

Stack Buffer Overflow

Hacking Ético

©Ismael Ripoll &
Hector Marco

Universidad Politècnica de València

February 1, 2022

Índice

1 Objetivos

2 Programa vulnerable

3 Actividades

Objetivos

- ➔ Conseguir ejecutar un shell en la máquina de un compañero.
- ➔ Integrar en el exploit de la práctica anterior con la programación de sockets.
- ➔ Ver que la ejecución remota es solo un paso más en la explotación.

Programa vulnerable (I)

- ➔ Bájate de Poliformat el programa echo.c.

```
/*
 * echo.c: servidor trivial de echo.
 *
 * Usage: echo [port]
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <resolv.h>
#include <arpa/inet.h>
#include <errno.h>
#include <unistd.h>
#include <strings.h>

#define MAXBUF      1024

struct sockaddr_in self, client_addr;
int clientfd, sockfd, size;
```

Programa vulnerable (II)

```
int addrlen=sizeof(client_addr);

int vulnerable (int fd){
    char buffer[32];
    /*---Echo back anything sent---*/
    size = recv(fd, buffer, MAXBUF, 0);
    send(clientfd, buffer, size, 0);
}

int main(int argn, char *argv[]){
    int optval = 1;
    if (2 != argn ){
        printf("Usage: echo [port]\n");
        exit(-1);
    }
    if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0 ) {
        perror("Socket");
        exit(errno);
    }
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEPORT, &optval, sizeof(
        optval));
}
```

Programa vulnerable (III)

```
bzero(&self, sizeof(self));
self.sin_family = AF_INET;
self.sin_port = htons(atoi(argv[1]));
self.sin_addr.s_addr = INADDR_ANY;
if ( bind(sockfd, (struct sockaddr*)&self, sizeof(self)) != 0 ) {
    perror("Bind");
    exit(errno);
}
if ( listen(sockfd, 20) != 0 ) {
    perror("Listen");
    exit(errno);
}
system("echo -n 'Soy: ' ; id -un\n");
printf("Esperando conexiones al puerto: %d\n", atoi(argv[1]));
while (1) {
    clientfd = accept(sockfd, (struct sockaddr*)&client_addr,
                     &addrlen);
    printf("%s:%d connected\n", inet_ntoa(client_addr.sin_addr),
          ntohs(client_addr.sin_port));

    /* Lee y retorna del socket cliente */
}
```

Programa vulnerable (IV)

```
        vulnerable(clientfd);

        shutdown(clientfd, SHUT_RDWR);
        printf("%s:%d bye bye\n", inet_ntoa(client_addr.sin_addr),
               ntohs(client_addr.sin_port));
    }
    return 0;
}
```

- ➡ Compilalo con los siguientes flags:

```
$ gcc -ggdb -m32 -fno-stack-protector -fno-pie -no-pie
    echo.c -o echo
```

- ➡ Deshabilita el ASLR cada vez que lo ejecutes:

```
$ setarch x86_64 -R ./echo 9000
```

Programa vulnerable (V)

- ➔ Observa que se utiliza una opción (atributo) del socket no estandar que permite reutilizar el puerto nada más cerrarlo. Esto nos permite poder utilizar el programa echo muchas veces seguidas. De lo contrario tendríamos que esperar unos varios minutos o utilizar otro puerto.

Actividades (I)

- 1 Estudia el funcionamiento del programa.
- 2 Identifica el fallo.
- 3 Construye un payload que manifieste el fallo. Tienes que conseguir un crash.
- 4 Ajusta el payload para poner el valor `0x44444444` en la dirección de retorno.
- 5 Haz que `echo` termine llamando a `exit()`. Busca la dirección con `objdump`.
- 6 Haz que llame a `system()` (Puedes utilizar la instrucción `callq` que hay en `main()`).

Actividades (II)

- 7 Ahora queda por pasar el argumento a `system()`. Busca en la pila la dirección de `buffer` y colócala en el payload para que `system` la pueda utilizar.

Consejo 1

No utilices directamente la dirección de `buffer` porque las funciones que se ejecutan después la sobre escribirán, pero puedes dejar lo que quieres pasar a `system()` **“después de la dirección de retorno”**. Esto te garantiza que no será sobre escrito.

Actividades (III)

Consejo 2

Las posiciones de la pila **dependen de las variables de entorno**. Cuantas más variables de entorno haya, más abajo estará. Por tanto, **dependiendo de como ejecutes echo, las posiciones cambiarán**. Puedes averiguar la dirección modificando el código de echo para que lo imprima, o usar el gdb (**attacheandote al proceso**), o usar `ltrace`; lo que prefieras.