole una cantidad de tiempo absurda, potencialmente mas que la vida estimada del universo.

4. P3

En teoría, dado que TCP no es mas que un protocolo de ventana corredera, es posible, ahora es valido notar que a esto se le generan varios desafios solo en la conexion mas simple que se podria implementar, solo para empezar el problema de alguna forma identificar los Sequence Num correctos para poder engañar al cliente con un paquete, luego esta el problema de fragmentacion de datos, podria ser que interrumpamos una fragmentacion ya en curso y esto termine con problemas que hara que la conexion sea interrumpida no generando algun vector de ataque posible, ahora son problemas fuertes pero no imposibles de sobrellevar, de ahi que ya existen distintas herramientas que lo logran, como scapy.

5. P4

Si se podría hacer lo mismo, y se podría argumentar que de hecho da un poco igual que el ataque lo estemos generando en la misma maquina ya que finalmente lo que esta pasando es un envió de paquetes IP en donde impersonamos al servidor, lo único que esta sucediendo al tener la aplicacion dentro de nuestro equipo es que los paquetes al pasar por la tabla de ruteo, el equipo se esta dando cuenta que el remitente es también el receptor, y lo envía a su propio puerto destinatario, en este caso al puerto sobre nuestro pobre protocolo quien confía que los paquetes que le llegan por IP son justamente del servidor lo que termina muy mal para el, escribiendo la linea de hackeo enviada por el atacante en el archivo de salida.

6. P5

El archivo es mas pequeño, dado que efectivamente, los paquetes inyectados eran menores a los que se tenían en el protocolo no hay mayor complejidad al respecto.