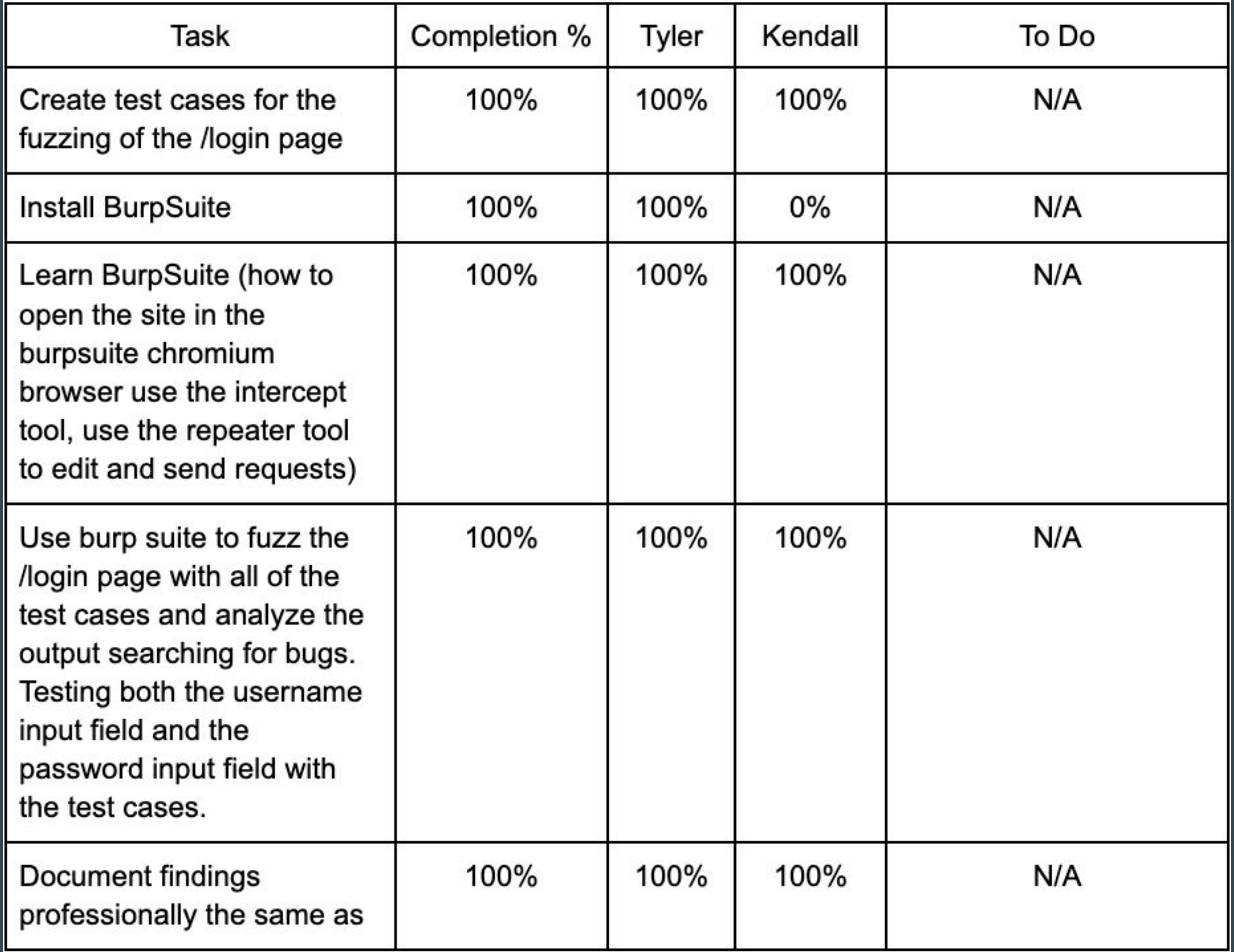
Automated Transfer Credit Evaluator (ATCE)
Milestone 6 Evaluation

•••

Tyler Dionne & Kendall Kelly

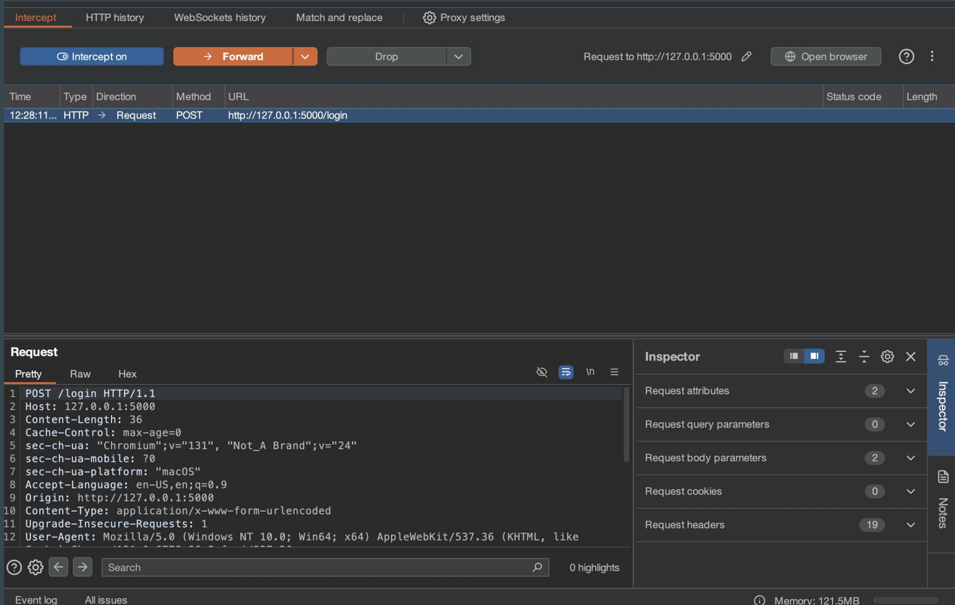


Task	Completion %	Tyler	Kendall	To Do
Create test cases for the fuzzing of the /login page	100%	100%	100%	N/A
Install BurpSuite	100%	100%	0%	N/A
Learn BurpSuite (how to open the site in the burpsuite chromium browser use the intercept tool, use the repeater tool to edit and send requests)	100%	100%	100%	N/A
Use burp suite to fuzz the /login page with all of the test cases and analyze the output searching for bugs. Testing both the username input field and the password input field with the test cases.	100%	100%	100%	N/A
Document findings professionally the same as	100%	100%	100%	N/A

a professional fuzzing environment				
Install ZAP	100%	100%	0%	N/A
Learn how to use ZAP (load in target address and then run the Spider auto analyzer tool to generate a report)	100%	0%	100%	N/A
Analyze the ZAP report taking into account the findings under the “Alerts” tab analyzing each entry and the severity	100%	100%	100%	N/A
Overview of general web application security improvements that can be made in the main Flask python file to create an overall secure application that resists common exploits	100%	100%	100%	N/A

Testing Login and Registration with Burpsuite

- Intercept login page requests to test security
- Run the Flask app locally and then use BurpSuite to intercept/modify requests
- We focused on input validation, error handling, and security configurations



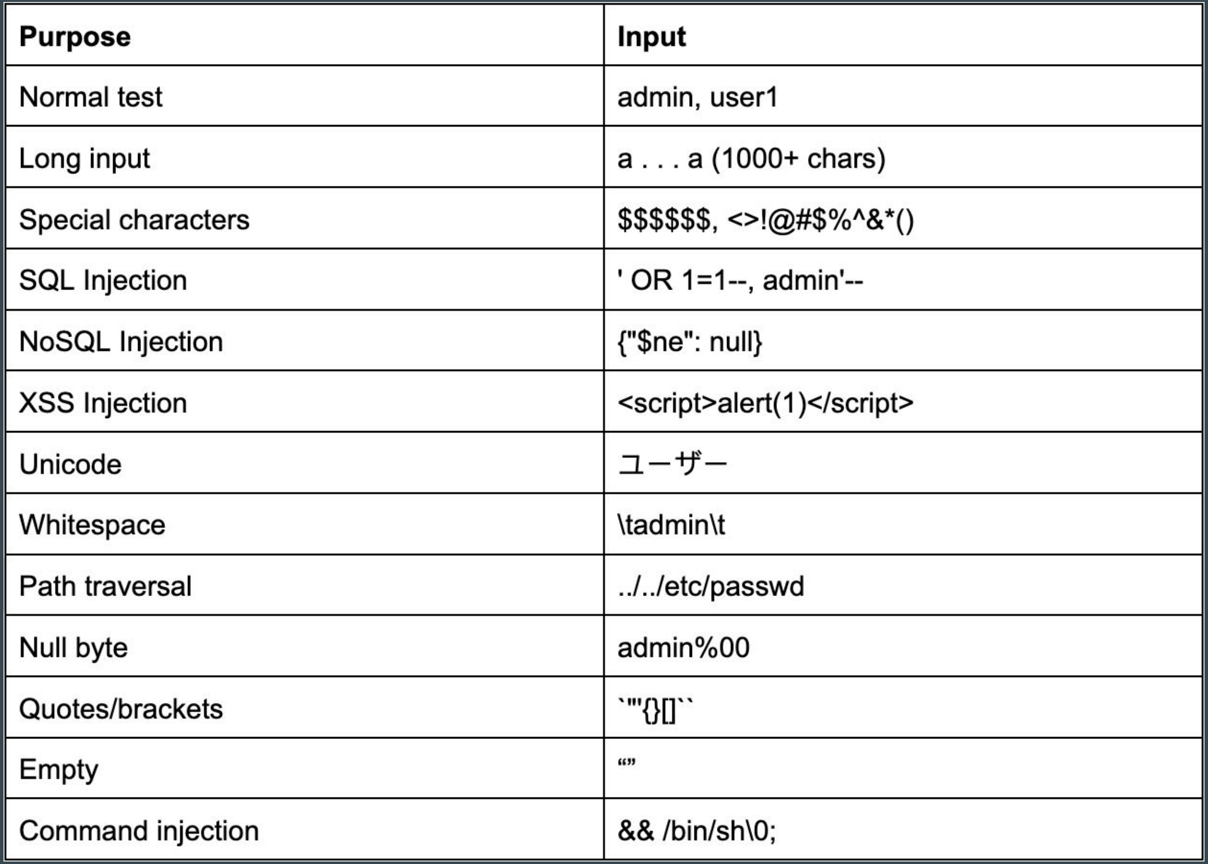
Time	Type	Direction	Method	URL	Status code	Length
12:28:11...	HTTP	→ Request	POST	http://127.0.0.1:5000/login		

Request

Pretty Raw Hex

```
POST /login HTTP/1.1
Host: 127.0.0.1:5000
Content-Length: 36
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
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 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
```

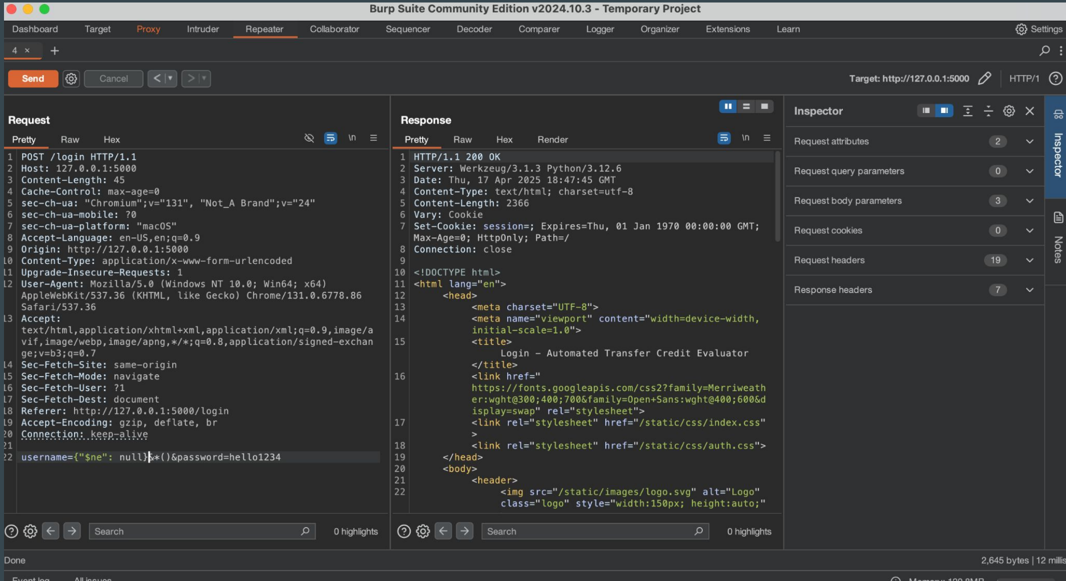
Login Page Fuzzing Test Cases



Purpose	Input
Normal test	admin, user1
Long input	a . . . a (1000+ chars)
Special characters	$$$$$$, <>!@#$%^&*()
SQL Injection	' OR 1=1--, admin'--
NoSQL Injection	{"$ne": null}
XSS Injection	<script>alert(1)</script>
Unicode	ユーザー
Whitespace	\tadmin\t
Path traversal	../../etc/passwd
Null byte	admin%00
Quotes/brackets	`"'{}[]``
Empty	“”
Command injection	&& /bin/sh\0;

BurpSuite Testing Results

- All results and outputs were normal
- Application showed consistent error handling for all of our test cases
- No vulnerabilities were discovered during BurpSuite Fuzzing

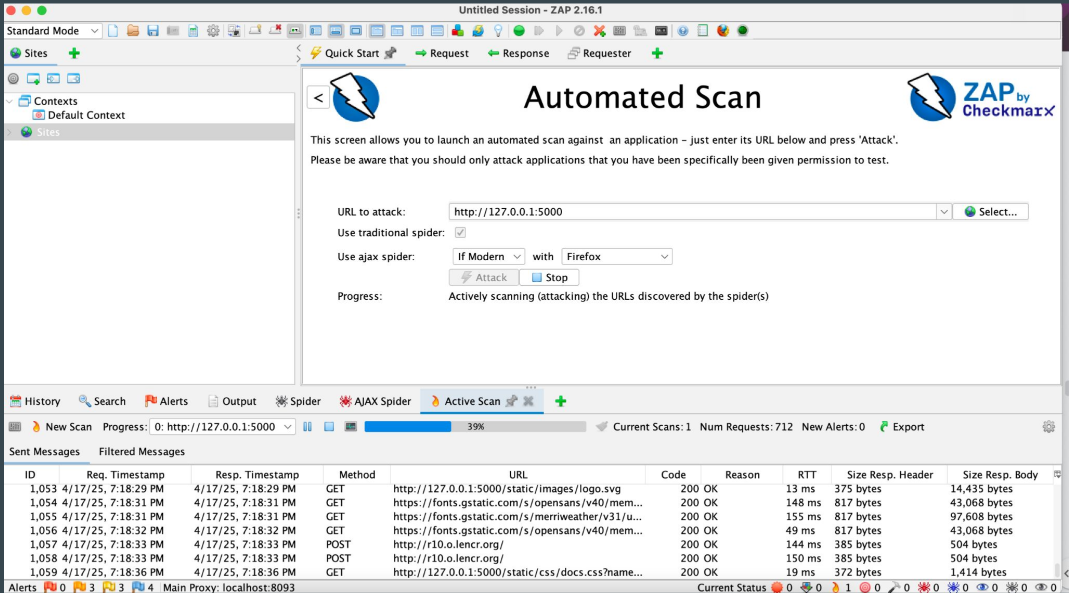

Burp Suite Community Edition v2024.10.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Send Cancel
Target: http://127.0.0.1:5000 HTTP/1
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 3
Request cookies 0
Request headers 19
Response headers 7
Search 0 highlights
Done
2,645 bytes | 12 millis

```
POST /login HTTP/1.1
Host: 127.0.0.1:5000
Content-Length: 45
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
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 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;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/login
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

username={"$ne": null}&*()&password=hello1234
```

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.12.6
Date: Thu, 17 Apr 2025 18:47:45 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2366
Vary: Cookie
Set-Cookie: session=; Expires=Thu, 01 Jan 1970 00:00:00 GMT;
Max-Age=0; HttpOnly; Path=/
Connection: close

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
        <title>
            Login - Automated Transfer Credit Evaluator
        </title>
        <link href="
        https://fonts.googleapis.com/css2?family=Merriweath
        er:wght@300;400;700&family=Open+Sans:wght@400;600&d
        isplay=swap" rel="stylesheet">
        <link rel="stylesheet" href="/static/css/index.css"
        >
        <link rel="stylesheet" href="/static/css/auth.css">
    </head>
    <body>
        <header>
            <img src="/static/images/logo.svg" alt="Logo"
            class="logo" style="width:150px; height:auto;"
```

OWASP ZAP Vulnerability Scan

- Run ZAP
- Start new session
- Set our Flask app as target
- Purpose is to identify common web vulnerabilities automatically



This screen allows you to launch an automated scan against an application – just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://127.0.0.1:5000
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox
Attack Stop
Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

Progress: 0: http://127.0.0.1:5000 39% Current Scans: 1 Num Requests: 712 New Alerts: 0 Export

Sent Messages Filtered Messages

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
1,053	4/17/25, 7:18:29 PM	4/17/25, 7:18:29 PM	GET	http://127.0.0.1:5000/static/images/logo.svg	200	OK	13 ms	375 bytes	14,435 bytes
1,054	4/17/25, 7:18:31 PM	4/17/25, 7:18:31 PM	GET	https://fonts.gstatic.com/s/opensans/v40/mem...	200	OK	148 ms	817 bytes	43,068 bytes
1,055	4/17/25, 7:18:31 PM	4/17/25, 7:18:31 PM	GET	https://fonts.gstatic.com/s/merriweather/v31/u...	200	OK	155 ms	817 bytes	97,608 bytes
1,056	4/17/25, 7:18:32 PM	4/17/25, 7:18:32 PM	GET	https://fonts.gstatic.com/s/opensans/v40/mem...	200	OK	49 ms	817 bytes	43,068 bytes
1,057	4/17/25, 7:18:33 PM	4/17/25, 7:18:33 PM	POST	http://r10.o.lencr.org/	200	OK	144 ms	385 bytes	504 bytes
1,058	4/17/25, 7:18:33 PM	4/17/25, 7:18:33 PM	POST	http://r10.o.lencr.org/	200	OK	150 ms	385 bytes	504 bytes
1,059	4/17/25, 7:18:36 PM	4/17/25, 7:18:36 PM	GET	http://127.0.0.1:5000/static/css/docs.css?name...	200	OK	19 ms	372 bytes	1,414 bytes

Alerts 0 3 3 4 Main Proxy: localhost:8093 Current Status 0 0 1 0 0 0 0 0 0 0

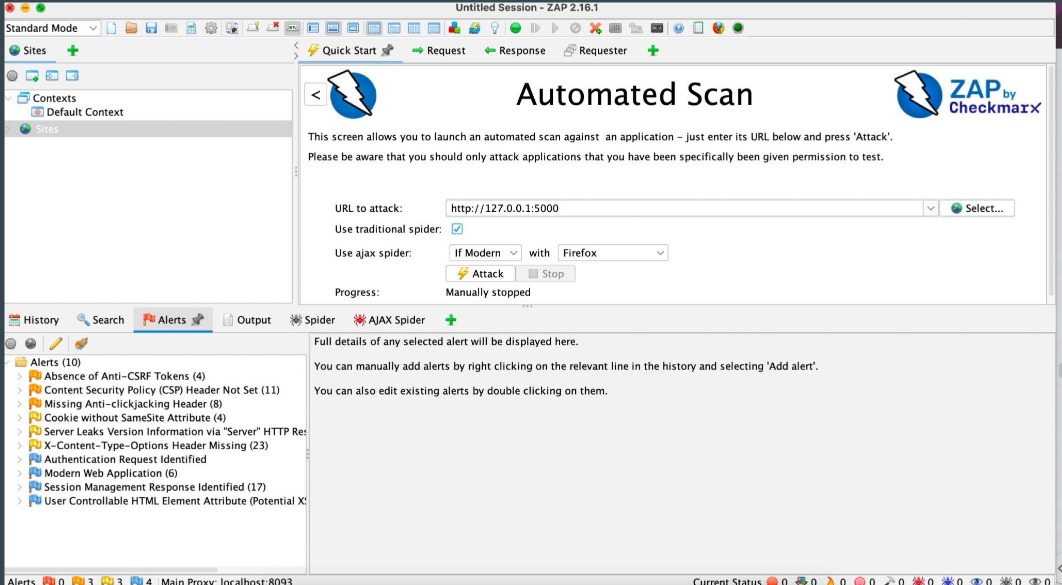
ZAP Vulnerability Report

- Absence of Anti CSRF Tokens (highest severity)

- Cross-Site Request Forgery (CSRF)
  - Malicious website tricks users browser into making unwanted request to another site where that user is authenticated.

- Ex. While logged into bank, visiting a malicious site secretly submits transfer request

- Implications:
  - Attackers can perform actions without your knowledge
  - Could lead to changes in your account or data theft
  - Most dangerous when admin accounts are targeted


Untitled Session - ZAP 2.16.1

Automated Scan

This screen allows you to launch an automated scan against an application – just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://127.0.0.1:5000

Use traditional spider: ✓

Use ajax spider: If Modern with Firefox

Progress: Manually stopped

Alerts (10)
- Absence of Anti-CSRF Tokens (4)
- Content Security Policy (CSP) Header Not Set (11)
- Missing Anti-clickjacking Header (8)
- Cookie without SameSite Attribute (4)
- Server Leaks Version Information via "Server" HTTP Res
- X-Content-Type-Options Header Missing (23)
- Authentication Request Identified
- Modern Web Application (6)
- Session Management Response Identified (17)
- User Controllable HTML Element Attribute (Potential X

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 0 3 3 4 Main Proxy: localhost:8093

Security Improvements

- Input validation

```
def sanitize_input(user_input):
    # take out special characters and limit input length
    return ''.join(char for char in user_input if char.isalnum() or char.isspace())[:50]
```

- Prevents injection attacks (SQL, Command, and XSS)

- CSRF Protection using Flask-WTF

```
from flask_wtf.csrf import CSRFProtect
…
csrf = CSRFProtect(app)
app.config['SECRET_KEY'] = 'your-secret-key'
```

- Prevents cross-site request forgery attacks

Security Improvements (Cont.)

- Rate Limiting

```
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address

limiter = Limiter(
    app,
    key_func=get_remote_address,
    default_limits=["200 per day", "50 per hour"]
)
```

- Prevents brute force and DoS attacks

- Security Headers

```
@app.after_request
def add_security_headers(response):
    response.headers['X-Content-Type-Options'] = 'nosniff'
    response.headers['X-Frame-Options'] = 'DENY'
    response.headers['X-XSS-Protection'] = '1; mode=block'
    return response
```

- Protects against client-side exploits

Lessons Learned

- Gained experience with tools for web app security
  - BurpSuite - intercept and modify requests
  - ZAP - automated vulnerability scanning
- Developed skills:
  - Flask framework and security implementation
  - Login page fuzzing and assessing vulnerabilities
  - Database integration with security
- Web app development and web security are complex
  - No straightforward path
  - Different applications will require different approaches to security
  - Security should be implemented throughout development