

**Application Security: Threat Modelling Assignment**

**James Finglas**

**B00094138**

Department of Informatics

School of Informatics and Engineering

Technological University Dublin

Submitted to Technological University Dublin in Partial Fulfilment of the requirements for the degree of:

Bachelor of Science (Honors) in Computing

Course: Application Security

Lecturer: [Stephen O'Shaughnessy](https://moodle.itb.ie/user/view.php?id=9218&course=1309)

Submission Date: 27/10/2019

# Lab Submission/Declaration Form

Submitted by: James Finglas

Course Coordinator: Stephen O'Shaughnessy

Module Name: Application Security

Lecturer: Stephen O'Shaughnessy

Submission Date: 27/10/2019

I declare that the work contained in this report which I now submit on the program of study leading to the award of Degree of Honors B.Sc. in Computer Science in Technological University for Dublin is my own work and has not been taken from the work of others. Any sources which have been referenced or analyzed have been cited using the Harvard/IEEE standard within the body of this report. It is entirely my own work except where otherwise stated and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilments of the requirements stated above.

I have read and understood the Technology University Dublin’s policy regarding plagiarism.

X

James Finglas

# Abstract

This assignment tackles developing a new threat model for a newly formed companies’ application. This will be a web-based application. The report Identifies company assets which must be protected and documents an architectural overview of the app. The report further decomposes the app down into its constituent functions and processes for the purposes of identifying trust boundaries. Threats are then identified and documented before being rated. Lastly the report documents the threat model which this process allows us to develop.

Contents

[Lab Submission/Declaration Form 2](#_Toc22231756)

[Abstract 3](#_Toc22231757)

[Table of Figures: 5](#_Toc22231758)

[Table of Tables: 5](#_Toc22231759)

[1.0 Introduction 6](#_Toc22231760)

[1.2 Summary 6](#_Toc22231761)

[1.3 Methodology 7](#_Toc22231762)

[1.4 Research Goals 7](#_Toc22231763)

[2.0 Primary Research Outcomes 8](#_Toc22231764)

[2.1 Creation of Security Profile 8](#_Toc22231765)

[2.2 Identify assets 10](#_Toc22231766)

[2.2 Create an Architecture overview 11](#_Toc22231767)

[2.3 Decompose the Application 13](#_Toc22231768)

[2.4 Identify Threats 15](#_Toc22231769)

[2.4.1 Network Threats 15](#_Toc22231770)

[2.4.2 Host Threats 15](#_Toc22231771)

[2.4.3 Application Threats 16](#_Toc22231772)

[2.5 Document the threats 17](#_Toc22231773)

[2.6 Rate the threats 28](#_Toc22231774)

[3.0 Conclusions 33](#_Toc22231775)

[4.0 Glossary 34](#_Toc22231776)

[5.0 Appendix 1: 35](#_Toc22231777)

[6.0 References 36](#_Toc22231778)

# Table of Figures:

[Figure 1 Listing assets table pertaining to the UrWallet.ie application development process. 10](#_Toc22489566)

[Figure 2 High level Architectural overview visualization for UrWallet.ie. 11](#_Toc22489567)

[Figure 3 A user / admin begins the process of logging in over a secure SSL connection. 12](#_Toc22489568)

[Figure 4 The logon server begins to authenticate the user’s username and password and identifies their privilege level if successfully authenticated. 12](#_Toc22489569)

[Figure 5 The logon server issuing a two-factor authentication code to the users email address or mobile device phone number. 13](#_Toc22489570)

[Figure 6 The logon server serves up logon pages to the back-end server which serves the pages up to the user, via secure SSl connections. 13](#_Toc22489571)

[Figure 7 Here we see typical post logon validation interaction between the user and the backend server which serving up pages, after access the database to retrieve information, all over secure SSL channels. 14](#_Toc22489572)

[Figure 8 Here we see the back-end server serving up transaction data to the users personal external bank via secure encrypted channels. 15](#_Toc22489573)

[Figure 9 UrWallet.ie decomposed view. 16](file:///D:\Storage\USB%20Backup\Year%204\Applicaton%20Security\Application%20Security-Threat%20Modelling%20Assignment.docx#_Toc22489574)

[Figure 10 Attack tree 1 Detailing a man in the middle attack. 20](#_Toc22489575)

[Figure 11 Attack tree 2 detailing an IP spoofing attack. 21](#_Toc22489576)

[Figure 12 Attack tree 3 detailing a port scan attack. 22](#_Toc22489577)

[Figure 13 Attack tree 4 detailing a DNS / Routing table / MAC table / TCP stack attack. 23](#_Toc22489578)

[Figure 14 Attack tree 5 detailing a direct service attack. 24](#_Toc22489579)

[Figure 15 Attack tree 6 detailing insecure account privilege level attack. 25](#_Toc22489580)

[Figure 16 Attack tree 7 detailing a brute force encryption attack. 26](#_Toc22489581)

[Figure 17 Attack tree 8 detailing poor input validation leading to injection attacks. 27](#_Toc22489582)

[Figure 18 Attack tree 9 detailing a repudiation attack. 28](#_Toc22489583)

[Figure 19 Attack tree 10 detailing an insecure code access attack. 29](#_Toc22489584)

[Figure 20 Attack tree 11 detailing poor error handling leading to potential direct service attack. 30](#_Toc22489585)

# Table of Tables:

[Table 1 detailing the security profile pertaining to input validation. 8](#_Toc22489586)

[Table 2 detailing the security profile pertaining to authentication. 8](#_Toc22489587)

[Table 3 detailing the security profile pertaining to authorization. 9](#_Toc22489588)

[Table 4 detailing the security profile pertaining to configuration management. 9](#_Toc22489589)

[Table 5 detailing the security profile pertaining to sensitive data. 9](#_Toc22489590)

[Table 6 detailing the security profile pertaining to parameter manipulation 9](#_Toc22489591)

[Table 7 detailing the security profile pertaining to error handling 9](#_Toc22489592)

[Table 8 detailing the security profile pertaining to logging & auditing. 10](#_Toc22489593)

[Table 9 Detailing a man in the middle attack. 20](#_Toc22489594)

[Table 10 detailing an IP spoofing attack. 21](#_Toc22489595)

[Table 11 detailing a port scanning attack. 22](#_Toc22489596)

[Table 12 detailing DNS / ROUTING / MAC table & TCP stack attacks. 23](#_Toc22489597)

[Table 13 detailing direct service /OS attack. 24](#_Toc22489598)

[Table 14 detailing insecure account accounts privilege attack. 25](#_Toc22489599)

[Table 15 detailing brute force encryption attack. 26](#_Toc22489600)

[Table 16 detailing injection attacks. 27](#_Toc22489601)

[Table 17 detailing repudiation attacks. 28](#_Toc22489602)

[Table 18 detailing insecure code attack. 29](#_Toc22489603)

[Table 19 detailing poor error handling attacks. 30](#_Toc22489604)

[Table 20 detailing a man in the middle attack with threat rating added. 31](#_Toc22489605)

[Table 21 detailing an IP spoofing attack with threat rating added. 31](#_Toc22489606)

[Table 22 detailing a port scanning attack with threat rating added. 32](#_Toc22489607)

[Table 23 detailing DNS / ROUTING / MAC table & TCP stack attacks with threat rating added. 32](#_Toc22489608)

[Table 24 detailing direct service /OS attack with threat rating added. 32](#_Toc22489609)

[Table 25 detailing insecure account accounts privilege attack with threat rating added. 33](#_Toc22489610)

[Table 26 detailing brute force encryption attack with threat rating added. 33](#_Toc22489611)

[Table 27 detailing injection attacks with threat rating added. 33](#_Toc22489612)

[Table 28 detailing repudiation attacks with threat rating added. 34](#_Toc22489613)

[Table 29 detailing insecure code attack with threat rating added. 34](#_Toc22489614)

[Table 30 detailing poor error handling attacks with threat rating added. 35](#_Toc22489615)

# Introduction

For this assignment we are tasked with developing a working threat model and visualizing it with our chosen software platform. We are provided with a fictional company, ‘UrWallet.ie’. The company is about to develop a new app with specific functionality in relation to financial web activity. The app will incorporate the follow functionality criteria:

* Two-factor authentication.
* User account pages which must list all transactions.
* Secure funds transfer to other accounts.
* Secure funds transfer to and from the user’s bank.
* Secure friend connection (To allow for sharing or receipts, bills and private messaging).

## 1.2 Summary

The first task I tackled was to identify the company assets as they pertain to the application and its functionality. These will all need to be protected by applicable trust boundaries throughout the program. For the purposes of this assignment we will acknowledge assets outside of the program, but we will only address them within the threat model if those assets create a data flow through the program or data which may be stored within the program.

Next, I created an architectural overview of the program. This is a high-level visual structure of the program outlining the basic assets, functions, data flow and interactions of the application and establishing basic trust boundaries.

Having established a high-level overview of the application, I began to decompose the application into its sub-systems including the network structure and the infrastructure of the host system. This allows me to begin to create a security profile for the application and identify threats.

The final three steps of the process of developing a threat model fold into each other. They encompass the identification of threats against the app. Once identified a threat must documented using a common threat template. The treat template defines a core set off attributes to capture for threats. Lastly the threats must be rated from highest to lowest regarding their impact on the system or risk of exposure of assets.

## 1.3 Methodology

For this assignment I elected to use The Microsoft Threat Modeling Tool 2016 [1]. The unified modeling language (UML) program UMLet was also considered in scope for the high-level diagrams designed for the assignment [2]. I based my entire design process on the Microsoft Threat Modeling Theorem [3].

## 1.4 Research Goals

To develop a threat model and visualize said threat model for a company’s new app and identify trust boundaries, and threats against the company’s assets within the functionality of the app. Once threats have been identified they will be rated according to severity and priority. The end goal is produce a design template aimed at guiding the development team through the security demands and objectives of the application throughout its software design life cycle (SDLC).

# Primary Research Outcomes

We begin the design process of our Threat Model by creating a security profile and identifying the assets which are of value to the company and likely to be the subjects of attack or invasion and must therefore be protected.

## 2.1 Creation of Security Profile

We must create a security profile to outline the steps we wish to take, what we wish to prevent and what we might need to do to prevent it further on in the threat model. The following tables outline the security profile I have created.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Input validation | Is user input adequately validated? |
|  | Is validation repeated at each trust boundary? |
|  | Are measure taken to ensure data integrity? |
|  | Are accounts created without root privileges? |
|  | Can an attacker inject code into user input forms? |
|  | Is data encrypted as it passes through the network |

Table detailing the security profile pertaining to input validation.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Authentication | Is data integrity verified? |
|  | Are SSL / TLS / RSA certificates used? |
|  | Are certificates validated and updated regularly? |
|  | Are passwords hashed and salted? |
|  | Can an attacker inject code into user input forms? |
|  | Is data encrypted as it passes through the network |
|  | Is two-factor authentication enforced? |

Table detailing the security profile pertaining to authentication.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Authorization | Is there a gatekeeper at each access point such as a firewall, intrusion detection system and / or intrusion prevention system? |
|  | Is access only granted upon successful authorization and authentication? |
|  | Is access to the servers secured? |
|  | Are any administration console ports secured with strong encrypted passwords? |

Table detailing the security profile pertaining to authorization.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Configuration Management | Are the most up to date security measures applied? |
|  | Are adequate firewall rules applied to prevent port attack and unauthorized access |
|  | Is administration access to firewalls, servers, routers and databases secured. |

Table detailing the security profile pertaining to configuration management.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Sensitive data | How are sessionID’s processed by the application? |
|  | Are sessionIDs encrypted and stored securely? |
|  | Are sessionId’s hashed and salted? |
|  | Is a strong enough encryption protocol used. |
|  | How often are encryption keys and certificates renewed and updated? |

Table detailing the security profile pertaining to sensitive data.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Parameter Manipulation | Are prepared statements used? |
|  | Is cross site request forgery screening employed? |
|  | Is cross site scripting prevention employed? |
|  | Are white lists used? |
|  | Are all input forms validated and screened? |
|  | Are HTTP headers, fields, state and cookies validated? |

Table detailing the security profile pertaining to parameter manipulation

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Error Handling | Are exceptions screened to only allow custom generic or specifically created messages to be viewed by users. |
|  | Is all data leakage via error responses secured? |

Table detailing the security profile pertaining to error handling

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Logging & Auditing | Is adequate logging employed? |
|  | Are any intrusion detection and prevention systems also logging? |
|  | How often are logs audited and is adequate error and warning software employed? |
|  | How long are logs stored? |

Table detailing the security profile pertaining to logging & auditing.

## 2.2 Identify assets

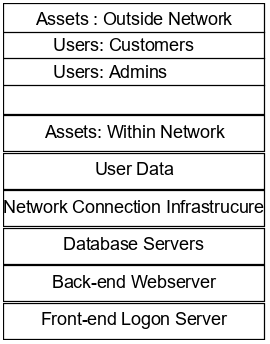


Figure Listing assets table pertaining to the UrWallet.ie application development process.

.

We first identified assets outside of the physical structure of the application and its processes. These consist of human agents. The customer natural is the source of most of the data within the network. Admins too are likely to maintain the network. Next, we identify assets within the physical structure of the application.

We begin with user-data is not just an asset, but perhaps the most valuable asset within the structure. It the source of all business within the application. Thus, the Data must be protected with proper implemented of security protocols regarding trust boundaries. Protecting data is the primary concern that extends to all other areas of protection. Which it is important to protect devices and connections to maintain the business model, it is of primary concern that the data be protected. Therefore, data protection is the number one priority with business continuity following at a close second.

The network connection physical and logical topology is next to be documented. This is the connection type and layout and security of the network upon which the platform is built. For example, both WIFI and physical cable connection to the front-end logon server will be required. This necessitates multiple secure protocols, as well as creating several attack vectors which are recorded later in this process. This asset must be secure at every stage of every connection both physically and logically.

Next, we must acknowledge the physical devices. We have three major servers within the application. First up we have the logon server which will authenticate users who attempt to logon to the service. Naturally this is the gateway the service and the portal through which all business is conducted, and all transactions are initiated so the logon on server is a strong asset. Secondly, we have the database server or servers which contain the data pertinent to the systems and its functionality. This data, as previously mentioned, is the principal asset, and thus the container; the database server, is also an important asset. Lastly. We have the back-end webserver gather from the user account data, all data related to the requests pages as requested by the user and although the details will not eb stored on this server, the data which will be transmitted via this server must be protected and the server must be protected as a data protection measure.

## 2.2 Create an Architecture overview

I began with a high-level overview of the app showing the entire functionality of each device.

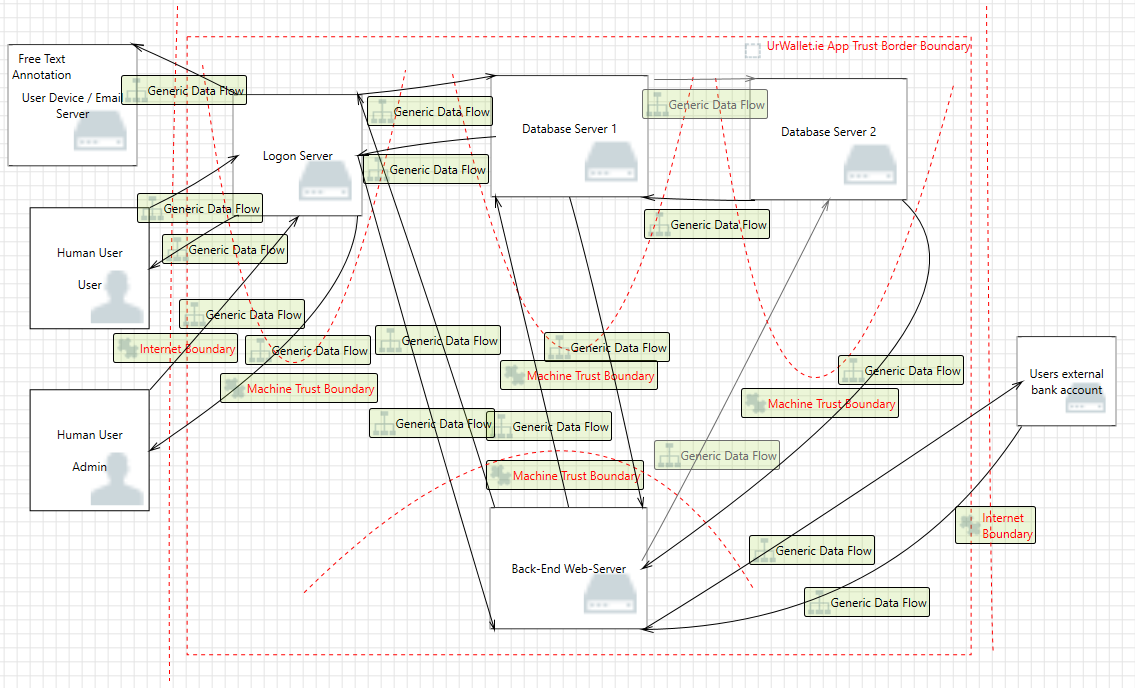


Figure High level Architectural overview visualization for UrWallet.ie.

User/admin: Users will need to access the system via either a browser or a direct access control point in a mobile device app. This will necessitate authentication protocols which incorporate two-factor authentication. This authentication will also need to be multi-tier with privilege escalation for administrators of the application.

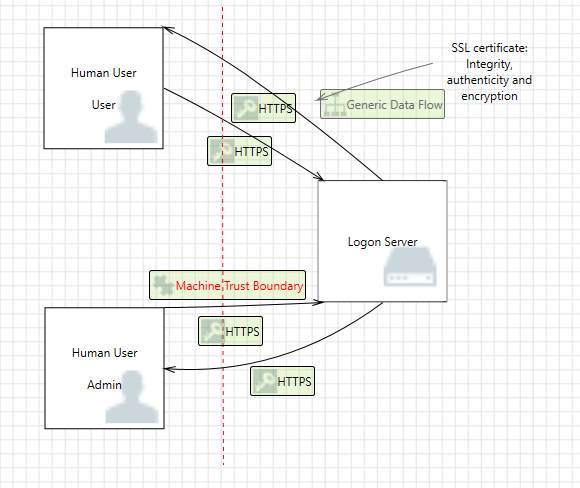


Figure A user / admin begins the process of logging in over a secure SSL connection.

Once the user begins the login process, they will enter a username and password, at which time these will be validated.

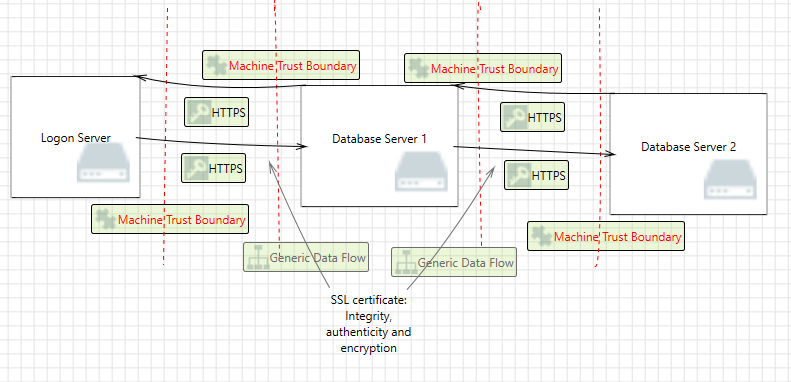


Figure The logon server begins to authenticate the user’s username and password and identifies their privilege level if successfully authenticated.

Upon successful validation, the logon server will generate and issue a two-factor authentication code with is emailed to the user or sent to their mobile device; over a secure connection.

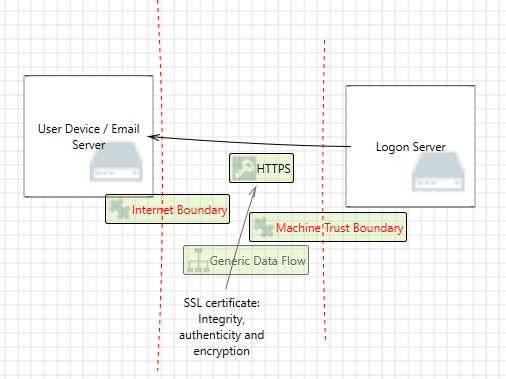


Figure The logon server issuing a two-factor authentication code to the users email address or mobile device phone number.

Once the user is fully authenticated, the logon server will generate an encrypted cookie which will contain the session ID, the users privilege level and all other relevant cookie information which will be stored in database 2 until the session expire. Once the session expires the cookie is deleted.

All this information is naturally displayed via the backend server which is communicating with the logon server at this time in order to display all the relevant log on pages.

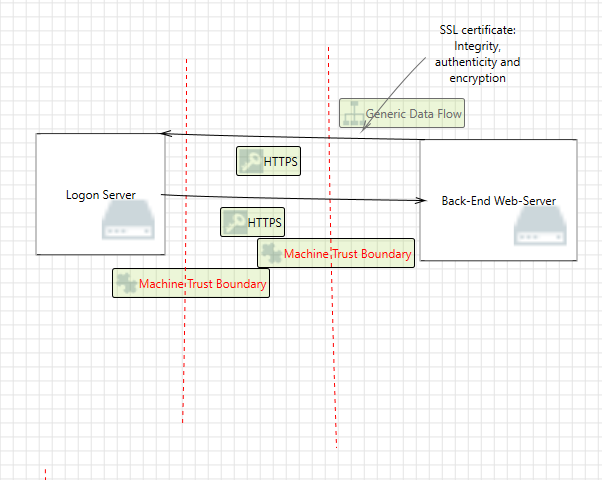


Figure The logon server serves up logon pages to the back-end server which serves the pages up to the user, via secure SSl connections.

Now the user can begin to communicate directly to the backend server which will serve up all future pages and access the database serves for information depending on user requests.

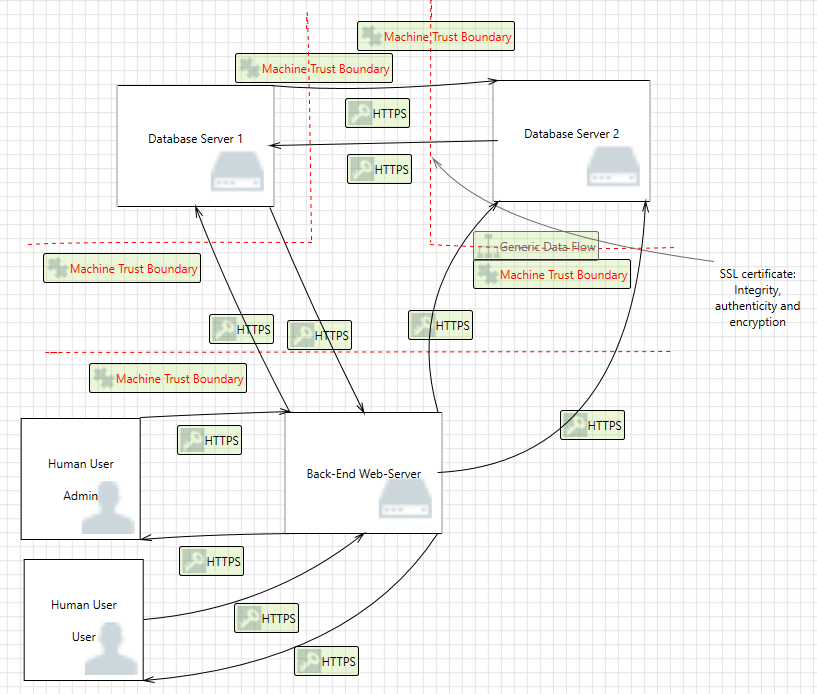


Figure Here we see typical post logon validation interaction between the user and the backend server which serving up pages, after access the database to retrieve information, all over secure SSL channels.

Not only does the backend server to handle the serving of webpages but it also manages the secure messaging facility. The final point of ingress and egress in the application is the communication between the backend server and the users personal external bank. The users bank will of course have its own log in requirements and trust boundaries. And secure encryption protocols should be used to connect to user banks which also serving up connections over secure SSL channels.

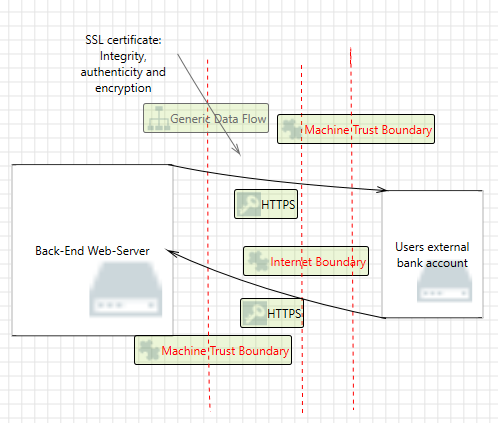


Figure Here we see the back-end server serving up transaction data to the users personal external bank via secure encrypted channels.

Network Topology: The exterior access points of the application will need to provide authentication integrity. This will require encryption. SSL is recommended however no connections lower than TSL 1.2 should be allowed as these encryption protocols are too weak. SSL certificates should be included in the application and maintained. Within the network Digital RSA signatures should be used to establish secure encrypted connections between all servers through direct trunk tunnels. With static IP’s port security and sticky mac addressed applied, as well as access control lists and intrusion detection and intrusion prevention software and hardware included in the topology. Access to external servers such as client emails, or mobile devices for issuance of bank transfers or two-factor authentication codes should also be covered by SSL protocols, RSA digital signatures and private tunnels where possible.

Logon Server: This server should be highly secured in the physical sense. Access to the room containing this server should be restricted to the highest levels. The Logon server will require Two-factor authentication protocols. Data should always be encrypted while passing through the server. Once Authenticated the server will generate an encrypted cookie establishing a local session ID which shall incorporate the user’s details, privilege authorization level. IP address and all other pertinent details.

Database server 1: All database servers should be highly secured in the physical sense. Access to the rooms containing these servers should be restricted to the highest levels. All databases should initially be established as a root user, with administrators being added as user with slight privilege escalations, enough to write to the databases for the purposes of creating and modifying accounts. Administrators should not be root users on the database. The first database should contain all pertinent user information so that users can authenticate and logon. But it should not contain actual user personal data or account data. This adds a further level of authentication to the users account and financial private data.

Database server 2: The second database will be highly secured behind the most stringent of firewalls, intrusion detection and prevention systems. Backups of this server must be regularly maintained in the case of ransom ware attack and to ensure business continuity.

External Banking Servers: Access to external banks should be established through encrypted static or dynamic trunking tunnels with the most stringent security protocols being used and strict port security protocol enabled.

## 2.3 Decompose the Application

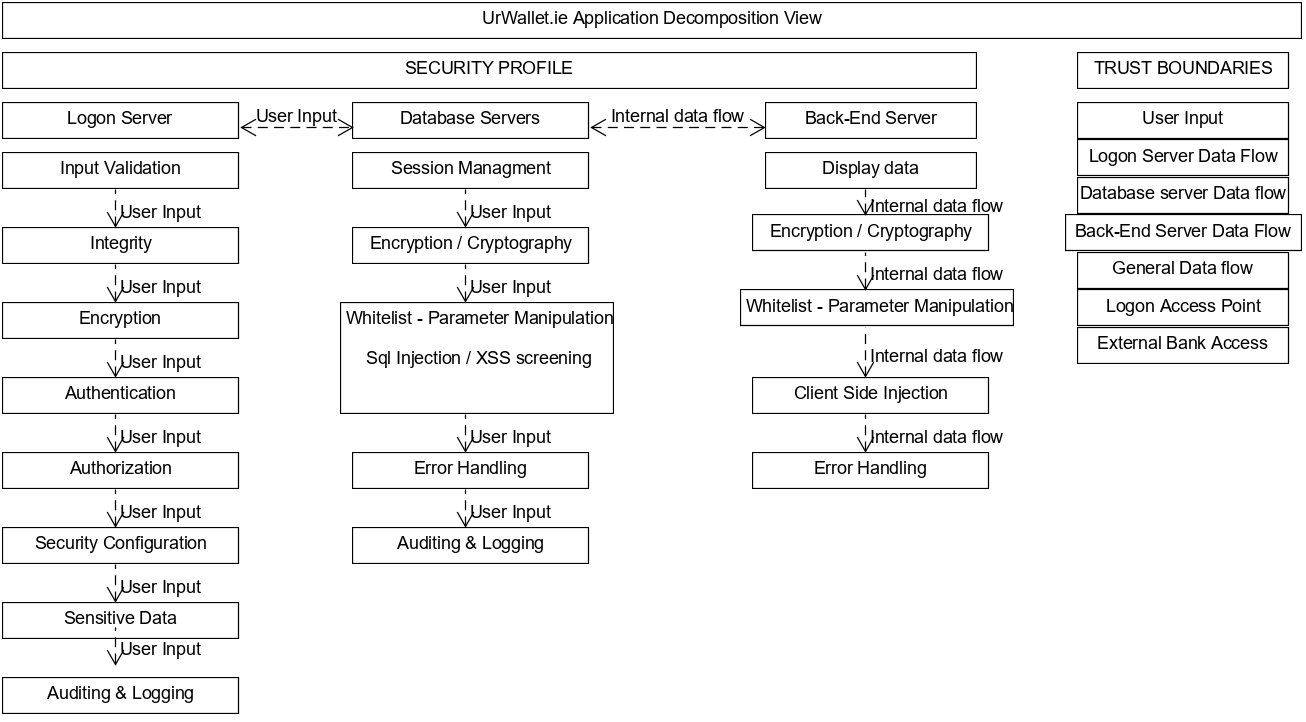


Figure UrWallet.ie decomposed view.

There are 3 parts of our decomposed security profile. Input validation, session management and display data client side. First up we have Input validation. As user input data the data is screened for invalid input and will then logon or be logged on. The logon server will check for a user name and password. Should the user name and password be verified, the logon server will forward a two-factor authentication code to the users selected email address or mobile device via a text message. Once this two-factor authentication is verified against the database server 1, a session id is generated stored in the database server 1 as well as a cookie being created contained the user id, authorization privilege level, and all relevant pertinent information. This cookie and session ID is then encrypted and stored on the database 1 server. The logon attempt, whether successful will be screened to see if it breaches any security configurations and the attempt to log on is audited and logged. User names do not need to be encrypted but all other information should be encrypted where possible with a minimum of sha-2 encryption with a two way hash. Each server should contain its own SSL certificate.

Next input is passed through the logon server, and requests for information are handled by the database server 2. Each request s screened to verify the session ID, cookie and user ID match and requested information is then passed to the back-end server which serves the information up to the user, following a final verification, via the app display or a web browser. All internal data traffic must follow the same standard encryption protocols to ensure data integrity, authorization and authenticity. All data passing through the data base servers must have error handling applied to it, and whitelists, prepared statements, cross site request forgery screening must also be applied. All database server activity should be audited and logged.

The back server will screen for the security previously mentioned vulnerabilities, with one addition. Client-side injection and man in the middle alterations to input, or transmission protocols, destination, source or any other pertinent data that may suggest the transmission has been intercepted. A final audit of all user activity on the back-end server should be maintained.

At all times, a verification check must be performed when data passes through any of the above identified trust boundaries. Both in terms of inbound traffic and outbound for maximum security. It should, be noted that secure code is not noted as a trust boundary, as all the app pertains to financial transactions and data. As such, all code is considered sensitive and is to be protected. The entire application should be closed source with as little code being visible as possible. All scripts should be imported to webpages.

## 2.4 Identify Threats

Threats will be identified in three areas:

1. Network threats.
2. Host Threat.
3. Application Threats.

## 2.4.1 Network Threats

Principal among networks threats is the basic man in the middle attack. This allows any credentials such as user names, password, cookies and other identification and authentication/authorization data being sent in plain text to be captured with ease. For this reason, all traffic must be fully encrypted with no less than sha-2 encryption.

With IP spoofing being so easy to replicate in the modern age, IP addresses should not used as identifiers. Unnecessary ports must be closed, and all software and protocols must be well maintained and updated to the highest and most up to date versions to maintain security. DNS routing tables, Mac address Tables, and the TCP network stack should be hardened and secured.

## 2.4.2 Host Threats

For the purposes of the assignment I have assigned ubuntu server edition as the operating system for all servers. The operating system and all installed software must be kept up to date with regularly scheduled maintenance on the primary servers and any back ups which may be implemented as fail over devices. The will maintain the highest levels of basic defense against 0-day attacks and common CVE’s. Unessential ports should be secured as well common ports of attack such as 21.22,443 etc. Ports which allow connection, for example port 80, must be manage closely with stringent rules against syn attack, flood attacks, spoofing and all known basic networks attacks with extensive firewall rules concerning inbound and outbound traffic, new and established connections and port scans. This should also be applied to all non-essential services and access points. This includes physical access points regarding access to the server and administration console access.

Consideration should be strongly given over to the addition of a hardware firewall place in front of the logon server with a further software failover software such as PFSense, included intrusion detection and prevention mechanics as well. A strong password policy should be implemented with use of no less than 8 characters with 14 being recommended and use of upper case, lowercase, numbers & special being characters being necessary. This is will harden account access against brute force attacks and enumeration attacks.

Access to any admin accounts, configuration access, root accounts both in terms of operating system access and database access, including any downstream or upstream security measures should be strongly controlled with unauthorized access being blocked, prohibited with adequate warnings included to lessen the chances of repudiation activity.

## 2.4.3 Application Threats

Best efforts must be employed through coding to avoid the follow application-based threats:

* Encryption protocols should be no lower than sha-2 with all RSA keys, SSL certificates and other mean of encryption being secured and regularly maintained and updated.
* Poor input validation which may lead to SQL injection attacks, buffer overflow attacks or cross site request forgery attacks or cross site scripting attacks.
* Where possible, all traffic and activity must be logged and audited for the purposes of negating repudiation-based activity and attacks. The is not limited to malicious attacks and includes administrator activity that leads the system vulnerable to attack in an unintended manner.
* Data being sent in plain text due to lack of, or poor implementation of strong encryption protocols. This would lead to credentials being stolen and accounts being accessed by those who should bot be able to via man in the middle attacks. This also includes storing credentials in plain text which should not be stored in plain text such as passwords, cookies, sessionID’s etc.
* Unauthorized physical or logical access to all or any of the servers that make up the applications topology.
* Administration level accounts to access to server’s databases or user accounts where it can be avoided. This means any databases, or program level access points should not be made under admins level privileges such that if an admin account is breached full access is granted to the system. This should be restricted to a super admin or physical access account with the highest levels of security.
* All code should be considered privileged code unless it is absolutely unavoidable to display some code in source code and should that be necessary, and sensitive data is pass through this code, no user input should be leaked in plain text through the management of error handling.
* General error handling must not allow system information or critically sensitive data to be leaked.

## 2.5 Document the threats

The follow list of attack trees with accompanying tables document the various attack vectors and agents that may be deployed against the system:

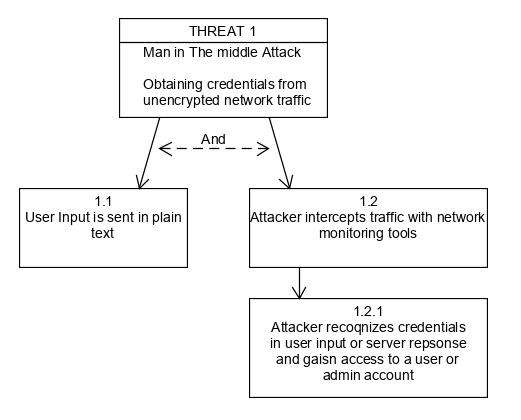


Figure Attack tree 1 Detailing a man in the middle attack.

|  |  |
| --- | --- |
| **PATTERN** | **MAN IN THE MIDDLE ATTACK** |
| Attack Goals | Obtain Credentials from unencrypted network traffic |
| Required Conditions | Data being sent in plain text unsecured with encryption |
| Attack Technique | Network traffic intercepted with network tools |
| Attack Results | Attack gains access to user accounts, admin accounts or gains configuration access |

Table Detailing a man in the middle attack.

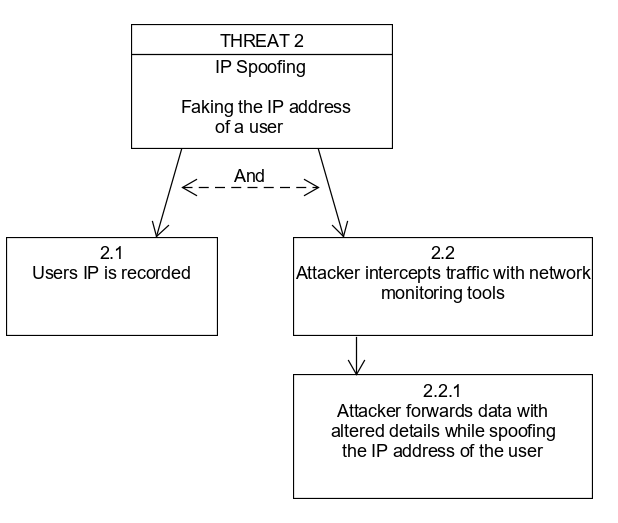


Figure Attack tree 2 detailing an IP spoofing attack.

|  |  |
| --- | --- |
| **PATTERN** | **IP SPOOFING** |
| Attack Goals | To fake the IP address of a user |
| Required Conditions | The attack must somehow know the IP address of the victim machine |
| Attack Technique | Intercept network traffic coming from the user or simply alter IP to pretend to be the user using networking software |
| Attack Results | User can pretend to serve requests from the user’s machine. |

Table detailing an IP spoofing attack.

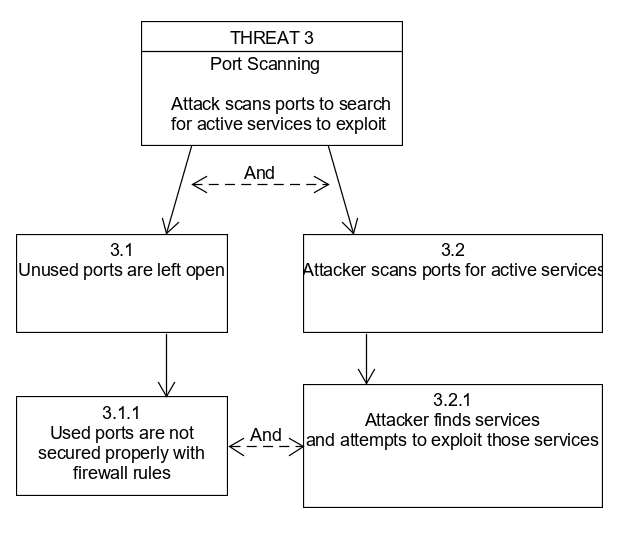


Figure Attack tree 3 detailing a port scan attack.

|  |  |
| --- | --- |
| **PATTERN** | **PORT SCANNING ATTACK** |
| Attack Goals | To detect ports through which the OS or services running on the OS can be directly attacked |
| Required Conditions | Ports left open which are not being used as well as port that are being used lacking adequate firewall protections limiting SYN and other types of scanning methods. |
| Attack Technique | Use of networking tools such as Nmap to detect and details ports and /or services as well as the underlying OS. |
| Attack Results | User can find avenues of attack and specific attack agents to deploy against the machine and / or services. |

Table detailing a port scanning attack.

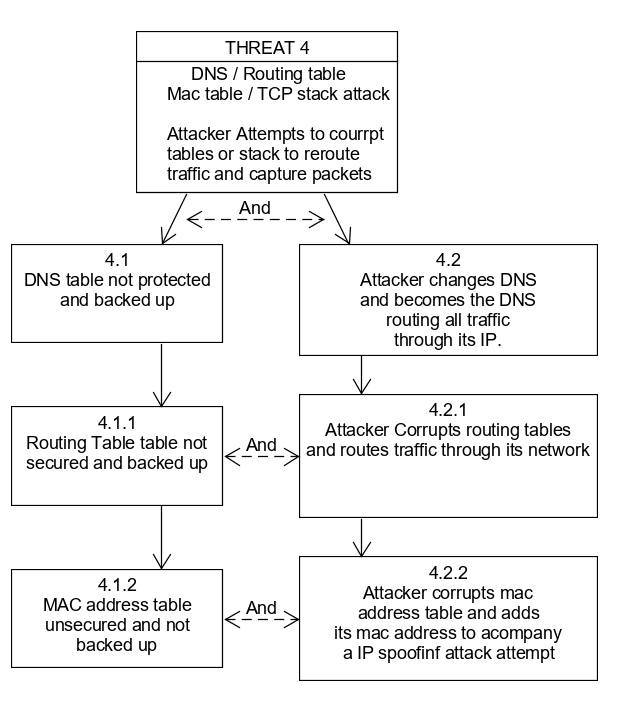


Figure Attack tree 4 detailing a DNS / Routing table / MAC table / TCP stack attack.

|  |  |
| --- | --- |
| **PATTERN** | **DNS / ROUTING / MAC TABLE & TCP STACK ATTACK** |
| Attack Goals | To corrupt DNS routing, MAC table or TCP stack to intercept and re-route traffic |
| Required Conditions | Insecure routing protocols and lack of encryption |
| Attack Technique | Man in the middle interception of traffic, PCAP analysis and forwarding of altered packets, or direct routing access and manipulation |
| Attack Results | Traffic is rerouted through the attacker’s machine |

Table detailing DNS / ROUTING / MAC table & TCP stack attacks.

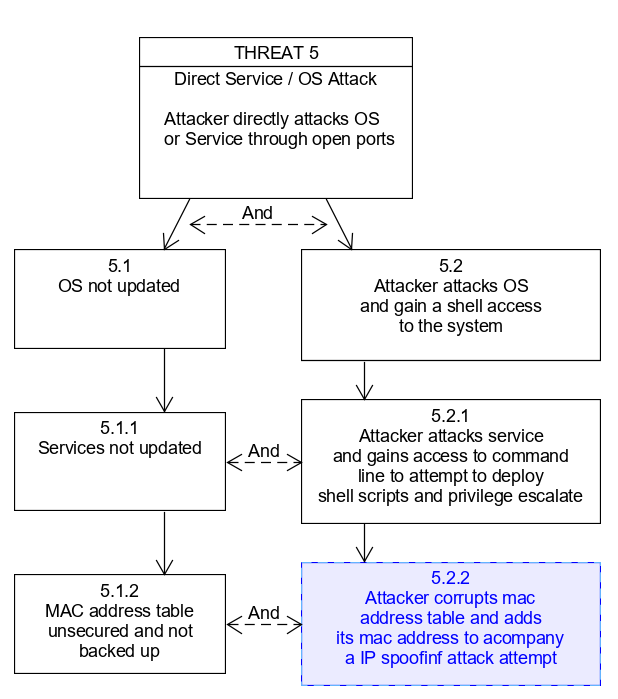


Figure Attack tree 5 detailing a direct service attack.

|  |  |
| --- | --- |
| **PATTERN** | **DIRECT SERVICE / OS ATTACK** |
| Attack Goals | To gain system access to execute shell commands |
| Required Conditions | OS / Services not updated |
| Attack Technique | Direct attack via unsecured ports using deployment tools such as Metasploitable / Armitage. |
| Attack Results | Attacker gains system access to deploy scripts or privilege escalate |

Table detailing direct service /OS attack.

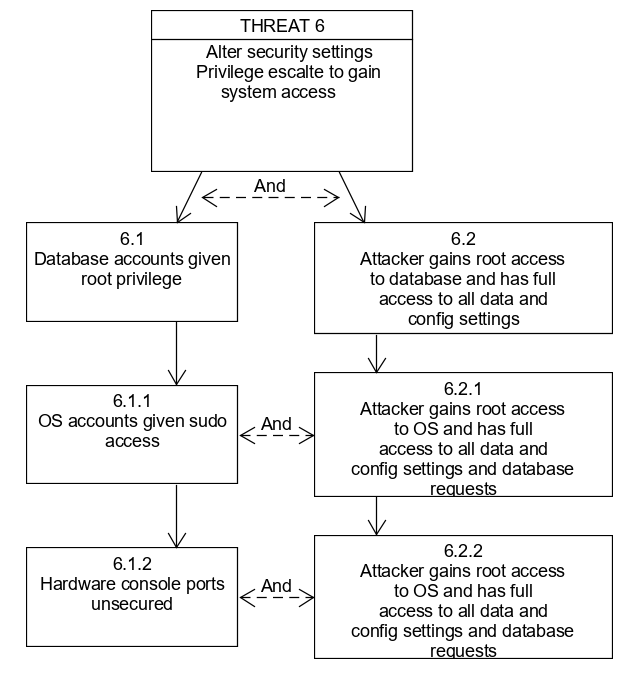


Figure Attack tree 6 detailing insecure account privilege level attack.

|  |  |
| --- | --- |
| **PATTERN** | **INSECURE ACCOUNT ACCESS** |
| Attack Goals | To gain access to insecure accounts which grant root access |
| Required Conditions | Use of administrator level accounts to create profiles, databases or logons. |
| Attack Technique | Main in the middle attack, or Enumeration coupled with brute force or social engineering attack. |
| Attack Results | Attacker gains root or administrator access to OS or database or router. |

Table detailing insecure account accounts privilege attack.

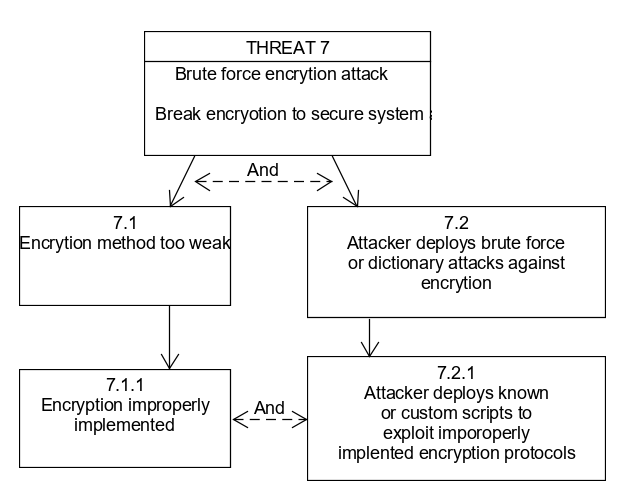


Figure Attack tree 7 detailing a brute force encryption attack.

|  |  |
| --- | --- |
| **PATTERN** | **BRUTE FORE ENCRYPTION ATTACK** |
| Attack Goals | To decrypt encryption and gain access to privileged data |
| Required Conditions | Use of weak encryption or inadequate implementation of encryption code |
| Attack Technique | Brute force or wordlist / dictionary attack directly against encryption using tools such as Hydra, or Medusa. |
| Attack Results | Attacker gains access to privileged sensitive data or system access. |

Table detailing brute force encryption attack.

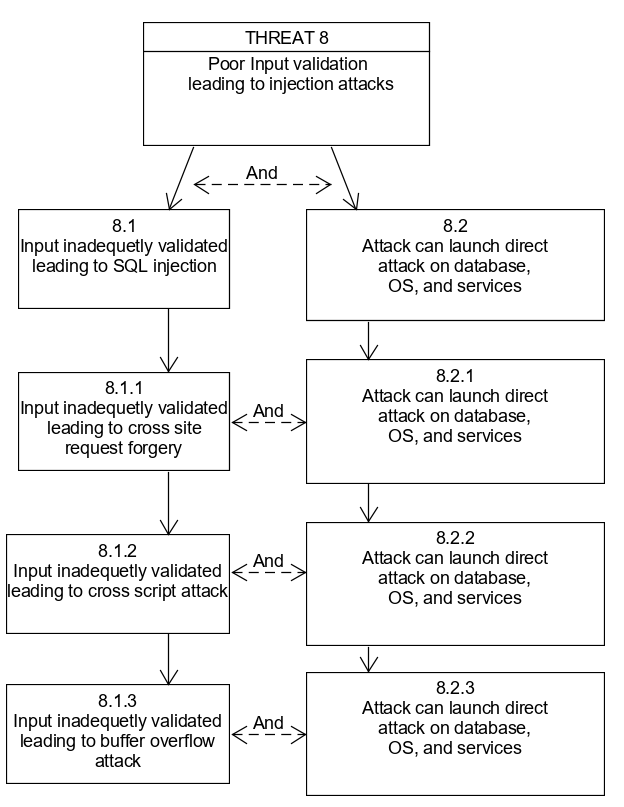


Figure Attack tree 8 detailing poor input validation leading to injection attacks.

|  |  |
| --- | --- |
| **PATTERN** | **INJECTION ATTACKS** |
| Attack Goals | To inject SQL, Cross site request forgery, cross site script or buffer overflow attacks into user input. |
| Required Conditions | Inadequate user input validation |
| Attack Technique | No tools needed. Simple input attacks into available user input methods within the application or client-side code. |
| Attack Results | Attacks are executed server-side and attack gains access to data, system or config settings. |

Table detailing injection attacks.

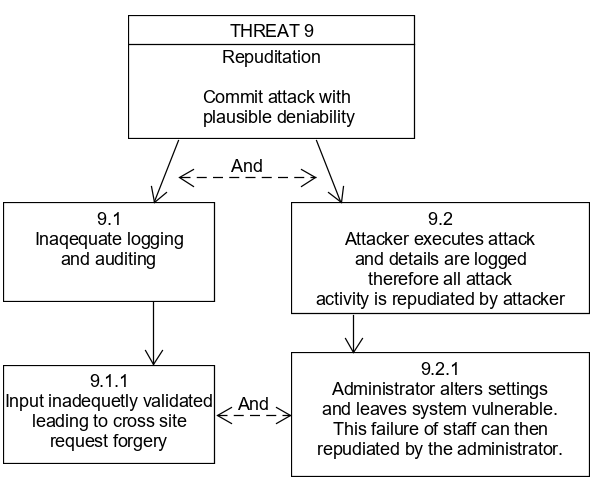


Figure Attack tree 9 detailing a repudiation attack.

|  |  |
| --- | --- |
| **PATTERN** | **REPUDITION ATTACK** |
| Attack Goals | To carry out an attack and retain plausible deniability through obfuscation of identity and location |
| Required Conditions | Lack of adequate logging and / or auditing |
| Attack Technique | No technique needed other than lack of implantation of logging and auditing by administrators |
| Attack Results | Attackers can attack the system with impunity due to their ability to deny their involvement because the attacks where not logged and cannot be audited. Administrator mistakes in configuration which leave the system open to exploit will also not be logged. |

Table detailing repudiation attacks.

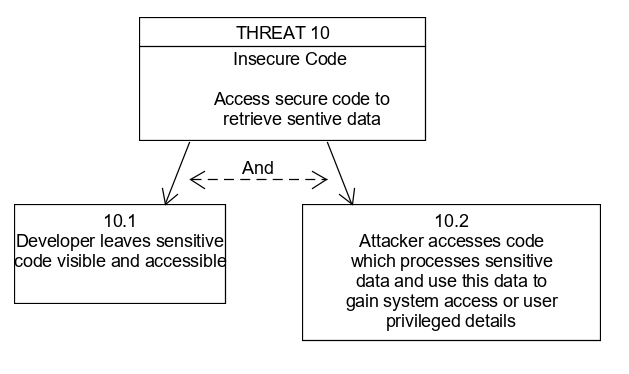


Figure Attack tree 10 detailing an insecure code access attack.

|  |  |
| --- | --- |
| **PATTERN** | **INSECURE CODE ATTACK** |
| Attack Goals | To gain access to privileged data, manipulate code to alter its function, developed a hacked version of the app or otherwise malicious access app functionality. |
| Required Conditions | Code which should be secure and closed source being left insecure |
| Attack Technique | Simply viewing source code that is not secured via a integrated development platform or software which can decompile code |
| Attack Results | Attacker may gain access to sensitive data or bypass logons and gain system access |

Table detailing insecure code attack.

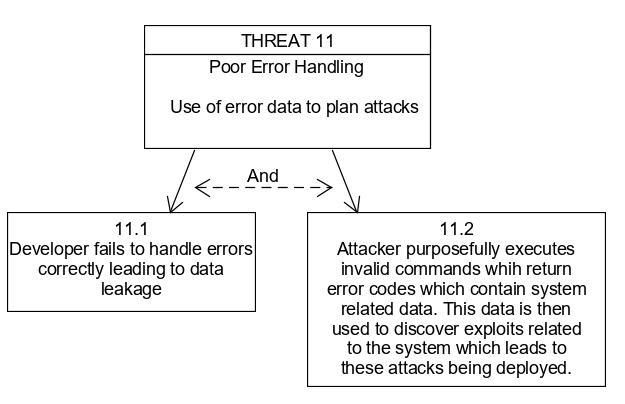


Figure Attack tree 11 detailing poor error handling leading to potential direct service attack.

|  |  |
| --- | --- |
| **PATTERN** | **POOR ERROR HANDLING ATTACK** |
| Attack Goals | To purposefully create errors in the hopes of revealing data leakage regarding system specifications and version numbers |
| Required Conditions | Inadequate implementation of error handling procedures |
| Attack Technique | To simply input invalid data such as negative numbers in calculation fields |
| Attack Results | Data is leaked through default error codes and messages regarding system information, specification, configuration and version numbers. Attackers may use this information to find specific vulnerabilities to against the system, |

Table detailing poor error handling attacks.

## 2.6 Rate the threats

The threats were rated using the ‘D.R.E.A.D.’ system proposed by Microsoft [4] and scored using the ‘D.R.E.A.D.’ scoring table as depicted in appendix 1. Based off this scoring system the ratings shall b as follows:

Scores between 0 – 4 = LOW RISK

Scores between 5 – 9 = MEDIUM RISK

Scores between 10 – 15 = HIGH RISK

|  |  |
| --- | --- |
| **PATTERN** | **MAN IN THE MIDDLE ATTACK** |
| Attack Goals | Obtain Credentials from unencrypted network traffic |
| Required Conditions | Data being sent in plain text unsecured with encryption |
| Attack Technique | Network traffic intercepted with network tools |
| Attack Results | Attack gains access to user accounts, admin accounts or gains configuration access |
| Risk Calculation | D 2 + R 3 + E 3 + A 3 + D 3 = 14 |
| Risk Rating | HIGH RISK |

Table detailing a man in the middle attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **IP SPOOFING** |
| Attack Goals | To fake the IP address of a user |
| Required Conditions | The attack must somehow know the IP address of the victim machine |
| Attack Technique | Intercept network traffic coming from the user or simply alter IP to pretend to be the user using networking software |
| Attack Results | User can pretend to serve requests from the user’s machine. |
| Risk Calculation | D 2 + R 2 + E 3 + A 3 + D 3 = 13 |
| Risk Rating | HIGH RISK |

Table detailing an IP spoofing attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **PORT SCANNING ATTACK** |
| Attack Goals | To detect ports through which the OS or services running on the OS can be directly attacked |
| Required Conditions | Ports left open which are not being used as well as port that are being used lacking adequate firewall protections limiting SYN and other types of scanning methods. |
| Attack Technique | Use of networking tools such as Nmap to detect and details ports and /or services as well as the underlying OS. |
| Attack Results | User can find avenues of attack and specific attack agents to deploy against the machine and / or services. |
| Risk Calculation | D 3 + R 3 + E 2 + A 2 + D 2 = 12 |
| Risk Rating | HIGH RISK |

Table detailing a port scanning attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **DNS / ROUTING / MAC TABLE & TCP STACK ATTACK** |
| Attack Goals | To corrupt DNS routing, MAC table or TCP stack to intercept and re-route traffic |
| Required Conditions | Insecure routing protocols and lack of encryption |
| Attack Technique | Man in the middle interception of traffic, PCAP analysis and forwarding of altered packets, or direct routing access and manipulation |
| Attack Results | Traffic is rerouted through the attacker’s machine |
| Risk Calculation | D 2 + R 2 + E 2 + A 3 + D 3 = 12 |
| Risk Rating | HIGH RISK |

Table detailing DNS / ROUTING / MAC table & TCP stack attacks with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **DIRECT SERVICE / OS ATTACK** |
| Attack Goals | To gain system access to execute shell commands |
| Required Conditions | OS / Services not updated |
| Attack Technique | Direct attack via unsecured ports using deployment tools such as Metasploitable / Armitage. |
| Attack Results | Attacker gains system access to deploy scripts or privilege escalate |
| Risk Calculation | D 3 + R 2 + E 2 + A 3 + D 3 = 13 |
| Risk Rating | HIGH RISK |

Table detailing direct service /OS attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **INSECURE ACCOUNT ACCESS** |
| Attack Goals | To gain access to insecure accounts which grant root access |
| Required Conditions | Use of administrator level accounts to create profiles, databases or logons. |
| Attack Technique | Main in the middle attack, or Enumeration coupled with brute force or social engineering attack. |
| Attack Results | Attacker gains root or administrator access to OS or database or router. |
| Risk Calculation | D 3 + R 1 + E 2 + A 3 + D 3 = 12 |
| Risk Rating | HIGH RISK |

Table detailing insecure account accounts privilege attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **BRUTE FORE ENCRYPTION ATTACK** |
| Attack Goals | To decrypt encryption and gain access to privileged data |
| Required Conditions | Use of weak encryption or inadequate implementation of encryption code |
| Attack Technique | Brute force or wordlist / dictionary attack directly against encryption using tools such as Hydra, or Medusa. |
| Attack Results | Attacker gains access to privileged sensitive data or system access. |
| Risk Calculation | D 3 + R 1 + E 1 + A 3 + D 2 = 10 |
| Risk Rating | HIGH RISK |

Table detailing brute force encryption attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **INJECTION ATTACKS** |
| Attack Goals | To inject SQL, Cross site request forgery, cross site script or buffer overflow attacks into user input. |
| Required Conditions | Inadequate user input validation |
| Attack Technique | No tools needed. Simple input attacks into available user input methods within the application or client-side code. |
| Attack Results | Attacks are executed server-side and attack gains access to data, system or config settings. |
| Risk Calculation | D 3 + R 3 + E 3 + A 3 + D 3 = 15 |
| Risk Rating | HIGH RISK |

Table detailing injection attacks with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **REPUDITION ATTACK** |
| Attack Goals | To carry out an attack and retain plausible deniability through obfuscation of identity and location |
| Required Conditions | Lack of adequate logging and / or auditing |
| Attack Technique | No technique needed other than lack of implantation of logging and auditing by administrators |
| Attack Results | Attackers can attack the system with impunity due to their ability to deny their involvement because the attacks where not logged and cannot be audited. Administrator mistakes in configuration which leave the system open to exploit will also not be logged. |
| Risk Calculation | D 1 + R 3 + E 3 + A 3 + D 3 = 13 |
| Risk Rating | HIGH RISK |

Table detailing repudiation attacks with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **INSECURE CODE ATTACK** |
| Attack Goals | To gain access to privileged data, manipulate code to alter its function, developed a hacked version of the app or otherwise malicious access app functionality. |
| Required Conditions | Code which should be secure and closed source being left insecure |
| Attack Technique | Simply viewing source code that is not secured via a integrated development platform or software which can decompile code |
| Attack Results | Attacker may gain access to sensitive data or bypass logons and gain system access |
| Risk Calculation | D 3 + R 3 + E 2 + A 3 + D 3 = 14 |
| Risk Rating | HIGH RISK |

Table detailing insecure code attack with threat rating added.

|  |  |
| --- | --- |
| **PATTERN** | **POOR ERROR HANDLING ATTACK** |
| Attack Goals | To purposefully create errors in the hopes of revealing data leakage regarding system specifications and version numbers |
| Required Conditions | Inadequate implementation of error handling procedures |
| Attack Technique | To simply input invalid data such as negative numbers in calculation fields |
| Attack Results | Data is leaked through default error codes and messages regarding system information, specification, configuration and version numbers. Attackers may use this information to find specific vulnerabilities to against the system, |
| Risk Calculation | D 1 + R 3 + E 3 + A 3 + D 3 = 13 |
| Risk Rating | HIGH RISK |

Table detailing poor error handling attacks with threat rating added.

# Conclusions

The threat model creation process is an ever evolving documentational process. As such, this report should not be seen as a end point, but a beginning point from which the development team can begin to design around all of the major risks identified during the process of developing the above threat model. It is highly anticipated for example that as each major risk is explored and mitigations begin to be designed, new medium and low risk vulnerabilities and points of weakness will be identified. This document should be updated to include these new aspects of the threat model and risk assessment. The document is not intended to be used by those without programming experience and as such is considered a technical manual to be used by the development team and use case testing team followed perhaps be penetration testers who can test if proper mitigations have been put in place as well as testing if any holes have been left through which known attacks can be deployed. At this point in the threat model, the overview is considered to high level and as each stage of the design process is completed, time should be given over to developing deeper threat model to replace or update this document. The document should be considered in its entirety and not just the risk assessment. It is important to understand the basic processes as some of the risk identified can be mitigated in part of the program and still be present in another, and as such, care must be given over to identify all points of vulnerability and access with regard to each attack method and each access point or vulnerability must be mitigated.

# 4.0 Glossary

Two Factor Authentication: This is a double authentication method whereby a user must first provide a user name & password to the login server, and the server issues a code, via an email address or phone number which must then be entered alongside the user name & password.

Business Continuity: The practice of implementing protocols and procedures that allow a business to recover and continue to conduct business in the event of a attack or hardware/software failure which brings the network down and stops the business model from functioning.

TLS: Transport Layer Security protocol which provides encryption to internet traffic. This is an outdated method though newer versions of the protocol do exist and are still in use. No lower than version 1.2 should be used.

SSL: Secure Socket Layer protocol. This is the successor protocol to TLS and is more secure than TLS. As such SSL certificates are recommended for maximum security.

0-day attack: a new unknown attack which has only just been discovered so defense may be weak against it.

CVE: Common vulnerability exposure. CVE’s document well known established attacks and any recommended counter measures.

PFSense: This a third-party open source firewall which runs on both the Linux and windows platform which provides a web-based interface for admins to adjust the configuration and is generally placed down or upstream of intended machine to be protected with traffic being denied or forwarded through the firewall.

XXS: Cross site scripting attack. This is where a using inputs a script such as java script into an input field, user input validation fails to screen against it and the code is executed by the program.

SYN: Transmission control protocol commonly used to initiate attacks such as D-Dos attacks and various other types.

OS: Operating system.

DNS: Domain name server.

PCAP: Packet capture file.

Metasploitable: A program used to deploy and execute attacks. Mainly cli based.

Armitage: A version of Metasploitable which comes equipped with a graphical user interface.

Hydra / Medusa: variants of programs used to a lunch brute force and wordlist or dictionary attacks.

# 5.0 Appendix 1:

Thread Rating Table as published: Microsoft, **‘Table 3.6 Thread Rating Table’, [ONLINE] Available at:** <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v=pandp.10)?redirectedfrom=MSDN#risk--probability--damage-potential> **[Accessed Oct. 17, 2019].**

|  | **Rating** | **High (3)** | **Medium (2)** | **Low (1)** |
| --- | --- | --- | --- | --- |
| D | Damage potential | The attacker can subvert the security system; get full trust authorization; run as administrator; upload content. | Leaking sensitive information | Leaking trivial information |
| R | Reproducibility | The attack can be reproduced every time and does not require a timing window. | The attack can be reproduced, but only with a timing window and a particular race situation. | The attack is very difficult to reproduce, even with knowledge of the security hole. |
| E | Exploitability | A novice programmer could make the attack in a short time. | A skilled programmer could make the attack, then repeat the steps. | The attack requires an extremely skilled person and in-depth knowledge every time to exploit. |
| A | Affected users | All users, default configuration, key customers | Some users, non-default configuration | Very small percentage of users, obscure feature; affects anonymous users |
| D | Discoverability | Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable. | The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use. | The bug is obscure, and it is unlikely that users will work out damage potential |

# 6.0 References

[1] Microsoft, ‘Microsoft Threat Modeling Tool 2016’, [ONLINE] Available at: <http://www.microsoft.com/en-ie/download/details.aspx?id=49168> [Accessed Oct, 12. 2019].

[2] UNLet, ‘UMLet 14.3 Free UML Tool for Fast UML Design’. [ONLINE] Available at: <https://www.umlet.com/> [Accessed Oct. 8, 2019].

[3] J.D. Meier, (et al) June 2003, ‘Chapter 3 – Threat modelling’, [ONLINE] Available at: <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v=pandp.10)?redirectedfrom=MSDN> [Accessed Oct. 12, 2019].

[4] Microsoft, **‘Table 3.6 Thread Rating Table’, [ONLINE] Available at:** <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v=pandp.10)?redirectedfrom=MSDN#risk--probability--damage-potential> **[Accessed Oct. 17, 2019].**