Lab 2 - Vulnerability Scanning

Secure System Development - Spring 2025

In this lab, you'll:

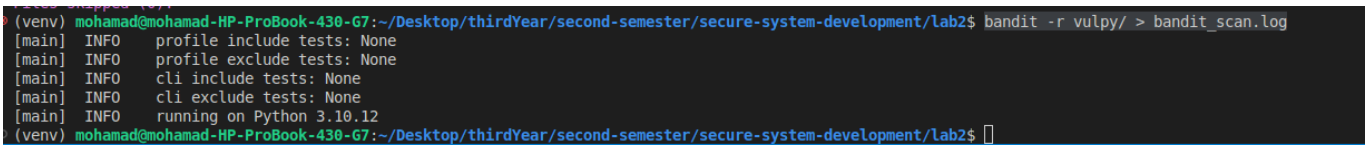
- Test out popular Static Application Security Testing (SAST) tools with different programming languages.
- Learn how to exploit basic web app vulnerabilities.
- Create a report with screenshots and explanations of your findings.

Task 1 - SAST Tools

1.1 Bandit (Python)

Link to Bandit logs of scanning

```
python3 -m venv venv
source venv/bin/activate
pip install bandit
git clone git@github.com:fportantier/vulpy.git
bandit -r vulpy/ > bandit_scan.log
```



Low Severity Issue:

```
--------------------------------------------------
>> Issue: [B404:blacklist] Consider possible security implications
associated with the subprocess module.
   Severity: Low   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info:
https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_imports.html#b4
04-import-subprocess
   Location: vulpy/bad/brute.py:3:0
2
3 import subprocess
4 import sys
```

Explanation:

The `subprocess` module in Python is used to execute system commands. If we don't handle inputs

properly, attackers could inject malicious commands. This can result in a security hole where someone could execute arbitrary commands on the system.
CWE-78 deals with improper handling of input, which can lead to command injection.
Solution: Be careful when using `subprocess`. Always sanitize input properly or use safer alternatives.

Medium Severity Issue:

```
--------------------------------------------------
>> Issue: [B113:request_without_timeout] Call to requests without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info:
https://bandit.readthedocs.io/en/1.8.3/plugins/b113_request_without_timeout
.html
   Location: vulpy/bad/api_post.py:30:8
29     api_key = api_key_file.open().read()
30     r = requests.post('http://127.0.1.1:5000/api/post', json=
{'text':message}, headers={'X-APIKEY': api_key})
31     print(r.text)
```

Explanation:
When you make HTTP requests without setting a timeout, your program might freeze if the server doesn't respond quickly enough. This could lead to performance issues or cause the system to become unresponsive.
CWE-400 covers how missing timeouts in requests can lead to denial of service attacks.
Solution: Always add a timeout when making HTTP requests. This ensures the program doesn't hang indefinitely if something goes wrong with the request.

High Severity Issue:

```
--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with
debug=True, which exposes the Werkzeug debugger and allows the execution of
arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info:
https://bandit.readthedocs.io/en/1.8.3/plugins/b201_flask_debug_true.html
   Location: vulpy/bad/vulpy-ssl.py:29:0
28
29 app.run(debug=True, host='127.0.1.1', ssl_context=('/tmp/acme.cert',
'/tmp/acme.key'))
```

Explanation:
Running a Flask app with `debug=True` exposes detailed error messages and the Werkzeug debugger, which could allow attackers to execute arbitrary code on your system. This is a significant risk in production

environments.
CWE-94 is about code injection, where attackers can execute their code remotely due to poor security configurations.
Solution: Always disable debugging (`debug=False`) in production to avoid exposing sensitive information and prevent remote code execution.

1.2 Flawfinder (C)

Link to Flawfinder logs of scanning

```
pip install flawfinder
git clone git@github.com:hardik05/Damn_Vulnerable_C_Program.git
flawfinder Damn_Vulnerable_C_Program/ > flawfinder_scan.log
```

Level 1 (Low Severity):

```
Damn_Vulnerable_C_Program/linux/imgRead_socket.c:74:  [1] (buffer) read:
  Check buffer boundaries if used in a loop including recursive loops
  (CWE-120, CWE-20).
```

Explanation:
The `read` function is being used to read data into a buffer, but it's not checking if the buffer is large enough. This can cause a **buffer overflow** if more data is read than the buffer can handle, leading to potential security risks.
CWE-120 refers to buffer overflows, and **CWE-20** addresses improper input validation.
Solution: Make sure to check the size of the buffer before using the `read` function. It's a good practice to use safer functions that limit the amount of data being read.

Level 2 (Medium Severity):

```
Damn_Vulnerable_C_Program/dvcp.c:58:  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination (CWE-
120).
  Make sure destination can always hold the source data.
```

Explanation:
The `memcpy` function is used to copy data, but if it's not properly checked, it can overwrite memory areas, causing a **buffer overflow**. This can lead to crashes or, in the worst case, allow attackers to execute malicious code.
CWE-120 again refers to buffer overflows, and **CWE-120** refers to improper input validation.
Solution: Always ensure that both source and destination buffers are large enough before using `memcpy`. Using safer alternatives can also help avoid these issues.

Level 3 (Medium Severity): I didn't find any issue related to this level)

False positive:

```
Damn_Vulnerable_C_Program/dvcp.c:33:  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move things
  around to create a race condition, control its ancestors, or change its
  contents? (CWE-362).
```

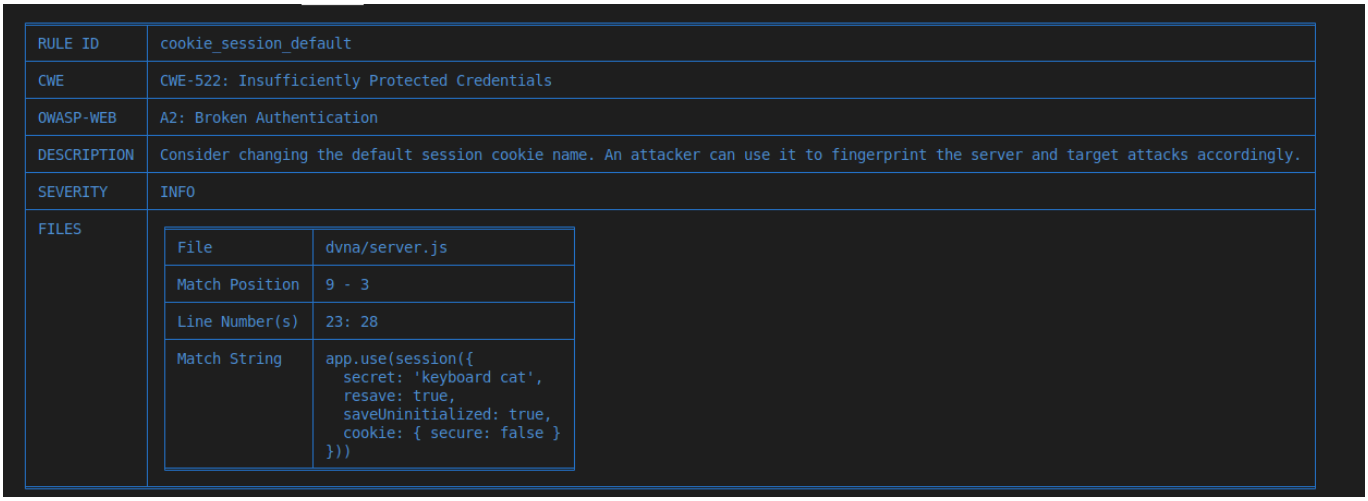
Explanation:
One false-positive finding from the logs could be the warning related to the use of `fopen`. The warning suggests checking for potential security risks, like attackers redirecting files or causing race conditions. However, this might not be an issue if the program carefully controls where files are opened, such as ensuring the file path is always secure and validated. If there's no user input involved in choosing the file, and proper checks are in place, then this warning doesn't apply and can be considered a false positive.

1.3 njsscan (NodeJS)

Link to njsscan logs of scanning

```
pip install njsscan
git clone git@github.com:appsecco/dvna.git
njsscan dvna/
```

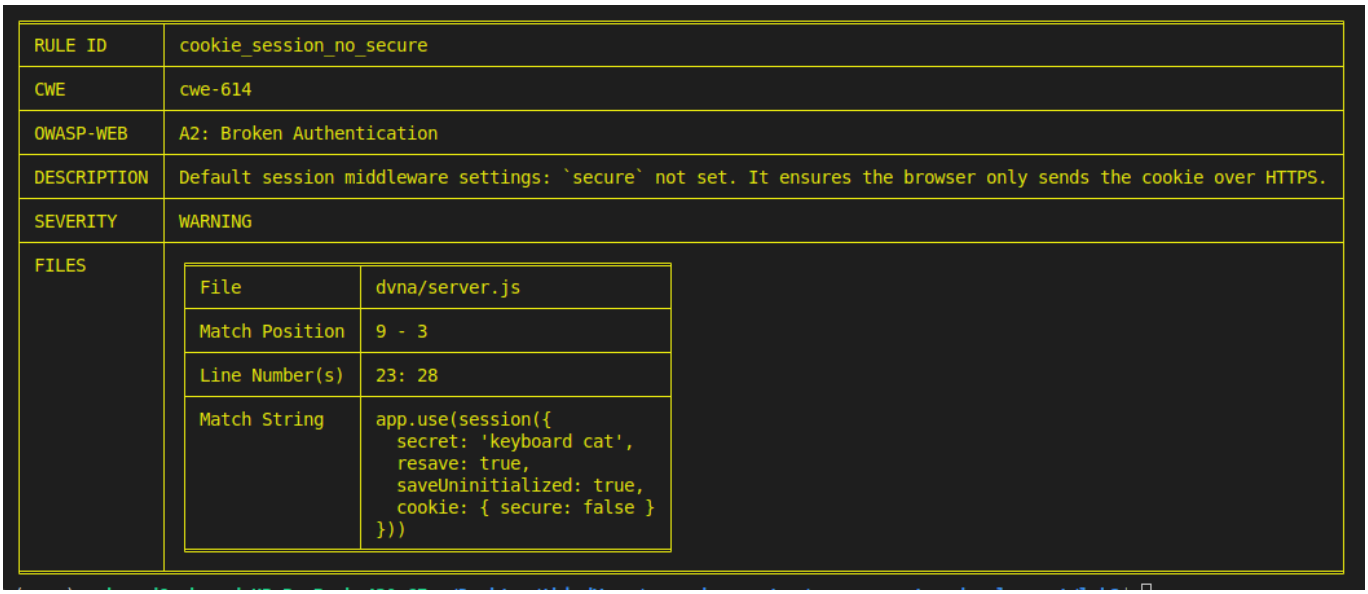
INFO Severity:



RULE ID	cookie_session_default
CWE	CWE-522: Insufficiently Protected Credentials
OWASP-WEB	A2: Broken Authentication
DESCRIPTION	Consider changing the default session cookie name. An attacker can use it to fingerprint the server and target attacks accordingly.
SEVERITY	INFO
FILES	File: dvna/server.js Match Position: 9 - 3 Line Number(s): 23: 28 Match String: app.use(session({ secret: 'keyboard cat', resave: true, saveUninitialized: true, cookie: { secure: false } }))

Explanation:
The default session cookie name could reveal information about the server, making it easier for an attacker to recognize and target specific vulnerabilities.
CWE-522 relates to insufficient protection of credentials, meaning that sensitive information, like session cookies, is not sufficiently protected from attackers.
Solution: Change the default session cookie name to make it harder for attackers to detect and exploit the server based on the cookie name.

WARNING Severity:



RULE ID	cookie_session_no_secure
CWE	cwe-614
OWASP-WEB	A2: Broken Authentication
DESCRIPTION	Default session middleware settings: `secure` not set. It ensures the browser only sends the cookie over HTTPS.
SEVERITY	WARNING
FILES	File: dvna/server.js Match Position: 9 - 3 Line Number(s): 23: 28 Match String: app.use(session({ secret: 'keyboard cat', resave: true, saveUninitialized: true, cookie: { secure: false } }))

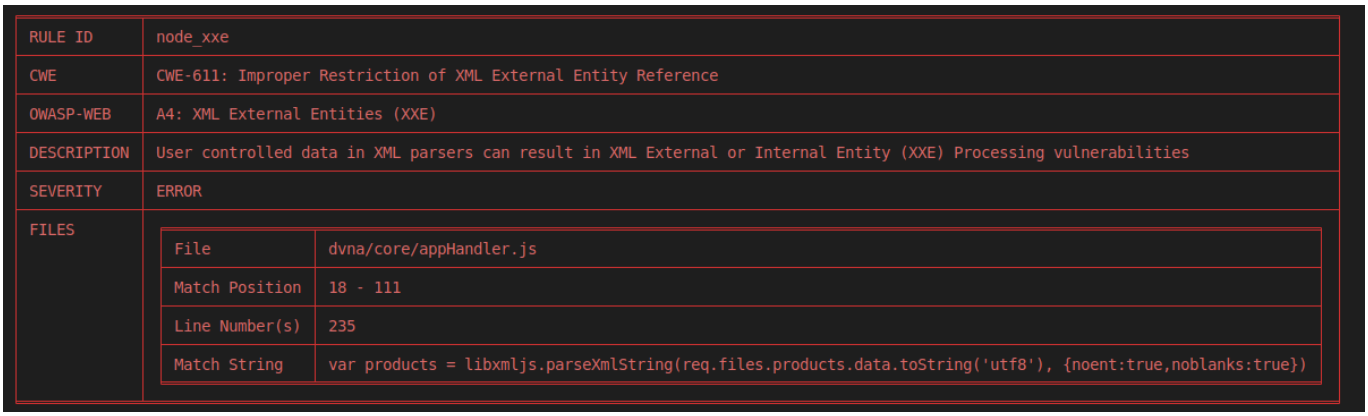
Explanation:

The `secure` flag on cookies ensures they are only sent over secure HTTPS connections. Without this flag, the cookie could be exposed during transmission over an insecure connection, making it vulnerable to interception by attackers.

CWE-614 refers to session fixation, where improper session management can allow attackers to hijack or manipulate sessions.

Solution: Always set the `secure` flag for session cookies so they are transmitted only over HTTPS. This minimizes the risk of data being intercepted.

ERROR Severity:



RULE ID	node_xxe
CWE	CWE-611: Improper Restriction of XML External Entity Reference
OWASP-WEB	A4: XML External Entities (XXE)
DESCRIPTION	User controlled data in XML parsers can result in XML External or Internal Entity (XXE) Processing vulnerabilities
SEVERITY	ERROR
FILES	File: dvna/core/appHandler.js Match Position: 18 - 111 Line Number(s): 235 Match String: var products = libxmljs.parseXmlString(req.files.products.data.toString('utf8'), {noent:true,noblanks:true})

Explanation:

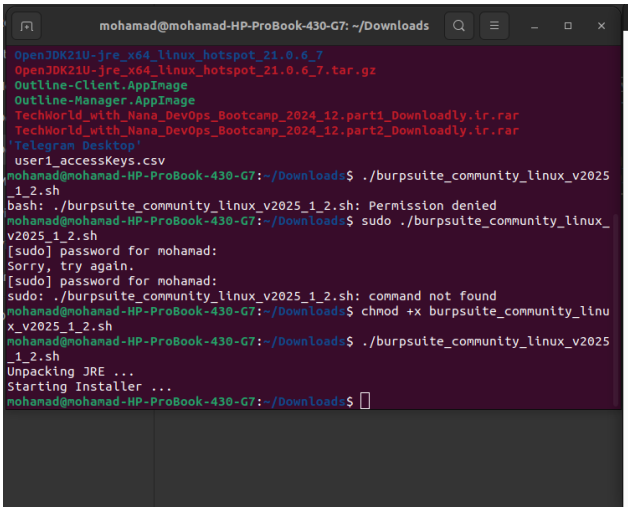
The vulnerability here is due to improper handling of XML data. An attacker can exploit a system by sending specially crafted XML that includes external entity references. These references can cause the system to process unintended data, leading to security issues like data leakage or remote code execution.

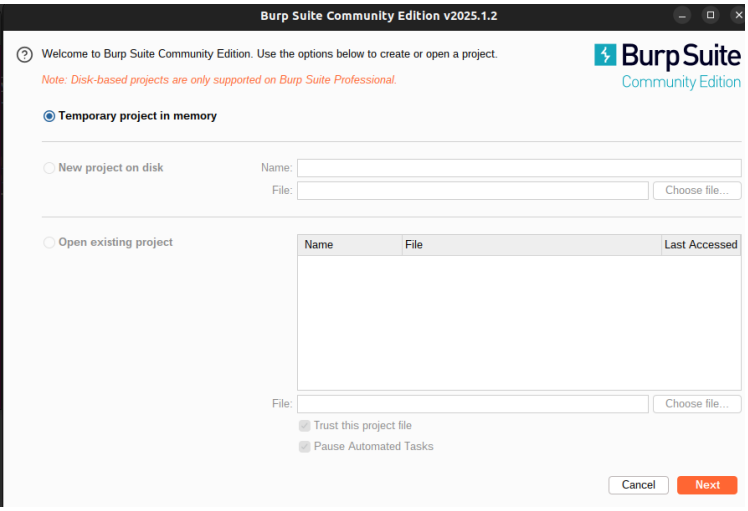
CWE-611 addresses vulnerabilities related to XML External Entity (XXE) attacks, where an attacker manipulates XML parsers to gain access to sensitive information.

Solution: Disable external entity processing in XML parsers to prevent XXE attacks. This helps secure the system from potentially dangerous XML inputs.

Task 2 - Web Security Mini Labs

1. Install BurpSuite (Community Edition)





---

2. Running Vulnerable Applications

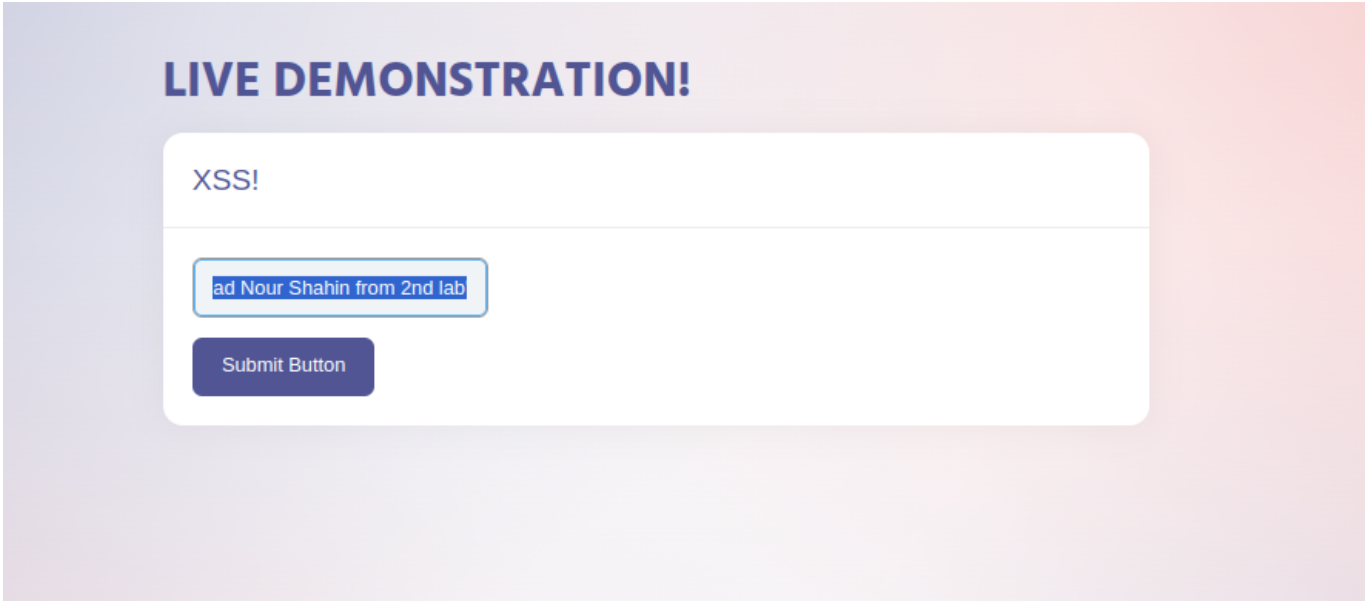
2.1 Cross-Site Scripting (XSS)

Running the XSS image

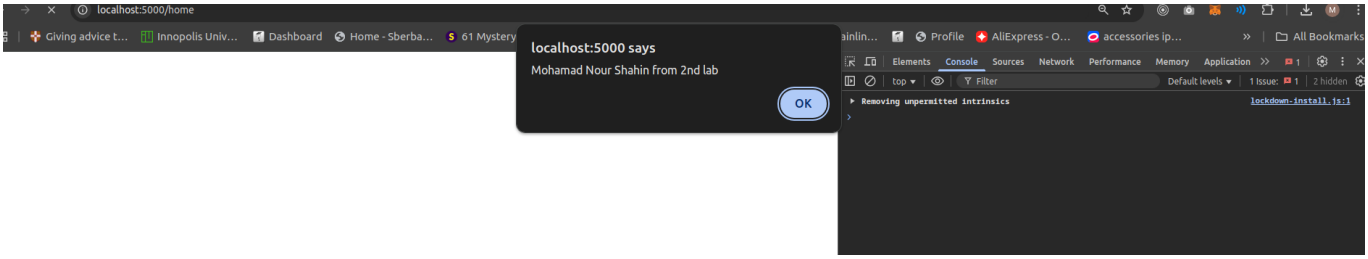
```
docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:xss
```

```
ohamad@mohamad-HP-ProBook-430-G7:~/Downloads$ docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:xss
nable to find image 'sh3b0/vuln:xss' locally
ss: Pulling from sh3b0/vuln
d20c808ce19: Pull complete
3879ca88737: Pull complete
5f42fd8906f: Pull complete
557e2c8196f: Pull complete
217645038d7: Pull complete
38bc1756869: Pull complete
f4fb700ef54: Pull complete
fdce7af2d85: Pull complete
27da1379ac0: Pull complete
d9dce4a769d: Pull complete
5b2c35b71b7: Pull complete
igest: sha256:f0f97aaa3494122a41f2746851f3474c4f6899948e4509212cabf39796530a71
tatus: Downloaded newer image for sh3b0/vuln:xss
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 254-198-468
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET / HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/css/Normalize.css HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/css/datepicker3.css HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/css/styles.css HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/js/lumino.glyphs.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/js/hints.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/img/badge.svg HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:14] "GET /static/js/jquery-3.6.0.min.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:15] "GET /static/js/bootstrap.min.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:16] "GET /static/fonts/fontello/fontello.woff2?51654781 HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:17] "GET /favicon.ico HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:50] "POST /home HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:40:51] "GET /favicon.ico HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:05] "POST /home HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:06] "GET /favicon.ico HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:26] "POST /home HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:27] "GET /favicon.ico HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:34] "GET /home HTTP/1.1" 405 -
72.17.0.1 - - [26/Feb/2025 12:41:37] "GET /favicon.ico HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET / HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/css/Normalize.css HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/css/datepicker3.css HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/css/styles.css HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/js/lumino.glyphs.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/js/hints.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/img/badge.svg HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:42] "GET /static/js/jquery-3.6.0.min.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:43] "GET /static/js/bootstrap.min.js HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:43] "GET /static/fonts/fontello/fontello.woff2?51654781 HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:43] "GET /favicon.ico HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:46] "POST /home HTTP/1.1" 200 -
72.17.0.1 - - [26/Feb/2025 12:41:47] "GET /favicon.ico HTTP/1.1" 200 -
```

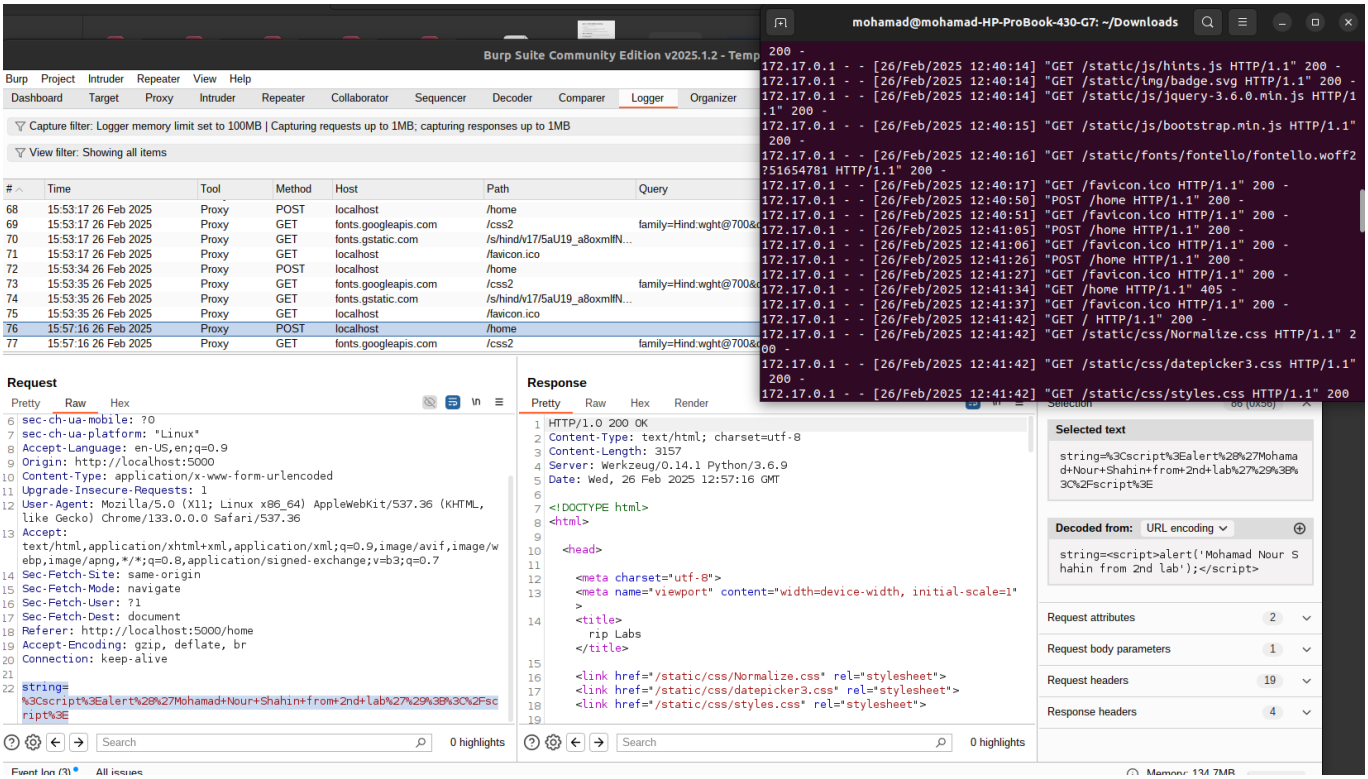
Injecting a script



Results of injection



Captured in Burp Suite



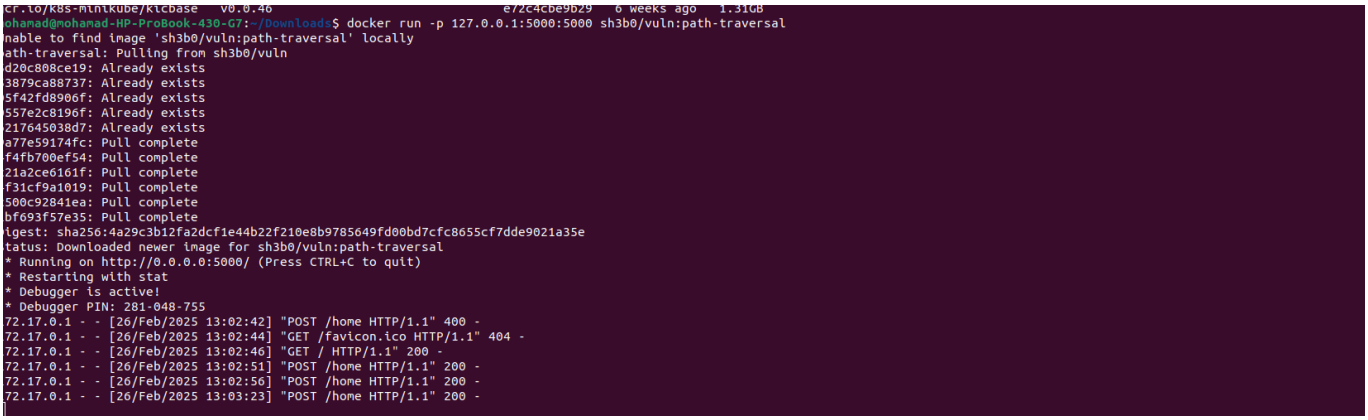
Why XSS is Dangerous: XSS allows attackers to inject malicious scripts into web pages viewed by other users. This can lead to session hijacking, data theft, and even malware distribution. To mitigate this,

websites should sanitize and validate user inputs, implement Content Security Policy (CSP), and use secure frameworks that prevent script injection.

2.2 Path Traversal

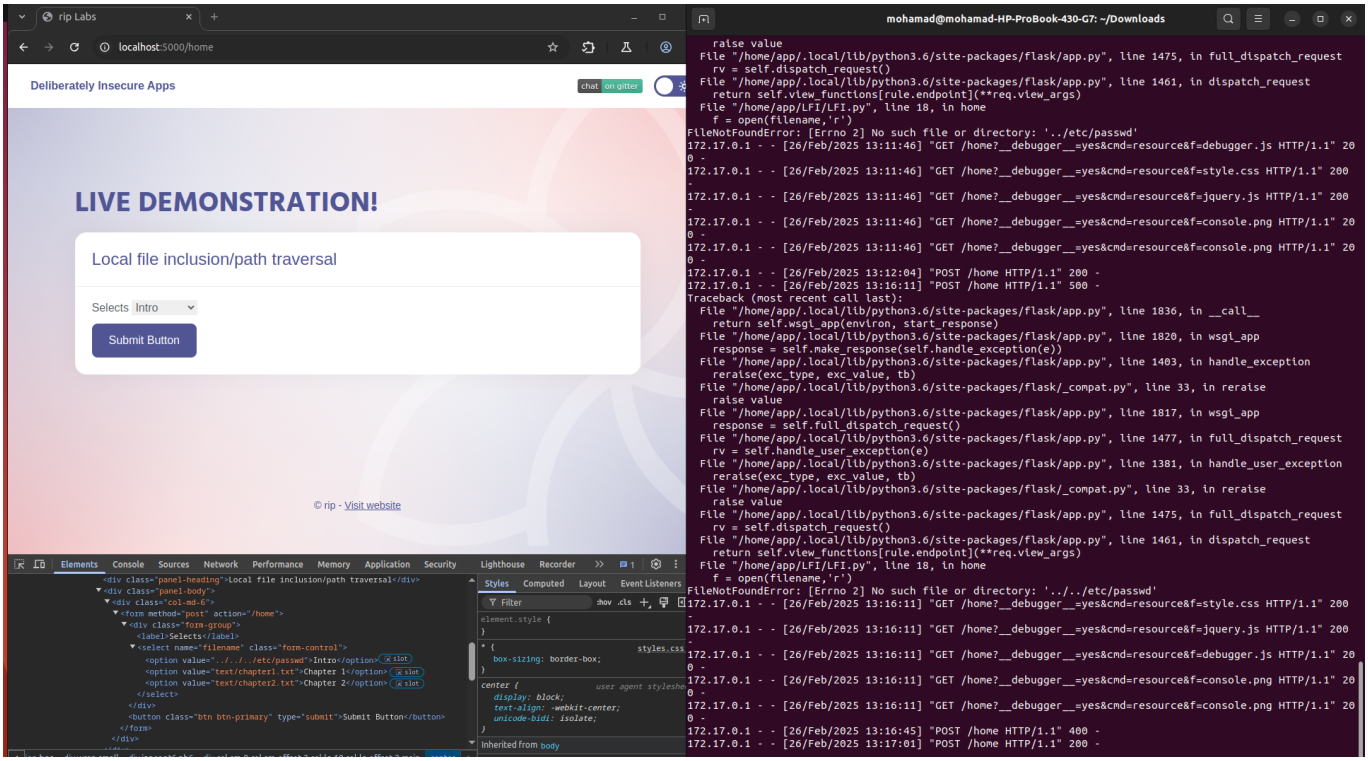
Running the path traversal image

```
docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:path-traversal
```

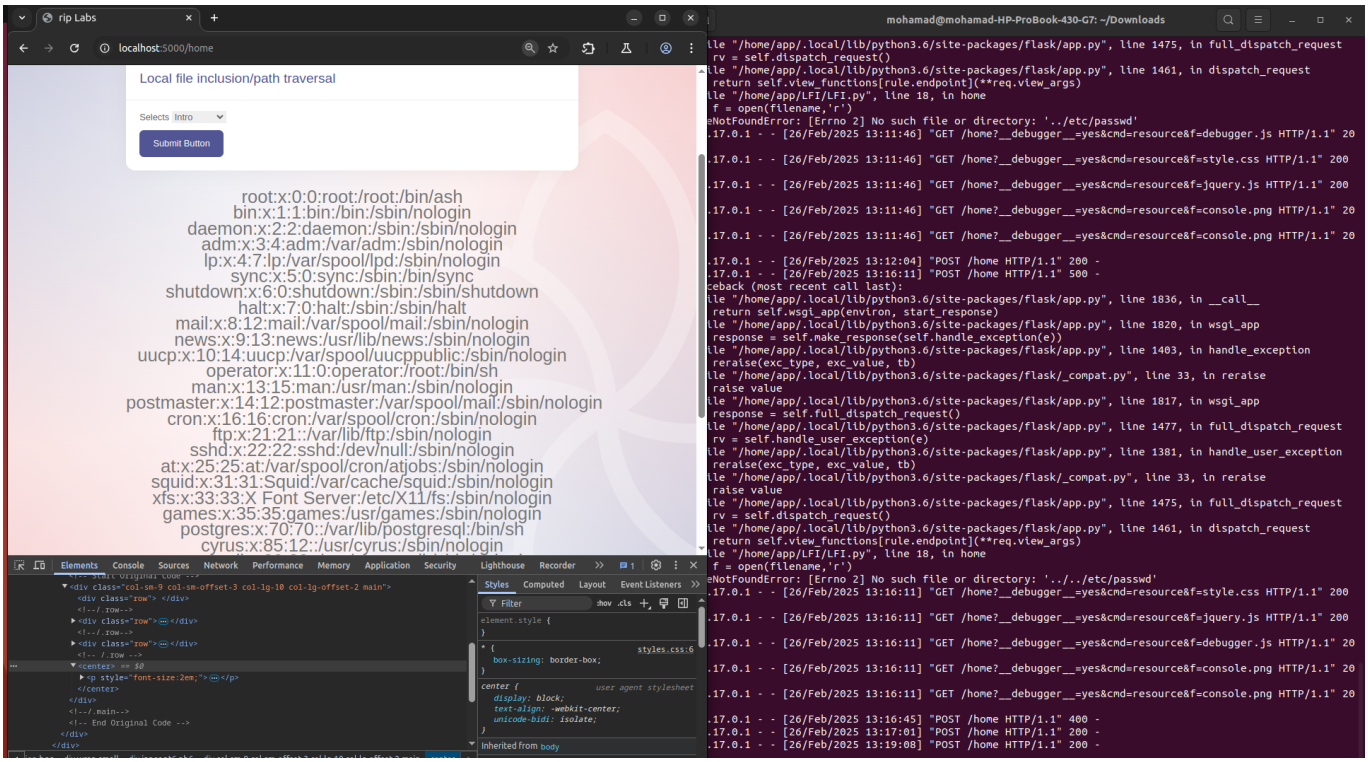


Exploiting path traversal by modifying a request

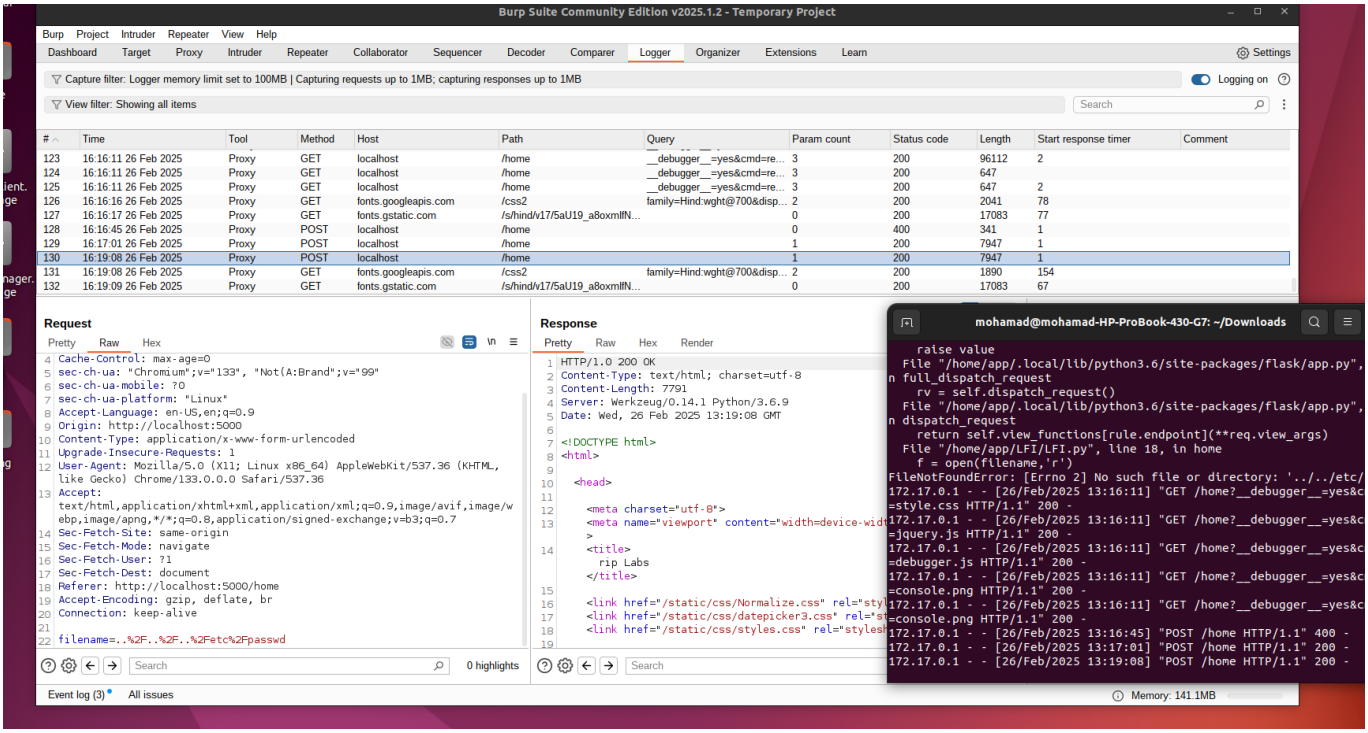
Changing the value to `../../../etc/passwd`



After submitting



Captured in Burp Suite-1

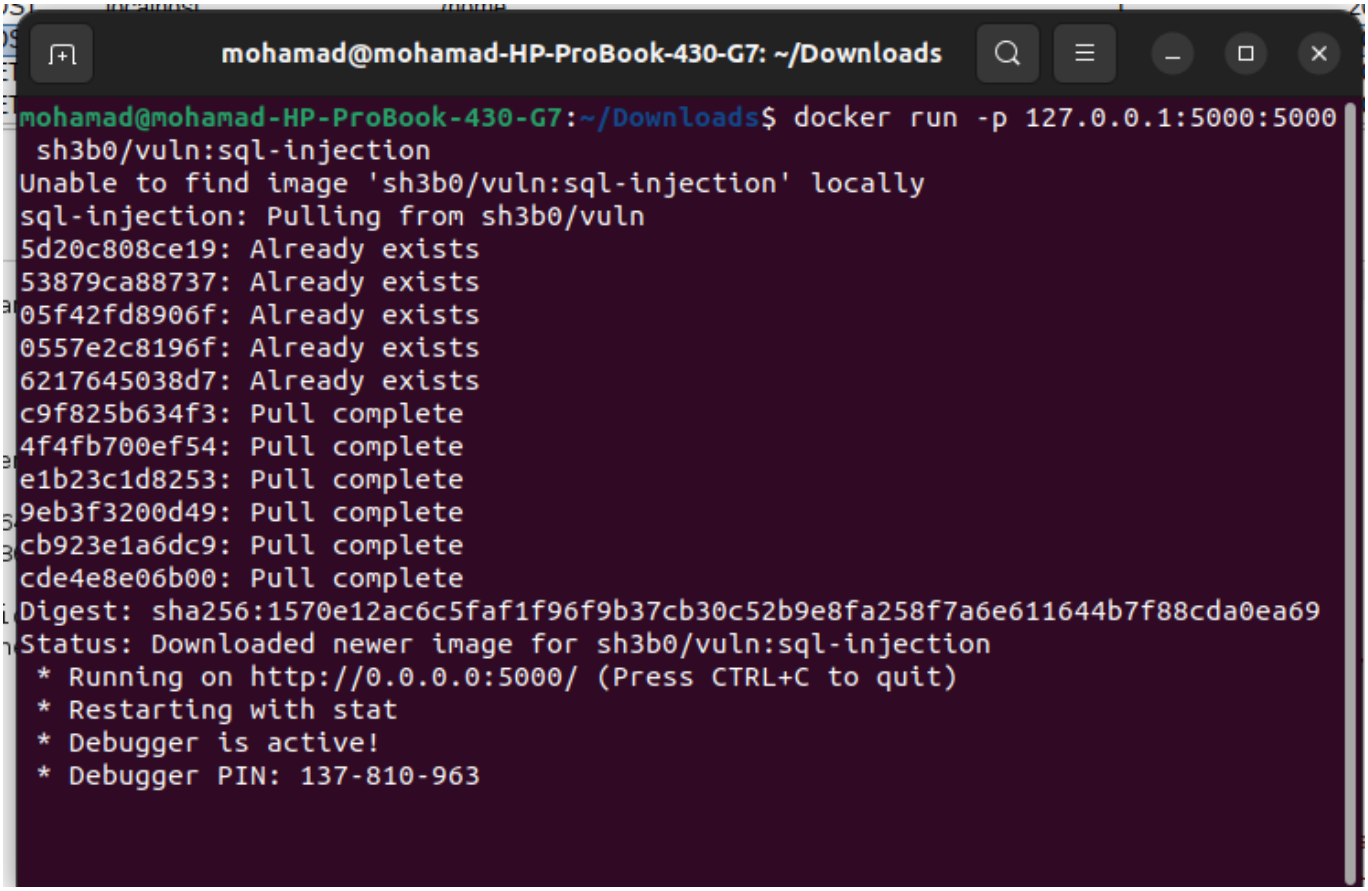


Why Path Traversal is Dangerous: Path traversal attacks allow attackers to access restricted directories and files, potentially exposing sensitive data such as configuration files or credentials. To prevent this, developers should normalize input paths, use allowlists for file access, and restrict user input.

2.3 SQL Injection

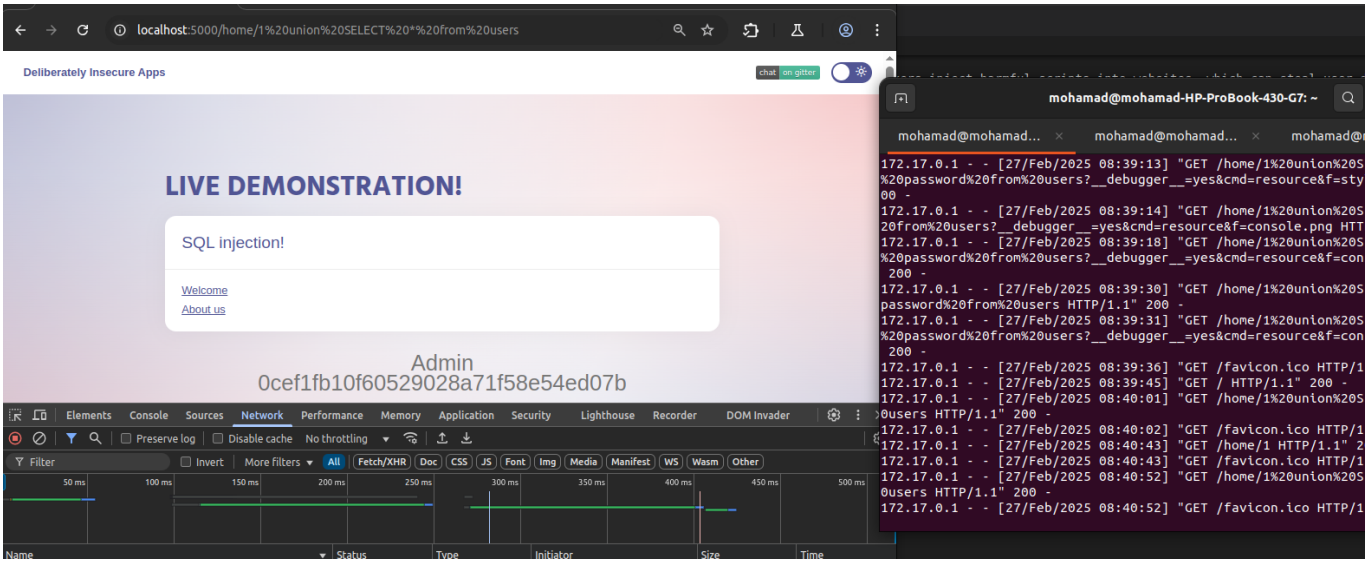
Running the SQL Injection image

```
docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:sql-injection
```



Attempting SQL Injection

Injecting `1 UNION SELECT * FROM users` into an input field.

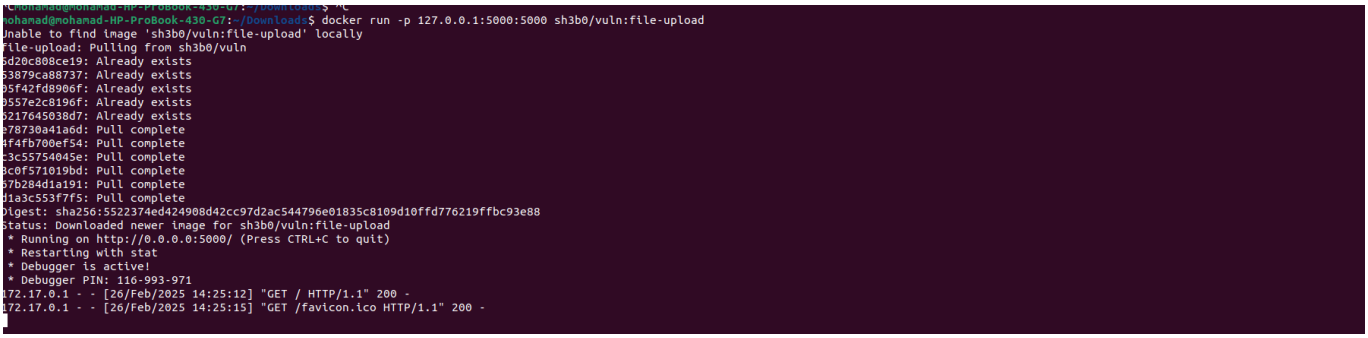


Why SQL Injection is Dangerous: SQL Injection can allow attackers to manipulate database queries, potentially exposing or modifying sensitive data. To prevent this, use prepared statements, ORM frameworks, and enforce strict input validation.

2.4 File Upload Exploit

Running the file upload image

```
docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:file-upload
```

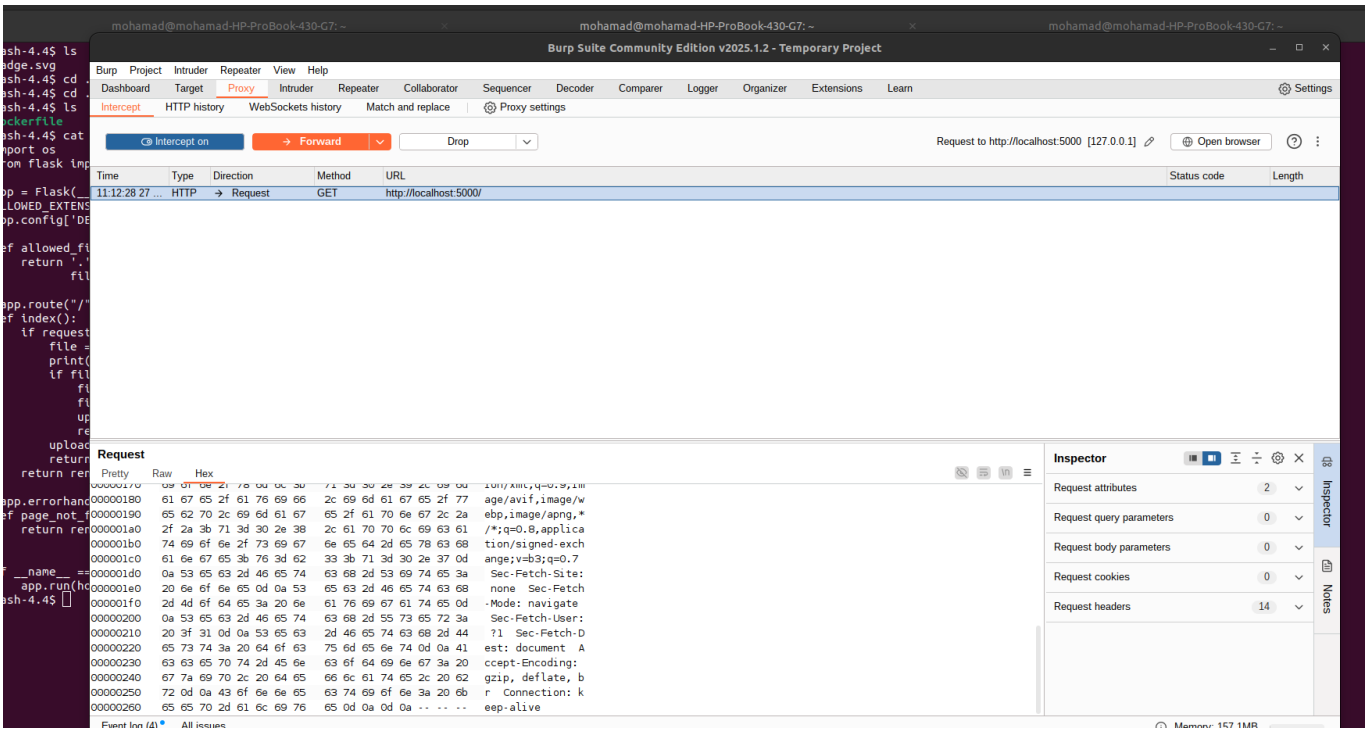


Bypassing file upload restrictions

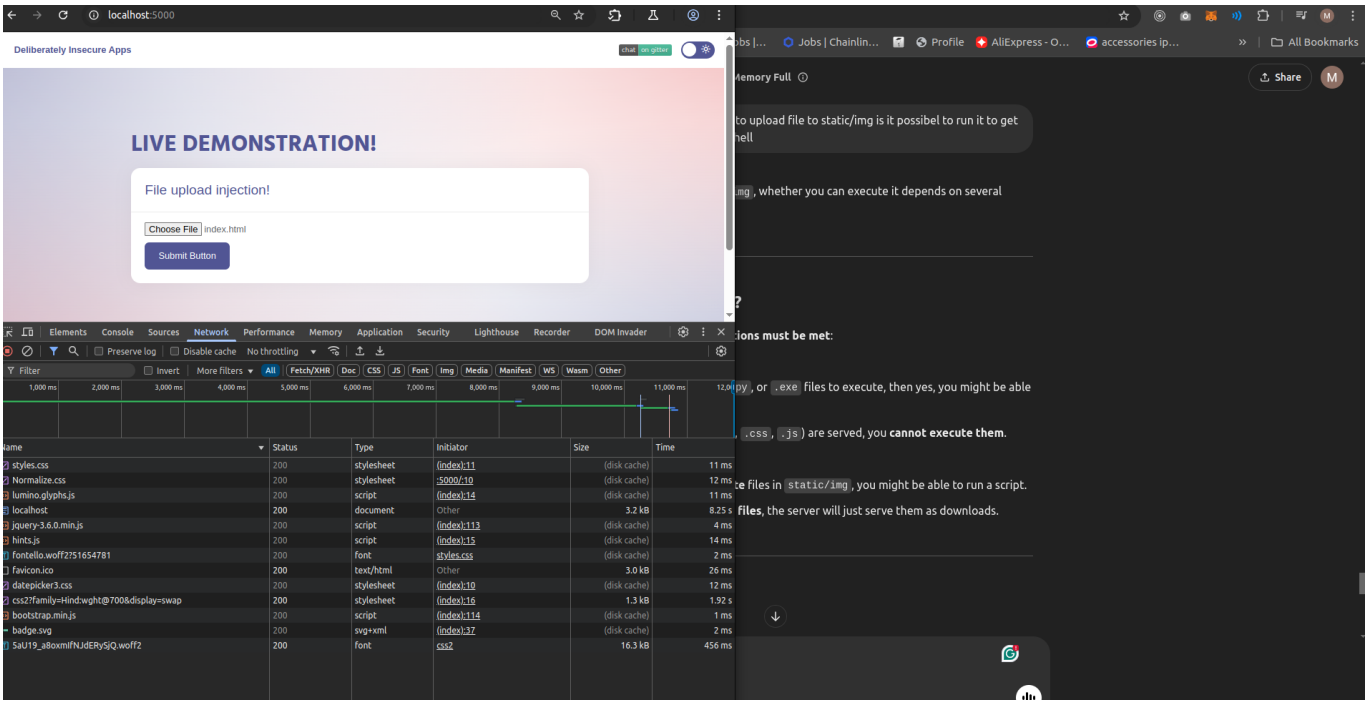
Uploading an HTML file and intercepting the request in Burp Suite to modify the file path.

```
ash-4.4$ cat File-upload.py
mport os
rom flask import Flask, request, render_template

pp = Flask(__name__, static_url_path='/static', static_folder='static')
LLOWED_EXTENSIONS = app.config['ALLOWED_EXTENSIONS'] = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'html'])
pp.config['DEBUG'] = True

ef allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS

app.route("/", methods=['GET', 'POST'])
ef index():
    if request.method == 'POST':
        file = request.files['file']
        print(file)
        if file and allowed_file(file.filename):
            filename = file.filename
            file.save(os.path.join('uploads/', filename))
            uploaded = "File was uploaded"
            return render_template("index.html",uploaded = uploaded)
        uploaded = "something went wrong!"
        return render_template("index.html",uploaded = uploaded)
    return render_template("index.html")

app.errorhandler(404)
ef page_not_found(e):
    return render_template("404.html")


f __name__ == "__main__":
    app.run(host='0.0.0.0')
```

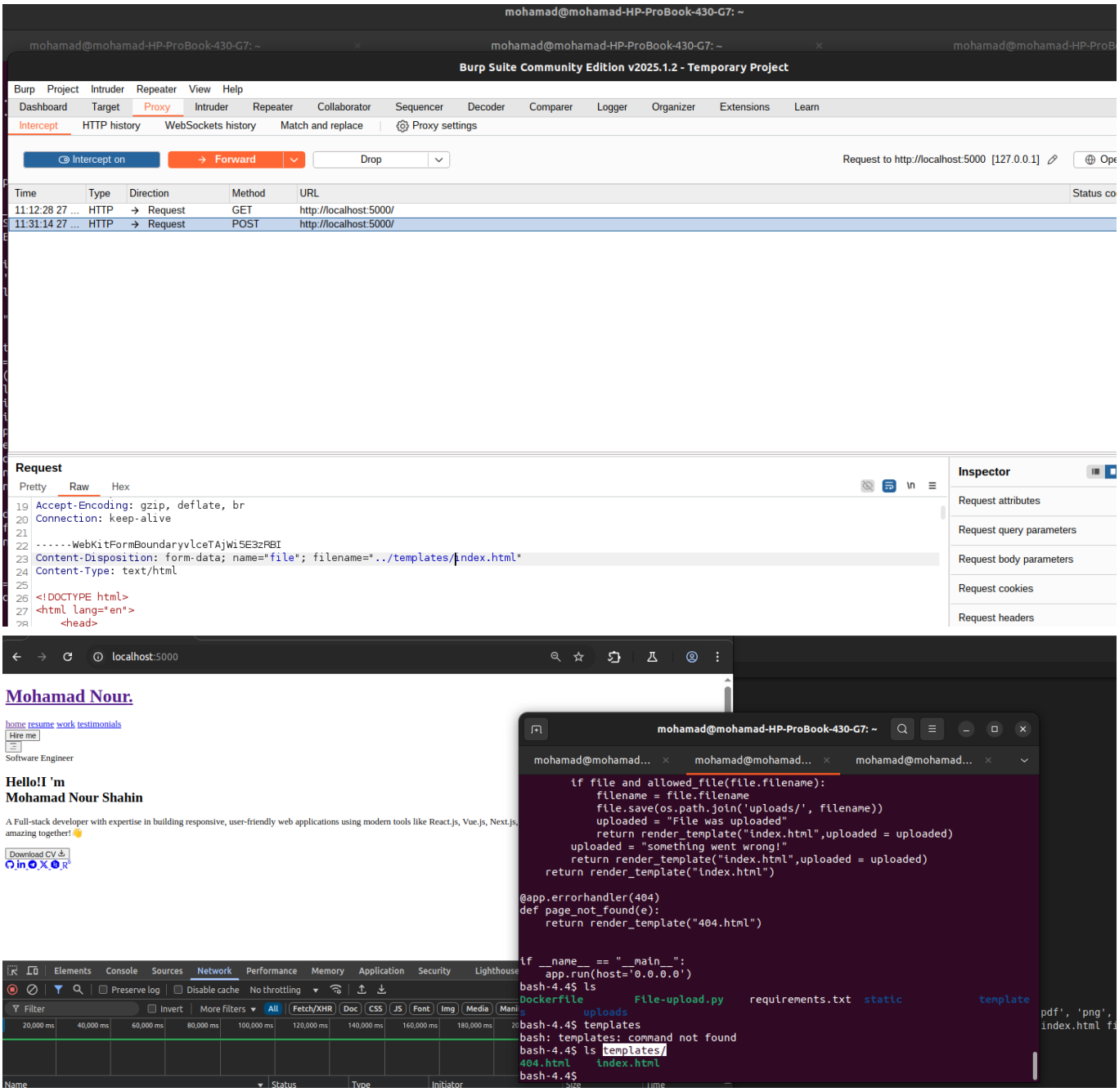
Turning on intercept in Burp Suite



Uploading and submitting the HTML file



Successful exploitation



Why Unrestricted File Upload is Dangerous: Attackers can upload malicious files to deface a website, spread malware, or gain unauthorized access. This can lead to data breaches, phishing attacks, or even full system compromise. To prevent this, always validate file types, enforce server-side checks, limit upload permissions, and store files outside the web root with randomized names.

2.5 Command Injection

Running the command injection image

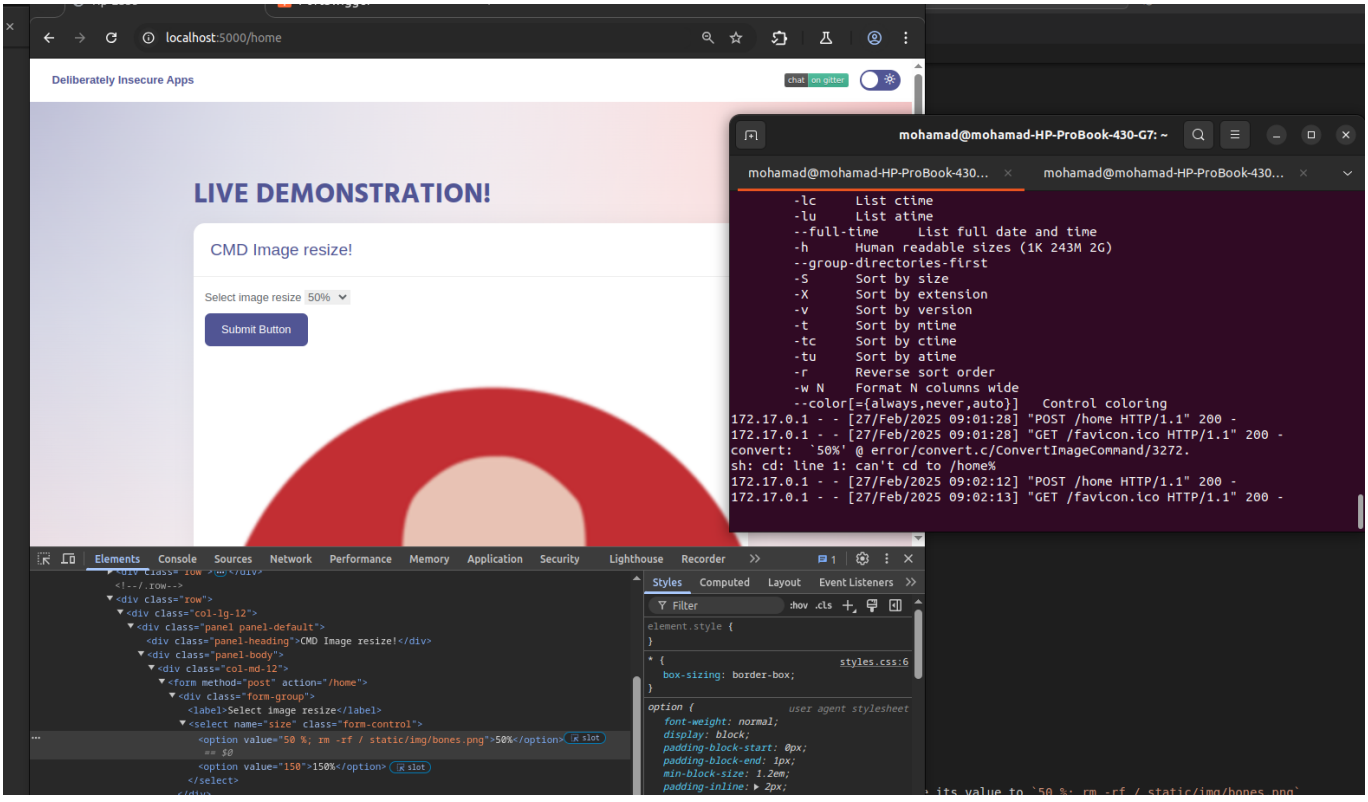
```
docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:file-upload
```

```
mohamad@mohamad-HP-ProBook-430-G7:~$ docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:command-injection
Unable to find image 'sh3b0/vuln:command-injection' locally
command-injection: Pulling from sh3b0/vuln
5d20c808ce19: Already exists
16a1e357d681: Pull complete
d3f9eb6ecca0: Pull complete
9dc528f883ed: Pull complete
77cc860d68fd: Pull complete
8496d86fc735: Pull complete
4f4fb700ef54: Pull complete
f61c0fb18c6a: Pull complete
5f2812bb7c7f: Pull complete
9f3af57112c1: Pull complete
d74022104470: Pull complete
Digest: sha256:9dac3a070309af595902684f2a6f56bbd96a0ff01b8f014b899ac87d1927a659
Status: Downloaded newer image for sh3b0/vuln:command-injection
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 159-780-626
```

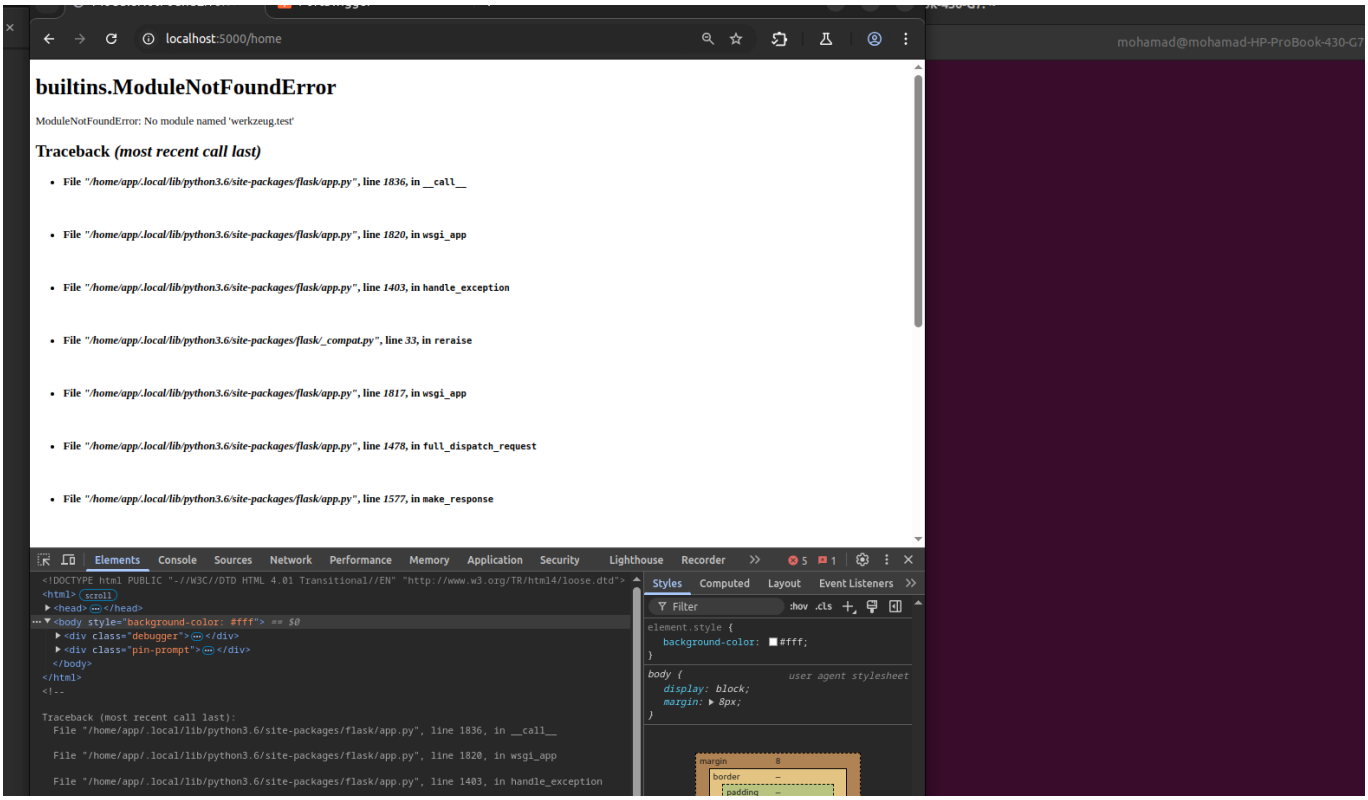
Injecting a command into an HTML element

Modifying a field to include:

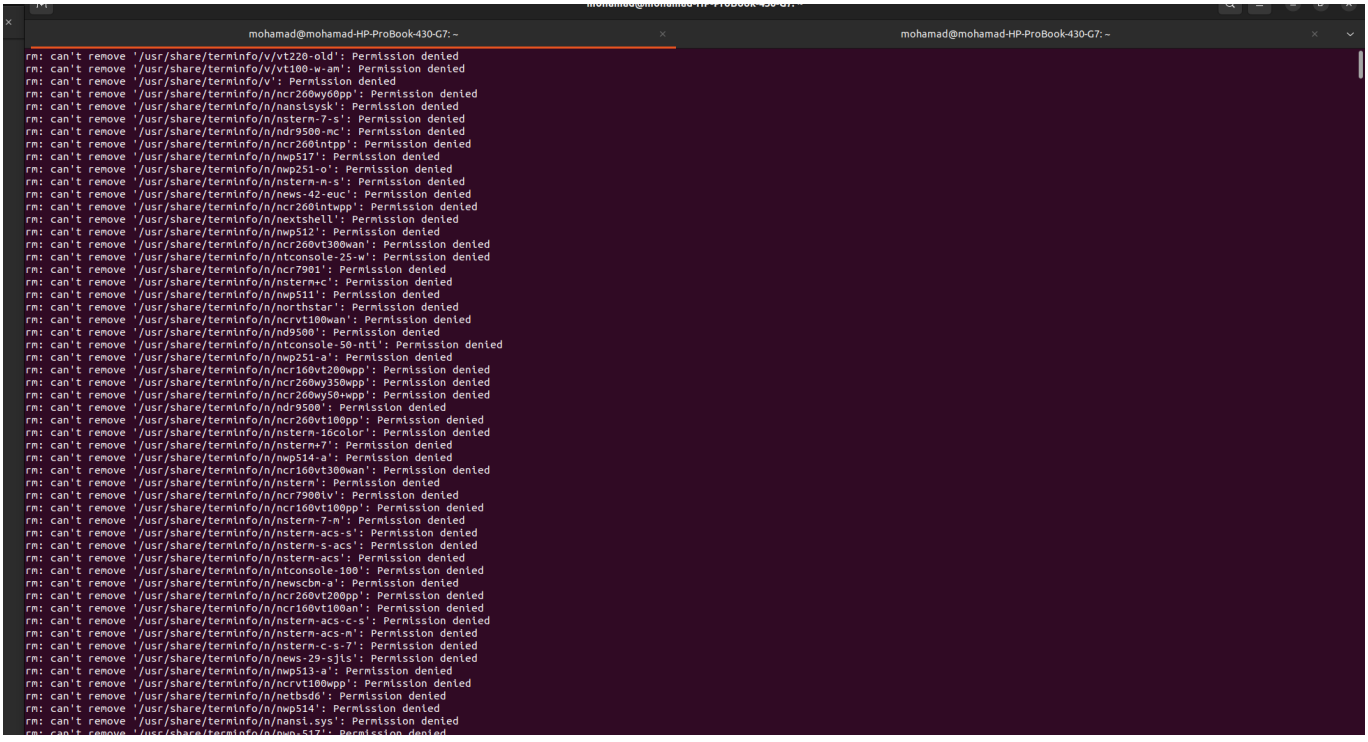
```
50%; rm -rf /static/img/bones.png
```



After injection



Captured output in terminal



Why Command Injection is Dangerous: Command injection allows attackers to execute arbitrary commands on the server, potentially leading to data theft, system compromise, or destruction. To prevent this, avoid executing system commands with user input, use parameterized APIs, and apply strict input validation.