Junio, 2022

Protocolo de monitoreo proxy SOCKSv5[RFC1928]

Resumen:

Este RFC describe el protocolo de monitoreo del proxy SOCKSv5[RFC1928] del trabajo especial de la materia Protocolos de Comunicación para la cursada del primer cuatrimestre del año 2022.

Tabla de contenidos:

1. Introducción:

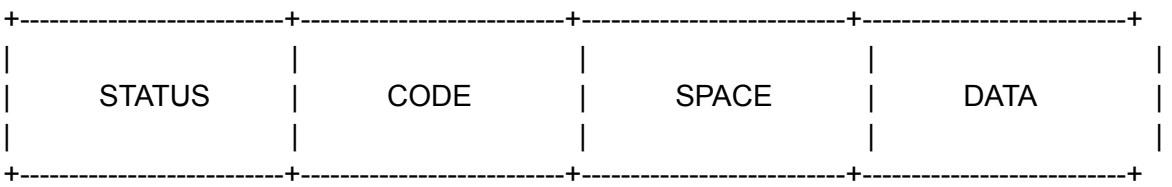
El siguiente es un protocolo de nivel aplicación basado en SCTP[RFC4960]. El cual será utilizado para el monitoreo de las estadísticas y usuarios de un servidor proxy SOCKSv5[RFC1928]. Es un protocolo de texto, por ende se envían y reciben los caracteres ascii. Al inicio tiene un proceso de autenticación y una vez autenticado, el cliente puede enviar comandos al servidor.

2. Consideraciones generales:

Para todos los requests, las respuestas satisfactorias comienzan con el caracter '+', las respuestas de error son iniciadas con el carácter '-', son seguidos por el código de estado, luego un único espacio (' ') seguido por la data (argumentos o respuestas). La data es terminada con un salto de línea. La estructura de las requests será la siguiente:

Dónde code indica el comando, space es uno o más espacios y args los argumentos, terminado con un ‘\n’.

La estructura de las respuestas será la siguiente:



Dónde status indica si la respuesta es correcta o hubo un error, code indica a que request le responde, siendo este el número del comando al que le responde, space es un espacio, y data es la data que se devuelve dependiendo del comando, terminado con un ‘\n’. La data puede o no estar presente, cada comando lo especifica.

El progreso de la sesión en el protocolo pasará por los siguientes estados:

- **GREETING → 0**
- **AUTHORIZATION → 1**
- **COMMANDS → 2 al 8**

Los Comandos (code) son los siguientes:

0 - greeting
1 - authorization
2 - stats
3 - close_connection
4 - disector_activation
5 - get_disector
6 - add_user
7 - delete_user
8 - list_users

3. Operaciones básicas / greeting y auth state:

Por defecto, el servidor de monitoreo escuchará en el puerto 8080 y en la interfaz de loopback. Cuando un cliente desea utilizar el servicio, establece una conexión SCTP con el servidor, el cual enviará un mensaje de greeting, que envía “+0 \n”:

S: +0 \n

Luego, espera a que el cliente envíe la autenticación mediante el uso de una contraseña. Deberá enviar un mensaje con el valor ‘1’ seguido de un espacio y luego la contraseña:

C: 1 <password>\n

La contraseña será comparada con la almacenada en el servidor en búsqueda de coincidencias y se enviará la respuesta correspondiente:

S: +1 \n En caso de coincidencia
S: -1 \n En caso de no encontrar la contraseña

En caso de recibir “-1 \n” el cliente puede continuar la negociación enviando otras contraseñas para intentar autenticarse.

Tras recibir el mensaje ‘+1’ el cliente procede a enviar los comandos deseados. Los comandos consisten en un número ascii y de ser necesario es seguida por argumentos. Los comandos deben ser seguidos de un único espacio y luego por el argumento.

4. Comandos

Una vez que el cliente está autorizado, podrá enviar los siguientes comandos:

- **“2 \n”**
  Son las estadísticas del servidor, retorna ‘+2’ y las métricas almacenadas en el servidor en una nueva línea divididas por un espacio entre cada una.

  Retorna:
  +2 <total_connections> <current_connections> <total_bytes_transferred>\n

  Donde <total_connections> <current_connections> y <total_bytes_transferred> están compuestos por caracteres uno o más caracteres [0-9].
  Retorna ‘-2 \n’ en caso de existir un error.

- **“3 \n”**
  Cierra la conexión con el servidor. Te deja deslogueado para la próxima conexión.
  Retorna “+3 \n” y cierra en caso de funcionar bien. Retorna “-3 \n”

- **“4 <arg>\n”**
  Cambia la activación del disector. el arg puede ser ‘+’ o ‘-’, si el disector ya se encuentra en el estado que se pide, se mantiene en ese estado
  Retorna ‘+4 \n’ en el caso satisfactorio y ‘-4 \n’ en caso de que se produzca un error.

- **“5 \n”**

  Pide el status del disector.

  Retorna “+5 <status>\n”. Retorna “-5 \n” en caso de error.

  Status puede ser ‘+’ en caso de activado y ‘-’ en caso de desactivado

- **“6 <user:password>\n”**

  Agrega un usuario y retorna ‘+6 \n’ en caso de agregarlo correctamente. Importante, ni el usuario ni la contraseña pueden tener los caracteres ‘\n’ ni ‘:’.

  En caso de no ser agregados correctamente se devuelve ‘-6 <data>\n’. <data> en este caso puede tener 3 valores:

  - 0 si hubo un error en el servidor
  - 1 si el usuario ya existe
  - 2 si la cantidad de usuarios es la máxima
  - 3 si el usuario o la contraseña contienen ‘:’ o ‘\n’. También si el usuario o la password son demasiado largas (máximo 255 bytes cada uno). O si no se especifica contraseña.

- **“7 <user>\n”**

  Elimina el usuario indicado y retorna ‘+7 \n’ si la operación fue exitosa. Retorna ‘-7 <data>\n’ en caso de error.

  data puede ser:

  - 0 si hubo un error en el servidor
  - 1 si el usuario a eliminar no existe

- **“8 \n”**

  Pide la lista de usuarios.

  Retorna:

  +8 \n<user1> <user2> … <usern>\n

  Retorna ‘-8 \n’ en caso de error

5. Ejemplo de comunicación:

S: +0 \n
C: 1 pass\n (contraseña incorrecta)
S: -1 \n
C: 1 password\n (contraseña correcta)
S: +1 \n
C: 2 \n
S: +2 1 0 425\n (total connections, current connections, total bytes)
C: 6 juan:hola\n
S: +6 \n (se creó el usuario ‘juan’ con contraseña ‘hola’)

C: 7 pepe\n (usuario no definido)
S: -7 \n
C: 7 juan\n
S: +7 \n