

Protocolos de Comunicación

Informe Trabajo Práctico Especial.



Grupo 6

Integrantes:

- Joaquin Garcia Del Rio - (59051).
- German Ariel Martinez - (58574).
- Franco Navarro - (59055).
- Santiago Preusche - (59233).

Primer Cuatrimestre - 2022

Descripción de la aplicación	2
1 - Conexiones concurrentes y simultáneas	3
2 - Soportar requests en formato IPv4, IPv6 y FQDN	3
3 - Métodos de autenticación	4
4 - Reporte de fallos	4
5 - Recolección de métricas	4
6 - Manejo de usuarios	5
7 - Manejo de administradores	5
8 - Sniffing	6
9 - Sistema de registro de logs	6
Descripción de la servidor de monitoreo	7
Problema durante desarrollo	8
Limitaciones de la aplicación	8
Extensiones	8
Conclusiones	8
Ejemplos de prueba	9
Guia de instalación	10
Configuración del servidor	10

Descripción de la aplicación

Para este Trabajo Práctico Integrador se desarrolló un servidor Proxy que utilice el protocolo Socks V5. Para poder implementarlo correctamente se siguieron los lineamientos presentados en los RFCs 1928, el cual describe el protocolo en sí, y el 1929, el cual se utilizó para implementar el sistema de autenticación con usuario y contraseña.

Al mismo tiempo nuestro servidor proxy puede:

1. Atender múltiples clientes en forma concurrente y simultánea.
2. Soportar requests de conexión en formato IPv4, IPv6 y FQDN.
3. Reportar fallos usando toda la potencia del protocolo.
4. Recolectar métricas de la aplicación.
5. Manejo de usuarios y administradores.
6. Obtener usuarios y contraseñas de paquetes interceptados por el proxy.
7. Generar logs de uso de la aplicación.

A continuación se procederá a dar una breve explicación de las funcionalidades mencionadas anteriormente.

1 - Conexiones concurrentes y simultáneas

Para esta funcionalidad se utilizaron todos los conocimientos enseñados por la cátedra sobre el manejo de sockets en C. Al correr por primera vez el servidor se crean 4 sockets pasivos no bloqueantes que están a la escucha de una nueva solicitud de conexión por parte del cliente y/o administrador. Más en detalle, se crean:

- Socket pasivo IPv4 para clientes.
- Socket pasivo IPv6 para clientes.
- Socket pasivo IPv4 para administrador.
- Socket Pasivo IPv6 para administrador.

Una vez fueron creados estos sockets el sistema, en complemento con la librería provisionada por la cátedra "**selector.h**", se bloquea y se queda a la espera de que algún cliente busque conectarse para tratar su solicitud.

2 - Soportar requests en formato IPv4, IPv6 y FQDN

Una vez el proxy trata la solicitud de conexión del cliente, por el mismo funcionamiento del protocolo, se entra en una etapa de negociación cliente-proxy antes de conectar al proxy con el servidor deseado por el cliente. La negociación funciona de la siguiente manera:

1. El cliente manda un mensaje de **Greeting** al proxy, este mensaje está formado por la versión de Socks en la que trabaja el cliente junto a una lista de métodos de autenticación que él puede realizar.
2. El proxy válida este mensaje y le responde con otro indicando con qué método de autenticación vamos a comunicarnos. En caso de que no estemos trabajando en la misma versión de protocolo, o no soportemos ningún de los métodos de autenticación realizables por el usuario, el proxy le pide al cliente que se desconecte por medio de este mismo mensaje.
3. A continuación el cliente le envía las credenciales acordes al método de autenticación seleccionado, si es que no debe desconectarse.
4. Una vez nos autenticamos el cliente nos envía un mensaje de **Request** el cual se encuentra formado por:

VER	CMD	RSV	ATYP	ADDR	PORT
-----	-----	-----	------	------	------

Donde:

- VER = versión del protocolo (5 en nuestro caso).
- CMD = el comando a realizar, hay 3 posibles, CONNECT, BIND, UDP Associate. A fines de este trabajo nuestro proxy solo aceptará requests de CONNECT.
- RSV = campo reservado.
- ATYP = el tipo de Address que nos enviará (IPv4,IPv6 o FQDN).
- Address y puerto al cual quiere conectarse.

En base al campo ATYP nosotros cambiaremos la forma de conectarnos al servidor origin, si es una IPv4 o 6 se intentará conectar al cliente de forma directa con el origin. En caso de que nos haya enviado una FQDN se utilizará la función "**getaddrinfo**" para obtener una lista de posibles direcciones la cual será iterada hasta lograr que el proxy establezca conexión con alguna de ellas. En ambos casos si el resultado es no satisfactorio se le notificará al cliente para que se desconecte por medio de una reply a la request de conexión.

5. Una vez le llegue el resultado de la request al cliente y este es satisfactorio la etapa de negociación habrá finalizado y comenzara la comunicación cliente-proxy-origin.

3 - Métodos de autenticación

En el paso anterior se mencionó que el cliente, durante la etapa de negociación, indicará los métodos de autenticación por medio de los que él podría conectarse. Nuestro servidor soportara:

- Autenticación por usuario y contraseña.
- Ningún tipo de autenticación.

4 - Reporte de fallos

Siguiendo los lineamientos planteados por el RFC-1928 durante la etapa de negociación los errores serán comunicados siguiendo el formato, sintaxis y convenciones indicados en el documento antes mencionado, es decir, por medios de los mensajes de la etapa de negociación.

5 - Recolección de métricas

Nuestro sistema recolecta métricas para poder representar de una mejor forma el estado y/o uso del servidor proxy. Recolectamos información de:

1. Conexiones históricas: cantidad total de conexiones realizadas hasta la fecha.

2. Conexiones concurrentes: cantidad de usuarios conectados en este instante.
3. Bytes transferidos: cantidad de bytes enviados vía nuestro proxy.

6 - Manejo de usuarios

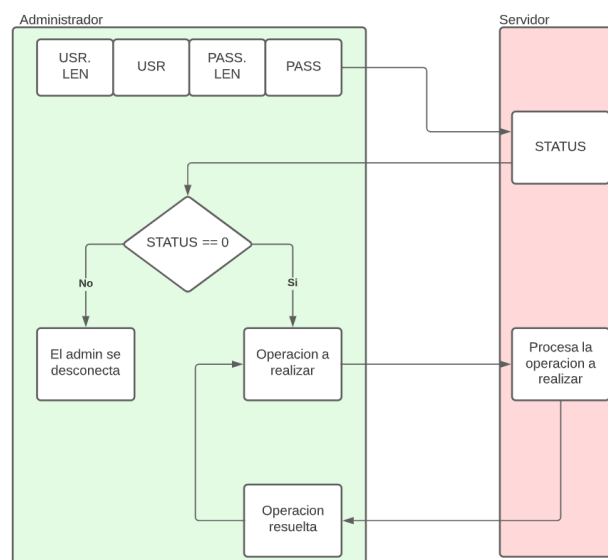
Los usuarios serán declarados al momento de correr el servidor proxy por medio de la línea de comandos. Como máximo se tendrán hasta 10 usuarios por defecto. Los usuarios requerirán de ser creados junto a una contraseña, ambos campos serán utilizados por el servidor proxy en caso de que el mismo haya seleccionado un método de autenticación por usuario y contraseña cuando se intente conectar el cliente.

7 - Manejo de administradores

Al comienzo de este informe nosotros hablamos sobre que nuestro servidor proxy tiene reservados ciertos sockets pasivos para clientes y para administradores. Los clientes se conectaran al puerto 1080 y los administradores al 8080, una vez que un administrador se haya conectado, el mismo se comunicara con nuestro servidor por medio de un protocolo creado por nosotros, a través del mismo el administrador podrá:

- Obtener las métricas del sistema.
- Crear nuevos usuarios.
- Modificar el tamaño de los buffers del sistema.
- Listar todos los usuarios creados.

El administrador se comunica con el servidor por medio de una aplicación cliente que funciona de la siguiente manera:



8 - Sniffing

Soportamos sniffing de credenciales (usuario y contraseña) para POP3. Esto se logra parseando los paquetes que pasan por el proxy. Esto se hace de la siguiente manera:

1. Primero parseamos el paquete enviado por el origen en busca de un "+OK".
2. Una vez encontrado se parsean los mensajes USER y PASS que son los que utiliza POP3 para poder acceder al servidor. Guardamos el usuario y contraseña encontrados y analizamos la respuesta del origen.
3. Si la respuesta es un "+OK" es que fueron correctas, en caso negativo, descartamos las credenciales antes guardadas y volvemos a pasar las nuevas credenciales, si las hay.
4. Una vez terminado el parseo con éxito, las credenciales recolectadas se imprimen por STDOUT del lado del servidor.

Aclaración: no permite obtener las credenciales si se hace a través de pipelining.

9 - Sistema de registro de logs

Nuestro sistema de logueo imprime por STDOUT las conexiones y desconexiones de los usuarios a nuestro servidor. Estos logs presentan el siguiente formato:

```
Running SocksV5 server...
[2022-06-21 15:08:28] - Client connected succesfully: Non-auth user      A      1
27.0.0.1      62814      142.251.133.196 80      0
[2022-06-21 15:08:28] - Client disconnected: Non-auth user      A      127.0.0.
1      62814      142.251.133.196 80      0
```

Donde, primero se muestra la fecha y hora en formato GMT, luego se muestra la acción que realizó el usuario (si se desconectó o si se conectó), a esto le sigue un mensaje indicando el protocolo de autenticación que utiliza. Se mostrará "Non-auth user" si el usuario se conectó sin ningún método de autenticación y en el caso de que haya optado por conectarse vía el método de autenticación por usuario y contraseña se mostrará el usuario del mismo. Luego se observa el tipo de registro. Luego se mostrarán las IPs (versión 4 o 6) del origen y el usuario junto con los puertos de esta, y por último el status code de la request Socks V5.

Descripción de la servidor de monitoreo

Primero el administrador envía (por medio de la aplicación cliente) un mensaje al servidor para poder autenticarse, este proceso requiere que el administrador ingrese sus credenciales con el siguiente formato:

<code><username>:<password>\n</code>
--

Y luego se le enviara al servidor de la siguiente forma:

USR. LEN	USR	PASS. LEN	PASS
----------	-----	-----------	------

Donde USR. LEN y PASS. LEN representa la longitud del nombre de usuario (USR) y la contraseña (PASS) a enviar. Luego, el servidor recibe este mensaje, corrobora que las credenciales del administrador sean correctas y le responde con un paquete que posee el siguiente formato:

STATUS

El campo de estatus puede obtener 3 valores, 0 en el caso de que el administrador se haya autenticado con éxito, 1 en el caso de que las credenciales sean incorrectas y 2 en el caso de que el mensaje esté mal formulado.

Luego, la aplicación recibe este mensaje y lo analiza, si se logró autenticar despliega el menú de opciones disponibles, en caso no satisfactorio desconecta al cliente. Si se logró autenticar al administrador este podrá enviar comandos al servidor, estos son:

1. Pedir métricas del servidor: devuelve las métricas calculadas durante la ejecución del proxy. El formato del mensaje que representa a este comando es:

<code>1\n</code>

2. Crear un nuevo usuario: el mensaje deberá tener este formato:

<code>2\sUSR\sPASS\n</code>

Donde USR y PASS son, respectivamente, el usuario y la contraseña nueva a registrar en el proxy (separadas por espacios).

3. Listar usuarios: lista los usuarios que fueron registrados en el proxy, formato:

3\n

4. Modificar el buffer: modifica el tamaño del buffer del proxy, formato

4\sBUFF_SIZE\n

Donde BUFF_SIZE es el nuevo tamaño del buffer.

Problema durante desarrollo

- Durante la conexión con el origen a la hora de hacer el connect por un problema de estructuras, que no estábamos manejando correctamente, se nos generó un nudo de problemas que hacía que nos fuera imposible conectarnos con el origen
- Al llegar al estado COPY nos estábamos pisando información de la request, esto se produjo a que trabajamos con una estructura UNION para simplificar el acceso a las máquinas de estado pero después decidimos dejar de usarla y nos olvidamos de reflejar el cambio en la estructura principal de nuestro código. Esto produjo que, dado a que el contenido de los UNION comparten memoria, se generen overlaps y se pisen datos.

Limitaciones de la aplicación

Al saturar el servidor con un proceso bash creado por nosotros llamado **server_destroyer** (el cual se encuentra ubicado en la carpeta Test del directorio principal del proyecto) nuestro servidor soporta altas cantidades de usuarios (probamos con 500) y los procesa correctamente, sin embargo si seguimos incrementando se vuelve más lento.

Extensiones

Para solucionar la limitación anterior estaría bueno ver cómo manejar el proxy mediante hilos para subdividir la carga de tareas y agilizar el proceso, reduciendo así, la cantidad de iteraciones que debe hacer el selector de file descriptores.

Conclusiones

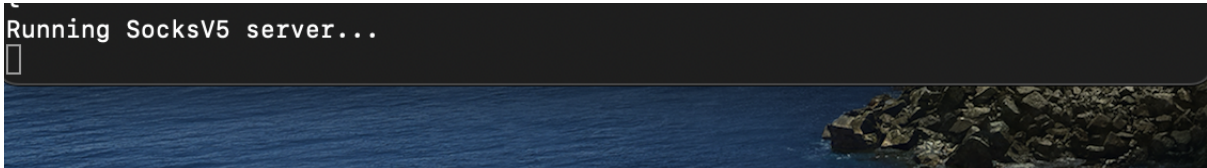
Para finalizar este proyecto hablaremos un poco de lo aprendido. Hoy en día el internet influye mucho en la vida de las personas, detrás de lo que todos perciben como común hay todo un mundo de problemas que pueden surgir

que se fueron solucionando con el paso de los años y el continuo desarrollo de la tecnología. Se logró desarrollar un servidor proxy SocksV5 funcional capaz de redireccionar a miles de personas que busquen navegar por medio de él de forma concurrente y simultánea.

Ejemplos de prueba

Primero corremos el servidor proxy en una terminal, esto lo podemos hacer ubicándonos en el directorio principal del servidor y corriendo el comando `“./socks5d”`, puede correrse `“./socks5d -h”` para ver las distintas configuraciones de ejecución. Ejemplo:

```
Running SocksV5 server...
█
```



En este caso creamos el usuario `“usr:usr”` = usuario:contraseña y al administrador `“admin:admin”` utilizando la misma sintaxis.

Desde otra terminal podemos conectarnos al proxy, esto lo haremos desde herramientas como curl o netcat.

Ejemplo con el comando **curl**:

```
german@MacBook-Pro-de-German ~ % curl -x socks5h://localhost:1080 www.google.com
/humans.txt -U usr:usr
Google is built by a large team of engineers, designers, researchers, robots, and
others in many different sites across the globe. It is updated continuously, a
nd built with more tools and technologies than we can shake a stick at. If you'd
like to help us out, see careers.google.com.
german@MacBook-Pro-de-German ~ % █
```

Desde el proxy podemos comprobar que la conexión y request pudo realizarse correctamente ya que se registró la conexión por medio del servicio de logs. Notemos que como solicitamos algo via un usuario el mismo queda registrado en los logs.

```
Running SocksV5 server...
[2022-06-21 15:08:28] - Client connected succesfully: Non-auth user      A      1
27.0.0.1      62814      142.251.133.196 80      0
[2022-06-21 15:08:28] - Client disconnected: Non-auth user      A      127.0.0.
1      62814      142.251.133.196 80      0
█
```

Ejemplo con el comando **netcat**:

```
german@MacBook-Pro-de-German ~ % nc -X 5 -x localhost:1080 www.google.com 80
GET /humans.txt HTTP/1.1

HTTP/1.1 200 OK
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Type: text/plain
```

De la misma forma desde el servidor se observa:

```
Running SocksV5 server...
[2022-06-21 15:13:16] - Client connected succesfully: Non-auth user      A      0
.0.0.0 62827 142.251.133.196 80 0
```

Y a la hora de desconectarnos del netcat se observa:

```
[2022-06-21 15:14:36] - Client disconnected: Non-auth user      A      0.0.0.06
2827 142.251.133.196 80 0
```

Guia de instalación

Primero nos clonamos el repositorio de Github con el proyecto, el mismo puede encontrarse [aquí](https://github.com/navfran98/pdc-tpe-socksv5.git) (<https://github.com/navfran98/pdc-tpe-socksv5.git>).

Una vez clonado nos posicionamos dentro de la carpeta del mismo y por medio de la terminal corremos el comando:

“make all”.

Luego de esto podremos levantar el servidor corriendo:

“./socks5d <parámetros de configuración>”

El servidor ya se encuentra corriendo y completamente disponible para aceptar conexiones. Si queremos conectarnos como administrador correr:

“./socks5_clt <ip del proxy> <puerto del proxy>”

Luego por pantalla se mostrarán los pasos a seguir para su correcta utilización.

Configuración del servidor

Para configurar el servidor añadir argumentos por línea de comandos a la hora de correr el proxy. Para ver toda la lista de opciones de configuración ejecutar:

```
“./socks5d -h”
```