Social Insecurity - Penetration Testing Report

Single group: Vegard Rongve Student number: 225590

Introduction

Tester: Vegard Rongve

Application: Social Insescurity

Test period: 01.09.2023 - 27.09.2023

The social insecurity web application is a platform where users can connect with friends and share blog posts, encompassing both textual and image-based content. After uploading a post, other users are allowed to comment on eachothers posts. All users of the app will first need to register before logging in.

The primary ojective of this assignment is to systimatically identify potential vulnerabilities within social insecurity web application, and subsequentially explore methods to exploit them. This assignment is made to improve our abilities to think like potential attackers, with the ultimate aim of enhancing our future application development practice to priotize security or our abilties to work as a penetration tester.

Vulnerability Assessment

Black Box Testing

Weak Password

Category:

- A07: Identification and Authentication Failures

Descirption:

- Users of the application have complete freedom to choose their own passwords, without any strict password policy in place. This means they can even select a single-letter password.

Potential Impact:

- The potential impact of having a weak password is significant, as an attacker might get hold sensitiv data (depending on the purpose of the application). An attacker can by using for instance the brut force technique, relatively easy break the password and get access to a user account.
-

Affected part of the application:

- The affected part is the registration form of the application where the user is setting a password.

User enumeration

Category:

- A05: Security Misconfiguration

Description:

- Interacting with the application allows users to identify valid usernames. When a user enters a valid username during the login process, the application indicates that the password is incorrect. Conversely, if an invalid username is entered, the application informs the user that the user does not exist.

Potential impact:

- It makes it easier for malicous actors to identify valid usernames. Armed with a list of valid usernames, attackers can launch various credential-based attacks, such as bruce force attacks.
- User enumartion can be seen as a privacy violation, as i exposes information about which usernames are registered with the service.

Affected part of the applicaton:

- The affected part of the app is the login section. The application provides a notification message in the top of the broweser window when an invalid password or username is entered.

Bypassing access control by modifying URL

Cateogory:

- A01:Broken Access Control

Description:

- By modifying the url, given a valid username, it is possible to login in to the application as that user without entering the password.

Potential impact:

- It allows unauthorized access to an account by simply knowing the username, which could lead to unauthorized data access, account manipulation, or other securtiy risks.

Affected part of the application:

- The Broken Access Control issue affects the user authentication and authorization system of the application. Specifically, it allows unauthorized access to user accounts by manipulating the URL with a valid username, granting access to the user's account without requiring the correct password.

Edit another user's profile

Cateogory:

- A01:Broken Access Control

Description:

- The app is lacking session management. Currently an attacker is able to modify anyones profile by for instance adding the user as friend, and then pressing on the username of that user. By doing so the user will navigate to the profil page of the selected user. Here, a user is able to press the edit button and then modify the profile to another user.

Potential impact:

- The potential impact of the describe issue is significant and could have several adverese consequences for the application's security and user data.
  - Unauthorized profile modification:
    - Attackers can modify the profiles of others without their consent or proper authorization. This could lead to the unauthorized modification of personal inforamtion or other user-specific data, potentially causing confusion, misinformation, or privacy breach.
  - Data integrity:
    - Unauthorized profile modification could result in a loss of data integrity as user profiles may contain critical information related to their identify, preferences, or interactions with the application. This could impact the trustworthiness of the platform.

Affected part of the application:

- The affected part of the described issue above is the profle page where the user can edit data about their self.

File upload

Category:

- A03:Injection

Description:

- The web app does not give an guidelines for what kind of files that can be uploaded and not. It looks like the app accepts any kind of file extensions. This means that an attacker might potentially upload malicious and dangenrous files.

Potential Impact:

- Malware distribution. Users can upload malicious files such as viruses, Trojans, or ransomware. These files may be disguised as innocent documents, images, or other harmless formats, making it easier for attackers to spread malware to other users who download or open these files.
- Uploading certain file types can introduce security vulnerabilities to your server. For example, allowing users to upload scripts (e.g., PHP, JavaScript) can lead to potential code execution vulnerabilities if the server doesn't properly validate and sanitize user inputs.

Affacted part of the application:

- The affected part the of the application is the stream page, where users can upload a text and/or image which they want others to see. These post will then be visible to all users of the application, meaning that malicious scripts could potenially reach out to all the users.

Injecting code into database

Category:

- A03:Injection

Description

- Due to bad/no form validation i'm able to post a script to the server by entering the script as text in the comment field on the stream page. And due to the lack of output encoding, the web application threats this as a javascript each time the page refreshes.

Potential impact:

- The script injected to the database could potentially be used to create forms where a user might enter sensitive data and sends it to an attacker when saved. So the potential consquence of these kind of attacks are severe.

Affected part fo the application:

- The stream page where all the posts are listed.
- The edit profil page.

Multiple users with same password and username

Category:

- A01:Broken Access Control

Description:

- In the social security app, I have identified a critical issue. The system allows the registration of multiple users with identical usernames due to a lack of essential server-side validation checks

Potential impact:

- This vulnerability is severe risk as it leads to confusion among users, making it potentially impossible for a logged-in user to discern their actual identify. Furthermore, when making data modifications, the actions of the one logged-in user affect all users sharing the same username, resulting in significant data loss and integrity compromise.

Affected part of the application:

- The issue starts with the registration form, where it is possible to register multiple users with the same username. The problem extends throughout the application, impacting all pages where data is modified and stored in the database, as all users sharing the same username are affected.

White Box Testing

Password hashing

Category:

- A02:Cryptographic Failures

Description:

- By looking at the source code no password hashing are identified, which is counted as an cryptographic failure by OWASP, and it is actually on the second place on the OWASP Top Ten Vulnerabilities list.

Potential impact:

- Storing plaintext passwords in a database makes it easier for attackers to steal user credentials if they gain access to the database. A security breach can lead to unauthorized actions taken on behalf of the user.
- Many users reuse passwords across multiple services. If your application does not hash passwords, attackers can potentially reuse these stolen credentials to access the users account on other platforms

Affected part of the application:

- Register user and login.

Vulnerability Exploitation

1. Broken Access Control

Step.1

I registered two user in the app, "vrongve" and "trongve". After registering the two users, i logged in and updated the about page for "vrongve".

Step.2

After updating the user profile for "vrongve" i logged out of the app by cliking the Log Out button. Then i logged in as "trongve" and pressed the friends page where i added the user "vrongve" as a friend.

Step.3

After seeing that "vrongve" appeard as my frind in the list I clicked on the "vrongve" username link, which then took me to the about page of that user.

Step.4

Now the logged in user, which currently is "trongve", is able to press the edit button and then edit the about form for "vrongve.

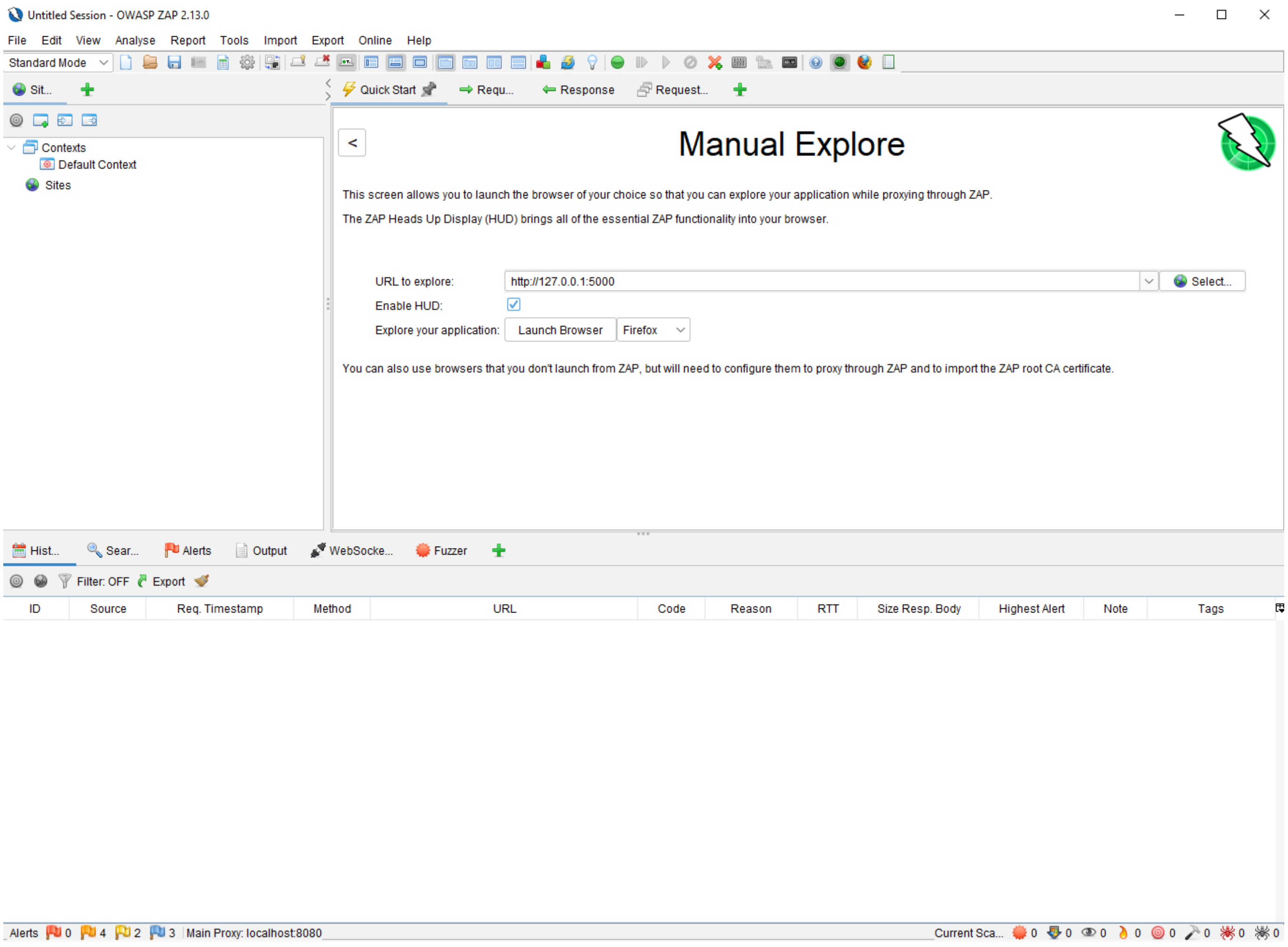
Result:

- After exploiting the broken access control of the app I was able to entierly delete or modify all the userinfo entered by any user on the profile page in the web application. As long an attacker knows the username the attacker may add the user as friend and then clicking on the username to navigate to the profile page or simply just modifying the URL by changing the username. If an attacker would enter the following URL "http://127.0.0.1:5000/profile/vrongve" the attacker would be able to modify the profile page for that user.

2. Identification and Authentication Failures

Step.1

Download OZASP Zap and launch the web page through the manual explorer in OWASP Zap.



This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.

The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore: http://127.0.0.1:5000

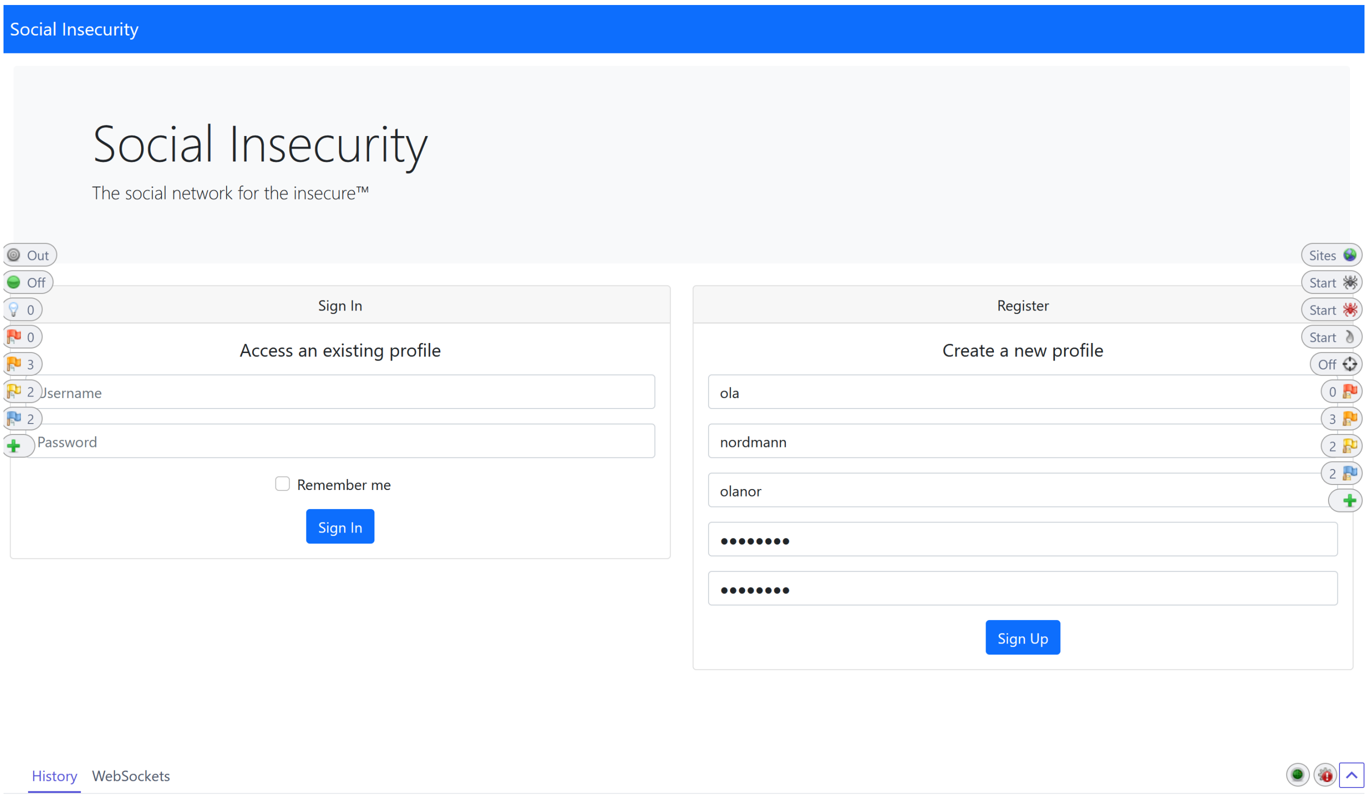
Enable HUD: ✓

Explore your application: Launch Browser Firefox

You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.

Step.2

Register a user with username "olanor" and password "abcd1234".


Social Insecurity

Social Insecurity

The social network for the insecure™

Sign In

Access an existing profile

Username

Password

Remember me

Sign In

Register

Create a new profile

ola

nordmann

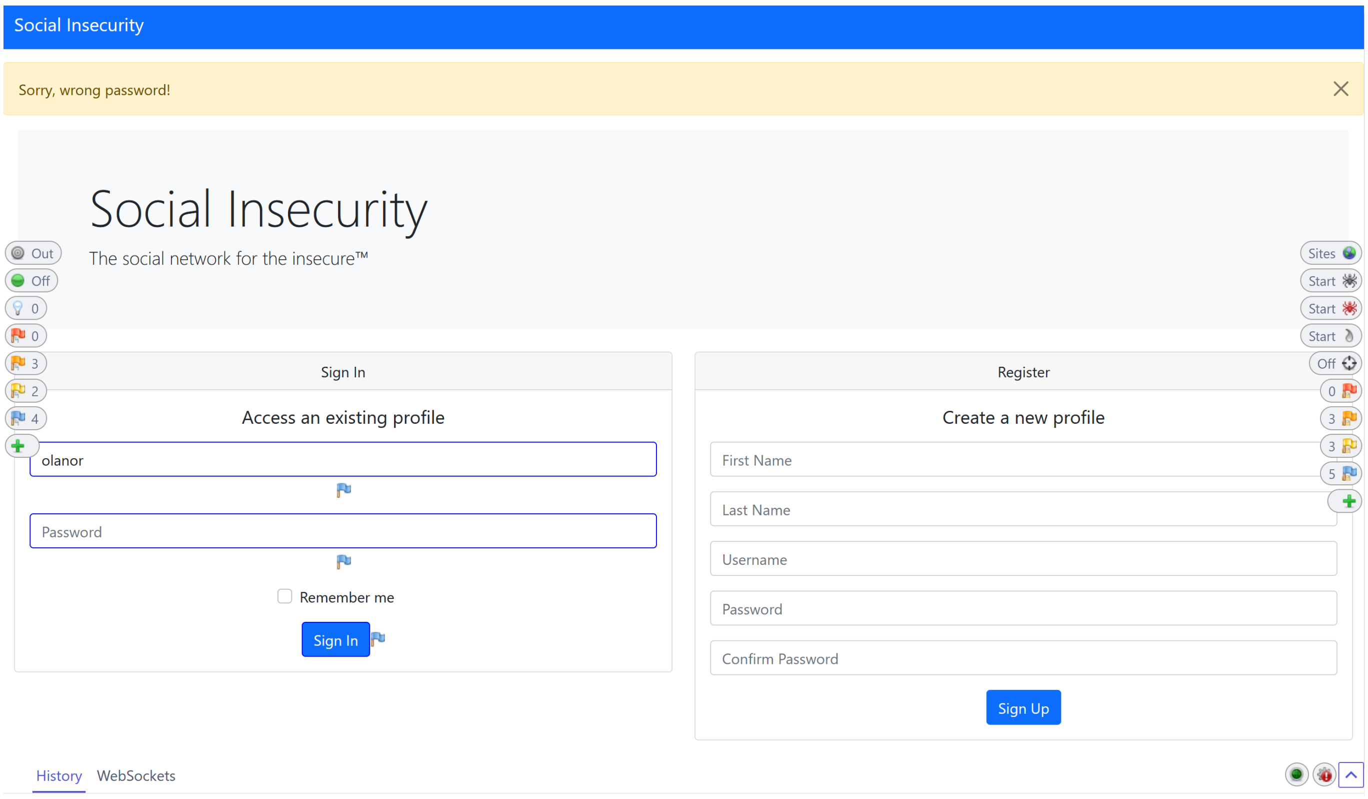
olanor

••••••••

••••••••

Sign Up

After registering the user, try logging in with username "olanor", but set the password equal to "changeme". The application will then tell that the password is incorrect.


Social Insecurity

Sorry, wrong password!

Social Insecurity

The social network for the insecure™

Sign In

Access an existing profile

olanor

Password

Remember me

Sign In

Register

Create a new profile

First Name

Last Name

Username

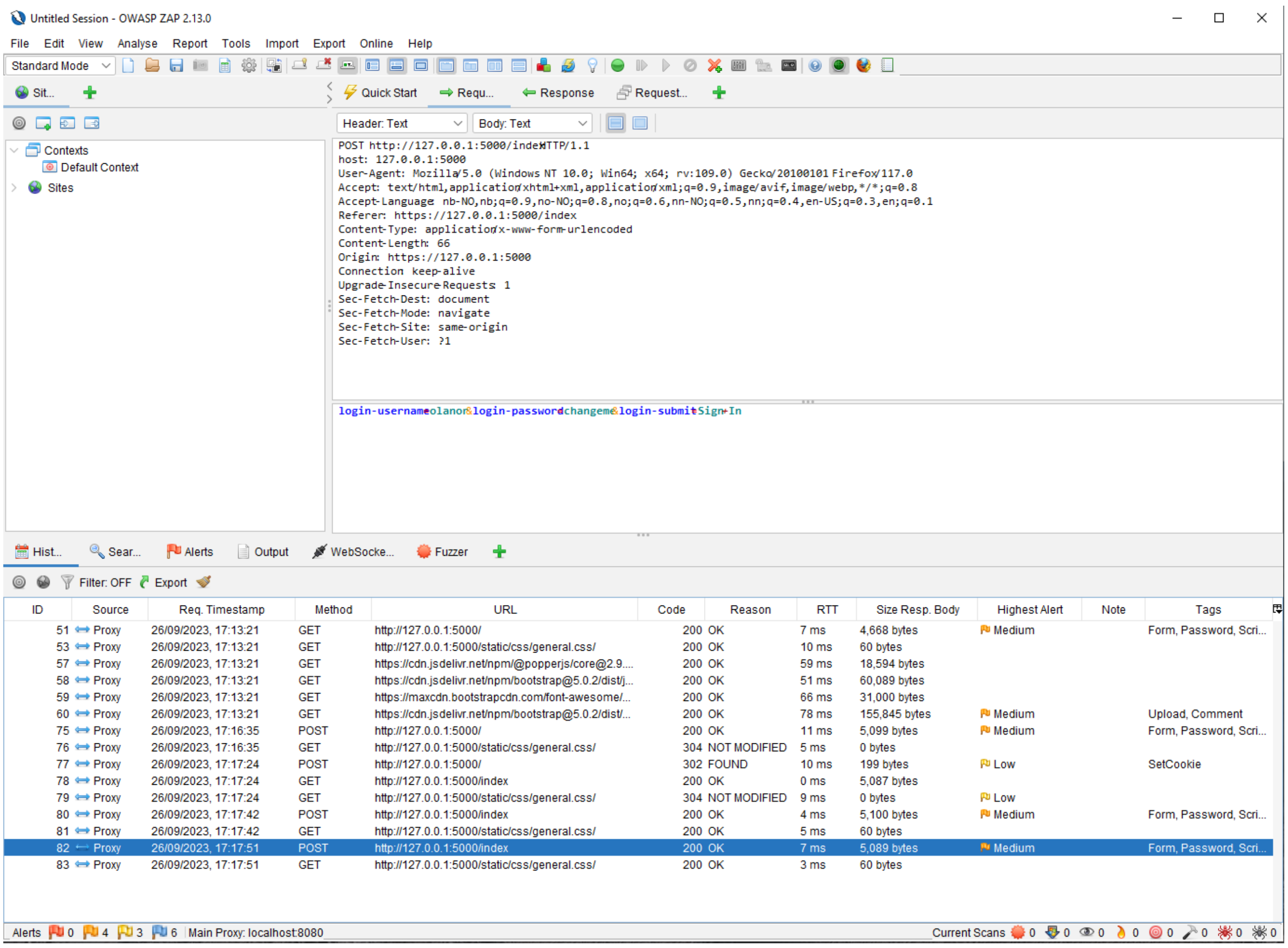
Password

Confirm Password

Sign Up

Step.3

No i can go into the OWASP Zap application and look at the latest POST request. I highlighted the POST line in the history view and then pressed the "request" tab in the OWASP Zap application. From this view i can see that the username entered was "olenor" and the password entered was "changeme".

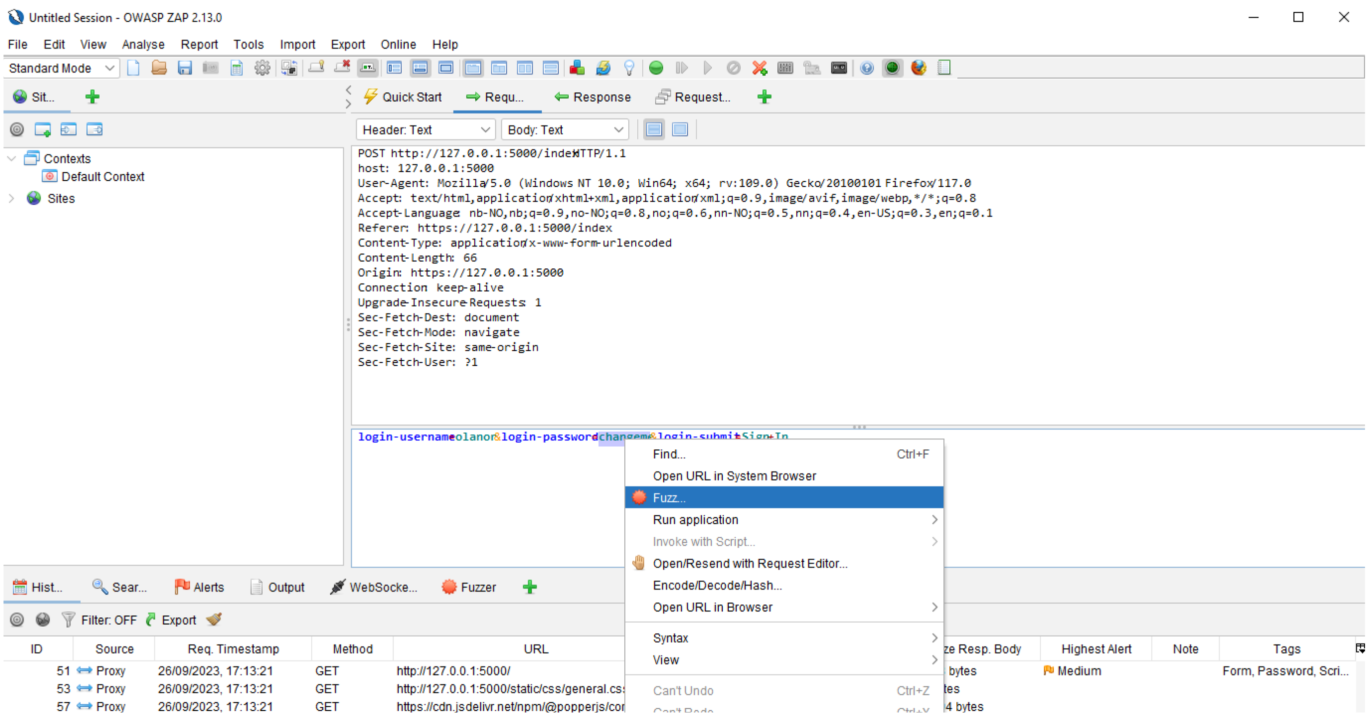


```
POST http://127.0.0.1:5000/index HTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.1:5000/index
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: https://127.0.0.1:5000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

login-username=olanor&login-password=changeme&login-submit=Sign+In
```

ID	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
51	Proxy	26/09/2023, 17:13:21	GET	http://127.0.0.1:5000/	200	OK	7 ms	4,668 bytes	Medium		Form, Password, Scri...
53	Proxy	26/09/2023, 17:13:21	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	10 ms	60 bytes			
57	Proxy	26/09/2023, 17:13:21	GET	https://cdn.jsdelivr.net/npm/@popperjs/core@2.9....	200	OK	59 ms	18,594 bytes			
58	Proxy	26/09/2023, 17:13:21	GET	https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/j...	200	OK	51 ms	60,089 bytes			
59	Proxy	26/09/2023, 17:13:21	GET	https://maxcdn.bootstrapcdn.com/font-awesome/...	200	OK	66 ms	31,000 bytes			
60	Proxy	26/09/2023, 17:13:21	GET	https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/...	200	OK	78 ms	155,845 bytes	Medium		Upload, Comment
75	Proxy	26/09/2023, 17:16:35	POST	http://127.0.0.1:5000/	200	OK	11 ms	5,099 bytes	Medium		Form, Password, Scri...
76	Proxy	26/09/2023, 17:16:35	GET	http://127.0.0.1:5000/static/css/general.css/	304	NOT MODIFIED	5 ms	0 bytes			
77	Proxy	26/09/2023, 17:17:24	POST	http://127.0.0.1:5000/	302	FOUND	10 ms	199 bytes	Low		SetCookie
78	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/index	200	OK	0 ms	5,087 bytes			
79	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/static/css/general.css/	304	NOT MODIFIED	9 ms	0 bytes	Low		
80	Proxy	26/09/2023, 17:17:42	POST	http://127.0.0.1:5000/index	200	OK	4 ms	5,100 bytes	Medium		Form, Password, Scri...
81	Proxy	26/09/2023, 17:17:42	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	5 ms	60 bytes			
82	Proxy	26/09/2023, 17:17:51	POST	http://127.0.0.1:5000/index	200	OK	7 ms	5,089 bytes	Medium		Form, Password, Scri...
83	Proxy	26/09/2023, 17:17:51	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	3 ms	60 bytes			

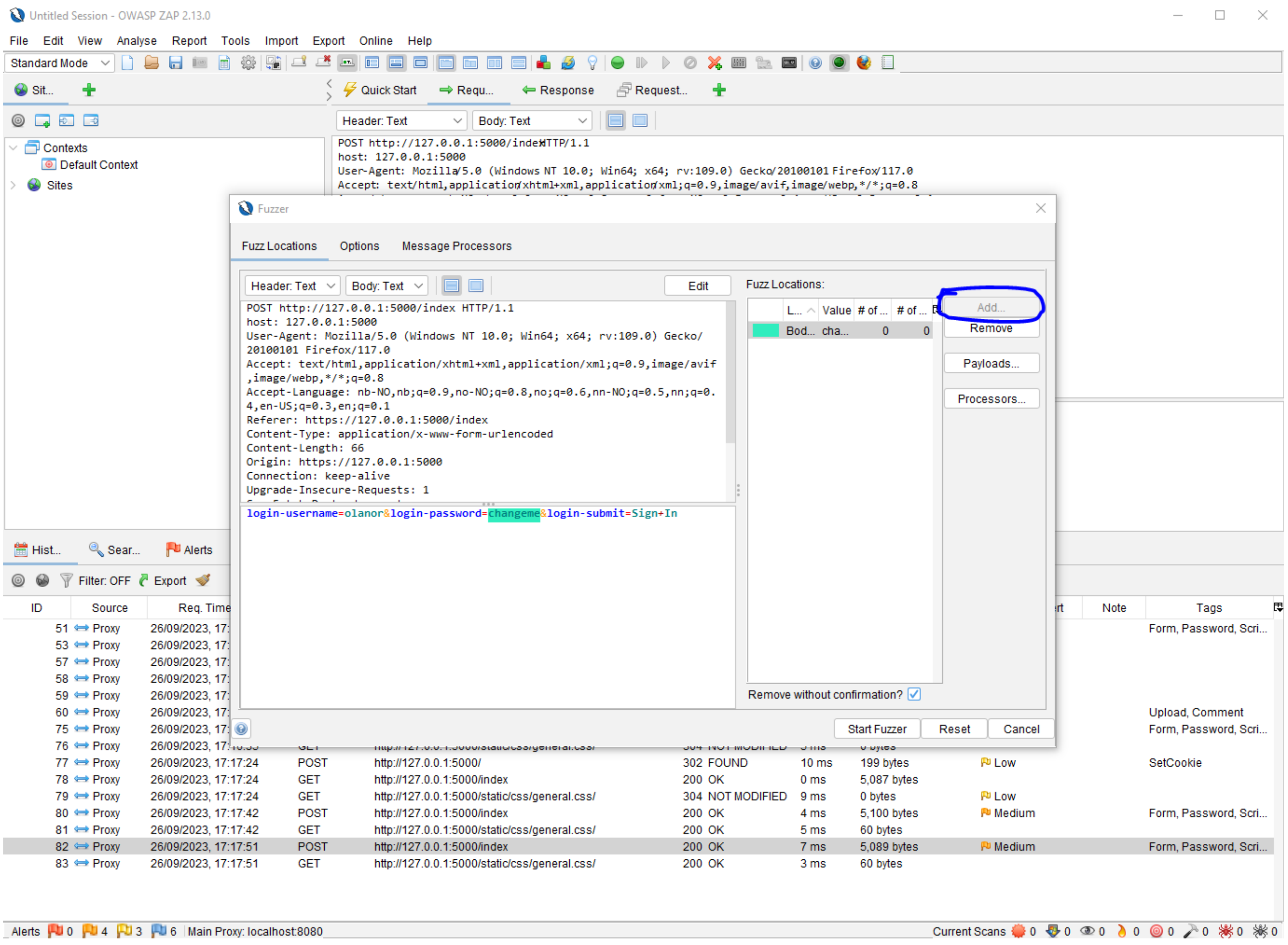
Step.4

After locating the "changeme" password I highlighted "changeme" and right-clicked. Then i selected "Fuzz" -> "Add" -> "Add" -> Type "File".



```
POST http://127.0.0.1:5000/index HTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.1:5000/index
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: https://127.0.0.1:5000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

login-username=olanor&login-password=changeme&login-submit=Sign+In
```

ID	Source	Req. Timestamp	Method	URL	Size Resp. Body	Highest Alert	Note	Tags
51	Proxy	26/09/2023, 17:13:21	GET	http://127.0.0.1:5000/	bytes	Medium		Form, Password, Scri...
53	Proxy	26/09/2023, 17:13:21	GET	http://127.0.0.1:5000/static/css/general.cs	tes			
57	Proxy	26/09/2023, 17:13:21	GET	https://cdn.jsdelivr.net/npm/@popperjs/cor	4 bytes			



```
POST http://127.0.0.1:5000/index HTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/
20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.
4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.1:5000/index
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: https://127.0.0.1:5000
Connection: keep-alive
Upgrade-Insecure-Requests: 1

login-username=olanor&login-password=changeme&login-submit=Sign+In
```

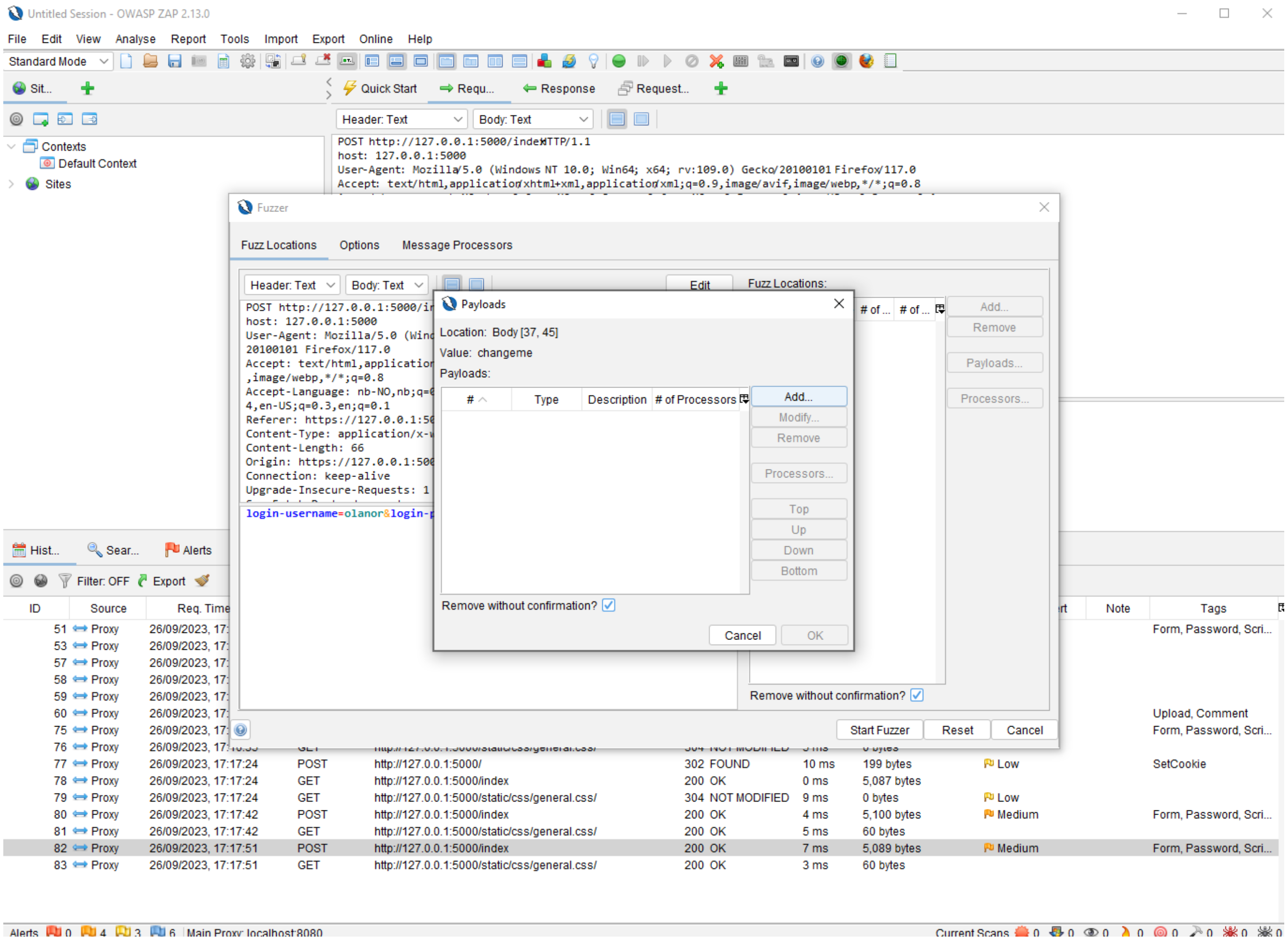
Fuzz Locations:

L...	Value	# of ...	# of ...
Bod...	cha...	0	0

Remove without confirmation? ✓

ID	Source	Req. Time		URL	Code	Reason	RTT	Size	Highest Alert	Note	Tags
51	Proxy	26/09/2023, 17:									Form, Password, Scri...
53	Proxy	26/09/2023, 17:									
57	Proxy	26/09/2023, 17:									
58	Proxy	26/09/2023, 17:									
59	Proxy	26/09/2023, 17:									
60	Proxy	26/09/2023, 17:									Upload, Comment
75	Proxy	26/09/2023, 17:									Form, Password, Scri...
76	Proxy	26/09/2023, 17:			304	NOT MODIFIED		0 bytes			
77	Proxy	26/09/2023, 17:17:24	POST	http://127.0.0.1:5000/	302	FOUND	10 ms	199 bytes	Low		SetCookie
78	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/index	200	OK	0 ms	5,087 bytes			
79	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/static/css/general.css/	304	NOT MODIFIED	9 ms	0 bytes	Low		
80	Proxy	26/09/2023, 17:17:42	POST	http://127.0.0.1:5000/index	200	OK	4 ms	5,100 bytes	Medium		Form, Password, Scri...
81	Proxy	26/09/2023, 17:17:42	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	5 ms	60 bytes			
82	Proxy	26/09/2023, 17:17:51	POST	http://127.0.0.1:5000/index	200	OK	7 ms	5,089 bytes	Medium		Form, Password, Scri...
83	Proxy	26/09/2023, 17:17:51	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	3 ms	60 bytes			

Alerts 0 4 3 6 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

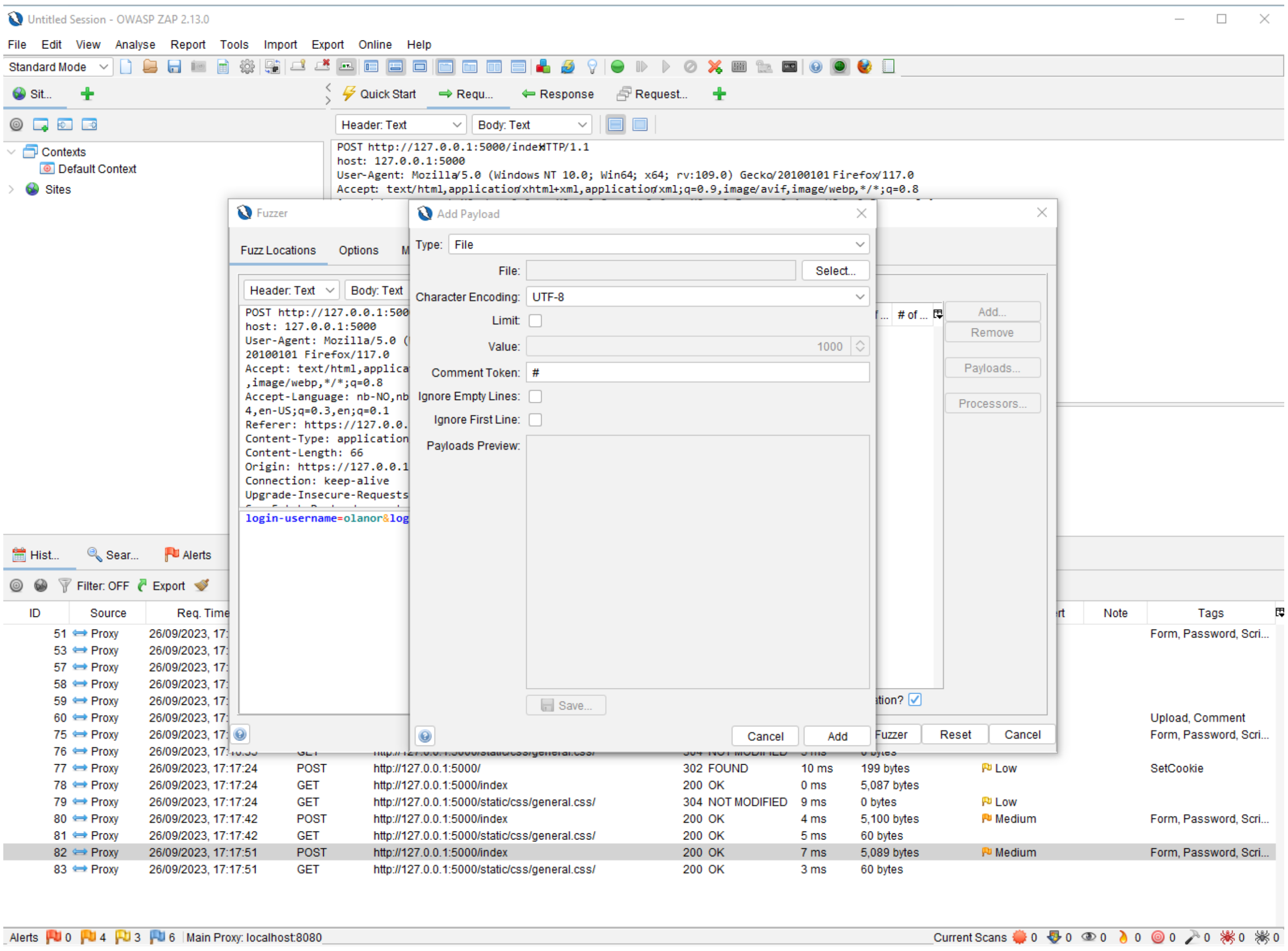


Location: Body [37, 45]
Value: changeme
Payloads:

#	Type	Description	# of Processors

Remove without confirmation? ✓

```
POST http://127.0.0.1:5000/index HTTP/1.1
host: 127.0.0.1:5000
login-username=olanor&login-
```



Header: Text Body: Text

```
POST http://127.0.0.1:5000/indexHTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
```

```
POST http://127.0.0.1:500
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (
20100101 Firefox/117.0
Accept: text/html,applica
,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb
4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.
Content-Type: application
Content-Length: 66
Origin: https://127.0.0.1
Connection: keep-alive
Upgrade-Insecure-Requests

login-username=olanor&log
```

Add Payload

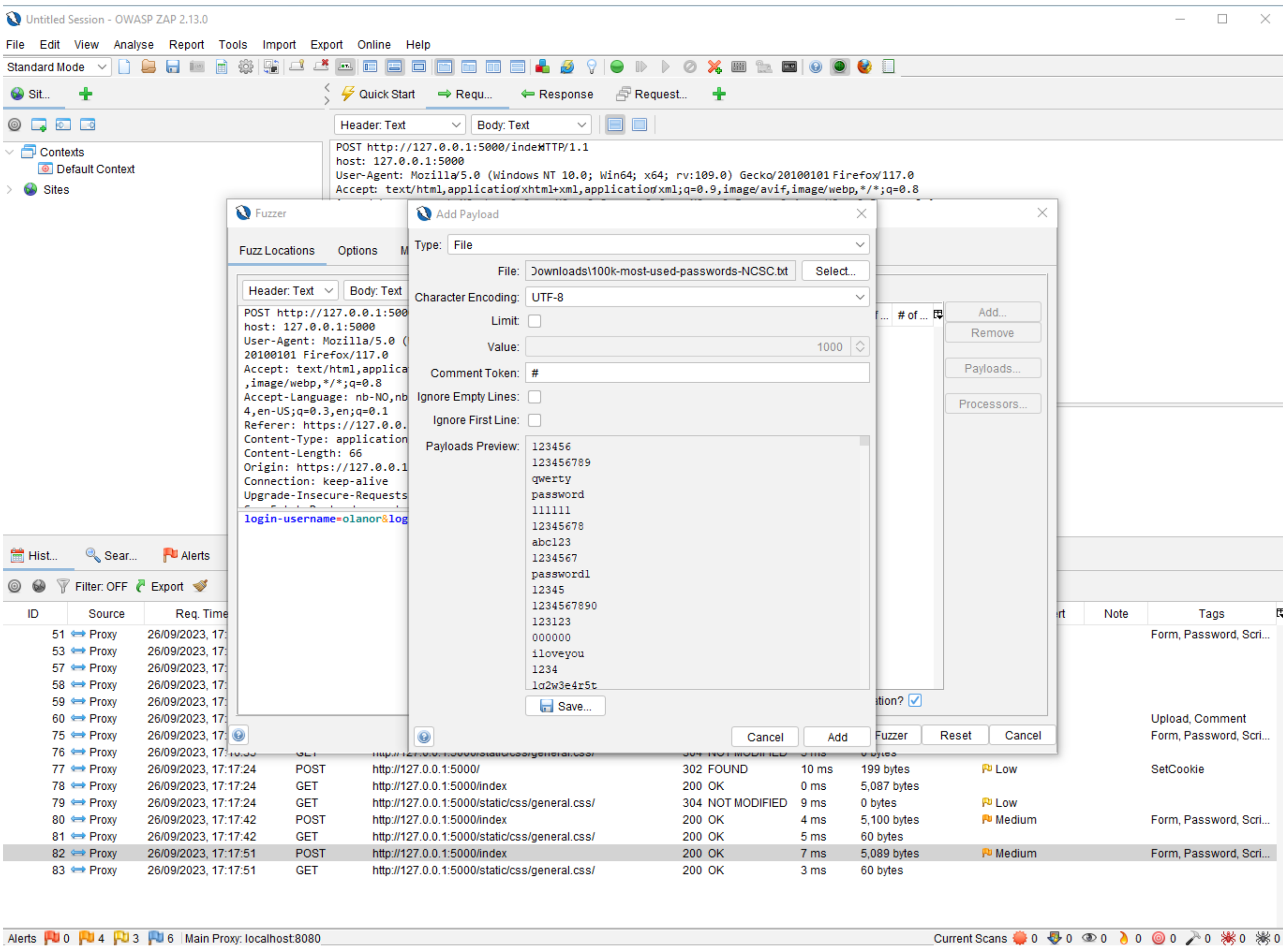
Type:	File
File:	Select...
Character Encoding:	UTF-8
Limit:	☐
Value:	1000
Comment Token:	#
Ignore Empty Lines:	☐
Ignore First Line:	☐
Payloads Preview:	

ation? ✓

ID	Source	Req. Time							Note	Tags
51	Proxy	26/09/2023, 17:								Form, Password, Scri...
53	Proxy	26/09/2023, 17:								
57	Proxy	26/09/2023, 17:								
58	Proxy	26/09/2023, 17:								
59	Proxy	26/09/2023, 17:								
60	Proxy	26/09/2023, 17:								Upload, Comment
75	Proxy	26/09/2023, 17:								Form, Password, Scri...
76	Proxy	26/09/2023, 17:								
77	Proxy	26/09/2023, 17:17:24	POST	http://127.0.0.1:5000/	302 FOUND	10 ms	199 bytes	Low		SetCookie
78	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/index	200 OK	0 ms	5,087 bytes			
79	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/static/css/general.css/	304 NOT MODIFIED	9 ms	0 bytes	Low		
80	Proxy	26/09/2023, 17:17:42	POST	http://127.0.0.1:5000/index	200 OK	4 ms	5,100 bytes	Medium		Form, Password, Scri...
81	Proxy	26/09/2023, 17:17:42	GET	http://127.0.0.1:5000/static/css/general.css/	200 OK	5 ms	60 bytes			
82	Proxy	26/09/2023, 17:17:51	POST	http://127.0.0.1:5000/index	200 OK	7 ms	5,089 bytes	Medium		Form, Password, Scri...
83	Proxy	26/09/2023, 17:17:51	GET	http://127.0.0.1:5000/static/css/general.css/	200 OK	3 ms	60 bytes			

Alerts 0 4 3 6 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

After selecting as type I pressed the select file which open a file explorer. Then i located the password wordlist I had downloaded earlier and uploaded that file to OWASP Zap.


Untitled Session - OWASP ZAP 2.13.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Quick Start Requ... Response Request...
Contexts
Default Context
Sites
Fuzzer
Fuzz Locations Options
Add Payload
Add... Remove Payloads... Processors...
Hist... Sear... Alerts
Filter: OFF Export
Start Fuzzer Reset Cancel

```
POST http://127.0.0.1:5000/indexHTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
```

```
POST http://127.0.0.1:500
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (
20100101 Firefox/117.0
Accept: text/html,applica
,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb
4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.
Content-Type: application
Content-Length: 66
Origin: https://127.0.0.1
Connection: keep-alive
Upgrade-Insecure-Requests

login-username=olanor&log
```

Type:	File
File:	Downloads\100k-most-used-passwords-NCSC.txt
Character Encoding:	UTF-8
Limit:	☐
Value:	1000
Comment Token:	#
Ignore Empty Lines:	☐
Ignore First Line:	☐

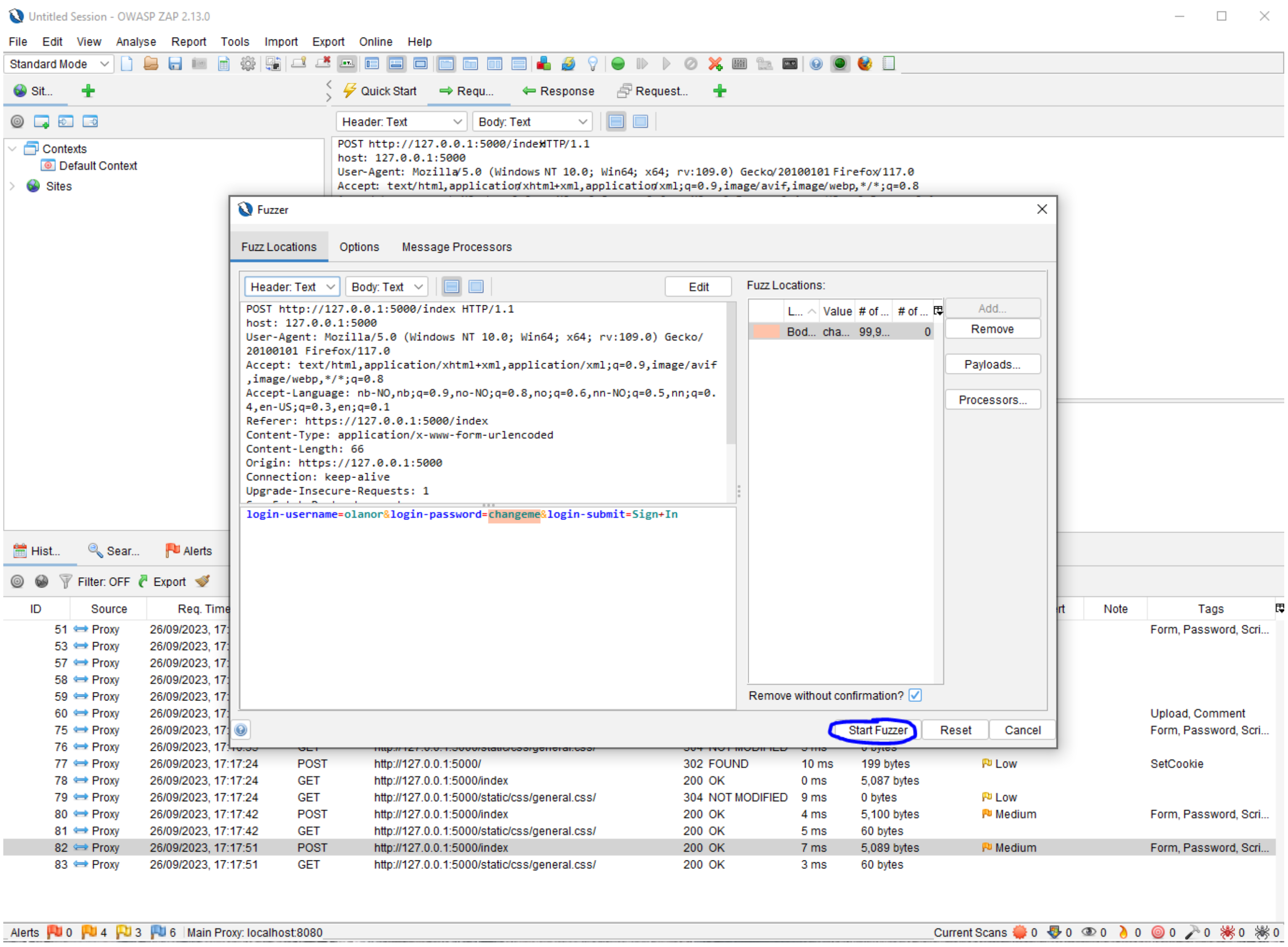
Payloads Preview:

```
123456
123456789
qwerty
password
111111
12345678
abc123
1234567
password1
12345
1234567890
123123
000000
iloveyou
1234
1q2w3e4r5t
```

ID	Source	Req. Time	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
51	Proxy	26/09/2023, 17:									Form, Password, Scri...
53	Proxy	26/09/2023, 17:									
57	Proxy	26/09/2023, 17:									
58	Proxy	26/09/2023, 17:									
59	Proxy	26/09/2023, 17:									
60	Proxy	26/09/2023, 17:									Upload, Comment
75	Proxy	26/09/2023, 17:									Form, Password, Scri...
76	Proxy	26/09/2023, 17:	GET		304	NOT MODIFIED		0 bytes			
77	Proxy	26/09/2023, 17:17:24	POST	http://127.0.0.1:5000/	302	FOUND	10 ms	199 bytes	Low		SetCookie
78	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/index	200	OK	0 ms	5,087 bytes			
79	Proxy	26/09/2023, 17:17:24	GET	http://127.0.0.1:5000/static/css/general.css/	304	NOT MODIFIED	9 ms	0 bytes	Low		
80	Proxy	26/09/2023, 17:17:42	POST	http://127.0.0.1:5000/index	200	OK	4 ms	5,100 bytes	Medium		Form, Password, Scri...
81	Proxy	26/09/2023, 17:17:42	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	5 ms	60 bytes			
82	Proxy	26/09/2023, 17:17:51	POST	http://127.0.0.1:5000/index	200	OK	7 ms	5,089 bytes	Medium		Form, Password, Scri...
83	Proxy	26/09/2023, 17:17:51	GET	http://127.0.0.1:5000/static/css/general.css/	200	OK	3 ms	60 bytes			

Alerts 0 4 3 6 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

After uploading the file i pressed "Start fuzzer". OWASP Zap then starts to test all the different passwords listed in the uploaded file.



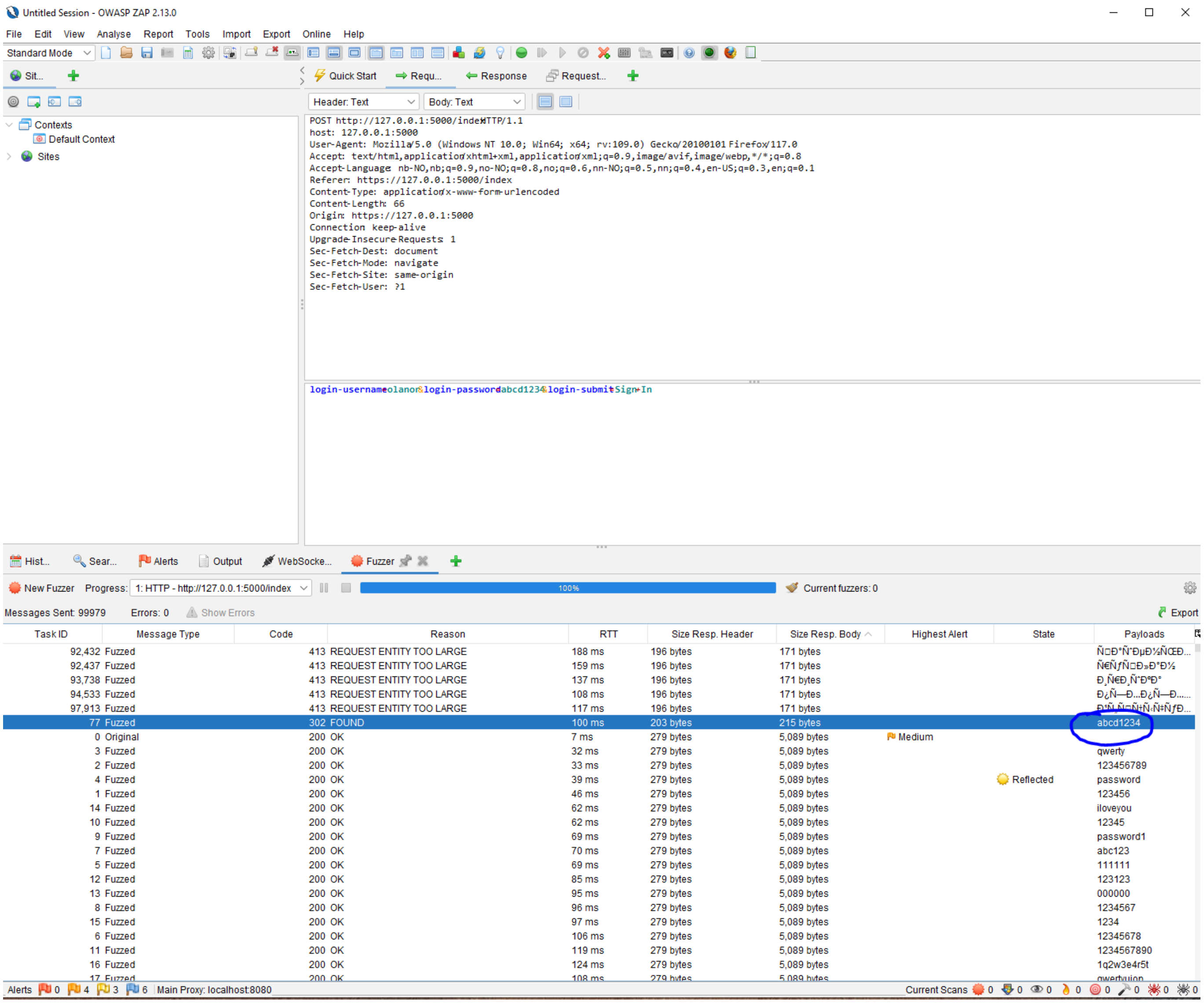
```
POST http://127.0.0.1:5000/index HTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/
20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.
4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.1:5000/index
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: https://127.0.0.1:5000
Connection: keep-alive
Upgrade-Insecure-Requests: 1

login-username=olanor&login-password=changeme&login-submit=Sign+In
```

```
77 Proxy 26/09/2023, 17:17:24 POST http://127.0.0.1:5000/                     302 FOUND        10 ms 199 bytes   Low    SetCookie
78 Proxy 26/09/2023, 17:17:24 GET  http://127.0.0.1:5000/index                200 OK           0 ms  5,087 bytes
79 Proxy 26/09/2023, 17:17:24 GET  http://127.0.0.1:5000/static/css/general.css/ 304 NOT MODIFIED 9 ms 0 bytes   Low
80 Proxy 26/09/2023, 17:17:42 POST http://127.0.0.1:5000/index                200 OK           4 ms  5,100 bytes Medium Form, Password, Scri...
81 Proxy 26/09/2023, 17:17:42 GET  http://127.0.0.1:5000/static/css/general.css/ 200 OK        5 ms  60 bytes
82 Proxy 26/09/2023, 17:17:51 POST http://127.0.0.1:5000/index                200 OK           7 ms  5,089 bytes Medium Form, Password, Scri...
83 Proxy 26/09/2023, 17:17:51 GET  http://127.0.0.1:5000/static/css/general.css/ 200 OK        3 ms  60 bytes
```

Main Proxy: localhost:8080

Step.5

When OWASP Zap completed the testing I sorted the tests based on body size and found that there was one test that stod out. I then looked at the value in the payloads column and could see that the one that stod out was the correct password for that username.



```
POST http://127.0.0.1:5000/indexHTTP/1.1
host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.4,en-US;q=0.3,en;q=0.1
Referer: https://127.0.0.1:5000/index
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: https://127.0.0.1:5000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
```

```
login-username=olanor&login-password=abcd1234&login-submit=Sign+In
```

Progress: 1: HTTP - http://127.0.0.1:5000/index 100% Current fuzzers: 0

Messages Sent: 99979 Errors: 0

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
92,432	Fuzzed	413	REQUEST ENTITY TOO LARGE	188 ms	196 bytes	171 bytes			Ñ□Ð°Ñ˜ÐµÐ½ÑŒÐ...
92,437	Fuzzed	413	REQUEST ENTITY TOO LARGE	159 ms	196 bytes	171 bytes			Ñ€ÑƒÑ□Ð»Ð°Ð½
93,738	Fuzzed	413	REQUEST ENTITY TOO LARGE	137 ms	196 bytes	171 bytes			Ð¸Ñ€Ð¸Ñ˜Ð°Ð°
94,533	Fuzzed	413	REQUEST ENTITY TOO LARGE	108 ms	196 bytes	171 bytes			Ð¿Ñ—Ð…Ð¿Ñ—Ð......
97,913	Fuzzed	413	REQUEST ENTITY TOO LARGE	117 ms	196 bytes	171 bytes			Ð¹Ñ¸Ñ□Ñ†Ñ‹Ñ‡ÑƒÐ...
77	Fuzzed	302	FOUND	100 ms	203 bytes	215 bytes			abcd1234
0	Original	200	OK	7 ms	279 bytes	5,089 bytes	Medium		
3	Fuzzed	200	OK	32 ms	279 bytes	5,089 bytes			qwerty
2	Fuzzed	200	OK	33 ms	279 bytes	5,089 bytes			123456789
4	Fuzzed	200	OK	39 ms	279 bytes	5,089 bytes		Reflected	password
1	Fuzzed	200	OK	46 ms	279 bytes	5,089 bytes			123456
14	Fuzzed	200	OK	62 ms	279 bytes	5,089 bytes			iloveyou
10	Fuzzed	200	OK	62 ms	279 bytes	5,089 bytes			12345
9	Fuzzed	200	OK	69 ms	279 bytes	5,089 bytes			password1
7	Fuzzed	200	OK	70 ms	279 bytes	5,089 bytes			abc123
5	Fuzzed	200	OK	69 ms	279 bytes	5,089 bytes			111111
12	Fuzzed	200	OK	85 ms	279 bytes	5,089 bytes			123123
13	Fuzzed	200	OK	95 ms	279 bytes	5,089 bytes			000000
8	Fuzzed	200	OK	96 ms	279 bytes	5,089 bytes			1234567
15	Fuzzed	200	OK	97 ms	279 bytes	5,089 bytes			1234
6	Fuzzed	200	OK	106 ms	279 bytes	5,089 bytes			12345678
11	Fuzzed	200	OK	119 ms	279 bytes	5,089 bytes			1234567890
16	Fuzzed	200	OK	124 ms	279 bytes	5,089 bytes			1q2w3e4r5t
17	Fuzzed	200	OK	108 ms	279 bytes	5,089 bytes			qwertyuiop

Alerts 0 4 3 6 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

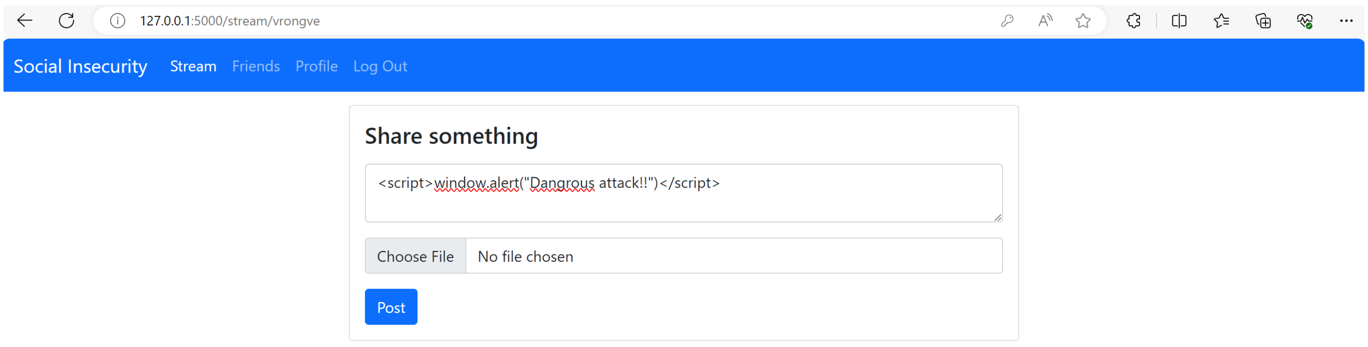
Result

Because the social insecurity application stores passwords in plaintext without any password requirements, and it confirms the validity of a username, it is highly vulnerable to brute-force attacks. So by using a brut force technique i was able to defeat the authenticaion mechanism of the application.

3. Cross site scripting #1 - Input form

Step.1

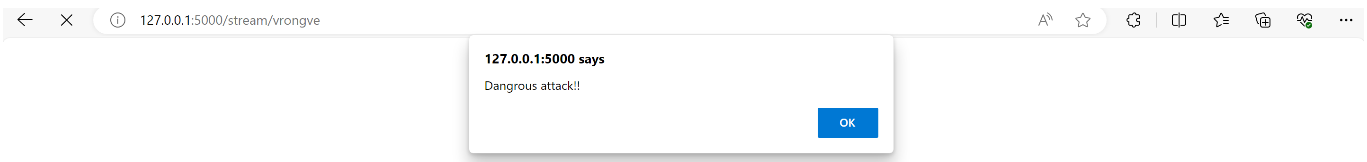
I entered into the post comment field on the stream page and submitted the commnet.



```
<script>window.alert("Dangrous attack!!")</script>
```

Step.2

After clicking the 'Post' button, the page refreshes, and an alert popup message appears. This message continues to appear each time the stream page is refreshed. Note: A similar behavior can occur on the 'Edit Profile' page due to the absence of form validation and output encoding. Instead of treating the output as text, it's interpreted as JavaScript.



127.0.0.1:5000 says

Dangrous attack!!

Result:

This exploitation shows that the application is vulnerable for cross site scripting which could lead to attackers stealing sensitive user data, such as login credentials, session cookies or personal information.

4. Cross site scripting #2 - File upload

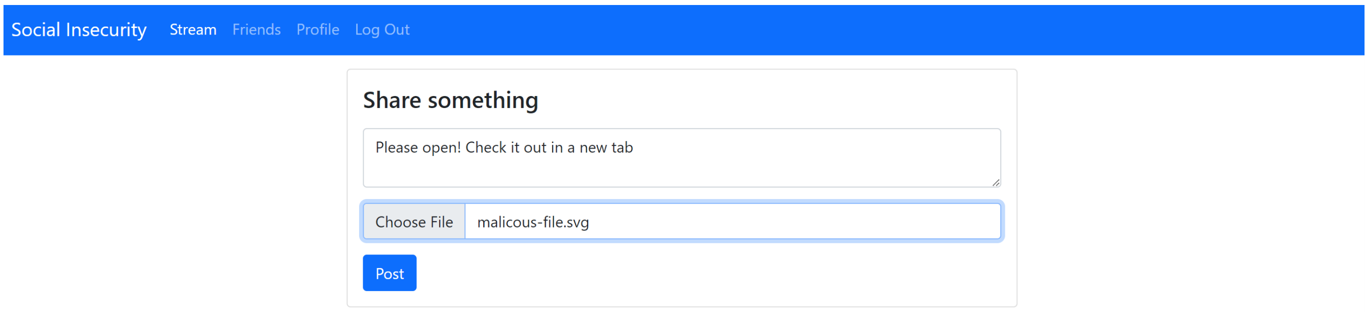
Step.1

I created an SVG file containing javascript code to simulate a Cross-Site Scripting attack scenario.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
  <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#004400"/>
  <script type="text/javascript">
    alert("XSS by BHARAT");
    </script>
</svg>
```

Step.2

I pressed chose file and then uploaded the malicous-file.svg file which contained the script above.

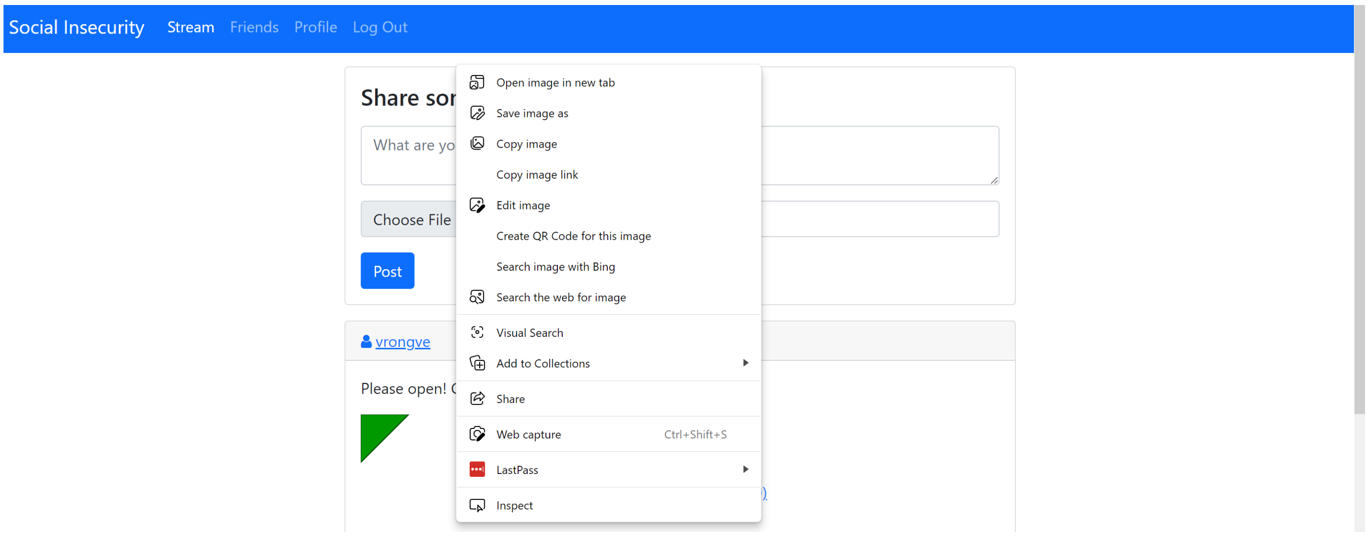


Please open! Check it out in a new tab

malicous-file.svg

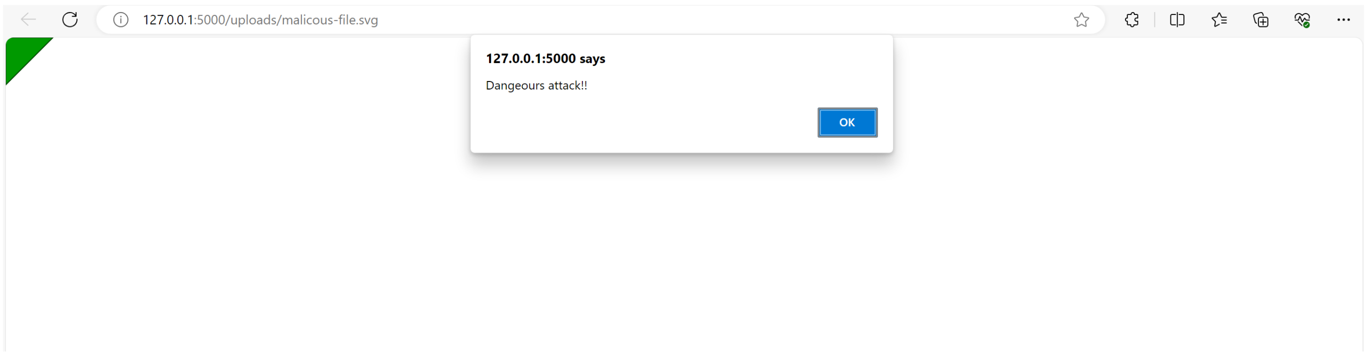
Step.3

After posting the file I could see the new post in the list of posts on the stream page. I then right clicked the svg image and pressed "open image in new tab".



Step.4

When I open a new tab, I observe that the JavaScript has executed, as indicated by the appearance of a pop-up alert message.



127.0.0.1:5000/uploads/malicous-file.svg

127.0.0.1:5000 says

Dangeours attack!!

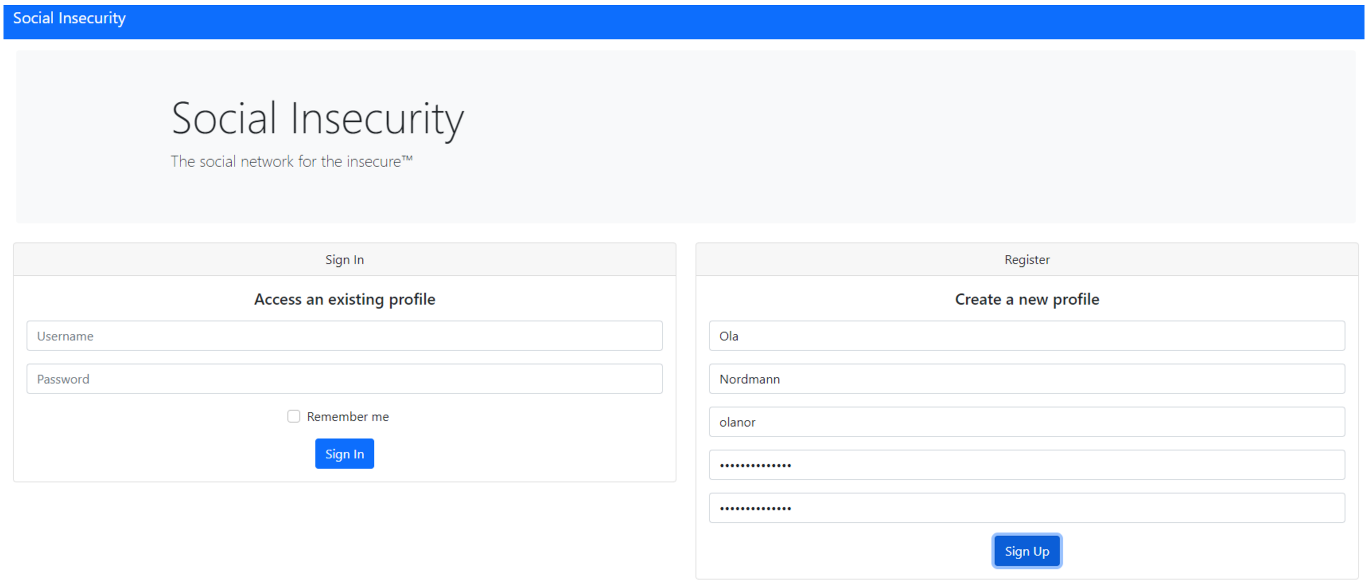
Result:

This exploitation shows that the application is vulnerable for cross site scripting which could lead to attackers stealing sensitive user data, such as login credentials, session cookies or personal information.

5. Cryptographic Failures

Step.1

I opened the social insecurity app and registered a user by entering firstname: Ola, lastname: Nordmann, username: olanor and password: huskeraldri123.

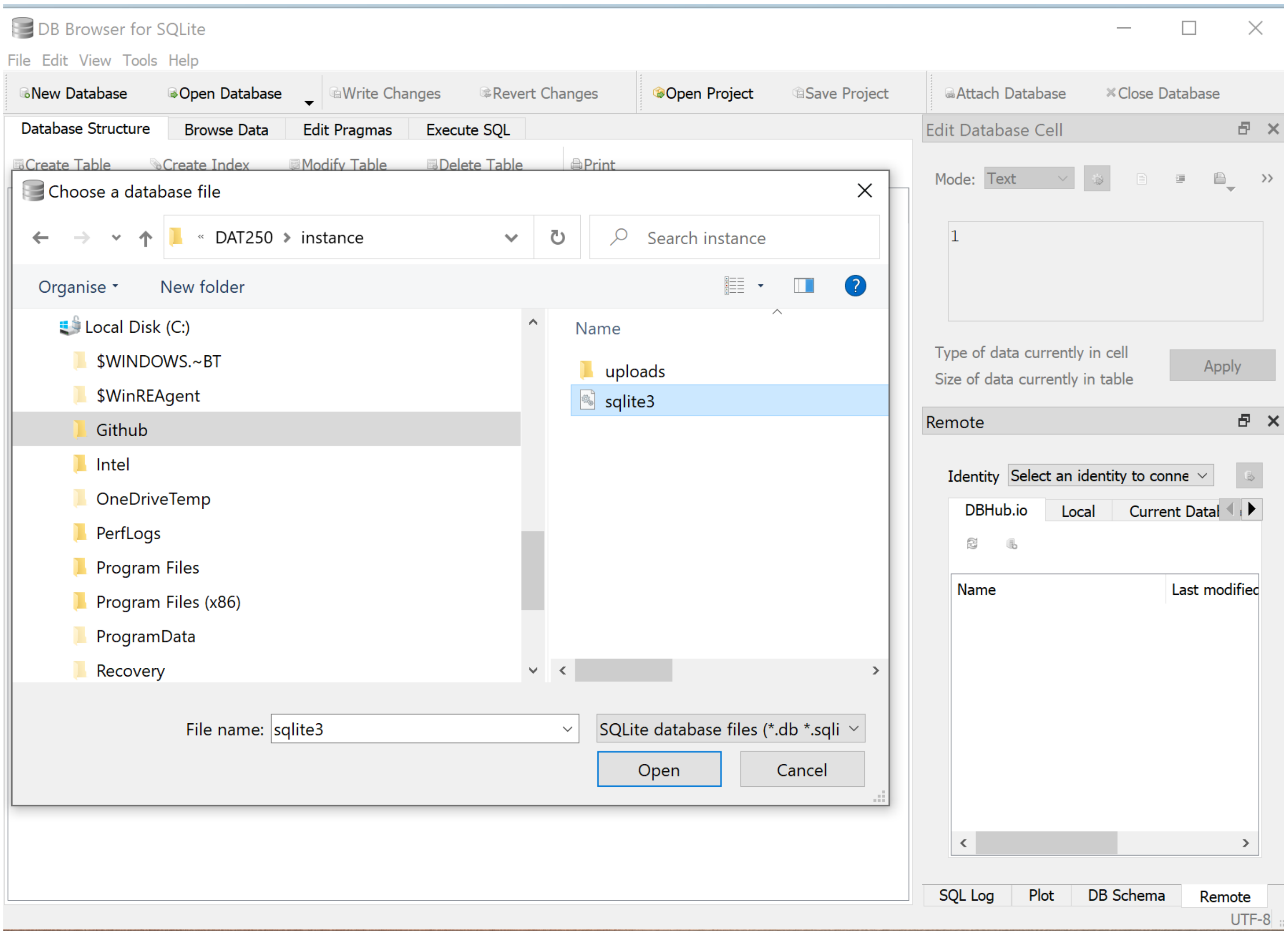


Step.2

After the user-registration I looked around in the source which I have access to and found that a sqlite file was located in the instance folder. After locating this file I went on to download DB Browser (SQLite), which is an open-source and user-friendly tool for working with SQLite databases. The software provides a graphical interface for interacting with SQLite databases, making it easier for users to create, view, edit and manage SQLite database files.

Step.3

After getting the DB Browser (SQLite) application up and running I went on to open the sqlite file located in the application with the DB Browser software.



File name: sqlite3
SQLite database files (*.db *.sqli
Mode: Text
Type of data currently in cell
Size of data currently in table
Identity: Select an identity to conne

After opening the file i went on to locate the users table and browsed the current data. By browsing the sqlite file used for storing the data in the social insecurity app I could easily see that the password stored where not hashed.

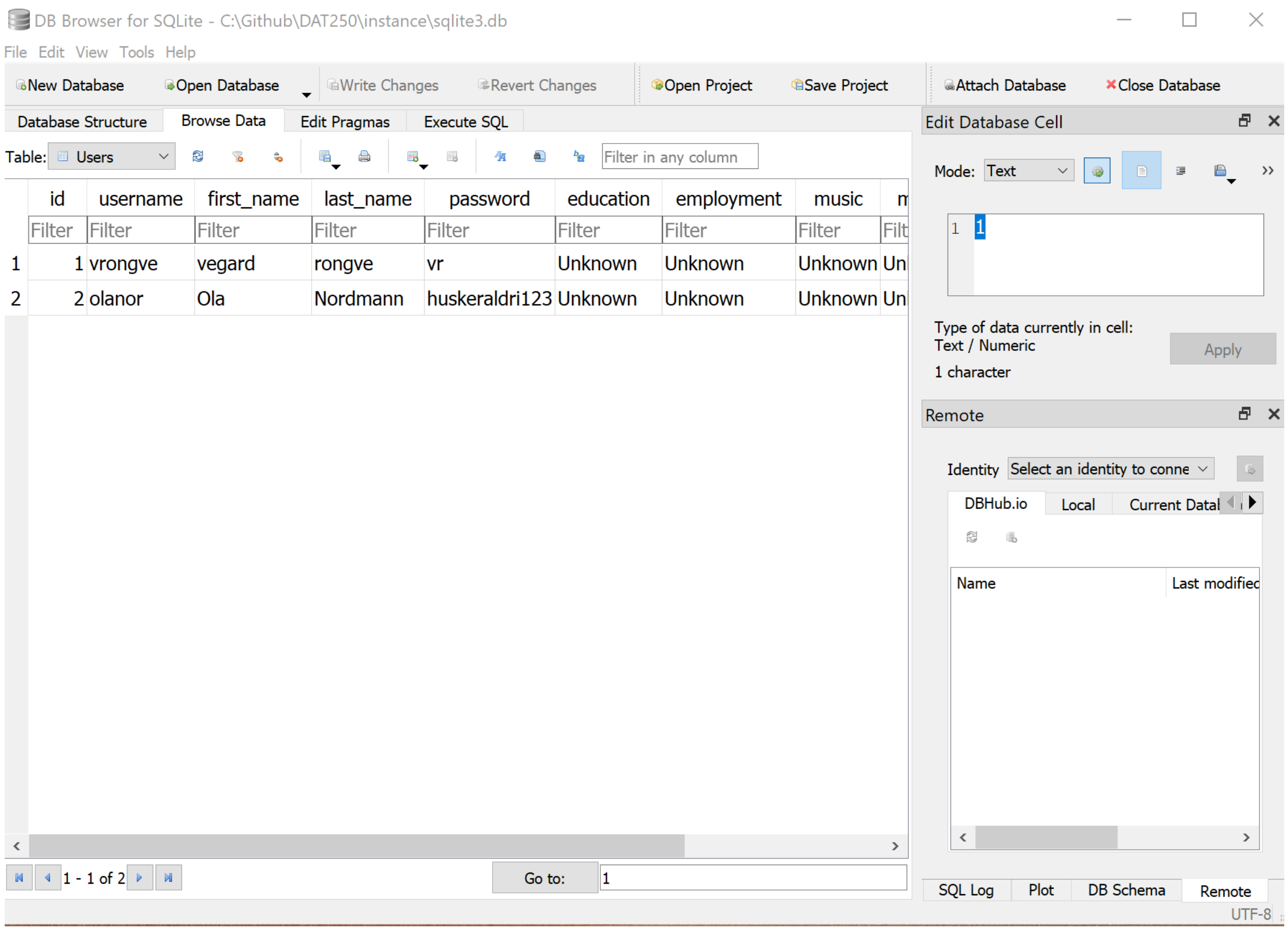

DB Browser for SQLite - C:\Github\DAT250\instance\sqlite3.db

Table: Users

	id	username	first_name	last_name	password	education	employment	music	m
1	1	vrongve	vegard	rongve	vr	Unknown	Unknown	Unknown	Un
2	2	olanor	Ola	Nordmann	huskeraldri123	Unknown	Unknown	Unknown	Un

1 - 1 of 2

Mode: Text

Type of data currently in cell: Text / Numeric

1 character

Result:

If an attacker where to get access to this sqlite file the attacker would get hold of all the usernames and passwords. The attacker wouldent even have to break the hashing as passwords was stored as plain text. An attacker would get complete control over the app and any user using the app.

Impact Analysis

Discuss the potential consequences if these vulnerabilities were present in a real-world application.

1. Cross site scripting

Potential consequence if a real world application has the same cross site scripting vulnerabiltiy as identified in the social insecurity application.

- Data tefth: Attackers can steal sensitive user data, such as login credentials, session cookies, or personal information, by injecting malicious scripts into a website.
- Phising: Attackers can create convincing phishing pages by injecting malicious scripts into legitimate websites. Users may unknowingly enter their credentials or sensitive information on these fake pages.
- Clickjacking: Attackers can overlay malicious content on a legitimate website, tricking users into clicking on elements that perform unintended actions without their knowledge.
- Data manipulation: XSS can be used to modify data within a web application, potentially leading to data corruption or unauthorized changes.

2. Broken access control

Potential consequence if a real world application has the same broke access control vulnerabiltiy as identified in the social insecurity application.

- Unauthorized data access: Attackers can access sensitive information and data belonging to other users, violating their privacy. This information may include personal details, contact information, or confidential data.
- Data modification: Attackers can maliciously modify or delete data within another user's profile. This could lead to data corruption, loss of information, or unauthorized changes to user settings.
- Reputation demage: Users affected by this vulnerability may lose trust in the web application and the organization running it, leading to reputational damage and a loss of customers or users.
- Finacial loss: Security breaches, especially those leading to data theft or fraudulent activities, can result in financial losses for both the affected users and the organization responsible for the application

3. Cryptograpic Failures

Potential consequence if a real world application has cryptograpic failures.

- Data exposure: Attackers may gain access to encrypted data and, through cryptographic weaknesses, decrypt it to reveal sensitive information. This could include personally identifiable information (PII), financial data, or confidential business data.
- Access to passwords and usernames: If a real world application did not hash the passwords stored, an attacker could potentially read all usernames and passwords if the attacker where to get hold of the file storing the data.

4. Idetification and authentication

Identification and authentication vulnerabilities in a web application can have significant and wide-ranging consequences, especially if they lead to weak passwords and make brute-forcing attacks easier.

- Unauthorized access: Weak authentication mechanisms can allow attackers to gain unauthorized access to user accounts, systems, or sensitive data within the application.
- Data breach: Attackers who successfully exploit identification and authentication vulnerabilities can access and potentially steal sensitive user data, such as personal information, financial data, or confidential documents.

Lessons Learned

In the course of this project, several valuable insights and lessons were gained, shedding light on both the challenges faced and the effective tools and techniques employed in the process of securing web applications.

Tool Selection:

The primary tools utilized for this project were web browsers and OWASP Zap. While web browsers provided a practical means to explore the application from an attacker's perspective, OWASP Zap offered an automated approach to vulnerability detection and exploitation. The combination of these tools allowed for a comprehensive assessment of the web application's security posture.

Attacker's Perspective:

One of the most effective techniques employed was approaching the application with an attacker's mindset. This approach proved to be invaluable in uncovering vulnerabilities that might otherwise go unnoticed.

Challenges with OWASP Zap:

The project also highlighted certain challenges, particularly in regards to using OWASP Zap. These challenges were primarily rooted in the lack of prior exposure and training. Unfortunately, due to unavailability and the absence of recorded lectures, acquiring proficiency in OWASP Zap was more challenging than anticipated.

Knowledge Acquisition:

Despite the challenges, the project provided a significant learning opportunity. A deeper understanding of the various ways in which web applications can be exploited was acquired. This newfound knowledge is valuable and will be leveraged in future web application developments.

In conclusion, this project served as a valuable learning experience, highlighting the significance of adopting an attacker's perspective, the challenges associated with security tools, and the critical need for ongoing education and awareness in the field of web application security.