

Lab 2 - Vulnerability Scanning

Ali Hamdan

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)

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

```
[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
(sec_env) alihamd@Alis-MacBook-Air-2 lab2 % git clone https://github.com/fportantier/vulpy.git
Cloning into 'vulpy'...
remote: Enumerating objects: 437, done.
remote: Total 437 (delta 0), reused 0 (delta 0), pack-reused 437 (from 1)
Receiving objects: 100% (437/437), 2.90 MiB | 6.29 MiB/s, done.
Resolving deltas: 100% (223/223), done.
(sec_env) alihamd@Alis-MacBook-Air-2 lab2 % bandit -r vulpy/ > bandit_scan.log
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.13.2
(sec_env) alihamd@Alis-MacBook-Air-2 lab2 %
```

Low Severity Issue:

```
-----
>> Issue: [B110:try_except_pass] Try, Except, Pass detected.
Severity: Low Confidence: High
CWE: CWE-703 (https://cwe.mitre.org/data/definitions/703.html)
More Info:
https://bandit.readthedocs.io/en/1.8.3/plugins/b110_try_except_pass.html
Location: vulpy/bad/libsession.py:21:4
20         session = json.loads(base64.b64decode(cookie))
21     except Exception:
```

```
22         pass
23
```

Explanation:

The code is catching an exception but does nothing (pass). This suppresses errors, making debugging harder and potentially allowing silent failures in security-related functions.

CWE-703 Improper Handling of Exceptional Conditions

Solution: Instead of pass, log the error and handle it properly:

```
import logging
try:
    session = json.loads(base64.b64decode(cookie))
except Exception as e:
    logging.error(f"Error decoding session: {e}")
    raise # Reraise exception for debugging
```

Medium Severity Issue:

```
-----
>> Issue: [B108:hardcoded_tmp_directory] Probable insecure usage of temp
file/directory.
Severity: Medium   Confidence: Medium
CWE: CWE-377 (https://cwe.mitre.org/data/definitions/377.html)
More Info:
https://bandit.readthedocs.io/en/1.8.3/plugins/b108\_hardcoded\_tmp\_directory.html
Location: vulpy/bad/api_post.py:6:20
5
6  api_key_file = Path('/tmp/supersecret.txt')
7
```

Explanation:

Storing sensitive files in /tmp/ is insecure because it is a world-writable directory, allowing unauthorized users to read, modify, or delete the file. Attackers can exploit this to access sensitive data or perform symlink attacks.

CWE-377 covers how improper handling of temporary files can lead to security risks.

Solution: Use Python's tempfile module to create secure temporary files or store sensitive data in a restricted directory with proper permissions.

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/good/vulpy.py:53:0
52
53 app.run(debug=True, host='127.0.1.1', port=5001,
extra_files='csp.txt')
54
```

Explanation:

Running a Flask app with `debug=True` enables the Werkzeug debugger, which allows arbitrary code execution if an attacker gains access. This can lead to remote code execution and compromise the entire system.

CWE-94 covers how improper code execution can result in severe security risks.

Solution: Don't use `debug=True` in production. Instead, set `debug=False` and use proper logging mechanisms for debugging.

1.2 Flawfinder (C)

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

Level 1 (Low Severity):

```
Damn_Vulnerable_C_Program/libAFL/damn_vulnerable_c_program_shmem/imgRead.c
:35: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
```

Explanation:

The `strlen` function does not check if a string is **null-terminated**. If it encounters a non-null-terminated string, it may over-read memory, potentially causing crashes or exposing sensitive data.

CWE-126 refers to buffer over-read vulnerabilities, where a function reads beyond the allocated memory.

Solution: Always ensure strings are properly null-terminated before using `strlen`, and use safer functions like `strnlen` to limit the read length.

Level 2 (Medium Severity):

```
Damn_Vulnerable_C_Program/dvcp.c:16: [2] (buffer) char:
Statically-sized arrays can be improperly restricted, leading to
potential
```

```
overflows or other issues (CWE-119!/CWE-120). Perform bounds checking,
use
functions that limit length, or ensure that the size is larger than the
maximum possible length.
```

Explanation:

Statically-sized character arrays can lead to buffer overflows if input data exceeds the allocated size. Without proper bounds checking, this can cause memory corruption, crashes, or even arbitrary code execution.

CWE-119 covers improper memory restrictions, and **CWE-120** specifically addresses buffer overflows.

Solution: Always validate input length before storing it in a fixed-size buffer, use `strncpy` or `snprintf` instead of unsafe functions like `strcpy`, and consider dynamically allocating memory when needed.

Level 3 (Medium Severity): No issue related to level [3]

False positive:

```
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 warning for the `read` function in `Damn_Vulnerable_C_Program/linux/imgRead_socket.c:74` may be a false positive if the code already ensures that the buffer boundaries are properly checked before each read operation. The tool flags it for potential buffer overflow (**CWE-120**) and improper input validation (**CWE-20**).

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_no_maxage										
CWE	CWE-613: Insufficient Session Expiration										
OWASP-WEB	A2: Broken Authentication										
DESCRIPTION	Session middleware settings: `maxAge` not set. Use it to set expiration date for cookies.										
SEVERITY	INFO										
FILES	<table><tr><td>File</td><td>dvna/server.js</td></tr><tr><td>Match Position</td><td>9 - 3</td></tr><tr><td>Line Number(s)</td><td>23: 28</td></tr><tr><td>Match String</td><td><pre>app.use(session({ secret: 'keyboard cat', resave: true, saveUninitialized: true, cookie: { secure: false } }))</pre></td></tr></table>			File	dvna/server.js	Match Position	9 - 3	Line Number(s)	23: 28	Match String	<pre>app.use(session({ secret: 'keyboard cat', resave: true, saveUninitialized: true, cookie: { secure: false } }))</pre>
File	dvna/server.js										
Match Position	9 - 3										
Line Number(s)	23: 28										
Match String	<pre>app.use(session({ secret: 'keyboard cat', resave: true, saveUninitialized: true, cookie: { secure: false } }))</pre>										

Explanation:

Not setting a maxAge for session cookies means sessions may persist indefinitely, allowing attackers more time to hijack active sessions.

CWE-613 relates to insufficient session expiration, where sessions do not terminate as expected, increasing the risk of unauthorized access.

Solution: Configure the session cookie with an appropriate maxAge to ensure sessions expire after a defined period, thereby limiting the window for potential attacks.

WARNING Severity:

RULE ID	express_lfr_warning	
CWE	CWE-23: Relative Path Traversal	
OWASP-WEB	A5: Broken Access Control	
DESCRIPTION	Untrusted user input in express render() function can result in arbitrary file read if hbs templating is used.	
SEVERITY	WARNING	
FILES	File	dvna/routes/app.js
	Match Position	9 - 65
	Line Number(s)	23
	Match String	res.render('app/bulkproducts',{legacy:req.query.legacy})
	File	dvna/routes/app.js
	Match Position	9 - 11
	Line Number(s)	37: 39
	Match String	res.render('app/admin', { admin: (req.user.role == 'admin') }))

Explanation:

Untrusted user input passed into the Express res.render() function may allow an attacker to manipulate file paths, potentially leading to arbitrary file reads via relative path traversal. This can expose sensitive data or configuration files.

CWE-23 addresses relative path traversal vulnerabilities, which occur when user-supplied input is used to

construct file paths without proper validation.

Solution: Sanitize and validate all user inputs used in file path constructions, and implement strict whitelisting for acceptable file paths to prevent traversal attacks.

ERROR Severity:

RULE ID	node_deserialize		
CWE	CWE-502: Deserialization of Untrusted Data		
OWASP-WEB	A8: Insecure Deserialization		
DESCRIPTION	User controlled data in 'unserialize()' or 'deserialize()' function can result in Object Injection or Remote Code Injection.		
SEVERITY	ERROR		
FILES			
	File	dvna/core/appHandler.js	
	Match Position	18 - 81	
	Line Number(s)	218	
	Match String	var products = serialize.unserialize(req.files.products.data.toString('utf8'))	

Explanation:

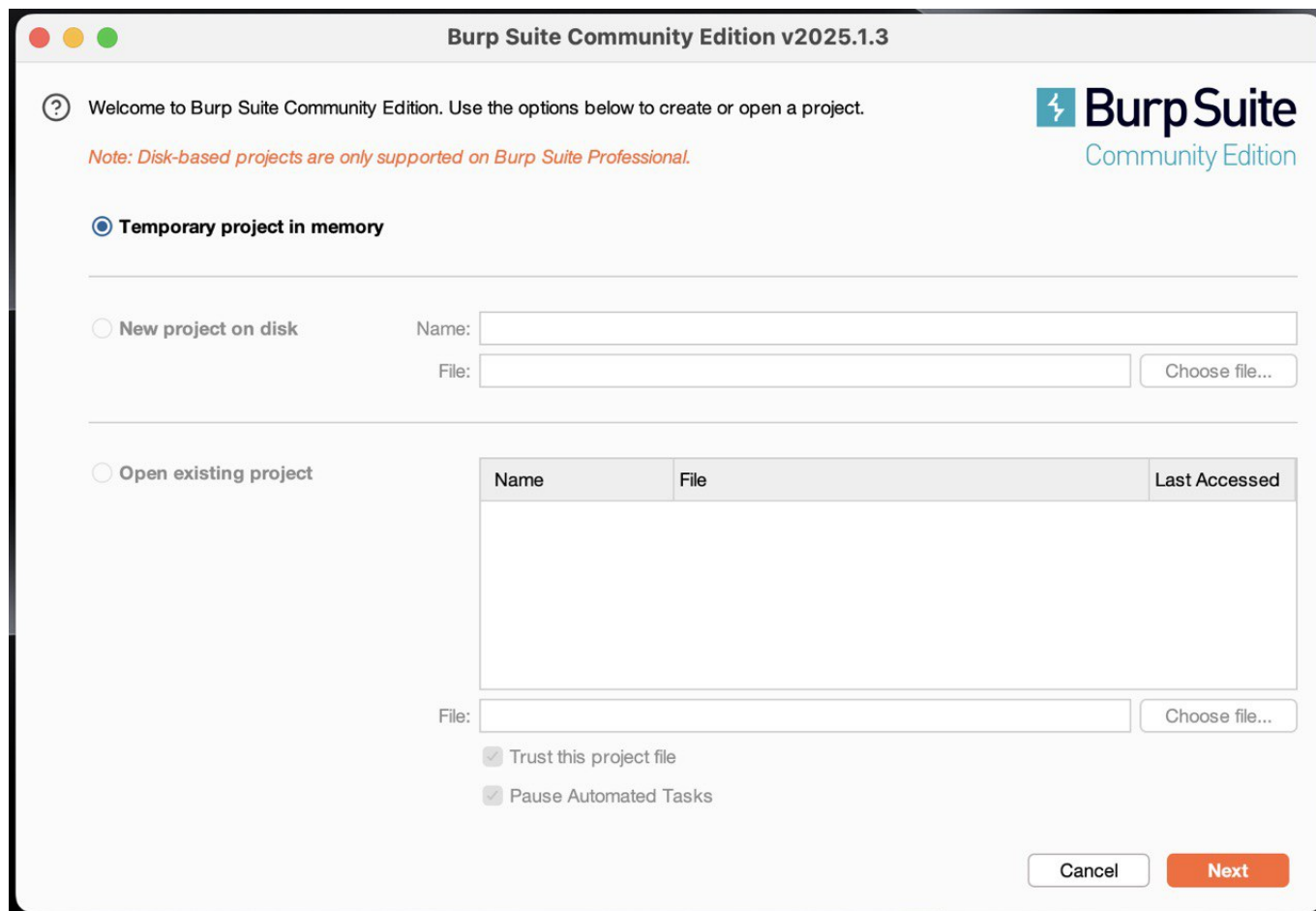
The code unserializes user-supplied data from req.files.products.data.toString('utf8') without proper validation, which can allow an attacker to inject malicious objects and potentially execute arbitrary code.

CWE-502 pertains to the deserialization of untrusted data, which can lead to object injection or remote code execution if the data is not properly validated and sanitized.

Solution: Validate and sanitize the input data before deserialization or use secure deserialization methods that enforce strict type checks to prevent injection of malicious objects.

Task 2 - Web Security Mini Labs

- 1. Install BurpSuite (Community Edition)



2. Running Vulnerable Applications

2.1 Cross-Site Scripting (XSS)

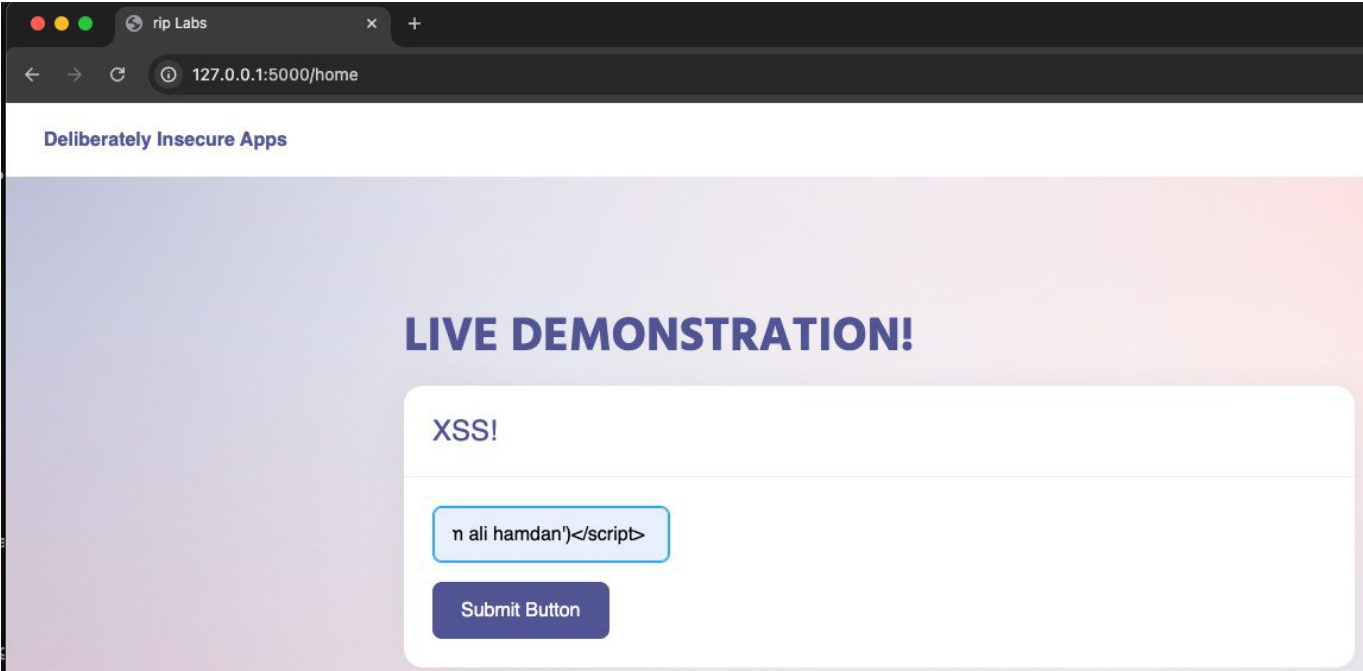
Running the XSS image

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

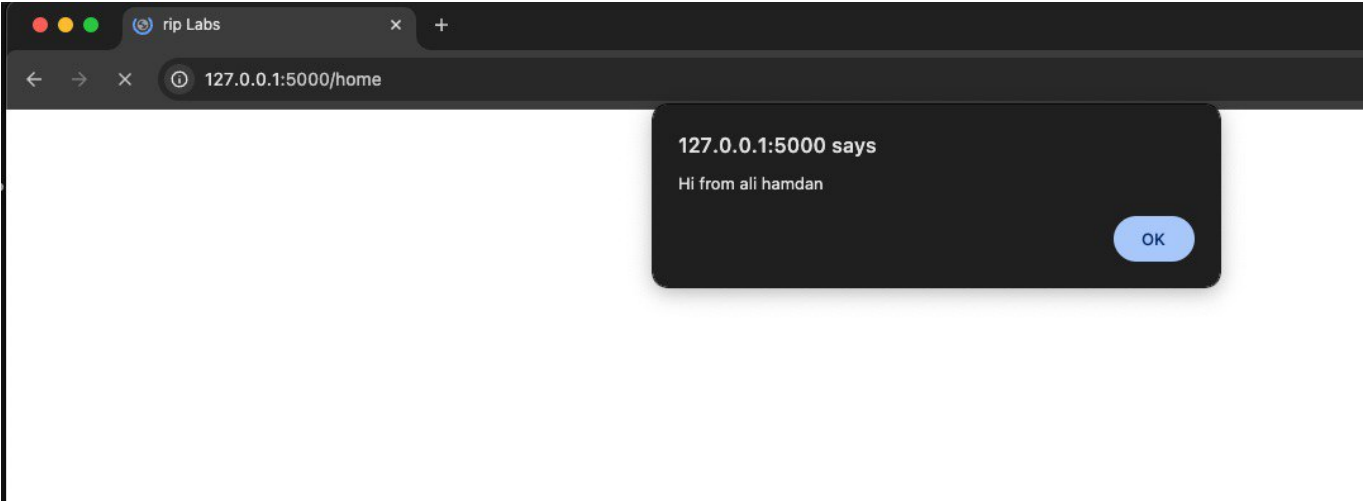
```
Air-2 ~ %
alihamdan@Alis-MacBook-Air-2 ~ % docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:xss

WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 199-818-631
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/css/Normalize.css HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/css/datepicker3.css HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/css/styles.css HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/js/lumino.glyphs.js HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/js/jquery-3.6.0.min.js HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/js/hints.js HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/js/bootstrap.min.js HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/img/badge.svg HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:29] "GET /static/fonts/fontello/fontello.woff2?51654781 HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:30] "GET /apple-touch-icon-precomposed.png HTTP/1.1" 200 -
172.17.0.1 - - [04/Mar/2025 18:06:30] "GET /apple-touch-icon.png HTTP/1.1" 200 -
```

Injecting a script



Results of injection



Captured in Burp Suite

The screenshot shows a web browser interface with a list of requests on the left and a detailed view of a specific request and response on the right.

Request List:

#	Time	Method	URL	Status	Size	Start	End
72	21:30:47 4 Mar 2025	GET	/static/img/badge.svg HTTP/1.1	200	8141	10	
73	21:30:47 4 Mar 2025	GET	/static/js/jquery-3.6.0.min.js HTTP/1.1	200	65823	18	
74	21:30:47 4 Mar 2025	GET	/static/js/bootstrap.min.js HTTP/1.1	200	34117	21	
75	21:30:47 4 Mar 2025	GET	/static/fonts/fontawesome.woff2?51654781 HTTP/1.1	200	1065	23	
76	21:30:47 4 Mar 2025	GET	/static/fonts/fontawesome.woff2?51654781 HTTP/1.1	200	89845	4	
77	21:30:47 4 Mar 2025	GET	/favicon.ico HTTP/1.1	200	32163	2	
78	21:30:47 4 Mar 2025	POST	/home HTTP/1.1	200	1907	95	
79	21:30:47 4 Mar 2025	GET	/favicon.ico HTTP/1.1	200	9182	90	
80	21:30:47 4 Mar 2025	GET	/favicon.ico HTTP/1.1	200	3157	4	
81	21:30:47 4 Mar 2025	Proxy	127.0.0.1	200	2997	7	
82	21:30:58 4 Mar 2025	Proxy	127.0.0.1	200	3299	8	
83	21:31:02 4 Mar 2025	Proxy	127.0.0.1	200	2997	3	

Request Details:

```

1 POST /home HTTP/1.1
2 Host: 127.0.0.1:5000
3 Content-Length: 70
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "macOS"
8 Accept-Language: en-US,en;q=0.9
9 Origin: http://127.0.0.1:5000
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
13 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
14 Accept:
15 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
16 image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
17 Sec-Fetch-Site: same-origin
18 Sec-Fetch-Mode: navigate
19 Sec-Fetch-User: ?1
20 Sec-Fetch-Dest: document
21 Referer: http://127.0.0.1:5000/
22 Accept-Encoding: gzip, deflate, br
23 Connection: keep-alive
24 string=%3Cscript%3Ealert%28%27Hi+from+ali+hamdan%27%29%3C%2Fscript%3E+

```

Response Details:

```

1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 3143
4 Server: Werkzeug/0.14.1 Python/3.6.9
5 Date: Tue, 04 Mar 2025 18:30:58 GMT
6
7 <!DOCTYPE html>
8 <html>
9
10 <head>
11
12 <meta charset="utf-8">
13 <meta name="viewport" content="width=device-width, initial-scale=1">
14 <title>
15 rip Labs
16 </title>
17
18 <link href="/static/css/Normalize.css" rel="stylesheet">
19 <link href="/static/css/daterangepicker3.css" rel="stylesheet">
20 <link href="/static/css/styles.css" rel="stylesheet">
21
22 <!--Icons-->
23 <script src="/static/js/lumino.glyphs.js">
24 </script>
25 <script src="/static/js/hints.js">

```

Why XSS is Dangerous: XSS (Cross-Site Scripting) attacks enable attackers to inject malicious scripts into web pages, which are then executed in the browsers of unsuspecting users. This can lead to session hijacking, credential theft, and the manipulation of page content, undermining user trust and compromising sensitive data. CWE-79 relates to XSS vulnerabilities.

Solution: Employ input validation and output encoding to sanitize user data, implement Content Security Policies (CSP), and use secure frameworks that auto-escape outputs. These best practices help prevent malicious script injection and protect user data.

2.2 Path Traversal

Running the path traversal image

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

```

alihamdan@Alis-MacBook-Air-2 ~ % docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:path-traversal
Unable to find image 'sh3b0/vuln:path-traversal' locally
path-traversal: Pulling from sh3b0/vuln
c21a2ce6161f: Download complete
4f31cf9a1019: Download complete
4f4fb700ef54: Already exists
c500c92841ea: Download complete
9a77e59174fc: Download complete
1bf693f57e35: Download complete
Digest: sha256:4a29c3b12fa2dcf1e44b22f210e8b9785649fd00bd7cfc8655cf7dde9021a35e
Status: Downloaded newer image for sh3b0/vuln:path-traversal
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 199-818-631
172.17.0.1 - - [04/Mar/2025 18:39:41] "POST /home HTTP/1.1" 400 -
172.17.0.1 - - [04/Mar/2025 18:39:41] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [04/Mar/2025 18:39:45] "GET / HTTP/1.1" 200 -

```

Exploiting path traversal by modifying a request

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

After submitting

The screenshot shows a web browser window with the URL `127.0.0.1:5000/home`. The page title is "Local file inclusion/path traversal". Below the title, there is a "Selects Intro" dropdown menu and a "Submit Button". The browser's developer tools are open, showing the "Elements" panel with the form structure. The "Console" panel shows a series of log messages, including the successful retrieval of the file content.

```
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/bin/sh
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/spool/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
```

Captured in Burp Suite-1

The screenshot shows a captured HTTP request and response in Burp Suite. The request is a POST to `/home` with a `filename` parameter set to `../../../../etc/passwd`. The response is an HTML page with a header and body.

No.	Date	Host	Method	URI	Status	Size	Content-Type	
106	21:56:17 4 Mar 2025	Proxy	GET	fonts.googleapis.com/css2	200	1907	66	
107	21:56:17 4 Mar 2025	Proxy	GET	fonts.gstatic.com/s/hind/v17/5aU19_a8oxm1fNJ...	0	200	9182	82
108	21:56:58 4 Mar 2025	Proxy	POST	127.0.0.1/home	1	200	7947	9
109	21:56:58 4 Mar 2025	Proxy	GET	fonts.googleapis.com/css2	200	1907	77	
110	21:56:59 4 Mar 2025	Proxy	GET	fonts.gstatic.com/s/hind/v17/5aU19_a8oxm1fNJ...	0	200	9182	80

Request

```
POST /home HTTP/1.1
Host: 127.0.0.1:5000
Content-Length: 41
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
Origin: http://127.0.0.1:5000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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://127.0.0.1:5000/home
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
filename=../../../../etc/passwd
```

Response

```
<link href="/static/css/Normalize.css" rel="stylesheet">
<link href="/static/css/datepicker3.css" rel="stylesheet">
<link href="/static/css/styles.css" rel="stylesheet">

<!--Icons-->
<script src="/static/js/lumino.glyphs.js">
</script>
<script src="/static/js/hints.js">
</script>
<link href="https://fonts.googleapis.com/css2?family=Hind:wght@700&display=swap" rel="stylesheet">

</head>
<body>
  <header class="header">
    <div class="wrap wide">
      <div class="inner flx flx-ac flx-jsb">
        <div class="left flx flx-ac">
          <div>
            </div>
            <div class="name pl1">
```

Why Path Traversal is Dangerous: Path traversal vulnerabilities let attackers manipulate file paths to access restricted files, exposing sensitive data or compromising the system.

Solution: Sanitize file path inputs, restrict access to approved directories, and enforce strict file permissions.

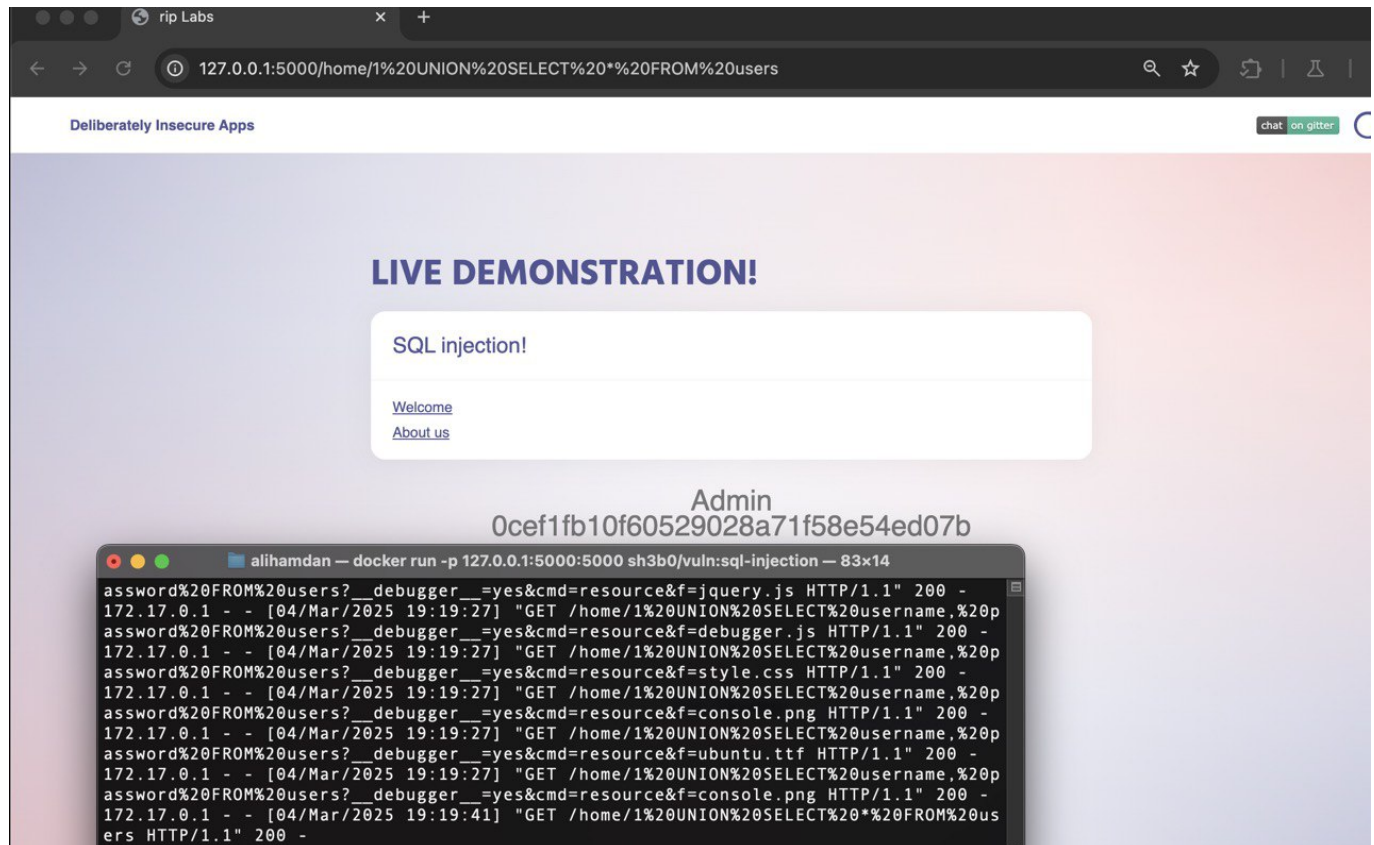
2.3 SQL Injection

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: Explanation: SQL Injection is dangerous because it lets attackers manipulate queries to access, modify, or delete sensitive data, potentially compromising the entire system. Solution: Use parameterized queries, validate inputs, enforce least privilege, and perform regular security reviews.

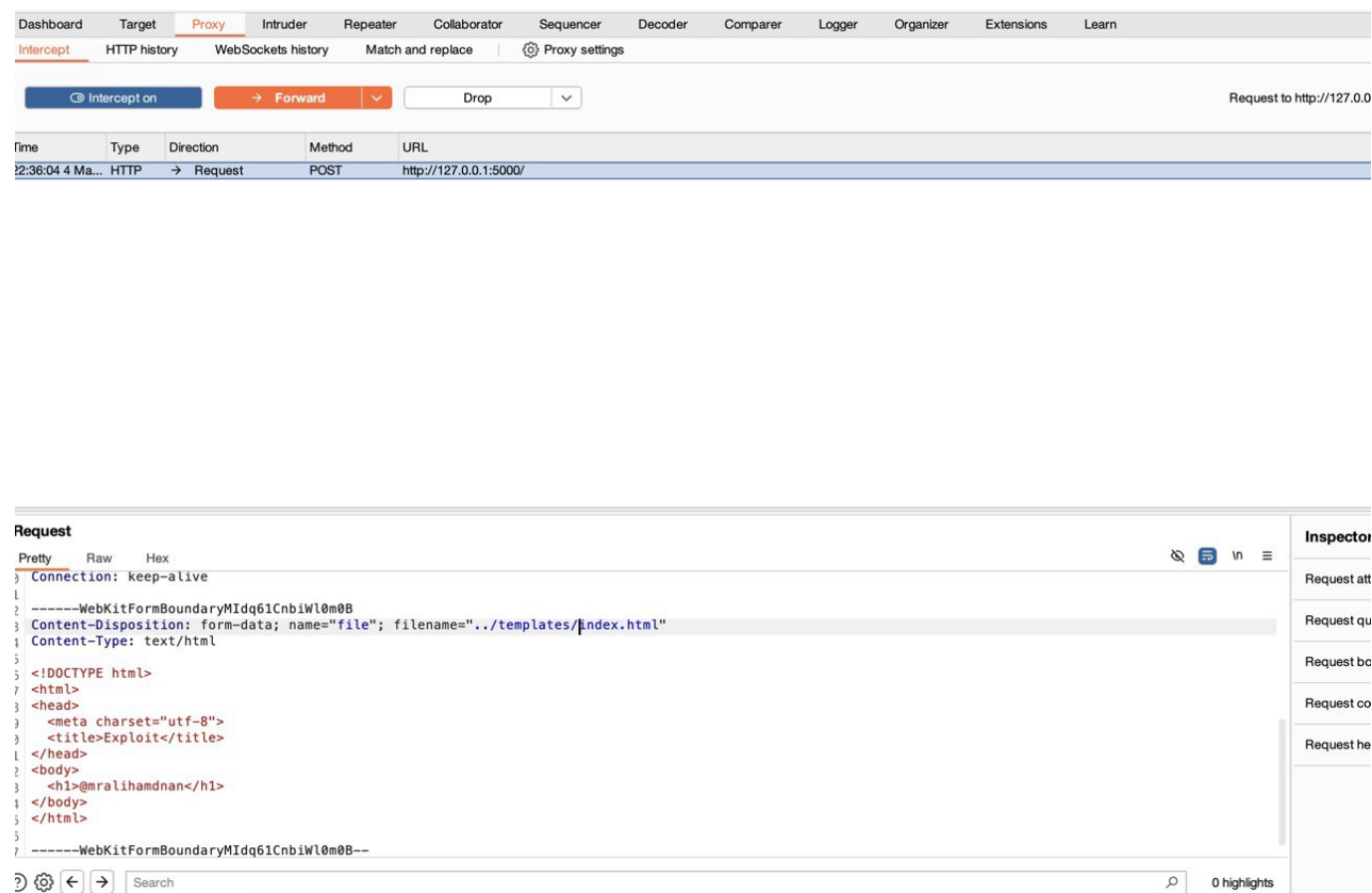
2.4 File Upload Exploit

Running the file upload image

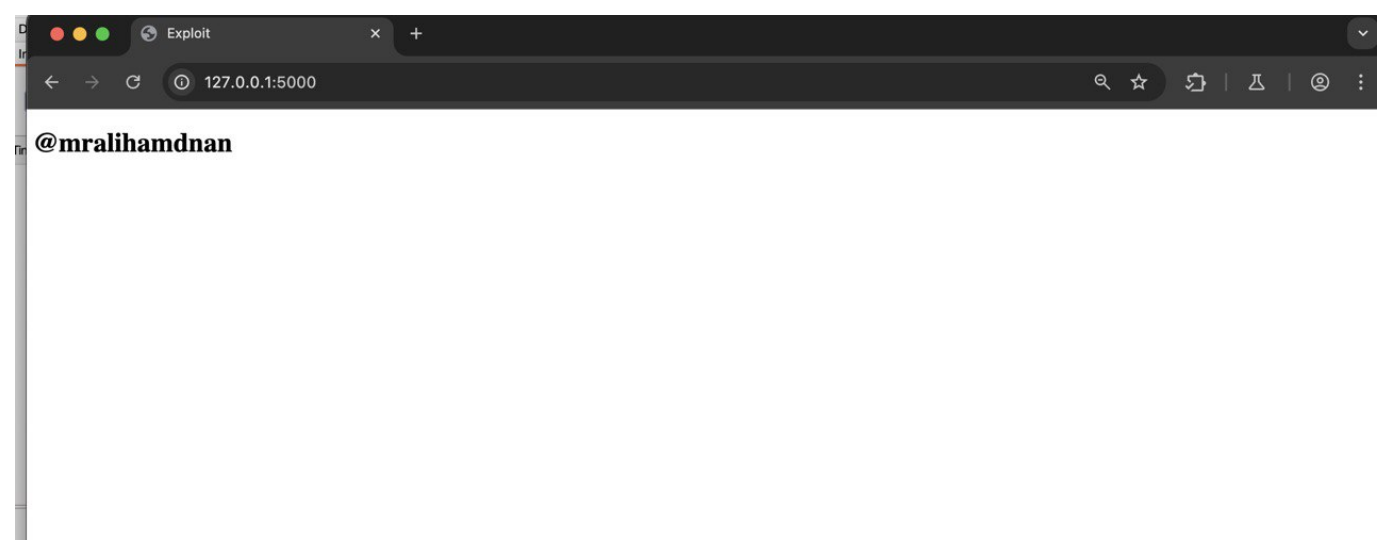
```
docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:file-upload
```

Bypassing file upload restrictions

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



Results



Why Unrestricted File Upload is Dangerous: Explanation: File upload vulnerabilities are dangerous because they enable attackers to upload malicious files, such as web shells, which can lead to remote code execution and unauthorized access.

Solution: Validate file types, sanitize filenames, store uploads outside the web root, and enforce file size limits to protect the system.

2.5 Command Injection

Running the command injection image

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

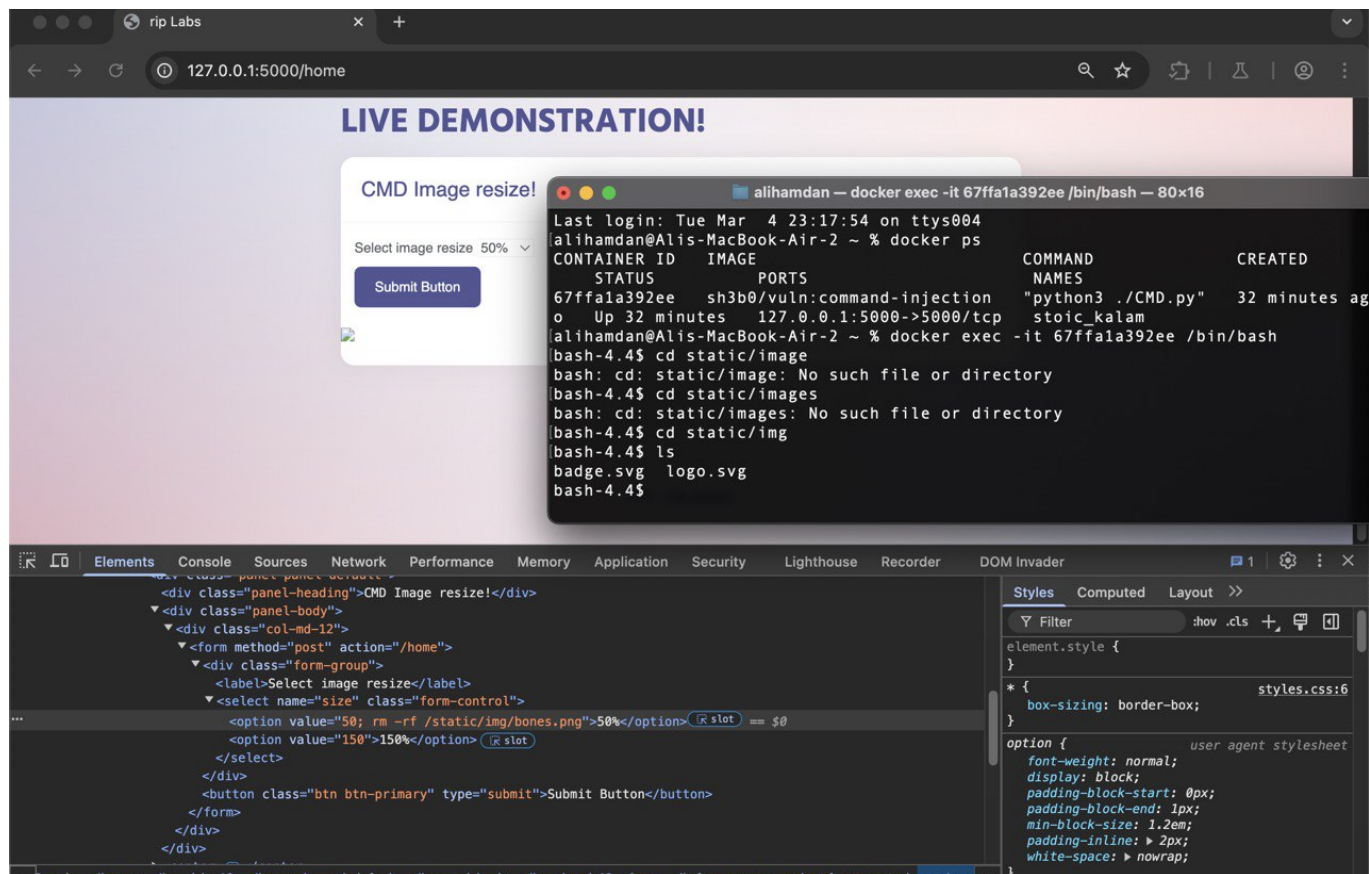
```
alihamdan@Alis-MacBook-Air-2 ~ % docker run -p 127.0.0.1:5000:5000 sh3b0/vuln:command-injection
Unable to find image 'sh3b0/vuln:command-injection' locally
command-injection: Pulling from sh3b0/vuln
f61c0fb18c6a: Download complete
9dc528f883ed: Download complete
16a1e357d681: Download complete
4f4fb700ef54: Already exists
d74022104470: Download complete
5f2812bb7c7f: Download complete
77cc860d68fd: Download complete
d3f9eb6ecca0: Download complete
8496d86fc735: Download complete
9f3af57112c1: Download complete
Digest: sha256:9dac3a070309af595902684f2a6f56bbd96a0ff01b8f014b899ac87d1927a659
Status: Downloaded newer image for sh3b0/vuln:command-injection
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 199-818-631
```

Injecting a command into an HTML element

Modifying a field to include:

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

After injection



Why Command Injection is Dangerous: Explanation: Command injection vulnerabilities allow attackers to execute arbitrary system commands on the host server. This can lead to unauthorized access, data breaches, or full system compromise.

Solution: Validate and sanitize all user inputs, use parameterized APIs or libraries that avoid command concatenation, and enforce strict allow-listing of commands. Additionally, run services with the least privileges and isolate untrusted inputs to minimize potential damage.