Abstract

Add nice abstract statement

Final Report

SePr 2016

Radu Stoica, Armin Roushan, Jan-Niklas Schneider, Georgiana Manolache, Nena O’Driscoll, Raditya Pratama

Table of Contents

[1 Introduction 2](#_Toc462040552)

[2 Web Application Overview 2](#_Toc462040553)

[3 Architecture 2](#_Toc462040554)

[4 Threat analysis 3](#_Toc462040555)

[4.1 Data assets 4](#_Toc462040556)

[4.2 Authorization requirements 5](#_Toc462040557)

[4.3 Risk analysis 5](#_Toc462040558)

[5 Secure design 6](#_Toc462040559)

[6 Implementation choices 6](#_Toc462040560)

[7 Code example? 6](#_Toc462040561)

[8 References 6](#_Toc462040562)

# Introduction

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

## User Group

The application allows for two different roles, namely the administrator and the customer. Creation wise an administrator user has to be added manually in the database. A new customer user can be added through the administrator page. Similar to other banking applications, eFontys allows a user to have an account with a balance, with which the user can interact. All their functionalities more specifically per role can be found below.

### Customer

The customer has an account with the following functionalities:

* 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 functionalities as the customer extended with the following:

* Add and remove users (except for himself)
* Edit other account details

## Network Architecture

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



# Web Application Overview

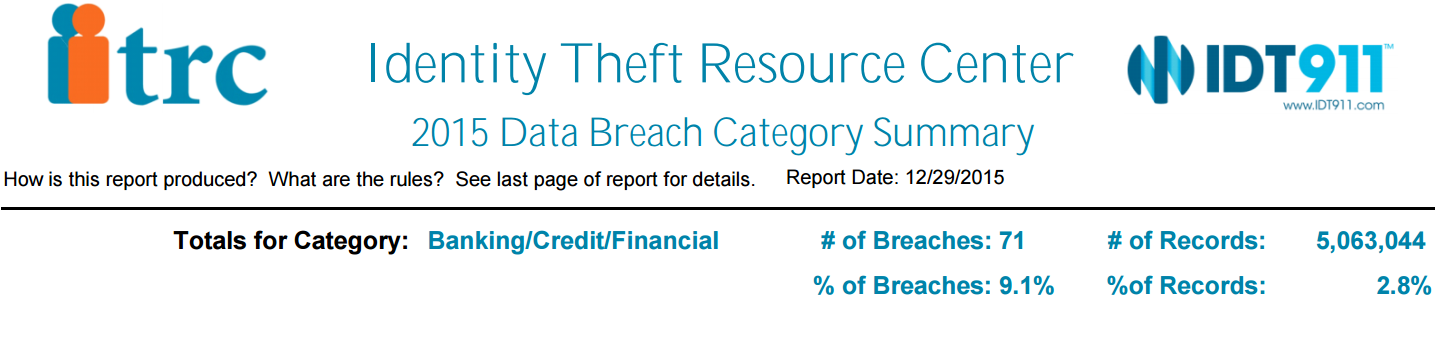
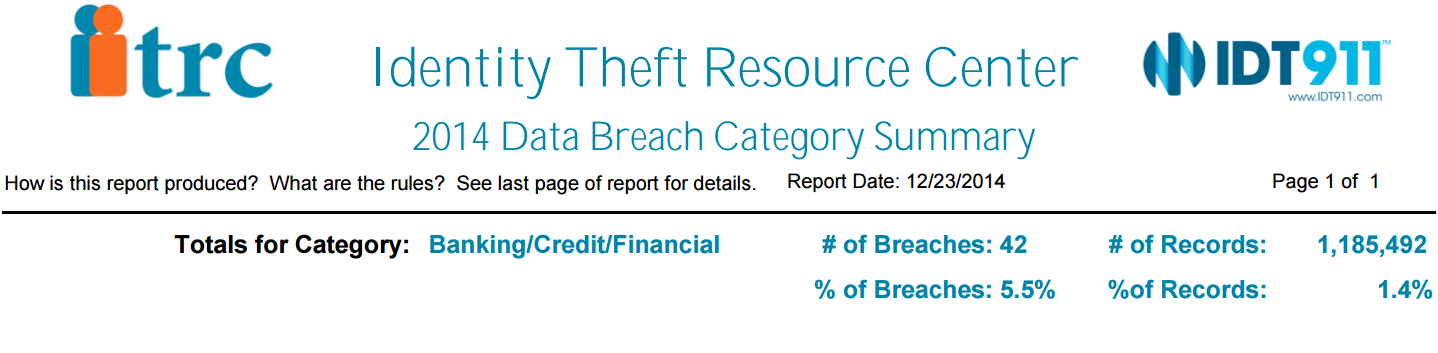
# Architecture

# 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.  Accoding to ITCR (Identity Theft Resource Center) Annual Breach Reports, only in the US 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).

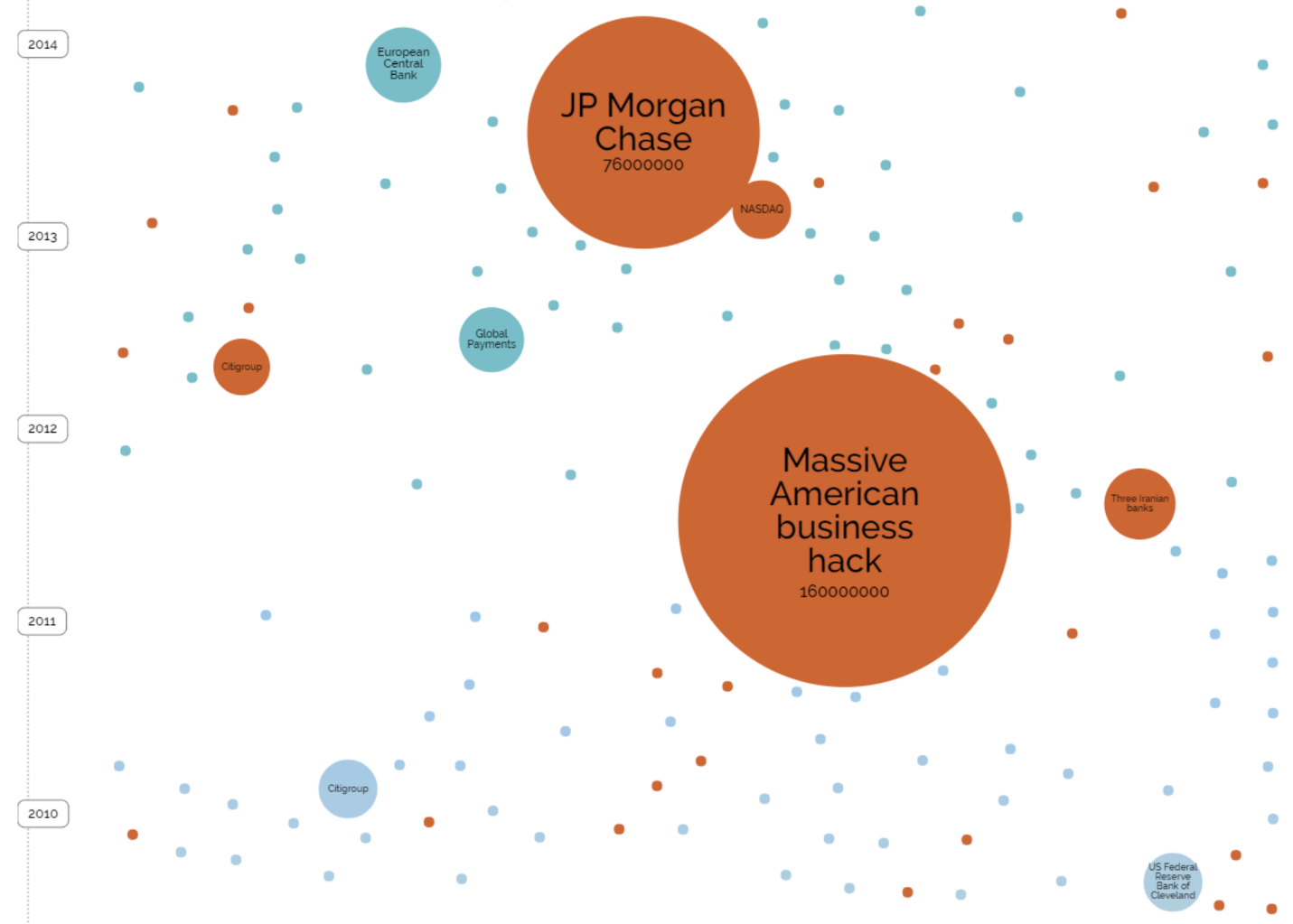
Protecting financial data and assets is, thus, vital for a financial organization. The following subchapters describe the risk analysis for the online banking web application.

Figure 4‑1 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 4‑2 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/)

## Data assets

The online banking website keeps all information at a high security level.

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. |

## Authorization requirements

Table 4‑2 Authorization Requirements

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

## 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 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

## 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.

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

[pic still not here yet]

# Implementation choices

Here the implementation choices regarding the technologies and security are described.

## Technologies

The application was to be built in language all group members already had some form of experience in or could in a reasonable time frame catch up on. Due to available skills the already made eFontys, which consisted of a combination of HTML5, CSS3 and PHP, was chosen. This would allow us to fully focus on securing the web application.

## Security

PHP being infamous for its vulnaribilies had us focus on both the smaller (no magic quotes, no error reporting)and bigger (SQL injection attacks prevention) picture.

# Code example?

# References

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

*Data Breach Reports 2015*. (2015). Retrieved from Identity 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

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