Assignment #2

Release Date: Beginning of Week 4
Due Date: End of Week 6
TA emails:
Office Hours: Mon-Fri, 10AM-4PM

Introduction

This assignment focuses on secure Web development. For this assignment, you are tasked with turning your spell checking system from Assignment 1 into a Web service using Django, while focusing on the security of the Web service you are implementing. After you develop the secure Web service, you will test it to ensure it is not vulnerable to common attacks. Though it is not part of the grade, you should continue to use the secure development practices that you established in Assignment 1.

Setup

Completion time – 1 hour

For this assignment you are required to use the Flask Web framework. The Flask web framework can be installed using pip.

```
$ sudo pip install flask
```

If your system does not have pip installed, you can install it through your system's package manager or from https://pypi.org/project/pip/.

For python projects, tox is a good framekwork for running tests. For more information on Tox, see https://tox.readthedocs.io/en/latest/. Tox can be installed using pip.

```
$ sudo pip install tox
```

However, in order to get tox to work well with Travis CI, you may need the tox-travis plugin for Travis. For more information on this plugin, see https://tox-travis.readthedocs.io/en/stable/. This can also be installed using pip.

```
$ sudo pip install tox-travis
```

Deliverables & Grading

The source code and reports are to submitted through NYU Classes. Your submission should be a compressed archive file, with all reports under the “Reports” folder and all source code under the repository folder. An example file tree is given below.

```
├── Reports
│   ├── lname_report1.pdf
│   └── lname_report2.pdf
└── repository
    ├── .gitignore
    ├── .travis.yml
    ├── README
    ├── ...
    ├── tests
    └── src
```

1. Web service code - n pts.
2. Tests – m pts.
3. Write-ups – 100 – (n + m) pts.

Total 100 pts.

Part 1

Completion time – 3 to 10 hours, depending on experience

In the first part of this program, you are tasked with creating a Web service to provide spell checking capability (using the program you created in Assignment 1) to registered users. In order to achieve this, you Web service must provide at least the following functionality, at the following locations:

1. User registration: /your/webroot/register
2. User login: /your/webroot/login
3. Two-factor authentication: /your/webroot/login
4. Text submission: /your/webroot/spell_check
5. Result retrieval: /your/webroot/spell_check

User Registration

Users must be registered in order to use your service. Your registration page is required to have the following forms for the user to fill in:

1. A form for the user to enter a username, with id=uname.
2. A form for the user to enter a password, with id=pword

3. A form for the user to enter a two-factor authentication device, with id=2fa

The standard rules of registration apply. Usernames must be unique, and users must supply at least a username and password to be able to register for the service. When registration is complete, users should be shown a success message in an element with id=success. If the registration failed, the user should be shown a failure message in the element with id=success. The words “success” or “failure” (capitalization irrelevant) must be present in the message.

User Login

After a user registers, he or she should be able to login to your Web service. Your login page is required to have the following forms for the user to fill in:

1. A form for a user to enter a username, with id=uname.
2. A form for a user to enter a password, with id=pword.
3. A form for the user to enter a two-factor authentication code, with id=2fa

In the following cases, your login page should return an error in an element with id=result with the assocaited string in it:

1. Incorrect username or password: “Incorrect”
2. Two-factor authentication failure: “Two-factor” and “failure”

If a user logs in successfully, your login page should return a success message in an element with id=result. The message must contain “success.” Your Web service should do all of the necessary steps for session management.

Text Submission

After a user is logged in, he or she has the ability to submit bodies of text to check the spelling of the words in the text. The text should be submitted through a form with id=inputtext. Your Web service should then take this text and use the spell checker you wrote in Assignment 1 to determine which words are misspelled.

Result Retrieval

After the results of the C program are returned to your Web service, your Web service must output the results to the user. The supplied text should be output in an element with an id=textout and the misspelled words, separated by commas, should be output in an element with id=misspelled.

Security Requirements

Each of these functionalities must be provided securely. When writing your Web service, you must defend against common attacks against Web services such as

- XSS

- CSRF
- session hijacking
- Command injection
- etc.

For this assignment, it is sufficient to keep the data in memory only. In other words, using a database is not required, and therefore you do not need to worry about SQLi attacks.

Report

In addition to the Web service code, you are required to submit a write-up that details the following:

- design decisions for the Web service
- how you mitigated different categories of common Web vulnerabilities

The following example is insufficient for protecting against XSS, but illustrates the types of explanations we are looking for in the report.

> *For this program we were required to take text input from the user through a text input field. However, unsanitized input that could later be shown to the user could lead to potential cross-site scripting vulnerabilities. For this reason, we decided to strip all of the '<' and '>' characters from user inputted strings.*

Part 2

Completion time – 3 to 5 hours, depending on experience

After you design and implement your Web service using Flask, you must test your Web serice to detemine if it is vulnerable to common Web service vulnerabilities such as session hijacking, XSS, CSRF, etc. To do this you should attempt to perform such attacks on your Web service. Like Assignment 1, when you find a vulnerability you should patch it and create a new test that would ensure that such a vulnerability does not occur in the future.

In addition to submitting all new tests and patched vulnrerabilities, you must also submit a write-up that describes the vulnerabilities you found, why they occurred, why you didn't catch them during the first part of the assignment, and how you patched them.

Extra Credit

Extra credit may be provided to submissions that go above and beyond the requirements of the assignment. There is no definitive formula or assignment for extra credit.

Hints

You may wish to include your spell checker code as a git submodule. You can read more about git submodules at https://git-scm.com/book/en/v2/Git-Tools-Submodules.
Are your passwords stored securely? Are cookies able to be read by JavaScript? If so, how can you ensure that the session will not be hijacked? There are two types of XSS attacks specifically mentioned in the lectures. Is your Web service protecting against both of them? Can users execute Linux commands through your service?

Late Policy

Late assignments will not be accepted.