Lovejoy Antiques and AWS VPC

Contents

[Important Information 3](#_Toc184301721)

[Task 0 – Self-reflection 4](#_Toc184301722)

[Task 1 – User registration 5](#_Toc184301723)

[Registration features and code screenshots 5](#_Toc184301724)

[Database Tables 10](#_Toc184301725)

[Why do you think it is secure? 12](#_Toc184301726)

[Task 2 – Develop a secure login feature 13](#_Toc184301727)

[Login features and code screenshots 13](#_Toc184301728)

[Why do you think it is secure? 18](#_Toc184301729)

[Task 3 – Implement password strength and password recovery 19](#_Toc184301730)

[Password entropy, recovery code screenshots 19](#_Toc184301731)

[Task 4 – Implement a “Evaluation Request” web page 25](#_Toc184301732)

[Request Evaluation features and code screenshots 25](#_Toc184301733)

[Why do you think it is secure? 28](#_Toc184301734)

[Task 5 – Request Listing Page 29](#_Toc184301735)

[Request Listing features and code screenshots 29](#_Toc184301736)

[Why do you think it is secure? 31](#_Toc184301737)

[Task 6 – AWS Virtual Private Cloud settings screenshots 31](#_Toc184301738)

[Instances Running 31](#_Toc184301739)

[Security groups rules 32](#_Toc184301740)

[Routing tables 33](#_Toc184301741)

[All subnets 33](#_Toc184301742)

[Resource Map 35](#_Toc184301743)

# Important Information

Code file location - <https://1drv.ms/f/s!ApPLUhOl36G5gZ4KjSjTaYlWsojq7w?e=CdD9IS>

Panopto recording - <https://sussex.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=20b5765a-297d-4040-a8de-b23c0151ae38>

SQL Database setup file - <https://1drv.ms/u/s!ApPLUhOl36G5gaEOXFbaDLIGbRrluA?e=SNw88R>

# Task 0 – Self-reflection

|  |  |  |  |
| --- | --- | --- | --- |
| **Marking criteria** | **Sub criteria** | **Tick/cross** | **Marks**  (from the main marking grid, assign fair marks to yourself) |
| Password policy | Password entropy | X | 13 |
| Security questions | X |
| Password recovery | X |
| Vulnerabilites | SQL injection, | X | 12 |
| XSS, | X |
| CSRF, | X |
| File Upload and | X |
| any other obvious vulnerability. | X |
| Authentication/Encrypted storage | User registration, User login | X | 9 |
| Email verification for registration, | X |
| 2 factor authentications (PIN and or email) | X |
|  | Encrypted storage | X |
| Obfuscation/Common attacks | Brute force attack – Number of attempts | X | 8 |
| Botnet attack – Captcha | X |
| Dictionary attack/Rainbow table attack | X |
| Features of web application | Database design | X | 30 |
| User registration | X |
| User login | X |
| Forgot password | X |
| Evaluation | X |
| List evaluation | X |
| VPC | Evidence provided | X | 10 |
| Video | All the marking criteria covered | X | 6 |
| Self-reflection | This marking grid fill out properly | X | 4 |
|  |  |  | Total marks = 92 |

# Task 1 – User registration

## Registration features and code screenshots

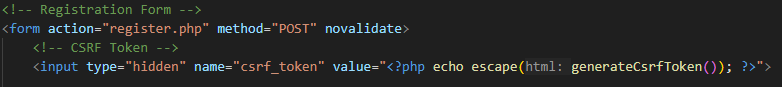
User registration form:

A screenshot of a computer

Description automatically generated

* Within this form the user needs to enter in all of the credentials

Protection against CSRF and SQL Injection:



A computer screen with text

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer code with colorful text

Description automatically generated

Same origin policy and HTTPS

A screen shot of a computer code

Description automatically generated

Session fixation

A screen shot of a computer code

Description automatically generated

Email

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generatedPassword (1st image checks against 10 million common passwords)

A screen shot of a computer program

Description automatically generated

A computer screen with white text

Description automatically generated

Other Validations

A computer screen shot of text

Description automatically generated

XSS Protection (a sample of all the input forms)

A computer screen shot of text

Description automatically generated

A black screen with white text

Description automatically generated

Prepared statements

A computer screen shot of a program code

Description automatically generated

Verification Email (sent using PHPMailer)

A computer screen shot of a program code

Description automatically generated

A close-up of a email

Description automatically generated

Token for the email

A black screen with white text

Description automatically generated

Checks the token

A computer screen shot of code

Description automatically generated

If token correct A screen shot of a computer program

Description automatically generated

Resending verification email (checks if the user is locked from sending more verification emails and amount of attempts) A computer screen with text on it

Description automatically generated

A screen shot of a computer code

Description automatically generated

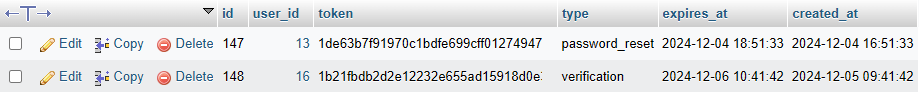
## Database Tables

Users

A screenshot of a computer

Description automatically generated

Tokens



User Attempts

A screenshot of a computer

Description automatically generated

User Security Answers

A screenshot of a computer

Description automatically generated

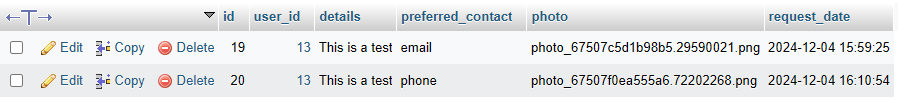
Security Questions  
A screenshot of a computer

Description automatically generated

User 2FA



Evaluation Requests



## Why do you think it is secure?

Same-origin, HTTPS and session fixation

* Uses the same-origin policy and HTTPS, using a self-generated certificate for HTTPS
* Regenerates session if you go onto a different page (embed in the header is the init.php file)

Sanitising Inputs and CSRF

* The inputs are put through html special chars and UTF-8
* Once the input is put through, they are sanitised against SQL Injections
* Uses CSRF token on the registration form

Password Entropy

* Forces the user to enter in a password that has 8 or more characters, lower- and upper-case letter, number and special characters
* If the password passes this but is still a vulnerable password (Pa$$word1) then it will still reject this so that it more protected against dictionary attacks
* These passwords are hashed and salted so that if a user enters in the same password as someone else then the hash will be different, protecting against rainbow table attack

Email

* Checks if the email is already used by a user
* Have to verify your email before you can login, so users can’t use a fake email or create loads of accounts without real emails

Security Questions

* The user has to select unique security questions
* These security answers are hashed using the same hashing algorithm (BCRYPT) as the password

Verification Email

* Sends a verification email as soon as the user has registered an account
* The previous email is invalid if the user asks for another email
* This email link expires after 24 hours
* Can’t send more than 5 emails in 24 hours
* Sanitises the token so that the user can’t enter in a fake one for an SQL Injection

# Task 2 – Develop a secure login feature

## Login features and code screenshots

Login form

A screenshot of a computer

Description automatically generated

CRSF, XSS and SQL Injection protection

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

Validate user inputs

A computer screen shot of a program code

Description automatically generated

A black screen with text

Description automatically generated

User attempts (Brute force)



A computer code on a black background

Description automatically generated

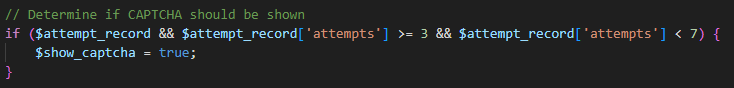
A black screen with colorful text

Description automatically generated

Locking Account or IP for 30 minutes (Max attempts 7)

A screen shot of a computer program

Description automatically generated

CAPTCHA to protect against Botnet attack (after 3 attempts) 

A black screen with white text

Description automatically generated

If not verified disallows login and adds an attempt to the login attempt, uses the same policy of maximum 5 attempts and lock for 24 hours (shown in registration)

A screen shot of a computer

Description automatically generated

A screenshot of a email

Description automatically generated

A screenshot of a email

Description automatically generated

Can’t access the resend verification page if you are logged in

A screen shot of a computer

Description automatically generated

Authenticate User

A computer screen shot of text

Description automatically generated

Once logged in sends a 2FA 6-digit code to email, using the same PHPMailer settings as Task 1

A white background with black text

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Stores the 2FA code hashed using BCRYPT and it expires within 10 minutes

A computer screen with text and images

Description automatically generated

A screen shot of a computer code

Description automatically generated

Can resend the 2FA Code up to 5 times with a new code each time

A computer screen shot of text

Description automatically generated

Can try to enter in an incorrect 2FA Code up to 5 time before it locks (checks the hashes)

A computer screen with colorful text

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

Once verified it will go the index page

A screen shot of a computer program

Description automatically generated

A screenshot of a website

Description automatically generated

Can’t access the 2FA form unless you are logged in



**Forgot Password shown in Task 3**

**User 2FA Database shown in Task 1**

## Why do you think it is secure?

Same-origin, HTTPS and session fixation

* Same as in Task 1

Sanitising inputs and CSRF Token

* Uses XSS and sanitises on all inputs in the 2FA, email, password and forgot password
* Has a CSRF Token for all forms

Protection against Botnet and brute force (Login)

* Only allows up to 7 login attempts for an email and then locks out account
* Only allows up to 7 attempts for an IP
* Shows a CAPTCHA to complete after 3 attempts

2FA Code

* Only allows 5 2FA code attempts and then locks out
* Can only send up to 5 2FA Codes
* Previous code is invalid after new sent
* Code expires after 10 minutes
* The Code is hashed and salted using BCRYPT to be stored in the database

Verification

* Only allow verified users to be able to login

Forgot Password

* Allow user to send a reset password link to their email if forgotten password (with a limit of 5)

# Task 3 – Implement password strength and password recovery

## Password entropy, recovery code screenshots

* Password Entropy (8 characters, special character, capital letter, lower case letter and number)

A screen shot of a computer program

Description automatically generated

* Checks against 10 million common passwords to protect against dictionary attack

A computer screen shot of a program code

Description automatically generated

A black background with white text

Description automatically generated

Snippet of file, can be found in the scripts folder under 10 million passwords

A screenshot of a computer

Description automatically generated

* Forgot password in login form

Check CSRF and sanitise input

A computer code with text

Description automatically generated

A screen shot of a computer code

Description automatically generated

Shows that an email is sent even if the email doesn’t exist, so that the user can’t get database information based on if an email is sent



A black screen with white text

Description automatically generated

Send a link to the email entered in using the PHPMailer settings from Task 1 and rate limit so that it can only send a limit of 3 within 24 hours

A screenshot of a chat

Description automatically generated

A screen shot of a computer code

Description automatically generated

Much like the verification, the previous email expires once you request another email.

The token expires after 1 hour

A black screen with white text

Description automatically generated

A screen shot of a computer code

Description automatically generated

The password reset form, using the same password entropy and common password check as register

A screenshot of a login page

Description automatically generated

Once logged in you can reset your password using a email link (the same as in forgot password so if you are locked out from that you will be locked out from this one)

A screenshot of a computer

Description automatically generated

Additionally, you can reset your password using your Security questions

A screenshot of a computer

Description automatically generated

XSS, CSRF and SQL Injection protection

A screen shot of a computer code

Description automatically generated

A computer screen shot of code

Description automatically generated

Checks each input isn’t empty and that it matches the hash in the database

A computer screen shot of code

Description automatically generated

If all correct then it will go to the reset password form with a tag of security\_questions=1 tag, this is checked by against the $\_session token[‘can\_reset\_password’] (all session tokens are reset if you go to a different page)

A computer screen shot of a code

Description automatically generated

A screen shot of a computer

Description automatically generated

Form validation using CSRF, sanitising, checking password validation and removing the token data

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

Password reset form (after security questions) uses the same password entropy and common password check

A screenshot of a login page

Description automatically generated

Can’t access these forms unless you are logged in or through a valid password reset link



**All forms use same-origin, HTTPS and session fixation as shown in task 1.**

**The user attempts table can be found in task 1.**

# Task 4 – Implement a “Evaluation Request” web page

## Request Evaluation features and code screenshots

Request evaluation form

A screenshot of a register

Description automatically generated

Users get a choice between email or phone as a method of contact

XSS, SQL Injection and CSRF validation

A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

Validate details and preferred contact

A screen shot of a computer code

Description automatically generated

Validate file upload

A screen shot of a computer program

Description automatically generated

Check MIME file

A screen shot of a computer program

Description automatically generated

Check valid image (JPEG, PNG ,GIF)

A screen shot of a computer code

Description automatically generated

Check valid size

A computer screen shot of a code

Description automatically generated

If passed, encrypt the photo using my encryption key in my config file

A screenshot of a computer program

Description automatically generated

A computer screen with text

Description automatically generated

Generate a unique file name

A screen shot of a computer code

Description automatically generated

Save the encrypted file to uploads

A screen shot of a computer

Description automatically generated

Set strict file permissions (0600 means gives the owner of a file full read and write access, while preventing other users from accessing the file)

A screen shot of a computer code

Description automatically generated

**A computer screen shot of a program

Description automatically generated**

Inputting this into the SQL table

**A screen shot of a computer program

Description automatically generated**

Can’t access form unless logged in



A screen shot of a computer code

Description automatically generated

**The evaluation\_requests database can be found in task 1**

## Why do you think it is secure?

Same-origin, HTTPS and session fixation

* Same as in Task 1
* Can’t access form unless logged in

Sanitising inputs and CSRF Token

* Uses XSS and sanitises on the details input
* Has a CSRF Token for all forms

Details

* Details have to be inputted and are sanitised
* Details have to be below 1000 characters

File Upload

* The photo gets check that is either a PNG, JPEG or a GIF
* The size of the image is below 2mb
* Check the image for MIME file so you can’t spoof a file type
* Set the permissions for the file as chmod 600 which gives the owner of a file full read and write access, while preventing other users from accessing the file
* The file is encrypted in the storage and is only decrypted when an admin requests for the requests

# Task 5 – Request Listing Page

## Request Listing features and code screenshots

The evaluation page

A screenshot of a computer

Description automatically generated

XSS

A screen shot of a computer code

Description automatically generated

Get the requests form the database

A screen shot of a computer

Description automatically generated

Gets the file path and check that it exists and if so, it gets the content

A screen shot of a computer program

Description automatically generated

Decrypt the photo

A screen shot of a computer program

Description automatically generated

A computer code on a black background

Description automatically generated

Checks that the user accessing the page is an admin



A screen shot of a computer code

Description automatically generated

A computer screen with text

Description automatically generated

**The encrypted photos can be found in the uploads file**

## Why do you think it is secure?

Same-origin, HTTPS and session fixation

* Same as in Task 1
* Can’t access form unless admin

XSS

* The details are all sanitised using html special chars

Photo

* The photos are encrypted until you try and access the requests and then they are decrypted and can only be seen on that page

# Task 6 – AWS Virtual Private Cloud settings screenshots

## Instances Running

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Security groups rules

Apache MySQL and PHP

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Windows server 2016

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Routing tables

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

## All subnets

A screenshot of a phone

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a group

Description automatically generated

A screenshot of a group

Description automatically generated

## Resource Map

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated