Application Security

CS-GY – 9163 Lab 1b: Fix Your Web App

Suyash Soumya(ss11449)

The following assignment aims to fix the vulnerabilities from Lab 1a. The vulnerable application has been modified to no longer be prone to these vulnerabilities.

Passwords were stored in the database with vulnerable encryption:

The application allowed users to register using a username, email and password. However the passwords were stored in the MySQL database using a very vulnerable encryption algorithm MD5.

Although it was better than having no encryption, MD5 was a widely used algorithm but over the years it has been proven to have several vulnerabilities. MD5 shows that attacks from Brute Force attack and Collision attack has been successful in breaking the application.

In order to fix this, the passwords were hashed using a more secure encryption algorithm, in this case, sha 256. This gave us a secure password. To secure it further, the original password was concatenated with salts at the beginning and end. Then this salted password was run through sha 256 encryption to obtain a secure key.

When Logging in, we use the same encryption method. When the user enters his credentials, the plaintext password is appended with salts used previously before and after. Then it is encrypted using the same algorithm sha 256 and then matched with the database. If it matches the records then he is redirected to the upload imaging page.

Passwords entered in the HTML code was in plaintext

The password box in the HTML page accepted values as “text”. Hence anyone with access to the screen of the logging user can view the password as they enter in.

In order to fix this, we make the input type in the HTML code as “password”, which hides the text of the password and represents the values as dots.

Vulnerable to SQL injections

The input values in the registration page and login page had handled the SQL injections in the first part of the assignment, however the web application did not implement this for all pages, namely, for the image uploading, delete and caption pages.

To fix the SQL injection vulnerabilities in these pages, escape characters were not allowed in any of the fields.

Not checking for existing user records

The web application allowed for users to make as many accounts as they want. This can lead the database to have multiple accounts with the same name, hence collisions.

In order to fix this, we limit the database to accept only unique users. This prevents the web application to be more stable against brute force attacks.

Vulnerable to Brute Force Attacks

There is no mechanism to prevent brute force attacks, DDOS and automated Bot attacks.

In order to fix this, we incorporate captcha functionality which requires user to validate in order to be allowed access for which we use the Google recaptcha APIs.

Finding Vulnerabilities in partner student's (Sean Wang) application:

The app used python library: flask_uploads to upload images. In the code provided, the UploadSet is restricted to IMAGES, where only files ending with images can be accepted. However, this is an easily exploitable defect.

To attack this vulnerability, since the image will be displayed on screen as the information for pixel is loaded on the memory, the file can contain lines as codes and mask itself as an image file to cause unwanted results. In addition, the restricted file type is inclusive of all image types, whereas the instruction given explicitly stated: PNG, JPS, or GIF images.

A requirement of the first assignment was to allow visitors to the site to view uploaded images, 10 per page (use next/back links or buttons to bring them to the next 10 pictures), sorted by most recent.

A defect was found when checking through this requirement. The sorting algorithm used was implemented as part of the python library with getctime. However, the time is stored and managed reversely causing the web application to display most recent last. There is not an effective solution to fit this defect as the sorting algorithms and the information gatherer is given from the python library, however, an effort was made to ensure that the image will be display in chronological order.

The length of displayable images per page is restricted to 10 as commented and indicated in the code. However, there is an exploitable defect when rendering the display page. The next/back links can lead to errors when there are less than 10 images per page or too many images on the database. The rendering loads all images onto the server and goes through the list of images by slicing through 10 images.

A possible attack is to attack the web by uploading an insane large number of images to crash the server due to overload.

Extra credit: allow users to sign up for an account which will allow them to edit their picture captions and delete their uploaded photos:

This part of the assignment is incomplete as it is an optional requirement. However, in the main python code, they were leaking information of the pages. Sign-up and Sign-in links were present on the home page causing potential exploitable problems that can be devastating to the server running this problem.

Instructions on how to get the code running:

Set up database:

1. Start Xampp Control Panel and start Apache and MySql
2. Set up database in the MYSQL Server with name= “demo”.
3. Create table in “demo” called “visitors”.

   This table should have the following columns -

   “name” – VARCHAR (255)
   “photo” – Longblob

4. Create database in “registration” and a table called “users”.

   This table should have the following columns-

   “id” – integer(11) (primary key) [should be auto-incrementing]
   “username” – VARCHAR(255)
   “email” – VARCHAR(255)
   “password” – VARCHAR(255)

**we use username=’root’ password=’’

Running the code:

1. Start Xampp Control Panel and start Apache and MySql
2. Create a folder in Xampp/htdocs as images
3. Open browser and enter the homepage URL http://localhost/register.php
4. Login/Register as a new User by entering username, email and password.
5. Initially the home screen shows no picture.
6. Once user is authenticated/registration is completed, you will reach Log In page, click on Continue to image upload link
7. Now Select image to upload. Browse to folder and select the image you want to upload.
8. Enter the Caption.
9. Click on Upload image button
10. Reach the main upload page where all the images are displayed.
11. All the Caption Names are displayed on the left side of the screen in ascending order.
12. After uploading 10 or more images a button to go to next page is shown to go to next page. On next page you get the option of going back by clicking the GoBack button.
13. Option to delete the last picture uploaded from database at the end of page.
14. Can logout from the application on clicking the logout button or go to homepage by clicking homepage button.