Threat Modeling Report

Created on 11/21/2025 11:44:56 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

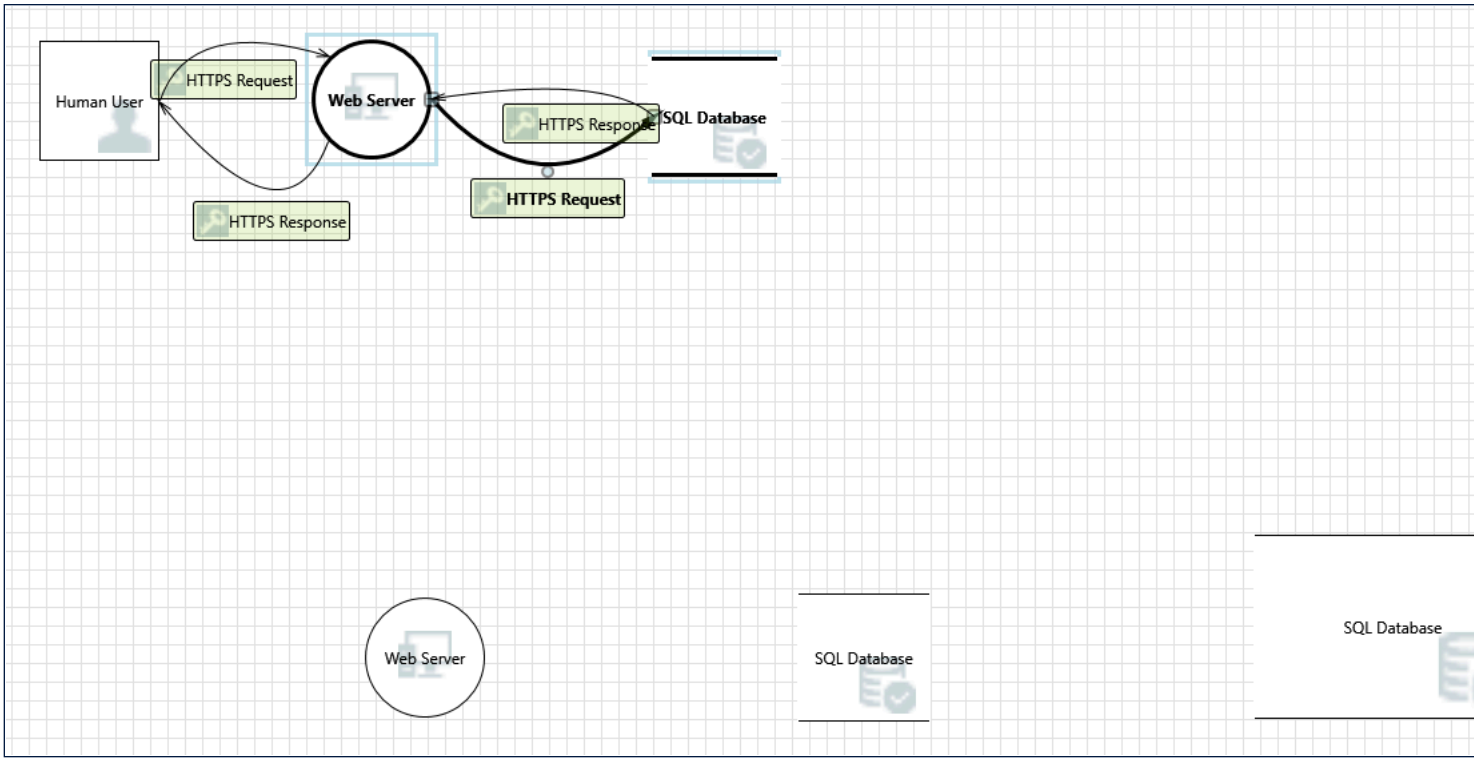
Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	17
Total	17
Total Migrated	0

Diagram: SQL Database

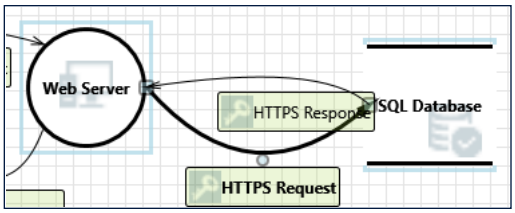


SQL Database Diagram Summary:

Not Started	0
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	17

Total	17
Total Migrated	0

Interaction: HTTPS Request



1. 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: <no mitigation provided>

2. Risks from Logging [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources.
Justification: <no mitigation provided>

3. 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: <no mitigation provided>

4. Lower Trusted Subject Updates Logs [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log.
Justification: <no mitigation provided>

5. Data Logs from an Unknown Source [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them.
Justification: <no mitigation provided>

6. Insufficient Auditing [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle a repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data.
Justification: <no mitigation provided>

7. Potential Weak Protections for Audit Data [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect

Justification: <no mitigation provided>

8. Authorization Bypass [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Can you access SQL Database and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface.
Justification: <no mitigation provided>

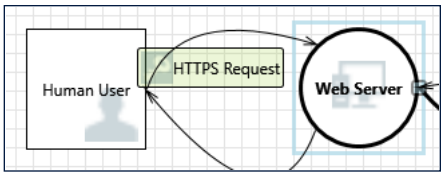
9. Weak Credential Storage [State: Mitigation Implemented] [Priority: High]

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: <no mitigation provided>

10. 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: <no mitigation provided>

Interaction: HTTPS Request



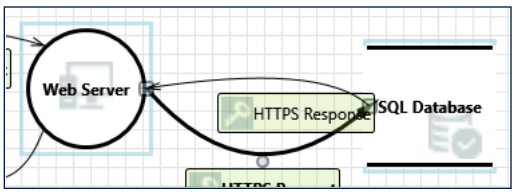
11. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Web Server may be able to impersonate the context of Human User in order to gain additional privilege.
Justification: <no mitigation provided>

12. 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: Mitigated

Interaction: HTTPS Response



13. Spoofing of Source 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 incorrect data delivered to Web Server. Consider using a standard authentication mechanism to identify the source data store.
Justification: <no mitigation provided>

14. Risks from Logging [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources.
Justification: <no mitigation provided>

15. 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: Mitigated

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

Category: Tampering
Description: The web server 'Web Server' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'SQL Database' inputs and output.
Justification: Mitigated

17. Weak Access Control for a Resource [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Improper data protection of SQL Database can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification: <no mitigation provided>