Social Insecurity - Penetration Testing Report

run shift+ctrl+v to visualize the markdown in separate window

export to PDF

Introduction

Tester: Vegard Rongve

Application: Social Insescurity

Test period: 01.09.2023 - 30.09.2023

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 (given that the attacker knows the username).

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 excepts 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.
- Unrestricted file uploads can result in significant storage costs, especially if users upload large files or upload files at a high volume. This can strain your server's resources and budget.
- Attackers can conduct attacks such as cross-site scripting by uploading malicous scripts and tricking other users into executing them. By doing so, the attacker could trick the user to give them their password and username.

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 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.

Form validation

Category:

- A04:Insecure design

Description:

- The app lacks form validation. For example, a user might share something without actually writing or uploading anything on the stream page. A user can also be registered without filling out all the input fields on the registration form.

Potential impact:

- The potenial impact of bad form validation in these cases are not very big, but the data quality in the web application can become very poor, which might lead to frustrated users and reputation demage.

Affected part of the application:

- User registration form on the login in page and the stream page where posts are posted.

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

Weak password. Brut force?

3. Cross site scripting #1 - Input form

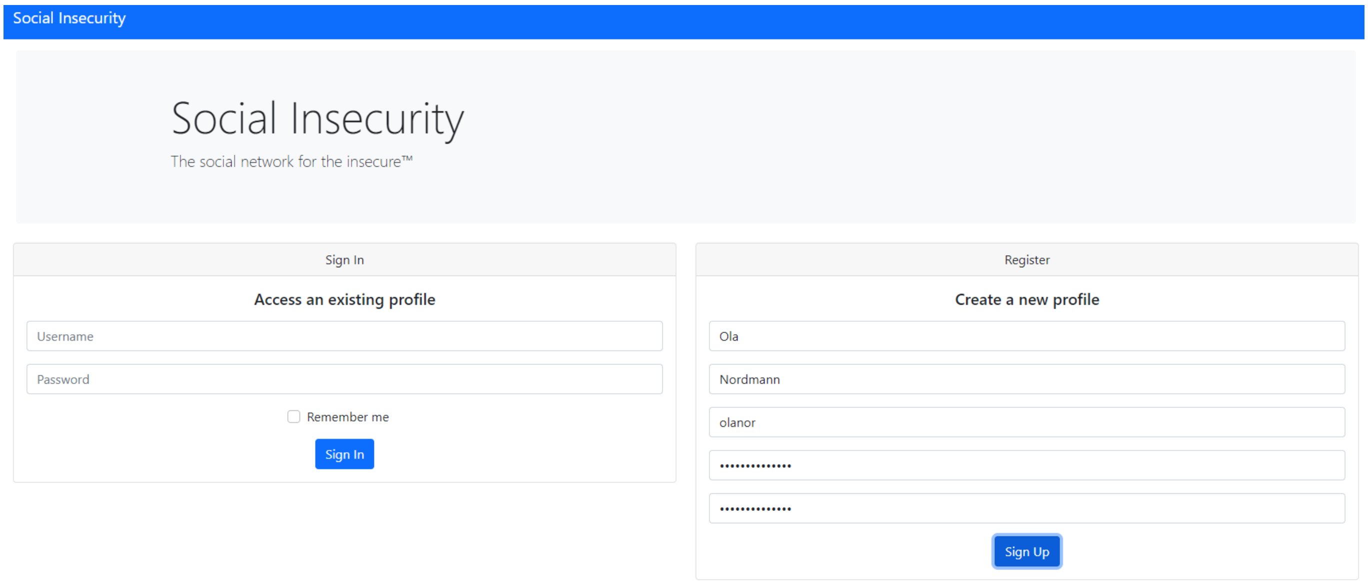
I entered into the post comment field an submitted it. This is then saved in the server. And when the list of comments is diplayed, which is done everytime the page loads, the script injected is runned.

4. Cross site scripting #2 - File upload

5. Cryptographic Failures - Password storing

Step.1

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



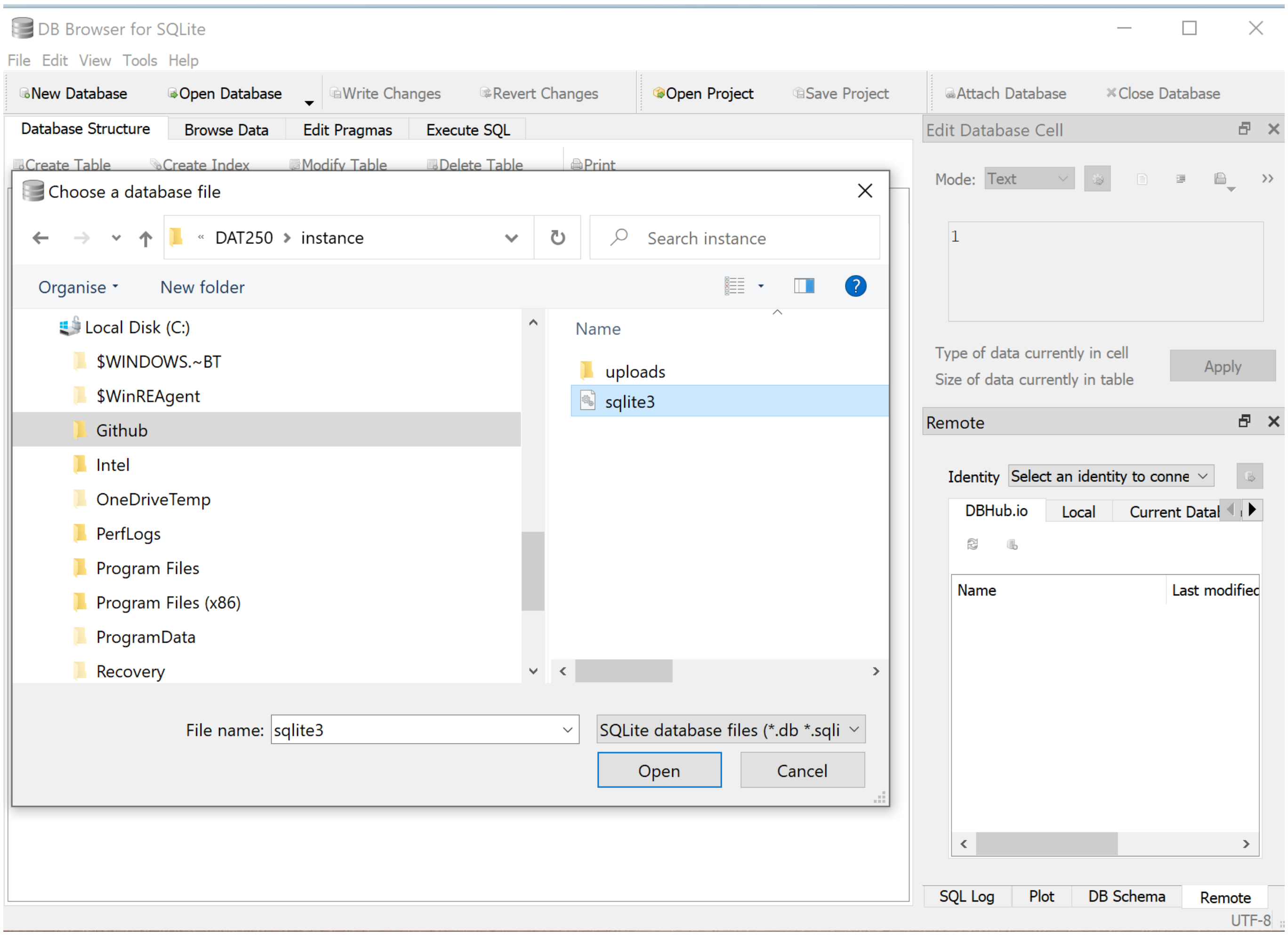
Ola
Nordmann
olanor
••••••••••••••
••••••••••••••

Step.2

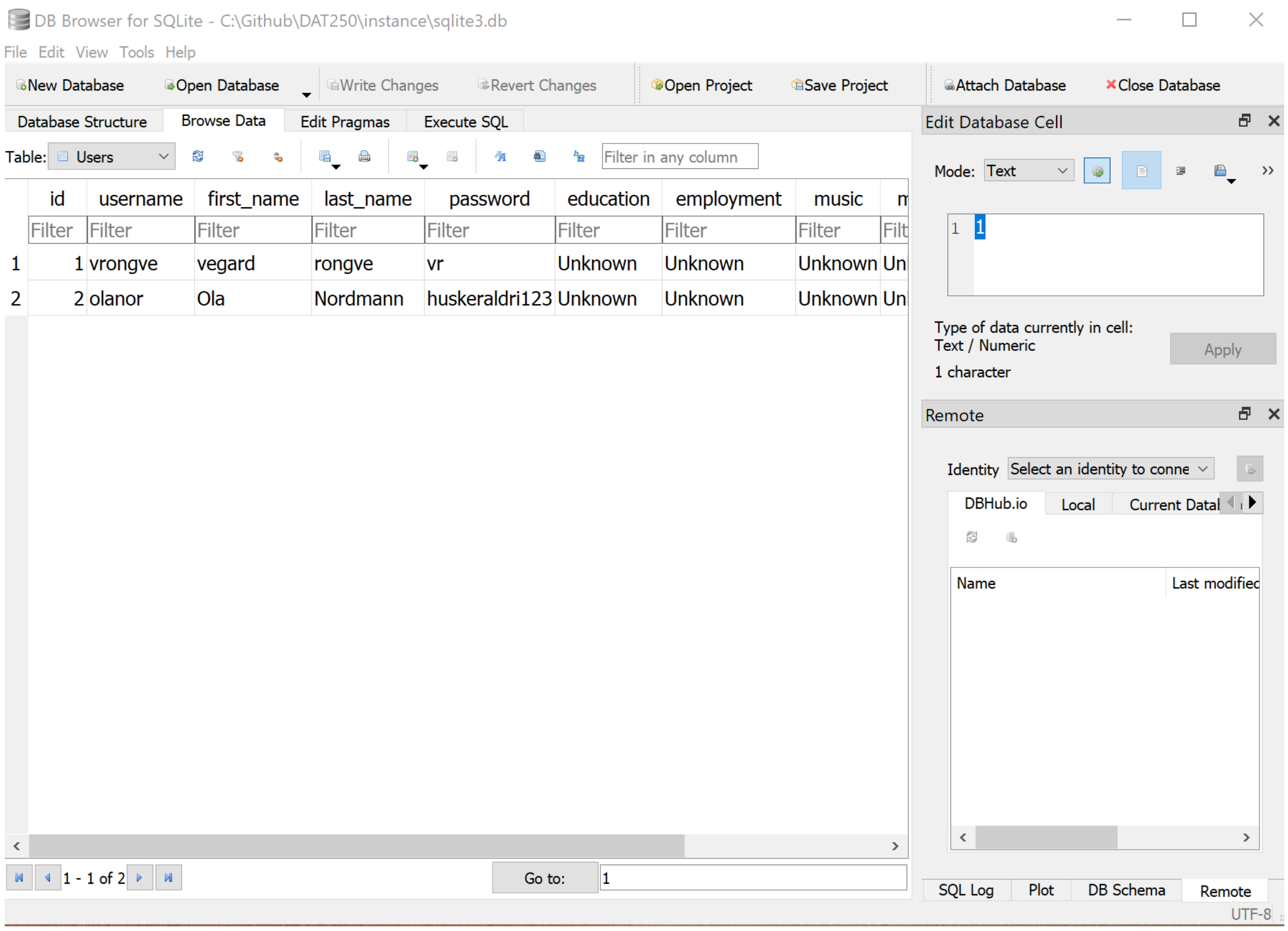
After the user-registration I looked around in the source which I have access to and found that a sqlite file was located in the instance folder. After locating this file I went on to download DB Browser (SQLite), which is an open-source and user-friendly tool for working with SQLite databases. The software 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.



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.



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

1 - 1 of 2

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.

Vulnerability Exploitation

1. Select at least five distinct vulnerabilities that you have identified.
2. Choose appropriate penetration testing tools (such as Metasploit, Burp Suite, OWASP Zap, sqlmap) and techniques to exploit these vulnerabilities.
3. Clearly document the steps you took to exploit each vulnerability, including any commands, payloads, or configurations used. These instructions should be detailed enough for the teaching assistants or the lecturers to reproduce your attack.
4. Explain the results of each successful exploitation, including the extent of the compromise, potential data accessed, and system control achieved.

Impact Analysis

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

Lessons Learned

Reflect on the challenges you encountered, the tools and techniques you found most effective, and the importance of securing web applications.