Kevin Zhang

Application Security CS GY 9163

**Homework 2: When a Wreck Reaches the World Wide Web**

On NYU Classes, submit a link to your GitHub repository. The repository should be **private**, and you should add the instructor/TA's GitHub account as a contributor to give them access for grading. For this section, your instructor is: **Kevin Gallagher**, GitHub ID `**kcg295**`. Your TA is: **Evan Richter**, GitHub ID `**evanrichter**`

**Author note:** Please note that all comments to code created by me are indicated by **//KZ:.** This is to assist in grading and locating changed parts of the code.

Part 0: Setting up Your Environment

Setting up Django

* Used pip to install Django 3.2.2
* Installed Python 3.9.5
* Utilized Virtual box

In order to complete this assignment, you will need the git VCS, Travis, python 3 and the Django web framework. You can install Django using the following command:

sudo pip3 install django

Some additional tools that may be useful for this assignment (but are not necessary) are **sqlite**, **burp suite**, the **python requests library**, and the **web development console** of your favorite browser. **If you are runing a \*NIX system, these tools should be pre-installed and/or available in your distribution's package manager**.. **However, we will be checking for signed commits, since they are security relevant**.

Part 1: Auditing and Test Cases

Start off by copying the files from this repository into your own, and add them to git. The files and directories you need are:

GiftcardSite LegacySite images templates manage.py

After you compy these directories and files over, be sure to generate the database that django relies on. This can be done by running the commands:

python manage.py makemigrations LegacySite

python manage.py makemigrations

python manage.py migrate

sh import\_dbs.sh

Read through the models.py and views.py files (and the helper functions in extras.py) in the LegacySite folder to get a feel for what the web site is doing and how. You can also try running the test server and interacting with the site by running the following command and browsing to 127.0.0.1:8000.

python manage.py runserver

For this part, your job will be to find some flaws in the program, and then create test cases that expose flaws in the program. You should write:

1. *One* attack, that exploits a XSS (cross-site scripting) vulnerability.
2. *One* attack that allows you to force another user to gift a gift card to your account without their knowledge.
3. *One* attack that allows you to obtain the salted password for a user given their username. The database should contain a user named ``admin.''
4. *One* attack that exploits another attack not listed above on the server. Some hints for this section are: looking at the way the passwords are stored, or looking at how interactions are done with the giftcardreader binary.
5. A text file, bugs.txt explaining the bug triggered by each of your attacks, and describing any other vulnerabilities or broken functionalities you came across. There are more than the bugs mentioned above.

These attacks can take the form of a supplied URL, a POST made to the web page, a gift card file, a web page, a javascript function, or some other method of attack. To create your attacks, you may want to look at the HTML source code of the templates and the code of each view, and find a way they can be exploited. Tools like burp suite can help in finding ways to attack the site, but are not required. Please submit these attacks in a folder called "part 1" in your git repository.

Finally, fix the vulnerabilites that are exploited by your attacks, and verify that the attacks no long succeed on your site. You are allowed to use django plugins and other libraries to fix these vulnerabilities. To make sure that these bugs don't come up again as the code evolves, write some test cases for django that test for these vulnerabilites. Then have Travis run these tests with each push.

When you are finished with this section, please mark your part 1 submission by tagging the desired commit with the tag "part\_1\_complete"

Part 2: Encrypting the Database

Currently the website uses a database that contains valuable gift card data. If an attacker gets access to this gift card data, they can use the cards they got to obtain free merchandise, or even pay of their tuition with the NYU tuition gift cards! For this reason your company needs to make sure that even if the database somehow leaks, the attacker will have a hard time using the cards.

Your company asked Shoddycorp's Cut-Rate Contracting to encrypt the database, but it seems they did not know how to do that, or did not want to. The code you received does not encrypt the database at all, but your company wants to ensure the data's protection at rest.

Your second job, therefore, is to modify this code to encrypt the data in the database. You are allowed to use django plugins or external libraries to implement this. Please see the lecture content for tips on proper key management and the different methods of doing database encryption.

When you are finished with this part of the assignment, please briefly explain how you implemented the database encryption, how you managed keys, and why you choose to manage keys that way. This should be stored in a file called "encryption\_explanation.txt" in a folder called "part 2" in the git repository.

When you finish this part of the assignment, please mark your part 2 submission by tagging the desired commit with the tag "part\_2\_complete"

**Grading**

Total points: 100

Part 1 is worth 65 points:

* 25 points for your attack cases
* 15 points for all fixes
* 10 points for the bug writeup
* 10 points for Travis regression testing
* 05 points for signed git commits.

Part 2 is worth 35 points:

* 10 points for encrypted database models
* 10 points for proper key management
* 15 points for your writeup.