Website with backend functionality for the church Iglesia Rey De Justicia
-Giovanni Moncibaez

Security Assessment Report

Version N.1

April 29, 2023

Security Assessment – Website: Rey De Justicia

Table of Contents

1. Summary

The goal in creating this website was to provide a needed service to an up-and-coming church named Rey De Justicia. This website's purpose is to connect the congregation with the information they need, as well as invite new people to visit the church.

1. Assessment Scope

Tools used in the creation and testing of the website:

- PHP Storm: PHP IDE with front end technology included.
- XAMPP: Open-source Web Server Solution Pack
- OSWAP ZAP: Test website.
- GitHub: Code Hosting platform
- Google / Edge: Browsers used to test how the website looked and functioned.
- OSs used: Windows.

PHP storm is an excellent IDE that can help with full stack development due to its ability to work with: HTML, CSS, JS, and PHP. XAMPP had a server, MySQL, and php.exe all download into one bundle that made it easy to run a server locally and test the function of my website. OSWAP ZAP is a tool I learned about in TryHackMe that I am using to try and find vulnerabilities in my website. GitHub was a perfect place to share my code with family members that had experience in Backend development that allowed them to check for common security vulnerabilities inside of my website. Google and Edge were used to test out the functionality of the website for different user experiences.
Limitations:

- Finances to pay for a domain, webhost, online DB, security testing, etc.
- Only one person is working on the creation of the website.
- Time constraints
- Lack of knowledge

A great limitation to the project would be Finance. Currently, I'm working and creating this website for free. Since money is very limited currently, being able to afford a webhost, domain, and other elements in web development are difficult. Soon, the church will be able to fund my project by buying a domain and a webhost, but currently they cannot. This greatly effects being able to test my website for vulnerabilities. I would like to pay for Burp Suite Vulnerability scanner but currently using OSWAP ZAP. Another limitation is that I'm the only one that is developing the website. Going to school, doing other assignments, can make changes go slower. Lastly, my lack of knowledge is a limitation, as I am currently learning how to develop and test my website. It is a great experience, but it can leave my website to many vulnerabilities.

2. Summary of Findings

Of the findings discovered during our assessment, 5 were considered High risks, 2 Moderate risks, and 2 Low risks. The SWOT used for planning the assessment are broken down as shown in Figure 1.

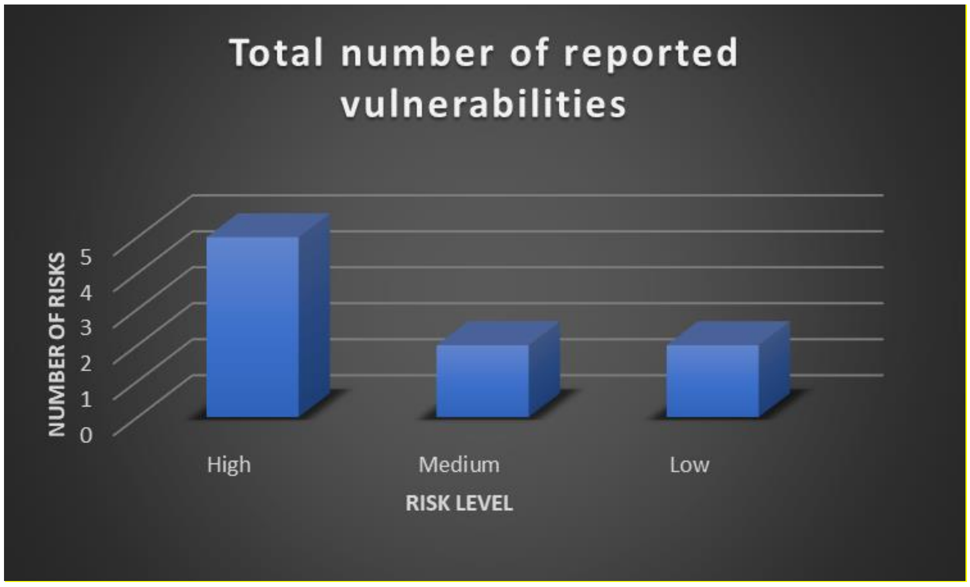
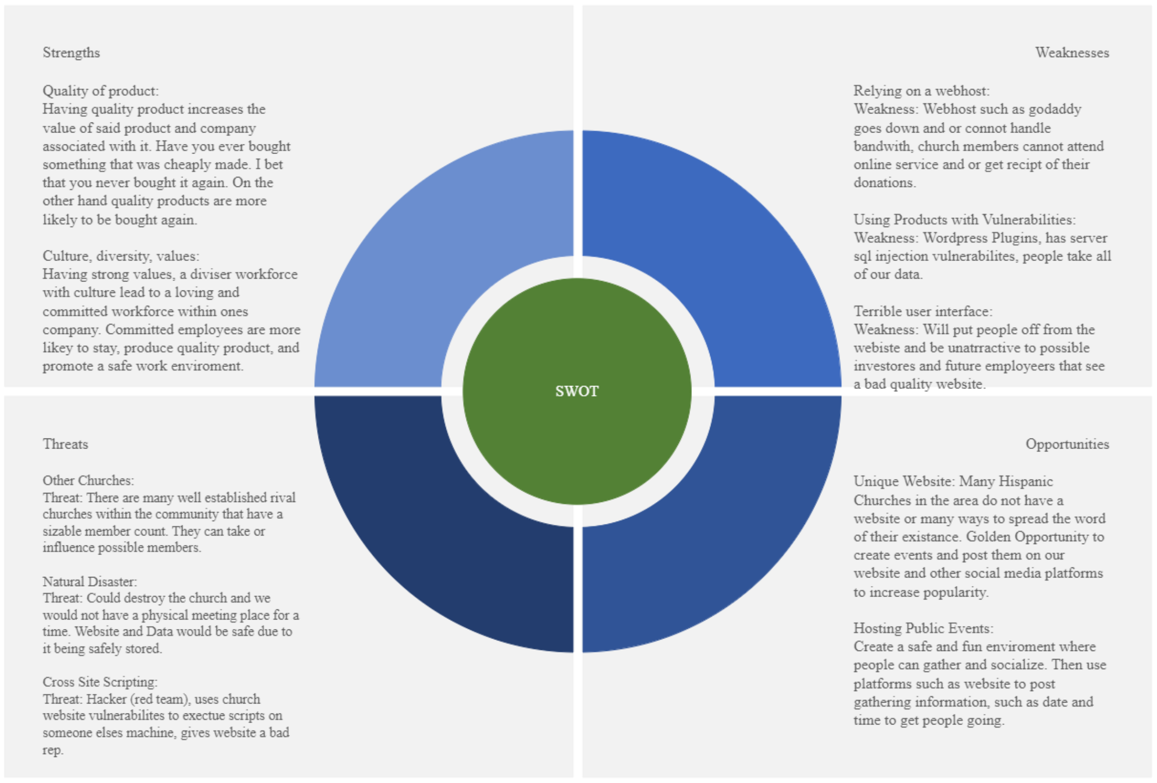


Figure 1. Findings by Risk Level

The ranking of the categories of risk level were determined based upon how much of a risk it was and how long it would take to fix. More information can be found at 3.1.1 Risk Level Assessment Table.



Strengths

Quality of product:
Having quality product increases the value of said product and company associated with it. Have you ever bought something that was cheaply made. I bet that you never bought it again. On the other hand quality products are more likely to be bought again.

Culture, diversity, values:
Having strong values, a diviser workforce with culture lead to a loving and committed workforce within ones company. Committed employees are more likey to stay, produce quality product, and promote a safe work enviroment.

Weaknesses

Relying on a webhost:
Weakness: Webhost such as godaddy goes down and or connot handle bandwith, church members cannot attend online service and or get recipt of their donations.

Using Products with Vulnerabilities:
Weakness: Wordpress Plugins, has server sql injection vulnerabilites, people take all of our data.

Terrible user interface:
Weakness: Will put people off from the webiste and be unatrractive to possible investores and future employeers that see a bad quality website.

Threats

Other Churches:
Threat: There are many well established rival churches within the community that have a sizable member count. They can take or influence possible members.

Natural Disaster:
Threat: Could destroy the church and we would not have a physical meeting place for a time. Website and Data would be safe due to it being safely stored.

Cross Site Scripting:
Threat: Hacker (red team), uses church website vulnerabilites to exectue scripts on someone elses machine, gives website a bad rep.

Opportunities

Unique Website: Many Hispanic Churches in the area do not have a website or many ways to spread the word of their existance. Golden Opportunity to create events and post them on our website and other social media platforms to increase popularity.

Hosting Public Events:
Create a safe and fun enviroment where people can gather and socialize. Then use platforms such as website to post gathering information, such as date and time to get people going.

Figure 2. SWOT

Issues that occurred during the development were natural disaster, cross site scripting, non-hashed password, and a bad user interface. These issues were found by using white and black box testing. This would help me find and implement new features to the website.

3. Summary of Recommendations

Changes from the first iteration of the website include a revamp of the HTML and CSS to make it more user friendly. Implementation of a backend using PHP and a MySQL database. Used prepared statements and password encryption to increase security. Simple sign in feature that allows users to create an account and get their donations put within the database for future use. The use of OneDrive and a flash drive to say all the files are also in use.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Ensure that the system follows regulations that ensure the safety of user data.
- Determine if the application was securely maintained.
- Discover if the user interface is easy and intuitive.

2. Detailed Findings

List of Vulnerabilities and threats:

Bad authentication: High Risk (Table 1), determined to be a high-risk weakness and vulnerability that can be exploited and cause significant harm to the organization and users by breach of data through a faulty authentication.

Cross Site Scripting: High Risk (Table 1), determined to be a high-risk threat that is perpetuated by inadequate security assessment of backend operations. Significant harm to the organization and users by unvalidated web request in some cases.

SQL injection: High Risk (Table 1), determined to be a high-risk threat that exposes sensitive data by attackers that retrieve and alter data stored on an SQL server.

Ransomware: High Risk (Table 1), determined to be a high-risk threat do to it being a malicious software that prevents the victim from accessing files.

Phishing attacks: High Risk (Table 1), determined to be a high-risk threat, where malicious actors impersonate staff or users into giving sensitive information.

Bad access control: Moderate Risk (Table 1), determined to be a moderate-risk vulnerability, where pervious contributors or unwanted people have access to the website and its database.

Insufficient Monitoring: Moderate Risk (Table 1), determined to be a moderate-risk vulnerability, where if logs are not monitored enough, a threat could go unnoticed.
Out of date plugins: Low Risk (Table 1), determined to be a low-risk vulnerability, where outdated plugins used have vulnerabilities that bad actors can exploit, easily fixed.
Webhost complications: Low Risk (Table 1), determined to be a low-risk vulnerability, where a webhost may have connectivity issues. Constant issues will lead to the pursuit of a better webhost.

3. Recommendations

List of fixes:
Back Up Policies: Easy Fix (Table 2), website's files and database are saved on the OneDrive cloud system and a 12gb SanDisk Flash drive that is stored in a safe within my room. Files are also uploaded onto GitHub.
Internal Actor Threats: Moderate-Easy Fix (Table 2), all changes to the website must go through me. The accounts that will be associated with the webhost will only be known by me and its password will change every 3 months. No one will log into said accounts on any other machine than my own. Account will not be saved on any browser autofill and PC will be destroyed beyond repair once machine is not longer in use.
Post Methods: Easy Fix (Table 2), added post methods to forms and inc.php files that handle sensitive information. This means that if someone were to access the PHP without clicking the form button, they would be redirected back to the home page and not access the PHP file.
Prepared statements: Moderate Fix (Table 2), added prepared statements to my SQL statements in the PHP file to try and prevent SQL injection. Learned from tryhackme SQL injections.
Hashed Passwords: Moderate Fix (Table 2), added function in PHP file that hashes the passwords in the database. Will include photo of code and hashed passwords in database in section 4. Figures and Code.
Escape String: Easy Fix (Table 2), added escape string function to PHP form variables that escapes special characters in a string for use in an SQL query.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment

Each Business Risk has been assigned a Risk Level value of High, Moderate, or Low. The rating is an assessment of the priority with which each Business Risk will be viewed. The definitions in **Error! Reference source not found.** apply to risk level assessment values (based on probability and severity of risk). While Table 2 describes the estimation values used for a risk's "ease-of-fix".

Table 1 - Risk Values

Rating	Definition of Risk Rating
High Risk	Exploitation of technical or procedural vulnerability will cause substantial harm to the business processes. Significant political, financial, and legal damage is likely to result
Moderate Risk	Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity and/or availability of the system, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment to organization.

Rating	Definition of Risk Rating
Low Risk	Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The confidentiality, integrity and availability of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment
Informational	An “Informational” finding, is a risk that has been identified during this assessment which is reassigned to another Major Application (MA) or General Support System (GSS). As these already exist or are handled by a different department, the informational finding will simply be noted as it is not the responsibility of this group to create a Corrective Action Plan.
Observations	An observation risk will need to be “watched” as it may arise as a result of various changes raising it to a higher risk category. However, until and unless the change happens it remains a low risk.

Table 2 - Ease of Fix Definitions

Rating	Definition of Risk Rating
Easy	The corrective action(s) can be completed quickly with minimal resources, and without causing disruption to the system or data
Moderately Difficult	Remediation efforts will likely cause a noticeable service disruption. • A vendor patch or major configuration change may be required to close the vulnerability. • An upgrade to a different version of the software may be required to address the impact severity. • The system may require a reconfiguration to mitigate the threat exposure. • Corrective action may require construction or significant alterations to the way business is undertaken
Very Difficult	The high risk of substantial service disruption makes it impractical to complete the corrective action for mission critical systems without careful scheduling. • An obscure, hard-to-find vendor patch may be required to close the vulnerability. • Significant, time-consuming configuration changes may be required to address the threat exposure or impact severity. • Corrective action requires major construction or redesign of an entire business process
No Known Fix	No known solution to the problem currently exists. The Risk may require the Business Owner to: • Discontinue use of the software or protocol • Isolate the information system within the enterprise, thereby eliminating reliance on the system. In some cases, the vulnerability is due to a design-level flaw that cannot be resolved through the application of vendor patches or the reconfiguration of the system. If the system is critical and must be used to support on-going business functions, no less than quarterly monitoring shall be conducted by the Business Owner, and reviewed by IS Management, to validate that security incidents have not occurred

3.1.2 Tests and Analyses

Performed White and Black Box testing. User interface testing, and analyses of user friendliness and understandability. See in depth in 3.2.

3.1.3 Tools

OSWAP ZAP: used to scan website for problems and vulnerabilities. Found a good number of Alerts. Fixed the highest alerts and now only have medium and low-level alerts. Soon to fix them all as I learn more. See photo in 4. Figures and Code

3.1.3 Methodology in depth

Tools used:

OWASP ZAP and Humans (family and church members)

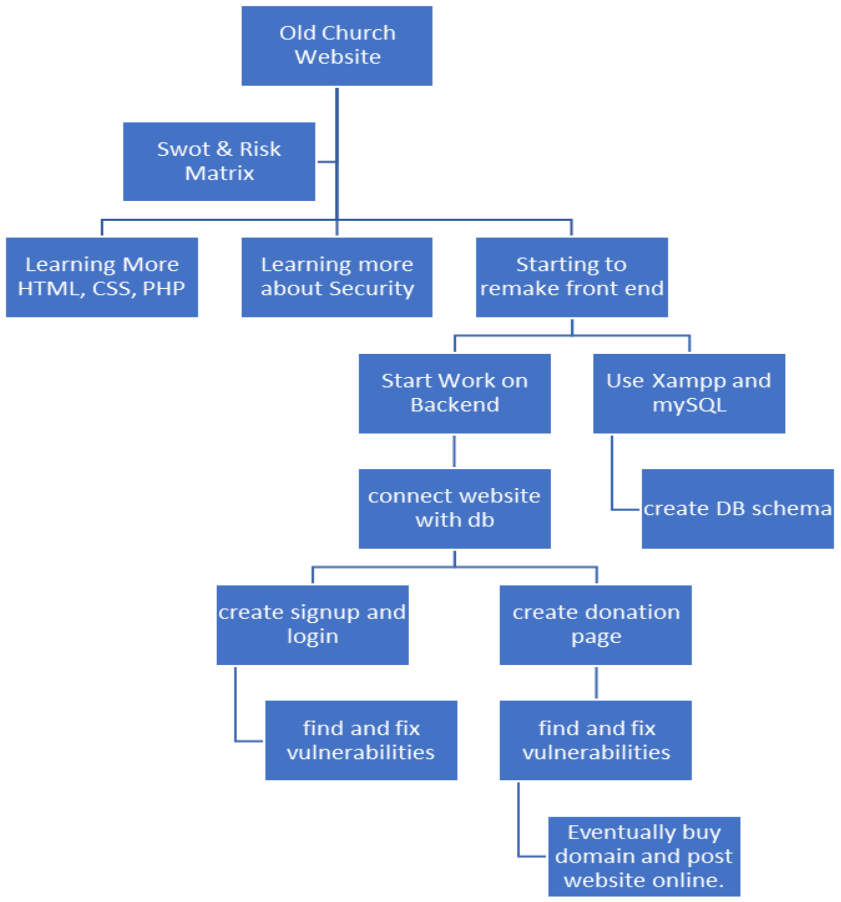
Black Box testing: Conducted Black Box testing by running Website on XAMPP and allowing members of the church to get on my laptop and browse the website and create accounts. They have no knowledge of how the inner workings of website work.

White Box testing: Uncle and Cousins who work in back end, saw code, and tested some common vulnerabilities on the forms inside the website. They had full knowledge of inner workings of website.

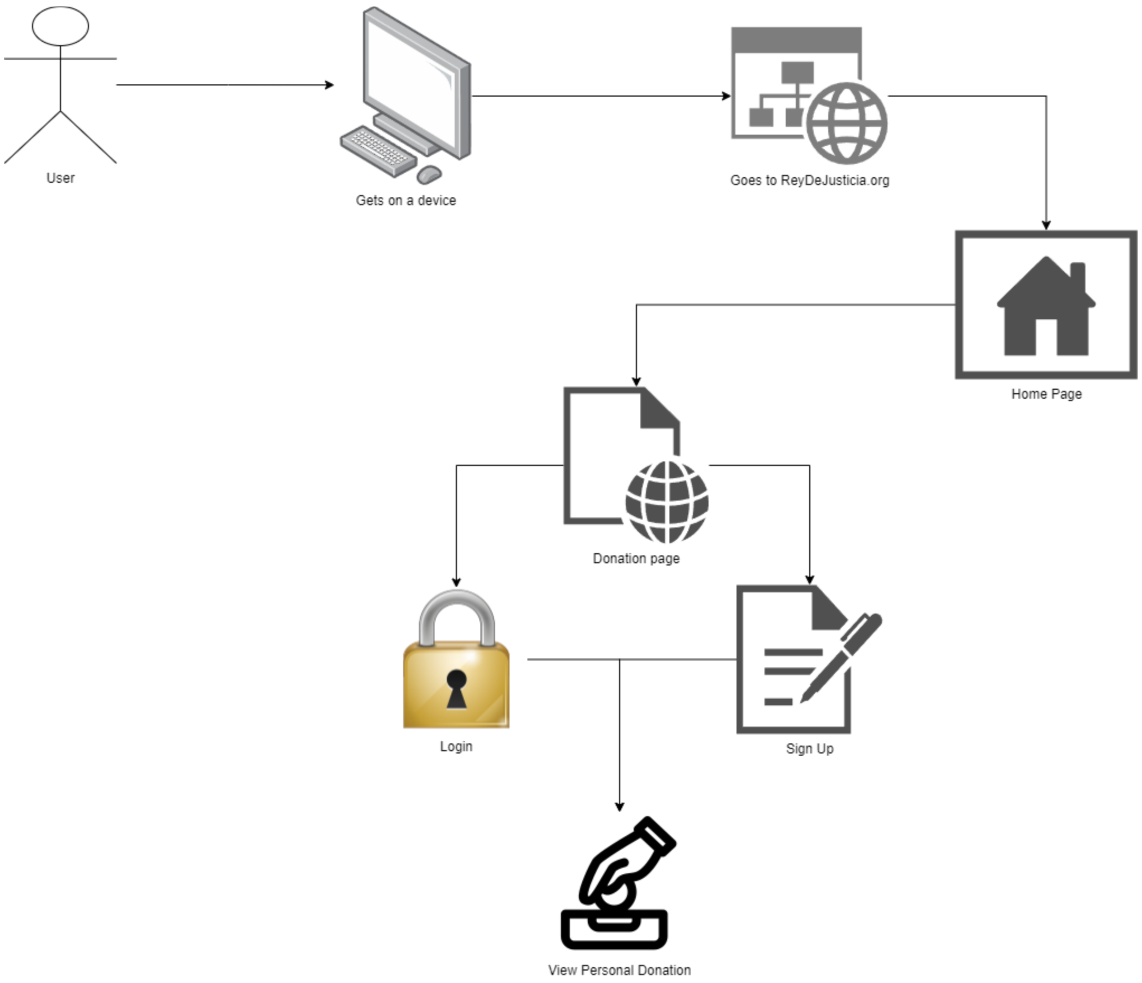
Penetration / Security Scanning: Ran the Website Locally through XAMPP and then ran OWASP ZAP. Did active scan, spider, and ajax spider. Found major and minor problems with website. Focused on fixing major problems, good number of problems that still need to be fixed.

4. Figures and Code

4.1.1 Process of creating the Website and How it works.


Old Church Website
Swot & Risk Matrix
Learning More HTML, CSS, PHP
Learning more about Security
Starting to remake front end
Start Work on Backend
Use Xampp and mySQL
connect website with db
create DB schema
create signup and login
create donation page
find and fix vulnerabilities
find and fix vulnerabilities
Eventually buy domain and post website online.

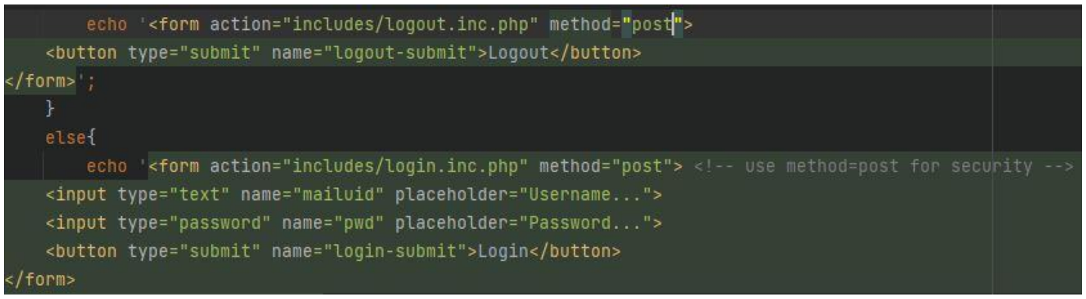
This is a quick timeline of events that lists the major milestones in the website's creation process.



This is a simple Diagram that shows the step by step process a user will take to view their personal donations.

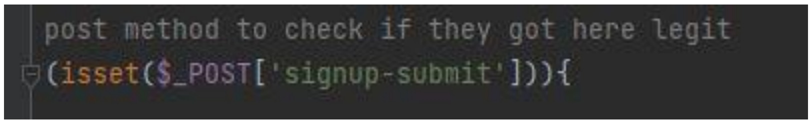
4.1.2 Other figure of code

View my Video on how the Website Looks Running: https://youtu.be/nKKxW5lSqvE



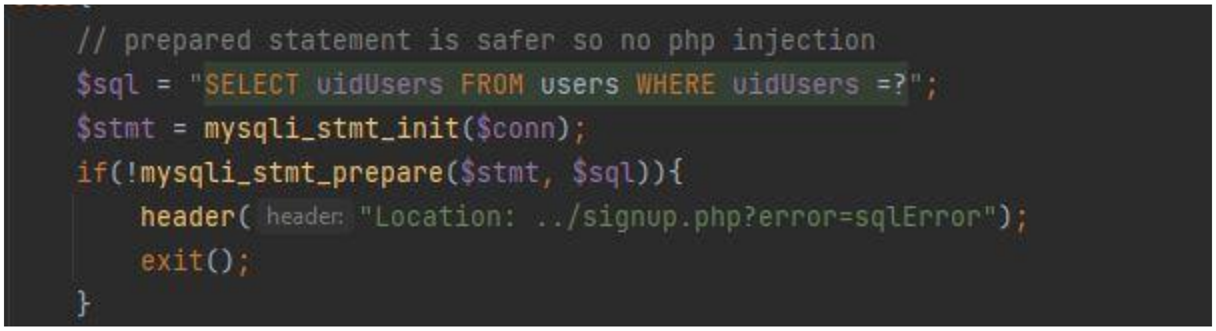
```
        echo '<form action="includes/logout.inc.php" method="post">
    <button type="submit" name="logout-submit">Logout</button>
</form>';
    }
    else{
        echo '<form action="includes/login.inc.php" method="post"> <!-- use method=post for security -->
    <input type="text" name="mailuid" placeholder="Username...">
    <input type="password" name="pwd" placeholder="Password...">
    <button type="submit" name="login-submit">Login</button>
</form>
```

Description of image above: using post method to help encrypt form with HTTP and then using type = “password” so when the password is being entered, it is not show. Good for when people or cameras are around!



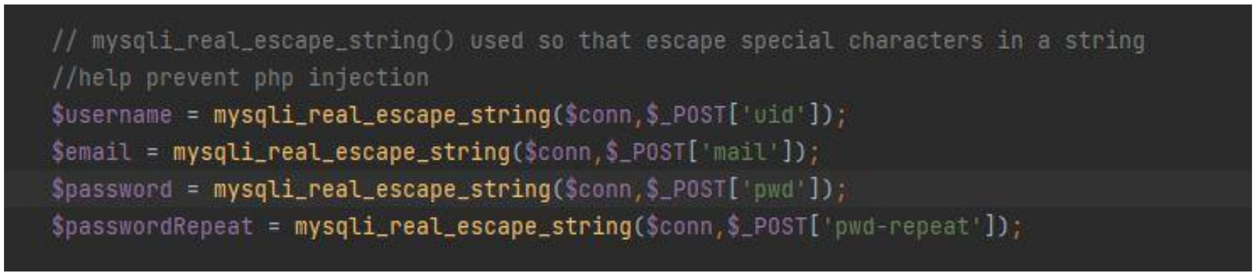
```
post method to check if they got here legit
(isset($_POST['signup-submit'])){
```

Description of image above: used that so this inc.php file is only able to be accessed when it is from the form and not from the URL. This is to fight against cross site scripting.



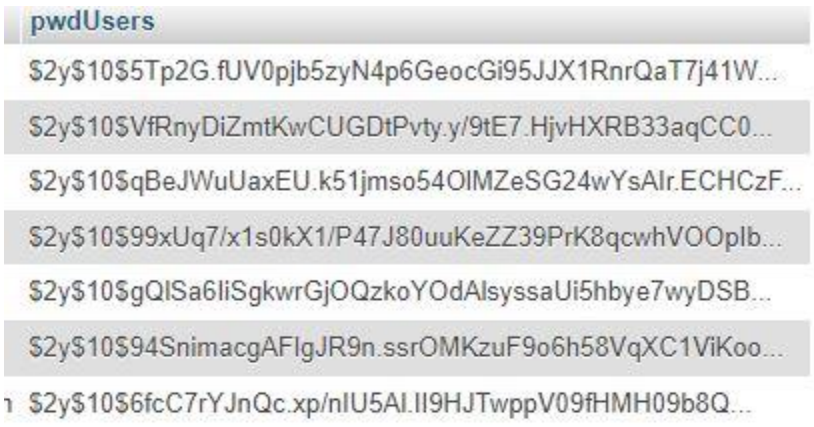
```
    // prepared statement is safer so no php injection
    $sql = "SELECT uidUsers FROM users WHERE uidUsers =?";
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header( header: "Location: ../signup.php?error=sqlError");
        exit();
    }
```

Description of image above: using prepared statements makes sure that whatever the user enters is treated as a string literal in SQL.



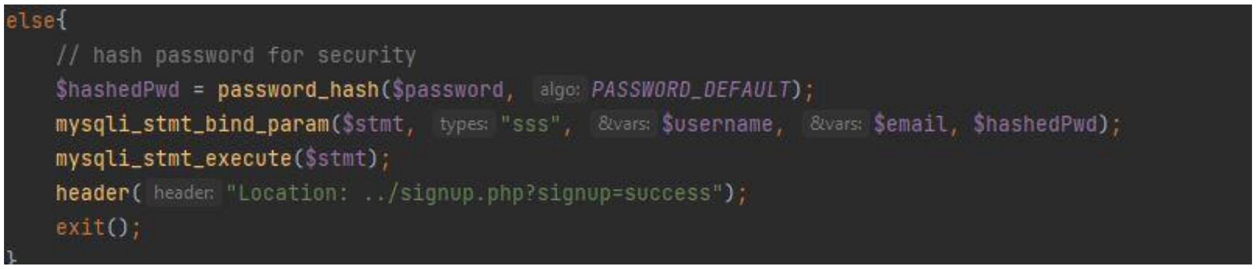
```
// mysqli_real_escape_string() used so that escape special characters in a string
//help prevent php injection
$username = mysqli_real_escape_string($conn,$_POST['uid']);
$email = mysqli_real_escape_string($conn,$_POST['mail']);
$password = mysqli_real_escape_string($conn,$_POST['pwd']);
$passwordRepeat = mysqli_real_escape_string($conn,$_POST['pwd-repeat']);
```

Description of image above: mysqli_real_escape_string escapes special characters in a string for use in an SQL query.



pwdUsers
$2y$10$5Tp2G.fUV0pjb5zyN4p6GeocGi95JJX1RnrQaT7j41W...
$2y$10$VfRnyDiZmtKwCUGDtPvty.y/9tE7.HjvHXRB33aqCC0...
$2y$10$qBeJWuUaxEU.k51jmso54OIMZeSG24wYsAIr.ECHCzF...
$2y$10$99xUq7/x1s0kX1/P47J80uuKeZZ39PrK8qcwhVOOplb...
$2y$10$gQISa6liSgkwrGjOQzkoYOdAlsyssaUi5hbye7wyDSB...
$2y$10$94SnimacgAFlgJR9n.ssrOMKzuF9o6h58VqXC1ViKoo...
$2y$10$6fcC7rYJnQc.xp/nIU5AI.II9HJTwppV09fHMH09b8Q...

Description of image above: The nice, hashed password in my database safer from when I had them stored as regular text.



```
else{
    // hash password for security
    $hashedPwd = password_hash($password,  algo: PASSWORD_DEFAULT);
    mysqli_stmt_bind_param($stmt,  types: "sss",  &vars: $username,  &vars: $email, $hashedPwd);
    mysqli_stmt_execute($stmt);
    header( header: "Location: ../signup.php?signup=success");
    exit();
}
```

Description of image above: function found on YouTube that hashes passwords.

5. Works Cited

"Pentesting Fundamentals." *TryHackMe*, https://tryhackme.com/room/pentestingfundamentals.

"Introduction to Owasp Zap." *TryHackMe*, https://tryhackme.com/room/learnowaspzap.

"SQL Injection." *TryHackMe*, https://tryhackme.com/room/sqlinjectionlm.

"Prepared Statements - Manual." *Php*, https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php.

Krossing, Dani, director. *PHP Tutorials*. *YouTube*, YouTube, https://www.youtube.com/playlist?list=PL0eyrZgxdwhwBToawjm9faF1ixePexft- . Accessed 30 Apr. 2023.

Krossing, Dani, director. *HTML & CSS*. *YouTube*, YouTube, https://www.youtube.com/playlist?list=PL0eyrZgxdwhwP0AxnbBiDBCi53LK9uCMZ. Accessed 30 Apr. 2023.