Informe Trabajo Práctico Especial

Protocolos de Comunicación - Primer Cuatrimestre 2022

Grupo 2

Chao Florencia - Legajo 60054
Konfederak Sol - Legajo 60255
Finucci Roca Hernan - Legajo 60178
Fernandez Truglia Ariadna - Legajo 60199

21 de Junio de 2022

Índice

1. Introducción **3**

2. Desarrollo del proxy **3**

3. Prawtosv5 **4**

1. Negociación Inicial: 4

1.1 Pedido: 4

1.2 Respuesta: 4

2. Obtención de Métricas (GET): 5

2.1 Pedido: 5

2.2 Respuesta sobre el pedido: 5

3. Modificadores de Métricas (Users): 6

3.1 Agregar un usuario: 6

3.2 Baja de un usuario: 6

3.3 Modificación de la contraseña de un usuario: 6

3.3 Respuesta del pedido: 7

4. Configuración remota: 7

4.1 Habilitar/deshabilitar el sniffer de contraseñas: 7

4.2 Respuesta sobre el pedido: 8

5. Configuración remota: 8

5.1 Cerrar la conexión del cliente: 8

5.2 Respuesta sobre el pedido: 8

Decisiones sobre el protocolo: 9

4. Problemas encontrados durante la implementación del TPE **9**

6. Posibles extensiones **11**

8. Ejemplos de prueba **11**

9. Guía de instalación y configuración **14**

1. Introducción

El fin de este trabajo práctico es implementar un servidor proxy Socks5 utilizando los RFC 1928 y 1929. Las funcionalidades a implementar son las siguientes:

- Autenticación mediante usuario y contraseña.
- Monitoreo de datos transferidos y de usuarios conectados.
- Sniffeo de credenciales POP3.
- Herramienta que permita, en ejecución, cambiar la configuración del servidor proxy.
- Soportar consultas recibiendo IPv4, IPv6 o nombre de dominio.

2. Desarrollo del proxy

Nuestro proxy es un servidor no bloqueante que permite hasta 1024 conexiones en un mismo momento, tanto para IPv4 como para IPv6. Esto se al límite de file descriptors que puede manejar select().

Al iniciar la comunicación con el proxy mediante el uso del cliente, primero se envía un paquete de negociación inicial llamado AUTH. En este, se envían las credenciales al servidor con el fin de saber si el usuario tiene acceso a las funcionalidades provistas. El servidor es el encargado de parsear la información y determinar si se debe continuar con la conexión o terminar la sesión, devolviendo que los datos enviados no son válidos.

Luego de la autenticación inicial, el usuario tiene acceso a las siguientes funcionalidades:

1. Solicitar la cantidad de bytes transferidos en esta sesión.
2. Solicita la cantidad total de conexiones del servidor.
3. Solicita la cantidad de usuarios utilizando el cliente actualmente.
4. Solicita el nombre de todos los usuarios del servidor, sean tanto usuarios normales como administradores.
5. Agrega un usuario a la base de datos del servidor.
6. Agrega un administrador a la base de datos del servidor.
7. Eliminar un usuario o administrador de la base de datos.
8. Modificación de credenciales de un usuario o administrador de la base de datos
9. Herramienta para habilitar o deshabilitar el sniffer de POP3.
10. Herramienta para cerrar la sesión.

Cada conexión con el servidor con el proxy tiene asociada una estructura llamada Session, donde se guarda toda la información relevante, como puede ser la información de la conexión, información del usuario y estado actual, entre varios datos más. Dicho estado es relevante para poder navegar por la máquina de estados.

Respecto a la persistencia de la información del servidor, se optó por un sistema volátil, es decir, que no persiste una vez que se cierra la sesión, debido a su facilidad de implementación.

3. Prawtosv5

Este protocolo busca poder obtener métricas e información de un proxy SOCKSv5 y que la modificación del mismo sea en runtime. El mismo es un protocolo orientado a sesión y funciona sobre SCTP. Se pueden hacer tres tipos de comandos: GET, SET y QUIT. GET permitirá obtener diferentes

datos almacenados en el servidor. El comando SET se ejecuta con el fin de modificar la información de los usuarios/administradores del servidor y modificar alguna configuración del servidor.

1. Negociación Inicial:

El usuario genera el primer pedido de conexión para poder conectarse al servidor. El mismo se hace a través de un pedido inicial que le permite identificarse al mismo.

1.1 Pedido:

VER	ULEN	UNAME	PLEN	PASSWD
1	1	1 to 255	1	1 to 255

Donde:

- VER se refiere a la versión del protocolo:
  - X'01' VERSIÓN 1
- ULEN: se refiere a longitud del campo UNAME
- UNAME: es el nombre del usuario
- PLEN: se refiere a la longitud del campo PAWSSWD
- PAWSSWD: es la contraseña

1.2 Respuesta:

VER	STATUS
1	1

Donde:

- VER se refiere a la versión del protocolo:
  - X'01' VERSIÓN 1
- STATUS: es el estado de la conexión
  - Conexión exitosa
    - X'00'
  - Fallo del servidor
    - X'01'
  - Versión no soportada
    - X'02'
  - Identificación de usuario incorrecta
    - X'03'

2. Obtención de Métricas (GET):

2.1 Pedido:

TYPE	CMD

1	1

Donde:

- TYPE se refiere al tipo de comando:
  - X'00' GET
- CMD
  - GET de bytes transferidos:
    - X'00'
  - GET de conexiones históricas
    - X’01’
  - GET de conexiones concurrentes
    - X’02’
  - GET de listar usuarios
    - X’03’

2.2 Respuesta sobre el pedido:

STATUS	CMD	NARGS	ARGS
1	1	1 to 255	variable

Donde:

- STATUS es el estado de la conexión:
  - Conexión exitosa
    - X'00'
  - Fallo del servidor
    - X'01'
  - Tipo de comando no soportado
    - X'02'
  - Comando no soportado
    - X'03'
- CMD es el comando solicitado
  - GET de bytes transferidos:
    - X'00'
  - GET de conexiones históricas
    - X’01’
  - GET de conexiones concurrentes
    - X’02’
  - GET de listar usuarios
    - X’03’
- NARGS es la cantidad de argumentos del campo ARGS
- ARGS son los argumentos del comando solicitado, cuyo primer byte corresponde a la longitud del argumento

3. Modificadores de Métricas (Users):

3.1 Agregar un usuario:

TYPE	CMD	ADMIN	ULEN	UNAME	PLEN	PASSWD
1	1	1	1	1 to 255	1	1 to 255

Donde:

- TYPE se refiere al tipo de comando:
  - X'01' USERS
- CMD se refiere al comando solicitado
  - X’00’ CREATE
- ADMIN es un flag para indicar el tipo de usuario
  - X’00’ ADMIN
  - X'01' USER
- ULEN es longitud del campo UNAME
- UNAME es el nombre del usuario
- PLEN es la longitud del PASSWD
- PASSWD es la contraseña

3.2 Baja de un usuario:

TYPE	CMD	ULEN	UNAME
1	1	1	1 to 255

Donde:

- TYPE se refiere al tipo de comando:
  - X'01' USERS
- CMD se refiere al comando solicitado
  - X’01’ DELETE
- ULEN es longitud del campo UNAME
- UNAME es el nombre del usuario

3.3 Modificación de la contraseña de un usuario:

TYPE	CMD	ULEN	UNAME	PLEN	PASSWD
1	1	1	1 to 255	1	1 to 225

Donde:

- TYPE se refiere al tipo de comando:
  - X'01' USERS

- CMD se refiere al comando solicitado
    - X’02’ EDIT
- ULEN es longitud del campo UNAME
- UNAME es el nombre del usuario
- PLEN es la longitud de PASSWD
- PASSWD es la nueva contraseña

3.3 Respuesta del pedido:

STATUS
1

Donde:

- STATUS se refiere al estado sobre el pedido
    - Operación exitosa
        - X'00'
    - Fallo del servidor
        - X'01'
    - Tipo de comando no soportado
        - X'02'
    - Comando no soportado
        - X'03'
    - Error en la longitud para campos de usuarios
        - X’04’
    - Error en credenciales de usuario/no existe usuario
        - X’05’

4. Configuración remota:

4.1 Habilitar/deshabilitar el sniffer de contraseñas:

TYP	CMD
1	1

Donde:

- TYP se refiere al tipo de comando:
    - X'02' SNIFF
- CMD: flag para habilitar o deshabilitar el sniffeo de contraseña
    - Habilitar
        - X'00'
    - Deshabilitar
        - X'01'

4.2 Respuesta sobre el pedido:

STATUS
1

Donde:

- STATUS se refiere al estado sobre el pedido
  - Operación exitosa
    - X'00'
  - Fallo del servidor
    - X'01'
  - Tipo de comando no soportado
    - X'02'
  - Comando no soportado
    - X'03'

5. Configuración remota:

5.1 Cerrar la conexión del cliente:

TYPE
1

- TYP se refiere al tipo de comando:
  - X'02' QUIT

5.2 Respuesta sobre el pedido:

STATUS
1

Donde:

- STATUS se refiere al estado sobre el pedido
  - Operación exitosa
    - X'00'
  - Fallo del servidor
    - X'01'
  - Tipo de comando no soportado
    - X'02'
  - Comando no soportado
    - X'03'

Decisiones sobre el protocolo:

Luego de analizar diferentes alternativas, se decidió permitir argumentos variables en el caso de los GETs ya que podían recibir como resultado diferentes longitudes de respuestas, como por ejemplo al momento de listar los usuarios. Pero, para que sea más sencillo de parsear, hemos decidido que no siempre el tamaño sea variable, sino solamente cuando sea necesario. Por lo tanto, cada request tiene sus diferentes argumentos dependiendo de lo que sea necesario para no enviar información innecesaria. Por ejemplo, para modificar un usuario, no se necesita saber si el mismo es administrador o no; mientras que para agregarlo si es necesario.

El caso de get users es uno muy particular. El cliente no sabe a la hora de hacer el pedido cuantos bytes va a recibir, parámetro necesario para la función recv(). En un principio nuestra idea era simplemente implementar un límite máximo de bytes a recibir, pero finalmente decidimos optar por una solución más compleja y flexible. Si el cliente quiere hacer un pedido de usuarios, entonces inicialmente va a estar esperando una respuesta de 4 bytes, compuesta por el tipo de comando, el comando en si, la cantidad de usuarios (nargs) y finalmente el primer byte del campo variable args. En este primer campo se pasa la longitud del primer nombre de usuario. Con esta información se hace otro recv() más, esta vez solicitando tantos bytes como la longitud del usuario más un byte extra, que representa la longitud del próximo usuario. Este proceso se repite tantas veces como la cantidad de usuarios a devolver.

Para poder distinguir los diferentes tipos de pedidos, decidimos implementar un campo type, para distinguir si se trata de un seteo o de una obtención de información y un tipo cmd para poder distinguirlas entre ellos.

Por otro lado, cabe destacar que solamente los usuarios pueden visualizar las métricas y crear usuarios pero si pueden utilizar la autenticación para utilizar el socks5.

El servidor viene por defecto con dos usuarios administradores “admin1” y “admin2” para poder realizar los comandos. Sus contraseñas son iguales a sus nombres.

4. Problemas encontrados durante la implementación del TPE

Si bien la principal dificultad de este trabajo era el diseño de nuestro propio protocolo, encontramos muchos impedimentos a la hora de configurar el socks5. Esto es debido a varios factores:

En un primer lugar ninguno de los integrantes contaba con un sistema operativo nativo Linux, por lo que a la hora de compilar el proyecto no se soportaban varios flags necesarios para la generación del trabajo tal como el flag *-D_POSIX_C_SOURCE=200112L* . Tampoco era una opción utilizar las máquina virtuales ya que lidiamos con ellas a lo largo de la cursada y frecuentemente se “laggeaban”. Si bien se podría argumentar que no era necesario ya que el TPE debía funcionar en pampero, también tuvimos dificultades a la hora de ponerlo a prueba ya que varias de las veces nos figuraban los puertos requeridos en uso o que no era posible conectarse al mismo. Otro problema era que funciones pertenecientes a *c* tal como *pthread_create()* generaban segmentation fault.

En segundo lugar, al tratar de modularizar las estructuras utilizadas para el socks5 se generaron problemas en la alocación de memoria que hacía que fallara la integridad del trabajo. Generalmente estos problemas requerían de mucho tiempo de debugeo y a la vez impedían el avance del proyecto. También encontramos dificultades a la hora de generar el request para los parsers ya que en varias ocasiones notamos que se enviaban mal los parámetros. Muchas de las veces nosotros pensamos que la lógica de nuestro parser estaba mal cuando el problema era otro.

En tercer lugar, al comienzo del proyecto se nos dificulto comprender por donde se debía encarar el proyecto. Si bien las clases de consulta nos fueron guiando en el proceso, por momentos

nos sentíamos abrumados con tanta información. Muchas veces, las volvíamos a ver para despejar dudas de como implementar algunas cosas.

5. Limitaciones

Con respecto al sniffeo de contraseñas de POP3 y los registros de acceso no se logran ver en consola ya que se genera un segmentation fault debido a que al querer imprimir en pantalla la información no estamos pudiendo acceder a la estructura donde guardamos la información. De igual manera está todo implementado y en funcionamiento. Se puede observar a través de un debugger (en nuestro caso usamos gdb y lldb) cómo esta información efectivamente se va guardando en la estructura dedicada a almacenar estos datos(*pop3_sniff*).

```
ariadna@ariadna-Legion-Y540-15IRH:~/Desktop/pdc-20221c/client$ ncat -v -C --proxy-type socks5 --proxy
 localhost:1080 127.0.0.1 110
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Connected to proxy 127.0.0.1:1080
Ncat: No authentication needed.
Ncat: connection succeeded.
+OK Dovecot (Ubuntu) ready.
user ariadna
+OK
pass password123
```

Ejemplo de inicio de sesion a postfix

```
   26 | static void close_session(selector_key * event);
      |             ^~~~~~~~~~~~~~
ariadna@ariadna-Legion-Y540-15IRH:~/Desktop/pdc-20221c$ ./server
Failed to bind socket to sockaddr: Address already in use
127.0.0.1
saliendo de check con 5
saliendo de check con 5
saliendo de check con 6
En POP success
Segmentation fault (core dumped)
```

Se puede ver cómo se imprime en pantalla “En POP success”, lo que representa que efectivamente se parseo con éxito las credenciales de POP3 y estas son válidas. En el caso de que las credenciales introducidas fueran inválidas, no se generaría un segmentation fault ya que el servidor no las trataría de guardar en la estructura de logueo.

```
saliendo de check con 5
saliendo de check con 6
En POP success

Breakpoint 1, print_sniff_consume (sniff=0x0) at src/pop3sniff.c:149
(gdb) n
(gdb) print sniff->username
$1 = "ariadna\r\000", '\005' <repeats 246 times>
(gdb) print sniff->password
$2 = "password123\r\000", '\005' <repeats 242 times>
```

Por otro lado, la resolución de domain se encuentra comentada en el código ya que fue implementada pero no funciona correctamente ya que en el pthread el llamado a función void genera unas excepción.

6. Posibles extensiones

Debido a las limitaciones actuales de nuestro servidor proxy, se podría corregir los defectos encontrados en la actualidad para poder hacer una correcta visualización de las contraseñas y usuarios

capturados en el POP3. Al mismo tiempo, se podría extender sobre DNS el sniffing de las contraseñas. También al solucionar el error de la impresión de las contraseñas se podrían visualizar los usuarios que ingresan en el servidor.

En relación a la resolución de nombres, habría que solucionar el problema que causa el pthread. Pero dejando eso de lado, se podría implementar pipelining y conexiones persistentes.

Mientras que, con respecto al protocolo, se podrían agregar diferentes servicios para agregar funcionalidades, como puede ser diferenciación en los niveles de acceso a la información o soportar pipelining si se agrega en el servidor.

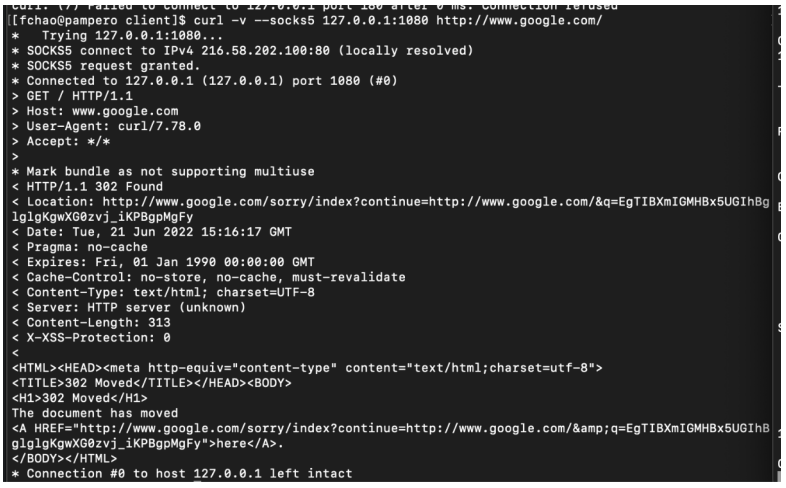
7. Conclusiones

Más allá de los inconvenientes que se presentaron en el proceso de desarrollo, se pudo realizar correctamente el objetivo principal del trabajo práctico: diseñar e implementar un protocolo propio que interactúa exitosamente con el servidor proxy.

Cabe destacar que nuestro protocolo Prawtosv5 es agnóstico a la lógica implementada por el proxy socks5. Es decir, que si en un futuro se quiere modificar la implementación del servidor, esto no ofrece ninguna dificultad agregada.

8. Ejemplos de prueba

Algunos ejemplos de prueba son los siguientes:
1: realizar un curl a una página web y poder observar los bytes transferidos del mismo.



pedido del cliente

```
10 - Quit

Choose an option:
1

---------------------------------------


Retrieving transfered bytes...


Operation successfull

Bytes transfered: 797

GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
```

respuesta enviada de prawtos

2: Creación de usuarios y administradores:

```
Choose an option:
6

---------------------------------------


---- Create admin ----
Username: skonfederak
Password: skonfederak


Operation successfull
GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
```

```
Access denied: user is not an admin
[fchao@pampero client]$ ./client

Welcome to the Prawtos client!
Please sign in with your credentials
Username: skonfederak
Password: skonfederak

Successful connection

GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
```

Creación e inicio de sesión de un usuario administrador.

```
Choose an option:
5

---------------------------------------


---- Create user ----
Username: fchao
Password: fchao


Operation successfull
GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit
```

```
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$
[fchao@pampero client]$ ./client

Welcome to the Prawtos client!
Please sign in with your credentials
Username: fchao
Password: fchao

Access denied: user is not an admin
[fchao@pampero client]$ ./client
```

Creación y validación de autenticación de un usuario común.

3: establecer la dirección donde se deben conectar tanto el cliente como el servidor con el flag -L desde la entrada estándar:

```
Failed to bind socket to sockaddr: Address already in use
127.0.0.1
unable to bind configuration socket: Address already in use
[[fchao@pampero pdc-20221c-1]$ ./server -L 127.0.0.2
```

florenciachao — fchao@pampero:~/Documents/TP_Protos/pdc-20221c-1/..

```
[Username: ^C
[[fchao@pampero client]$ ./client -L 127.0.0.2

Welcome to the Prawtos client!
Please sign in with your credentials
[Username: admin
[Password: alggo

Successful connection
```

4: conectar más de un usuario a la vez y poder visualizar cómo aumentan las conexiones históricas y las conexiones concurrentes.

```
[Password: alggo

Successful connection

GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
```

```
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
3

-------------------------------------


Retrieving concurrent connections...


Operation successfull

Concurrent connections: 2

GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
```

```
Retrieving users...


Operation successfull

 ---- Users ----
admin2
admin
pruebita
fchao
skonfederak

GET:
   1: Total bytes transfered
   2: Historical connections
   3: Concurrent connections
   4: List all users

SET:
   5: Add user
   6: Add user admin
   7: Remove user
   8: Change user password
   9: Enable/disable password sniffer

10 - Quit

Choose an option:
```

Al mismo tiempo, se puede visualizar el listado de los usuarios.

9. Guía de instalación y configuración

En el archivo README se encuentra toda la información relevante respecto a la instalación y configuración del proyecto.