**5118 COMP – Secure Software Development**

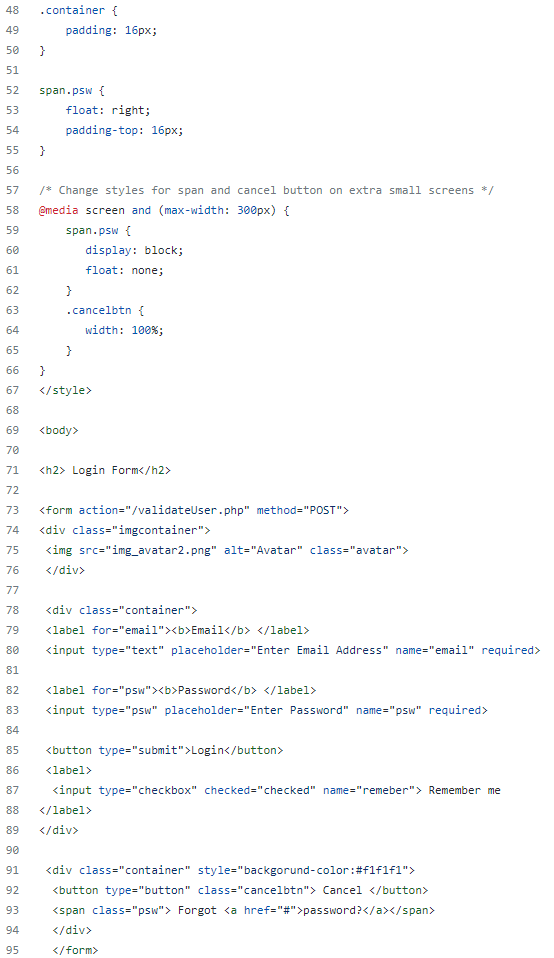
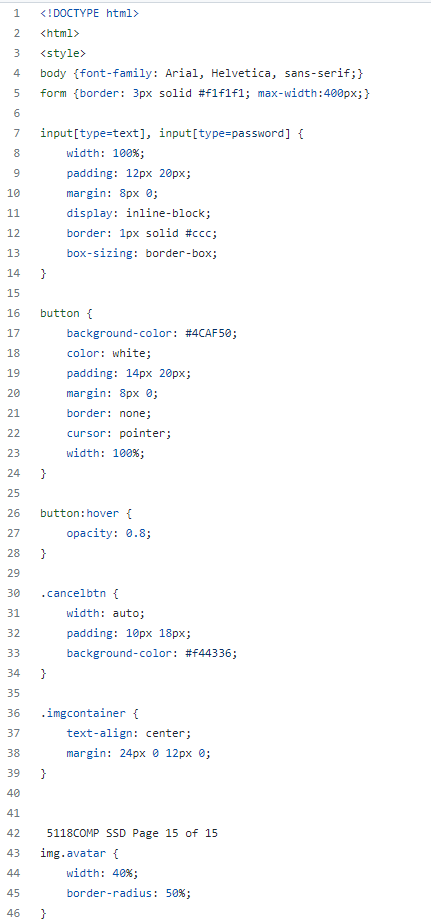
**Ethan Bennett, James Taylor Chapman, Joel Spencer, Luke Kelly, Lewis Wooding**

**Report –**

**Feature Explanations/Technical Descriptions:**

**Feature 1:**

Login and Signup Page:

Login Code -

**Code and how it works:**

The code above is the code for the login on the website which involves many types of features to ensure the login is suited to the needs we want and at the same time has a secure and safe operation when in use. Start of the code shows the basic styles and CSS functions to choose items such as colours and sizing of texts etc. things like the simple input including the password and text inputs and the CSS styles for the text etc. This type of CSS selection continues with all the buttons, inputs, avatar images and spans. Then it is the actual login form and how it functions and displayed for example to validate the user the backend source for that operation is the use of PHP to allow the interaction between the pages and logins when the user is entering and working with the login form. The email and password fields are required, and the login will not allow any continuation without a data entry in those fields and the code specifies and a message will appear when in use that is it indeed required in the entry. There is a simple checkbox added which as well includes a remember me statement to login when the user comes back to the login and this is backed up as well in the database using a relevant variable to ensure the data entered is remembered for the next session. Once entered a VARCHAR variable was created in the database to allow to max characters and string of 100 for the email and password and then the validation will occur once the data is entered and either the entries will be correct and the login is a success or the entries will be invalid and the SQL database will ensure based on our setup that the password will need to be re entered if the case is not valid. The entries may be too short or long and once rectified the user will be allowed to access the login and it will be a success and then the session can be activated and ready for the user.

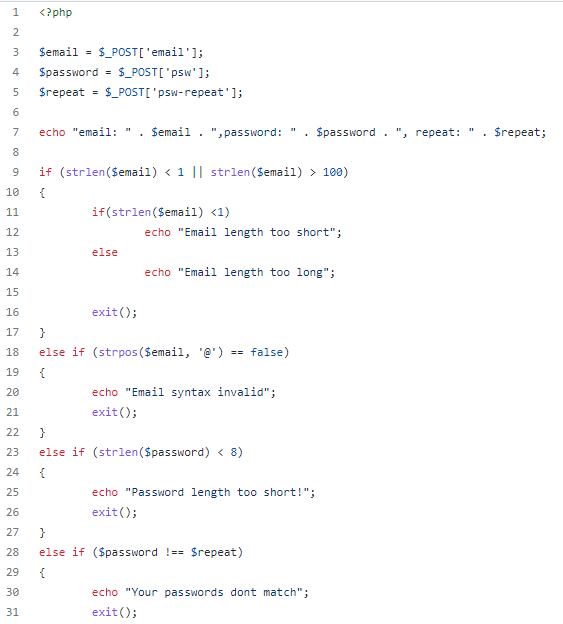
**Backend important technologies:**

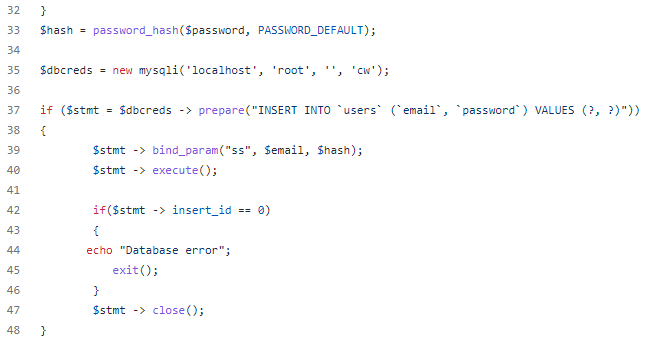
One of the backend technologies we used for the login page/the website was PHP and bootstrap which was a good way to create and work with our web application and have a consistent and relevant web development. Generating the web pages and specifically the login page was much simpler and more effective compared to using Java for example. Using the .php file extension instead of a .html would ensure the web pages and content would be created and ready for use. Things like user access and encryption of data was an important factor and use when it came to work on php and bootstrapping for our web application.

**How the data is transferred:**

For the transfer of the data for the login page we used SQL language which enabled is to complete the transaction and transfer of our data from the database which was created for the website in the background. Using both the initial database we created and then SQL to extract from the database it allowed to work with the pages on our site and the data to be in the correct and visible form that we needed to work on the site. Declaring variables in SQL allowed us to individually categorise was fields and field sizes etc we wanted for items and content on the website. For example, on the login page it was the password length and correctness that was needed to be verified and checked and using the variables in the database such as a VARCHAR allowed us to do this along with many other functions and features. Examples of the variables to transfer and gather data included bind\_parameter which allowed us to access and verify emails entered to the login form and execute different tasks. Variables such as loops for the data and store\_result for the database results and bind\_result with uid and uhash to allow the passwords etc to be checked and verified.

Sign Up Code:





**Code and how it works:**

The signup page allows the user to create an account and sign up to the website and it follows a similar style in terms of how the login works in terms of password checker and how to database is linked with the site follows the same coding styles as the login. There is a password, email and password repeat string which allows the login and then denies the login pending on the information entered by the user. Email syntax, password length and passwords matching will all be taken into consideration when the user inputs data and in terms of the session being able to work on or the signup being unsuccessful. Execution or exit of the feature will take place based on the input from the user and then it is a pure validation check to either allow login successful or unsuccessful. The password also has a certain criteria that it has to meet in order to ensure that the user can be more secure when being logged in, as it is a lower chance of getting hacked as the password will be harder to crack based off of the special characters.

**Backend important technologies:**

One of the backend technologies we used for the signup page was PHP and bootstrap again which followed the previous sign up page and was a good way to create and work with our web application. Generating the web pages and specifically the signup page was much simpler and more effective compared to using Java. Using the .php file extension instead of a .html would ensure the web pages and content would be created and ready for use. Things like user access and encryption of data was an important factor and use when it came to work on php and bootstrapping for our web application. This allowed us to create an effective signup page to create an account and using the backend database and SQL allowed is to meet character sizing of variables and load the data how we wanted.

**How the data is transferred:**

For the transfer of the data for the signup page we used the same format and performance we used during the transfer of the login page and we used SQL language which enabled is to complete the transaction and transfer of our data from the database which was created for the website in the background. Using both the initial database we created and then SQL to extract from the database it allowed to work with the pages on our site and the data to be in the correct and visible form that we needed to work on the site. Declaring variables in SQL allowed us to individually categorise was fields and field sizes etc we wanted for items and content on the website. For example, on the login page it was the password length and correctness that was needed to be verified and checked and using the variables in the database such as a VARCHAR allowed us to do this along with many other functions and features.

**Feature 2:**

Upload of any sort e.g. images/text/video

An explanation of the major code features in your website,

This feature allows the user to upload images to the website, these images are a main part of the blog as it allows the users to share pictures between each other. The website should only allow you to upload images by limiting the file extension and checking the header before allowing it to be uploaded to the server.

How the code functionally works

The code works by first filling each variable with the files data such as the file size, name and extension. It then compares these variables to the allowed array which decides if it can carry on. After passing this the code will check the size of the file and not allow anything over 20MB to be uploaded. The code will also check if that file is already installed on the server by checking the file name, if it does it will stop you as duplicate images will not be efficient for the website server. Once all these checks have been passed the file will be uploaded and the user will be given a success message.

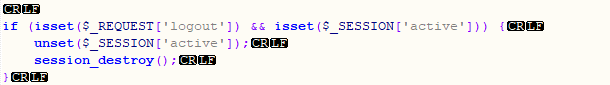
**Feature 3:**

Session Securing (Tokens)

Secure Session Code:

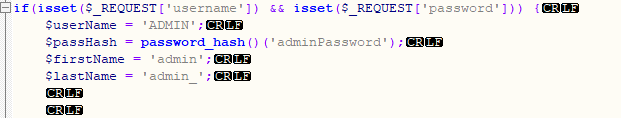
**Code and how it works:**

A session is used to identify between different user applications on a web application. When a user logs into an application a session is created on the server in order to identify the other request that a user is making. A session will continue if the user is on the system and is destroyed when the user logs out of the session

****

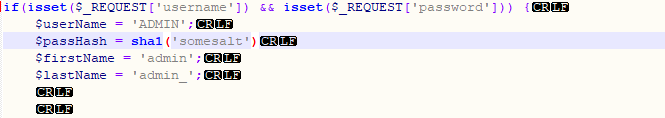
A session ID is string that is transmitted between the client and the server.

To protect the user’s session ID against any malicious actions password hashing has been used in order to make sure that the ID is as hard to get as possible.

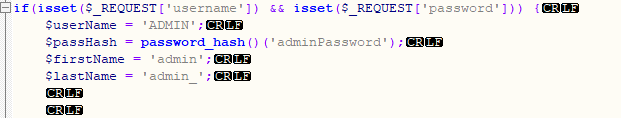
****

The way that session IDs can be found/got by the hijacker are;

 session sniffing: the attacker uses a sniffer to get network traffic which will contain the session ID when transferring data.

Predictable session token ID’s: this issue is created by not using strong encryption like sha1 for example ****

This isn’t secure as sha1 has been considered unsecure since 2005. So instead of using this for our code we have used “password\_hash” which is much more secure

****

Man, in the browser attack: this is more to-do with the client as it takes their computer to be infected first then the trojan can act when the user goes to the target site. This then allows the hijacker to change the details of the account. Due to the actions being done from the user’s device with the same session ID the server believes that the user is doing these requests when in fact the hijacker is.

Cross site scripting: this is when malicious scripts are injected into a website and it is then running on the user’s device. This is due to unsensitized data being inputted into the site and allowing scripts to be entered and ran. This can then get the session ID of a user or any other information that a hijacker may want.

Once a hijacker has gained access to the session, they can do anything that they want with the account that the legit user could do as they can act as if they are them. Examples of what they can do is to change passwords and usernames on the account, see and make transactions on the account and see financial data on the account and request private data.

In order to make sure that users are protected against this an intrusion detection system and intrusion prevention system can be used to defend against hijacking attacks. As nothing is 100% secure there is no set way to fully secure against session hijacking but there are many ways to defend against it.

The code is used to generate a session id so that Session hijacking can’t be used in order to attack the site. A session ID is a unique number that a site assigns a user’s session while they are using the site. The way that it protects the users is that it is generated using a hash algorithm so that it can’t be taken by someone trying to gain access to the session. It is more complex than a non-generated session id which is what makes it better.

**Backend important technologies:**

We changed the SHA-1 which is older and less secure changed it too password\_hash which is more updated secure and ease of use due to the function generating a safe salt on its own. By doing this it means that there is more defence against predicting tokens, it doesn’t fully protect the user as more software is needed in order to make sure that the accounts are as secure as possible and if they aren’t how to manage the situation.

**How the data is transferred:**

The session ID is stored as a field in the database along with the username and password (which is encrypted) so that the user can be identified by the session that they are on. If this were to be used as a fully functioning site then the following suggestions should be implemented in order to make sure that the user is more protected: the site could store the mac address or the ip or both of the user so that it can be cross-referenced each time the user logs onto the site. By doing this it means that although the session ID would have been killed from logging out when they come back to the site and are assigned a new session ID their mac and IP addresses can be checked against the device that has logged on in order to make sure that the account hasn’t been breached in any way. An argument that could be made against this is if the user logs in from another device and a way around this would be to create an automatic email service that sends an email to the user to alert them of account activity, if it was them then they can confirm and login and if it wasn’t them then they have to option to log out of the account and change the password to something more secure. This isn’t the most optimal form of security as the best would be for no-one to be able to access the account via the session ID at all but as nothing is 100% secure this is impossible.

**Extra information necessary to explaining the features:**

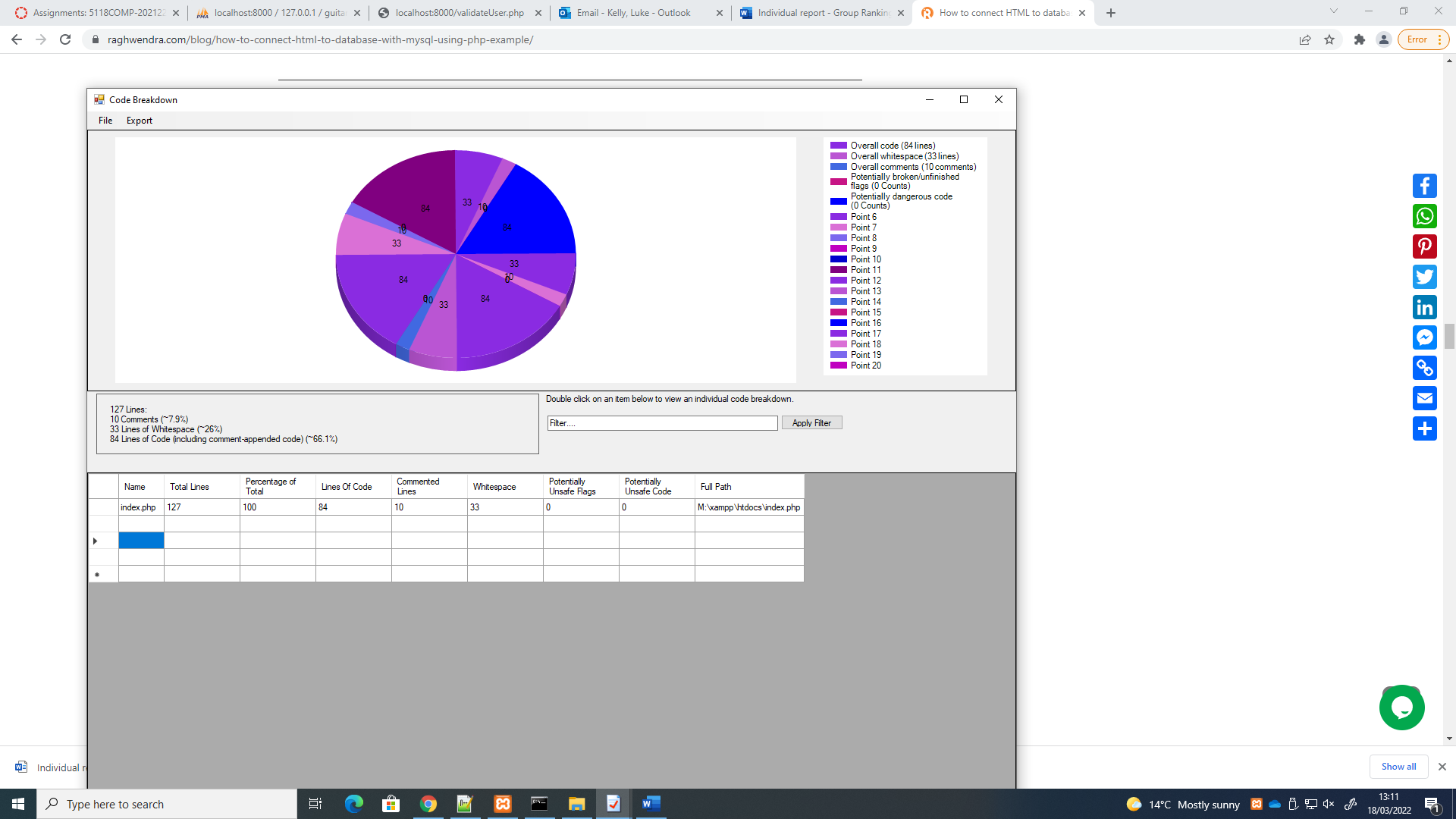
In the code there is a section that takes user data and stores it so that they can log on and create a session. This isn’t implemented into the final code as it was used as a test to make sure that the user can login and then a session ID can be generated to prove that they are the legitimate user. This data shouldn’t be used as the login data of the user as it isn’t securely stored as it is only stored in plaintext with no encryption behind it to secure it.

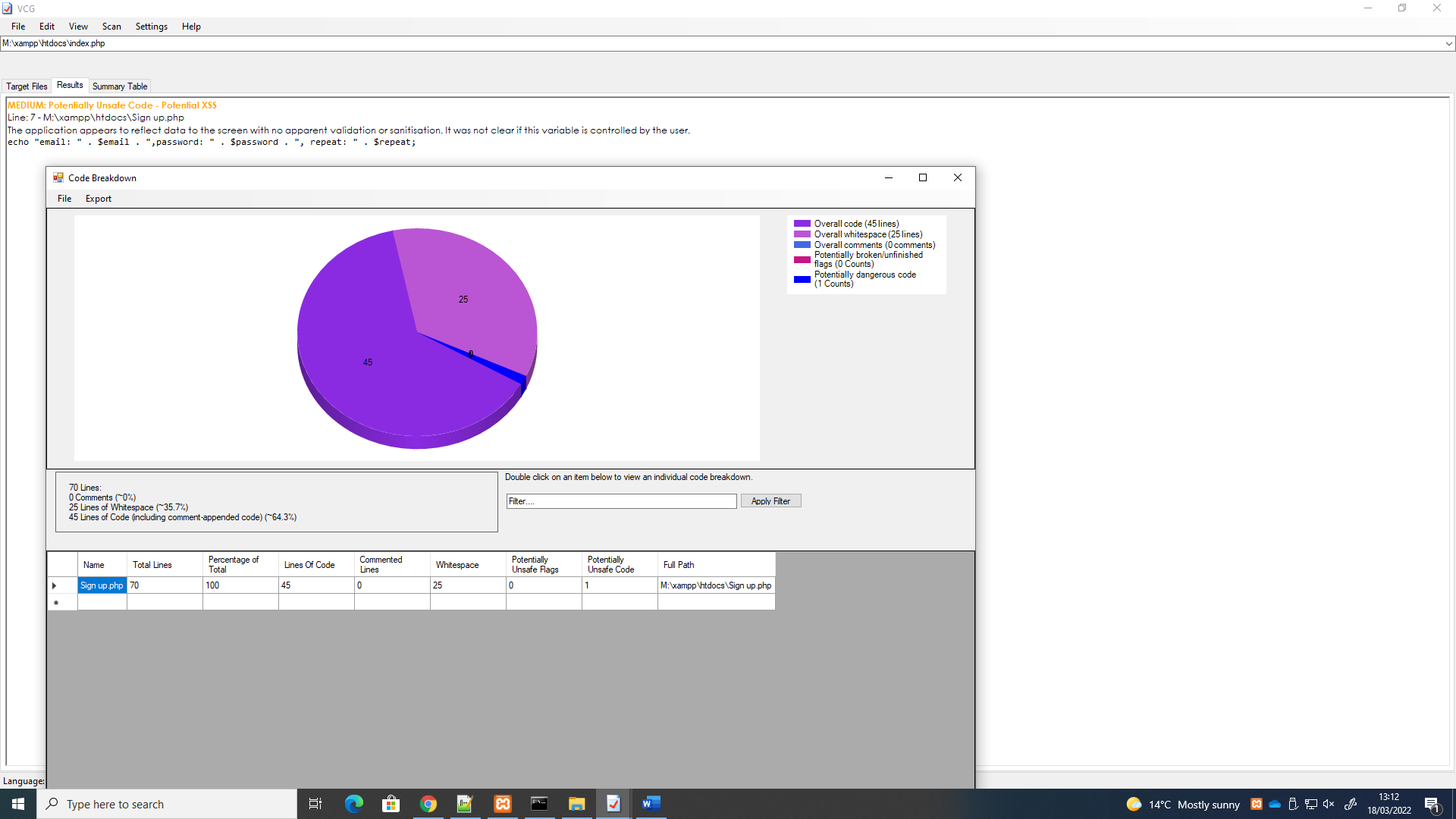
**Description of security processes followed during development/ Specific software development techniques and tools used:**

The security processes we used in order to ensure that the website was as secure as possible were using testing tables so that we could test everything we had such as data types/functions to ensure that they all worked. In the table, we included the name of the test, expected and real outcomes along with the amendments we made to ensure that the final product of the website was as secure as possible. On our table, if we did not achieve our expected outcome in the testing section, then we proceeded to make amendments, so the result is as secure and stable as possible. We looked at the procedures that should be taken when it comes too building and the development and things like having a secure setup and having the correct access levels in parts of the site. Also, things like backups in the site and correct database implementation was a vital and important way to ensure that the site is secure and ready for use. We also used the website GitHub which allowed us to keep all our code in a secure environment in order to prevent human interference and other external users from stealing/damaging it. GitHub also allowed us to upload work to the repository which everyone could access, and this allowed us to have an equal distribution of work and make sure that all of it was secure.

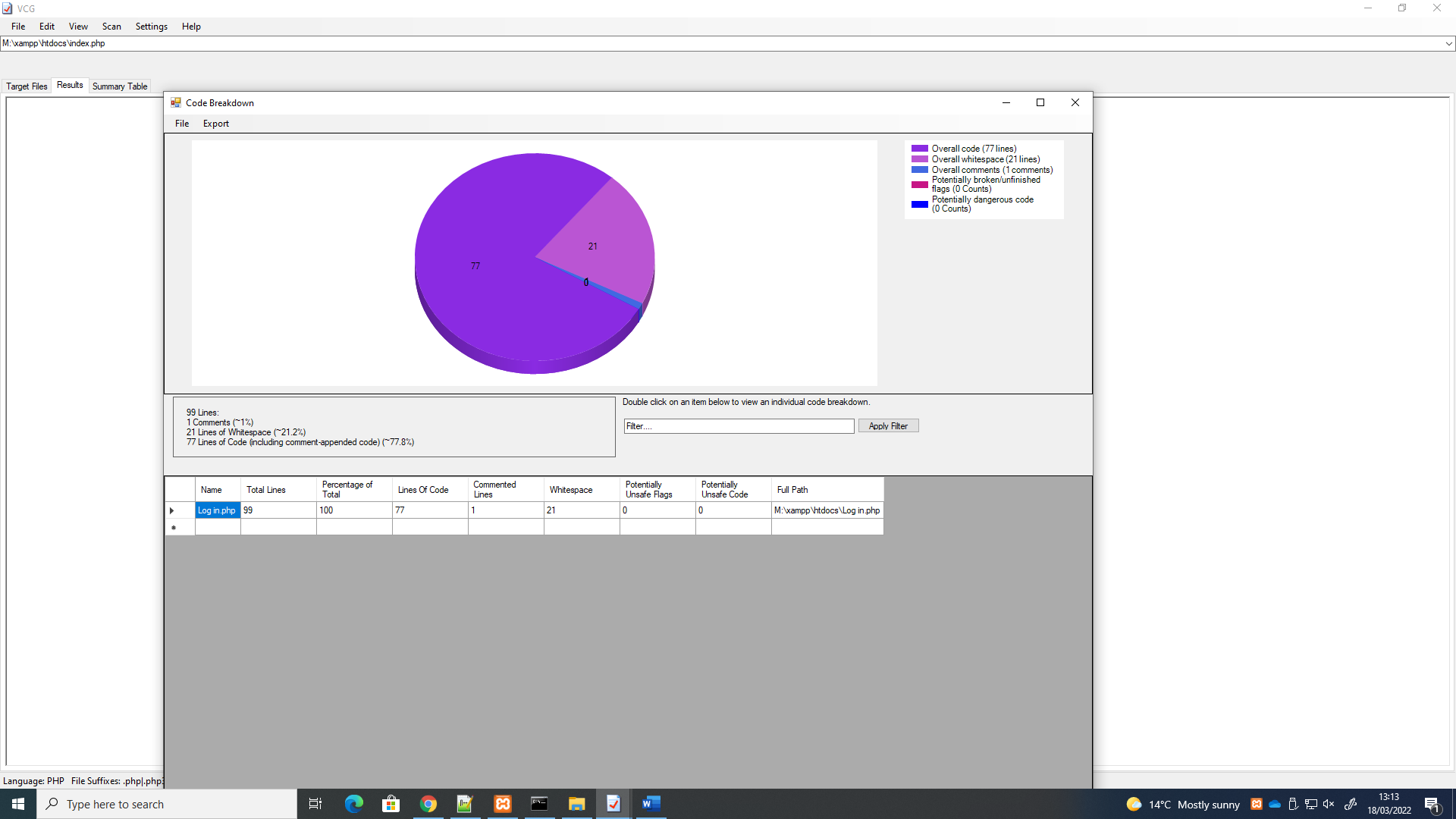
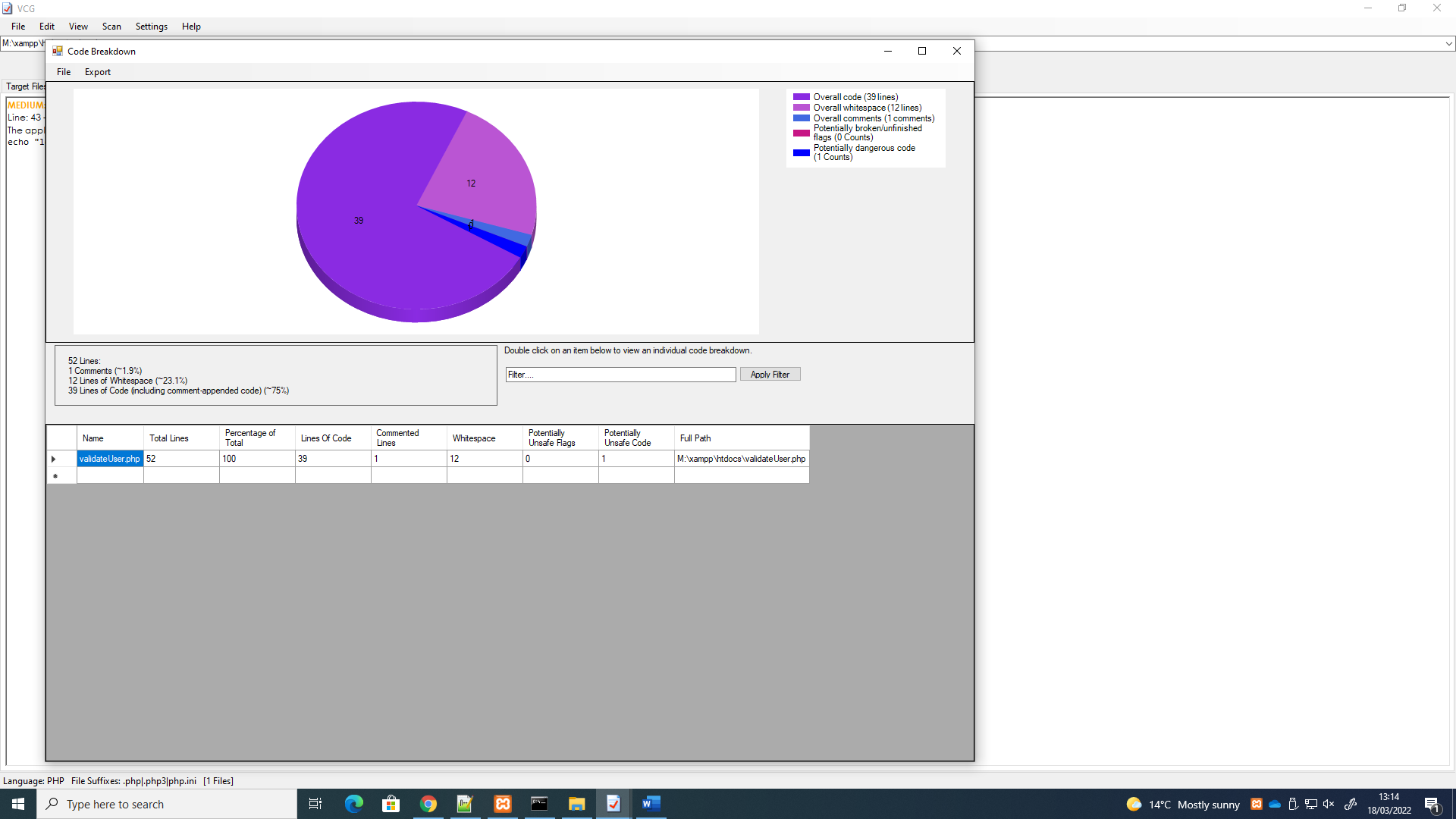
We also used a tool called OWASP ZAP which is a program used in order to manage security and find out any vulnerabilities in our code so that we can identify them and change if needed. It is good as it is an open-source application and its free, along with the fact that you don’t have to wait till the program is deployed. We also made sure that all our code was read over again so that we could spot minor errors before using zap so that most functions are all working for zap to scan them, and this makes it easier to secure everything because will be working and zap will find the vulnerabilities. We also stored the database credentials as encrypted for the sign-up section so that there will be no vulnerabilities there as the passwords can’t be seen. The credentials also used hash and salt so that brute force attacks are more unlikely, and the password is more randomized and thus is more secure than just a standard hashing process. We also made it so that when logging into the system, the passwords are typed into stars rather than actual text so that people looking over the user’s shoulder cannot see what the password is on the screen. We have also password protected the database where the data is stored so that if there is a hacker who is trying to gain access to the content then they cannot as the database will be protected by the admin.

**Testing Tool:**

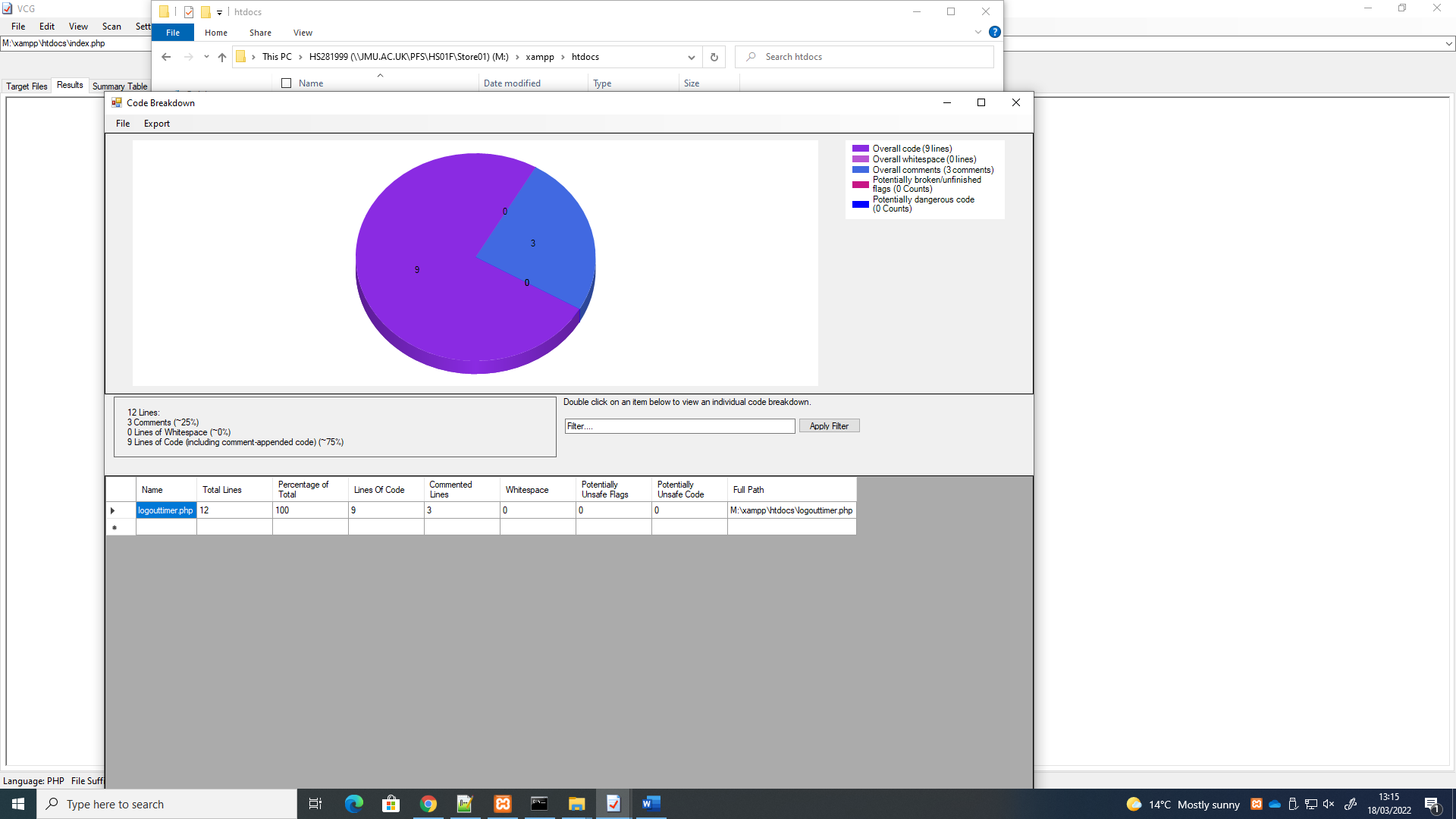
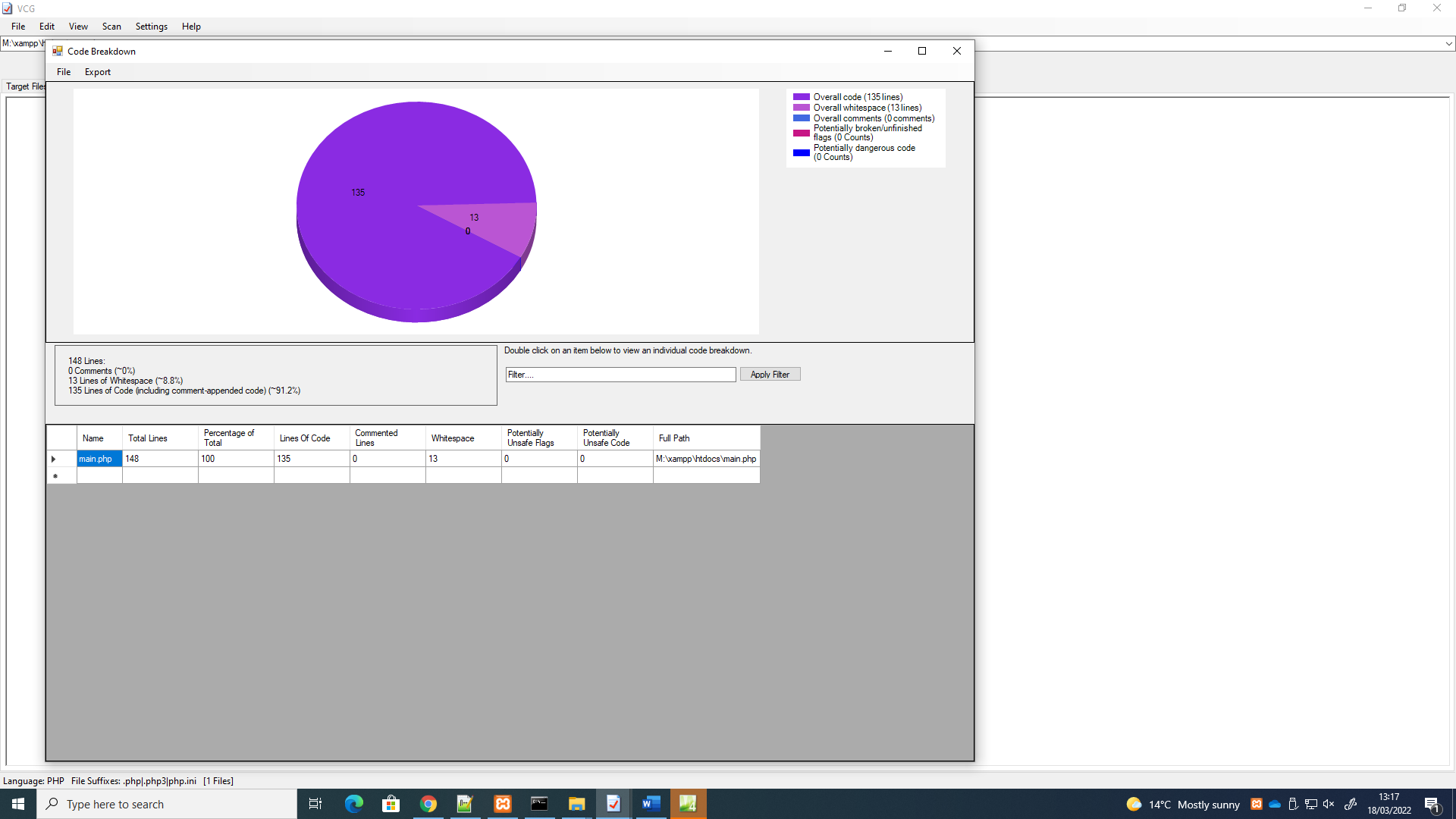
VCG Test Results:

Index.php results

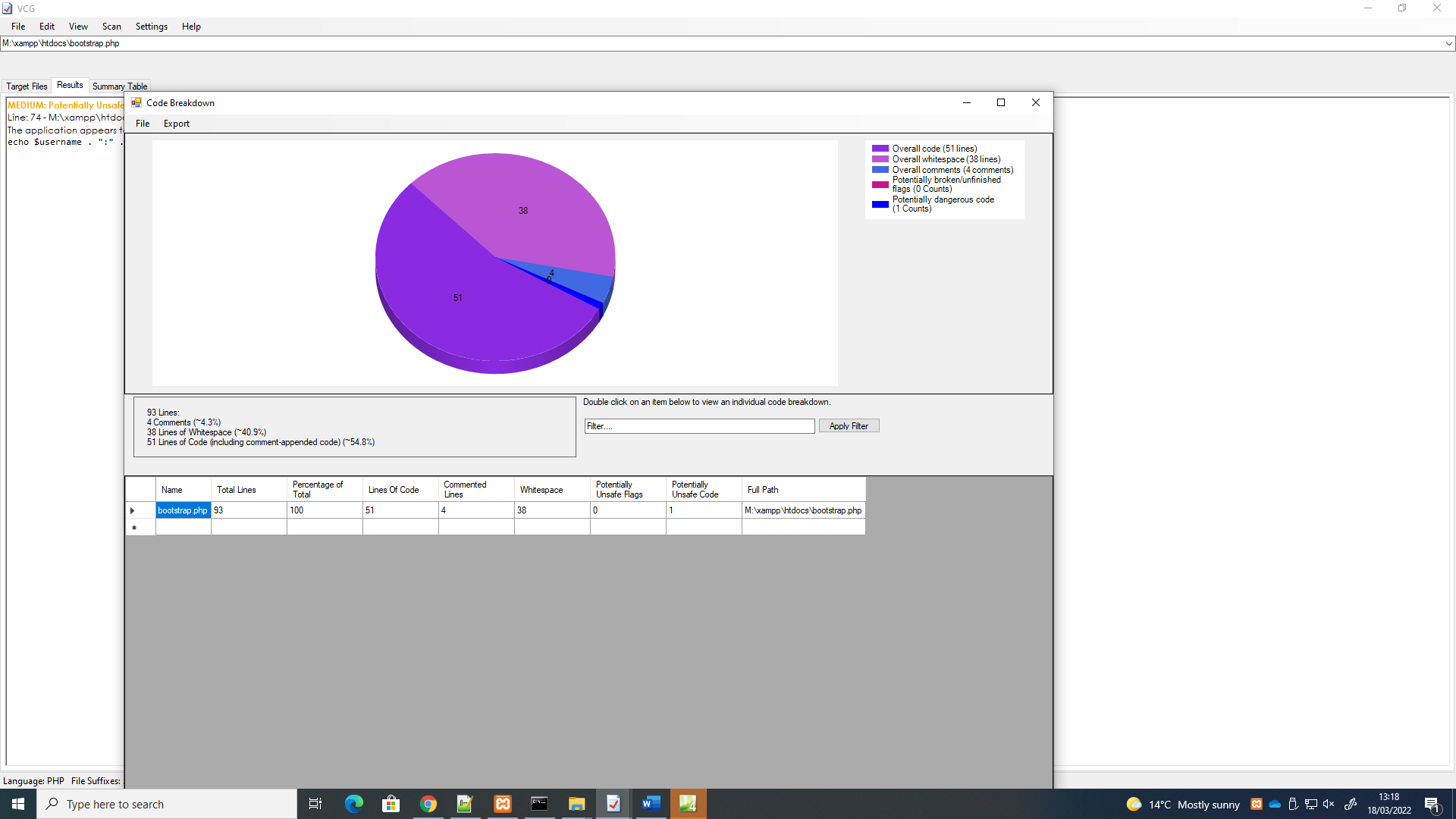
Signup.php results

Login.php results

Validateuser.php results

logouttimer.php results

main.php results

 Bootstrap.php results

**Detailed testing procedure (test plan):**

Test Plan:

|  |  |  |  |
| --- | --- | --- | --- |
| Test | What is Expected | What Happened After Test | Changes/Edits made on Test |
| Login/Signup | **For both features to accurately load and work how intended** | **Both features worked as wanted, only issue was the login was not fully secure** | **After the testing we knew the login setup was not secure, so we changed the code to ensure security** |
| Dynamic Content Loads on pages (text/images etc.) | **Text/images/any dynamic content on the pages need to load and be usable at intended for the user** | **Some elements including a dynamic video wouldn’t initially load and some images were poor quality** | **Changes in the code to edit image placement and video to ensure video plays when the user works with it** |
| Database (if linked and works) | **For all the content the database needs to be functioning otherwise not content or coding will be eligible and working** | **Database wasn’t working as expected and had errors in terms of linking and the overall functionality** | **Once errors were found by trail and error was came up with the solutions to ensure the database was successfully linked and working as intended** |
| Password/email checker on signup | **When the user logs/signs up on the content the password and email need to be checked and validated before further action** | **Testing both fake and real emails and passwords we found out that the checker was working as expected and only the parameters we set could be acceptable when logging in** | **After the test there was no real issues with the check, we did go back edit some parameters such as character size, but the feature worked as expected** |
| Security Vulnerability 1(fixed)SQL injection | **For the SQL injection to be effective and attack parts of our website but then secure once knowing the vulnerability** | **After securing all data was sanitized and no attack or threat can be taken place after the test or a further injection taking place** | **Securing the code to prevent injection attacks and not allowing any loopholes for any potential further attacks** |
| Security Vulnerability 2(fixed)Securing Session ID | **When the user is logging in/signing up and that is successful the session and the ID needs to be in operation and secured when that session is in fruition** | **Initially took us a few attempts to ensure the session was fully secure so edits to the code for example editing ID timers and how they operate needed to be done** | **Once these changes were made the session was fully secure and if the session was inactive and then it would be destroyed securely if the suer wasn’t fully working on the session** |
| Security Vulnerability 3(unfixed) Restrictions on File Upload | **When file uploads were being activated only a certain amount can be uploaded which is the unfixed issue whereas all files need to be uploaded** | **Initially most if not all files were able to be uploaded which was too secure so we needed to ensure that there wasn’t security in place so all files cannot be uploaded** | **Once the changes to make the files unsecure then it was ready to go in terms of having these restrictions in place** |
| Security Vulnerability 4(unfixed) log out timer | **So, when the user logs in to the site a timer will be activated for inactivity and if secure would log the user out, but we need to edit to ensure this is fully broken and unsecure and keeps the user logged in** | **The test seemed to be working but the timer on the log out would initially, so we needed to make the changes to get rid of the timer** | **We made changes to the timer by removing the parameters in the code so the user wouldn’t be logged out** |
| Website Loads | **When entering the URL website loads fully with all content and data working and functioning** | **When testing the site all content loaded and the website was fully visible to how we wanted** | **No edits or changes were made in the process due to all content being as wanted and loading correctly** |
| Pages are linked and function correctly | **Navigation around the site to be simple and effective for the user in terms of page linking etc.** | **Pages were linked effectively, and the content created on these pages were how we wanted it to be showcased** | **No edits or changes were required due to the site being effective and working as expected** |

**Research for all 4 vulnerabilities:**

**1st and 2nd vulnerabilities explanations(fixed):**

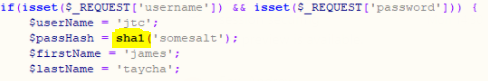
**1** MITM/ Spoofing attacks

**Technical description of attack**

In this scenario the attacker would spoof the website making the user believe they are on our legitimate website; this would trick the user into entering information such as usernames and passwords which the attacker will then have access to.

**Technical description of vulnerability with screenshot/ conditions needed to work**

The vulnerability here is the encryption used, SHA-1 is used here which is not suitable as it has been considered insecure since it was broken in 2017. It is insecure as the same hash can be made if given the same input, this means it is vulnerable as it can be reversed using a hashing table.



**Fixed code**

We have fixed this vulnerability by using a PHP function called password\_hash, this function hashes the password with a strong one-way algorithm. It adds a random salt to the password too, so it is not vulnerable to rainbow tables or dictionary attacks.

**Old insecure code new secure code**

****

**2** SQL injection

**Technical description of attack**

SQL injection is when the user takes advantage of a data input ability such as a text box asking for their username and runs an SQL command, If unprotected the SQL command will be able to run on the server. A simple command shown here can be very dangerous as it can show all usernames and passwords.

SELECT \* FROM Users WHERE UserId = 105 OR 1=1;

This is a common attack with a relatively basic fix.

**Technical description of vulnerability with screenshot/ conditions needed to work**

The code below is vulnerable to this sort of attack as it has no SQL parameters stopping an attacker.

**Screenshot of vulnerable code here**

**Fixed code**

We have now fixed this vulnerability by …. You can see below in the screenshots how this has been implemented.

**Old and new code showing the difference**

**3rd vulnerability evidence and explanation(unfixed):**

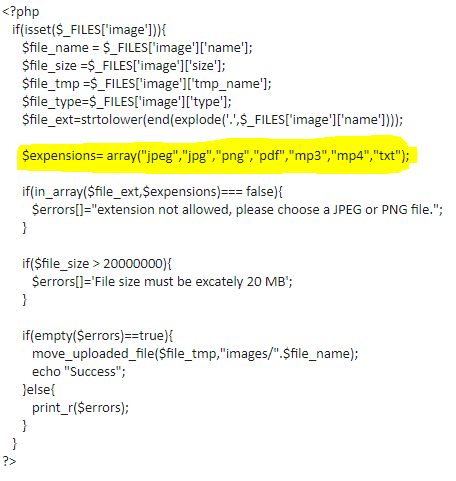
**3.** unrestricted file upload **(non secure))**

**Technical description of attack**

This type of attack takes advantage of no file upload limitation, this means any type of file can be uploaded to the server which the site is run on. Having a file run on the server side can be an extremely damaging attack as it can give full control to the attacker. This type of attack is the 3rd most common vulnerability found by wordfence in 2018.

**Technical description of vulnerability with screenshot/ conditions needed to work**

There is no sanitization on the file name meaning the attacker can upload any type of file extension, this file can then be later accessed by the attacker via the web.

The code below is vulnerable to this sort of attack as it more than the necessary file extensions. ****

**What could you do to fix this vulnerability?**

One way to fix this vulnerability would be to add an array of allowed extensions, in this case .gif, .jpeg. For added validation we could use the getimagesize() feature of PHP which will read the header information of the file to determine if it is an image. This would help prevent any files being uploaded that contain executable code.

**4th vulnerability evidence and explanation(unfixed):**

**4.**  Logout timer

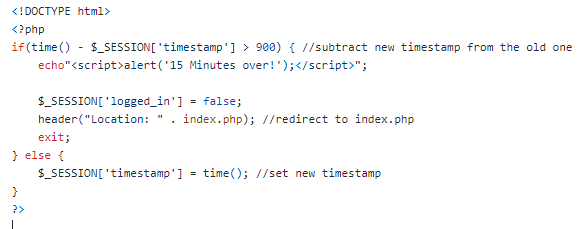
**Technical description of attack**

This attack would take advantage of the website not correctly logging out, when the user does not input for a set amount of time the website should automatically logout. This help authenticate that the user is in the correct account.

**Technical description of vulnerability with screenshot/ conditions needed to work**

The vulnerability shown below shows how

**Screenshot of vulnerable code here**

****

**What could you do to fix this vulnerability?**

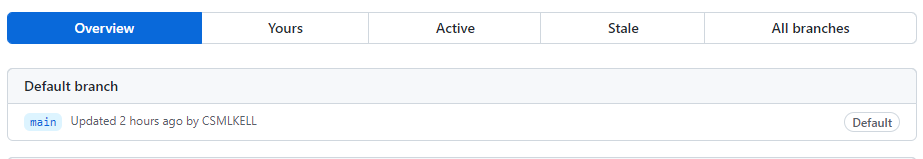
We could fix this vulnerability by implementing a working log out feature, this is a basic feature that would end the session id once clicked. The user would then have to resign in to have access again, this increases security as it makes it less likely an unauthorized user is accessing the account. To make this feature work it needs to log the user out not just redirect to the signup page.

**Source control repository usage and logging (GitHub):**

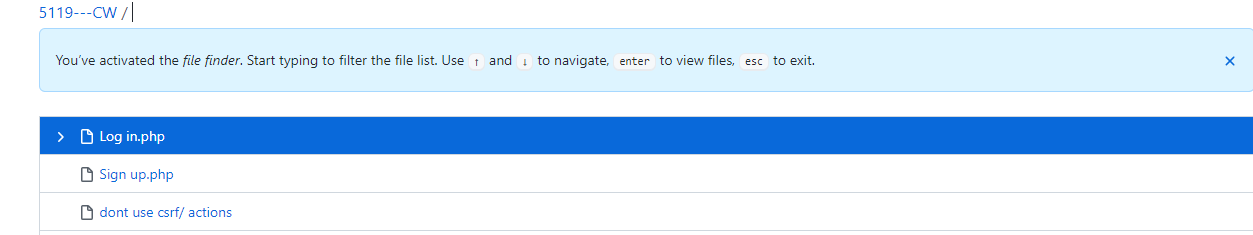
GitHub usage:

For the side of the project of which involved collaborative working and source control examples and usage we used GitHub due to the features, techniques and ease of access to work on the code and use the feature effectively how we wanted. We used GitHub for multiple reasons for its quality of features and the way we were able to easily communicate and operate on the project effectively.

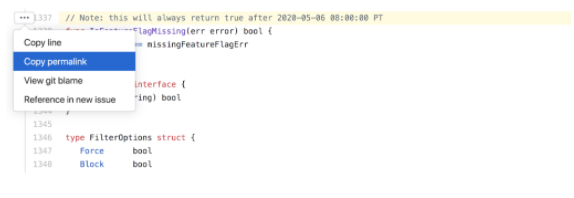
Branching:

The image above is an example of the branching technique in GitHub which allows users within the repository to view and edit the files in the repository. There are different examples of branches from the default to the protected branches which offer different rules and the way things be stored and edited in the repository. The initial and first branch is the default and then added branches can be added and created after.

Fuzzy Finder:



The fuzzy finder is a general and basic technique and good practise which allows the navigation between the branches and files in GitHub. It is also classed as a command which can be accessed and used by selecting [t] on the keyboard then the navigation can be allowed through the processes and other functions. It is useful to quickly access through the files at a high rate and there are no dependencies and is extremely simple to use for basic and high-end users.

Code Snippet Linking and Editing:

This is a standard feature and practise that isn’t necessary known as others but can be useful and especially when collaborative working is play with the example of other users adding and working on types of code and files in the repositories. This feature allows the linking between lines of code specifically when viewing a file and the content in the file. Once selected the user will be directed straight to the selected lines and that URL will always work unless the initial link is edited or changed. Pressing [y] and [copy permalink] functions allows the URL to both work and then continually work after.

**GitHub commands and functions:**

Examples below some of the basic commands and functions with there descriptions of the github commands that can be commonly used in general, but most used by our group during the project. All the below showcase most of what we have done and used during our github use.

Removing a file and or folder – [git rm -r [file-name.txt]

Creating a new branch/ deleting a branch – [git branch [branch name] [git branch -d [branch name]

Showing the commits and differences between Branches – [git log branch1/A ... branch2/B]

Adding/Working with remote repository – [git remote add origin/[username]/[repository-name]. git

In your individual report, please rank each group member (including yourself) out of 100% indicating how much they contributed to the group effort.

|  |  |
| --- | --- |
| Group Number: |  |
| Group Members | **Contribution (out of 100%)** |
| Lewis | 20% |
| Ethan | 20% |
| James | 20% |
| Joel | 20% |
| Luke | 20% |

**Lewis – report writer**

Vulnerability research

Security modelling & testing (testing tool)

Technical description of website features

**Ethan – report writer**

Security modelling & testing (test plan, testing tool)

Technical description of website features

Evidence of source control repository usage

**James - coder & report writer**

Security modelling & testing (security processes taken in development)

Technical description of website features

Session securing code

Worked on login section

Design of site

**Joel - coder**

Security modelling & testing

File upload system

Design of site

Made signup, login and review page

**Luke - code**

Security modelling & testing

Login and Sign up page

Creation and connection of database to web app

Implementation of file upload feature (Left vulnerable)