CS6903 - Network security

Project-1A: TransactiWar – Phase-2

Introduction

Functionality:

- The user is directed to the website's login page, where they have the option to either log in if they already have an account or sign up using their username, email, and password.
- After successful login, the user is redirected to the home page, which contains:
    - User name
    - Current balance
    - Transaction history
        - Contains all the transactions with the given details: Date, Sender/Receiver Username, Remark, and Amount.
    - Navigation bar
- The user can use the Navigation bar to navigate to the different pages by clicking on the links:
    - Home
    - Search
    - Profile
    - Transfer
    - Logout button
- The user can use the search page to search for other users
    - It displays all the existing usernames and a button that links to their profile page
- The profile page displays the following contents:
    - Username: a fixed value
    - Email: can be edited by the user
    - Bio: can be edited
        - By default, it is empty
    - Profile picture: can upload a picture
        - A default pic is uploaded for every new user

    - Profile password can be changed.
        - To change the password, the user needs to enter their existing password for validation and the new password.
- The user can transfer money from the transfer page, which requires the following input:
    - Receiver username
    - Amount to transfer
    - Remark

Security Mechanisms

- Input validation and sanitization
    - All the inputs received are validated using the respective validation rule, like usernames must be alphanumeric and can contain a ‘_’, etc.
    - To prevent XSS attacks, all inputs displayed on the webpages are sanitized using the htmlspecialchars function to ensure that all HTML characters are converted to special entities.
- CSRF prevention
    - All the HTTP forms on the website use CSRF tokens.
    - For each request sent to the server, a new CSRF token is created. This token is stored in the PHP session storage.
    - The CSRF token is included in the HTTP form and sent to the user
    - For each POST request that the server receives, the CSRF token is validated before performing any changes to the user data
- Authentication and Session Management
    - The website uses password-based authentication
    - Cookies are used for session management
    - Before any user data is changed, the request sent to the server is authenticated.
    - To make authentication easier, all requests are sent to the index.php file using the Apache mod_rewrite module.
    - When the user signs in using a valid user name and password, a session cookie is sent to the user

    - For each request sent by the user, the session cookie is validated
    - To prevent session fixation, we enabled secure sessions in the php.ini file
- TLS
    - Since HTTP traffic is not encrypted, we use TLS for end-to-end encryption, authentication, and message integrity.
    - TLS also prevents replay attacks
- Validating Image Uploads
    - To prevent LFI attacks, the file uploaded by the user will be validated before storing it on the server's file system.
    - To check if the file is a valid PNG, an image is created using the file using the GD PHP library.
    - The file path is also checked to ensure that only the base file name is used.
    - The uploaded files are stored(/var/www/uploads) in a different directory from the server code(stored in /var/www/html). This also prevents the server from executing the file.
    - The max size of the file is set in the php.ini file to prevent overloading the server with large files.
- Cookie protection
    - The cookies are always sent only on an HTTPS connection
    - The cookies cannot be accessed by JavaScript (HTTP only cookie)
    - Other sites cannot access the cookies(SameSite: Strict)
- File system access control
    - We used the Apache HTTPD server as a proxy
    - We configured the server such that only the /var/www/html and /var/www/uploads directories are accessible by the PHP processes.
    - This will prevent users from requesting other files that are part of the server (for example, server certificates or /etc/passwd)
- Using SQL transactions
    - To perform credit transfer, SQL transactions are used.

  - Before reflecting the balance change, both the sender and receiver rows are locked.
  - Once the rows are locked, a new transaction history row is added
  - The balance of the sender and receiver is updated
  - If there’s an error in any of the above SQL statements, the changes will be rolled back.
  - Additionally, there are constraints in the SQL tables that prevent a negative balance

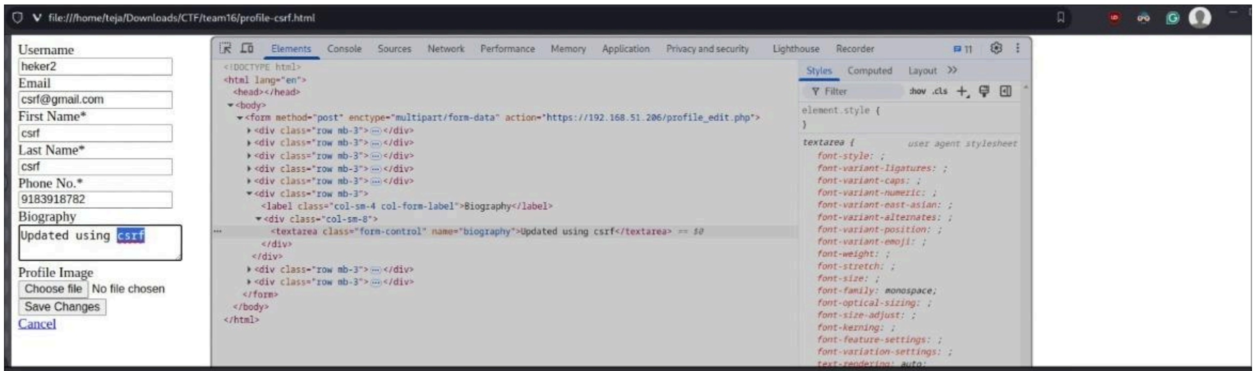
Website Vulnerabilities

1. MITM attacks

Man-in-the-middle attacks are prevented since messages are encrypted over the link(Using TLS). Proxies cannot decrypt TLS traffic.

War game attack analysis

- CSRF attack on Team-16:
  - We attacked Team-16’s profile page by CSRF, and we could update the user's profile info, such as first name, last name, phone number, and bio.
  - A user is already logged in to a browser.
  - We created an HTML file, which sends a POST request to update the profile page of a user.
  - We ran the HTML on a browser in a different tab, mimicking a CSRF attack.
  - Since a browser shares the same cookies across multiple tabs, the POST request is sent using the same cookie value as the user session and thus updates their profile.
  - An image is shared for your reference.

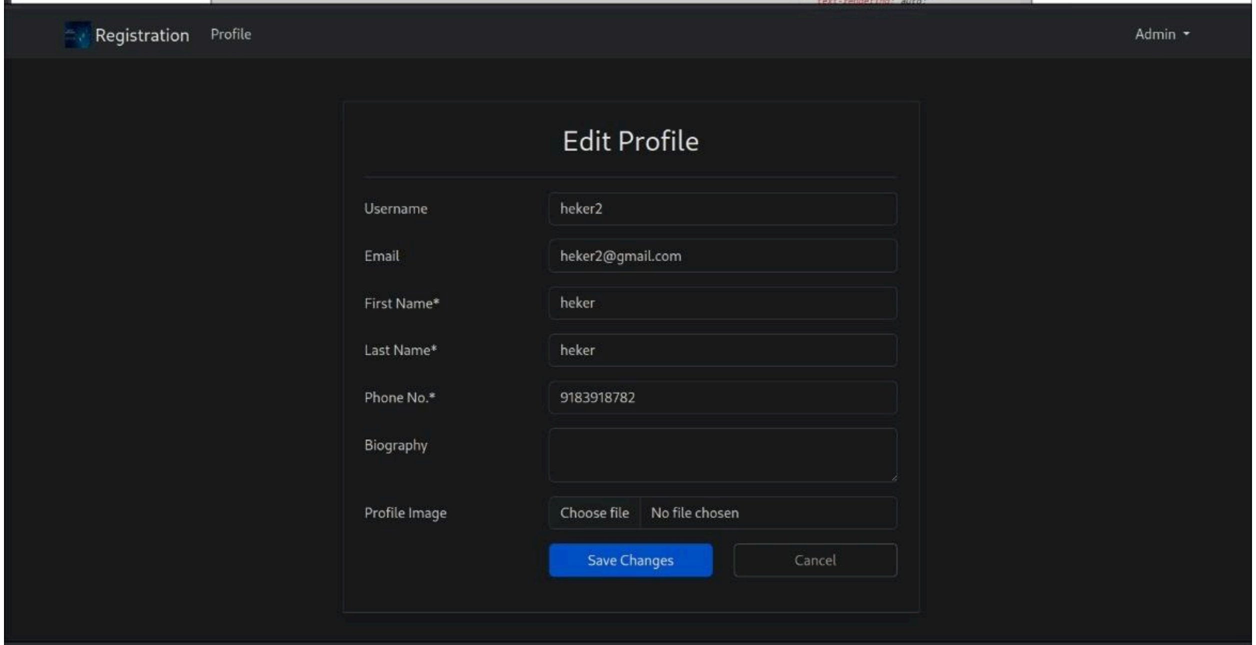


file:///home/teja/Downloads/CTF/team16/profile-csrf.html

Username	heker2
Email	csrf@gmail.com
First Name*	csrf
Last Name*	csrf
Phone No.*	9183918782
Biography	Updated using csrf
Profile Image	Choose file No file chosen

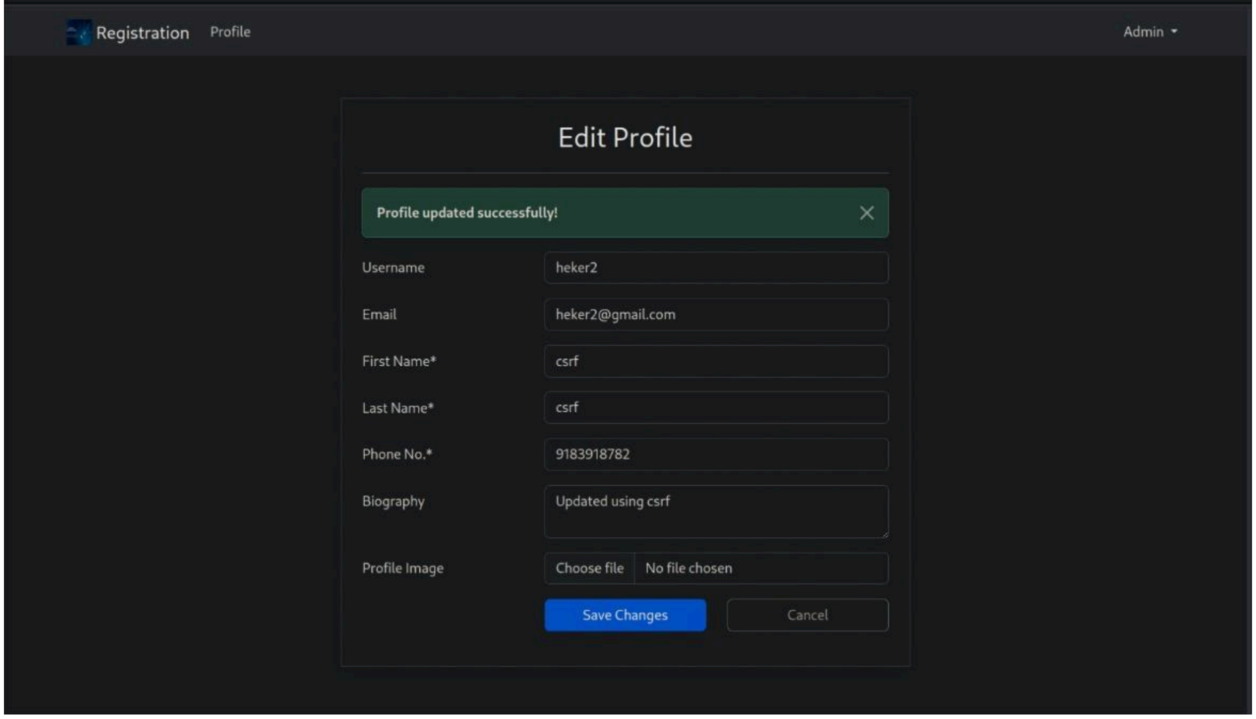
```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <form method="post" enctype="multipart/form-data" action="https://192.168.51.206/profile_edit.php">
      <div class="row mb-3"> ⋯ </div>
      <div class="row mb-3"> ⋯ </div>
      <div class="row mb-3"> ⋯ </div>
      <div class="row mb-3"> ⋯ </div>
      <div class="row mb-3"> ⋯ </div>
      <div class="row mb-3">
        <label class="col-sm-4 col-form-label">Biography</label>
        <div class="col-sm-8">
          <textarea class="form-control" name="biography">Updated using csrf</textarea> == $0
        </div>
      </div>
      <div class="row mb-3"> ⋯ </div>
      <div class="row mb-3"> ⋯ </div>
    </form>
  </body>
</html>
```

```
element.style {
}
textarea {                      user agent stylesheet
  font-style: ;
  font-variant-ligatures: ;
  font-variant-caps: ;
  font-variant-numeric: ;
  font-variant-east-asian: ;
  font-variant-alternates: ;
  font-variant-position: ;
  font-variant-emoji: ;
  font-weight: ;
  font-stretch: ;
  font-size: ;
  font-family: monospace;
  font-optical-sizing: ;
  font-size-adjust: ;
  font-kerning: ;
  font-feature-settings: ;
  font-variation-settings: ;
  text-rendering: auto;
```



Edit Profile

Username	heker2
Email	heker2@gmail.com
First Name*	heker
Last Name*	heker
Phone No.*	9183918782
Biography	
Profile Image	Choose file No file chosen

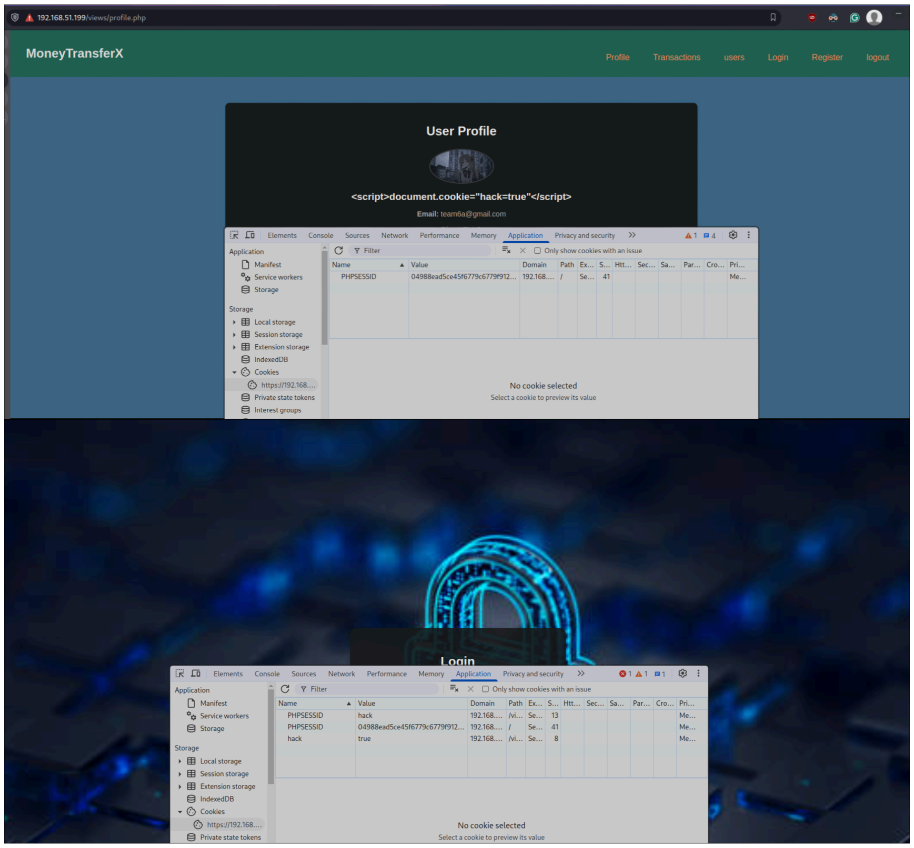


Edit Profile

Profile updated successfully!

Username	heker2
Email	heker2@gmail.com
First Name*	csrf
Last Name*	csrf
Phone No.*	9183918782
Biography	Updated using csrf
Profile Image	Choose file No file chosen

- XSS Attack on Team 9
    - Team 9 did not perform input sanitization on the username field. When creating an account, the username can be given as “<script>alert(document.cookie)</script>”
    - After creating the account, go to the search page and enter the search string “a”
    - Since the website does not sanitize inputs, the browser executes the username as a script.
- Session Fixation on Team 9
    - Using the XSS vulnerability, the session cookie can be fixed by JavaScript
    - The cookie can be accessed by HTTP because the session cookie is not set as an HTTP-only cookie(this can be changed in the php.ini settings)
    - In the php.ini file, the session.use_strict_mode option will prevent uninitialized session cookie values, but this option is disabled by default
    - By setting an invalid session cookie, the user can also be logged out. This will be a DoS attack.



<script>document.cookie="hack=true"</script>

Email: team6a@gmail.com

Name	Value	Domain	Path	Ex...	S...	Htt...	Sec...	Sa...	Par...	Cro...	Pri...
PHPSESSID	04988ead5ce45f6779c6779f912...	192.168....	/	Se...	41						Me...

Name	Value	Domain	Path	Ex...	S...	Htt...	Sec...	Sa...	Par...	Cro...	Pri...
PHPSESSID	hack	192.168....	/vi...	Se...	13						Me...
PHPSESSID	04988ead5ce45f6779c6779f912...	192.168....	/	Se...	41						Me...
hack	true	192.168....	/vi...	Se...	8						Me...

Lessons Learnt & future security enhancements

Even though there were no vulnerabilities found in our website, there are security enhancements to be made. The following are the security enhancements that can be implemented:

1. Improve session management: In the current implementation, there is no timeout for the session token. A correct implementation will refresh the session token after a time interval. It is also recommended that the session token be refreshed for every state change that occurs in the website(Example: login, logout, transaction)
2. Using built-in security frameworks: It is recommended to use security frameworks rather than implementing our security features from scratch. Even though it is necessary to learn how the security frameworks work, implementing them from scratch could cause more vulnerabilities.
3. Using a better way to store credentials: The current implementation uses an .env file to store SQL database passwords. It is better to store the credentials in an encrypted format where only the authorized users can access them.
4. Updating dependencies: If our website uses plugins/dependencies, then a vulnerability in the plugin could also be exploited. Checking for vulnerabilities in dependencies and keeping them updated is also important.
5. Data at rest should be encrypted: The current website implementation does not use encryption for storing MySQL data. A further improvement to the website would be to configure data-at-rest encryption in MySQL.

Contributions

1. Divyanshu Ranjan - CS24MTECH11013
   - Frontend Code
2. Jayanth Jatavath - CS24MTECH11014
   - Frontend Wireframe Design
3. Peddi Manognya - CS24MTECH12020
   - Frontend Code
4. Krishna Teja B - CS24MTECH12011
   - Docker setup

  - Backend code(php, mysql)
5. Sai Sravan V - CS24MTECH02007
  - Input validation(how to prevent attacks)
  - Testing attacks website and attacks on the website

References

1. MySQL Transactions: https://dev.mysql.com/doc/refman/8.4/en/commit.html
2. HTTP header-based authentication: https://security.stackexchange.com/questions/91546/any-reasons-for-using-basic-http-authentication
3. MySQL UUID support: https://dev.mysql.com/blog-archive/mysql-8-0-uuid-support/
4. MySQL handling UUID: https://dev.mysql.com/blog-archive/storing-uuid-values-in-mysql-tables/
5. Apache TLS Configuration: https://httpd.apache.org/docs/2.4/ssl/ssl_faq.html#gid
6. Sanitizing Images: https://www.reddit.com/r/web_design/comments/3zsrqr/sanitizing_uploaded_images_through_php/
7. PHP payloads in PNG: https://www.synacktiv.com/publications/persistent-php-payloads-in-pngs-how-to-inject-php-code-in-an-image-and-keep-it-there.html
8. Uploading files in PHP: https://stackoverflow.com/questions/33898834/uploading-a-profile-picture-and-displaying-it
9. HTML Templates in PHP: https://stackoverflow.com/questions/13071784/html-templates-php
10. Docker compose example config: https://github.com/Fresh-Advance/development/blob/master/services/mysql.yml
11. Integrate Apache and PHP-FPM: https://www.reddit.com/r/PHPhelp/comments/yvoo9r/apachephp_and_phpfpm_in_separate_docker/
12. PHP router: https://www.freecodecamp.org/news/how-to-build-a-routing-system-in-php/

13. PHP and PHP-FPM setup: https://www.pascallandau.com/blog/php-php-fpm-and-nginx-on-docker-in-windows-10/)
14. Variables in Docker compose: https://serversforhackers.com/c/div-variables-in-docker-compose
15. PHP and PHP-FPM setup 1: https://stackoverflow.com/questions/29905953/how-to-correctly-link-php-fpm-and-nginx-docker-containers
16. YouTube tutorial on PHP 1: https://www.youtube.com/watch?v=BSvzZvw_T64&list=PLQH1-k79HB396mS8xRQ5qih5iqkQw-4aV
17. YouTube tutorial on PHP 2: https://www.youtube.com/watch?v=yy8QogjpuiI&list=PLIorEuqMFFjMOoduM9Ijk7Y7Oz88lG8Q1&index=19
18. Apache Redirect All requests to index.php: https://www.slashnode.com/articles/devops/2013-12-24-redirect-all-requests-to-index-php
19. Using exit after redirect: https://thedailywtf.com/articles/WellIntentioned-Destruction
20. Docker compose docs: https://docs.docker.com/reference/cli/docker/compose/
21. PHP File upload guide: https://inspector.dev/ultimate-guide-to-php-file-upload-security/
22. PHP Send file: https://stackoverflow.com/questions/2882472/send-file-to-the-user
23. PHP Session settings: https://www.php.net/manual/en/session.security.ini.php
24. PHP official Documentation: https://www.php.net/manual/en/index.php
25. PHP mysqli Documentation: https://www.php.net/manual/en/intro.mysqli.php

Anti-Plagiarism statement

We certify that this assignment/report is our work, based on our study and/or research, and that we have acknowledged all material and sources used in its preparation, whether books, articles, packages, datasets, reports, lecture notes, or any other document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that We have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, we understand our responsibility to report honor violations by other students if we become aware of them.

Team 6:

- Krishna Teja B - CS24MTECH12011
- Sai Sravan V - CS24MTECH02007
- Divyanshu Ranjan - CS24MTECH11013
- Jayanth Jatavath - CS24MTECH11014
- Peddi Manognya - CS24MTECH12020

Date: March 30, 2025