

# Craft

## 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

- Python code review
- Linux enumeration
- GIT
- Python eval injection
- Vault SSH

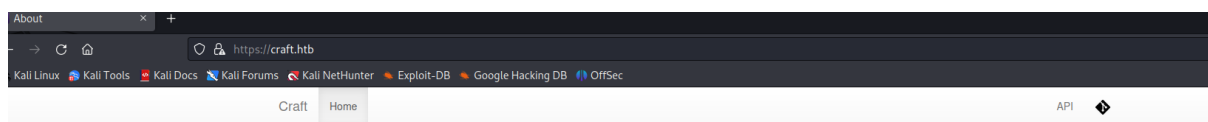
## Exploitation

As always we start with the nmap to check what services/ports are open

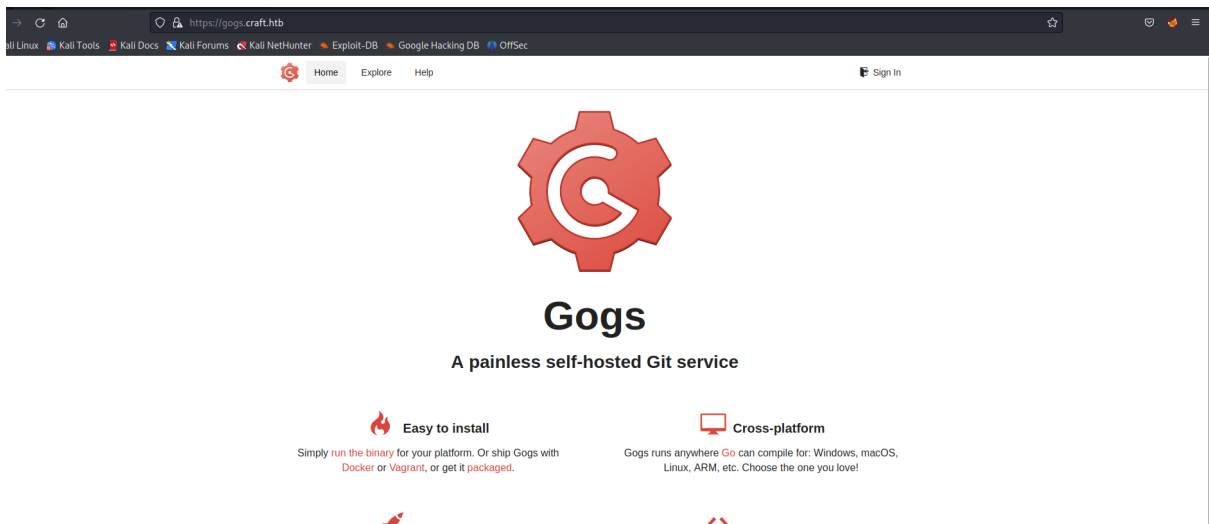
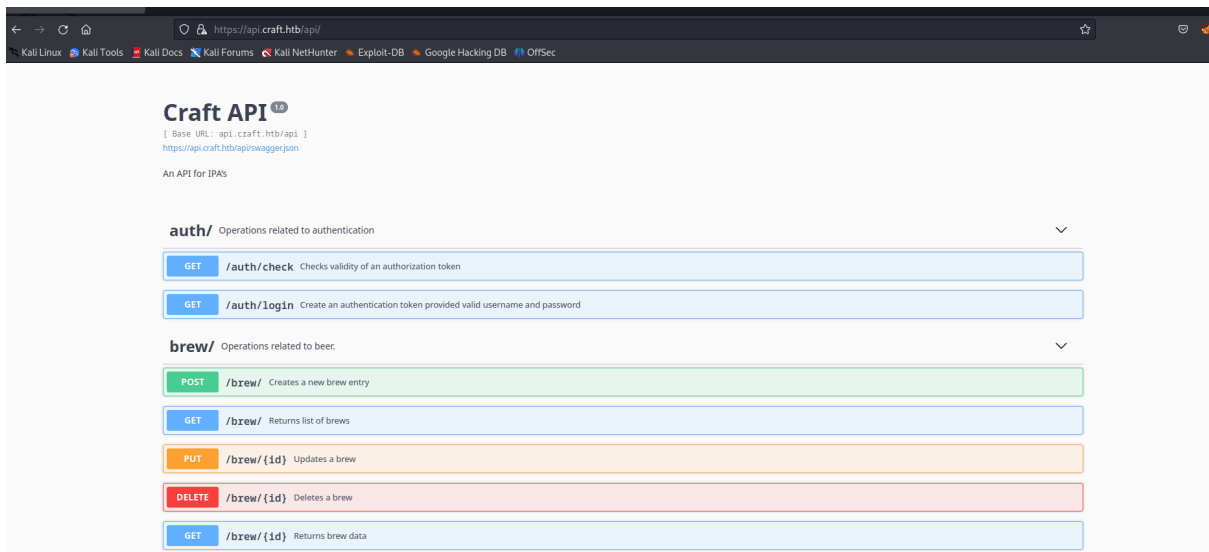
```
└─# nmap -A 10.10.10.110
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-11 06:06 EDT
Stats: 0:03:31 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 06:09 (0:00:00 remaining)
Nmap scan report for 10.10.10.110
Host is up (0.088s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
|_ ssh-hostkey:
|_   2048 bde76c22817adb3ec0f0731df3af7765 (RSA)
|_   256 82b5f9d1953b6d800f3591862db3d766 (ECDSA)
|_   256 283b2618ecdfb336859c27548d8ce133 (ED25519)
443/tcp    open  ssl/http nginx 1.15.8
|_ http-title: About
|_ tls-nextprotoneg:
|_   http/1.1
|_ 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
|_ tls-alpn:
|_   http/1.1
|_ ssl-date: TLS randomness does not represent time
|_ http-server-header: nginx/1.15.8
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.93%E=4%D=8/11%OT=22%CT=1%CU=35647%PV=Y%DS=2%DC=T%G=Y%TM=64D6091
OS:1%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=108%TI=Z%CI=Z%II=I%TS=8)OPS
OS:(O1=M53CST11NW7%O2=M53CST11NW7%O3=M53CNNT11NW7%O4=M53CST11NW7%O5=M53CST1
OS:1NW7%O6=M53CST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN
OS:(R=Y%DF=Y%T=40%W=7210%O=M53CNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=0%A=S+F=A
OS:S%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%A=Z%F=R%O=0%RD=0%Q=)T5(R
OS:=Y%DF=Y%T=40%W=0%S=Z%A=S+F=AR%O=0%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%Z%F
OS:=R%O=0%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+F=AR%O=0%RD=0%Q=)U1(R=Y%DF=N%
```

We see only two ports open, so let's start from the web ports

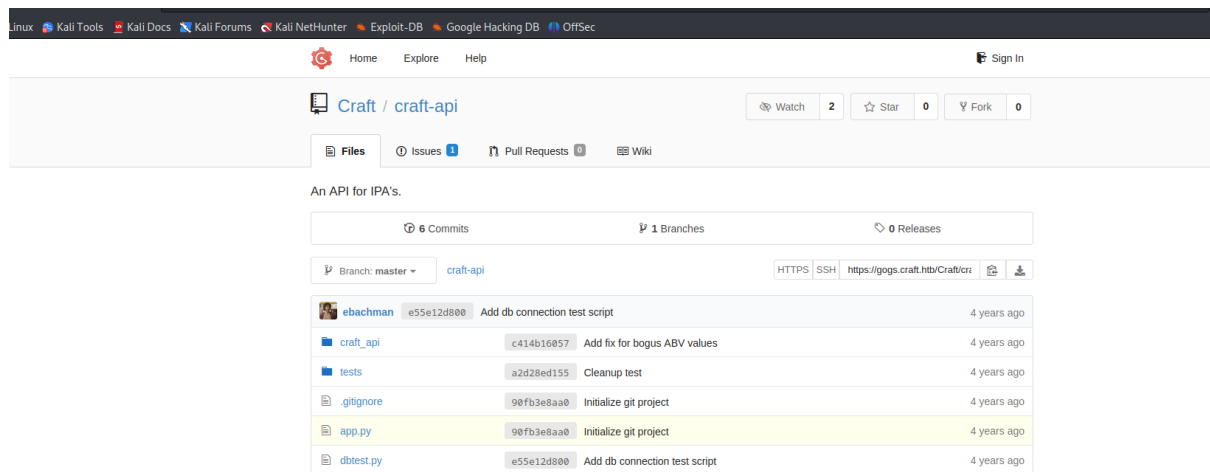
Opening the browser have us the following page



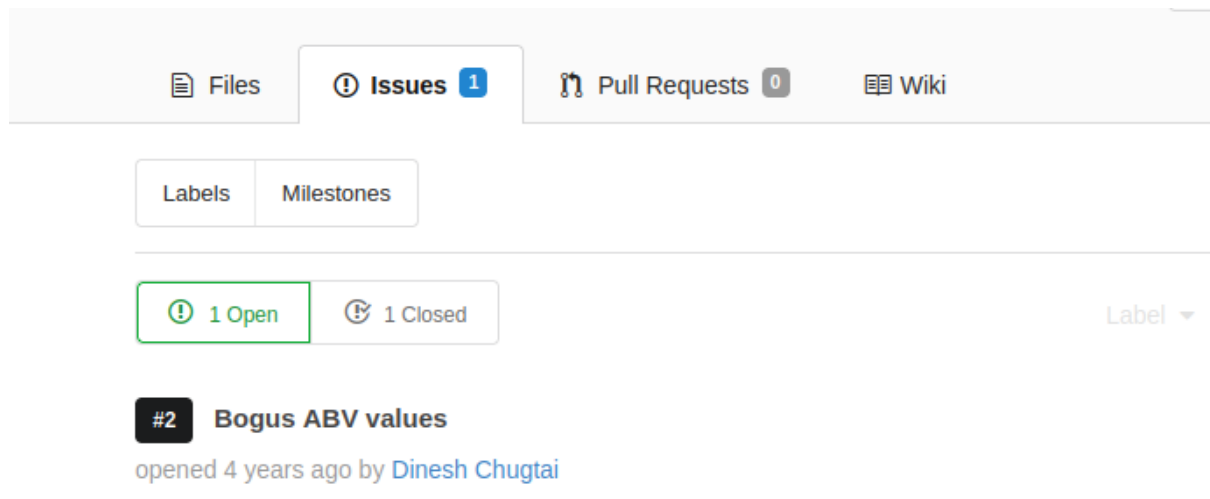
From that page we can access API docs as well as GOGS



There is not much we can do with API right now, so we started exploring GOGS where we found a python source code of the application



And there was something in the Issuess tab,



Closer look of the issue informed as about problem with the “ABV” which is taken from the user and that user authentication is handled by the JWT token

A screenshot of a GitHub pull request conversation. The conversation consists of five comments from different users, each with a profile picture and a role label (Owner or Collaborator). The comments discuss a potential security issue related to ABV values in a database. The first comment shows a curl command that sends a JSON payload with a 'bullshit' name and a '15.0' ABV value. The second comment asks if the user can test it later. The third comment says 'Sure, will push a fix shortly...'. The fourth comment says 'Fix is live and seems to be working :)' and includes a commit hash 'c414b16057'. The fifth comment says 'I fixed the database schema so this is not an issue now.. Can we remove that sorry excuse for a "patch" before something awful happens?'. On the right side, there is a sidebar with 'Labels' (No Label), 'Milestone' (No Milestone), 'Assignee' (No assignee), and '3 Participants' (three profile pictures). At the bottom, there is a button that says 'Sign in to join this conversation.'

Dinesh Chugtai commented 4 years ago

It's possible to add bogus ABV values to the database. For instance, a brew with 15.0 ABV... can we add a check to make sure ABV is sane before writing to the DB?

```
irew/ --data '{"name":"bullshit","brewer":"bullshit", "style": "bullshit", "abv": "15.0"}'
```

Erich Bachman commented 4 years ago Owner

Can you do it yourself and commit? I'll test it later.

Dinesh Chugtai commented 4 years ago Collaborator

Sure, will push a fix shortly...

Dinesh Chugtai commented 4 years ago Collaborator

Fix is live and seems to be working :)

[c414b16057](#)

Bertram Gilfoyle commented 4 years ago Collaborator

I fixed the database schema so this is not an issue now.. Can we remove that sorry excuse for a "patch" before something awful happens?

[Sign in to join this conversation.](#)

Inspecting the problematic code showed us that the ABV value is taken from the user and not sanitised and also put in the EVAL function, (eval is a dangerous function because abused can lead to the remote code execution)

A screenshot of a code diff in a pull request. The diff shows changes to the file 'craft\_api/api/brew/endpoints/brew.py'. The diff is split into two sections: a red section for deletions and a green section for additions. The red section shows the removal of a 'create\_brew' function. The green section shows the addition of a new 'create\_brew' function that includes a check for the ABV value. The check uses the 'eval' function to evaluate the ABV value, which is a security vulnerability. The diff also shows the addition of a new route for the 'create\_brew' endpoint.

1 changed files with 7 additions and 3 deletions

Split View Show Diff Stats

View File

```
@@ -38,9 +38,13 @@ class BrewCollection(Resource):
38 38     """
39 39     Creates a new brew entry.
40 40     """
41 - create_brew(request.json)
42 - return None, 201
43 +
44 + # make sure the ABV value is sane.
45 + if eval('%s > 1' % request.json['abv']):
46 +     return "ABV must be a decimal value less than 1.0", 400
47 + else:
48 +     create_brew(request.json)
49 +     return None, 201
50
51 @ns.route('/<int:id>')
52 @api.response(404, 'Brew not found.')
```

We continued the enumeration of the repository and in the history of commits we found credentials for a user Danesh

Cleanup test [Browse Source](#)

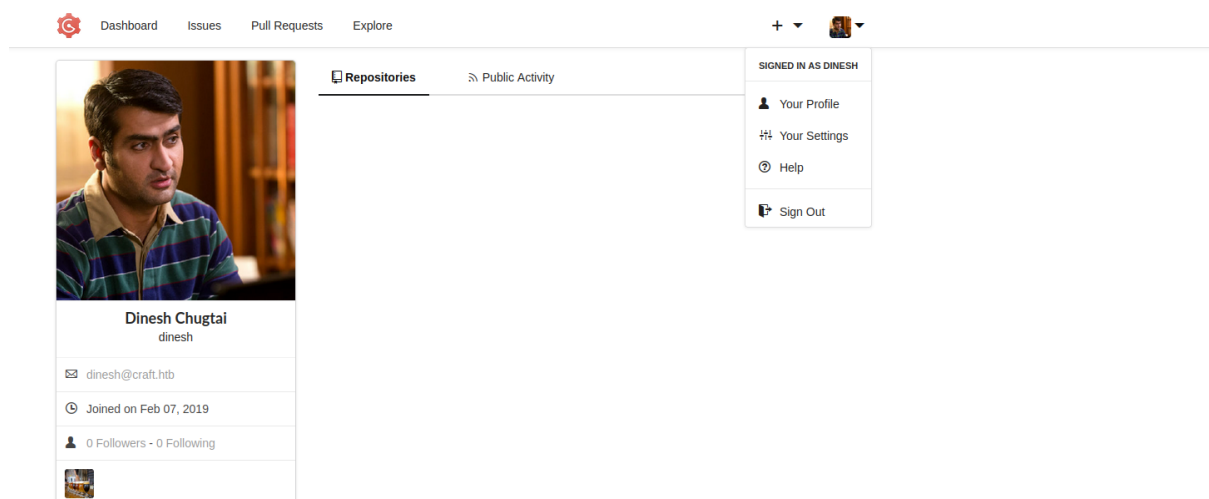
 dinesh 4 years ago parent 10e3ba4f0a commit a2d28ed155

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 @@
3 import requests
4 import json
5
6 -response = requests.get('https://api.craft.htb/api/auth/login', auth=('dinesh', 'aUh0A8PbVJxg'), verify=False)
7 +response = requests.get('https://api.craft.htb/api/auth/login', auth=('', ''), verify=False)
8 json_response = json.loads(response.text)
9 token = json_response['token']
```

We logged as this user to the GOGS



But it didn't give us any new accesses

Yet, as we remembered the authentication to the application is handled by JWT token, so we returned to the API and with Danesh credentials we generated the new JWT token

/auth/login

Create an authentication token provided valid username and password

Parameters

No parameters

Execute

Clear

Responses

Response content typeapplication/json

Curl

curl -X GET "https://api.craft.htb/api/auth/login" -H "accept: application/json"

Request URL

https://api.craft.htb/api/auth/login

Server response

CodeDetails

200

Response body

{  
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IHNvbiJkaWY0ZGlzZXhwIjoxNjE1MTIwMC5kbG91bnRlcnQyLnRvdGVudC0uMmumNrjFn71UR5kvDFFCXrWU"  
}

Download

Response headers

API offers PUT method to update values, including the value of ABV (which we know is vulnerable to remote code execution due to presence of dangerous function and unsanitized user's input)

PUT

/brew/{id} Updates a brew

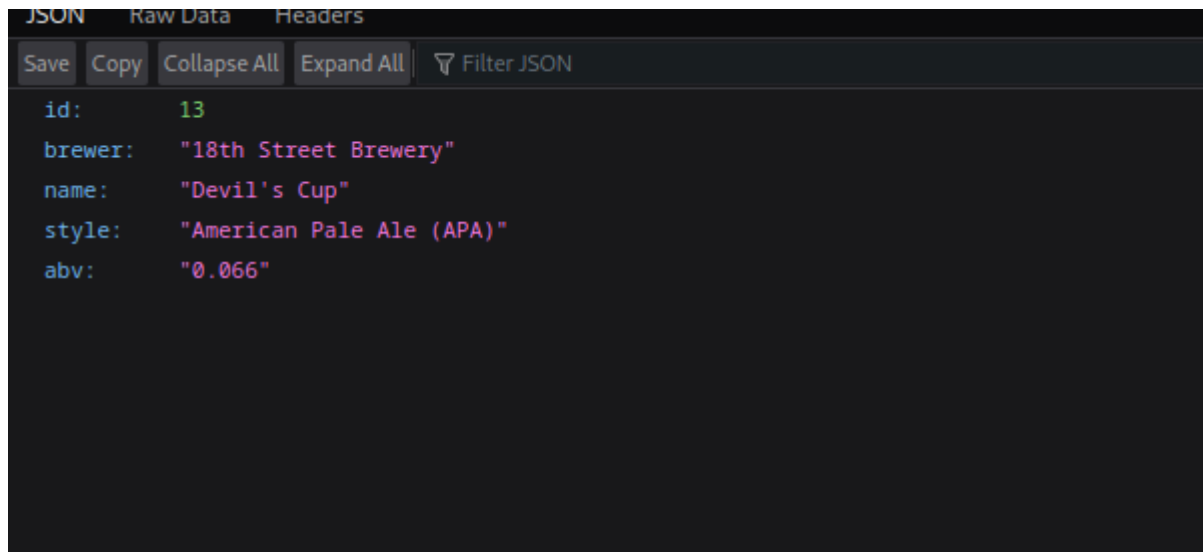
Parameters

Cancel

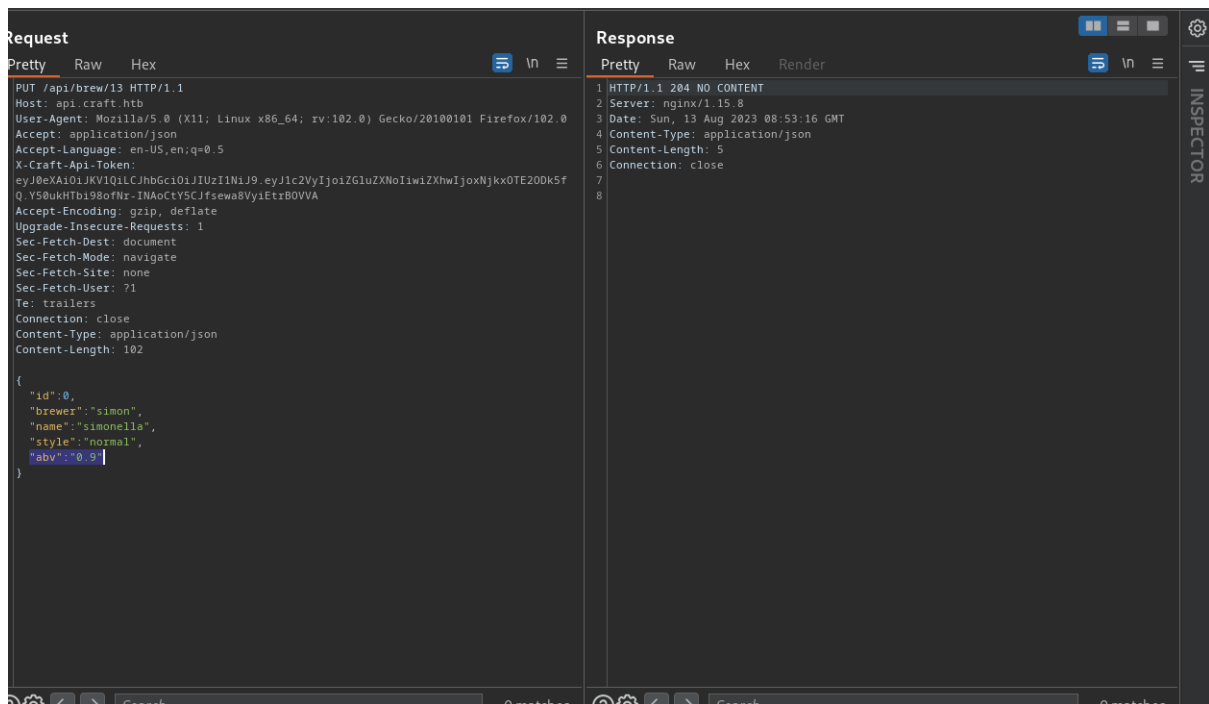
Name	Description
<b>payload</b> <span>required</span> (body)	<div>Edit Value   Model</div> <pre>{   "id": 0,   "brewer": "string",   "name": "string",   "style": "string",   "abv": "string" }</pre> <div>Cancel</div> <div>Parameter content type</div>

<b>Curl</b>	
<pre>curl -X PUT "https://api.craft.htb/api/brew/1" -H "accept: application/json" -H "Content-Type: application/json" -d '{"id": 0, "brewer": "string", "name": "string", "style": "string", "abv": "string"}'</pre>	
<b>Request URL</b>	
<a href="https://api.craft.htb/api/brew/1">https://api.craft.htb/api/brew/1</a>	
<b>Server response</b>	
<b>Code</b>	<b>Details</b>
<i>Undocumented</i>	
TypeError: NetworkError when attempting to fetch resource.	
<b>Responses</b>	
<b>Code</b>	<b>Description</b>
204	<i>Brew successfully updated.</i>
404	<i>Brew not found.</i>

The original value for the brew is as follow

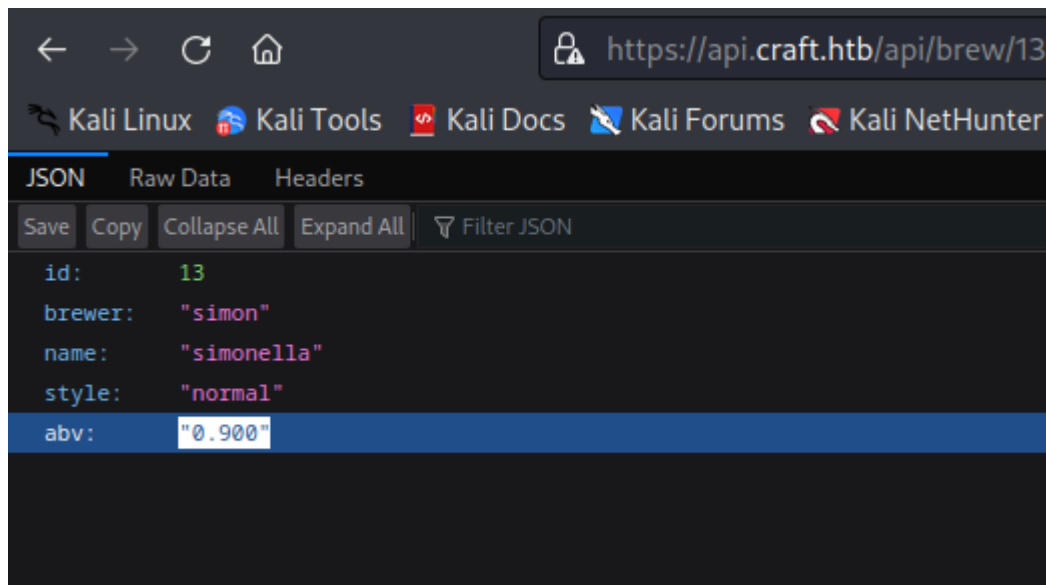


We sent the request to update this value (our request contains forged JWT token to ensure that we sent the request as Danesh)

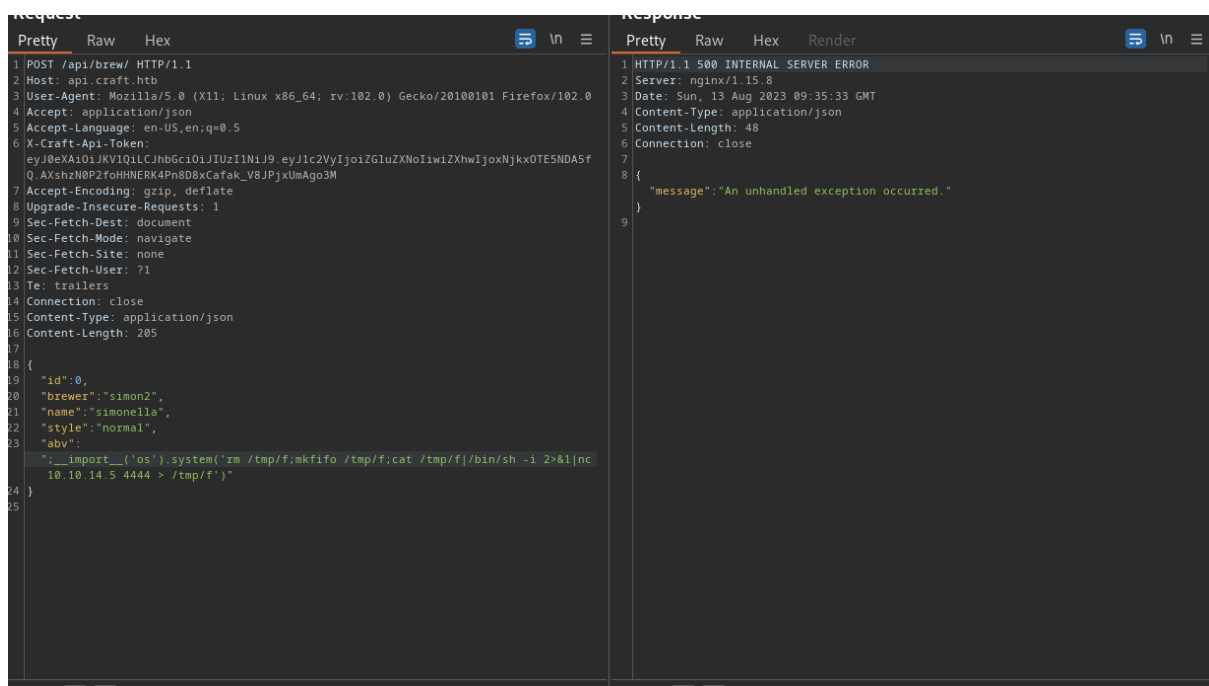


And our malicious request was accepted - the values were updated

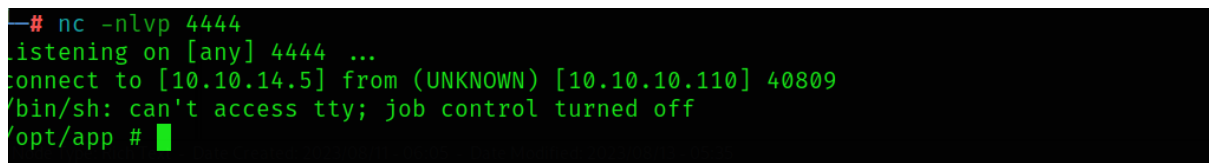




The next step was to get a reverse shell on the system, by passing a malicious command as the ABV value



And we got on the system



```
WXR-xr-x 1 root root 4096 Feb 10 2019 .
WXR-xr-x 1 root root 4096 Feb 10 2019 ..
WXR-xr-x 1 root root 0 Feb 10 2019 .dockerenv
WXR-xr-x 1 root root 4096 Feb 6 2019 bin
WXR-xr-x 5 root root 340 Aug 12 13:43 dev
WXR-xr-x 1 root root 4096 Feb 10 2019 etc
WXR-xr-x 2 root root 4096 Jan 30 2019 home
WXR-xr-x 1 root root 4096 Feb 6 2019 lib
WXR-xr-x 5 root root 4096 Jan 30 2019 media
WXR-xr-x 2 root root 4096 Jan 30 2019 mnt
WXR-xr-x 1 root root 4096 Feb 9 2019 opt
-r-xr-xr-x 169 root root 0 Aug 12 13:43 proc
WX----- 1 root root 4096 Feb 9 2019 root
WXR-xr-x 2 root root 4096 Jan 30 2019 run
WXR-xr-x 2 root root 4096 Jan 30 2019 sbin
WXR-xr-x 2 root root 4096 Jan 30 2019 srv
-r-xr-xr-x 13 root root 0 Aug 12 13:43 sys
WXRwxrwt 1 root root 4096 Aug 13 09:35 tmp
WXR-xr-x 1 root root 4096 Feb 9 2019 usr
WXR-xr-x 1 root root 4096 Jan 30 2019 var
#
```