Final Report

User blog service

Kalina petrova (2221667)

jiaqi ni (221774)

Mengchua liu (2151655)

jiefan lin (2617269)

ishant upadhyay (2309173)

jianfei feng (2204529)

2016

Contents

[Introduction 3](#_Toc464836891)

[Web application functionalities 4](#_Toc464836892)

[User groups 4](#_Toc464836893)

[Use case scenarios 4](#_Toc464836894)

[1. Register 4](#_Toc464836895)

[2. Log in 5](#_Toc464836896)

[3. Update avatar 5](#_Toc464836897)

[4. Comment 5](#_Toc464836898)

[5. Change password 5](#_Toc464836899)

[6. Administrator bans user 6](#_Toc464836900)

[7. Administrator deletes comment. 6](#_Toc464836901)

[Threat Analysis 7](#_Toc464836902)

[CIA Analysis 7](#_Toc464836903)

[Permission scheme 7](#_Toc464836904)

[Risk Analysis 7](#_Toc464836905)

[Brute force and dictionary attacks 7](#_Toc464836906)

[SQL injections 8](#_Toc464836907)

[Cross site scripting (XSS) 8](#_Toc464836908)

[Cross-site request forgery (CSRF) 8](#_Toc464836909)

[File upload 9](#_Toc464836910)

[File inclusion 9](#_Toc464836911)

[Misuse cases 10](#_Toc464836912)

[Implementation choices 12](#_Toc464836913)

[General security 12](#_Toc464836914)

[Registration 12](#_Toc464836915)

[Login 14](#_Toc464836916)

[Change password 14](#_Toc464836917)

[Upload avatar 14](#_Toc464836918)

[Add comment 15](#_Toc464836919)

[Administration 15](#_Toc464836920)

[Indexing and directory listening 15](#_Toc464836921)

[Conclusion 16](#_Toc464836922)

[References 16](#_Toc464836923)

[Appendices 17](#_Toc464836924)

[I. Personal evaluations and peer assessments 17](#_Toc464836925)

# Introduction

In this document we will be presenting our secure design for a custom web application. We will be explaining what kind of techniques we have applied to our web application design and how did we implement them.

We conducted our development in three different main stages. During the first stage we created a simple vulnerable web application, just enough to know what our main functionalities are. After creating this application we started researching into possible threads and vulnerabilities that applied to our concrete implementation. We then updated our implementation to a beta version and gave it to another group for pen testing. When we received their pen-test report, we took all the vulnerabilities they found and updated our application to its final version.

In this document we will be discussing three main points.

First we will concentrate on the main idea behind the application is – the functionalities and overall model. We won’t concentrate much on the code, but more on the ideas and architecture.

In the next parts we will be focusing on our research findings and how they apply to our implementation. We analyze the possible threads and risks and how we decided to best deal with them.

# Web application functionalities

The main purpose of this project is to develop a simple web application with some secure techniques. The web application is developed with HTML/CSS Bootstrap, JavaScript/JQuery for the front side and PHP/MySQLi for the back office. Our web application is called “User-Blog”, which basically allows users to register, update personal profile and leave comments to each other.

## User groups

|  |  |
| --- | --- |
| Role | Description |
| Visitor | Can only view comments, but are not allowed to post anything on the webpage. They are allowed to register an account and login with it as registered users. |
| Registered user | Are allowed to view and post comments to the board. They are also allowed to upload a profile picture and change their passwords if they wished. |
| Administrator | Are allowed to view the administration page and delete user comments as well as ban users.  They cannot tamper with the comments or account of administrators with higher access level however. |

## Use case scenarios

|  |  |
| --- | --- |
| Use case ID | Use case name |
| 1 | Register |
| 2 | Log in |
| 3 | Update avatar |
| 4 | Comment |
| 5 | Change password |
| 6 | Administrator bans user |
| 7 | Administrator deletes comment |

### Register

**Goal:** User register a new account.

**Actor:** User

**Precondition:** User is currently in the homepage.

**Main Success Scenario:**

1. User input username.
2. User input password.
3. User input password again.
4. User input email address.
5. User clicks the register button.
6. Application shows that the user has successfully registered.

**Extensions:**

2.1 The password does not satisfy the following criteria: “Password must be at least 8 characters long and must contain at least 1 upper case, 1 lower case and 1 digit”

3.1 The repeat password is not the same as the previous password inputted in the second step.

4.1 Email address is not with valid format.

5.1 Username already exist in the database.

**Post condition:**

The user successfully registered a new account.

### Log in

**Goal:** User logs in with his/her account.

**Actor:** User

**Precondition:** User is currently in the homepage and he/she has a valid account.

**Main Success Scenario:**

1. User input username.
2. User input password.
3. User clicks the login button.
4. Application shows that the user has successfully logged in.

**Extensions:**

3.1 Wrong username and password combination.

**Post condition:**

The user successfully logged in.

### Update avatar

**Goal:** User changes his/her avatar.

**Actor:** User

**Precondition:** User has already logged in.

**Main Success Scenario:**

1. User click “browse” button.
2. User choose a file.
3. User clicks the Submit button.
4. Application shows that the user has successfully updated the avatar.

**Extensions:**

3.1 User choose a file whose format is NOT “jpg”, “jpeg”, “gif” or “png”.

**Post condition:**

The user successfully updated the avatar.

### Comment

**Goal:** User sends some comments.

**Actor:** User

**Precondition:** User has already logged in.

**Main Success Scenario:**

1. User types some comments.
2. User clicks the “submit” button.
3. The comment is successfully sent.

**Extensions:**

None.

**Post condition:**

The comment is successfully sent.

### Change password

**Goal:** User changes the account password.

**Actor:** User

**Precondition:** User has already logged in.

**Main Success Scenario:**

1. User input old password.
2. User input the new password.
3. User input the new password again.
4. User clicks “change password” button.
5. Application shows that the user has successfully updated the password.

**Extensions:**

2.1 The password does not satisfy the following criteria: “Password must be at least 8 characters long and must contain at least 1 upper case, 1 lower case and 1 digit”

* 1. Repeat password does not match the new password inputted in the second step.

4.1 Old password is not correct.

**Post condition:**

The user successfully changed the account password.

### Administrator bans user

**Goal:** Administer deactivate a normal user’s account.

**Actor:** User

**Precondition:** User is logged in as administer.

**Main Success Scenario:**

1. User goes to the “admin” page.
2. User clicks “ban user” button.
3. Application shows that selected user’s account is deactivated.

**Extensions:**

None.

**Post condition:**

The selected account is successfully banned.

### Administrator deletes comment.

**Goal:** Administer deletes some comments.

**Actor:** User

**Precondition:** User is logged in as administer.

**Main Success Scenario:**

1. User goes to the “admin” page.
2. User clicks “delete comment” button.
3. Application shows that selected comment is deleted.

**Extensions:**

None.

**Post condition:**

The user successfully deleted that selected comment.

# Threat Analysis

In this chapter we will explain all the research we have conducted on possible threads and risks for our implementation. Furthermore we will also propose possible solutions for preventing and dealing with such vulnerabilities.

## CIA Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Asset | Confidentiality | Privacy | Integrity | Availability |
| User id & Password | High | High | High | High |
| Pictures | Low | Medium | None | High |
| Change Password | High | High | High | High |
| Comments | Low | Medium | None | Medium |

**Description:** These are the four assets for our website. The passwords are highly valued because they are the means for getting onto the website and changing the content. The pictures are meant to be publicly viewable and therefore have low confidentiality and medium privacy. The availability for pictures is high. The Change Password is meant to be the private part for the users and therefore have high confidentiality, privacy, integrity and availability. The Comments are meant to be send to other users publicly and therefore it has low confidentiality, medium privacy and high availability.

## Permission scheme

|  |  |  |  |
| --- | --- | --- | --- |
| Data Group | Owner | Other Members | Web Administrator |
| Username and Password | CRUD | - | CRUD |
| Pictures | CRD | R | CRUD |
| Change Password | CRU | - | CUD |
| Comments | CRD | CR | CRUD |

Note: CRUD (Create, Read, Update, Delete)

## Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Threat | Risk | Impact | Needed Security level Risk impact |
| Account Theft | High | High | High |
| Unauthorized View of Pictures | Medium | Medium | Medium |
| Malicious file upload | High | High | HIgh |
| Unauthorized view of admin page | Medium | High | Medium |
| Unauthorized Change Password | High | High | High |
| Unauthorized View of Comment | Medium | High | High |

### Brute force and dictionary attacks

As an application that mainly concerns itself with user integrity, one of the first attack types we should look through are brute force and dictionary attacks. These are the two most common ways of guessing a user’s password – dictionary attacks do so by using files containing common passwords and phrases to guess while a brute force simply checks all possible combination of characters up to a given length.

Since there is no way to prevent such attacks the only option we have is to make it as hard as possible for the intruders.

Possible ways of diminishing the effectiveness of such attacks are:

* Salt passwords, by appending randomly generated long strings to the password.
* Hash the password with slow hashing functions.
* Adding a secret key.
* Force users to create long and complex passwords.
* Timeout login function after specific number of failed attempts.

### SQL injections

One of the most common ways to attack the server database. These attack make use of user input fields to send custom queries to the database, which can then return sensitive information that should not be accessible by the certain user. These kinds of attacks usually happen when SQL statements are constructed dynamically and when strings are concatenated to form the statements.

There are a number of ways to prevent SQL injection attacks:

* Making use of “prepared statements”.
* Parametrize queries.
* Validating the input on the backend side.
* Encrypt sensitive data.

$query = $con->prepare("SELECT username,password,salt, avatar, accesslevel FROM users WHERE username = ? LIMIT 1");

$query ->bind\_param('s',$userName);

$query->execute();

$result = $query->get\_result();

### Cross site scripting (XSS)

Cross site scripting is a vulnerability that allows the attacker to execute malicious scripts through a client-side code injection. Since our application’s main functionality is the posting of comments on the website, our main concern would be the postings of malicious files and sites. (Stored XSS)

Possible solutions for XSS attacks:

* Using htmlentities() method which will do the code conversion.
* htmlspecialchars().
* Using a whitelist of approved html elements.

### Cross-site request forgery (CSRF)

In a cross-site request forgery attack, the hacker usually exploits how the web application manages user authentication. These attacks involve the victim to send a request without their knowledge or consent. For these sorts of attacks to work, the victim has to be authenticated on the targeted site, or in simpler words, to be logged in.

Our site has a password change functionality which could be a target for CSRF attacks.

Ways for us to prevent such things from happening:

* Timeouts for inactive sessions – make sure the session expires.
* Use unpredictable tokens to validate the request and but cannot be guessed by the attacker.
* Eliminate XSS vulnerabilities on the website.

### File upload

Since we also have an upload functionality for our site we have to make sure that no malicious files could be uploaded to the server.

There are a number of things we could check to prevent any malicious files from being uploaded and executed on our server.

* Check extensions.
* Check file size.
* Rename files when uploaded.
* Make sure final extension when uploading is the correct one, so we could not temper with the data.

### File inclusion

Since our pages are included via an include function and we use the $\_GET[‘page’] function, we have to make sure we can only include our own pages.

Ways to deal with this:

* Make sure that only pages allowed by the server are accessible.
* Don’t show errors when attack fails.

## Misuse cases

**Register/Log in/change password:**



For the register/log in/change password function, we mainly focusing on preventing “SQL injection”.

**Update avatar:**



For the update avatar function, we mainly focusing on preventing “path traversal”.

**Comment:**



For the comment function, we mainly focusing on preventing “cross site scripting”.

Additionally, indexing is prevented for all pages and uploaded files. In other words, users will get “403 access denied” error if he/she tries to access some page or uploaded file by URL. In this case, hackers will not be able to access any file by “path traversal”.

# Implementation choices

## General security

We decided to make use of already available security and stability through the use of PHP 7.0 instead of its predecessors and MySQLi instead of MySQL.

Aside from the speed boosts we gain from using the newest version of PHP, we also analyzed how it contributes to the stability and security of our implementation:

* unserialize() allows for selectively allowable classes.
* session\_start() is now optionally able to override session configuration directives in php.ini.
* two new functions have been introduced to generate cryptographically secure integers and strings.
* For greater stability, uniform variable syntax has been introduced.

The reason we chose to use MySQLi instead of MySQL are pretty straightforward:

* Gives us a way to parametrize queries.
* Support for “prepared statements” – safer way of sending queries to the database and protects from SQL injections.
* Support for multiple statements.
* Supports transactions.
* Object oriented.
* MySQLi supports most of the MySQL features and since PHP 5.5 it has been deprecated. (As a matter of fact it is even removed as of PHP 7.0 so we couldn’t really use it anyway.)

***Note: Frankly PDO might have been a more flexible and portable way of executing SQL queries but we did not have time to change everything.***

Additionally we researched ways to setup our server as securely as possible **[3].**

* Disabling error shows so that attackers can’t find out at what lines their attacks are failing. (Log them instead.)
* Turn off magic\_quotes\_gpc() because it’s not very effective and it’s better to use mysqli\_escape\_string() instead.
* Restrict file upload sizes.
* Restrict post sizes.

## Registration

We decided to restrict the inputs allowed by the user as much as possible. We created two layers of checking – client side and server side. The first part is only to give the user an idea of how his input should look like and give him general feedback during the process of filling in the form. This is done by some simple javascript validation functions. Since these scripts are quite easy to bypass we also added the second layer of checking.

The second part is the actual check before sending in the query to be processed. Here we create a number of functions to check each field using the same rules as in the user side.

For the purpose of checking the fields we first sanitize them, then check them with regular expressions to see if any of them contain unwanted characters or are formatted correctly.

* Check length of name and contained characters.
* Force user to create a complex password.
* Asks the user for a valid formatted email.
* Check password matches.

**function** validatePassword($pass1, $pass2)

{

**global** $passError;

$reg = "/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])[0-9a-zA-Z]{8,}$/";

**if**(**empty**($pass1))

{

$passError = "Password required.";

**return** **false**;

}**else** **if**(!preg\_match($reg, $pass1))

{

$passError = "Password must be at least 8 characters long and must contain at least 1 upper case, 1 lower case and 1 digit";

**return** **false**;

}**else** **if**(**empty**($pass2))

{

$passError = "Please retype password.";

**return** **false**;

}**else** **if**($pass1 != $pass2)

{

$passError = "Passwords do not match.";

**return** **false**;

}

**return** **true**;

}

If all the input is correct then we create a randomly generated salt, hash the password and are ready execute the query. Here we make use of the prepared statements available to MySQLi and we parametrize the query.

**if**(!validateName($username))

{

$message = $userError;

}**else** **if**(!validatePassword($password1, $password2))

{

$message = $passError;

}**else** **if**(!validateEmail($email))

{

$message = $emailError;

}

**else**

$salt = createSalt();

$password = hash('sha256', $salt . $hash);

$query = $con->prepare("INSERT INTO users ( username, password, email, salt )VALUES ( ?, ?, ?, ? )");

$query ->bind\_param('saaa',$username,$password,$email,$salt);

$query->execute();

$message = "Succesfully registered";

## Login

For the login we generally make sure to prevent SQL injections, which we decided to do using prepared statements and parametrized queries. We also make sure again that the format if the username and password match the regular expressions we defined inside the registration form.

$userName = mysqli\_real\_escape\_string($con,$userName);

$password = mysqli\_real\_escape\_string($con,$password);

$regUser = "/^[a-zA-Z0-9\_-]{3,25}$/";

$regPassword = "/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])[0-9a-zA-Z]{8,}$/";

**if**(!preg\_match($regUser, $userName) || !preg\_match($regPassword, $password))

{

$message = "Wrong credidentials.";

}

**else**{

$query = $con->prepare("SELECT username,password,salt, avatar, accesslevel FROM users WHERE username = ? LIMIT 1");

$query ->bind\_param('s',$userName);

$query->execute();

$result = $query->get\_result();

## Change password

The change password could be changed only if the user knows his old password. Again we make use of similar techniques as we did in the registration and the login forms.

* Use prepared statements.
* Validate input with regular expressions.
* Validate old password.

## Upload avatar

Since attackers could try to upload malicious files and scripts, and thus gain access to everything that is on the server, it is quite important to make sure we can prevent such things from happening. To begin with we have two layers of checks for the file being uploaded – the first is to check the extension, and the second one is to rename the file with a custom generated name based on a timestamp. (This also prevents files with the same name to be uploaded) While the first check can be tampered with (say Mozilla Firefox’s tamper data add-on). Say we decided to rename the file .php instead of .jpg, the next check completely ignores that request.

$allowed=**array**('jpg','jpeg','gif','png');

$fileName=($\_FILES['Photo']['name']);

$file\_exten= **explode**('.',$fileName);

$file\_ext = **strtolower**(end($file\_exten));

$tmpName = $\_FILES['Photo']['tmp\_name'];

**if**(in\_array($file\_ext, $allowed))

{

//upload file

change\_profile\_image($username,$tmpName,$file\_ext);

This function makes a call to another one while preserving the first given extension in mind, not the one we specified in the tampered data.

**function** change\_profile\_image($username,$temp,$file\_ext)

{

$con = connectDatabase();

$avatarname=substr(**md5**(**time**()),0,10).'.'.$file\_ext;

$avatar = "Images/Profiles/".$avatarname;

$filepath=IMAGES.$avatarname;

move\_uploaded\_file($temp,$filepath);

As you can see, nothing is left from the original name and the extension that is passed down to the image is the original specified one that was previously checked.

An additional check that could be made is in connection with the size of the file. However this file upload limit could be set in the server configurations so there is no need to make additional checks in the code.

## Add comment

Adding the comments is again mainly secured by sanitizing the input and again making use of the prepared statements.

## Administration

We make use of an additional session that is set if a user has the sufficient access level. Then he could access the special administration page from where he could delete comments and ban users.

These actions cannot be accessed through the url as normal users and visitors will not be allowed to visit the page itself.

## Indexing and directory listening

During the pen testing, the group evaluating the security of our website managed to gain access to our directories and files by indexing, hence knowing all file names and how they are stored on the server.

We managed to prevent that by restricting access to all these files by updating our .htaccess file on the server side with the following line:

Options -Indexes

# Conclusion

In conclusion we could say that due to the first-hand experience of trying to secure an application and trying to exploit vulnerabilities on another gave us quite a lot of insight in now web security works. Our main objective was to analyze what sort of attacks the application was susceptible to, research how to prevent and handle such these threads and find ways to apply learnt techniques to our specific implementation.

Due to time constrains we could not add everything we first wished to try, but despite that we can assume that the application is secure not only from beginner kiddie scripters but also from more experienced attackers.

Our main focus was to restrict the user input as much as possible and prevent him from accessing files and directories he was not authorized to view. We also put a lot of thought into preventing the user of uploading malicious script that he could later execute on our website.

All in all, we are happy with the final product of our work and would try to improve our security skills in the future, whether with this project or with another.

# References

* **[1]** Salted Password Hashing – Doing it Right, visited on 12.10.2016

<https://crackstation.net/hashing-security.htm>

* **[2]** How to prevent SQL injections.

<http://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php>

* **[3]** 25 PHP Security Best Practices for Sys Admins

<http://www.cyberciti.biz/tips/php-security-best-practices-tutorial.html>

* **[4]** Cross-site request forgery (CSRF) Prevention Cheat Sheet

<https://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet>

* **[5]** PHP new features – Manual

<http://php.net/manual/en/migration70.new-features.php>

# Appendices

## Personal evaluations and peer assessments

### Jiaqi Ni

This project is the first time for me designing a secured web application. During the development of this project, it was actually quite easy to design and build the basic functionalities, although there were still some minor exception, which were all solved without taking too much time. However, securing the program is new for us and some unexpected exceptions did show up and take us quite a lot of time. Panties of research was conducted to find out the possible solutions, and fortunately, every insecure points revealed were successfully fixed. Meanwhile, we also learned a lot of new stuff during the process of solving these exceptions, which could definitely be highly helpful for further developer life.

In summary, this project is quite successful. Through this project, I have a better overview about how to design, build and secure a program, a better knowledge about how to work in as team, etc. Generally speaking, I gained a lot of valuable experience for secure programming, which could be pretty helpful for my further career.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group** | **Products: total of column is 100%** | | | **Process: Mark (1- 10)** | | | | |
| **Student name** | **Research** | **Development of web application / hacking attacks** | **Report writing** | **Pro-Active  attitude** | **Amount of work done** | **Quality of work done** | **Communication within group** | **Total appreciation as team member** |
| Jiaqi Ni | 16% | 20% | 18% | 8 | 8 | 8 | 7.5 | 8 |
| Kalina Petrova | 20% | 25% | 24% | 8 | 9 | 9 | 8 | 9 |
| Mengchuan Liu | 16% | 20% | 10% | 8 | 8 | 8 | 8 | 8 |
| Jiefan Lin | 16% | 15% | 10% | 8 | 8 | 7 | 7 | 7.5 |
| Jianfei Feng | 16% | 10% | 16% | 7 | 7 | 7 | 7 | 7.5 |
| Ishant Upadhyay | 16% | 10% | 22% | 7 | 8 | 8 | 7.5 | 7.5 |

### Kalina Petrova

This project was quite the fun and interesting experience for me. As a somewhat paranoid person when it comes to account integrity, I have always been interested in the security of my personal information. I have had personal experiences with account theft and know how tedious and nerve wrecking it is to gain back control over your account. This course helped me understand more in-depth how my security might be compromised and what types of techniques are usually used to access unauthorized information.

Researching and applying techniques to prevent certain types of attacks was the hardest part but also the most interesting one. I also had quite a lot of fun trying to hack another’s web application and break its structure.

I have built a number of websites before and looking back now, I know how vulnerable and susceptible they are to different types of attacks. I really feel like I have improved as a developer and that I could apply this new acquired knowledge to my future projects.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group** | **Products: total of column is 100%** | | | **Process: Mark (1- 10)** | | | | |
| **Student name** | **Research** | **Development of web application / hacking attacks** | **Report writing** | **Pro-Active  attitude** | **Amount of work done** | **Quality of work done** | **Communication within group** | **Total appreciation as team member** |
| Jiaqi Ni | 16% | 20% | 18% | 9 | 9 | 8.5 | 8.5 | 8.5 |
| Kalina Petrova | 20% | 25% | 24% | 8 | 9 | 9 | 8 | 8.5 |
| Mengchuan Liu | 16% | 20% | 10% | 8 | 8 | 8 | 8 | 8 |
| Jiefan Lin | 16% | 15% | 10% | 8 | 8 | 7 | 7 | 7.5 |
| Jianfei Feng | 16% | 10% | 16% | 7 | 7 | 7 | 7 | 7.5 |
| Ishant Upadhyay | 16% | 10% | 22% | 7 | 8 | 8 | 7.5 | 8 |

### Mengchua Liu

The project was a big challenge for everyone from the group and our ambition to make it perfect was driving us forward. The cooperation in the group was perfect, we rarely had any disagreements, we rather discussed and the problem and made solutions.

This last several weeks, I really worked on improving my security implementation skills. I had a much better understanding of team dynamics and how to contribute more effectively in the team development.

I struggled a bit in dealing with my tasks. My colleagues, especially Kalina, tried a lot to help me. She explains me how to start my tasks and helps me to fix my errors. I did learn not only from my tasks, but also from my colleagues’ part. And in the end, we manage to make a good project.

I learned how to prevent sql injection, xss, csrf and pt problems. And I implemented the login - version 1( the non-secure version) and the administration functions.

Overall we have a great team and finish our own beautiful project. But it is better to come up some new projects. As a student, we always have more willing to work on a new project.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group** | **Products: total of column is 100%** | | | **Process: Mark (1- 10)** | | | | |
| **Student name** | **Research** | **Development of web application / hacking attacks** | **Report writing** | **Pro-Active  attitude** | **Amount of work done** | **Quality of work done** | **Communication within group** | **Total appreciation as team member** |
| Jiaqi Ni | 21% | 22% | 12% | 8 | 8 | 8 | 7.5 | 8 |
| Kalina Petrova | 28% | 28% | 20% | 8 | 9 | 9 | 8 | 9 |
| Mengchuan Liu | 21% | 26% | 12% | 8 | 8 | 8 | 8 | 8 |
| Jiefan Lin | 10% | 7% | 12% | 8 | 8 | 7 | 7 | 7.5 |
| Jianfei Feng | 10% | 10% | 22% | 7 | 7 | 7 | 7 | 7 |
| Ishant Upadhyay | 10% | 7% | 22% | 7 | 8 | 8 | 7.5 | 7.5 |

### Jiefan Lin

Internet security is a complex problem. Internet problem-solving plan combines the secure administration with the secure technology. So the website must be identified effectively and be designed and controlled reasonably. A lot of people think that, hacker is the computer enthusiasts who perform bad thing. In this courses we make the website from the hacker’s view, which is really interesting. I have learned a great deal from that and have been greatly improved in programming. They way of learning I will apply to future study. Not merely to do your work, but also to show results to the audience and do a better job of communicating the nature and impact of our work. To succeed as a programmer, I simply have to quickly recognize common pitfalls. Much of the challenge is to cope with unceasing change. The confidentiality and integrity are required. Privacy and the security of information are important concerns in a project. This course reminds me that in the future program I will be in pursuit of code quality: study code mechanism, test frameworks, and writing quality-focused code.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group** | **Products: total of column is 100%** | | | **Process: Mark (1- 10)** | | | | |
| **Student name** | **Research** | **Development of web application / hacking attacks** | **Report writing** | **Pro-Active  attitude** | **Amount of work done** | **Quality of work done** | **Communication within group** | **Total appreciation as team member** |
| Jiaqi Ni | 16% | 20% | 12% | 8 | 8 | 8 | 7.5 | 8 |
| Kalina Petrova | 20% | 20% | 20% | 8 | 9 | 9 | 8 | 9 |
| Mengchuan Liu | 16% | 20% | 12% | 8 | 8 | 8 | 8 | 8 |
| Jiefan Lin | 16% | 14% | 12% | 8 | 8 | 7 | 7.5 | 7.5 |
| Jianfei Feng | 16% | 13% | 22% | 7.5 | 7 | 7 | 7.5 | 7.5 |
| Ishant Upadhyay | 16% | 13% | 22% | 7.5 | 8 | 8 | 7.5 | 7.5 |

### Jianfei Feng

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group** | **Products: total of column is 100%** | | | **Process: Mark (1- 10)** | | | | |
| **Student name** | **Research** | **Development of web application / hacking attacks** | **Report writing** | **Pro-Active  attitude** | **Amount of work done** | **Quality of work done** | **Communication within group** | **Total appreciation as team member** |
| Jiaqi Ni | 16% | 20% | 18% | 9 | 8 | 8 | 7.5 | 8 |
| Kalina Petrova | 20% | 25% | 24% | 9.5 | 9 | 9 | 8 | 9.5 |
| Mengchuan Liu | 16% | 20% | 10% | 8 | 8.5 | 8 | 8.5 | 8.5 |
| Jiefan Lin | 16% | 15% | 10% | 8 | 8 | 7 | 7 | 7.5 |
| Jianfei Feng | 16% | 10% | 16% | 7 | 7 | 7 | 7 | 7 |
| Ishant Upadhyay | 16% | 10% | 22% | 7 | 8 | 8 | 7.5 | 7.5 |

It was a short but hard project for me during this block. During this project, there are a lot of new things need to learn, and also this is my first time to work on security programming. At the beginning of this project, I have no idea how these code work for defeat hack, and how to implement it. Luckily, with the help of team members, the application was working properly at last. After this project, I find there are still a lot of stuff which I need to learn, and was glad to know some new security technology.

### Ishant Upadhyay

During my course of study of secure programming, I got change to learn about what is secure programming, why it is important. During my period of learning, I learned about different attacking techniques for hacking the application such as Sql injection, XSS, XSRF, path traversal, file upload. I also learned about how I can make our application safe from these attacks. Previously, I know how to build application but I have not given much important to secure it but after this course, I learned that it also very important to make the application secure a well.

It was a wonderful experience working with my talented group members. They all were very cooperative, friendly and helpful. We always used to discuss together if there were any problems and try to solve it with mutual decision. I would especially like to Kalina and Jiaqi for their continuous help and support.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group** | **Products: total of column is 100%** | | | **Process: Mark (1- 10)** | | | | |
| **Student name** | **Research** | **Development of web application / hacking attacks** | **Report writing** | **Pro-Active  attitude** | **Amount of work done** | **Quality of work done** | **Communication within group** | **Total appreciation as team member** |
| Jiaqi Ni | 16% | 20% | 18% | 8 | 8 | 8 | 7.5 | 8 |
| Kalina Petrova | 20% | 25% | 24% | 8 | 9 | 9 | 8 | 9 |
| Mengchuan Liu | 16% | 20% | 10% | 8 | 8 | 8 | 8 | 8 |
| Jiefan Lin | 16% | 15% | 10% | 8 | 8 | 7 | 7 | 7.5 |
| Jianfei Feng | 16% | 10% | 16% | 7 | 7 | 7 | 7 | 7.5 |
| Ishant Upadhyay | 16% | 10% | 22% | 7 | 8 | 8 | 7.5 | 7.5 |

In conclusion, I can say that I had a wonderful experience learning secure programming. We had lots of fun while learning it. It was very helpful and I would always try to implement what I have learnt from it to my future projects.