*Members:*

*Armin Roushan, Georgiana Manolache, Jan-Niklas Schneider, Nena O’driscoll, Raditya Surya, Radu Stoica*

SePr Final Report (eFontys)

[Year]

Contents

[Introduction 3](#_Toc464852582)

[1. Design Overview 4](#_Toc464852583)

[Introduction 4](#_Toc464852584)

[User Group 4](#_Toc464852585)

[Customer 4](#_Toc464852586)

[Administrator 4](#_Toc464852587)

[Network Architecture 4](#_Toc464852588)

[2. Threat Analysis 6](#_Toc464852589)

[Asset Management 7](#_Toc464852590)

[Authorization Requirements 8](#_Toc464852591)

[Risk Analysis 8](#_Toc464852592)

[3. Secure Design 10](#_Toc464852593)

[Attack Trees 10](#_Toc464852594)

[Misuse Cases 10](#_Toc464852595)

[Injection of malicious SQL string 10](#_Toc464852596)

[Insertion of XSS code 10](#_Toc464852597)

[Cross-Site Request Forgery 10](#_Toc464852598)

[Brute force attack 10](#_Toc464852599)

[Software mitigation 11](#_Toc464852600)

[Misuse Cases 11](#_Toc464852601)

[4. Implementation Choices 12](#_Toc464852602)

[5. Code Examples 12](#_Toc464852603)

[XSS Mitigation 12](#_Toc464852604)

[Brute Force Prevention 13](#_Toc464852605)

[SQL Injection Prevention 15](#_Toc464852606)

[CSRF Prevention Using Token 15](#_Toc464852607)

[Malicious File Upload Prevention 16](#_Toc464852608)

[6. Group Peer Assessment 17](#_Toc464852609)

[7. Personal Evaluation 18](#_Toc464852610)

[Radu 18](#_Toc464852611)

[Georgiana 18](#_Toc464852612)

[Armin 18](#_Toc464852613)

[Jan-Niklas 18](#_Toc464852614)

[Raditya Surya Pratama 18](#_Toc464852615)

[Nena 18](#_Toc464852616)

[8. Recommendation and Evaluations 19](#_Toc464852617)

[9. Project Conclusion 19](#_Toc464852618)

[References 20](#_Toc464852619)

[Appendix A: eFontys Pen-testing Report 21](#_Toc464852620)

[Introduction 21](#_Toc464852621)

[Portscan Server 21](#_Toc464852622)

[Forced Browsing 22](#_Toc464852623)

[Spider web application 23](#_Toc464852624)

[Testing for Vulnerabilities 24](#_Toc464852625)

[Testing for SQL Injection 24](#_Toc464852626)

[Testing for XSS 25](#_Toc464852627)

[Testing CSRF / XSRF 26](#_Toc464852628)

[Testing for File Inclusion 27](#_Toc464852629)

[File Upload 28](#_Toc464852630)

[Conclusion 29](#_Toc464852631)

[Appendix B: Cloud Service Pen-testing Report 30](#_Toc464852632)

[Introduction 30](#_Toc464852633)

[Goals 30](#_Toc464852634)

[Results Analysis 30](#_Toc464852635)

[Changes on The Server 33](#_Toc464852636)

[Other Screenshots From The Analysis 33](#_Toc464852637)

[Conclusion 36](#_Toc464852638)

# Introduction

Security breaches have been increasing considerably over the last decade, becoming an everyday problem for both low and high-profile organizations. In fact, hacked websites have become so common that typically only the biggest data breaches capture enough attention to make headlines (WhiteHat Security, 2015). Breach reports are being posted weekly, presenting in detail information about data exposure events. For example ITRC[[1]](#footnote-1) (IDTheftResourceCenter), which performs breach reports on United States websites, cover a wide range of categories where data might be put at risk. According to ITRC report for 2015 (Data Breach Report, 2015), banking and fanatical breaches have been ranked third most vulnerable, after business and medical and healthcare.

In this report, we present possible banking breaches and best-practices to improve security of an online financial service. We have implemented a simple online banking system and enhanced its security infrastructure. Finally, pen-testing is added to further test the security of our website.

# Design Overview

## Introduction

This chapter will cover the pre-knowledge, final design decisions and security regarding the SPA Fontys e-banking system called eFontys. The application is written in PHP, HTML5, CSS3 and MySQl.

## User Group

The application allows for two different roles, namely the administrator and the customer. A new customer can be added only through the administrator page.

### Customer

The customer has an account with the ability to:

* Edit account details
* Check current balance
* Deposit money into own account
* Transfer money from own account to recipient
* Logout

### Administrator

The administrator’s account has the same abilities as the customer but can also:

* Add and remove users
* Edit other account details

## Network Architecture

The network architecture looking as follows. With eFontys being hosted on the Athena server. The database and server are connected through the internet.

Figure 0‑1 Network Architecture Diagram



As shown on the Network Architecture, you can see that the Fontys bank application is hosted by a private server, which is a hosting service provided by Fontys called Athena.

The web application itself had really basic banking functionalities which is:

* Login
* Check balance
* Deposit Money
* Transfer Money

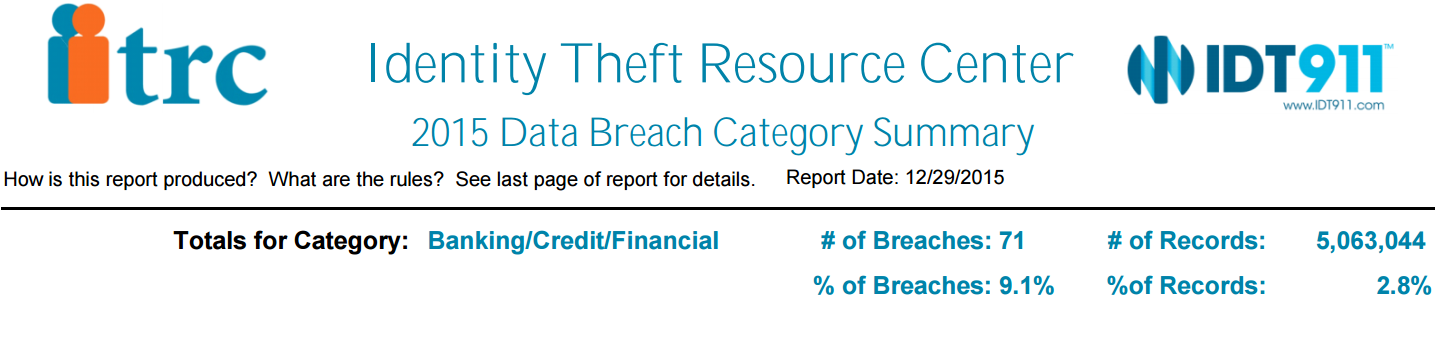
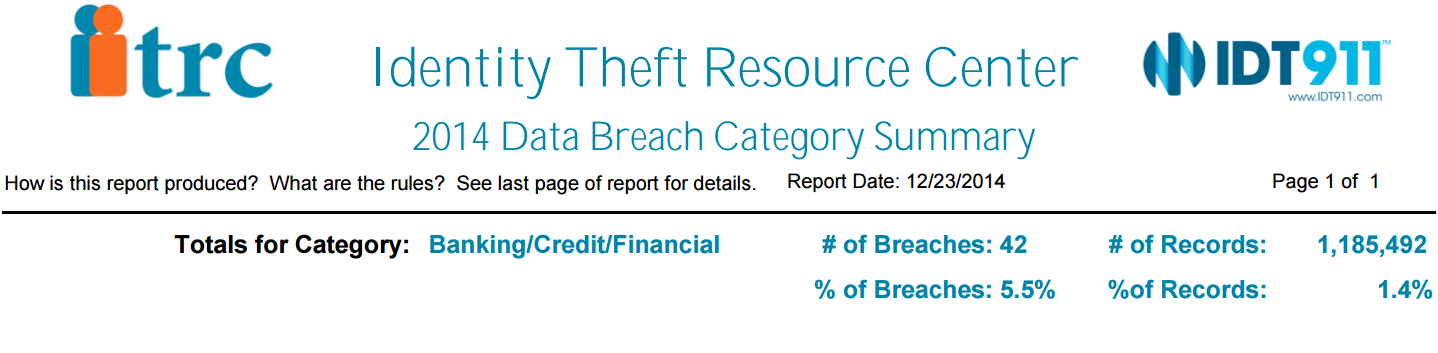
The web application is using MySQL to record all the data. The database has two tables which is ‘user’ and ‘account’ table that contain all the user records.

# Threat Analysis

Financial services and especially online banking services have always been a target for cyber criminals. The number of reported data security violation continues to increase while becoming more diverse and sophisticated.  According to ITCR (Identity Theft Resource Center) Annual Breach Reports, only in the United States in 2015 the recorded banking sector braches - 71 in total - were nearly double in number compared to 2014 with 42 breaches (Figure 4-1). Between 2010-2014, banks experienced attacks with losses over 30.000 records due to hacking or poor security as shown in Figure 4-2. Among the affected financial services organizations there is JP Morgan Chase, European Central Bank, US Federal Reserve Bank of Cleveland, Citigroup (World's Biggest Data Breaches, 2016). In a study conducted by the Verizon Business RISK Team in 2009, it was reported that the cyber criminals focus on the theft of personal identification number (PIN) information and their associated credit and debit account information (2009 Data Breach Investigations Report, 2009).

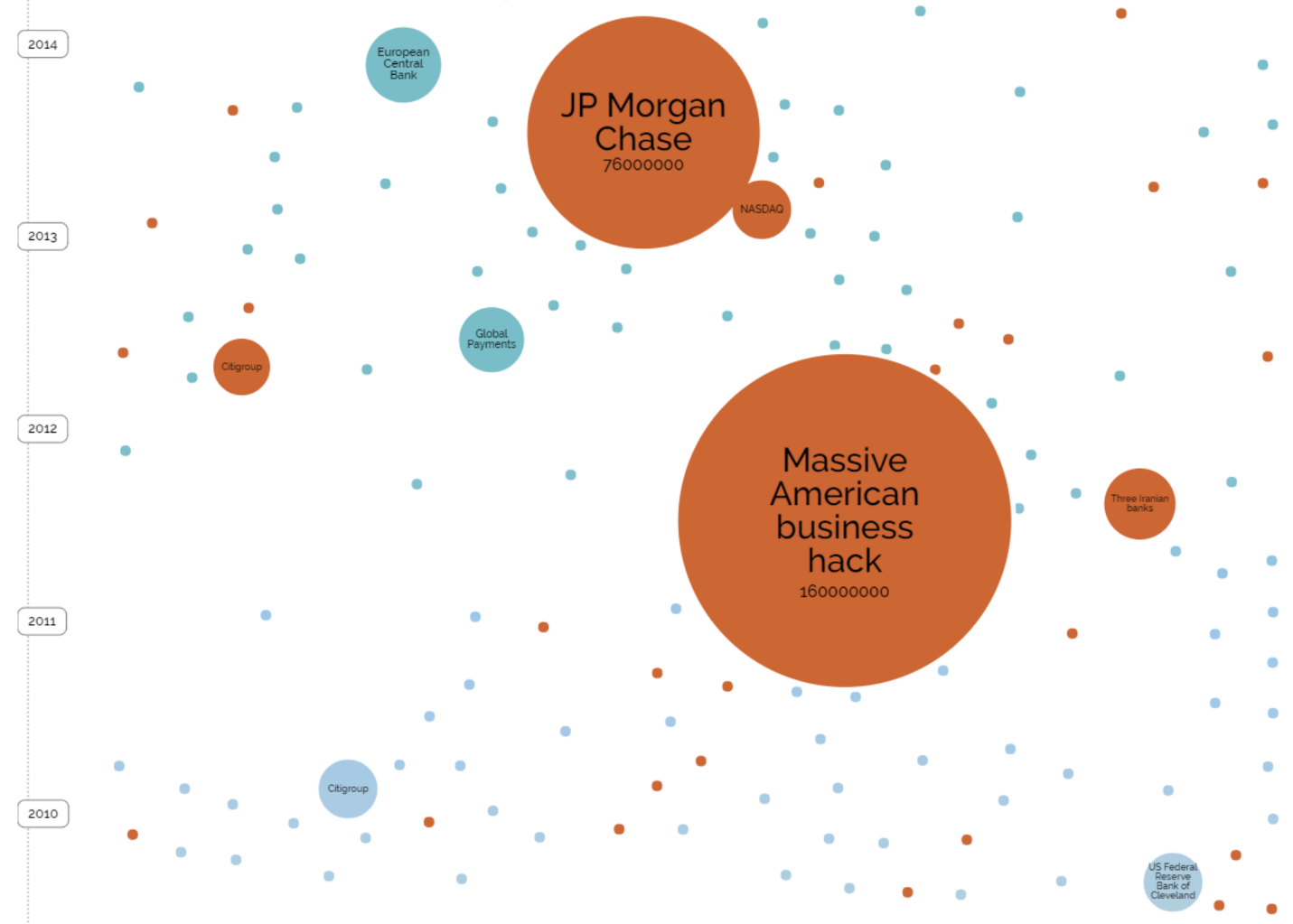
Protecting financial data and assets is, thus, vital for a financial organization. The following subchapters describe the risk analysis of the online banking web application.

Figure 0‑1 ITRC Stats Summary 2014-2015



*Source: Identity Theft Resource Center Breach* [*http://www.idtheftcenter.org/ITRC-Surveys-Studies/2015databreaches.html*](http://www.idtheftcenter.org/ITRC-Surveys-Studies/2015databreaches.html)

Figure 0‑2 Major financial data breaches 2010-2014 with losses greater than 30.000 records.

**

*Source: World’s Biggest Data Breaches* [*http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/*](http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/)

## Asset Management

The online banking application must maintain the majority of information at a high security level. Apart from the log record, data elements are strictly confidential, private, integral and availability is minimal. Table 4-1 describes the choses level of each data sets.

Table 4‑1 CIA Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bank data | Data elements | Data classification  (C, I, A) | Privacy | Explanation |
| User account | Username  Password (hash) | **H, H, H** | **H** | User credentials are strictly private and should not be available to anyone but user. All data classifications are important especially for bank reputation and financial loss. |
| User information | Name  Email  Phone number(s) Address | **H, H, H** | **H** | User information is highly private and confidential. All information must be confidential. |
| Account information | IBAN  Amount | **H, H, H** | **H** | The user IBAN is available at user permission (given by user himself) as transactions are performed to other users by specifying these data. |
| Log record | Date  Time  Type  Sender  Recipient | **H, H, M** | **M** | Log reports are available to the web administrator for auditing. Mainly for auditing. |

Legend: H = High, M = Medium.

## Authorization Requirements

Within the online banking application, the web admin has highest authority rights over the data; a web administrator is responsible for ***creating*** a new user account, ***read*** data, ***update*** data on user request or ***delete*** user account when customer decides to withdraw his account from the banking service. The same authorization is assigned to the rest of the data groups existent within the web application: user information, account information and log records. The user has limited authority (read, update) over data, to avoid misusages and potential fake users. Table 4-2 gives an overview of the access-rights per user groups.

Table 4‑2 Authorization Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Group data | User | Web administrator | Anonymous  (website visitor) |
| User account | **U** | **CRUD** | **-** |
| User information | **R, U** | **CRUD** | **-** |
| Account information | **R, U** | **CRUD** | **-** |
| Log record | **R** | **CRUD** | **-** |

Legend: C = Create, R = Read, U = Update, D = Delete.

## Risk Analysis

According to SafeNet (Top Online Banking Threats to Financial Service Providers in 2010 , 2010), leading market in protecting financial transactions, there are four majorly applied attacks. Most frequent is ***Phishing***, typically carried out through e-mail or instant messaging, providing links or instructions that direct the user to fraudulent Web sites portrayed as legitimate ones. ***Password Database Theft*** is a threat in which hackers get possession of costumer data from other less protected websites assuming user uses similar user ID and password. By collecting personal information**,** cyber criminals can assume individual identity, also known as ***Identity Theft*** threat. ***Man-in-the-Middle (MitM)*** is this type of threat in which the attacker can actively inject messages of its own into the traffic between the user's machine and the authenticating server. ***Man-in-the-Browser (MitB)*** is a variant of the MitM attack, that infects the user internet browser and inserts itself between the user and the Web browser, modifying and intercepting data sent by the user before it reaches the browser’s security mechanism.

Table 4-3 gives an overview of the possible risks that could occur on an online banking website.

Table 4‑3 Risk analysis

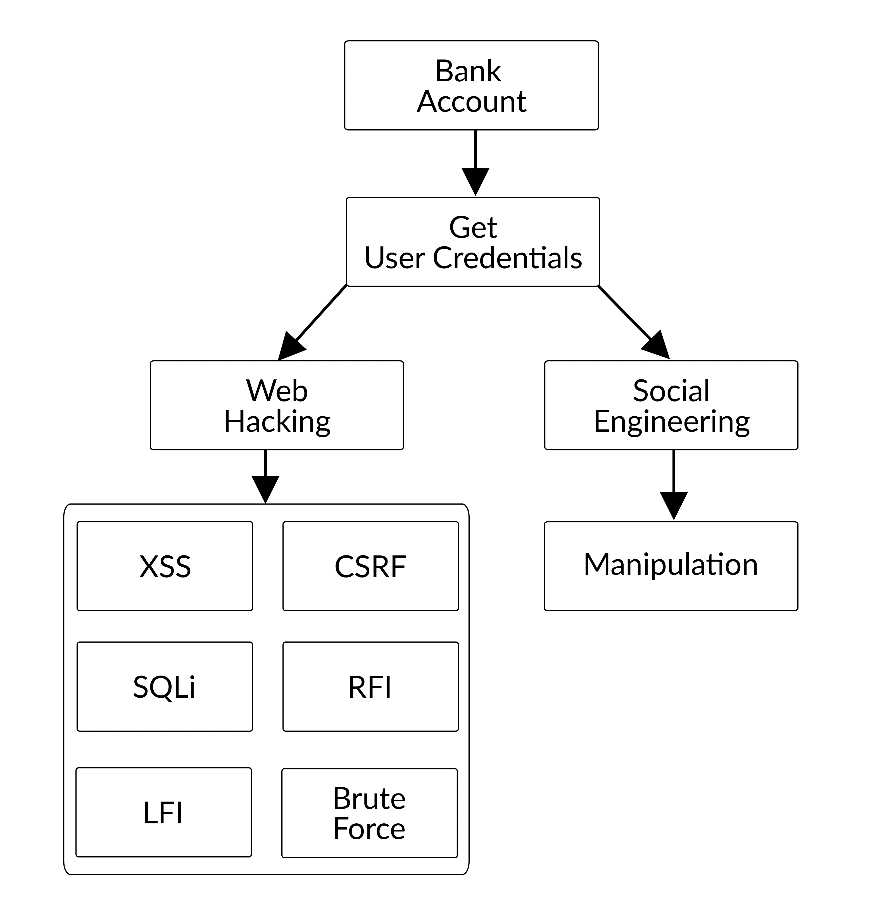
|  |  |  |  |
| --- | --- | --- | --- |
| Threat | Risk | Impact | Needed Security level Risk impact |
| Phishing | **High** | **High** | **High** |
| Password Database Theft | **High** | **High** | **High** |
| Man-in-the-Middle (MitM), Man-in-the-Browser (MitB) | **High** | **High** | **High** |
| Identity Theft | **High** | **High** | **High** |

# Secure Design

Before proceeding to the implementation of the website, security design consideration was taken into account. This step was necessary to have better overview of the parts which could be at risk.

## Attack Trees

Figure 0‑1 Attack trees



## Misuse Cases

### Injection of malicious SQL string

An attacker might try to insert malicious SQL command into the provided inputs on the web application in order to change the balance. SQL injection could also retrieve sensitive information such as user information, or get access to the admin account which is the administrator of the website.

### Insertion of XSS code

The attacker may insert different JavaScript or PHP code lines in order to cause damages to the website or to take control over the web application.

### Cross-Site Request Forgery

The website is creating a session in order for the user to log in. This can be a potential vulnerability of the website. The hacker may try to take advantage of it in order to get sensitive information.

### Brute force attack

By brute forcing the password people may try to retrieve a user password.

### Software mitigation

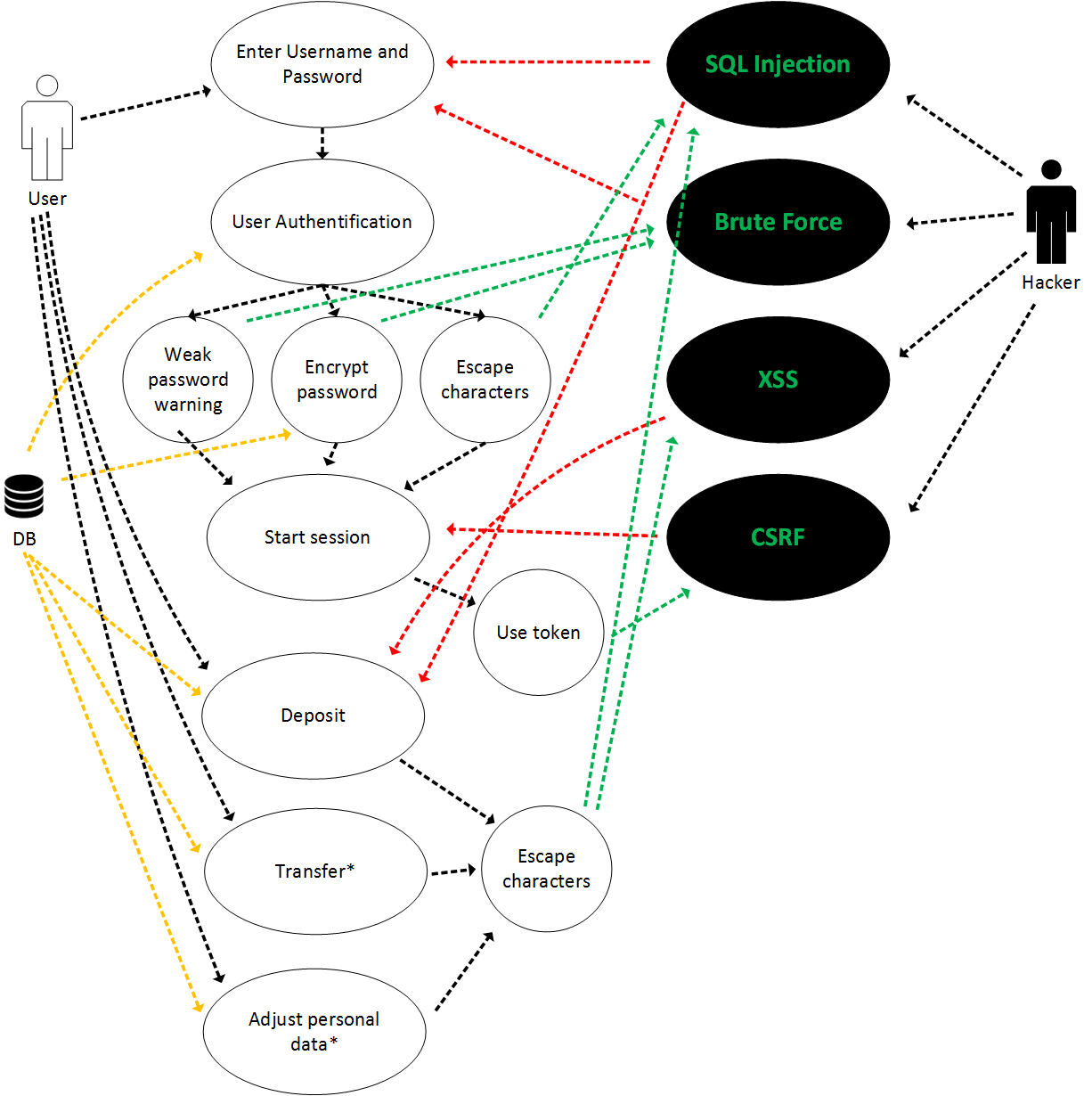
To prevent and block unwanted requested from hackers were implemented next functions

* Use escape special characters for inputs in order to prevent SQL injection.
* Use token for PHP session in order to prevent CSRF.
* Encrypt passwords in order to decrease the risk of finding the real password in case of a data leak.
* Check the data inserted in the inputs in order to prevent XSS code.

## Misuse Cases

Below is shown image with possible hacker’s attacks and mitigations:

Figure 0‑2 Misuses case diagram



# Implementation Choices

In our web application, we decided that we want to implement the secure design to prevent the following vulnerabilities:

* Cross-site scripting (XSS)
* Brute Force attack
* SQL Injection attack
* CSRF Attack
* Malicious File Upload attack

The mitigation that we implemented will be shown on the Code Example section.

# Code Examples

We will present the implementations that we have on our website for preventing the vulnerabilities that mentioned on the previous section.

## XSS Mitigation

XSS can be mitigated by sanitizing all the input and encode the output with htmlentities functions. We implement it almost in our source code, but we left one XSS vulnerabilities for other group to find in the transfer recipient name and show it in the notification error on the page.

For the input, we only check if the input is empty or not, and satify the validation or not. We let the user to input script tag or html tag but does not allow it to rendered by encoding the output of the inputted string.

We created a PHP function to encode all the output by using UTF-8 Encoding.

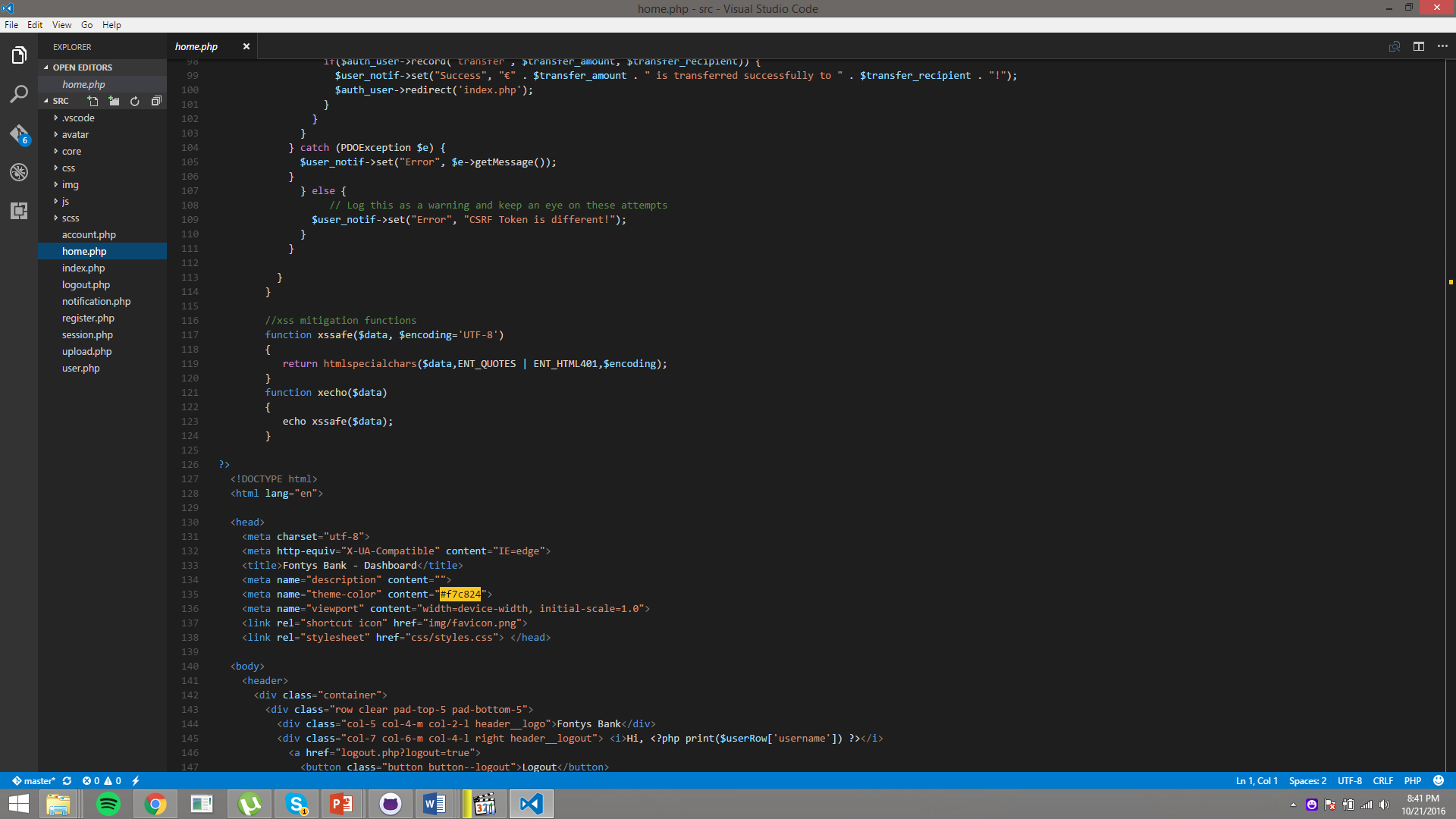


Figure 6 XSS SAFE Function

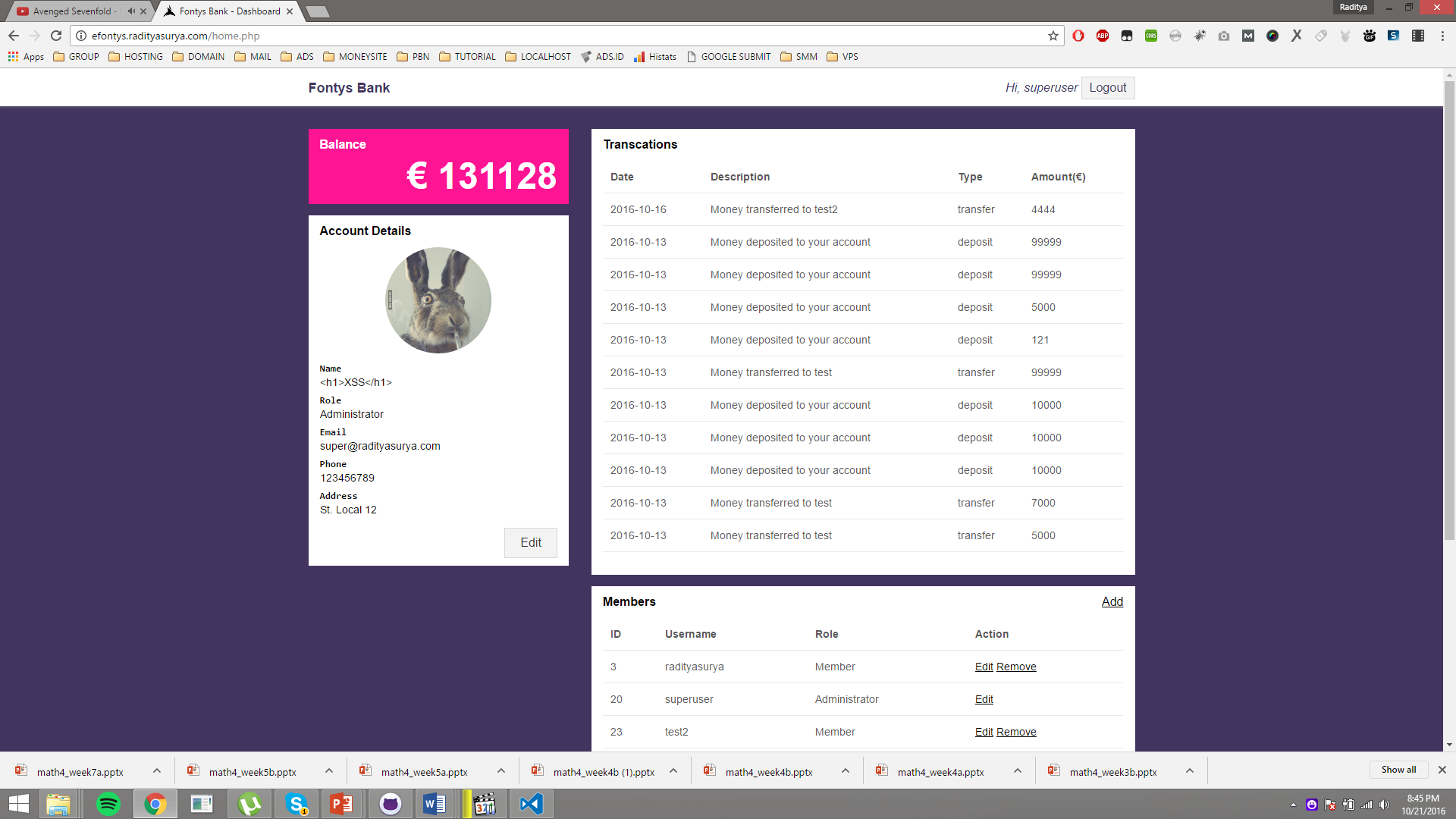


Figure 7 XSS Result

## Brute Force Prevention

Login is very crucial part of our banking system. We need to prevent any harm to this functionality. The step that we took to prevent any hacking activities to this page, is we encrypt the user password with PHP “password\_hash” function and then do the checking when login using PHP function “password\_verify”.

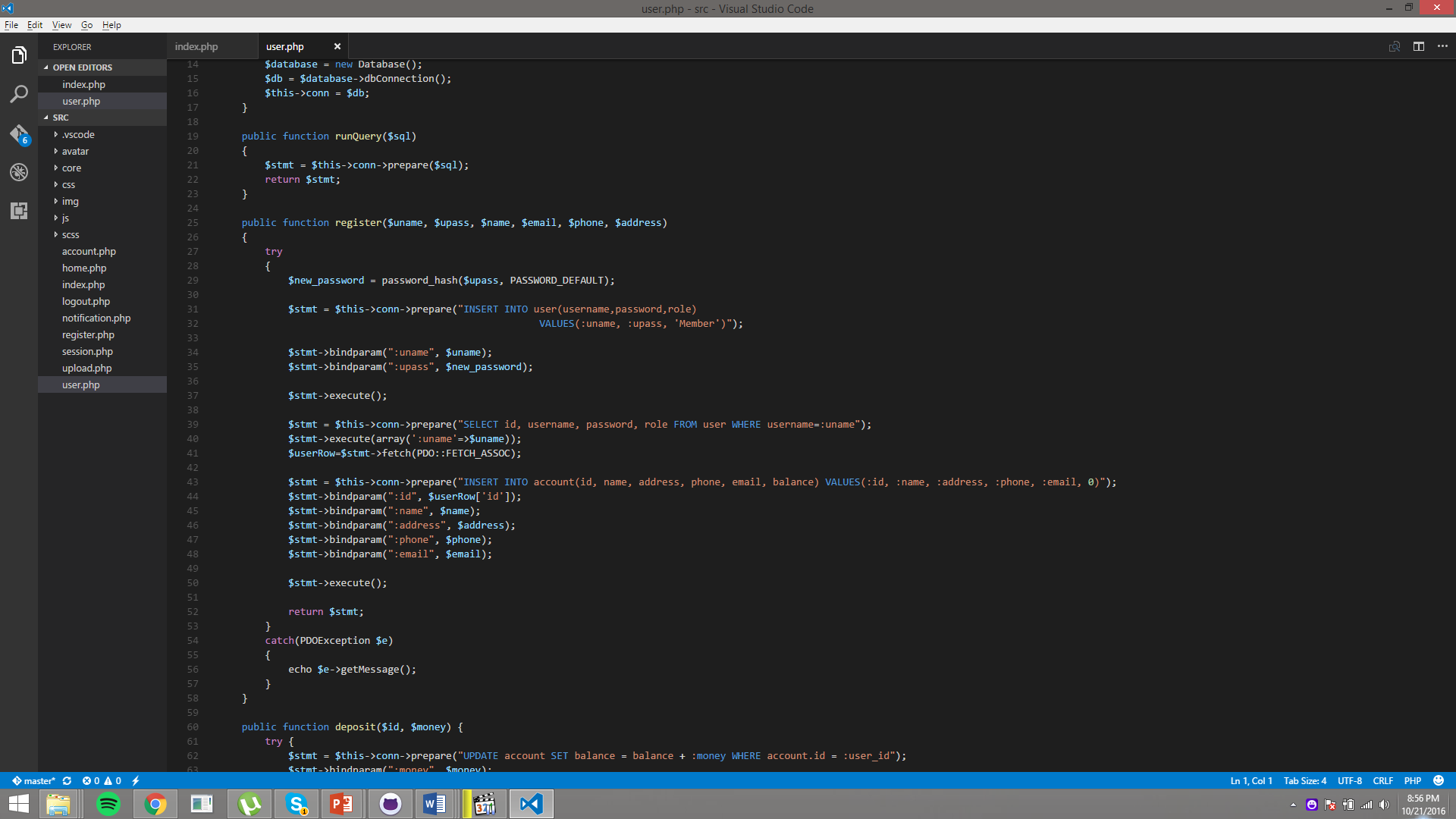


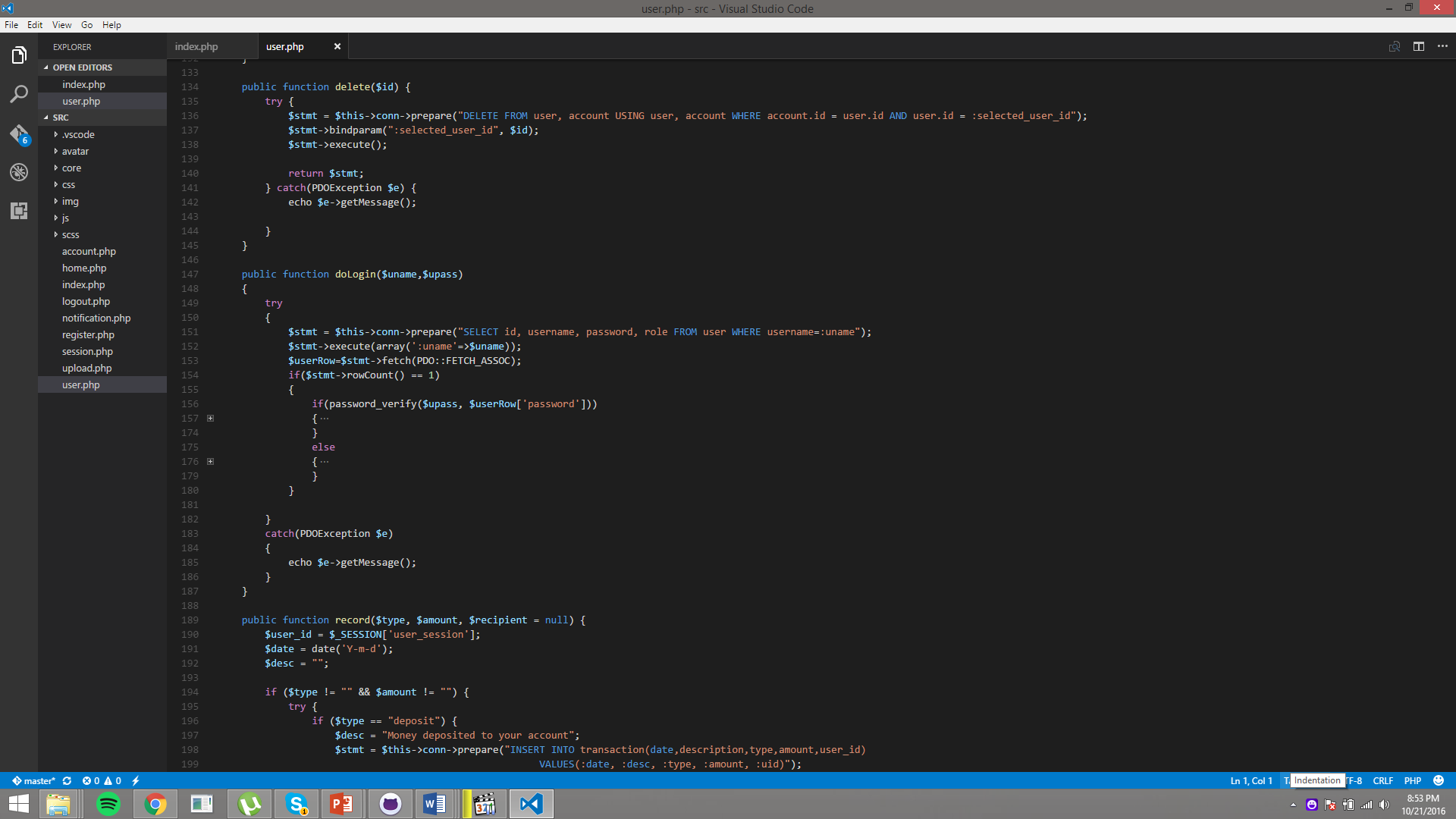
Figure 8 Password Encryption

Figure 9 Password Verifying

After encrypting the password, hackers still could guess the user password. Therefore, we decided to prevent brute forcing password to our login page, with limiting the login attempt to 3 times, and after that the user could not login for 30 seconds.

In this script, the login have a checker function that check the login attempts of this user, and then if the failed login is more than 3 times, it will be blocked for 30 seconds. And resetting the failed login number if the user is entering the correct credentials.

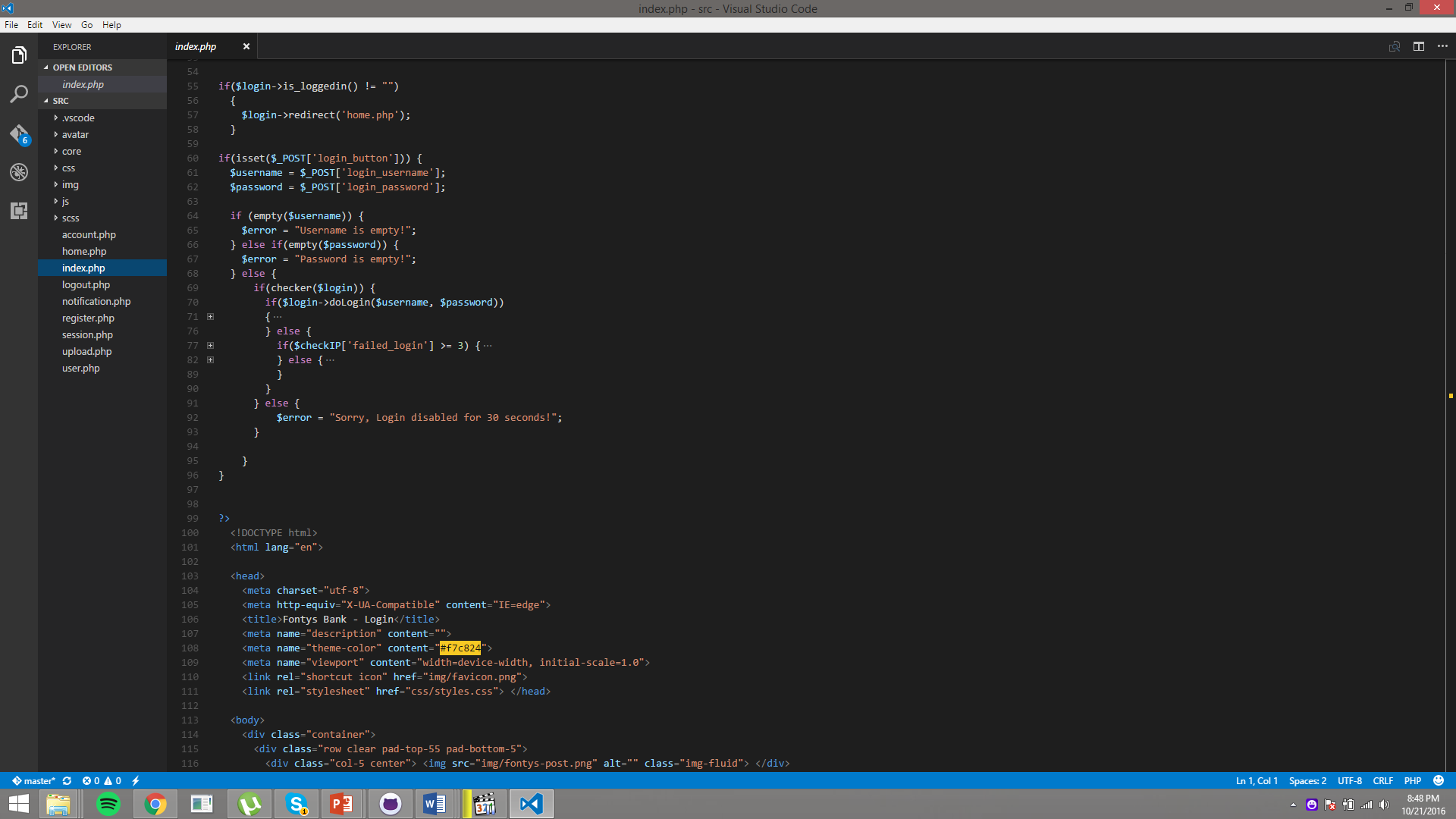


Figure 10 Login Attempt Checker

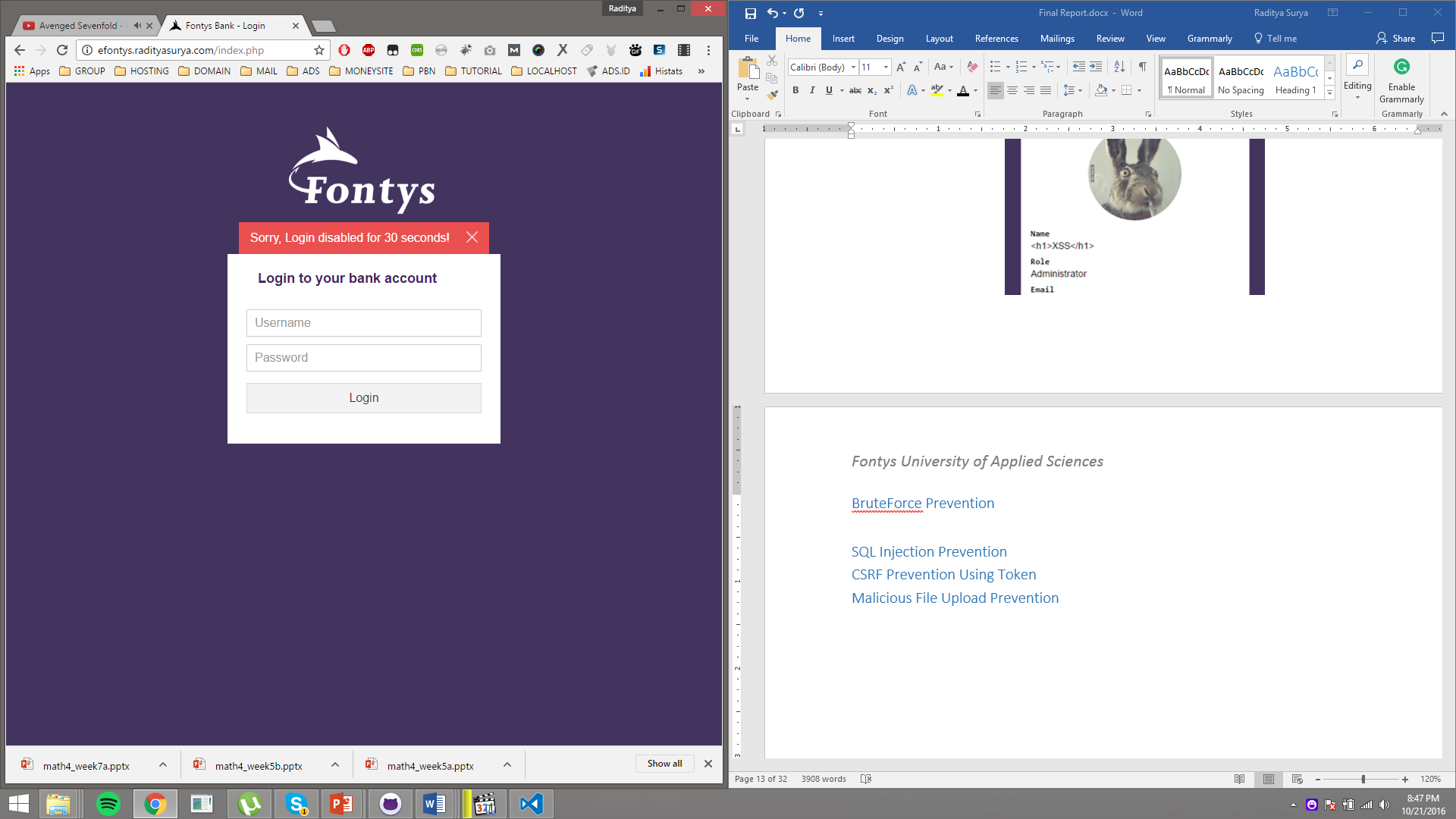


Figure 11 Login Disabled

## SQL Injection Prevention

SQL injection attack allows attacker to insert SQL query via the input data from the user to the website. This attack could expose protected or sensitive data from the database, it could also execute some operation related with the database.

We prevent this attack by using PDO (PHP Data Objects). We use the prepare statement and bind param so the attacker could not manipulate the query. And if the user still could pass the parameter, we still sanitize the input using “mysql\_real\_escape\_string” function.

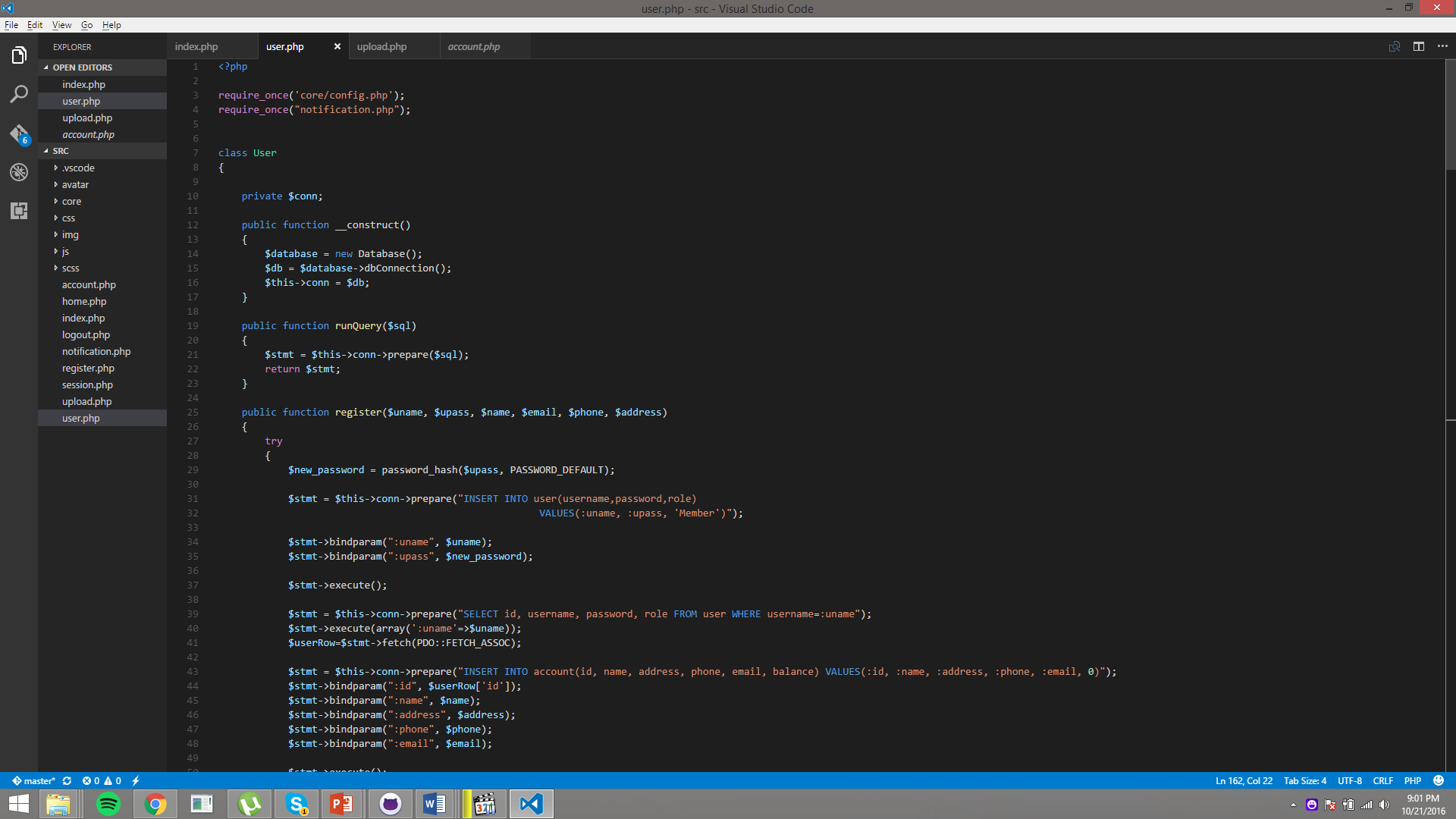


Figure 12 PDO Prepare Statement

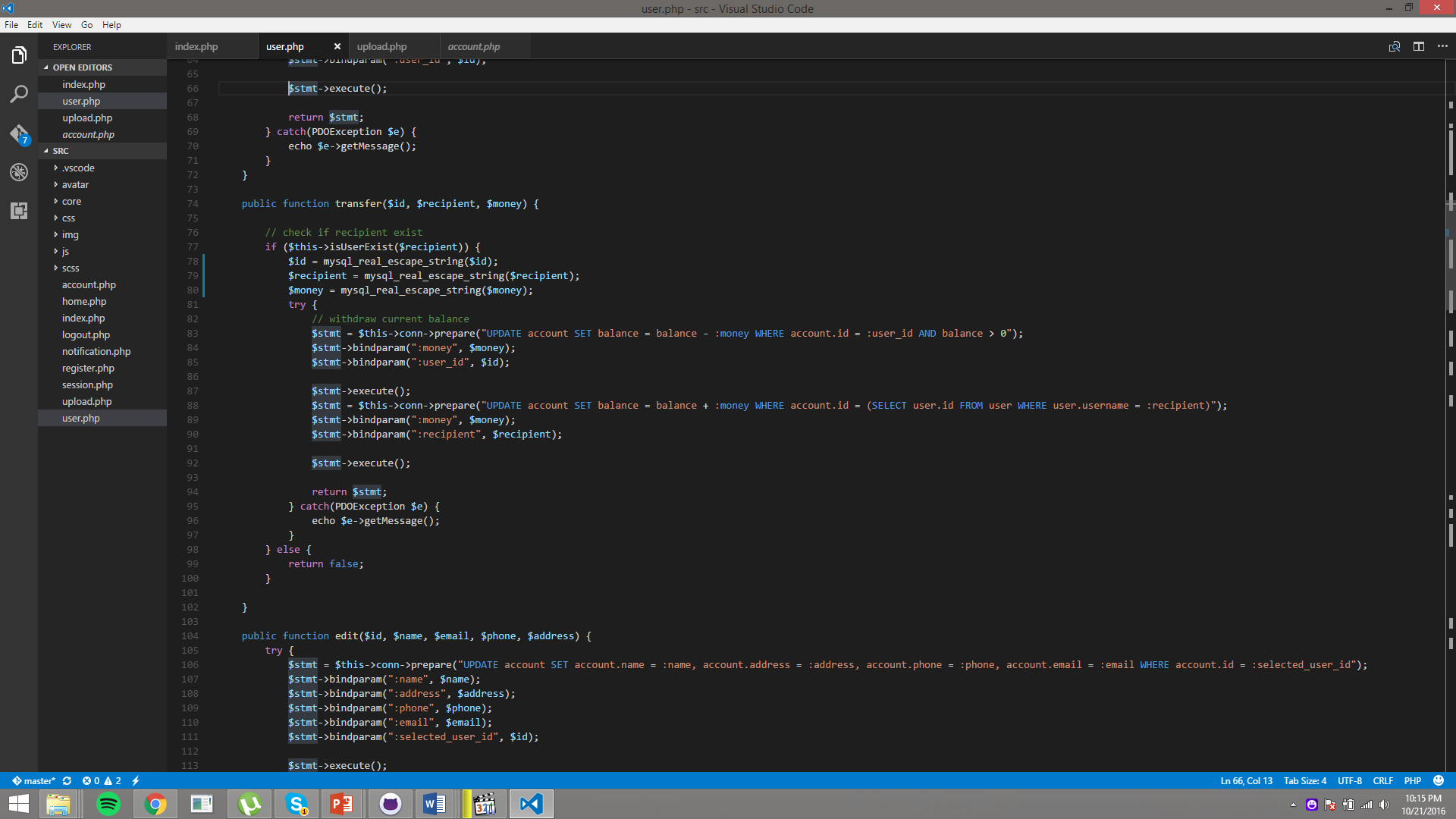


Figure 13 PDO Bind Param and Mysql Escaping

## CSRF Prevention Using Token

Cross-Site Request Forgery is an attack that causes a user’s web browser to perfom an unwanted action on the page that user is currently authenticated. In our case, attacker could transfer money or change information or maybe deposit some money to his/her account.

We prevent this attacks by creating an encrypted token and store it in the user session. When the user is submitting a form, we check the token in the hidden form input is equal with the user session token.

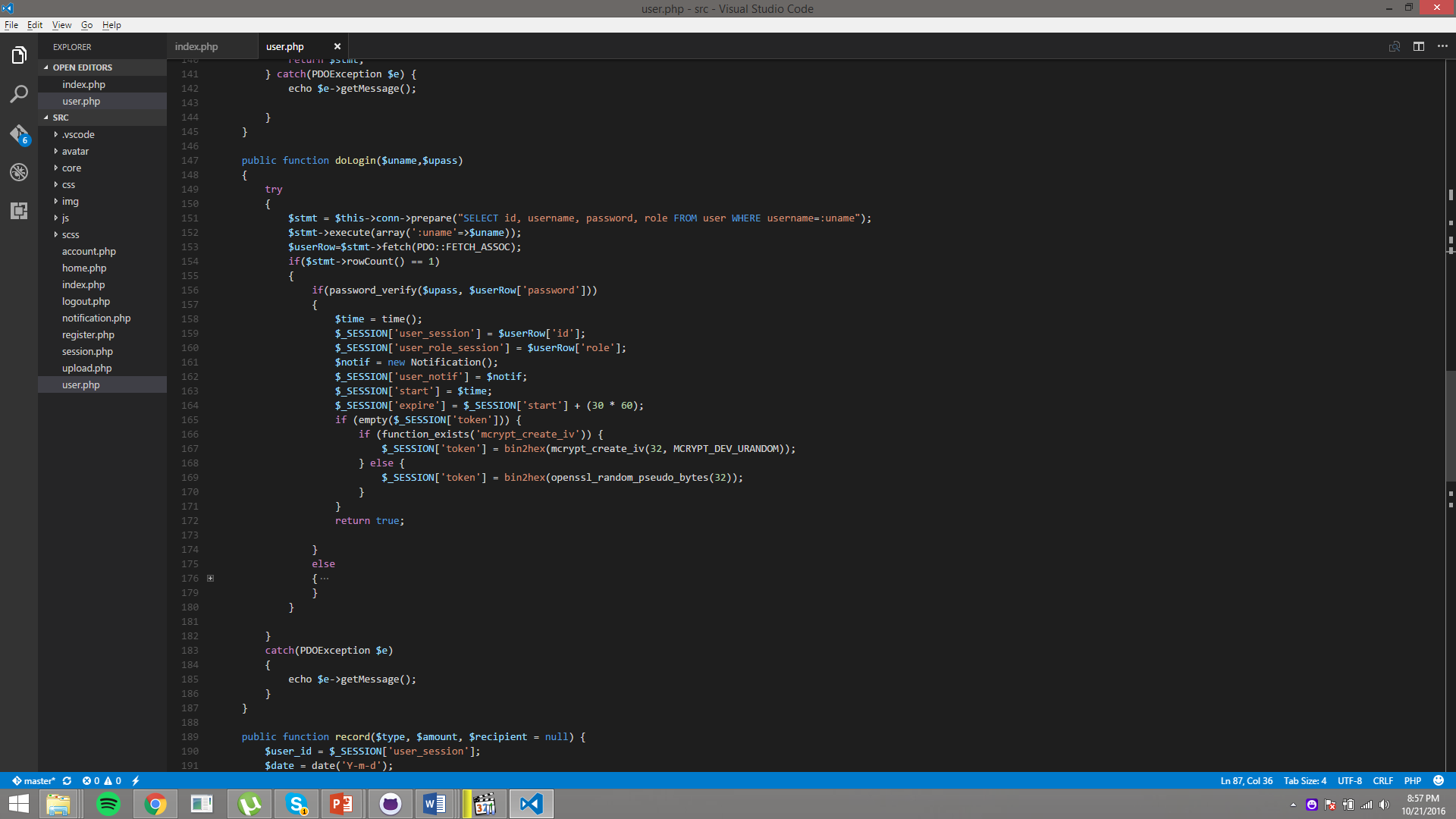


Figure 14 Generating Token Function

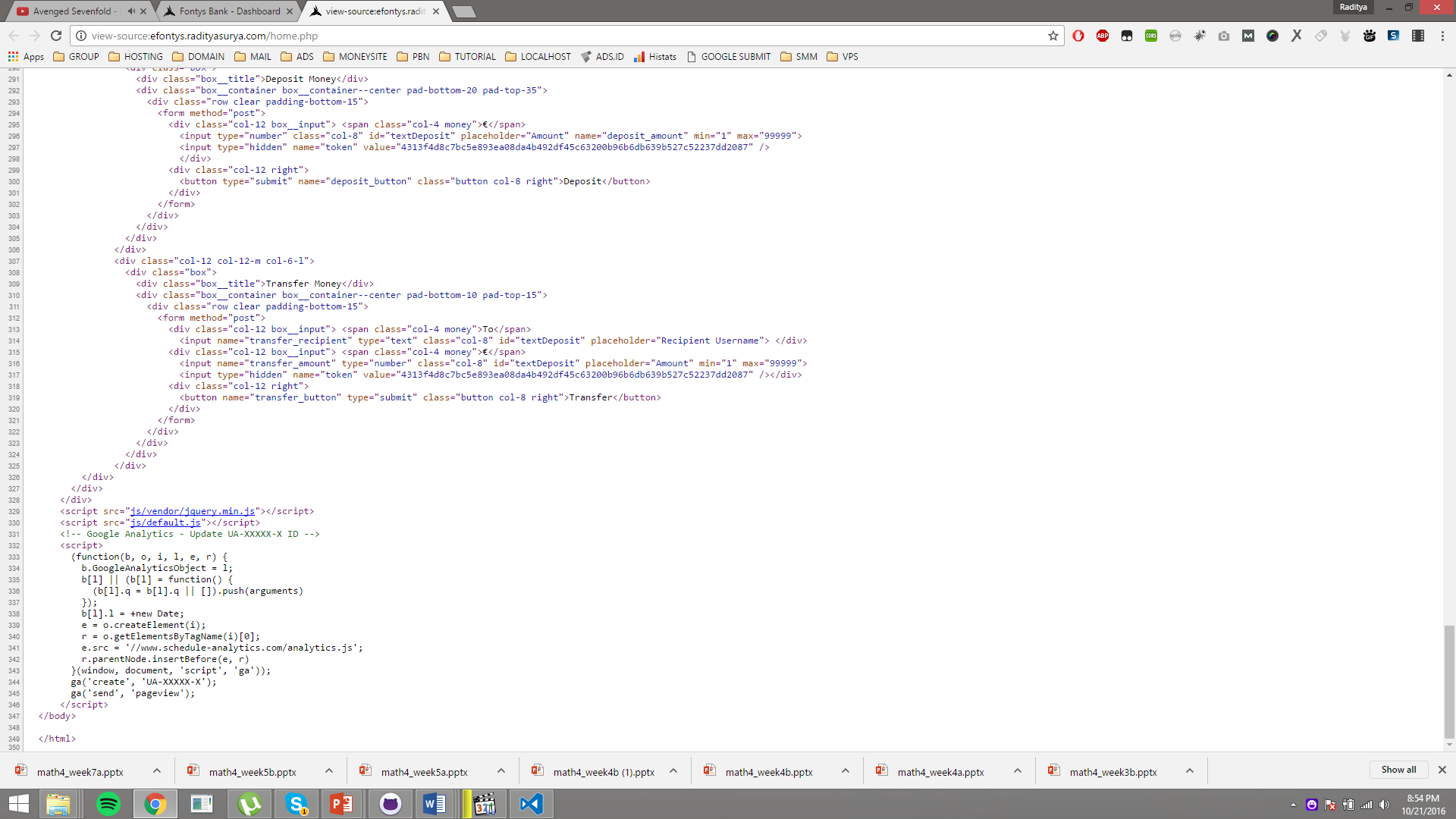


Figure 15 Generated Token

## Malicious File Upload Prevention

One of the feature that we have in our application is uploading avatar. This is also a very vulnerable parts in our website, because if a hacker could upload a malicious file or script, our website will be taken down easily.

To prevent this, we have a function to check the uploaded file extension if the extension is allowed to be uploaded or not. We also check the uploaded file sizes, and will show proper message.

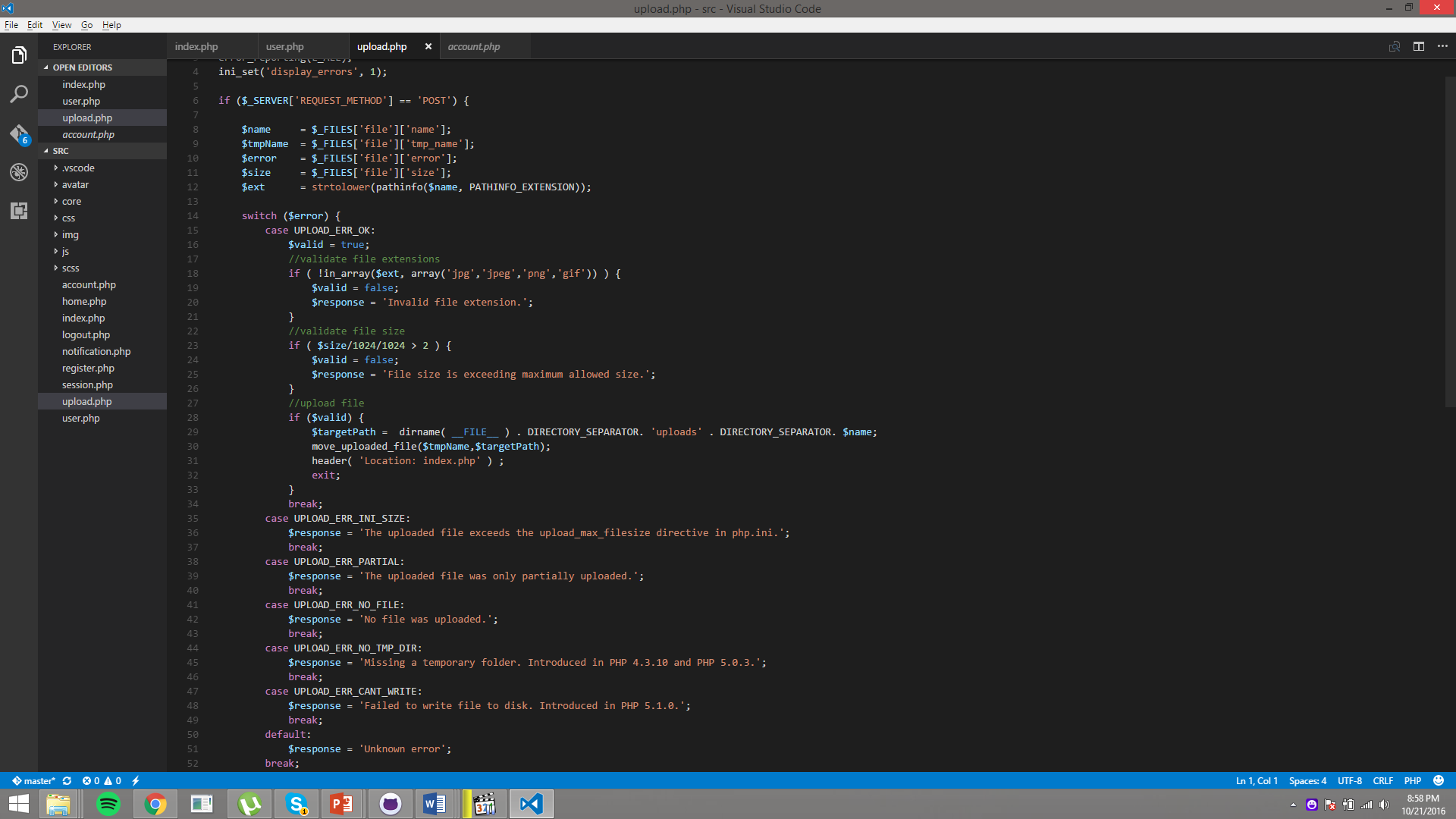


Figure 16 Upload Extension Checking

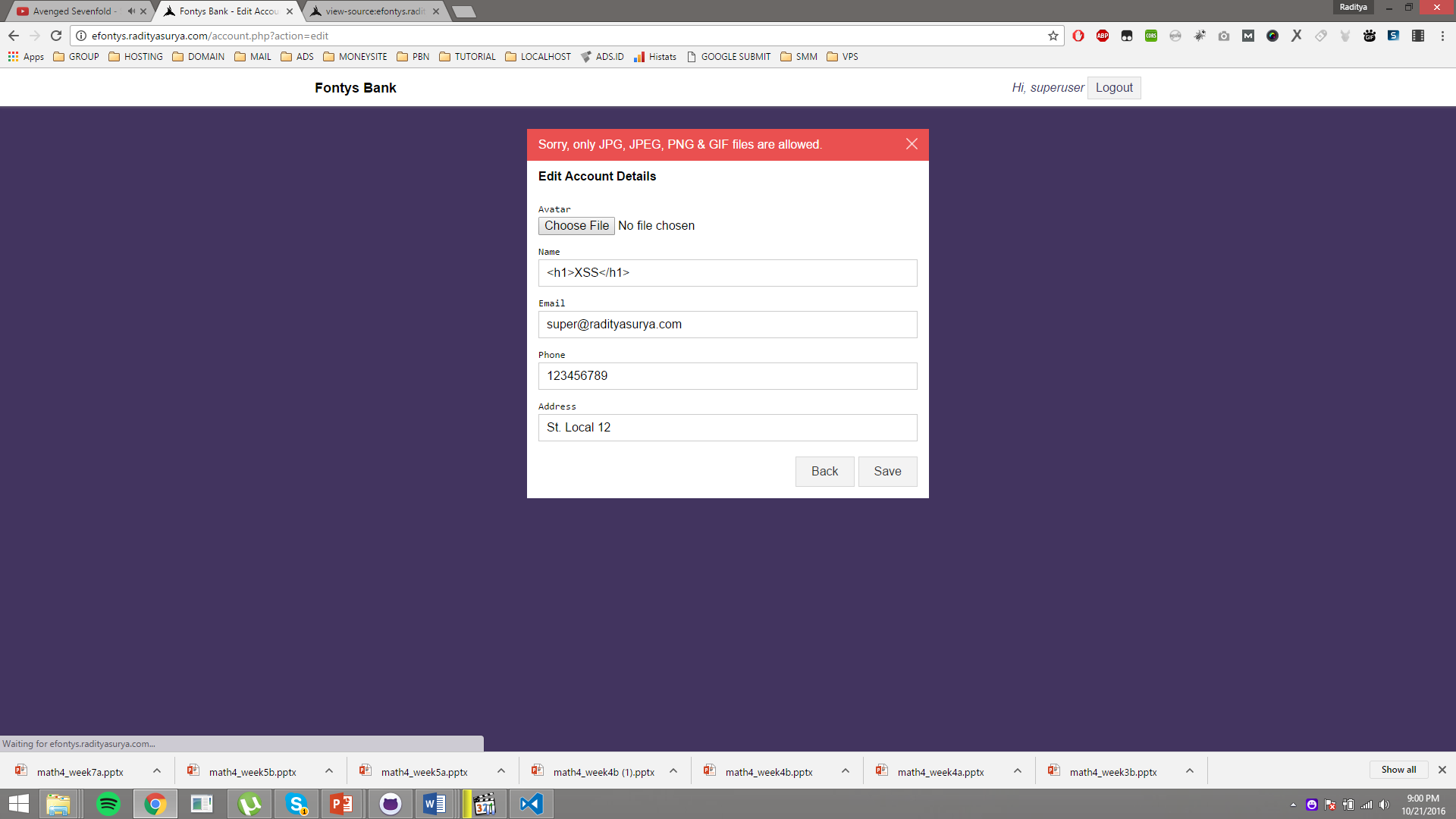
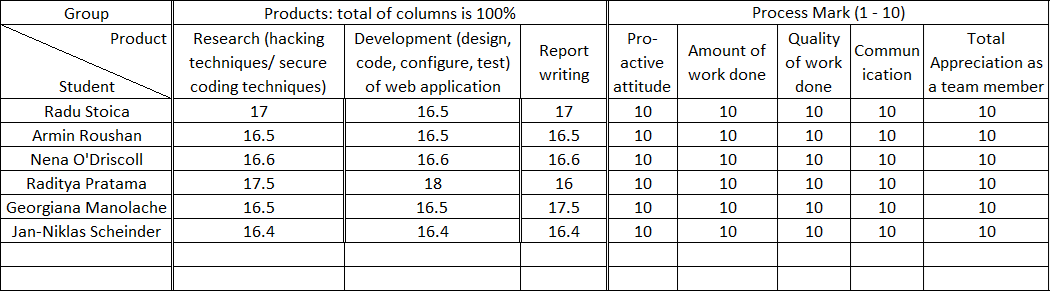


Figure 17 File Extension Error

# Group Peer Assessment



# Personal Evaluation

## Radu

This course has helped me to expand my knowledge about web development and web site vulnerabilities. I have learned how to operate the Kali Linux OS and how to use some of the software provided by this OS in order to analyse and exploit vulnerabilities on a specific website. I think the pen-testing session was really helpful as it gave us the opportunity to apply our knowledge from this course in a “real” environment. Also, the experience of group work for the web application and the presentations has helped me to develop both technical and soft skills. Another very helpful skill achieved during this course is writing more secure software/web application as now I know what kind of attacks to expect on my applications in the future and how to mitigate them.

## Georgiana

This course has helped me raising awareness in the way I handle security in my own code. It has taught me how to tackle, design and implement a higher secure software/web application. I have learned how to ‘think as a hacker’ during the pen-testing session and learn from other people’s mistakes. Thought the course I was also able to improve my team work and presentation skills.

## Armin

Before I started this course, I did not have any knowledge about secure programing and hacking techniques, but this course helped to enlighten my knowledge in this aspect of web development. I also would like to add that it was nice to get familiar with the tools we used during the course and apply some the techniques in practical.

## Jan-Niklas

The course secure programming showed me that security in web applications shouldn’t be taken lightly. An unsecured website can be very easily targeted by people with the right tools and knowledge. The techniques I learned in this course will help me in future building secure web applications and securing it against hackers. Overall, the course gave good insight into a big issue in modern web technology, security, how to protect against it but also giving practical knowledge.

## Raditya Surya Pratama

Security programming course taught me a new point of view for the security areas of web application. I learned a lot about any possible attacks and vulnerabilities that a website could have. I also learned a few hacking techniques to get familiar with it, and also learned how to prevent that kind of attacks on my own website. This course is really helpful for my future project, especially to prevent bugs and hacking possibilities.

## Nena

This course not only taught me that exploiting on the web isn’t just for Eliot’s or other modern day wizards, it reconfirmed how every team member has a valuable different view of solving a problem. The problem in this case being noticing vulnerabilities, and consequently either exploit or patch them. In future projects I hope to pay far more attention to the security of the application, as it is indeed a wild wide web.

# Recommendation and Evaluations

Our website has simple functionalities, for the future, we might implement more feature and functionalities. We should try to create more complex website with more mature functionality, and did the security testing to higher level.

We figured that the website that we thought was secure enough is still having some flaws. We also learned various types of hacking attack in the pen-testing and we also learned a lot in the preventing those attacks.

# Project Conclusion

We believe that the project for SePr was successful. The pen test results of our website showed that we have a secure website and only have minor vulnerabilities that could be fix easily.

# References

*2009 Data Breach Investigations Report*. (2009). Retrieved from Verizon Enterprise: http://www.verizonenterprise.com/resources/security/reports/2009\_databreach\_rp.pdf

*Data Breach Report*. (2015). Retrieved from ID Theft Center: http://www.idtheftcenter.org/images/breach/DataBreachReports\_2015.pdf

*Top Online Banking Threats to Financial Service Providers in 2010* . (2010). Retrieved from http://www2.gemalto.com/: http://www2.gemalto.com/email/2010/mitb-2010/whitepaper\_top-threat-to-financial-service-providers-in-2010\_final.pdf

WhiteHat Security. (2015). *Website Security Statistics Report*. Retrieved from Whitehat Security: https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf

*World's Biggest Data Breaches*. (2016, September). Retrieved from Information is beautiful: http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

# Appendix A: eFontys Pen-testing Report

This test report is written by User Blog Service Group.

## Introduction

In order to develop a secure web application, it must go through various testing. In order to know how secure the application is Pen-testing is done. Various tests are done in order to get the vulnerabilities of the application. Some of them are mentioned in this report.

Different tests were performed for the vulnerabilities of the application. Some of them are mentioned below:

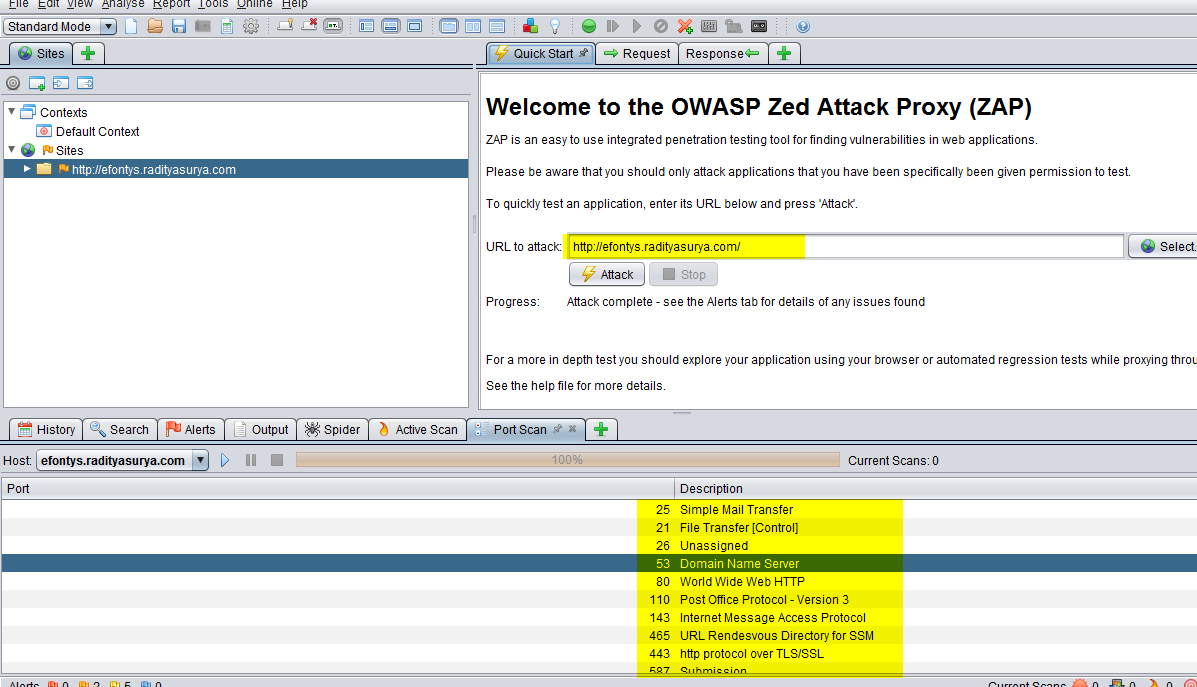
### Portscan Server

Port scan are usually used by hacker to find out the open port.

**Tools used**: OWASP ZAP Port Scanner

**Action:** The url of the website http://efontys.radityasurya.com/ is added to the tool and checked for open port.

**Result:** We were able to see all the open port.



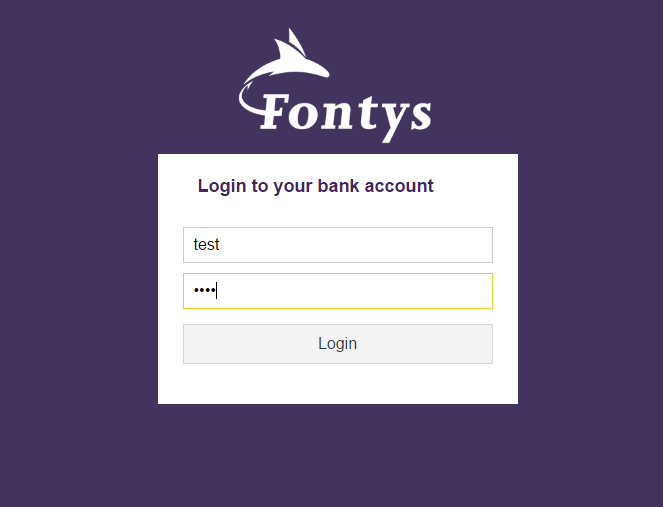
**Conclusion:** You can use some tools to prevent the port scanner.

### Forced Browsing

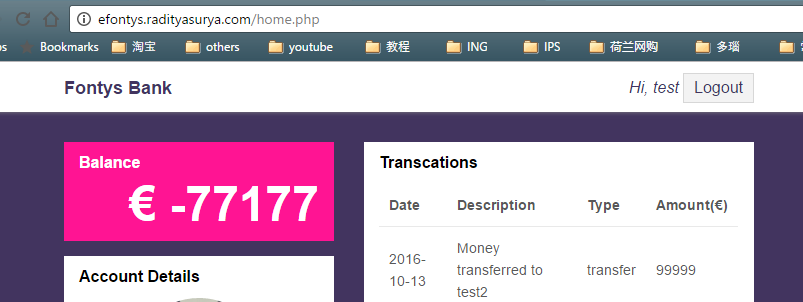
Forced browsing is an attack where the aim is to enumerate and access resources that are not referenced by the application, but are still accessible. Forced browsing is a simple browser attack that attempts to circumvent these controls by requesting authenticated areas of the application directly, without providing valid credentials, or by requesting pages beyond the access level of the logged-in user. If permissions on these pages have not been configured correctly, the pages will be displayed to the unauthorised user.

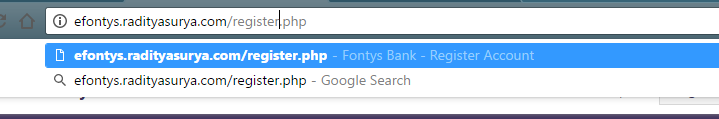
**Action:** The user is logged in with his/her normal account. And changed the url .

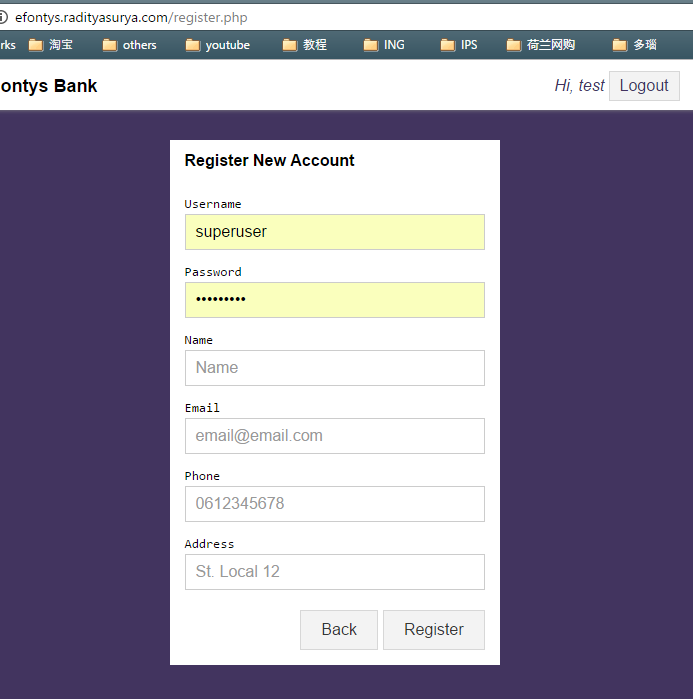
**Result:** We were able to login to register page.



Change the url from “home.php” to “register.php”







**Conclusion**: Now you can add a new account, which you should not be possible to since you are only a normal user not a super user.

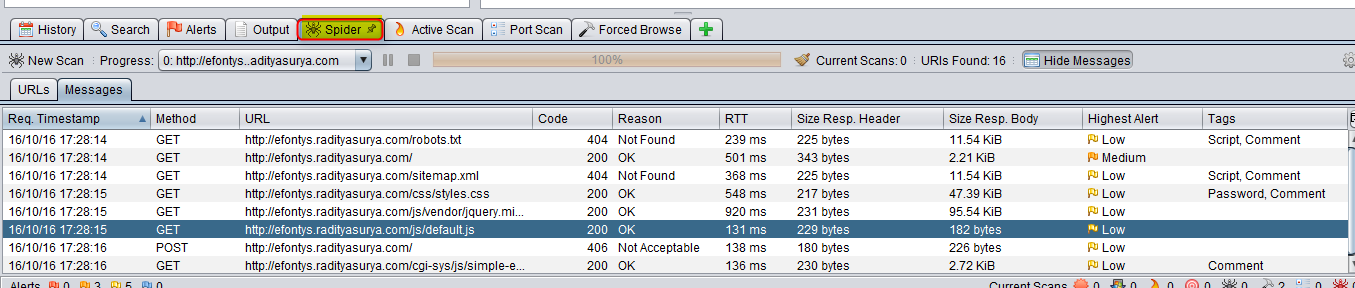
### Spider web application

Spider web application means crawling through the number of pages within a website which are not available for public users.

**Tool used**: OWASP ZAP Spider

**Action:** The url of the website http://efontys.radityasurya.com/ is added to the tool and checked for pages.

**Result:** We can crawl through number of pages.



## Testing for Vulnerabilities

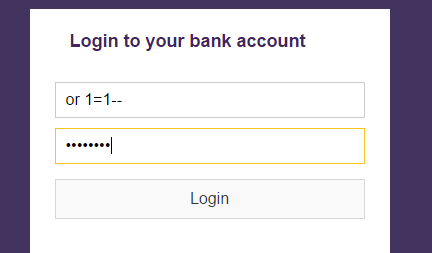
There are different types of injection attack we would use some of them are listed below:

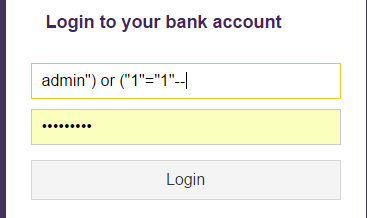
### Testing for SQL Injection

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system

**Action**: Different type of sql injection were enterd.

**Result:** Cannot access using sql injection.





**Conclusion**: The sql injection is well prevented.

### Testing for XSS

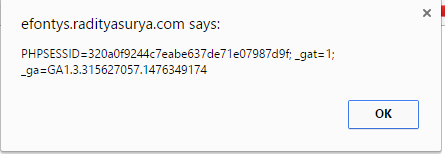
A cross-site scripting vulnerability allows the injection of malicious client-side scripts on a website, which is then served to users (client’s web browser)

Stored XSS:

**Action**: Tired to input XSS script transfer recipient name for cookies information.

**Syntax:** Hi <script>alert(document.cookie)</script>

**Result:** We got message cookie informations.

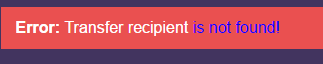


**Conclusion**: We can do XSS cookie stealing.

#### Reflected XSS:

**Action:** We input <font color=”blue”>

**Result:** We can see that the font change from white to blue.



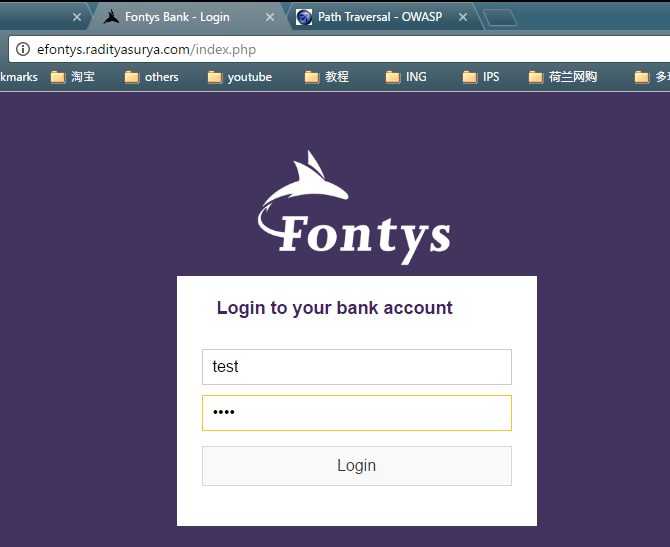
**Conclusion**: We can input script to change color.

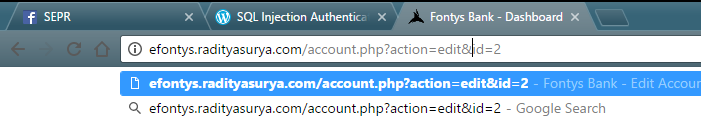
### Testing CSRF / XSRF

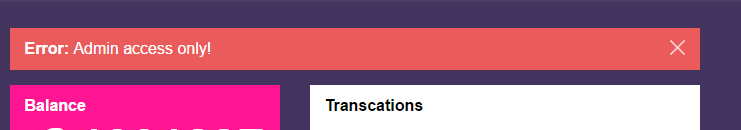
Cross-Site Request Forgery (CSRF) is an attack that tricks the victim into loading a page that contains a malicious request to perform an undesired function on the victim's behalf, like change the victim's e-mail address, home address, or password, or purchase something.

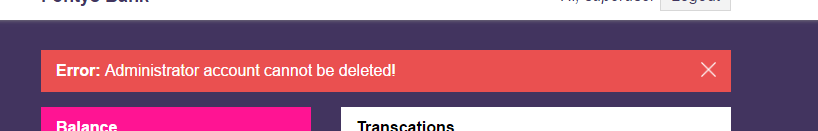
**Action:** We try to modify some account’s information by a normal user, which should only be possible by super account. Login with a normal user and the change the url.

**Result:** Error message was shown.









**Conclusion:** CSRF is also well prevented.

### Testing for File Inclusion

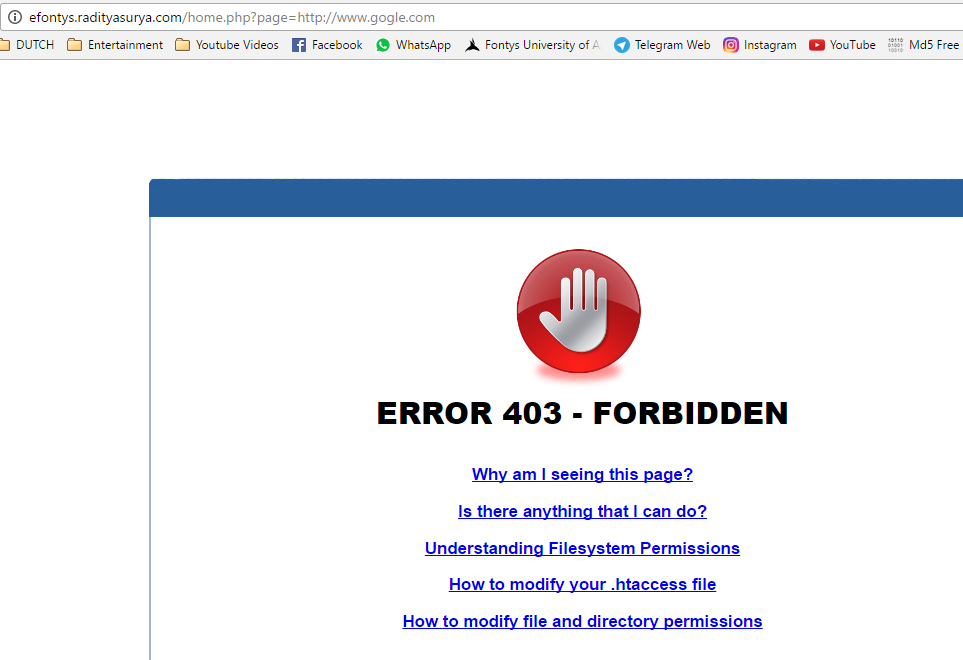
File inclusion vulnerability is a type of vulnerability most often found on websites. It allows an attacker to include a file, usually through a script on the web server. The vulnerability occurs due to the use of user-supplied input without proper validation.

#### RFI

RFI is the vulnerability found in web site that allows attackers to include remote file on the web server.

**Action:** We tried to change the url and add another url to the old url.

**Result:** Cannot execute another page.



**Conclusion:** Cannot add the Remote file inclusion.

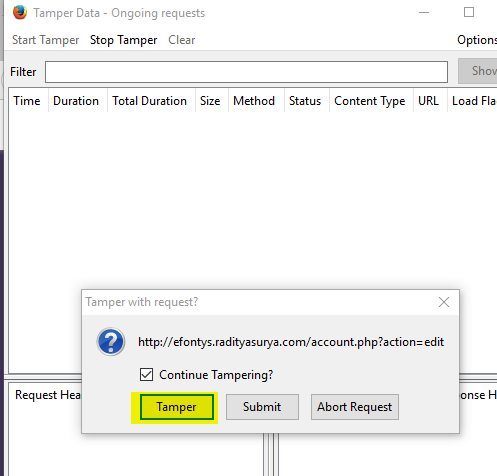
### File Upload

Uploading the malicious file such as shell.php is one the most attack which the attacker will do. We can upload php file by tampering the file extension and upload it the website and can get the whole information about the website database, etc.

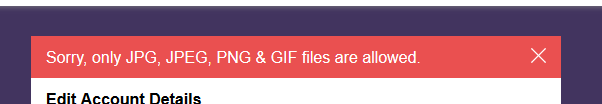
**Tool use**: Tamper Data add on for Mozilla FireFox.

**Action**: We try to upload the shell.php file to the upload image by using tamper data tool.

**Result:** We get error saying the extension only image files are allowed.







**Conclusion**: We cannot insert the php file. It is well secured.

## Conclusion

Overall, the application is secured. Most of the inputs are sanitized, so we cannot input any data except the required format.

# Appendix B: Cloud Service Pen-testing Report

This test report is written by eFontys Group.

## Introduction

The goal of this document is an overview of the performed pen testing on the web site cloud service. The document is structured into, initially, describing goals that are desired in the pen test, following its results.

## Goals

The goal of the penetration test is to perform attacks upon the web application and to expose its vulnerabilities.

We will start off by gathering information (e.g. through port scans and DNS discoveries) and using that to perform the following tests:

The following are tests we will perform against our opponent’s website:

1. SQL Injection Vulnerabilities:
   1. We will test for input vulnerabilities
   2. URL parameters – include malicious JavaScript or SQL statements in the URL
2. Cross Site Scripting
   1. We will use input vulnerabilities to embed SQL scripts or JavaScript that will execute unauthorized code.
3. Path Traversal
   1. We will try and find files by navigating the file structure of the website.
4. Upload malicious files into the cloud service.

## Results Analysis

|  |  |
| --- | --- |
| Vulnerability ID: | Path Traversal |
| Vulnerability Title | Path Traversal |
| Severity(1”low”-5”high”) | 3 |
| Summary | Prevent server side information from being displayed |
| Description | Vulnerability found using Nikto software: /SePrWebsite/index.php/sips/sipssys/users/a/admin/user: SIPS v0.2.2 allows user account info (including password) to be retrieved remotely. |
| Recommendation | Use routing and update the SIPS. |
|  |  |

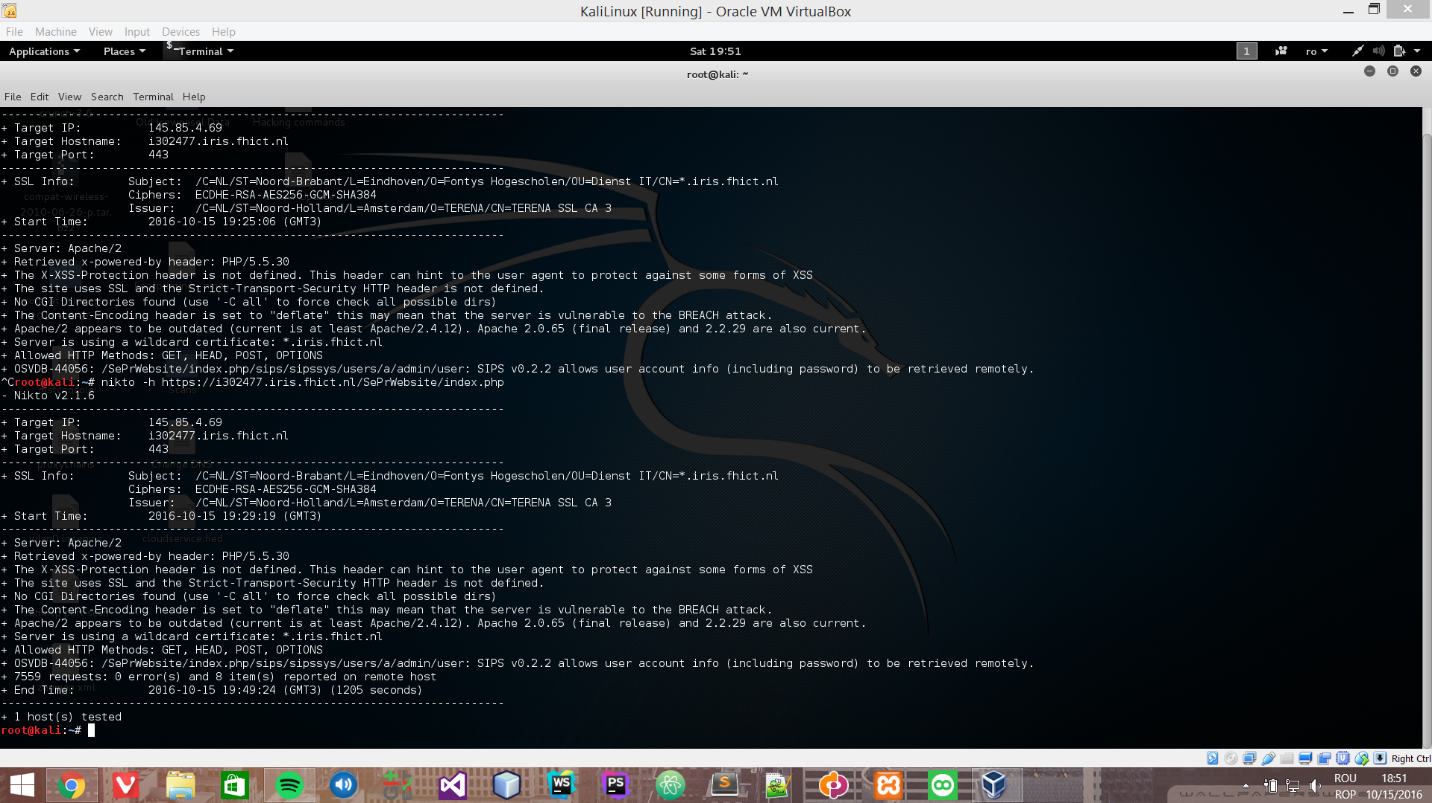


Figure 0‑1: Result of Path Traversal

|  |  |
| --- | --- |
| Vulnerability ID: | Brute Forcing – Disabling account |
| Vulnerability Title | Brute Forcing – Disabling account |
| Severity(1”low”-5”high”) | 1 |
| Summary | An account allows for three login fails, after which the account is disabled. To recover the account, the owner must contact the administrator. This allows for an attacker to disable many accounts if he brute force usernames (e.g. ‘admin’). |
| Description |  |
| Recommendation | Allow the user to more easily recover their account and replace disabling with a time-out |
|  |  |

|  |  |
| --- | --- |
| Vulnerability ID: | OWASP Vulnerability check - HTTP Headers |
| Vulnerability Title | XSS Vulnerability revealed by OWASP ZAP |
| Severity(1”low”-5”high”) | 4 |
| Summary | Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server |
| Description |  |
| Recommendation | Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. |
|  |  |

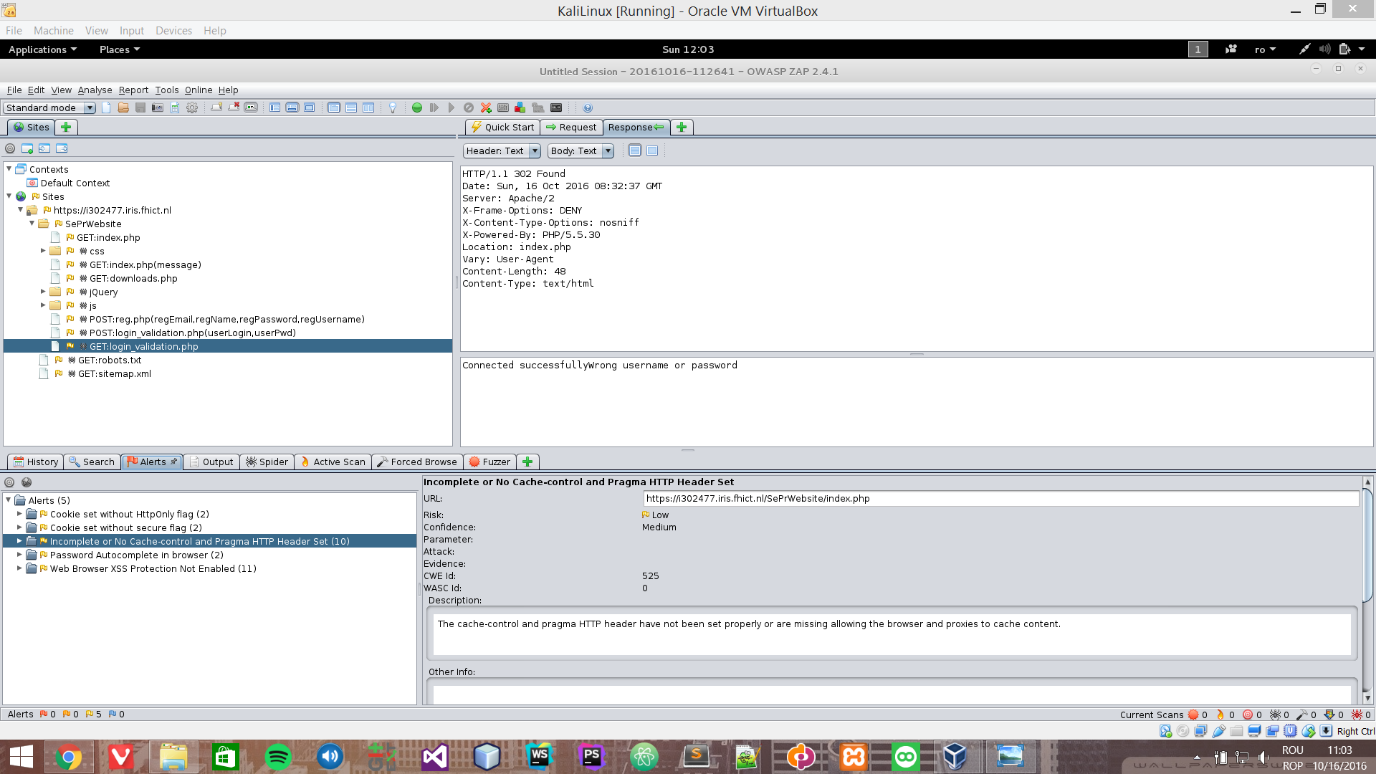


Figure 0‑2: Result OWASP

|  |  |
| --- | --- |
| Vulnerability ID: | Upload malicious files. |
| Vulnerability Title | Upload malicious files. |
| Severity(1”low”-5”high”) | 5 |
| Summary | We were able to upload a shell as a php file on the Cloud server and we got access to the whole server. |
| Description |  |
| Recommendation | Prevent the malicious files from being uploaded on the server. Also, check for the hidden ones under picture files. |
|  |  |

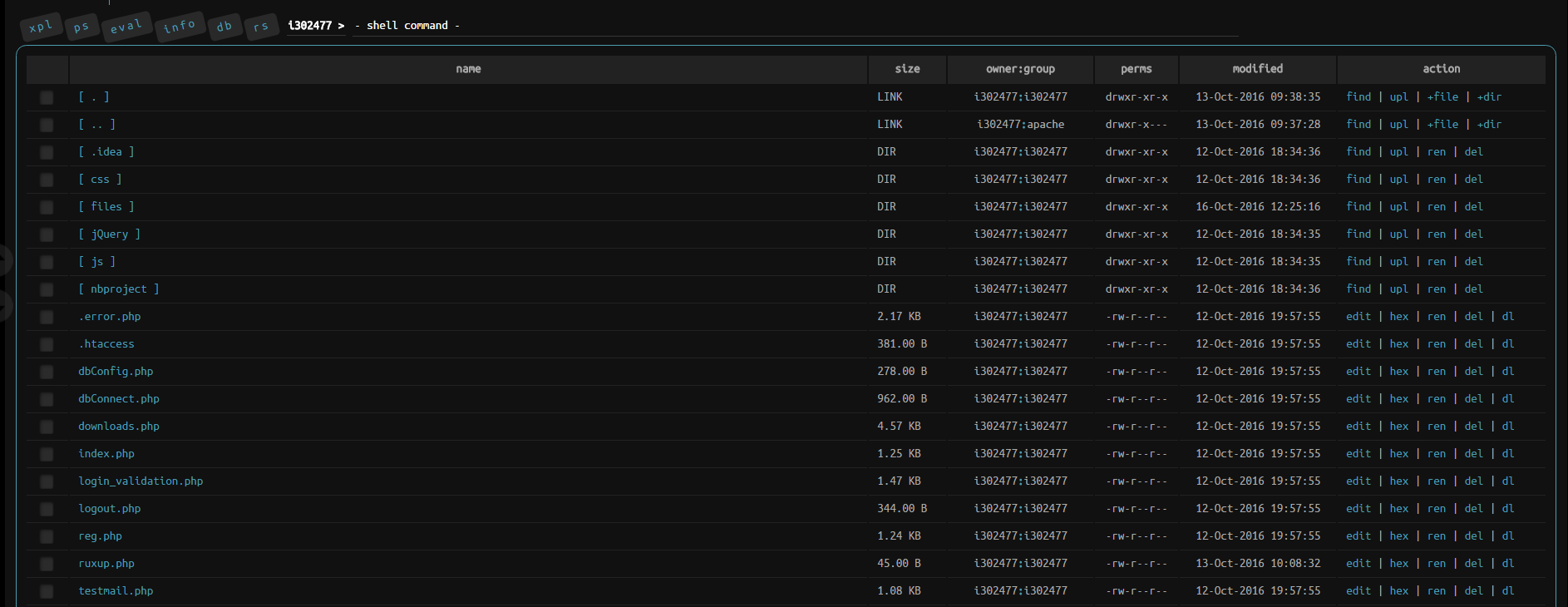


Figure 0‑3: File structure of the server

## Changes on The Server

We uploaded a php file called ruxup.php on the server.

We analyzed the web application using a couple of kali linux software and we couldn’t find any sql injection or XSS vulnerabilities. Also, we tried to use the sqlmap on a high level and there was no success.

*sqlmap -u https://i302477.iris.fhict.nl/SePrWebsite/index.php/sips/sipssys/users/a/admin/user/login\_validation.php?userLogin=admin --dbs --level=3 --risk=3 - no vulnerabilities*

*sqlmap -u https://i302477.iris.fhict.nl/SePrWebsite/index.php/sips/sipssys/users/a/admin/user/login\_validation.php?userPwd=admin --dbs --level=5 --risk=3 --dbms=mysql - no vulnerabilities.*

## Other Screenshots From The Analysis

* Using Burp suite. Tried some brute force as well.

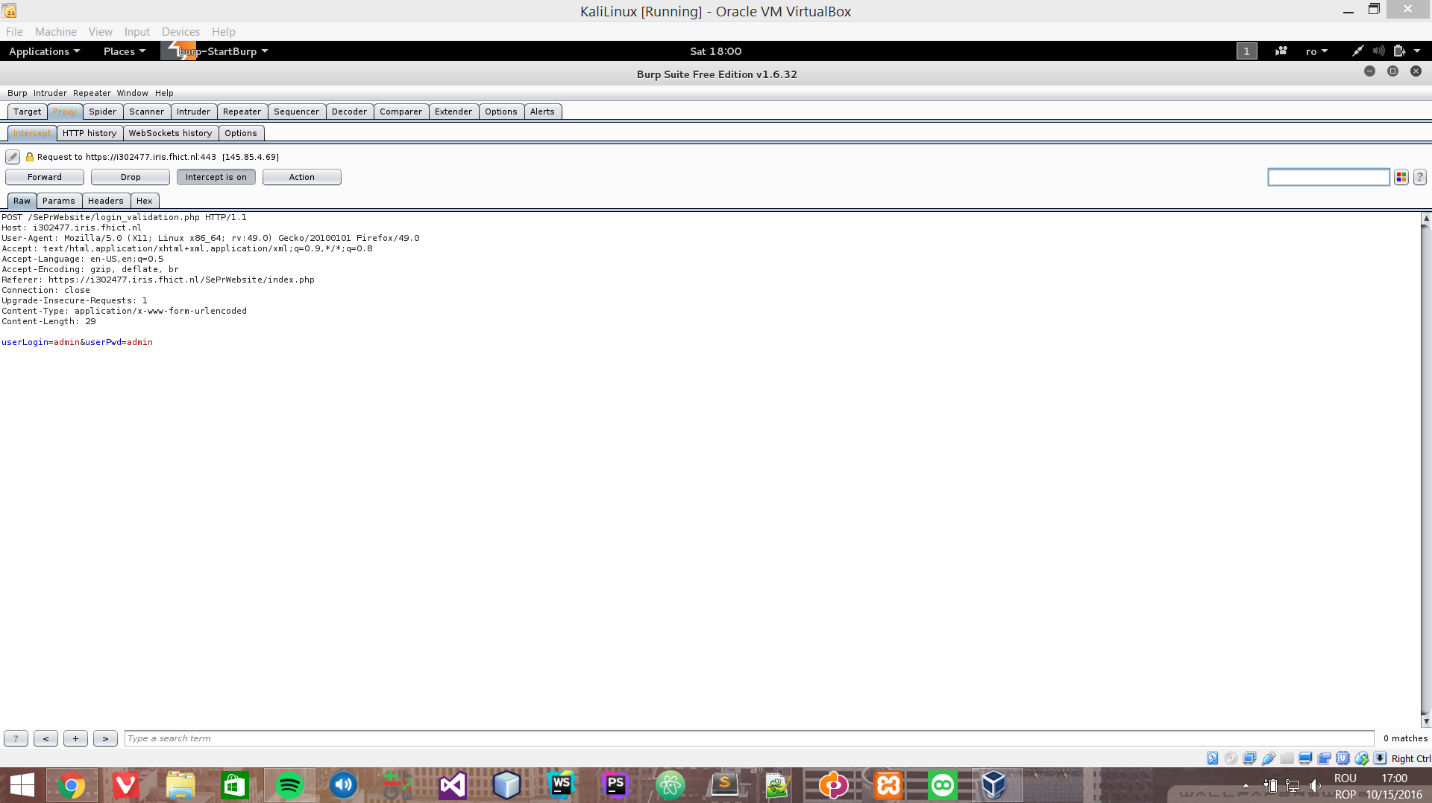


Figure 0‑4: Result of brute force with burp suite

* Sqlmap – no success.

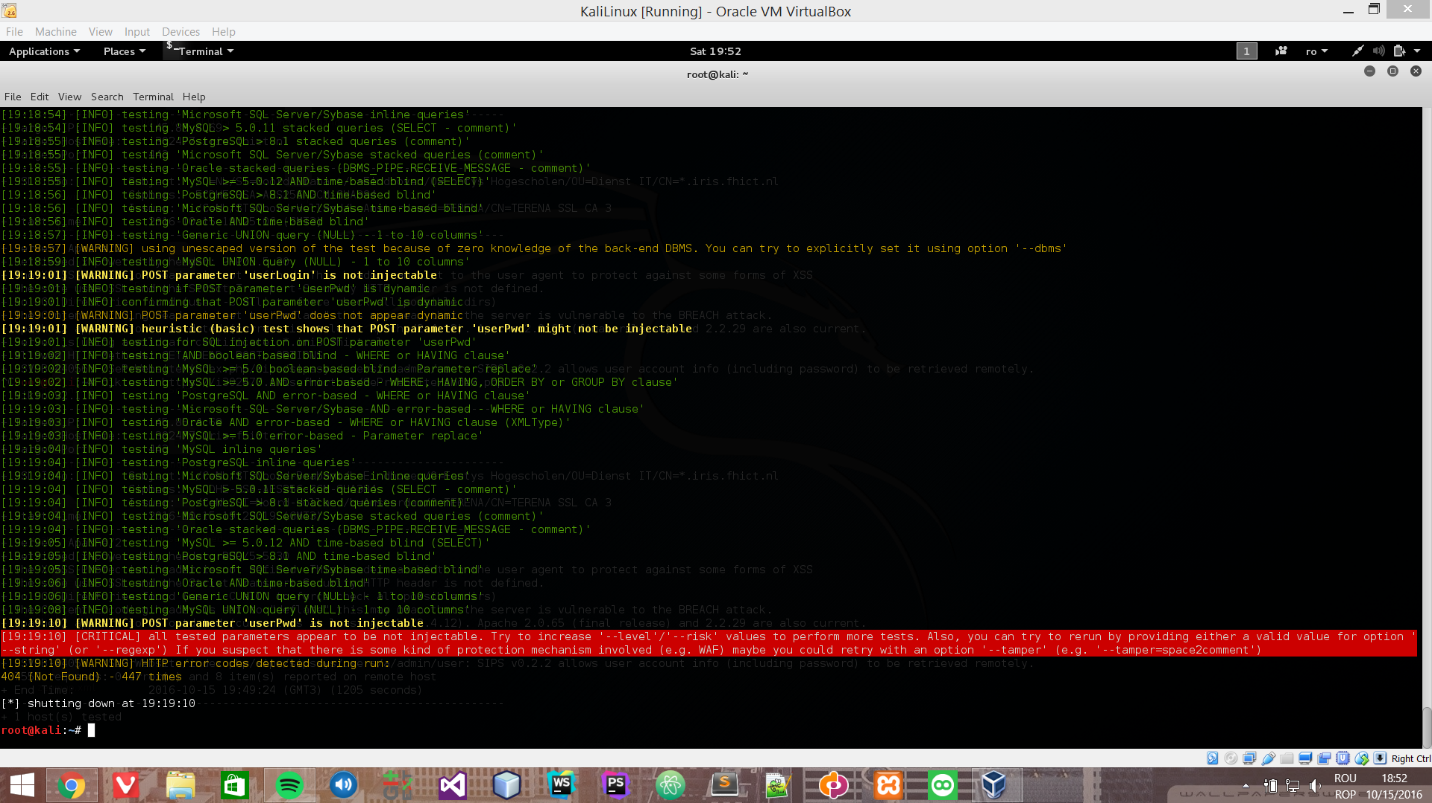


Figure 0‑5: Result Sqlmap

* Zenmap port scan – found some ports opened.

Scanning 145.85.4.69 [1000 ports]

Discovered open port 110/tcp on 145.85.4.69

Discovered open port 143/tcp on 145.85.4.69

Discovered open port 995/tcp on 145.85.4.69

Discovered open port 80/tcp on 145.85.4.69

Discovered open port 587/tcp on 145.85.4.69

Discovered open port 993/tcp on 145.85.4.69

Discovered open port 25/tcp on 145.85.4.69

Discovered open port 443/tcp on 145.85.4.69

Discovered open port 21/tcp on 145.85.4.69

Discovered open port 465/tcp on 145.85.4.69

Discovered open port 2222/tcp on 145.85.4.69

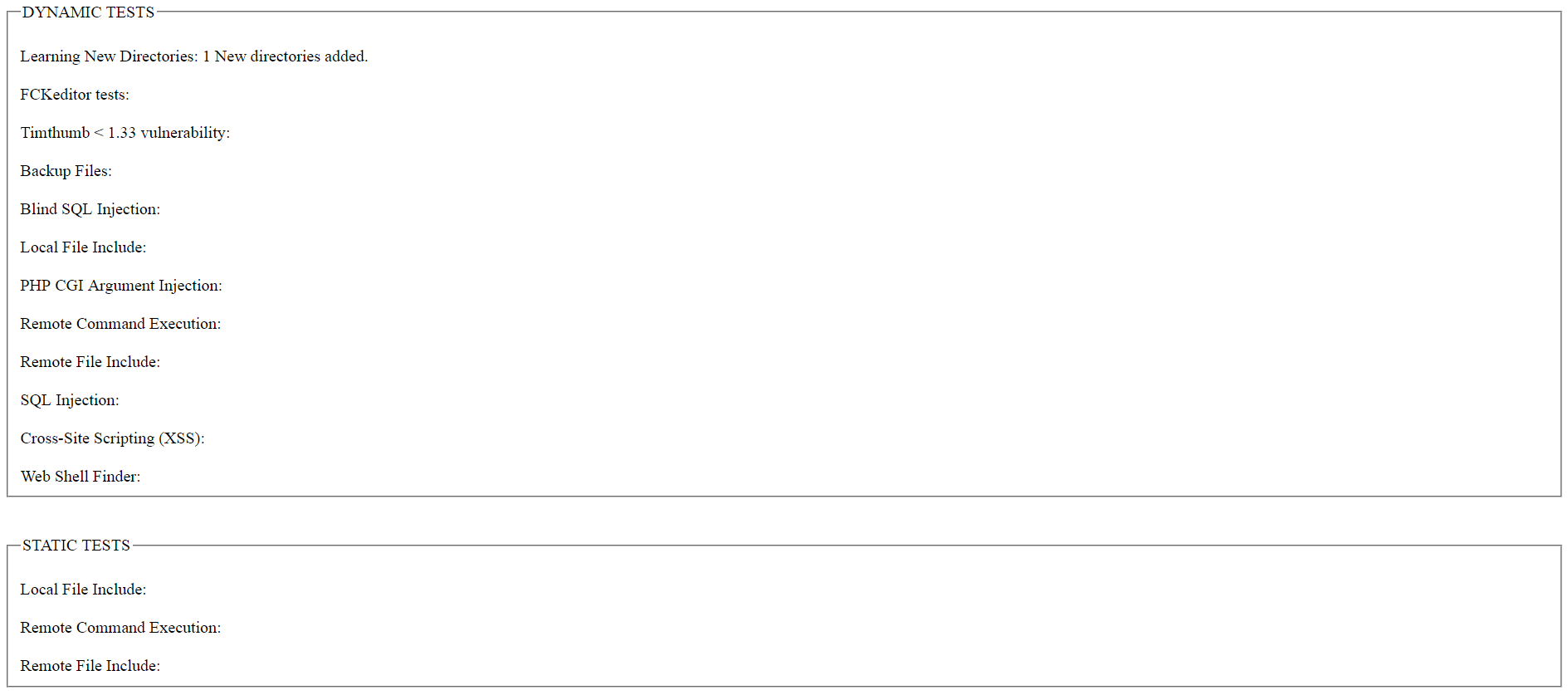
Discovered open port 563/tcp on 145.85.4.69

Discovered open port 119/tcp on 145.85.4.69

Discovered open port 53/udp on 145.85.4.69

Discovered open|filtered port 53/udp on 145.85.4.69 is actually open

* Uniscan report – no vulnerabilities found



* File uploading

The problem with the code is that there is no check regarding type of file being uploaded. Assuming that pictures/ is available in the web document root, a malicious file can be added. Further, the content of a uploaded file is not being verified which allows malicious content to be included.

malicious.php:

<?php

system($\_GET['cmd']);

?>

1. Once this file has been installed, the one can enter arbitrary commands to execute using a URL such as:

http://serveraddress/uploads/malicious.php?cmd=ls%20-l

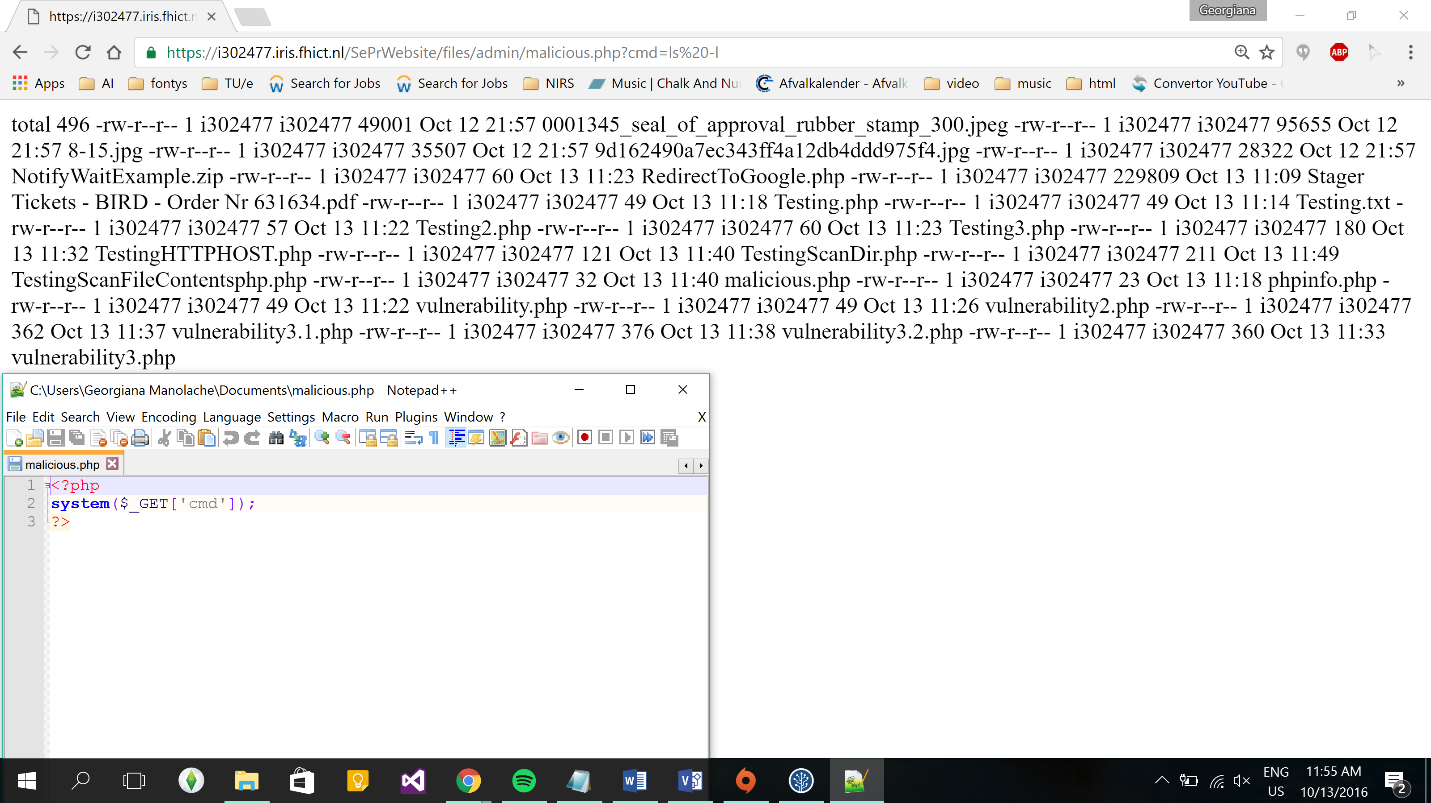


Figure 0‑6: Result of a command

## Conclusion

The pen test showed that the cloud service is to be very secure in terms of XSS or SQL injection attacks. It was not possible to perform a successful attack using the techniques named.

However, the cloud service proved to be very vulnerable to malicious files. Not the content nor the file type of uploaded files are verified for possible attacks. Hence, attacks were easily performed since no prevention of any kind was present.

1. ITRC = ID Theft Resource Center [↑](#footnote-ref-1)