PROJECT DOCUMENTATION

Web app sketch
Multi-factor authentication

Authors

Szymon Klimaszewski

Kajetan Łachański

Igor Pietruszczak

Aplication outline

User visiting the website is redirected to the /login page. There, they can either log in or register. During the registration process, the user authenticates their account using TOTP (a personal QR code is displayed) and PUSH notifications. If all the authorizations (username, password, TOTP, and PUSH) are correct during the login process, the user will be redirected to the /home page, which is inaccessible to unauthorized users.

Risk analysis

1. The website should only function through HTTPS, not HTTP.

Implementing HTTPS on localhost is not convenient, so we were allowed to omit it and only mention HTTPS in the documentation. HTTPS ensures encryption and integrity of transmitted data. HTTPS also authenticates websites, minimizing the risk of users accessing malicious sites.

Therefore, there is no easy option for data sent and received to be intercepted and leaked, which is one of the ways websites can be attacked.

2. Multi-Factor Authentication

In our application, in addition to username and password, we have implemented TOTP authentication and PUSH notifications. This significantly improves user security. It also reduces "physical" risks, such as password guessing, as a malicious user would not be able to log in without the TOTP code.

3. Data storage.

Our application implements storing data in an encrypted SQL database. Therefore, a leak of the database itself should not result in a leakage of user data.

Within the database, we have also implemented traditional measures to counter the common SQL injection attack. In our database queries, we use "parametrized queries" that ensure the data sent to SQL is not treated as code that could be executed, thus preventing any unexpected actions.

3. Anti-brute force.

To prevent malicious users from attempting to break passwords through brute-force attacks, we have implemented a simple algorithm. If a particular username attempts to log in more than X times within Y time period, they will be locked out for Z duration. This security measure significantly increases the time required for a brute-force attack to succeed.

We have also enforced a simple rule during password creation, requiring a minimum of 8 characters. This approach is not burdensome for users and further increases the time required for an attack.

4. Other security measures.

In our application, we implement several other simple and effective security measures.

Each user's username must be unique. This is used, for example, to store data about incorrect login attempts and temporarily block users.

All user inputs, such as passwords during registration, are checked against specific rules and then sanitized to remove dangerous characters (e.g., "<"). Such characters could pose a threat, especially when displayed on an HTML page.

Redirecting the user when they try to access unauthorized pages.

Different status information is stored in a secure session of the Flask application

5. Other measures that could be implemented

Periodic rotation of SQL database encryption or even multiple smaller databases.

Using salt when storing data (the same data is encrypted differently).

In summary:

1. Attack vector: interception of transmitted data
Security model: using HTTPS encryption

2. Attack vector: password suspicion/guessing/leakage
Security model: implementing multi-factor authentication

3. Attack vector: database leakage, SQL injection
Security model: encrypted SQL database with proper data structure ("parametrized queries")

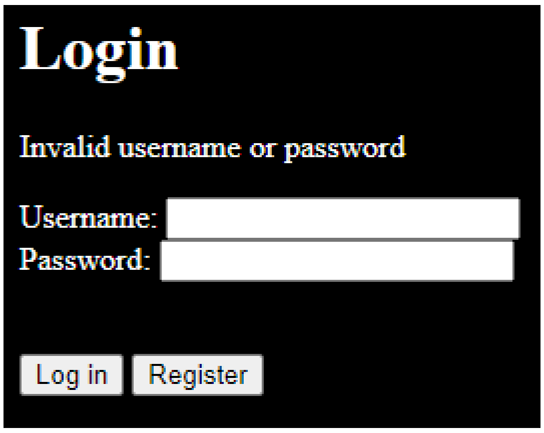
4. Attack vector: brute force
Security model: limiting failed login attempts and requiring an 8-character password

5. Attack vector: malicious user input
Security model: validating and sanitizing user input

6. Attack vector: attempting to access incorrect pages
Security model: requiring a secure session in Flask application and immediate redirection

Testing

Invalid username or password

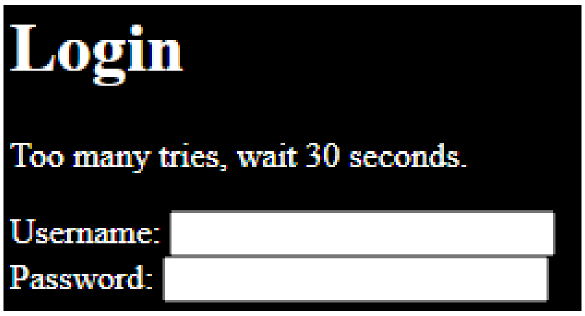


Login

Invalid username or password

Username:
Password:

Brute-force try

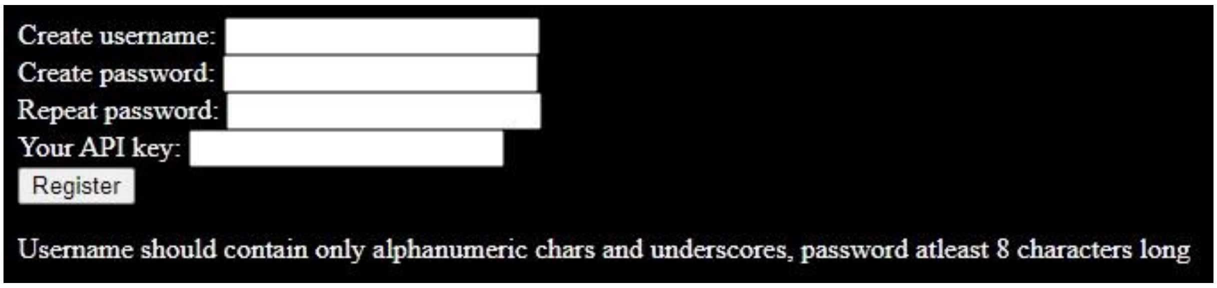


Login

Too many tries, wait 30 seconds.

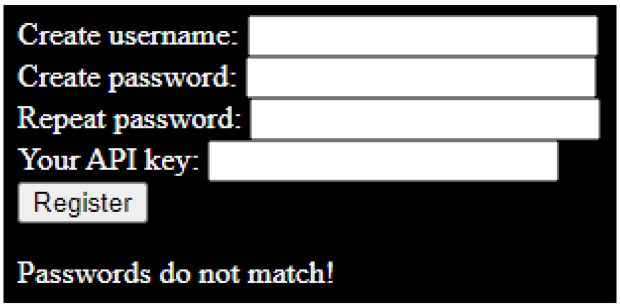
Username:
Password:

Invalid username/password during registration



Create username:
Create password:
Repeat password:
Your API key:

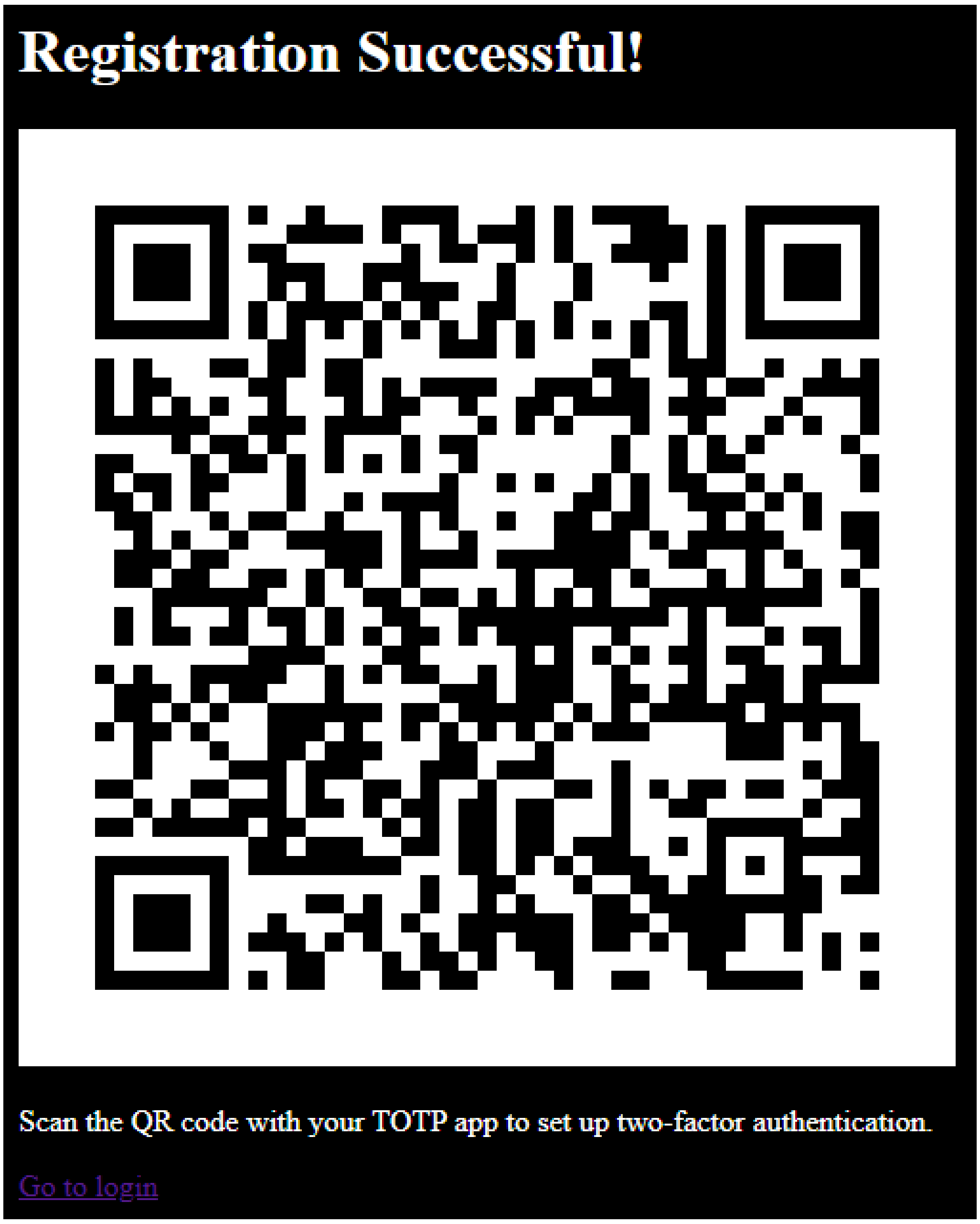
Username should contain only alphanumeric chars and underscores, password atleast 8 characters long



Create username:
Create password:
Repeat password:
Your API key:

Passwords do not match!

One-time QR code displayed during registration, for TOTP codes.

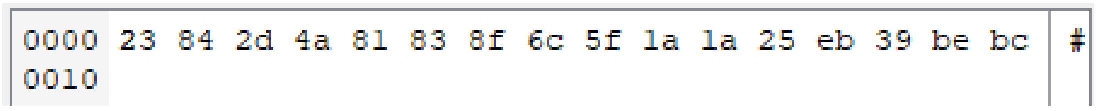


Registration Successful!

Scan the QR code with your TOTP app to set up two-factor authentication.

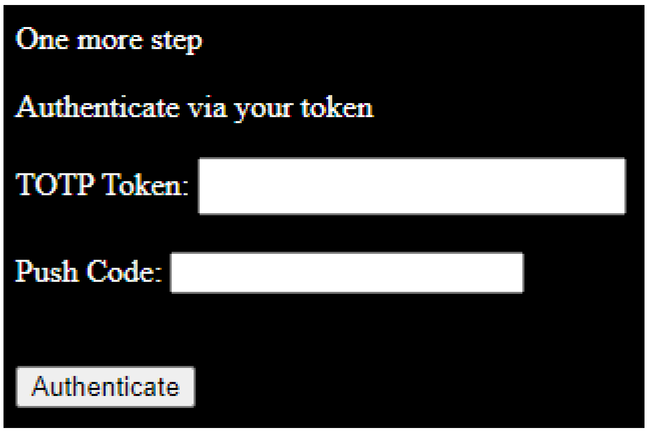
Go to login

SQL database (sample encrypted user's password)



```
0000 23 84 2d 4a 81 83 8f 6c 5f 1a 1a 25 eb 39 be bc  #
0010
```

Multifactor auth


One more step

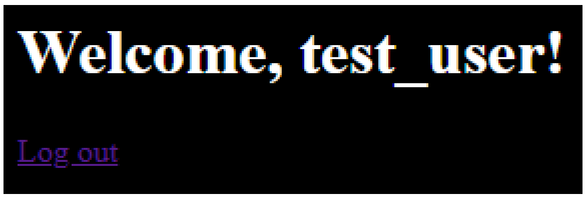
Authenticate via your token

TOTP Token:

Push Code:

Authenticate

Valid login


Welcome, test_user!

Log out