Threat Modeling Report

Created on 01/12/2025 14:23:12

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

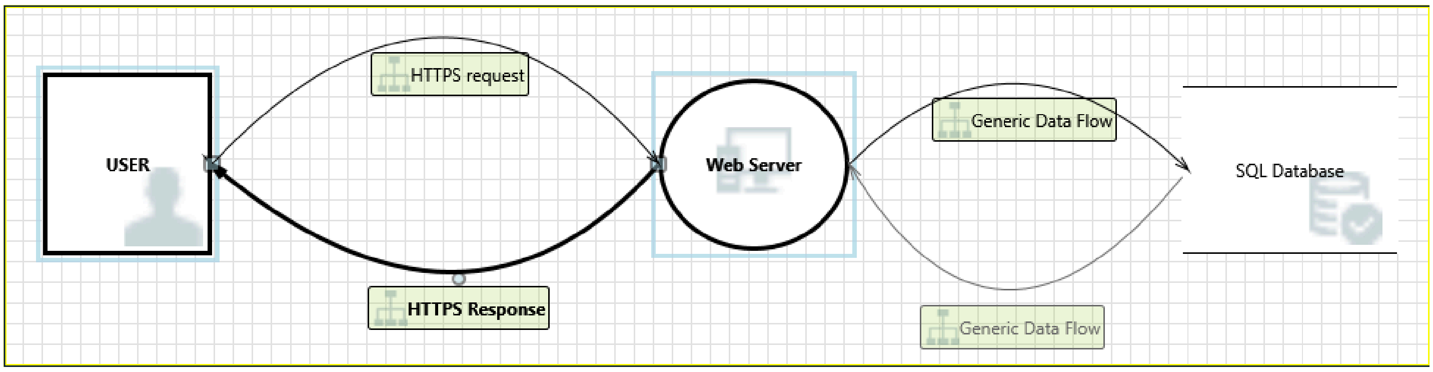
Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	8
Total	8
Total Migrated	0

Diagram: Diagram 1



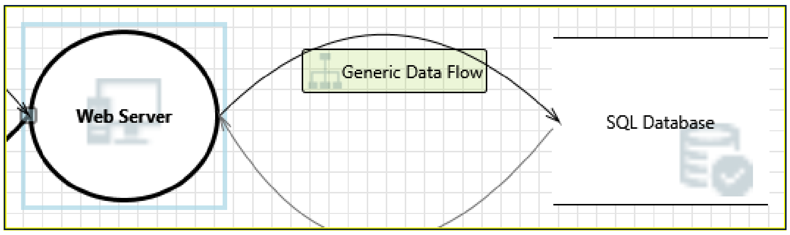
Validation Messages:

1. **Error:** The connector should be attached to two elements.

Diagram 1 Diagram Summary:

Not Started	0
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	8
Total	8
Total Migrated	0

Interaction: Generic Data Flow



1. Spoofing of Destination Data Store SQL Database [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	SQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of SQL Database. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	A Standard authentication mechanism was been used.

2. Potential SQL Injection Vulnerability for SQL Database [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.
Justification:	Dynamic SQL construction can unintentionally allow attackers to alter queries, and even parameterized inputs can be abused if implemented incorrectly, making thorough review essential.

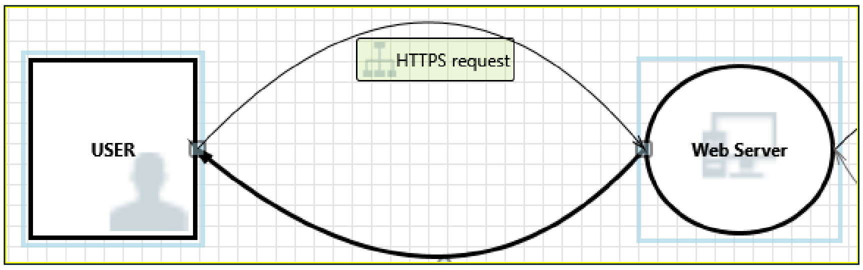
3. Weak Credential Storage [State: Mitigation Implemented] [Priority: High]

Category:	Information Disclosure
Description:	Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored
Justification:	Credentials are common targets for attackers, whether stored on the server or client. Using salted hashes or strong encryption reduces the impact of disclosure or tampering, and protecting client-side storage prevents attackers from easily stealing or modifying stored credentials.

4. Potential Excessive Resource Consumption for Web Server or SQL Database [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does Web Server or SQL Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification:	Resource consumption must be controlled because attackers can overwhelm servers or databases, causing slowdowns or outages. Explicit limits, timeouts, and safe resource handling help prevent deadlocks and ensure the system remains resilient under load.

Interaction: HTTPS request



5. Spoofing the USER External Entity [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	USER may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.

Justification:	User identities can be spoofed, allowing attackers to gain unauthorized access. Strong, standardized authentication helps verify legitimate users and prevents impersonation attacks.

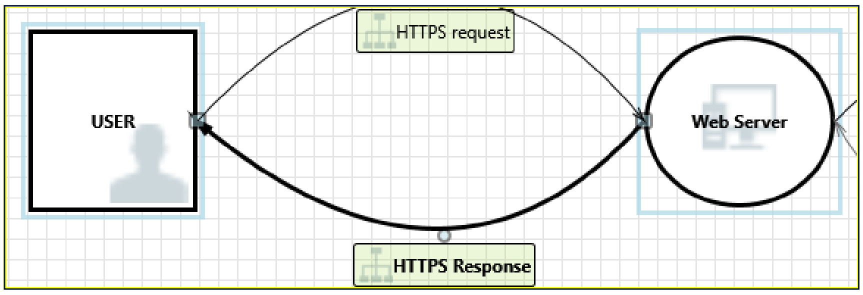
6. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	Without sanitizing untrusted input, attackers can inject malicious scripts that run in users' browsers, making the web server vulnerable to cross-site scripting attacks.

7. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	Web Server may be able to impersonate the context of USER in order to gain additional privilege.
Justification:	If the web server can impersonate the user's identity, it may gain unintended privileges, making strict separation of user and server identity essential to prevent unauthorized access.

Interaction: HTTPS Response



8. Authenticated Data Flow Compromised [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	An attacker can read or modify data transmitted over an authenticated dataflow.
Justification:	Authenticated data can still be intercepted or altered if it is not encrypted or integrity-protected, allowing attackers to read or modify sensitive information in transit.