

# Protocolos de comunicación

## Informe TPE

Grupo 12:

Rodrigo Fera	58079
Florencia Monti	55073
Lucas Gomez	60408
Segundo Espina	60150

# Índice

<b>Descripción del proyecto</b>	<b>3</b>
Hello	3
Autenticación	3
Request	4
Resolución de nombres	5
Connect	5
Response	5
Copy	5
Sniffing	5
Logging	5
Protocolo de monitoreo	6
<b>Problemas encontrados durante el desarrollo</b>	<b>6</b>
<b>Limitaciones de la aplicación</b>	<b>6</b>
<b>Posibles extensiones</b>	<b>7</b>
<b>Conclusiones</b>	<b>7</b>
<b>Ejemplos de prueba</b>	<b>7</b>
<b>Guía de instalación</b>	<b>9</b>
<b>Instrucciones para la configuración</b>	<b>9</b>
<b>Ejemplos de configuración y monitoreo</b>	<b>9</b>
<b>Documento de diseño del proyecto</b>	<b>11</b>

## Descripción del proyecto

El objetivo de este trabajo consiste en la realización de un PROXY-SOCKS v5 el cual posee las siguiente características:

- Capacidad de atender múltiples conexiones concurrentes.
- Soporte autenticación usuario/contraseña [RFC1929]
- Soporte de conexiones a direcciones IPv4, IPv6 o utilizando FQDN que resuelva a cualquiera de esos tipos de direcciones.
- Recolección de métricas que ayuden a monitorear la operación del sistema.
- Mecanismos para el manejo de usuarios y cambios en la configuración en tiempo de ejecución.
- Registro de accesos para cada uno de los usuarios.
- Monitoreo de tráfico y generación de credenciales de acceso para el protocolo POP3.

Las especificaciones del protocolo están descritas en el RFC1928.

La conexión entre el cliente y el origin se realiza en los siguientes pasos:

- **Hello**

Inicialmente se envía un paquete que identifica al protocolo:

VERSION	NMETHODS	METHODS
1	1	1 a 255

El servidor confirma que la versión sea correcta y elige entre uno de los métodos recibidos. Debe informar la respuesta con la siguiente información:

VERSION	METHOD
1	1

En caso de no elegir ninguno de los métodos que el cliente envió, debe retornar '0xFF' como método elegido y el cliente debe cerrar la conexión.

El proxy implementa los métodos:

- '0x00' No se requiere autenticación
- '0x02' Autenticación mediante usuario/contraseña

- **Autenticación**

Si el método elegido fue el '0x02' el cliente debe enviar sus credenciales [RFC1929]

En la negociación inicial, el cliente envía la solicitud con el siguiente formato:

VERSION	ULEN	UNAME	PLEN	PASSWD
1	1	1 a 255	1	1 a 255

Donde:

- El campo versión debe ser '0x01' (es distinta de la versión de SOCKS)
- ULEN: contiene la longitud del nombre de usuario
- UNAME: nombre de usuario
- PLEN: contiene la longitud de la contraseña
- PASSWD: contraseña

El servidor verifica que el nombre de usuario y la contraseña coincidan y envía la respuesta:

VERSION	STATUS
1	1

Si STATUS indica '0x00' la autenticación fue exitosa y un valor distinto de '0x00' indica que hubo una falla debe cerrar la conexión.

#### • Request

Si la autenticación fue satisfactoria o si se eligió el método de no autenticación, el cliente procede a enviar la solicitud:

VERSION	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	0x00	1	Variable	2

Donde:

- VERSION: Es la version de socks (en nuestro caso debe ser '0x05')
- CMD: Es el comando deseado, nuestro proxy solo implementa CONNECT ('0x01')
- ATYP: Es el tipo de IP al que se desea conectar, pudiendo ser IPv4 (0x01), FQDN (0x03) o IPv6 (0x04)
- DST.ADDR: Es la dirección destino
- DST.PORT: Es el puerto destino

El campo ATYP indica el tipo de dirección, lo cual sirve para conocer la longitud de la dirección destino. Teniendo direcciones IPv4 con 4 bytes, IPv6 con 16 bytes y FQDN donde la longitud es indicada en el primer byte del campo DST.ADDR.

- **Resolución de nombres**

Para la resolución de nombres, se realizan consultas DNS y en caso satisfactorio se obtiene una lista de direcciones IP. Estas consultas son realizadas en un hilo separado del servidor, con el objetivo de evitar que este se bloquee mientras se realiza la consulta.

- **Connect**

Al tener la dirección IP (o la lista de direcciones), se intentará resolver la conexión con el origen, intentando conectar con la primera de las direcciones. En caso de fallar continuar con las siguientes de la lista.

Luego se envía la respuesta al cliente informando si la conexión fue exitosa o, en caso de existir, el error que se haya producido.

- **Response**

La respuesta por parte del servidor será de la forma:

VERSION	REPLY	RSV	ATYP	BND.ADDR	BND.PORT
1	1	0x00	1	Variable	2

La versión debe ser la 0x05, en el campo REPLY se indica el estado de respuesta del servidor (ver RFC1928).

Los campos BND.ADDR y BND.PORT tendrán el valor 0x00.

- **Copy**

Una vez que la conexión fue exitosa, se procede a leer la información recibida por el cliente y enviarla al origen o viceversa.

- **Sniffing**

En caso de que la comunicación sea mediante el protocolo POP3, se registra por salida estándar el usuario y la contraseña del cliente. Es posible desactivar esta opción con el flag -N al iniciar el proxy o mediante comandos del protocolo de monitoreo (ver manual).

El servidor sabrá cuando capturar la credenciales ya que la conexión que utilice el protocolo POP3 lo hará en el puerto 110. Además, el primer mensaje será proveniente del origen con el contenido '+OK'. Luego, se buscan los mensajes USER y PASS, los cuales contienen la información deseada.

Los registros tienen primero la fecha (formato ISO'8601), el nombre del usuario (user unknown en caso de no estar autenticado), el protocolo (POP3), el destino (IP o nombre del dominio), puerto destino, usuario descubierto y contraseña descubierta.

- **Logging**

Se registran en salida estándar los LOGS del proxy, en estos se pueden ver las conexiones, una por línea. Donde se especifica la fecha de la conexión (formato

ISO-8601), el nombre del usuario (user unknown si no está autenticado), el tipo de registro ('A'), dirección IP y puerto de origen, y destino y puerto de destino. Para el campo de destino se devuelve la IP, salvo que la conexión se haya hecho mediante DNS, en donde se registra el nombre del dominio.

- **Protocolo de monitoreo**

El protocolo de monitoreo es de texto, orientado a sesión y funciona sobre SCTP. Es utilizado para obtener métricas almacenadas en el proxy (información volátil).

Además, permite cambiar los usuarios almacenados y la activación del disector, como también obtener esta información. Las especificaciones del protocolo de monitoreo están descritas en su respectivo documento.

Para el protocolo de monitoreo se decidió hacer request y replies que terminen con el caracter '\n', y que haya un espacio entre los campos de los mensajes, de esta manera el parseo se facilita, teniendo entre cada espacio el elemento deseado, y sabiendo que el mensaje termina con \n. Los comandos del protocolo están descritos en el respectivo RFC que se encuentra en el mismo proyecto, sin embargo, hay aclaraciones que se deben hacer para su uso en el caso del TP.

Para las pruebas, la contraseña en el caso de autenticación está seteada en el server como "password", por ende a la hora de autenticarse, hay que poner "password" en el campo de la contraseña.

## Problemas encontrados durante el desarrollo

Las dificultades que se plantearon a la hora de desarrollar el proyecto fueron:

- Al encontrarnos en el estado COPY y una de las conexiones fallaban, se debía cerrar dicha conexión. Se barajó la opción de utilizar flags que indiquen si se estaba haciendo read/write en el origin o en el cliente y así saber de qué file descriptor se trataba. Finalmente se optó por analizar el puntero de lectura de dicho buffer y tener un flag que indique si aún queda información por escribir en dicho buffer.
- Existieron algunos problemas con el manejo de los distintos errores para realizar el armado de la response (por ejemplo, host unreachable, command not supported, etc). La solución fue realizar estas consultas al momento de realizar el connect y almacenar el estado que se obtuvo.

## Limitaciones de la aplicación

Algunas de las limitaciones que posee el proyecto son:

- La máxima cantidad de clientes concurrentes se encuentra en 500. Este valor se encuentra limitado por la cantidad de file descriptors disponibles.
-

## Posibles extensiones

- Una posible extensión se encuentra en el protocolo de monitoreo. Al utilizar SCTP se podría implementar multi-homing para el monitoreo de múltiples servidores.
- El almacenamiento de usuarios se realiza mediante un arreglo. Al agregar un usuario nuevo se compara si el nombre de usuario ya existe entre todos los usuarios almacenados. Esto no es escalable si se agrega una cantidad de usuarios (actualmente el máximo es 10). Una posible solución es reemplazar el arreglo por una lista donde se inserten alfabéticamente en función de su nombre, para luego recorrer la lista utilizando algún algoritmo de menor complejidad (por ejemplo búsqueda binaria).
- El protocolo de monitoreo podría tener una mayor cantidad de comandos. Por ejemplo, para variar el tamaño del buffer y así optimizar la comunicación. Además sería más fácil realizar pruebas sin tener que reiniciar el servidor.
- El proxy solo soporta los métodos 'NO AUTHENTICATION REQUIRED' y 'USERNAME/PASSWORD'. Se podrían implementar los demás métodos descritos en el RFC1928
- El proxy solo soporta el comando 'CONNECT'. Se podrían implementar los demás comandos descritos en el RFC1928.

## Conclusiones

Se creó un proxy SOCKSv5 y un protocolo que permite el monitoreo de dicho proxy. Si bien existen muchas mejoras que se podrían implementar, se obtuvo un correcto funcionamiento del proyecto.

Las máquinas de estado fueron de mucha utilidad para entender e implementar correctamente el intercambio de información, ya sea entre cliente, proxy y origin como también dentro del propio servidor.

El trabajo ayudó a comprender mejor cómo se utilizan en la práctica los distintos protocolos vistos en clase. También sirvió para conocer cómo se implementa un protocolo desde cero y realizar correcciones a dicho protocolo a la hora de implementarlo.

## Ejemplos de prueba

Se realizaron los siguientes casos de prueba:

- en el primer caso hicimos una descarga de un archivo iso de 861MB sin el proxy y utilizando sha256sum para verificar la integridad del archivo descargado:

```
[rodrt@fedora] ~$ curl http://br.mirror.archlinux-br.org/iso/2022.06.01/archlinux-2022.06.01-x86_64.iso | sha256sum
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 861M  100 861M    0     0  56.0M    0  0:00:15  0:00:15 --:--:-- 59.5M
6b3bfe8d4e0df82cc3322f9565e92b0c44f27105889a665a8626ce47fbf7ab8 -
```

- luego se realizó la misma descarga pero esta vez utilizando el proxy:

```
[rodrt@fedora] ~$ curl -x socks5h://127.0.0.1:1080 http://br.mirror.archlinux-br.org/iso/2022.06.01/archlinux-2022.06.01-x86_64.iso | sha256sum
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 861M  100 861M    0     0  61.2M    0  0:00:14  0:00:14 --:--:-- 62.5M
6b3bfe8d4e0df82cc3322f9565e92b0c44f27105889a665a8626ce47fbf7ab8 -
```

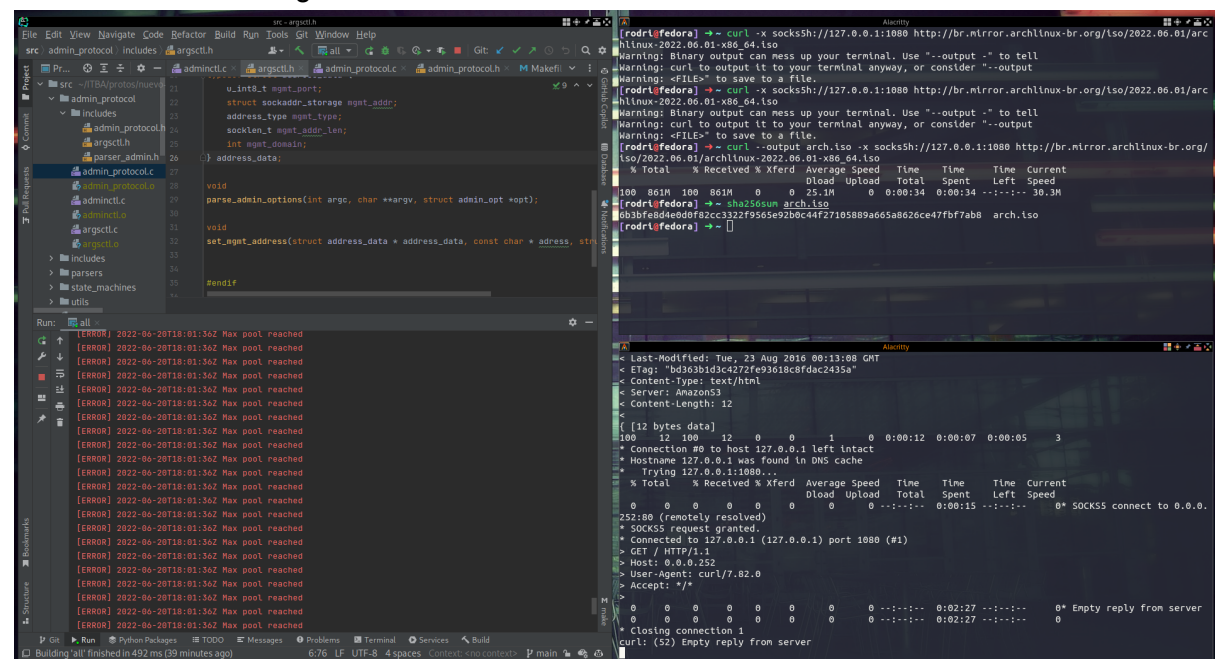
Como podemos observar en ambas descargas se obtuvo el mismo hash por lo que podemos concluir que no se pierde información en ningún caso y se llegó al mismo resultado con y sin el proxy. También podemos observar que el tiempo de descarga fue aproximadamente el mismo para ambos casos.

Luego hicimos la misma prueba pero cortando la descarga en la mitad y reanudandola.

```
[rodolfo@fedora] ~$ curl --output arch.iso -C - -x socks5h://127.0.0.1:1080 http://br.mirror.archlinux-br.org/iso/2022.06.01/archlinux-2022.06.01-x86_64.iso
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
27 861M 27 238M 0 36.9M 0 0:00:23 0:00:06 0:00:17 38.2M^C
[rodolfo@fedora] ~$ curl --output arch.iso -C - -x socks5h://127.0.0.1:1080 http://br.mirror.archlinux-br.org/iso/2022.06.01/archlinux-2022.06.01-x86_64.iso
** Resuming transfer from byte position 264974336
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 608M 100 608M 0 45.7M 0 0:00:13 0:00:13 --:--:-- 45.5M
[rodolfo@fedora] ~$ sha256sum arch.iso
6b3bfe8d4e0df82cc3322f9565e92b0c44f27105889a665a8626ce47fbf7ab8 arch.iso
```

Como podemos observar, tampoco cambia el hash del iso resultante, por lo que podemos concluir que no hubo pérdida de información y se mantuvo la integridad del archivo.

Otro caso de prueba fue descargar un archivo iso y a la vez generar el limite de conexiones soportadas por el proxy para ver cómo se comportaba. Esto se hizo con el comando “seq 1 500 | parallel -j 499 -l{} curl -v -x socks5h://127.0.0.1:1080 foo.leak.com.ar” mientras se realizaba la descarga.



Como podemos observar, una vez alcanzado el límite de conexiones concurrentes el proxy deja de aceptar las conexiones restantes. El tiempo de descarga aumentó considerablemente (antes 14-16 segundos y ahora 34 segundos) pero no se generó ningún error ni se perdió información del archivo en cuestión.

## Guía de instalación

Para poder ejecutar el protocolo de monitoreo correctamente se deberá instalar la librería ‘netinet/sctp.h’. Los pasos para la instalación del proyecto (cliente y servidor) se encuentran en el archivo README.



# Instrucciones para la configuración

En los manuales del servidor se encuentran las distintas opciones configurables.

## Ejemplos de configuración y monitoreo

Para utilizar el protocolo de monitoreo primero hay que autenticarse, luego de recibir el greetings, mediante la contraseña "password":

```
[INFO] 2022-06-20T17:04:13Z +0
GREETINGS

[INFO] 2022-06-20T17:04:13Z Please enter Password to login.
Password:

password█
```

Si la contraseña es correcta, se muestra el siguiente menú:

```
password
[INFO] 2022-06-20T17:05:09Z +1
Welcome Administrator!

[INFO] 2022-06-20T17:05:09Z These are all available commands:
STATS
    Prints useful statistics about the proxy server
CLOSE_CONNECTION
    Disconnects admin client
DISECTOR_ACTIVATION <+/->
    Activates or deactivates the disector
GET_DISECTOR
    Gets disector status
ADD_USER <user:pass>
    Adds a new user to the system
DELETE_USER <user>
    Deletes a user from the system
LIST_USERS
    Lists all users in the system
HELP
    Prints this message
```

Luego, el usuario elige el comando que desea usar:

```
> stats
[DEBUG] 2022-06-20T17:09:22Z +2 1 0 425

[INFO] 2022-06-20T17:09:22Z STATS:
Total conections: 1
Current connections: 0
Total bytes transfered: 425
```

Algunos de los comandos requieren argumentos:

```
> add_user juan:pass
[DEBUG] 2022-06-20T17:06:49Z +6

[INFO] 2022-06-20T17:06:49Z User added
```

Si el servidor responde con un código de error, el mismo es traducido:

```
> add_user juan:gg
[DEBUG] 2022-06-20T17:07:18Z -6 1

[INFO] 2022-06-20T17:07:18Z User already exists
```

Finalmente, se puede cerrar la coneccion:

```
> close_connection
[DEBUG] 2022-06-20T17:24:03Z +3

[INFO] 2022-06-20T17:24:03Z Logging out...
```

# Documento de diseño del proyecto

