Trabajo Práctico Especial

Alvarez Escalante, Tomás (60127)
tomalvarez@itba.edu.ar

Caeiro, Alejo (60692)
acaeiro@itba.edu.ar

Ferreiro, Lucas Agustín (61595)
lferreiro@itba.edu.ar

Gomez Kiss, Roman (61003)
romgomez@itba.edu.ar

Grupo 1

SOCKSv5 Proxy Server - ALPHA Protocol

72.07 Protocolos de comunicación
Instituto Tecnológico de Buenos Aires

Índice

Introducción

El objetivo de este trabajo es implementar un servidor proxy para el protocolo SOCKSv5, que atiende IPv4 e IPv6, de manera multiplexada, y a su vez desarrollar e implementar un protocolo propio para la administración y configuración de dicho servidor. Con este protocolo se pueden consultar ciertas métricas, como cantidad de conexiones desde que se encendió el proxy, bytes transferidos, entre otros.

Protocolos y aplicaciones utilizados

A continuación se detallan los protocolos utilizados a lo largo de todo el desarrollo del trabajo práctico:

SOCKSv5

Como el principal objetivo del trabajo es implementar un servidor proxy para el protocolo SOCKSv5, justamente se implementó el "Protocolo SOCKS versión 5". Para ello tuvimos en cuenta los RFC-1928 (que describe el protocolo) y el RFC-1929 (que describe las opciones de autenticación de usuario y contraseña para con el protocolo). Si cuando se levanta el servidor NO se especifica ningún usuario, entonces la autenticación se encontrará desactivada. En caso contrario, la autenticación se encontrará activada.

Cabe aclarar que la autenticación se podrá configurar en tiempo de ejecución mediante el cliente administrador.

TCP

Se utiliza TCP (RFC-793) como protocolo de transporte del protocolo SOCKSv5, basándonos en lo especificado en el RFC-1928.

UDP

Utilizamos el protocolo UDP (RFC-768) como protocolo de transporte para Alpha Protocol (nuestro protocolo de administración del servidor proxy). La principal motivación de utilizar UDP es que la cantidad de información transportada por las funciones planteadas entran fácilmente en un datagrama UDP, salvando como única excepción a esta regla el listado de la cantidad de usuarios. Otra de las ventajas del uso de UDP es para aprovechar el hecho de que no se necesita mantener una sesión abierta con el servidor administrador.

POP3

Utilizamos el protocolo POP3 (RFC-1939) para generar un registro de credenciales de acceso (usuarios y contraseñas) de forma similar a ettercap, y monitorear/sniffear el tráfico para recolectar métricas que pueden ser de interés.

DNS

Utilizamos el protocolo DNS (RFC-1035) para resolver los pedidos al proxy cuando se recibe un FQDN (Full Qualified Domain Name) mediante el uso de la función *getaddrinfo()*.

ALPHA Protocol

Nuestro protocolo diseñado, desarrollado e implementado por los integrantes del grupo para ofrecer un servicio de administración al servidor proxy SOCKSv5 de manera no bloqueante. ALPHA protocol es un protocolo binario, no orientado a sesión, que corre sobre UDP y que nace con la finalidad de monitorear y recuperar métricas, manejar usuarios y modificar la configuración del server proxy SOCKS5.

Tanto quien levanta el servidor proxy como el cliente administrador deben tener seteada una variable de entorno denominada *ALPHA_TKN*, la cual servirá como clave de acceso al servicio de administración del servidor proxy. Dicho token es unicamente numérico, entre 3 y 7 caracteres.

Como ALPHA Protocol corre sobre UDP, se plantearon dos tipos de headers: uno para manejar los requests al server y otro para manejar los responses del server.

Requests

En estos contamos con los siguientes campos:

- Version (1 byte)
- Command (2 bytes)
- ID (2 bytes)
- Token (4 bytes)
- Data (Unión con los distintos tipos de dato soportados)

Responses

Por otro lado, en los responses, contamos con los campos:

- Version (1 byte)
- Status code (1 byte)
- Command (2 bytes)
- ID (2 bytes)
- Data (Unión con los distintos tipos de dato soportados)

Cabe aclarar que, como se utiliza UDP, se hace uso del valor del campo *ID* para asegurar que la response recibida se corresponda a la request enviada. A su vez, el campo *Token* es usado para verificar que tanto el *ALPHA_TKN* seteado por quien levanto el servidor como por el cliente administrador coincidan.

Problemas encontrados

Dentro de los principales problemas podemos destacar la comprensión de la consigna, el protocolo y las herramientas provistas por la cátedra (como por ejemplo el manejo del selector y de la máquina de estados). Estas cuestiones llevaron a los integrantes del equipo a una ardua tarea de investigación, lectura de RFCs y revisión de las clases grabadas por la cátedra.

Otro problema significativo fue lograr una correcta comprensión de las distintas estructuras provistas por las librerías de sockets, es decir, cuando era correcto usar una estructura u otra.

Con relación a estos problemas, nos resultó muy desafiante y complejo poder dividir las tareas para el desarrollar el trabajo. Esto conllevo a que todos los integrantes del grupo tuvieran que estar presentes intentando resolver la problemática en conjunto, haciendo uso de la herramienta Visual Studio Code Liveshare.

También nos encontramos con el problema de no logramos soportar nombres de usuario con una longitud mayor a 5 caracteres, cuando el tamaño máximo establecido es de 128. Todos los integrantes del grupo intentamos debugear para encontrar el problema, pero no pudimos lograr nuestro cometido. De todas formas, dicha implementación funciona (para nombres de usuario de hasta 5 caracteres) y se plantea su solución como una posible extensión del proyecto.

Otro problema surge al momento de utilizar la herramienta JMeter para realizar pruebas de stress con nuestro proxy. No logramos configurar JMeter correctamente, lo cual conllevo a que no podamos realizar dichas pruebas.

Por último, otro problema enfrentado en gran parte de la toma de decisiones del proyecto, fue como implementar nuestro propio protocolo para el monitoreo del proxy.

Limitaciones y posibles extensiones

Una de las limitaciones más claras surge al momento de utilizar la syscall *pselect()*, lo que permite que unicamente se puedan monitorear como máximo 1024 file descriptors. De esta forma, una posible extensión del proyecto sería hacer uso de *poll()*, tal como lo recomienda el propio manual de programación de Linux. De esta forma podría aumentarse significativamente la cantidad de conexiones concurrentes soportadas por el servidor proxy.

Una segunda limitación es que el servidor no cumple con todos los requisitos detallados en la sección 3 y 4 del RFC-1928, ya que por ejemplo no se provee soporte para otros tipos de autenticación (como GSSAPI).

Otra limitación consiste en que las métricas recolectadas por el servidor no son persistentes, es decir, se pierden cuando se 'apaga' el servidor. El acto de persistir dichas métricas conllevaría un mayor dificultad de implementación y por eso no fue realizada (pues nos encontramos con un tiempo acotado para desarrollar el trabajo) y se deja planteada como una posible extensión del proyecto.

Por último, se decidió limitar los tamaños de los buffers a 4096 para no sobrecargar la memoria del sistema.

Una posible extensión sería guardar información más completa de todos los paquetes que son transportados por el servidor proxy SOCKSv5, diferenciando también aquellos que fueron sniffeados con POP3 y aquellos que fueron solicitados por algún usuario particular con acceso al servidor, de forma tal que mediante el uso del cliente administrador se pueda acceder a un registro de todos estos datos. Actualmente, dicha información se imprime en la terminal donde se levantó el servidor proxy, pero esta no está siendo almacenada para un futuro acceso.

Conclusiones

El desarrollo de todo el proyecto significó un gran desafío conceptual desde el primer momento, el cual fue acompañado con una gran cantidad de aprendizaje, revisión y entendimiento de implementación de los diversos protocolos mencionados anteriormente. De esta forma se obtuvo como resultado un servidor proxy SOCKSv5 que recolecta e imprime por consola información útil para los posibles administradores de este y brindarles la posibilidad de configurarlo en tiempo de ejecución.

Guía de instalación, configuración y monitoreo

Instrucciones para la instalación

Para correr el proyecto hay que posicionar en la carpeta root y correr:

```
make all

```

Dicho comando generará dos ejecutables: *./socks5d* y *./alpha_manager*.

Instrucciones para la configuración

El primero de ellos se utiliza para levantar el servidor proxy SOCKSv5. Dicho ejecutable cuenta con los siguientes flags:

```
-h                Imprime la ayuda y termina
-l <SOCKS addr>   Direccion IPv4 o IPv6 donde servira el servidor
-L <conf addr>    Direccion IPv6 o IPV6 donde servira el cliente
-p <SOCKS port>   Puerto entrante conexiones SOCKS
-P <conf port>    Puerto entrante conexiones configuracion
-u <name>:<pass> Usuario y contrasenia que puede usar el proxy
-v                Imprime informacion sobre la version y termina
-N                Deshabilita el sniffing

```

Por defecto, este ejecutable se encarga de inicializar los sockets pasivos IPv4 e IPv6 en las siguientes direcciones y puertos:

- 0.0.0.0 y ::0 con puerto 1080 para el servidor proxy SOCKSv5 (TCP).
- 127.0.0.1 y ::1 con puerto 8080 para el cliente administrador (UDP).

Cabe declarar, que si el proyecto se corre en pampero, se tiene que cambiar el puerto entrante de las conexiones SOCKS, ya que por defecto el puerto 8080 de pampero se encuentra en uso.

Por último, el ejecutable *./alpha_manager* se utiliza para acceder a la shell de administración del proxy. Dentro de esta, se encuentran disponibles los siguientes comandos:

```
help                 Imprime todos los comandos
list <page>          Lista los usuarios. Maximo 25 por pagina.
hist                 Devuelve cantidad de conexiones historicas
conc                 Devuelve cantidad de conexiones concurrentes
bytes                Devuelve cantidad de bytes transferidos
add <user:password>      Agrega un usuario
del <user>           Elimina un usuario
```

```
        auth-on             Habilita la autenticacion
        auth-off            Desabilita la autenticacion
        checkauth           Devuelve el estado de la autenticacion
        sniff-on            Habilita el sniffing
        sniff-off           Desabilita el sniffing
        checksniff          Devuelve el estado del sniffing

```

Debug del código

Se hizo uso de valgrind, pvs-studio, cppcheck y -fsanitize=address para realizar análisis dinámico y estático del código. Para realizar dicho análisis basta con posicionarse en la carpeta root del proyecto y correr:

```
        make check

```

Ejemplos de configuración

```
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d
Check that the environment token ALPHA_TKN exists
romgomez@Dell-Roman:~/TPE-PDC/src$ export ALPHA_TKN=01234567
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d
Token must be between 3 and 7 characters
romgomez@Dell-Roman:~/TPE-PDC/src$ export ALPHA_TKN=alpha
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d
ALPHA_TKN must be a numeric token
romgomez@Dell-Roman:~/TPE-PDC/src$ export ALPHA_TKN=1234
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d
INFO: Listening on TCP port 1080
INFO: Opened TCP IPv4 socket (0) for SOCKSv5 with address 0.0.0.0

INFO: Listening on TCP port 1080
INFO: Opened TCP IPv6 socket (3) for SOCKSv5 with address 0::0

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv4 socket (4) for manager with address 127.0.0.1

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv6 socket (5) for manager with address ::1
```

Figura 1: Ejemplo de configuración de *ALPHA_TKN* para levantar el servidor proxy SOCKSv5 y los posibles errores que este puede dar.

```
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d -v
SOCKSv5 version: 1.0
ITBA Protocolos de Comunicación 2022/2 -- Grupo 1
ALPHA PROTOCOL
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d -h
Usage: ./socks5d [OPTION]...

   -h               Imprime la ayuda y termina.
   -l <SOCKSaddr>   Dirección IPv4 o IPv6 donde servirá el proxy SOCKS.
   -L <mng-addr>    Dirección IPv6 o IPV6 donde servirá el servicio de administrador.
   -p <SOCKS-port>  Puerto entrante conexiones SOCKS.
   -P <mgn-port>    Puerto entrante conexiones configuracion
   -u <name>:<pass> Usuario y contraseña de usuario que puede usar el proxy. Hasta 10.
   -v               Imprime información sobre la versión y termina.
   -N               Deshabilita el sniffing.
```

Figura 2: Ejemplo de uso de los flags -v y -h para mostrar la versión y la lista de comandos del servidor proxy SOCKSv5.

```
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d -l 127.0.0.1 -L 127.0.0.2 -P 9999 -p 8080
INFO: Listening on TCP port 8080
INFO: Opened TCP IPv4 socket (0) for SOCKSv5 with address 127.0.0.1

INFO: Listening on TCP port 8080
INFO: Opened TCP IPv6 socket (3) for SOCKSv5 with address 0::0

INFO: Listening on UDP port 9999
INFO: Opened UDP IPv4 socket (4) for manager with address 127.0.0.2

INFO: Listening on UDP port 9999
INFO: Opened UDP IPv6 socket (5) for manager with address ::1
```

Figura 3: Ejemplo de uso de los flags -l, -L, -p y -P para cambiar los valores default de las direcciones IP y puertos.

```
romgomez@Dell-Roman:~$ netstat -nlp | grep socks5d
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN      4869/./socks5d
tcp6       0      0 :::8080                 :::*                    LISTEN      4869/./socks5d
udp        0      0 127.0.0.2:9999          0.0.0.0:*                           4869/./socks5d
udp6       0      0 ::1:9999                :::*                                4869/./socks5d
```

Figura 4: Uso de netstat para verificar lo anterior.

```
romgomez@Dell-Roman:~/TPE-PDC/src$ ./socks5d -u roman:gomezkiss
INFO: Listening on TCP port 1080
INFO: Opened TCP IPv4 socket (0) for SOCKSv5 with address 0.0.0.0

INFO: Listening on TCP port 1080
INFO: Opened TCP IPv6 socket (3) for SOCKSv5 with address 0::0

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv4 socket (4) for manager with address 127.0.0.1

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv6 socket (5) for manager with address ::1
```

Figura 5: Ejemplo de uso del flag -u para agregar un usuario.

```
lferreiro@WIN10PC:~/Protos/TPE-PDC$ ./src/socks5d -u lucas:12345
INFO: Listening on TCP port 1080
INFO: Opened TCP IPv4 socket (0) for SOCKSv5 with address 0.0.0.0

INFO: Listening on TCP port 1080
INFO: Opened TCP IPv6 socket (3) for SOCKSv5 with address 0::0

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv4 socket (4) for manager with address 127.0.0.1

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv6 socket (5) for manager with address ::1

INFO: ----------------------------------------------------
INFO: New connection created from 127.0.0.1:47004 in socket 6
INFO: Connection from 127.0.0.1:47004 closed
INFO: ----------------------------------------------------
INFO: ----------------------------------------------------
INFO: New connection created from 127.0.0.1:47006 in socket 6
INFO: Resolv FQDN information
INFO: Resolution of FQDN success
INFO: Connect to 142.250.219.14:80 from fd 552 to fd 396 in progress
INFO: Connected to 142.250.219.14:80 successfully
INFO: Connection from 127.0.0.1:47006 closed
INFO: ----------------------------------------------------
```

```
lferreiro@WIN10PC:~/Protos/TPE-PDC$ curl -x socks5h://localhost:1080 google.com
curl: (7) Unable to receive initial SOCKS5 response.
lferreiro@WIN10PC:~/Protos/TPE-PDC$ curl -x socks5h://lucas:12345@localhost:1080 google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
lferreiro@WIN10PC:~/Protos/TPE-PDC$
```

Figura 6: Ejemplo de uso de curl con usuario agregado en la imagen anterior.

Ejemplos de monitoreo

```
lferreiro@WIN10PC:~/Protos/TPE-PDC$ export ALPHA_TKN=12345
lferreiro@WIN10PC:~/Protos/TPE-PDC$ ./src/socks5d
INFO: Listening on TCP port 1080
INFO: Opened TCP IPv4 socket (0) for SOCKSv5 with address 0.0.0.0

INFO: Listening on TCP port 1080
INFO: Opened TCP IPv6 socket (3) for SOCKSv5 with address 0::0

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv4 socket (4) for manager with address 127.0.0.1

INFO: Listening on UDP port 8080
INFO: Opened UDP IPv6 socket (5) for manager with address ::1
```

```
lferreiro@WIN10PC:~/Protos/TPE-PDC$ export ALPHA_TKN=123
lferreiro@WIN10PC:~/Protos/TPE-PDC$ ./src/alpha_manager 127.0.0.1 8080
Alpha manager client $ bytes
Error: Authentication failed. Check ALPHA_TKN.
Alpha manager client $
```

Figura 7: Ejemplo de como falla el cliente administrador si no establecimos correctamente el mismo token que quien levanto el servidor proxy.

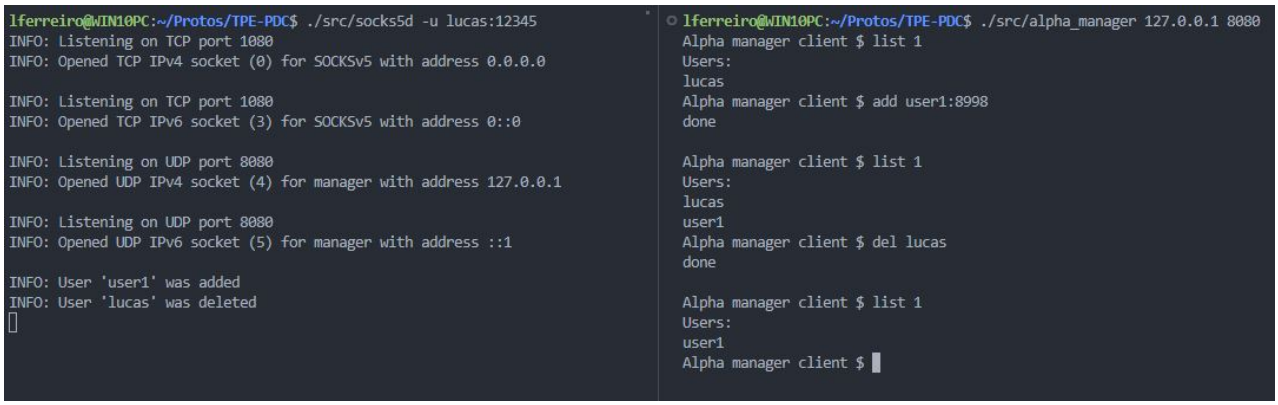

Figura 8: Ejemplo de uso de las funcionalidades para listar, agregar y eliminar usuarios.

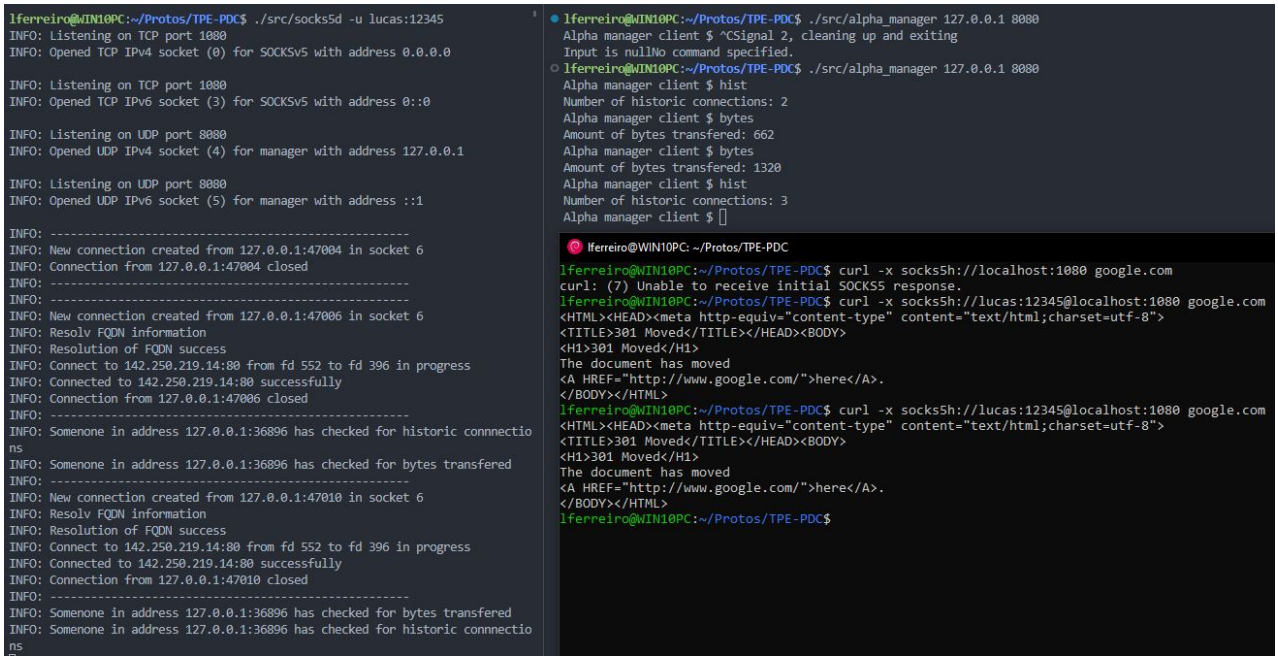

Figura 9: Ejemplo de uso de las funcionalidades para ver la cantidad de bytes transferidos y la cantidad de conexiones historicas.

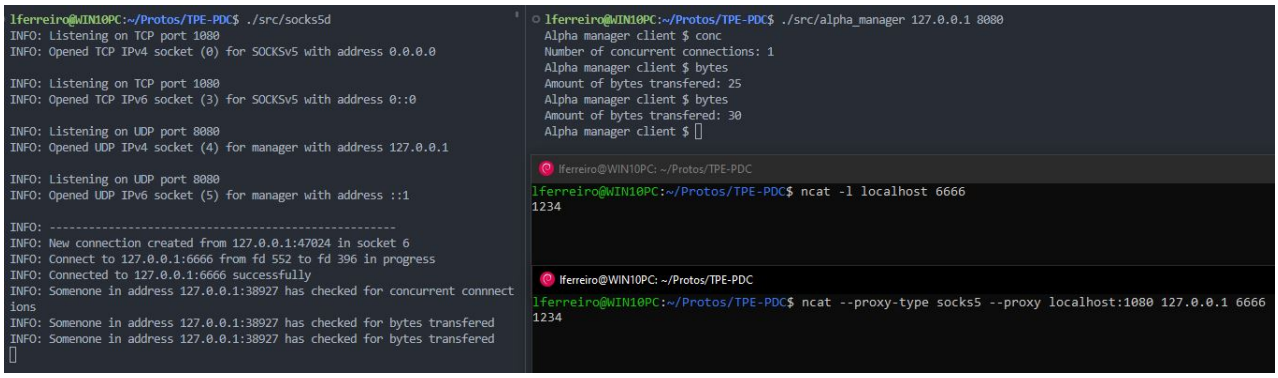

Figura 10: Ejemplo de uso de las funcionalidades para ver la cantidad de conexiones concurrentes. Adicionalmente, se hace uso de 'bytes' para contar con exactitud los bytes transferidos.

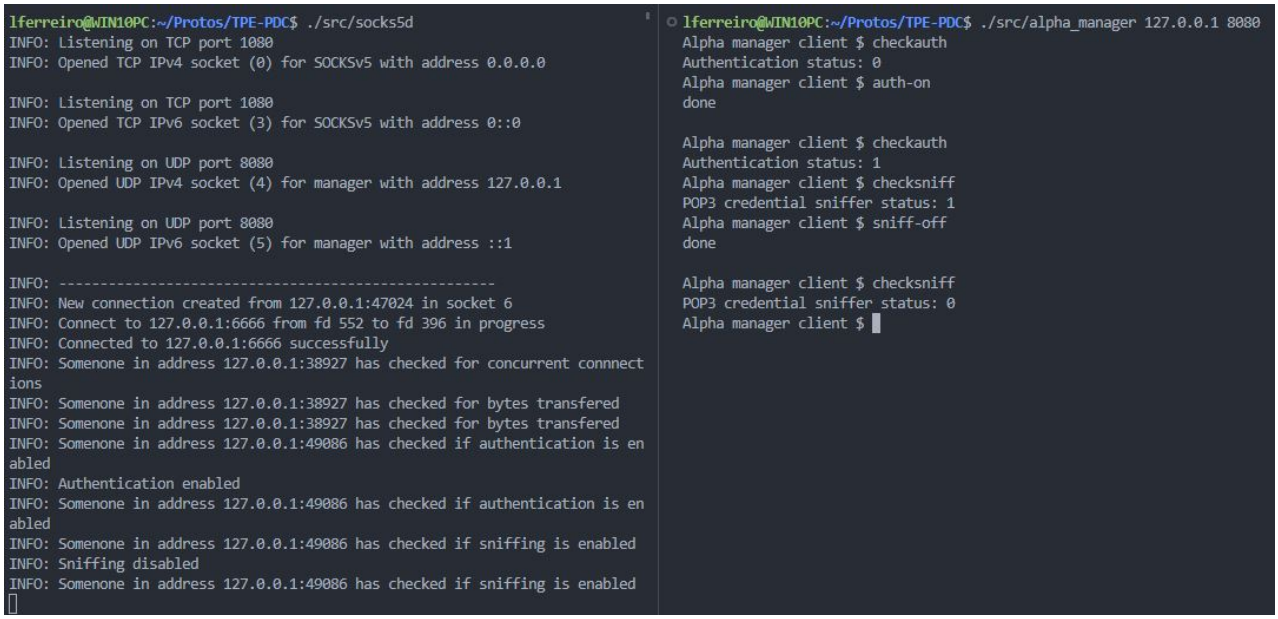

Figura 11: Ejemplo de configuración para la autenticación y el sniffing.

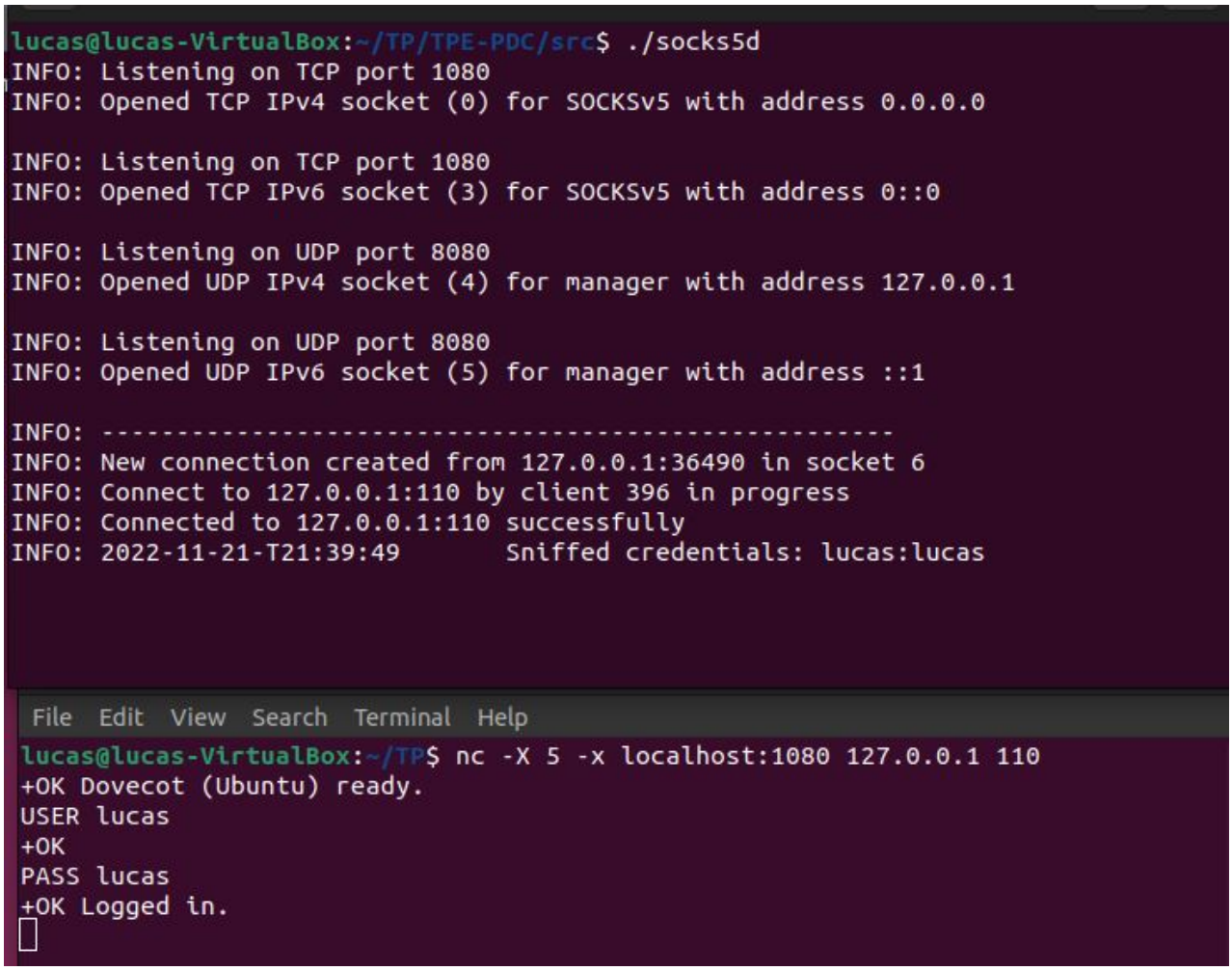

Figura 12: Ejemplo de como el proxy sniffea credenciales de paquetes POP3.

Ejemplos de prueba

Pruebas de performance

En las siguientes imágenes se pueden observar los resultados obtenidos de las pruebas de performance realizadas, en las cuales se muestra primero un pedido realizado con el proxy activo, y luego un pedido sin el proxy activo al solicitar imágenes de distinto tamaño.

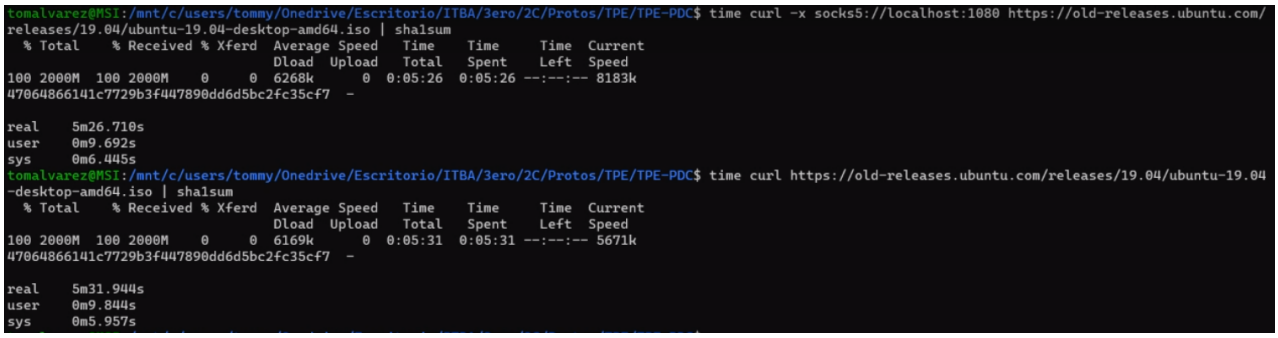

Figura 13: Ejemplo de testeo del proxy con una imagen .iso de 2.2GB

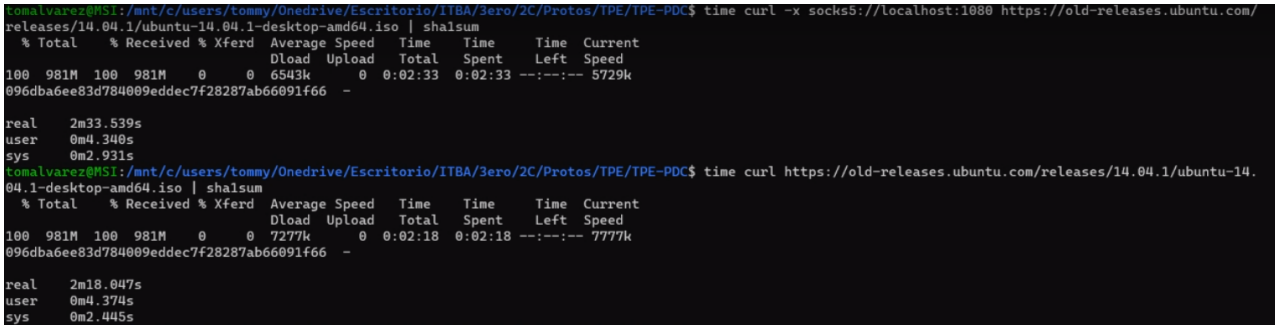

Figura 14: Ejemplo de testeo del proxy con una imagen .iso de 981MB

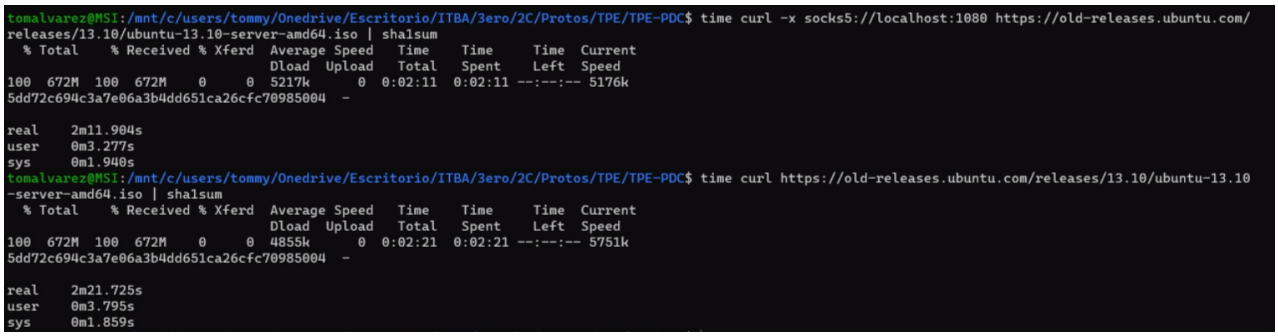

Figura 15: Ejemplo de testeo del proxy con una imagen .iso de 672MB

Nótese que en la totalidad de los casos, el uso de proxy si bien garantizo la integridad de los datos (ver los valores de cada checksum), la velocidad de los pedidos utilizando el proxy es relativamente similar a cuando este no se encuentra encendido.

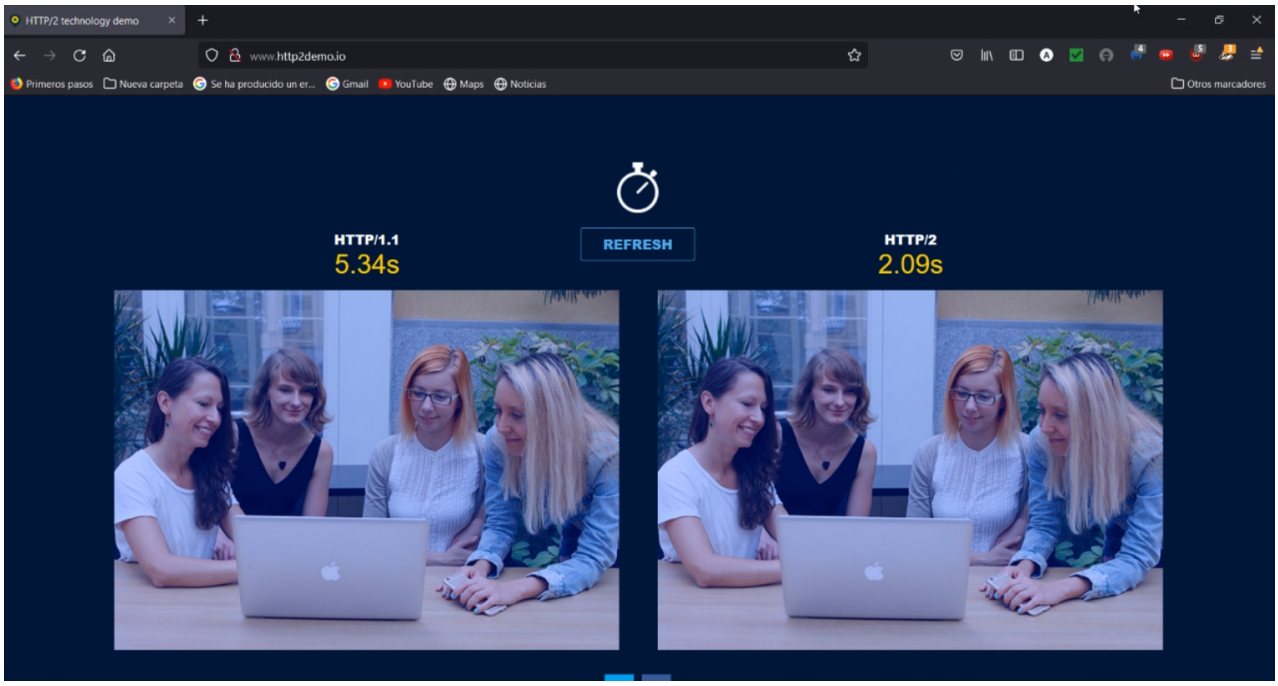

Figura 16: Ejemplo de testeo sin proxy activo con http://www.http2demo.io/ visto en clase

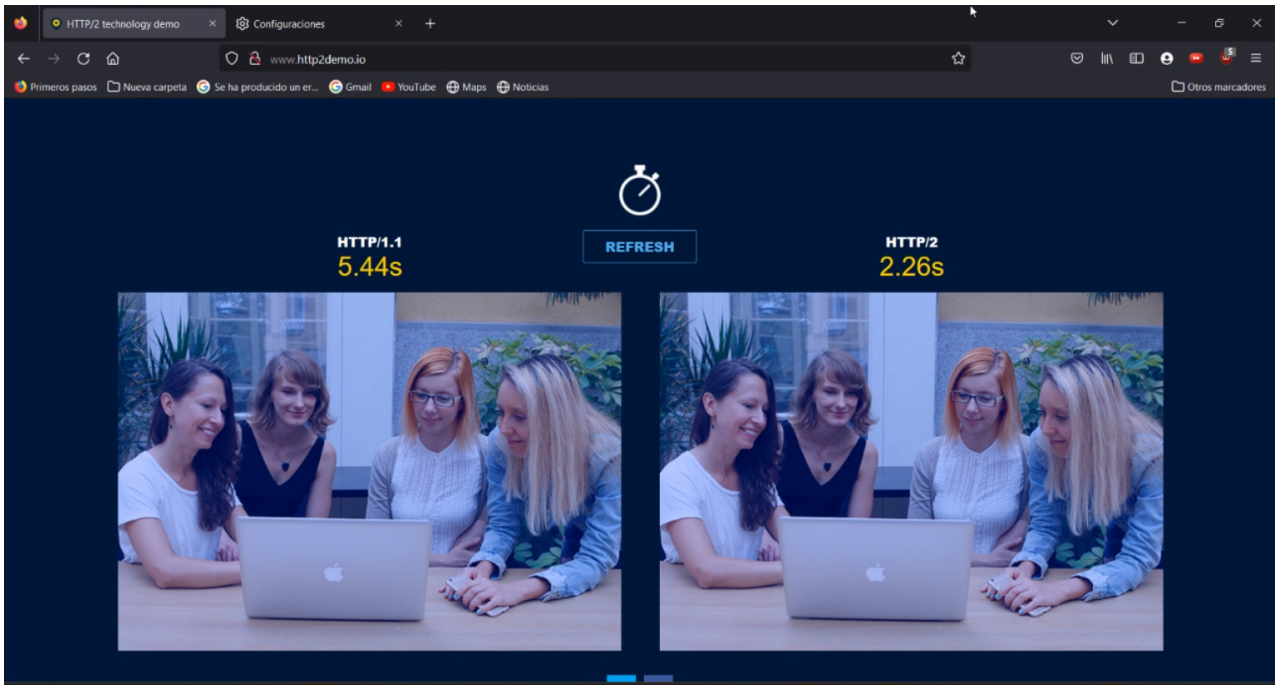

Figura 17: Ejemplo de testeo con proxy activo con http://www.http2demo.io/ visto en clase

Nótese aquí también como en el caso en el cual el proxy no está activo y siendo usado por el browser (Firefox) la velocidad del pedido tanto HTTP como HTTP2 es

relativamente similar a cuando el pedido se realiza utilizando el proxy activo en el browser.

Diseño del proyecto

El servidor consiste de un solo hilo de ejecución (salvo para la resolución de nombres), que ejecuta la syscall *pselect()*, quien será la encargada de monitorear los file descriptors y saber en qué momento estos están disponibles para operaciones de I/O. Para ello, a dichos file descriptors se les asoció la información del cliente correspondiente, ya que el servidor es concurrente.

De esta forma, se cuenta con una máquina de estados "stm"que brinda la posibilidad de realizar diferentes acciones dependiendo del estado en que se encuentra la comunicación con el cliente. Cada estado describe las acciones que se deben llevar a cabo en el caso de recibir información o tener que mandar información, donde dichas operaciones de escritura escriben todo lo permitido hasta vaciar los respectivos buffers.

La conexión entre el cliente y el servidor proxy se realiza basándose en las distintas etapas definidas en el RFC-1928. De esta forma, primero se envía un paquete "HELLO" de identificación del protocolo y de los métodos de autenticación del usuario. Así, si la autenticación se encuentra encendida, solo se permitirán accesos al servidor con autenticación y el próximo paso sería que el cliente envíe sus credenciales para autenticarse.

Finalmente, si la autenticación fue exitosa o esta se encontraba apagada, el cliente envía un pedido "REQUEST" de conexión y el servidor intenta crear la conexión con el servidor destino especificado. En este punto, el servidor proxy puede tomar dos caminos:

- Si se solicita una dirección IPv4 o IPv6, se intentará establecer la conexión.
- Si se especifica un FQDN (Full Qualified Domain Name), se hace uso de la función *getaddrinfo()* en un hilo diferente (pues es bloqueante) para resolver el nombre y tras ello se intentará establecer la conexión.

Una vez realizada la conexión con el destino, todos los paquetes serán enviados a través del proxy. Adicionalmente, el servidor proxy cuenta con un sniffer de credenciales POP3 (que por default se encuentra encendido), el cual parsea los paquetes que pasan por el proxy e imprime las credenciales sniffeadas (si estas son correctas).

Finalmente, cuando el cliente o el servidor proxy cierra la conexión, se liberan todos los recursos.