|  |  |
| --- | --- |
|  | CW03 Threat Modelling |
|  |  |
|  | James Cassidy 40267110  Secure Software Development  4/29/22 |

Table of Contents

[**Executive Summary** 3](#_Toc102336008)

[**Risks Identified** 3](#_Toc102336009)

[**Recommendation** 3](#_Toc102336010)

[**Approach Taken** 4](#_Toc102336011)

[**Business Architecture** 5](#_Toc102336012)

[**Architecture Decomposition** 6](#_Toc102336013)

[**Threat Identification** 7](#_Toc102336014)

[**Threat Rating/Risk Assessment** 10](#_Toc102336015)

[**STRIDE** 10](#_Toc102336016)

[**DREAD** 10](#_Toc102336017)

[**Traceability/Compliance Matrix** 11](#_Toc102336018)

[**Conclusion** 13](#_Toc102336019)

[References 13](#_Toc102336020)

# **Executive Summary**

In developing a threat model for the business application, I can adopt the perspective of a threat actor and see what damage they can do to the application.

Through extensive threat modelling of the Business’ Online Banking Application, multiple threats and risks have been discovered. Various security controls are already in place to prevent some of these threats. However, more mitigative measures and techniques will need to be implemented in order to prevent these risks from escalating further.

## **Risks Identified**

A number of risks were identified, including:

* **SQL Injection**
* **Session Replay**
* **Brute Force Attack**
* **Log Injection (Cross Site Scripting)**
* **ARP Spoofing**
* **Clickjacking**
* **Phishing**
* **DDoS Attack**

Further details regarding these risks can be found under, ‘Threat Identification’.

## **Recommendation**

As stated, various frameworks and security controls have already been implemented to counteract these risks. However, failure to develop other mitigation techniques will result in the compromise of business confidentiality, integrity, and availability and authenticity as customer data is stolen and tampered by threat actors.

# **Approach Taken**

Once a threat model had been established and various risks had been identified, both ‘STRIDE’ [1] and ‘DREAD’[2]methodologies were utilised.

After all vulnerabilities have been identified, STRIDE methodology will be used to classify the vulnerabilities discovered.

The categories included in STRIDE are:

* **Spoofing**
* **Tampering**
* **Repudiation**
* **Information Disclosure**
* **Denial of Service**
* **Elevation of Privilege**

Once classified, DREAD methodology will then be utilised to prioritize, rate, and compare the severeness of each risk that was classified using STRIDE.

The categories included in DREAD are:

* **Damage**
* **Reproducibility**
* **Exploitability**
* **Affected Users**
* **Discoverability**

In tackling this threat model, ‘OWASP Top Ten’ [1] was observed to monitor the most critical security risks that could occur within the online banking application.

# **Business Architecture**

This section should answer questions such as what does the business do? How does the business do it? What technologies are necessary for the business? What data flows in and out of the system? A diagram showcasing these in context is very helpful.

The online banking app developed by the business allows a user to login to their bank account through either their web browser, iOS, or Android phone. Two-factor Authentication is in place. An email can also be sent to the user should they require further assistance with their account.

Once logged in, the user is able to make various transactions such as withdrawal, transfer, deposit, loan payments and wire transfer.

Diagram, engineering drawing

Description automatically generatedThe diagram below shows these various dataflows in context:

# **Architecture Decomposition**

To understand how the system works, it can now be decomposed into its smaller subsystems and components. Here potential threat actors, assets and security controls have been established.

Several trust boundaries have been identified including a Demilitarized Zone (DMZ) for external-facing web components between the internet and Web Server, An internal boundary for the application server and Restricted Network for the datastores of the banking application.

Within the ASP.NET Server, Data Protection API (DPAPI) has been utilised to encrypt data using information from the current user. A Load Balancer has also been implemented to distribute web traffic should a DDoS Attack occur.

Trusted Network Boundary houses the security control Risked Based Approach (RBA) Fraud Detection, essentially meaning no fraudulent transaction can be passed if a threat actor gains control of a genuine user’s credit details.

Within the Restricted Network boundary exists Authentication Credential Store that houses user credentials as well as an Audit Log. Keeping a log allows almost any attack, should it occur, be traced back to the original threat actor. Within the Oracle Database, cryptographic storage controls have been utilised, including SHA-256 password hashing and 128-bit AES encryption.

Diagram

Description automatically generatedThe diagram below shows how all this has been implemented:

# **Threat Identification**

|  |  |
| --- | --- |
| **Threat ID 1** | SQL Injection bypass Authentication Login |
| **Threat Agent** | External |
| **Threat Description** | Threat actor crafts malicious SQL statement (ALWAYS TRUE) to gain access to other user accounts on database |
| **Threat Target** | User Credentials |
| **Attack Surface** | Login |
| **Impact** | Threat actor gains access to user credentials to access unauthorised accounts and make fraudulent transactions. |
| **Mitigation** | Use of Prepared Statements.  Stored Procedures.  Allow-List Input Validation.  Escaping all User Input. |

|  |  |
| --- | --- |
| **Threat ID 2** | Theft of Authentication Cookies through Session Replay |
| **Threat Agent** | External |
| **Threat Description** | Man-in-the-middle (MITM) eavesdrops on communication between genuine user and web server. MITM intercepts this and uses genuine user’s login credentials to gain access to their account. |
| **Threat Target** | User Credentials |
| **Attack Surface** | User Network |
| **Impact** | Threat actor gains access to user credentials to access unauthorised accounts and make fraudulent transactions |
| **Mitigation** | Lifespan of session cookies should be set for as short as possible.  Set web session as invalid as soon as genuine user logs out. |

|  |  |
| --- | --- |
| **Threat ID 3** | Brute Force Attack Password Through Login |
| **Threat Agent** | External |
| **Threat Description** | Threat actor attempts to guess user password through brute force by systematically checking trying every possible combination. |
| **Threat Target** | User Credentials |
| **Attack Surface** | Login |
| **Impact** | Threat actor will gain access to user credentials, allowing them unauthorised access to their bank account. |
| **Mitigation** | Establish Web Application Firewalls (WAF).  Account locked out after failed attempt. |

|  |  |
| --- | --- |
| **Threat ID 4** | Log Injection leading to Stored XSS Attack. |
| **Threat Agent** | External |
| **Threat Description** | Threat actor parses malicious script that gets written to log file. This compromises the web page of the bank and can compromise many other users that visit the website after. |
| **Threat Target** | User Credentials |
| **Attack Surface** | ASP.NET |
| **Impact** | Threat actor can gain access to several user credentials, allowing them unauthorised access to their bank account and make wire transfers. |
| **Mitigation** | Validate User input before logging takes place.  Establish Output Encoding.  Block known attack strings in a block list. |

|  |  |
| --- | --- |
| **Threat ID 5** | ARP Spoofing through User Network |
| **Threat Agent** | External |
| **Threat Description** | Threat actor can utilise ARP spoofing to hijack genuine user session by sending ARP packets to web server. Actor gains unauthorised access to user account |
| **Threat Target** | User Credentials |
| **Attack Surface** | User Network |
| **Impact** | Threat actor will gain access to user credentials, allowing them unauthorised access to their bank account and make ACH transfers. |
| **Mitigation** | Audit Log will be able to pinpoint the MAC Address of threat actor  Communication between client and server should take place on secure communication protocols such as TLS, IPSec and SSH.  Establish Access Control List. |

|  |  |
| --- | --- |
| **Threat ID 6** | Clickjacking through Login |
| **Threat Agent** | External |
| **Threat Description** | Threat actor successfully alters web page to display frame over login. Genuine user enters their account details into this, giving the threat actor access. |
| **Threat Target** | User Credentials |
| **Attack Surface** | Login |
| **Impact** | Threat actor will gain access to user credentials, allowing them unauthorised access to their bank account and make wire transfers. |
| **Mitigation** | Establish Content Secure Policy (CSP) alongside its frame-ancestor directive. Allows application developer to disable all frame use. |

|  |  |
| --- | --- |
| **Threat ID 7** | Phishing through Email SMTP |
| **Threat Agent** | External |
| **Threat Description** | Threat actor tricks user into handing over sensitive data such as their account details and credit card info. |
| **Threat Target** | User Credentials, Credit Card Info |
| **Attack Surface** | Email |
| **Impact** | Threat actor will gain access to user credentials, allowing them unauthorised access to their bank account and make wire transfers. |
| **Mitigation** | Establish an SSL Certificate to secure all traffic between the client and the web server. |

|  |  |
| --- | --- |
| **Threat ID 8** | DDoS Attack through SYN Flood Attack |
| **Threat Agent** | External |
| **Threat Description** | Threat actor sends SYN packets to the IIS Server. This overwhelms all available ports on the network, resulting in the server unable to respond to legitimate traffic. |
| **Threat Target** | IIS Server |
| **Attack Surface** | Internet |
| **Impact** | Threat actor successfully brings down the bank’s website, resulting in no availability to legitimate users |
| **Mitigation** | Load Balancer in place to reroute live traffic from one server to another if the server falls prey to a DDoS attack. |

|  |  |
| --- | --- |
| **Threat ID 9** | SQL Injection into Oracle DB |
| **Threat Agent** | Internal Database Admin |
| **Threat Description** | Threat actor (DB Admin) has root access to restricted network of the business and is able to perform a SQL injection to gain access to audit logs, payment info and customer credentials. |
| **Threat Target** | Authentication Credential Store, Audit Logs |
| **Attack Surface** | Restricted Network |
| **Impact** | Threat actor will gain access to user credentials, allowing them unauthorised access to their bank account and make unauthorised ACH transfers. |
| **Mitigation** | Implement Principle of Least Privilege with employees of the business.  Establish a filesystem Access Control List within restricted network. |

# **Threat Rating/Risk Assessment**

## **STRIDE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Threats | **S**poofing Identity | **T**ampering | **R**epudiation | **I**nformation Disclosure | **D**enial of Service | **E**levation of Privilege |
| TID-1 | ✓ | ✓ | ✓ | ✓ |  | ✓ |
| TID-2 | ✓ |  | ✓ | ✓ |  | ✓ |
| TID-3 | ✓ |  | ✓ | ✓ |  | ✓ |
| TID-4 | ✓ |  | ✓ | ✓ |  | ✓ |
| TID-5 | ✓ |  | ✓ | ✓ |  | ✓ |
| TID-6 | ✓ |  | ✓ | ✓ |  | ✓ |
| TID-7 | ✓ |  | ✓ | ✓ |  | ✓ |
| TID-8 |  |  | ✓ |  | ✓ |  |
| TID-9 | ✓ | ✓ | ✓ | ✓ |  | ✓ |

## **DREAD**

Using DREAD Risk Assessment model, a range of 1-3 was used.

A value of 3 for each signifies maximum damage, easy to reproduce, easy to exploit, affects the maximum amount of users and hard to discover for their respective definitions.

Each Threat ID is then given a rating of Low, Medium, High, or Very High depending on the score that has been assigned to it.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Threat | D | R | E | A | D | Total | Rating |
| TID-1 | 3 | 3 | 3 | 3 | 3 | 15 | Very High |
| TID-2 | 2 | 1 | 2 | 3 | 3 | 10 | Medium |
| TID-3 | 1 | 1 | 1 | 0 | 3 | 6 | Low |
| TID-4 | 3 | 1 | 3 | 3 | 3 | 13 | High |
| TID-5 | 1 | 3 | 3 | 1 | 1 | 9 | Medium |
| TID-6 | 3 | 1 | 3 | 3 | 1 | 11 | Medium |
| TID-7 | 1 | 1 | 3 | 3 | 1 | 9 | Medium |
| TID-8 | 3 | 2 | 3 | 3 | 1 | 12 | High |
| TID-9 | 3 | 3 | 3 | 3 | 3 | 15 | Very High |

# **Traceability/Compliance Matrix**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Threat Agent** | **Asset** | **Attack** | **Attack Surface** | **Attack Goal** | **Impact** | **Control** | **Mitigation** |
| TA01 | A01 | SQL Injection | Login | User Credentials | Very High: Access to personal user and system details | No whitelisting or prepared statements are in place to stop this | Option 1: Use of Prepared Statements  Option 2: Use of Stored Procedures  Option 3: Allow-list Input Validation  Option 4: Escaping All User Supplied Input |
| TA02 | A01 | Session Replay | User Network | User Credentials | Medium: Access to user account details and perform fraudulent transactions | No lifespan of session tickets set. | Option 1: Set lifespan of session for as short as possible  Option 2: Set web application session as invalid after timeout and user logs out. |
| TA01 | A01 | Brute Force Attack | Login | User Credentials | Low: Malicious user must enter user details until correct combination is found | Two Factor Authentication is in place alongside limited failed login attempts. | None Needed |
| TA01 | A06 | Log Injection  XSS Attack | ASP.NET | Application Log File | High: Log File contains malicious script that allows cross site scripting to occur. | No output encoding or validation of user input implemented. | Option 1: Validate user input before logging.  Option 2: Block List validation  Option 3: Output Encoding |
| TA02 | A01 | ARP Spoofing | User’s Network | User’s Credentials | Medium: MITM sends ARP packets to IIS and able to hijack user session. | RBA Fraud Detection in place to prevent any fraudulent transactions. | Option 1: Use protocols such as TLS, IPSec and SSH to authenticate and encrypt data in transit.  Option 2: Implement Access Control List |
| TA02 | A01 | Clickjacking | Login | User Credentials | Medium: Renders fake login box over real one | No security controls in place to prevent this | Option 1: CSP not to allow framing from other domains |
| TA03 | A01, A02 | Phishing | Email (SMTP protocol) | User Credentials, Credit Information | Medium: Access to all user’s information and credit card details. | No security controls in place to stop this | Option 1: SSL Certificate used to protect information sent between web server and users’ browsers and mobile devices. |
| TA01 | A05 | DDoS Attack | Web Browser, iOS, Android | IIS Server | High: Cause the bank webpage to crash meaning users can’t make transactions | Load Balancer in place in order to prevent overloading a particular resource of the network. | None needed |
| TA04 | A01,  A02 | SQL Injection | Restricted Network | Authentication Credential Store | High: Attacker usually internal database admin, has access to all user information, including credit card info | AES Encryption with key of 128 bits in place.SHA-256 Password Hashing also implemented | Option 1: Implement Principle of Least Privilege.  Option 2: Access Control List |

# **Conclusion**

With all threats assessed, there are several areas of concern that need addressed immediately.

Several incidents of SQL injection exist within this application. Assessing these risks through DREAD revealed that they have a **very high** rating. Failure to add stored procedures, prepared statements and escaping user input will result in a threat actor escalating privileges within the entire application.

The business must also ensure the principle of least privilege is used with their employees. A database admin that already has access to the restricted network may be exploited by a potential threat actor.

With all current security controls in place, the business can be assured that customer and employee data remain confidential, integral, and available at all times.

# References

|  |  |
| --- | --- |
| [1] | A. Charles, “What is STRIDE Methodology in Threat Modelling?,” 24 May 2021. [Online]. Available: https://www.koenig-solutions.com/blog/stride-methodology-in-threat-modelling#:~:text=What%20is%20the%20STRIDE%20Methodology,Authentication%2C%20and%20Non%2DRepudiation.. |
| [2] | “OWASP Top Ten,” 2021. [Online]. Available: https://owasp.org/www-project-top-ten/. |
| [3] | D. Czagan, “Qualitative risk analysis with the DREAD model,” 21 May 2014. [Online]. Available: https://resources.infosecinstitute.com/topic/qualitative-risk-analysis-dread-model/. |