Sistemas Operativos
Trabajo Practico N°3: Sockets

Grupo 5:

Alberto Abancens 62581
Gonzalo Rossin 60135
Uriel Mihura 59039

15/11/2021

1. Introducción

En este trabajo practico se debió completar los desafíos propuestos por la cátedra a través del programa *server* y por medio de ingeniería inversa, replicar su funcionamiento y desafíos.

2. Desafío 1

El primer desafío consistió en establecer una conexión con el servidor propuesto por la cátedra, para ello se tuvo que desarrollar un cliente que abra un socket en el puerto 8080 para conectarse al servidor, una vez conectado, el cliente se comunica con el servidor mediante la syscall *write*.

Respuesta: entendido

3. Desafío 2

El segundo desafío introdujo un número y un capítulo de una serie (The Wire S01E05) donde en cierto punto del capítulo se explica el algoritmo usado para decodificar un número sin sentido.

El algoritmo consiste en tomar el numero recibido y saltar a través del 5 al numero real. Es decir, si el numero que nos dan es un 1, saltando a través del 5 deberíamos tomar el 9, si nos dieran el 4 deberíamos tomar el 6,etc. En el caso particular de que el numero que nos dieran fuera el 5, deberíamos tomar el 0 y viceversa.

El desafío nos presenta el numero 5295 888 6288 que al aplicarle el algoritmo se obtiene el 0810 222 4822 que se corresponde con el número de teléfono del itba.

Respuesta: *itba*

4. Desafío 3

El tercer desafío introdujo un link a una imágen en blanco, para resolverlo decidimos probar diversos programas para cambiar los colores de la imagen.

Respuesta: *M4GFKZ289aku*

5. Desafío 4

El cuarto desafío muestra por pantalla un error común al escribir en un fd que no existe (EBADF), por lo que bastó con hacer un strace del server para encontrar en qué file descriptor se estaba queriendo escribir, y luego con redireccionar ese FD a uno útil (por ejemplo el de salida estándar), se encuentra la respuesta.

Respuesta: *fk3wfLCm3QvS*

6. Desafío 5

El quinto desafío muestra por pantalla la leyenda "respuesta = strings:277". Para hallar la respuesta hubo que correr el programa *strings* sobre el server cuya linea 277 retornaba la respuesta.

Respuesta: *too_easy*

7. Desafío 6

Para el sexto desafío pudimos encontrar la respuesta al ejecutar lo siguiente: *readelf -S server*. Donde se obtuvo la respuesta entre .comment y .shstrtab.

Respuesta: *RUN_ME*

8. Desafío 7

Este decía "filter error", el cual nos indicaba que había algo con respecto a la salida de error. Luego lo que se hizo es redireccionar el FD de error (2) a un archivo, dejando así en salida estándar la respuesta.

Respuesta: *K5n2UFfpFMUN*

9. Desafío 8

En este desafío pudimos observar que no se dio ningun tipo de información. Sin embargo, vimos que se encontraba un rectángulo de distinto color dentro de la pantalla del desafío. Al copiar el rectangulo obtuvimos lo siguiente: "La respuesta es BUmyYq5XxXGt".

Respuesta: *BUmyYq5XxXGt*

10. Desafío 9

Para este desafío se resolvió la expresión provista.

Respuesta: u^v

11. Desafío 10

El título del desafío nos reveló la información que necesitamos para resolverlo, citando a wikipedia: "Quine es un programa que produce su código fuente como su única salida". Conseguimos la respuesta al crear un archivo quine.c que cumpla ese objetivo.

Respuesta: *chin_chu_lan_cha*

12. Desafío 11

Para este desafío bastó con hacer un attach de gdb al server, y al retornar la syscall de *getpid*, reemplazar eax con 0x12345678.

Respuesta: *gdb_rules*

13. Desafío 12

Para este desafío se procedió a tomar todos los datos y fabricar un histograma en función de estos. En dicho histograma se puede observar que los datos presentan una distribución normal.

Respuesta: *normal*

14. Easter Eggs

Lo primero que hicimos al descargar el archivo Server, fue abrirlo con un txt. ahí vimos el primer easter egg, el animal extraño que pensé que era un caballo pero me dicen que es una vaca, por las ubres.

Luego de este descubrimiento vimos la lista de lo que parecían ser hashes, el cual solo el primero encontramos un hit, 'entendido'. Pues claro que reemplazamos este hash por todos los demás, creando así un archivo aparte, "server_hackeado", que por más risa que nos dió pasar volando todos los desafíos en el primer día, nos terminó siendo bastante útil para debuggear el nuestro o comparar desafíos, etc.

Luego se escuchó un rumor con algo de Valgrind, el cual nos llevó al segundo Easter Egg 'VALGRIND_ES_GENIAL'. Finalmente, al estar implementando

el Desafío 6, se encontró que para lograr ese comportamiento había que agregarle __attribute__((section(".CUSTOM_SECTION"))) a una función. Lo que nos hizo preguntar 'a cual función se la habrán agregado estos tipos?'. Media sinapsis mas tarde recordamos el nombre de la sección, '.RUN_ME'
facepalm
Claramente al ejecutar esta función, habría algún Easter Egg, pero lamentablemente no pudimos lograr ejecutarla. Talvez nos estemos equivocando, pero esto nos huele a Easter Egg. Parece que pasaremos hambre estas pascuas.