

Threat Modelling Assignment

**Pauline Finlay**

**B00107945**

***Department of Informatics***

***School of Informatics & Engineering***

***Technological University Dublin, Blanchardstown***

***Dublin 15.***

**Word Limit for Assignment: 3000 Actual Word Count: 3246**

**BSc (Hons) in Computing in Digital Forensics & Cyber Security**

**DFCS H4013 Application Security**

**31st October 2020**

**Plagiarism Declaration**

I hereby certify that this material, which I am submitting for assessment for the course title of BSc. (Hons) In Computing in Digital Forensics and Cyber Security is my own work and that I have:

* Clearly referenced/listed all sources as appropriate
* Given the sources of all pictures, data etc. that is not our own
* Any work of any other student(s) either past or present is clearly referenced
* Not submitted for assessment work previously submitted for any other course, degree or qualification
* Not incorporated any text acquired from external agencies other than extracts from attributed sources (including online facilities)
* Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
* My work may be electronically checked for plagiarism, including, but not exclusively, by the use of plagiarism detection software and stored for future comparison
* Any false claim for this work will be penalised in accordance with the College’s regulations

**Signed:** Pauline Finlay **Date:** 06/11/2020

**Abstract**

The preparation of this Threat Model is based on assumptions I have made about the architecture and design of the application. Threats that the application are susceptible to are identified, documented, rated and mitigations are suggested to lessen their impact on the system. In accessing the impact and validity of the threats testing of the application should be completed which may identify the need for further mitigation. In maintaining the system continuous monitoring and testing should be routinely carried out.

Contents

[Application Name and Description 1](#_Toc55469778)

[Owners, Authors, and Stakeholders 1](#_Toc55469779)

[Revision History 1](#_Toc55469780)

[Security Objectives 1](#_Toc55469781)

[Identify Assets 2](#_Toc55469782)

[Architecture Overview 3](#_Toc55469783)

[End-to-End Deployment Scenario 4](#_Toc55469784)

[Roles 5](#_Toc55469785)

[Key Scenarios 5](#_Toc55469786)

[Technologies 5](#_Toc55469787)

[Application Security Mechanisms 5](#_Toc55469788)

[Decompose the Application 7](#_Toc55469789)

[Trust Boundaries 7](#_Toc55469790)

[Data Flows 7](#_Toc55469791)

[User Login 7](#_Toc55469792)

[Authenticated User has access to application 8](#_Toc55469793)

[Authenticated User Syncs with Banking Server 8](#_Toc55469794)

[Authenticated User Issues Payment Request to Banking Server 9](#_Toc55469795)

[New Bills & Transaction History Synced from Third Party Billers 9](#_Toc55469796)

[Administrator Login 10](#_Toc55469797)

[Entry Points 10](#_Toc55469798)

[Exit Points 10](#_Toc55469799)

[Identify Threats - Threat Analysis Based on STRIDE Model 11](#_Toc55469800)

[Spoofing 11](#_Toc55469801)

[Tampering 12](#_Toc55469802)

[Repudiation 12](#_Toc55469803)

[Information Disclosure 12](#_Toc55469804)

[Denial of Service 13](#_Toc55469805)

[Escalation of Privilege 14](#_Toc55469806)

[Document the Threats 15](#_Toc55469807)

[Threat 1: Browser Client - Spoofed Login Page 15](#_Toc55469808)

[Threat 2: Reflected XSS Attack 15](#_Toc55469809)

[Threat 3: CSRF Forgery Attack 16](#_Toc55469810)

[Threat 4: Browser Client – Login Page Crashes 16](#_Toc55469811)

[Threat 5: SQL Injection Attack 17](#_Toc55469812)

[Threat 6: Crashing Authentication Server 17](#_Toc55469813)

[Threat 7: Network Traffic Interception 18](#_Toc55469814)

[Threat 8: Banking Server Crashes 18](#_Toc55469815)

[Threat 9: Persistent XSS Attack 19](#_Toc55469816)

[Threat 10: Insufficient Logging & Monitoring 19](#_Toc55469817)

[Threat 11: Authentication Server Process Memory Tampering 20](#_Toc55469818)

[Threat 12: Entities Repudiate Information Traffic 20](#_Toc55469819)

[Rate the Threats - DREAD Scoring 21](#_Toc55469820)

[Mitigation Plan 23](#_Toc55469821)

[Conclusion 24](#_Toc55469822)

[References 25](#_Toc55469823)

**Table of Tables**

[Table 1: Asset Table 2](#_Toc55469824)

[Table 2: Threat Table 11](#_Toc55469825)

[Table 3: DREAD Scoring 22](#_Toc55469826)

[Table 4: Mitigation Plan 23](#_Toc55469827)

**Table of Figures**

[Figure 1: Overall DFD for Money Management Web Application 4](#_Toc55469828)

[Figure 2: User Login 7](#_Toc55469829)

[Figure 3: Authorised user request transaction history 8](#_Toc55469830)

[Figure 4: Authorised user logs into banking server to request current information 8](#_Toc55469831)

[Figure 5: Authorised user logs into baking server to issue bill payment instruction 9](#_Toc55469832)

[Figure 6: New bill sent from biller to application 9](#_Toc55469833)

[Figure 7: Administrator login direct to application 10](#_Toc55469834)

# Application Name and Description

Mybillmanager.ie is a new bill and salary management app introduced to the market by TUD Money. This web application allows you manage your money and pay bills on time safely through a single application. The application incorporates two-factor authentication login. Among the features of the application are user account pages to display all transactions, a secure transfer facility to pay bills and a sync facility for real time account balances.

The scope of this assessment includes the authentication server and database, the web application server and user history database. In this report we identify their data flows, trust boundaries and attack surfaces.

The banking server and external billers are separate applications, and their security is outside the scope of this assessment.

# Owners, Authors, and Stakeholders

|  |  |  |
| --- | --- | --- |
| **Owners** | **Authors** | **Stakeholders** |
| mybillmanager.ie | Pauline Finlay | TUD Money |

# Revision History

|  |  |  |
| --- | --- | --- |
| **Name** | **Change** | **Date** |
| Pauline Finlay | Threat Model Created | 26/10/2020 |
|  |  |  |

# Security Objectives

To maintain the confidentiality, integrity, and availability of the data and the application, the following must be enforced:

* Protect user login credentials and Personally Identifiable Information (PII)
* Protect user bank account details, transaction history and billing information.
* Prevent unauthorised access to the authentication and application servers.
* Prevent unauthorised access to authentication and user history databases.
* Prevent unauthorised access to banking and third-party biller servers via application server.
* Maintain the company reputation.
* Ensure that the application is available 99.99 percent of the time.

## Identify Assets

Table 1: Asset Table

|  |  |  |
| --- | --- | --- |
| **Owner** | **Category** | **Description** |
| User Assets | Login Details | Username, Password, Email |
| Personally Identifiable Information (PII) | Name, Address, Email, Phone Number |
| Bank Details | Account No, Sort Code, Transaction History |
| Billing Information | Supplier Details, Account Holder Name, Account No’s, Billing Address |
| Application Assets | Access to Authentication Database | All data stored on the database |
| Access to User History Database | All data stored on the database |
| Availability of Authentication Server | If server goes down nobody can log onto the application |
| Availability of Application Server | If server goes down user has no access to their information |
| Login Session | The session associated with the logged in user |
| Admin Login Details | Username, Password |
| Bank Assets | Login to Banking Server | Prevent unauthorised access to the banking server |
| Third-party Billers | Connection to Third Party Biller Servers | Prevent unauthorised access to the biller servers |

# Architecture Overview

The application allows the user to control their finances from a single location. The user login is via a browser web page where a username and password are entered. These details are redirected to an authentication server which checks the details by making a call to an authentication database. There is 2 factor authentication in place using ASP.net Identity which sends a user confirmation request using Google Authenticator. Once the login details are authenticated a session is created allowing an authorised user to connect to the mybillmanager.ie application.

The services provided by the application allow the user to control their finances from a single location. The user can view all bank transactions by syncing their account(s) and credit card details with the application using the external API. The transaction details are written to an external ‘User History’ database. The application contains a user account page allowing the user to view all, or selected transactions by date, supplier or type by making a query to this database.

The application also has the major utility providers signed up and user accounts can be synced directly using the external API. The billing details from these suppliers will be updated directly to the application for the user to control their expenditure. There is also the facility for the user to enter one-off billing and enter the selected payment method. If a bank transfer is the chosen method the instruction can be set up in the application and a one-off payment instruction issued to the bank. The user can also schedule regular payments or by issuing an instruction to their bank. These instructions are issued using the external API.

The administration of the application is carried out by authorised administrators their login is controlled by direct access to the application server only. The server is secured in an access-controlled room located within the company headquarters. Access to this room is biometrically controlled in addition to pin entry. The building itself has security at the entrance and staff must swipe the photo id to be granted access. All visitors must always be signed in by an authorised staff member and escorted within the building.

The service provider servers and secure payment function controlled by the banking server are outside the scope of the application as security in these instances is the responsibility of the respective owners. The mybillmanager.ie application is responsible for updating security patches relating to the external API in relation to both monitoring access from the application side.

## End-to-End Deployment Scenario

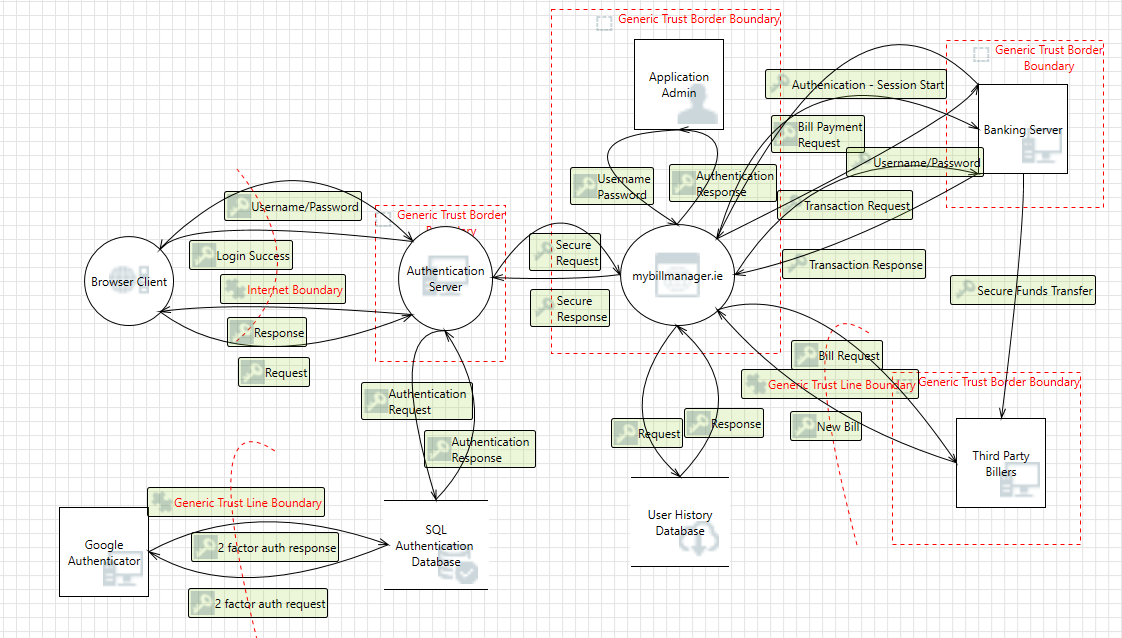


Figure 1: Overall DFD for Money Management Web Application

## Roles

Application roles are:

* Authenticated Application users
* Application Administrators

## Key Scenarios

Important application scenarios are:

* Authorised Administrator logs into the application
* Authorised user logs into the application.
* Authorised user reviews their transaction history.
* Authorised user requests current transaction and balance details from banking server.
* Authorised user issues payment request to banking server.
* Secure funds transfer from banking server to biller.
* New bills and account transactions issued by biller to user via the application.

## Technologies

The application uses the following technologies:

* Authorisation Server: Microsoft Internet Information Server (IIS)
* Web Application Server: Microsoft Internet Information Server (IIS)
* Presentation logic: ASP.NET (C#)
* Business logic: C# Class Libraries
* Data access logic: ADO.NET, T-SQL Stored Procedures
* Authentication Database Server: Microsoft SQL Server 2000
* User History Database Server: Microsoft SQL Server 2000
* Two Factor Authentication Login using ASP.NET Identity and Google Authenticator

## Application Security Mechanisms

The most important application security mechanisms known at this time are:

* Roles are used to authorize access to application logic.
* Users are authenticated using 2-factor authentication - ASP.NET Identity with Google Authenticator.
* Administration can be performed only by physically logging on to the server computer. No remote administration access is provided.
* The authentication server and web application server are both placed behind perimeter firewalls.
* External API patches are routinely inspected and updated

# Decompose the Application

## Trust Boundaries

Identified trust boundaries are:

* The perimeter firewalls of the authentication and application servers.
* The authentication server trusts call from the browser client
* The authentication database server trusts calls from the authentication server.
* The application server trusts calls from authenticated users
* The user history database server trusts calls from the Web application's identity.

## Data Flows

### User Login

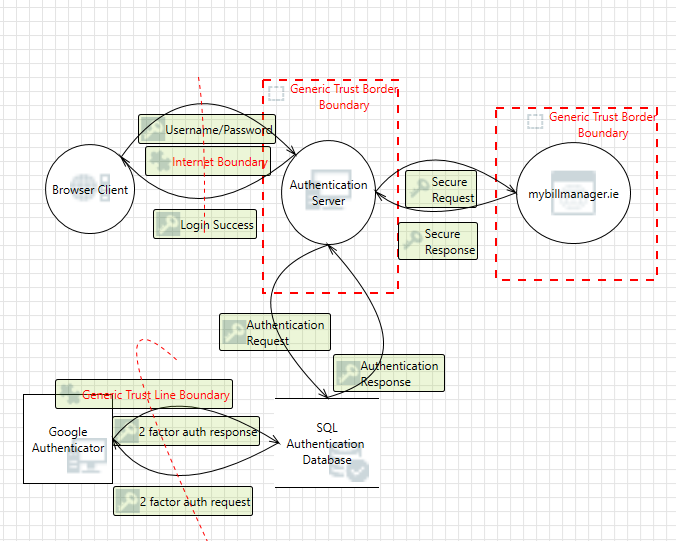


Figure 2: User Login

* The user logs on submitting a username and password through the logon form.
* The username and password are authenticated using the ASP.net identify framework.
* The authentication server sends a request to the database combined with google authenticator to confirm the identity of the user.
* Once confirmed secure encrypted communication commences with the browser client and mybillmanager.ie application.

### Authenticated User has access to application

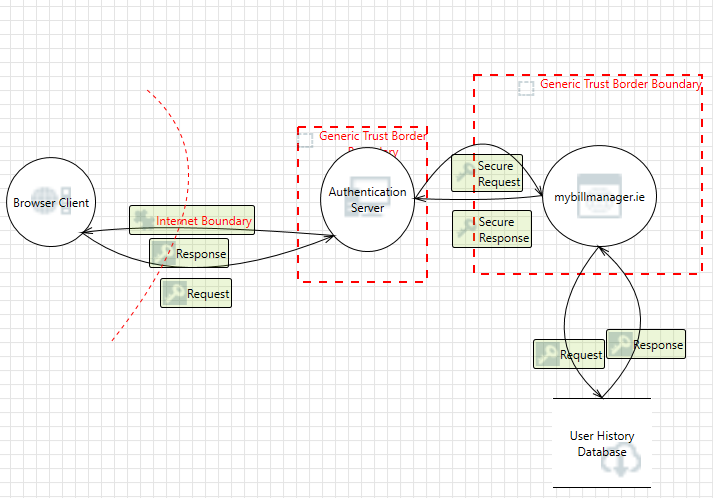


Figure 3: Authorised user request transaction history

* An authenticated user can browse transaction history on the user account pages reading the information from the user history database.

### Authenticated User Syncs with Banking Server

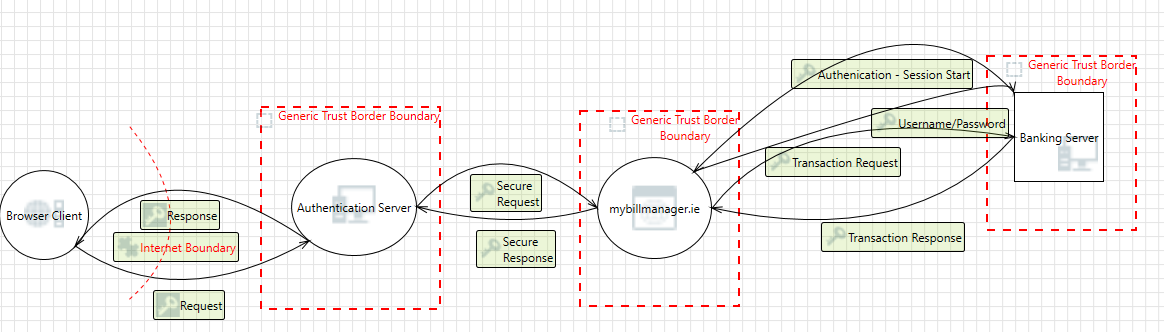


Figure 4: Authorised user logs into banking server to request current information

* Authenticated user logs into banking server and requests up to date balance and transaction information from banking server.
* Banking server syncs up to date information with the application.

### Authenticated User Issues Payment Request to Banking Server

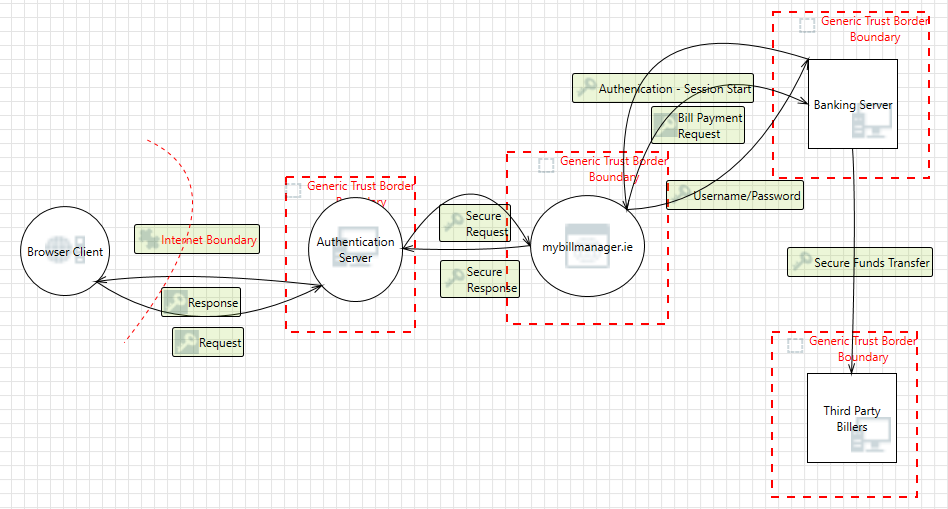


Figure 5: Authorised user logs into baking server to issue bill payment instruction

* Authenticated user logs into banking server, issues payment instruction to banking server.
* Secure funds transfer from banking server to biller.

### New Bills & Transaction History Synced from Third Party Billers

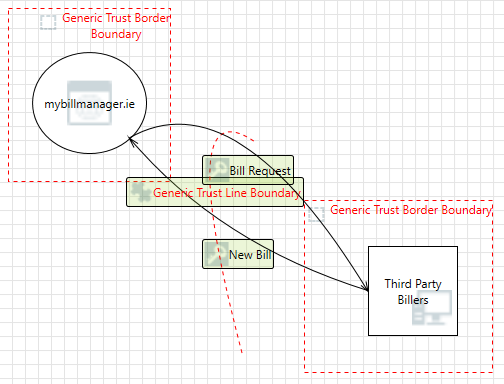


Figure : New bill sent from biller to application

* New bills and account transactions are issued by biller to application.

### Administrator Login

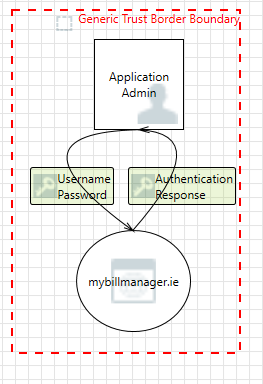


Figure : Administrator login direct to application

* Administrator logs into application server directly

## Entry Points

Entry points are:

* Port 80 for Web requests.
* Port 443 for SSL.
* All other ports are restricted by the firewall.
* The logon page, which is accessible to all Internet users. Logon is validated by using ASP.net 2 factor authentication.
* The administrator login is direct via the application server.

## Exit Points

Exit points are:

* The user account pages which write the user’s transaction history to the database.
* Authorised user connection to banking server
* API calls to banking server and third party biller servers

# Identify Threats - Threat Analysis Based on STRIDE Model

Table 2: Threat Table

|  |  |
| --- | --- |
| **STRIDE Category** | **DFD Element Affected & Result** |
| Spoofing  *attempting to gain access to a system by using a false identity* | **Browser Client:**   * Unauthorised access to authentication server   **Authentication Server:**   * Unauthorised access to browser client * Potential for unauthorised access to the authentication database   **Authentication Database:**   * Sending in correct info to Google authenticator * Data is written to attacker instead of database * Incorrect information delivered to server   **User History Database:**   * Data written to attacker instead of database * Incorrect info delivered to application server   **Application Admin:**   * Unauthorised access to application server   **Banking Server:**   * Information may be sent to attacker instead of legitimate server   **Third Party Biller Servers:**   * Unauthorised access to application server * Data sent to attacker instead of service provider |
| Tampering  *unauthorized modification of data* | **Authentication Server:**   * Cross Side Scripting attack (XSS) * Process Memory   **Authentication Database:**   * SQL Injection attack inserting malicious code into commands passed to database   **Data Flows:**   * Attacker may intercept traffic flowing between all entities |
| Repudiation  *ability of users (legitimate or otherwise) to deny that they performed specific actions or transactions* | * Browser client claims it did not receive data from outside entity * Authentication server claims it didn’t receive data from outside entity * Authentication database claims it did not write data from entity outside its trust boundary * Google Authenticator claims it did not receive data * Web application claims it did not receive data from an entity on the other side of trust boundary * User history database claims it did not write data received form outside entity * Banking server claims it did not receive data from outside its trust boundary * Third party biller claims it did not receive data from outside entity |
| Information Disclosure  *unwanted exposure of private data* | * Improper data protection of SQL Authentication Database * Authorisation bypass of the user history database * Weak access control of the user history database * Weak user authentication scheme |
| Denial of Service  *process of making a system or application unavailable* | **Data Flows:**   * Attacker interrupts traffic between entities   **Browser Client:**   * Crashes, stops or slows down limiting availability   **Authentication Server:**   * Server crashes, stops or slows down limiting availability * Excessive resource consumption   **Authentication Database:**   * Attacker prevents access to database prevents user from access * Excessive resource consumption   **Application Server:**   * Server crashes, stops or slows down limiting availability * Excessive Resource Consumption   **User History Database:**   * Excessive Resource Consumption |
| Escalation of Privilege  *user with limited privileges assumes the identity of a privileged user to gain privileged access to an asset* | **Browser Client:**   * Impersonate context of Authentication server * Attacker passes in data to change flow program * Cross Side Forgery Attack (CSRF) exploiting trust relationships   **Authentication Server:**   * Attacker changes flow of program execution   **Authentication Database:**   * Remotely execute code on authentication server   **Web Application:**   * Attacker changes flow of program execution * Impersonate context of banking server or third-party biller   **User History Database:**   * Remotely execute code on web application   **Banking Server:**   * Ability to remotely access application code   **Third party Biller:**   * Ability to remotely access application code   **System Admin:**   * If an attacker gains access to system and can escalate privileges to admin level they control everything |

# Document the Threats

## Threat 1: Browser Client - Spoofed Login Page

|  |  |
| --- | --- |
| **Description** | Threat Actor creates duplicate login page to capture the username and password of a legitimate user. |
| **STRIDE Classification** | * Spoofing * Information Disclosure |
| **Entry Point(s)** | * Browser Client – Login Page |
| **Asset(s) at Risk** | * Login Details |

## Threat 2: Reflected XSS Attack

|  |  |
| --- | --- |
| **Description** | If the treat actor can manipulate the user login page so that it returns a malicious java script to users, executing in their browser and compromising their application interaction. It is a client-side vulnerability which affects other users. |
| **STRIDE Classification** | * Tampering * Information Disclosure |
| **Entry Point(s)** | * Browser Client – Login Page |
| **Asset(s) at Risk** | * Authentication Server Access * Login Details * Application Access |

## Threat 3: CSRF Forgery Attack

|  |  |
| --- | --- |
| **Description** | Where the treat actor manipulates the user into carrying out a harmful activity on their behalf. The 1st step is to trick the user into clicking on a harmful link or loading a harmful page. Secondly the attacker crafts a legitimate looking request from the browser client to the web application. To be effective the user must be authenticated and if an admin account can be hacked it effects the entire application.  This is done by exploiting existing trust relationships, in this case between the browser client and authentication server and the authentication server and application server. |
| **STRIDE Classification** | * Tampering * Elevation of Privilege |
| **Entry Point(s)** | * Browser Client – Login page |
| **Asset(s) at Risk** | * Authentication Server * Web Application Server |

## Threat 4: Browser Client – Login Page Crashes

|  |  |
| --- | --- |
| **Description** | If the web hosting service hosting the Login Page crashes the user cannot access the application or any of their details |
| **STRIDE Classification** | * Denial of Service |
| **Entry Point(s)** | * Browser Client – Login Page |
| **Asset(s) at Risk** | * Access to Authentication Server |

## Threat 5: SQL Injection Attack

|  |  |
| --- | --- |
| **Description** | Threat Actor gains access to the authentication database or user history database using carefully crafted data inserted into the requests which are later passed to the databases. This occurs because SQL executes all syntactically correct queries it receives. Once executed an attacker has full control of the database and can change or delete all or specific information stored there. |
| **STRIDE Classification** | * Tampering * Information Disclosure * Denial of Service * Elevation of Privilege |
| **Entry Point(s)** | * Browser Client - Login Page |
| **Asset(s) at Risk** | * Login Details * Transaction Details * Billing Details * Access to Authentication Database * Access to User History Database |

## Threat 6: Crashing Authentication Server

|  |  |
| --- | --- |
| **Description** | Threat Actor floods authentication server with requests for authentication slowing down or causing the server to crash. |
| **STRIDE Classification** | * Denial of Service |
| **Entry Point(s)** | * Data Flow – traffic from browser to server |
| **Asset(s) at Risk** | * Access to Authentication Server |

## Threat 7: Network Traffic Interception

|  |  |
| --- | --- |
| **Description** | Threat actor interrupts or intercepts traffic flowing between the entities. Once they have gained access, they have the ability to change the data being transmitted or stop the transmission denying service. If the traffic is not encrypted sensitive information could be disclosed to the attacker. |
| **STRIDE Classification** | * Spoofing * Tampering * Information Disclosure * Elevation of Privilege |
| **Entry Point(s)** | * Data Flows |
| **Asset(s) at Risk** | * Login Details * Admin Login Details * Transaction History * Bill Payment Details * Service Provider Details |

## Threat 8: Banking Server Crashes

|  |  |
| --- | --- |
| **Description** | If threat actor breaches the banking server, access to current information and payment requests will be denied. Confidential user information may be compromised |
| **STRIDE Classification** | * Denial of Service |
| **Entry Point(s)** | * No Direct access to application |
| **Asset(s) at Risk** | * Personally Identifiable Information (PII) * Bank Details |

## Threat 9: Persistent XSS Attack

|  |  |
| --- | --- |
| **Description** | If the threat actor is able to store the malicious script on the authentication server it will be executed every time a user encounters it. |
| **STRIDE Classification** | * Tampering |
| **Entry Point(s)** | * Browser Client – Login Page |
| **Asset(s) at Risk** | * Web application server |

## 

## Threat 10: Insufficient Logging & Monitoring

|  |  |
| --- | --- |
| **Description** | It is not always possible to stop every attack but if there is no alert mechanism in place it may be possible to gain a persistent foothold in the application. |
| **STRIDE Classification** | * Tampering * Information Disclosure |
| **Entry Point(s)** | * Browser Client – Login Page * Open Ports |
| **Asset(s) at Risk** | * Authentication Server * Web Application Server |

## Threat 11: Authentication Server Process Memory Tampering

|  |  |
| --- | --- |
| **Description** | If Authentication Server is given access to memory, such as shared memory or pointers, or is given the ability to control what Browser Client executes (for example, passing back a function pointer.), then Authentication Server can tamper with Browser Client. |
| **STRIDE Classification** | * Tampering * Denial of Service |
| **Entry Point(s)** | * Authentication Server |
| **Asset(s) at Risk** | * Authentication Server Availability |

## Threat 12: Entities Repudiate Information Traffic

|  |  |
| --- | --- |
| **Description** | Entities within the system claim to have not sent or received information from another entity. |
| **STRIDE Classification** | * Repudiation |
| **Entry Point(s)** | * All Entities |
| **Asset(s) at Risk** | * All Assets |

# Rate the Threats - DREAD Scoring

Threats are rated using the **DREAD** model –

* **D**AMAGE – how much damage would be caused if the attack succeeded
* **R**EPRODUCIBILITY – the ease of reproducing the attack
* **E**XPLOITABILITY – the amount of time, effort and expertise needed to carry out the attack
* **A**FFECTED USERS – the percentage of users affected by the attack
* **D**ISCOVERABILITY – how easy is it for an attacker to discover the threat

Each vulnerability is assigned a value of 1 (lowest) to 3 (highest) depending on their severity or impact, to produce the DREAD score. The Risk Rating is assessed as high/medium/low based on the score:

* 5-8 Low Risk (L)
* 9-12 Medium Risk (M)
* 13-15 High Risk (H)

Table 3: DREAD Scoring

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Vulnerability** | **D** | **R** | **E** | **A** | **D** | **Total** | **Risk Rating** |
| Spoofed Login Page | 1 | 3 | 2 | 1 | 3 | 10 | **M** |
| Reflected XSS Attack | 3 | 2 | 2 | 3 | 3 | 13 | **H** |
| CSRF Forgery Attack | 3 | 2 | 2 | 3 | 2 | 12 | **M** |
| Browser Client – Login Page Crashes | 3 | 2 | 2 | 3 | 2 | 10 | **M** |
| SQL Injection Attack | 3 | 3 | 3 | 3 | 2 | 14 | **H** |
| Crashing Authentication Server | 3 | 3 | 2 | 3 | 3 | 14 | **H** |
| Network Traffic Interception | 3 | 3 | 3 | 3 | 3 | 15 | **H** |
| Banking Server Crashes | 2 | 2 | 2 | 2 | 2 | 8 | **L** |
| Persistent XSS Attack | 3 | 2 | 2 | 3 | 2 | 10 | **M** |
| Insufficient Logging & Monitoring | 2 | 2 | 2 | 3 | 2 | 11 | **M** |
| Authentication Server Process Memory Tampering | 3 | 1 | 1 | 2 | 1 | 8 | **L** |
| Entities Repudiate Information Traffic | 2 | 2 | 2 | 3 | 2 | 11 | **M** |

# Mitigation Plan

Table 4: Mitigation Plan

|  |  |
| --- | --- |
| **Vulnerability** | **Mitigation** |
| Spoofed Login Page | * Access prevented by 2-factor authentication |
| Reflected XSS Attack | * Filter user input * Encode data in HTTP responses * Use appropriate headers to ensure that the browser interprets responses where HTML or JavaScript are not expected: *Content-Type* and *x-Content-Type-Options* |
| CSRF Forgery Attack | * Use CSRF tokens known only by the client and authentication server and randomly generated to validate traffic |
| Browser Client – Login Page Crashes | * Ensure the service provider has a high availability score and a fallback server in case of interruption |
| SQL Injection Attack | * Validate and Sanitise user inputs * Use prepared statements to make requests from database * Reduce the attack surface by disabling any unused functionality * Set Privilege levels for database access * SQL Patch updates * Encrypt or hash sensitive data stored in the database |
| Crashing Authentication Server | * Place behind a Firewall * Configure ACL’s * Close unused ports |
| Network Traffic Interception | * Encrypt all traffic |
| Banking Server Crashes | * Outside of our control |
| Persistent XSS Attack | * Filter user input * Encode data in HTTP responses * Use appropriate headers to ensure that the browser interprets responses where HTML or JavaScript are not expected: *Content-Type* and *x-Content-Type-Options* |
| Insufficient Logging & Monitoring | * 24x7 logging & monitoring using an Intrusion Detection System (IDS) * Security Incident and Response Procedures |
| Authentication Server Process Memory Tampering | * Consider if the function could work with less access to memory, such as passing data rather than pointers. * Copy in data provided, and then validate it. |
| Entities Repudiate Information Traffic | * Use logging and auditing to record the source, time and summary of received data within the application |

# Conclusion

The threats and vulnerabilities identified in this report have been identified using the STRIDE framework. These should be mitigated against as per the mitigation plan. Once the mitigation plan has been implemented ethical hacking should be carried out to see if the plan has been successful. The results of the ethical hacking assessment may identify the need to implement further mitigation strategies. Continual monitoring of the system is also recommended to identify any intrusion and prevent the confidentiality of information and protect the integrity and availability of the application.

# References

Application Threat Modeling | OWASP. 2020. Application Threat Modeling | OWASP. [ONLINE] Available at: https://owasp.org/www-community/Application\_Threat\_Modeling. [Accessed 02 November 2020].

Archiveddocs. 2020. Threat Modeling | Microsoft Docs. [ONLINE] Available at: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v=pandp.10)?redirectedfrom=MSDN. [Accessed 02 November 2020].

Infosec, Inc.. 2020. Applications Threat Modeling - Infosec, Inc.. [ONLINE] Available at: https://resources.infosecinstitute.com/topic/applications-threat-modeling/. [Accessed 02 November 2020].

McGrath, M. (2013). Threat Modelling for Legacy Enterprise Applications (Unpublished master's thesis). Letterkenny Institute of Technology. Retrieved August 25, 2020, from https://research.thea.ie/handle/20.500.12065/1187

Prism. 2020. Prism Bills & Money. [ONLINE] Available at: https://www.prismmoney.com/. [Accessed 02 November 2020].

Template Example: Web Application Threat Model - Guidance Share. 2020. Template Example: Web Application Threat Model - Guidance Share. [ONLINE] Available at: http://www.guidanceshare.com/wiki/Template\_Example:\_Web\_Application\_Threat\_Model. [Accessed 02 November 2020].