urwallet.ie Threat Model

A threat model for the Web Application belonging to urwallet.ie as part of my class on Application Security for my (Hons)B.Sc. In Cyber Security and Digital Forensics 2019/2020, Please Note that the Application in this Threat Model is 100% fictional for the purpose of this class

DEAN   
  
github.com/TheCyberViking

**Application Security**

**CA – Threat Modelling**

**By Dean**

**B0009**

Department of Informatics

School of Informatics and Engineering

**BN420 Honours Bachelor of Science in Computing**

**Digital Forensics & Cyber Security**

**Application Security**

**Stephen**

**25/10/2019**

# Plagiarism Declaration

**DEPARTMENT OF INFORMATICS**

**HONOURS BACHELOR OF SCIENCE IN DIGITAL FORENSICS AND CYBER SECURITY**

**LECTURER: Stephen**

**DECLARATION ON PLAGIARISM**

I declare that the work I/We am (are) submitting for assessment by the Institute examiner(s) is entirely my (our) own work, except where the author or source has been duly referenced and attributed.

I confirm that this material has not been previously submitted for a degree or any other qualification at or any other institution. I further confirm that I have read and understood the Institute policy on plagiarism in assignments and examinations (3AS08.doc) and that I am not, so far as I am aware, in breach of any of these regulations.

Signed: Dean

Date: 25/10/2019

Table of Contents

[Threat Modeling Methodology 5](#_Toc22921452)

[Threat Model Information 6](#_Toc22921453)

[Security Objectives 7](#_Toc22921454)

[Information Storage 8](#_Toc22921455)

[Application Overview 9](#_Toc22921456)

[Trust Levels 9](#_Toc22921457)

[Technologies and key Features 10](#_Toc22921458)

[Architecture Overview 11](#_Toc22921459)

[Assets 12](#_Toc22921460)

[Application Decompositions 13](#_Toc22921461)

[Trust Boundary’s 14](#_Toc22921462)

[Data Flow 15](#_Toc22921463)

[Step 1 – 2 Factor Authentication 15](#_Toc22921464)

[Step 2 – User Login 16](#_Toc22921465)

[Step 3 – Transfer to user 17](#_Toc22921466)

[Step 4 – Transfer to bank 18](#_Toc22921467)

[Step 5 – User Registration 19](#_Toc22921468)

[Network Entry Point 20](#_Toc22921469)

[Network Exit Points 21](#_Toc22921470)

[Threats 22](#_Toc22921471)

[Stride Areas 22](#_Toc22921472)

[Stride 22](#_Toc22921473)

[Stride table 23](#_Toc22921474)

[Threat Tree 24](#_Toc22921475)

[Access to Personnel Information 24](#_Toc22921476)

[Code Injection Attack Threat Tree 24](#_Toc22921477)

[Denial of Service Threat Tree 25](#_Toc22921478)

[Theft of Authorization Cookies Threat Tree 25](#_Toc22921479)

[XSS Attack Threat Tree 25](#_Toc22921480)

[CSRF Attack Tree 26](#_Toc22921481)

[Compromise of Banking Details 26](#_Toc22921482)

[Threat Tree Reference 27](#_Toc22921483)

[Vulnerabilities 29](#_Toc22921484)

[Dread Table 30](#_Toc22921485)

[Reference 34](#_Toc22921486)

# Threat Modeling Methodology

The go to methodology when it comes to threat modelling there is two models to follow, the OWASP model and the Microsoft model, for this report we will be using the Microsoft model. According to Microsoft threat modelling methodology there are five major steps to threat modelling these steps are as follows:

* Defining security requirements
  + What are the security requirements of the current device or environment in which the testing is being conducted
* Creating an application diagram
  + Developing the layout of the current application to be tested, so that the investigator can get a solid understanding of how the application functions
* Identifying threats
  + Locating and documenting any possible threats that maybe directed at the device, application or location
* Mitigating threats
  + Researching and locating possible activities to get around the threats and protect the device
* Validating that threats have been mitigated
  + Doing testing to confirm the threat doesn’t exist and has been successfully mitigated

In this report we will look at the needed services, protocols and applications to figure out what needs, and at what vulnerabilities that could be present on the network.

# Threat Model Information

|  |  |
| --- | --- |
| **Threat Model Information** | |
| **Name** | Urwallet.ie |
| **Application Version** | Version 1 |
| **Description** | While credit and debit cards allow us to make purchases in-store and online and access cash, through ATM or cash-back purchases, they don't allow person-to-person fund transfers. Mobile payment apps are becoming increasingly important as they allow us to perform all the transactions of a debit of credit card, while also giving us the option of transferring funds directly to other people.  urwallet.ie is a new mobile payment app about to go into development. Your task is to complete an in-depth report on the complete threat modelling process. Suitable DFD’s must be used to identify all data flows, call flows, trust boundaries and attack surfaces. DFD’s at different levels are expected to decompose the application and identify threats specific to that component.  The Website will have the following features:  • Two-factor authentication  • User account pages (showing all transactions)  • Secure funds transfer to other accounts  • Secure funds transfer to and from your bank  • Secure friend connection (to share bill payments / message etc.) |
| **Documents Owner** | Dean |
| **Participants** | Dean |
| **Reviewer** | Stephen |

# Security Objectives

The object of this report is to identify possible threats in the working Application and Infrastructure, to best be able to develop standard operating procedures and mitigate the issues that may present security concerns on the network. This is so that we can work on allowing the development team to secure the application against possible threats or attackers.

Since this is a web application, the best standard to follow when it comes to vulnerabilities related to web applications would be OWASP, for this report we will be using OWASP threat checking modeling check vulnerabilities. OWASP developed and maintain a list of vulnerabilities that is updated yearly, at current 2019 bellow is the current OWASP top 10 list for 2019; this list is the current industry wide acceptable guideline when it comes to Web Application vulnerabilities.

|  |
| --- |
| **OWASP TOP 10 – 2019** |
| **A1 – Injection** |
| **A2 – Broken Authentication** |
| **A3 – Sensitive data exposure** |
| **A4 – XML External Entities (XXE)** |
| **A5 – Broken Access Control** |
| **A6 – Security Misconfigurations** |
| **A7 – Cross-Site Scripting (XSS)** |
| **A8 – Insecure Deserialization** |
| **A9 – Using Components with known Vulnerabilities** |
| **A10 – Insufficient Logging and Monitoring** |

# Information Storage

Since there is personal information stored on the system that belongs to registered users is important that the security follow a set standard of rules. Since this application is related to monetary payments there is a set standard that needs to be followed under European law, this is called “The Payment Card Industry Data Security Standard (PCI DSS)” and also after May 2018 there is another standard that needs to be followed called GDPR or “General Data Protection Regulation” in relation to data protection.

For each of these regulations there is a set of standards that need to be followed there is also a heavy fine due to the missus or miss handling of data being stored by the company.

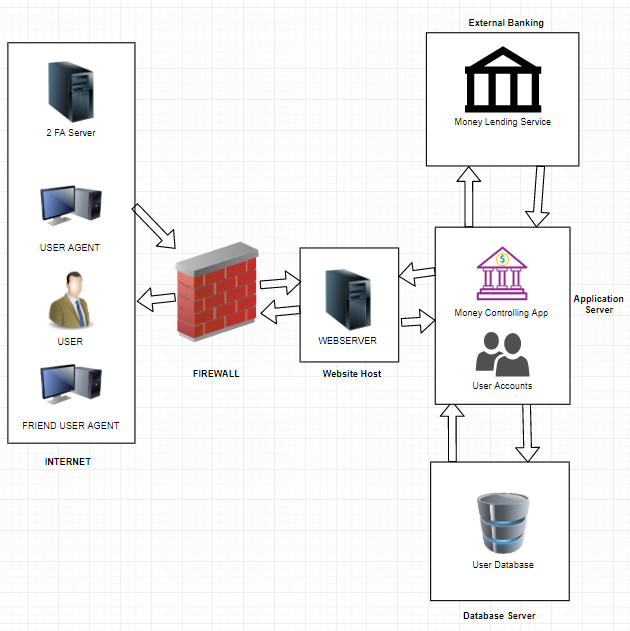
These standards are as follows,

|  |
| --- |
| **PCI DSS Requirements** |
| 1 – Install and maintain a firewall configuration to protect cardholder data |
| 2 – Do not use vendor-supplied defaults for system passwords and parameters |
| 3 – Protect stored cardholder data |
| 4 – Encrypt transmission of cardholder data across open, public networks |
| 5 – Use and regularly update anti-virus software |
| 6 – Develop and maintain secure systems and applications |
| 7 – Restrict access to cardholder data by business need-to-know |
| 8 – Assign a unique ID to each person with computer access |
| 9 – Restrict physical access to cardholder data |
| 10 – Track and monitor all access to network resources |
| 11 – Regularly test security systems and processes |
| 12 – Maintain a policy that addresses information security |

|  |
| --- |
| **GDPR Compliance Checklist** |
| 1 – Establish accountability and a governance framework |
| 2 – Scope and Plan your Project |
| 3 – Conduct a data Inventory and data flow audit |
| 4 – Conduct a detailed gap analysis |
| 5 – Develop an Operational Policy, procedures and processes |
| 6 – Secure personal data through procedural and technical measures |
| 7 – Communication with employees to teach about GDPR |
| 8 – Monitor and audit compliances currently in place |

# Application Overview

This is the section of the report that will mainly look at the overview of the working application and how a user interacts with it the user and services. This is just a generalized diagram based off of the know details from the brief.



## Trust Levels

|  |  |  |
| --- | --- | --- |
| **Trust Levels** | | |
| **ID** | **NAME** | **Description** |
| **1** | Anonymous Web User | An anonymous user connects via web browser to the website |
| **2** | User Login Input | Anonymous becomes user as they provide credentials |
| **3** | 2 Factor Auth Input | A User becomes trusted by the system when 2FA confirmed |
| **4** | Website Admin | Can modify the working website and web application |
| **5** | Server Admin | Can modify system configurations or server side applications |
| **6** | Database Admin | Can modify, add or delete a user from the database |
| **7** | Banking Admin | Controls transfer of funds from users |
| **8** | Transaction Admin | Controls transaction via the application |

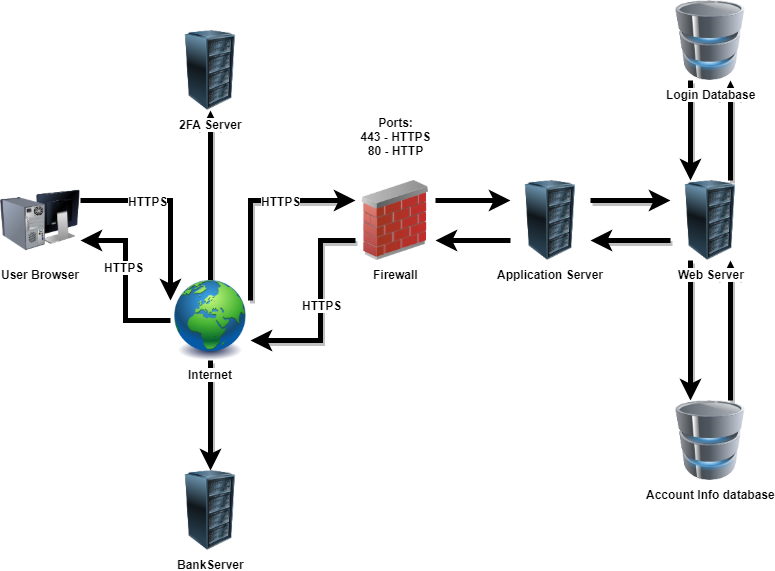
## Technologies and key Features

This section will look at the key technologies used in the device to be able to provide this service to users, below are the operating systems and services running to bring this service to the users.

|  |  |  |  |
| --- | --- | --- | --- |
| **SERVER** | **OPERATING SYSTEM** | **SOFTWARE USED** | **LANGUAGE USED** |
| **Web Server** | Windows Server 2016 | IIS 6 | HTML, CSS, JavaScript |
| **App Server** | Debian Linux 9.11 | Flask | Python |
| **Database Server** | Windows Server 2016 | MySQL 8.1 | MySQL |
| **2 Factor Auth Server** | Debian Linux 9.11 | Flask | Python |

# Architecture Overview

This section will look at the networking of the system and the application, this is generalised and based on the given specifications laid out in the original brief this is what I can assume is how the architecture of the network works.



This would work by the user connecting via the internet to the Application web server while at the same time using an external 2FA authentication services connected such as Google Authenticator.

|  |  |  |
| --- | --- | --- |
| **Trust Levels** | | |
| **ID** | **Name** | **Description** |
| 1 | User | Limited access to the network via the web application |
| 2 | Guest | Limited to No Access via the web application |
| 3 | Administrator | Limited access via the web application |
| 4 | Web Application | Full Access to the server side scripts and Database |

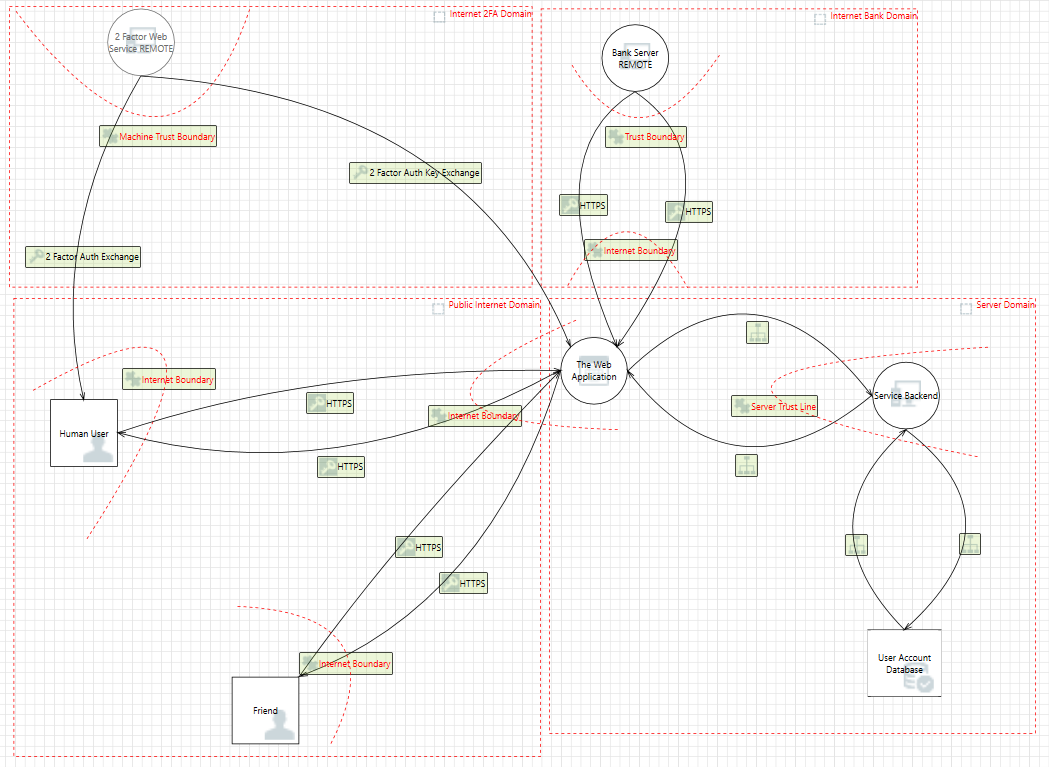
# Assets

In this portion of the report you will find a list of Assets currently being used on the network from my best assumption based on the original network design.

|  |  |  |  |
| --- | --- | --- | --- |
| **Assets** | | | |
| **ID** | **Asset Name** | **Description** | **Impact of Loss** |
| 1 | Server One | Web Application Server | **Severe** |
| 2 | Server Two | Database Server | **Severe** |
| 3 | Server Three | Web Application redundancy server | **Medium** |
| 4 | Server Four | Database redundancy server | **Medium** |
| 5 | Firewall One | Firewall | **Severe** |
| 6 | Firewall Two | Redundancy Firewall | **Medium** |
| 7 | Switch One | Network switch | **Severe** |
| 8 | Switch Two | Redundancy Network Switch | **Medium** |
| 9 | Router One | Network Router | **Severe** |
| 10 | Router Two | Redundancy network Router | **Medium** |
| 11 | Software | Proprietary Scripts and Software | **Severe** |
| 12 | Cabling | Connector Cables | **Severe** |
| 13 | Cabling | Backup Cabling | **Medium** |
| 14 | Server Five | 2 Factor Auth Server | **Severe** |
| 15 | Server Six | 2 Factor Auth Redundancy Server | **Medium** |
| 16 | DATA | User Login Data | **Severe** |
| 17 | DATA | Admin Login Data | **Severe** |
| 18 | DATA | Customer Personal Data | **Severe** |
| 19 | DATA | Finical data | **Severe** |
| 20 | DATA | Session Data | **Severe** |

# Application Decompositions

In this section we will break down the working of the application itself and user interaction, this is based on the assumed workings of the application this is the working model I have of the system. Here you can see the data flow diagram (DFD) that shows the data as it is transmitted between the user and the service. Here you can see the different bounders at which the data is transported and the protocols that are used for the data to be transferred.



## Trust Boundary’s

These are the listing of the Trust boundaries of the above Data Flow Diagram “DFD”

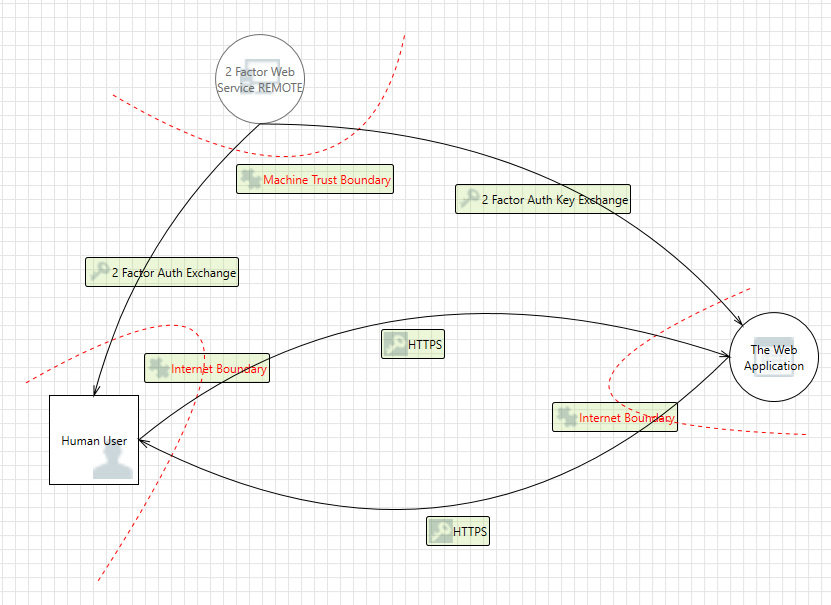
|  |  |
| --- | --- |
| **Boundary** | **Description** |
| User / Web Server | Data is passed over the internet via HTTPs |
| App / Bank | Data is passed over a secure tunnel to the banking service |
| 2FA user / 2FA Bank | Key sent to user and key is also sent to server secured |
| Database Update/Write/Dellete | Information sent via local cabling to Database Server |

# Data Flow

This section will take a look at the functions of the application when a user interacts with the features.

## Step 1 – 2 Factor Authentication

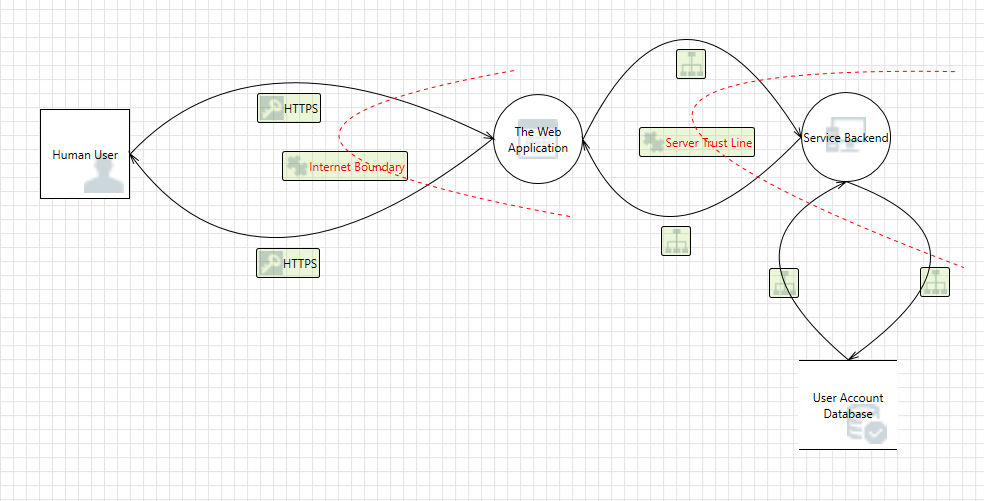
The user contacts to the 2 factor authenticate to get the 2FA temporary key



1. User request Key from the 3rd party key exchange service such as Google Auth
2. Server constantly maintains connection with key exchange server for checking
3. User inputs key in the time frame
4. Server compares the key from the exchange to the key from the user

## Step 2 – User Login

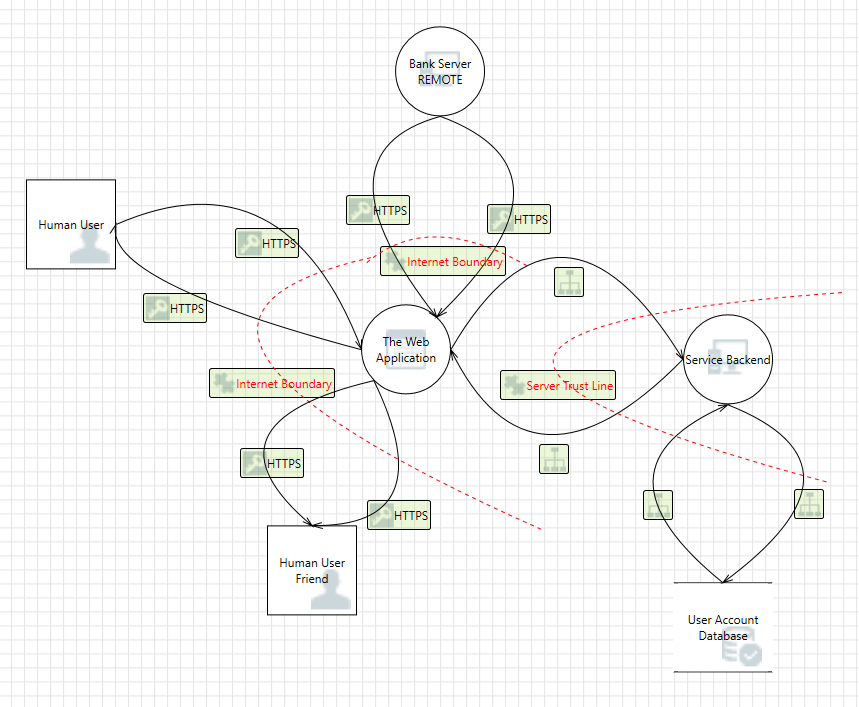
When user logins and passes 2 factor authentication the server then confirms the username and password against the account database and allows the data to be transmitted from the database.



1. User visits webpage
2. User sends login credentials to the service
3. Server waits for the 2FA key
4. Server compares login credentials against the Database

## Step 3 – Transfer to user

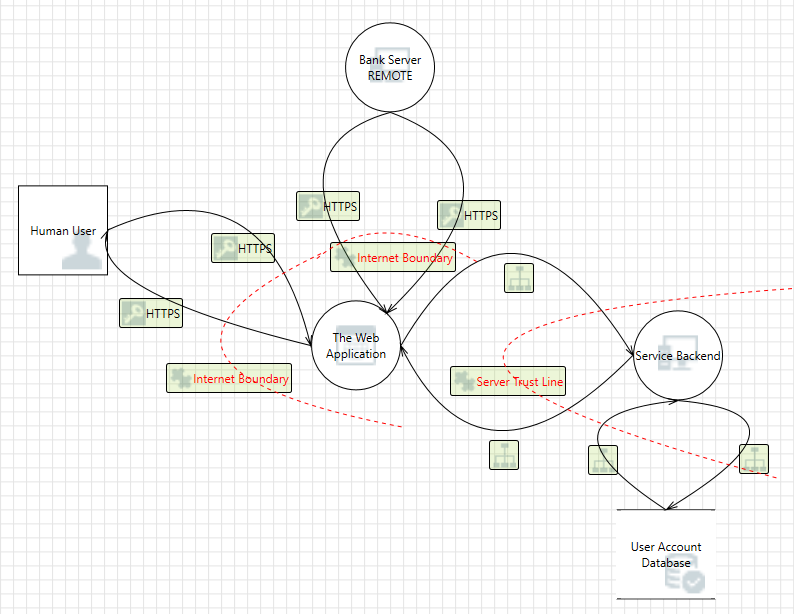
When money is sent to the user the bank is contacted via the web server / web application,



1. User Request money via web application to be sent to friend
2. Web Application checks against account Database to confirm user and amount available
3. Backend script requests Bank API to move money from one account to another
4. User account details are updated in the database
5. A Notification is sent to the Friends account announcing they have received money.

## Step 4 – Transfer to bank

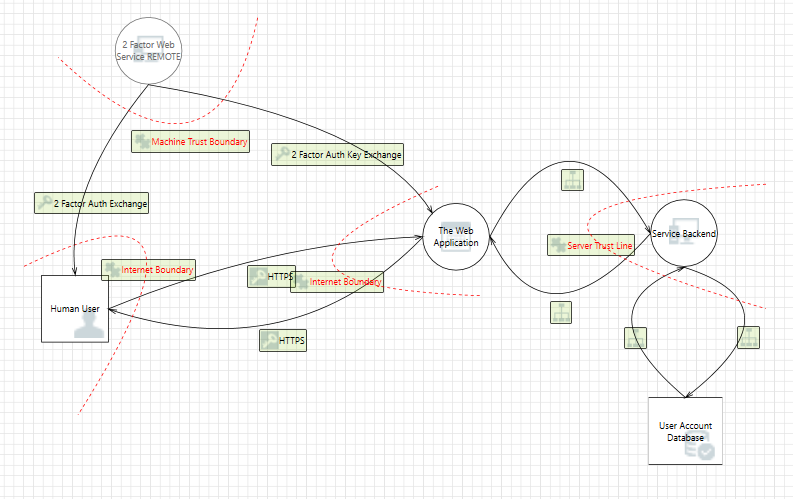
For a user to transfer money to an account the local database is check for information about the users account



1. User requests a transferee of money from the service to the user account via web application.
2. The server checks the database to see if the user has the amount
3. The amount is then transferred by the server via the api from the service bank account to the users bank account
4. A notification is sent to the user when the transfer is complete.

## Step 5 – User Registration

Here you will find the steps involved for a new user to be able crate a new user account



1. User visits webpage
2. User submits username, email, password to registration form
3. Server sends conformation email
4. Once account is confirmed it is added to the database
5. A 2FA key is generated based off a 3rd party key server and displayed to user to setup

# Network Entry Point

In this section you will find a table containing all the entry points that could be compromised.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Points** | | | |
| **ID** | **Name** | **Description** | **Trust Level** |
| 1 | HTTPS  Port 443 | The Website will only be available via SSL or TLS 3.0 | Anonymous User  Regular User  Guest  Administrator |
| 2 | Homepage  Port 80 / 443 | Main Webpage | Anonymous User  Regular User  Guest  Administrator |
| 3 | Login Page  Port 80 / 443 | Location where a user can login | Anonymous User  Regular User  Guest  Administrator |
| 4 | Login Function  Port 80 / 443 | Function to login into an account | Anonymous User  Regular User  Guest  Administrator |
| 5 | Account Page  Port 80 / 443 | Main display of the account page | User Logged in  Administrator |
| 6 | Banking Page  Port 80 / 443 | Page to display banking details | User Logged in  Administrator |
| 7 | Account Setting Page  Port 80 / 443 | Page to display user account settings | User Logged in  Administrator |
| 8 | Transfer Page  Port 80 / 443 | Money Transfer page | User Logged in  Administrator |
| 9 | Friends Page  Port 80 / 443 | Page with friends listed | User Logged in  Administrator |
| 10 | Email Reset Page  Port 80 / 443 | Page to Reset email address | User Logged in  Administrator |
| 11 | 2FA Reset Page  Port 80 / 443 | 2 Factor Auth request rest | User Logged in  Administrator |
| 12 | Confirmation Page  Port 80 / 443 | Money Transfer Confirmation page | User Logged in  Administrator |

# Network Exit Points

Here you will find all the exit points on the web application and server.

|  |  |  |  |
| --- | --- | --- | --- |
| **Network Exit Points** | | | |
| **ID** | **Name** | **Description** | **Trust Level** |
| 1 | Webserver To Database | Webserver connection to database for sending login and registration details | Administrator |
| 2 | Application to Bank | Transfer of information to the Bank via API connection | User Logged in  Administrator |
| 3 | Web application to User | Data transfer from the server to the user regarding account details | User Logged in  Administrator |
| 4 | 2FA key Exchange | Key Exchange from external 3rd party service the server for conformation | Administrator |

# Threats

## Stride Areas

In this portion of the report you will find information relating to Threats to the service.

|  |  |
| --- | --- |
| **Property** | **Description** |
| **Confidentiality** | Information is not made available to does without proper authorization to have access to them |
| **Integrity** | Information cannot be tampered with or modified by any other user who doesn’t not have proper authorization |
| **Availability** | Systems maintain a proper uptime and are available as needed by the users |
| **Authentication** | To confirm the person with the right authorization as access to what is needed |
| **Authorization** | A user only has access to the files at which they have permission to access |
| **Nonrepudiation** | An attack or a user cannot preform any tasks without being held accountable in some form. |

## Stride

The Stride table is a set of security properties developed by Microsoft to assistant in the identification of threats tough six main attack vectors of an application or system. These are listed below as an acronym of the Word Stride, each with its own area of security.

|  |  |  |  |
| --- | --- | --- | --- |
| **Stride Threat List** | | | |
|  | **Type** | **Example** | **Security Property** |
| **S** | Spoofing | Involved faking another users or services credential’s | Authentication |
| **T** | Tampering | Involved modification or changing of data while in transit | Integrity |
| **R** | Repudiation | Involves an attacker not being able to be traced after commencing an attack | Non-Repudiation |
| **I** | Information Disclosure | Ability to re data that should not be able to be read without privilege | Confidentiality |
| **D** | Denial of Service | Involves making the service un-accessible or un-useable for other users | Availability |
| **E** | Elevation of Privilege | Involves an attacker getting higher privileges then granted on the system | Authorization |

## Stride table

In this portion of the report you will see the stride table being used in the building of this report.

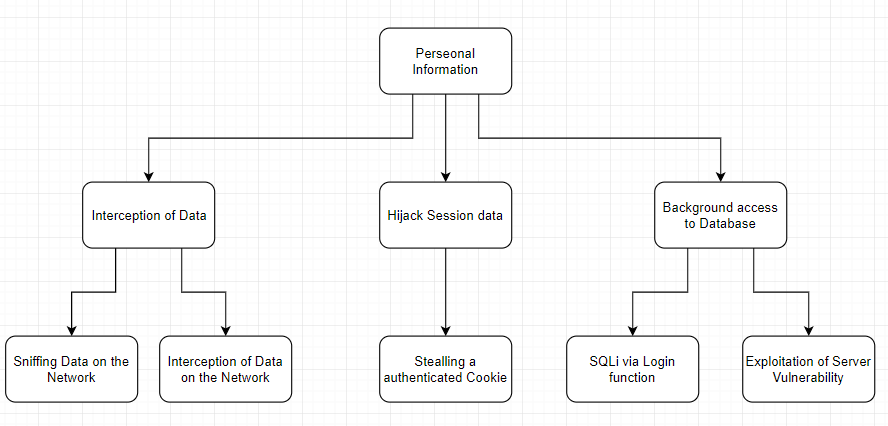
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Elements** | **Threats** | | | | | | **Application Applicable** |
|  | S | T | R | I | D | R |  |
|  |  | X |  | X | X |  | * HTTPS * UDEP * System Calls |
|  |  | X |  | X | X |  | * User Account Storage * Finial Data Storage * User Login Details |
|  | X | X | X | X | X | X | * Login Function * Registration Function * Transfer Money * Check Account Details |
|  |  |  |  |  | X | X | * Hosting of 2fa keys * Transfer of 2Fa Keys |
|  | X | X | X | X | X | X | * Backend Scripting * Transfer of Information * Web App hosting |
|  | X |  | X |  |  |  | * Administrator * User |

# Threat Tree

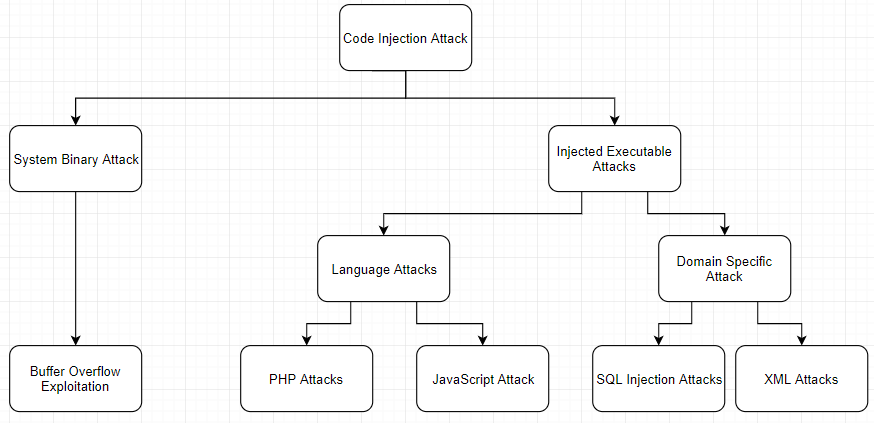
This section will have the use the Threat Tree model to show the possible systems that maybe attacked and how data is transferred, or who an attacker can gain access to different systems based on what type of exploit they have gained on the system.

The way to follow these threat trees is to follow the point of exploitation down the tree to see locations that can be further exploited, privilege escalated into or what information is sealable.

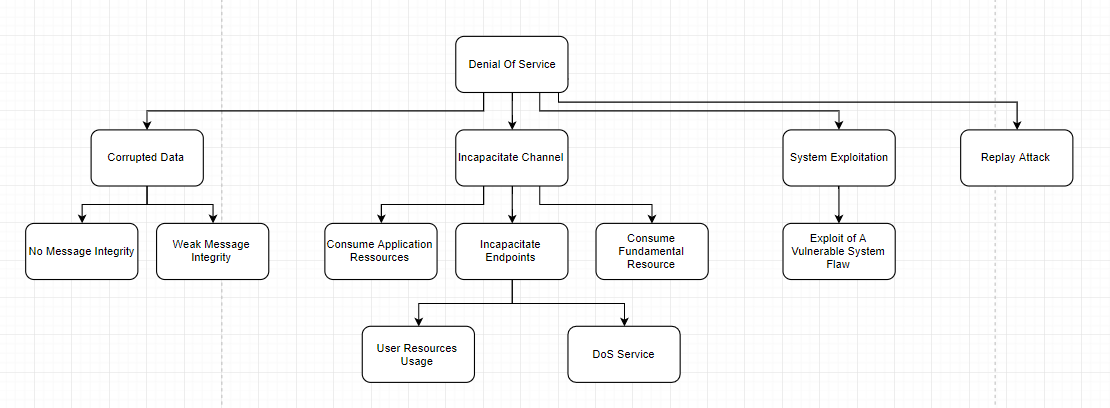
## Access to Personnel Information



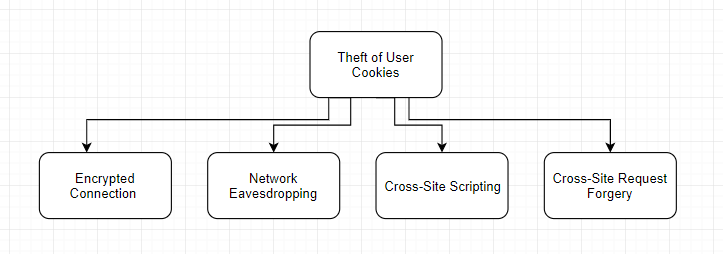
## Code Injection Attack Threat Tree



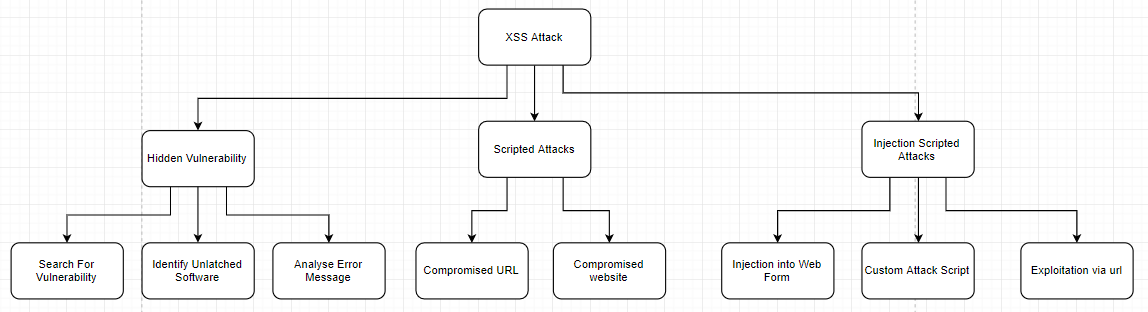
## Denial of Service Threat Tree



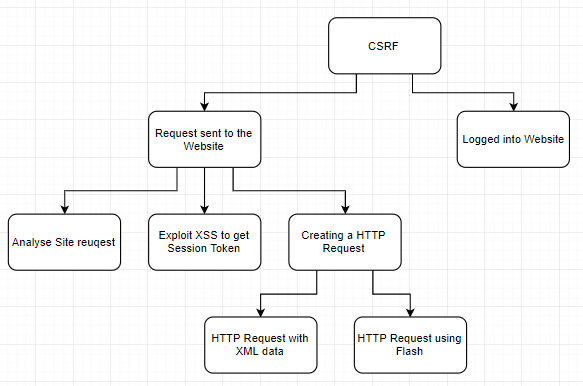
## Theft of Authorization Cookies Threat Tree



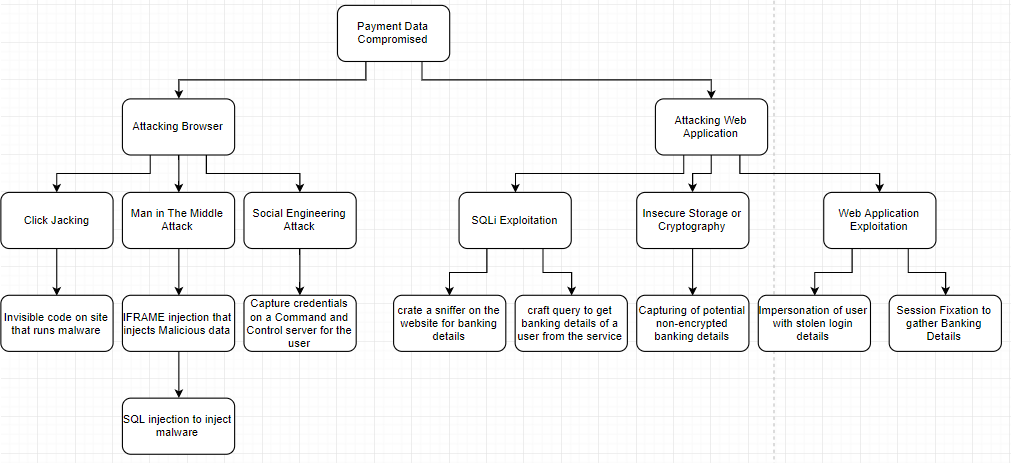
## XSS Attack Threat Tree



## CSRF Attack Tree



## Compromise of Banking Details



# Threat Tree Reference

1. Personnel Information
   1. Interception of Date
      1. Sniffing data on the network
      2. Interception of Data on the network
   2. Hijack Session data
      1. Stealing Authenticated Cookies
   3. Background Access to Database
      1. SQLi via Login Function
      2. Exploitation of Server Vulnerability
2. Code Injection Attack
   1. System Binary Attack
      1. Buffer Overflow Exploitation
   2. Injected Executable
      1. Language Attacks
         1. PHP attacks
         2. JavaScript Attacks
      2. Domain Specific Attacks
         1. SQL Injection Attacks
         2. XML Attacks
3. Denial of Service
   1. Corrupted Data
      1. No Message Integrity
      2. Weak Message Integrity
   2. Incapacitate Channel
      1. Custom Application Resources
      2. Incapacitate Endpoints
         1. User Resource Usage
         2. DoS Service
      3. Consume Fundamental resources
   3. System Exploitation
      1. Exploit of a Vulnerable System Flaw
   4. Replay Attack
4. Theft of User Cookies
   1. Encrypted Connection
   2. Network Eavesdropping
   3. Cross-Site Scripting
   4. Cross-Site request Forgery
5. XSS Attack
   1. Hidden Vulnerability
      1. Search for Vulnerabilities
      2. Identify unpatched software
      3. Analyze Error Messages
   2. Scripted Attacks
      1. Compromised URL
      2. Compromised Website
   3. Injection Scripted Attacks
      1. Injection into Web Form
      2. Custom Attack Script
      3. Exploitation Via URL
6. CSRF
   1. Request Sent to the Website
      1. Analyze Site request
      2. Exploit XSS to get Session Token
      3. Creating a HTTP Request
         1. HTTP Request with XML data
         2. HTTP Request use Flash
   2. Logged into Website
7. Payment Data Comprised
   1. Attacking Browser
      1. Click Jacking
         1. Invisible code on site that runs malware
      2. Man in the middle
         1. IFRAME Injection that injects malicious data
            1. SQL Injection to Inject Malware
      3. Social Engineering Attack
         1. Capture Credentials on a command and controller server for the user
   2. Attacking Web Application
      1. SQLI Exploitation
         1. Crate a sniffer on the website for banking details
         2. Craft query to pull banking data from the service
      2. Insecure Storage or Cryptography
         1. Capturing of potential non-encrypted banking details
      3. Web Application Exploitation
         1. Impersonation of a user with stolen login details
         2. Session fixation to gather banking details

# Vulnerabilities

In this section we will look at the potential vulnerabilities that may affect the system being reported on, this will be evaluated on the DREAD system a way of calculating the risk level of potential vulnerabilities.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Rating** | **HIGH (3)** | **Medium (2)** | **Low (1)** |
| **D** | Damage Potential | Attack could gain control | Leaking of sensitive information | Leaking of trivial information |
| **R** | Reproducibility | Attack could be carried out multiple times | Can only be carried out under certain conditions | Attacks requires specific set of conditions to be attackable |
| **E** | Exploitability | No skill required | A semi-skilled attack can use this exploit | This attack requires a very skilled attack |
| **A** | Affected Users | All users including administrator | Some users effected | Little amount of users or anonymous users |
| **D** | Discoverability | Known vulnerability published publically | Not a very well-known exploit or exploit code is not public | Vulnerability is obscure and or no public exploit is available |

## Dread Table

Here you will find the details of the vulnerabilities assumed to be in the application for this report. You will also find an individual breakdown of each threat after the main table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Threat** | **D** | **R** | **E** | **A** | **D** | **Total** | **Rating** |
| Attack captures User details by Sniffing traffic | 3 | 3 | 3 | 3 | 2 | 14 | HIGH |
| SQLi attack on the web app to get data from database | 3 | 3 | 3 | 3 | 2 | 14 | HIGH |
| Cross Site Scripting “XSS” | 2 | 3 | 2 | 3 | 3 | 13 | HIGH |
| Denial of Service Attack “DoS” | 1 | 3 | 2 | 3 | 3 | 12 | HIGH |
| Cross site Request Forgery “CSRF” | 3 | 3 | 2 | 3 | 3 | 14 | High |
| Lack of System protection | 3 | 3 | 3 | 3 | 3 | 15 | High |
| Bad System configurations | 3 | 2 | 2 | 3 | 1 | 11 | Medium |
| Leak of company email addresses | 1 | 3 | 1 | 1 | 1 | 6 | Low |

|  |  |
| --- | --- |
| **Threat Description** | **Attack captures User details by Sniffing traffic** |
| **Threat Target** | Attempting to capture data of a user on the same network by capture insecure data |
| **Risk Rating** | High |
| **Attack Techniques** | Use of tools to produce a man in the middle and capture network traffic |
| **Countermeasures** | Implement basic network protocol such as SSL and TLS to secure connection |

|  |  |
| --- | --- |
| **Threat Description** | **SQLi attack on the web app to get data from database** |
| **Threat Target** | The main Web Application, URL or any location at which an attacker could enter database related query’s |
| **Risk Rating** | High |
| **Attack Techniques** | Manual injection or automated injection via script |
| **Countermeasures** | Sanitization of the user input to stop an attacker adding dangerous imputs |

|  |  |
| --- | --- |
| **Threat Description** | **Cross Site Scripting “XSS”** |
| **Threat Target** | The Main web application and locations such as Search functions |
| **Risk Rating** | High |
| **Attack Techniques** | Manual injection or automated injection via script |
| **Countermeasures** | Sanitizing user input data as it is being searched in the function |

|  |  |
| --- | --- |
| **Threat Description** | **Denial of Service Attack “DoS”** |
| **Threat Target** | The web application and the web server itself |
| **Risk Rating** | High |
| **Attack Techniques** | Via a scripted application to send constant requests to the server |
| **Countermeasures** | Load balancing and denial of the packets could protect against this issue |

|  |  |
| --- | --- |
| **Threat Description** | **Cross site Request Forgery “CSRF”** |
| **Threat Target** | The user using the web application |
| **Risk Rating** | High |
| **Attack Techniques** | Via scripted application and exploitation of the main page |
| **Countermeasures** | Using a token based mitigation system will help prevent against this type of attack |

|  |  |
| --- | --- |
| **Threat Description** | **Lack of System protection** |
| **Threat Target** | The web server |
| **Risk Rating** | High |
| **Attack Techniques** | Checked via vulnerability scanning on the application |
| **Countermeasures** | Keep the system up to date and regularly check for vulnerabilities |

|  |  |
| --- | --- |
| **Threat Description** | **Bad System configurations** |
| **Threat Target** | The Web Server |
| **Risk Rating** | Medium |
| **Attack Techniques** | Checked via vulnerability scanning and exploit Database |
| **Countermeasures** | Check your configurations and confirm with a security professional |

|  |  |
| --- | --- |
| **Threat Description** | **Leak of company email addresses** |
| **Threat Target** | Details left on website by accident |
| **Risk Rating** | Low |
| **Attack Techniques** | Emails can be used in social engineering campaigns |
| **Countermeasures** | Hiding critical information from websites |

# Analysis Conclusion

Based on the above information gathering in this report and using the STRIDE and DREAD models we can get an idea of possible issues that maybe occurring or may turn into vulnerabilities in the application. There are a few issues that where a cause for concern of when it comes to the application these are outlined above.

## Risk mitigations

|  |  |
| --- | --- |
| **Threat Type** | **Mitigation techniques** |
| **Spoofing Identify** | * Setting up a secure authentication System * Protecting data that would stop attackers from spoofing |
| **Tampering with Data** | * Adding Digital Signatures to stop file tampering * Checking Hash values against each other’s * Using Tamper Resistant Protocols * Using Good Authorization methods for exams |
| **Repudiation** | * At every possibility instance do testing and auditing * Make sure to timestamp the files * Digital signatures should be used on files to find usages |
| **Information disclosure** | * Encryption of all files and data * Allowing proper authorization to access data * Using privacy protocols on the files and information |
| **Denial Of Service** | * Setting Blocking firewall rules * Activating Quality of Service functions * Setting up a load balancer on the network * Filtering Packets based on amount sent from host |
| **Elevation of Privileges** | * Only allow lowest possible privilege for a user or guest * Force Two factor authentications on users and administrators to prevent unauthorized access |

# Reference

* OWASP. (2017, May 31). *Application Threat Modelling*. Retrieved from OWASP: https://www.owasp.org/index.php/Application\_Threat\_Modeling
* OWASP. (2017, August 21). *Threat Modeling Cheat Sheet*. Retrieved from OWASP: https://www.owasp.org/index.php/Threat\_Modeling\_Cheat\_Sheet
* OWASP. (2017, July 13). *Threat Risk Modeling*. Retrieved from OWASP: https://www.owasp.org/index.php/Threat\_Risk\_Modeling
* Microsoft.com. (2019). *Microsoft Security Development Lifecycle Threat Modelling*. [online] Available at: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling [Accessed 25 Oct. 2019].
* Docs.microsoft.com. (2019). *Microsoft Threat Modeling Tool - Azure*. [online] Available at: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool [Accessed 25 Oct. 2019].
* Owasp.org. (2019). *Threat Risk Modeling - OWASP*. [online] Available at: https://www.owasp.org/index.php/Threat\_Risk\_Modeling [Accessed 25 Oct. 2019].
* Owasp-aasvs.readthedocs.io. (2019). *1.6 A STRIDE threat model has been produced — OWASP Annotated Application Security Verification Standard 3.0.0 documentation*. [online] Available at: https://owasp-aasvs.readthedocs.io/en/latest/requirement-1.6.html [Accessed 25 Oct. 2019].
* Ruiz, G. (2019). *OWASP Top 10 Security Risks – Part IV*. [online] Sucuri Blog. Available at: https://blog.sucuri.net/2019/01/owasp-top-10-security-risks-part-iv.html [Accessed 25 Oct. 2019].
* Itgovernance.eu. (2019). *GDPR Compliance Checklist | IT Governance Ireland*. [online] Available at: https://www.itgovernance.eu/en-ie/key-steps-to-gdpr-compliance-ie [Accessed 25 Oct. 2019].
* Itgovernance.eu. (2019). *What is the PCI DSS? Ireland*. [online] Available at: https://www.itgovernance.eu/en-ie/what-is-the-pci-dss-ie [Accessed 25 Oct. 2019].