# SAD Lab
# EXPERIMENT NO. 3

**Aim**: Exercise on threat modeling.

**Theory**:

1. What is threat modeling?

   Threat modeling is a structured process with these objectives: identify security requirements, pinpoint security threats and potential vulnerabilities, quantify threat and vulnerability criticality, and prioritize remediation methods.

   Threat modeling methods create these artifacts:

   - An abstraction of the system
   - Profiles of potential attackers, including their goals and methods
   - A catalog of threats that could arise

   **Work of threat modeling**

   Threat modeling works by identifying the types of threat agents that cause harm to an application or computer system. It adopts the perspective of malicious hackers to see how much damage they could do. When conducting threat modeling, organizations perform a thorough analysis of the software architecture, business context, and other artifacts (e.g., functional specifications, user documentation). This process enables a deeper understanding and discovery of important aspects of the system. Typically, organizations conduct threat modeling during the design stage (but it can occur at other stages) of a new application to help developers find vulnerabilities and become aware of the security implications of their design, code, and configuration decisions.

   Generally, developers perform threat modeling in four steps:

   - Diagram: What are we building?
   - Identify threats: What could go wrong?
   - Mitigate: What are we doing to defend against threats?
   - Validate: Have we acted on each of the previous steps?

**Best practices of threat modeling**

Conceptually, threat modeling is a simple process. So consider these five basic best practices when creating or updating a threat model:

1. Define the scope and depth of analysis. Determine the scope with stakeholders, then break down the depth of analysis for individual development teams so they can threaten to model the software.

2. Gain a visual understanding of what you're threat modeling. Create a diagram of the major system components (e.g., application server, data warehouse, thick client, database) and the interactions among those components.

3. Model the attack possibilities. Identify software assets, security controls, and threat agents and diagram their locations to create a security model of the system. Once you've modeled the system, you can identify what could go wrong (i.e., the threats) using methods like STRIDE.

4. Identify threats. To produce a list of potential attacks, ask questions such as the following:
    - Are there paths where a threat agent can reach an asset without going through a control?
    - Could a threat agent defeat this security control?
    - What must a threat agent do to defeat this control?

5. Create a traceability matrix of missing or weak security controls. Consider the threat agents and follow their control paths. If you reach the software asset without going through a security control, that's a potential attack. If you go through a control, consider whether it would halt a threat agent or whether the agent would have methods to bypass it.

---

2. Importance of threat modeling

Any application or system must be designed to withstand attacks. Yet, establishing the security requirements needed to achieve this can be complex. Attackers think and act differently from developers and users.

Threat modeling is a proactive method of uncovering threats not usually considered or found through code reviews and other types of audits. It enables a project team to determine which security controls an application needs to set effective countermeasures against potential threats, and how to resolve problems early on. This approach leads to far more secure applications, and by prioritizing anticipated threats, resources are used effectively.

Threat models are a vital part of a functional security development process. When threat modeling is part of the DevOps process developers can build security into a project during development and maintenance phases. This prevents common omissions such as failing to validate input, weak authentication, inadequate error handling and failing to encrypt data.

---

3. Process for threat modeling

Threat modeling consists of defining an enterprise's assets, identifying what function each application serves in the grand scheme, and assembling a security profile for each application. The process continues with identifying and prioritizing potential threats, then documenting both the harmful events and what actions to take to resolve them.


There are several different threat modeling frameworks and methodologies. However, the key steps are similar in most of these processes. They include:

- **Form a team**: This team should include all stakeholders, including business owners, developers, network architects, security experts and C-level execs. A diverse team will generate a more holistic threat model.

- **Establish the scope**: Define and describe what the model covers. For example, is it focused on an application, a network or the application and the infrastructure it runs on? Create an inventory of all components and data included and map them to architecture and data flow diagrams. Each data type must be classified.

- **Determine likely threats**: For all components that are threat targets, determine where threats exist. This what-if exercise builds broad, technical and unexpected threat scenarios, including threat or attack trees to identify possible vulnerabilities or weaknesses that could lead to compromise or failure. Threat modeling tools can help automate and streamline this step.

- **Rank each threat**: Determine the level of risk each threat poses and rank them to prioritize risk mitigation A simple but effective approach is to multiply the damage potential of a threat by the likelihood of it occurring.

- **Implement mitigations**: Decide how to mitigate each threat or reduce the risk to an acceptable level. The choices are to avoid risk, transfer it, reduce it or accept it.

- **Document results**: Document all findings and actions, so future changes to the application, threat landscape and operating environment can be quickly assessed and the threat model updated.

---

4. Models for threat modeling and STRIDE

**STRIDE**

Created by Microsoft in 1999, Spoofing Tampering Repudiation Information message Disclosure Denial of Service and Elevation of Privilege (STRIDE) is focused primarily on development and design. As the most mature threat modeling framework on the market, STRIDE has evolved considerably over the years to keep pace with the emergence of new types of threats.

**PASTA**

PASTA, or Process for Attack Simulation and Threat Analysis, establishes a seven-step process through which a business can evaluate a system from an attacker's perspective. It blends this approach with a comprehensive risk assessment and business impact analysis. In so doing, it allows one to both gain a better understanding of threat actors whilst also ensuring alignment between threat models and business objectives.

**Trike**

Originally created as a framework for carrying out security audits, the open-source Trike has since carved out a well-defined niche as a threat modeling tool for businesses that seek to consolidate threat modeling with risk management and risk assessments. In addition to defining and mapping existing systems and threat surfaces, Trike requires that a business establish its risk appetite prior to application.

**VAST**

VAST (Visual, Agile, Simple Threat modeling) was initially conceived in an effort to address the shortcomings of other threat modeling techniques. Built on the idea that different segments of an organization have different security concerns, VAST can model threats from both an application and an operational perspective. It is also specifically designed to support agile development, scalability, and automation.

**OCTAVE**

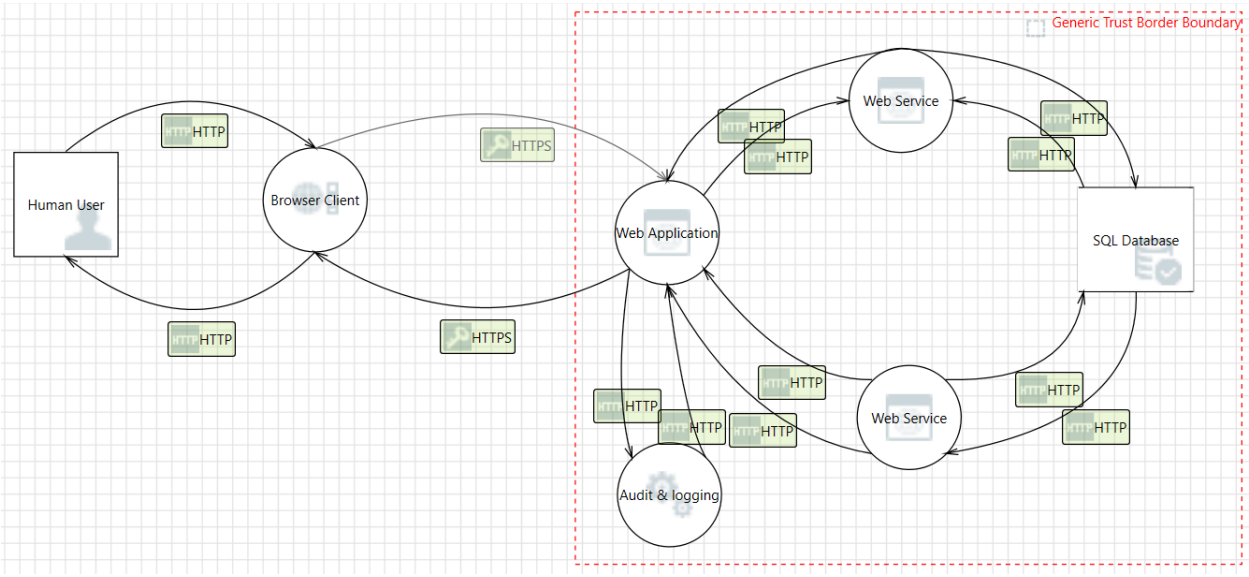
Another old hat framework, Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) was developed with a cybersecurity perspective in mind. It is focused primarily on organizational and operational risks, and seeks to reduce unnecessary documentation, better-define assets, and more effectively wrap threat models into a business's overall security strategy. It's an excellent option for an organization that seeks to promote better risk-awareness, although it doesn't scale particularly well.

**MITRE ATT&CK**

Built from real-world observation, MITRE ATT&CK is as much a knowledge base as a threat model. In addition to threat modeling, ATT&CK also provides frameworks for penetration testing, cybersecurity, and defense development. From a threat modeling perspective, ATT&CK is, as one might expect from the name, focused primarily on a cyberattack's lifecycle.

**Procedure**:

**Diagram**




**Analysis View**

1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]
   **Category**: Spoofing
   **Description**: Human User may be spoofed by an attacker and this may lead to unauthorized access to Browser Client. Consider using a standard authentication mechanism to identify the external entity.

2. Elevation Using Impersonation [State: Not Started] [Priority: High]
   **Category**: Elevation Of Privilege
   **Description**: Browser Client may be able to impersonate the context of Human User in order to gain additional privilege.

3. Weak Access Control for a Resource [State: Not Started] [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.

4. Spoofing of Source Data Store SQL Database [State: Not Started] [Priority: High]
   **Category**: Spoofing
   **Description**: SQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to Web Service. Consider using a standard authentication mechanism to identify the source data store.

5. Spoofing of Destination Data Store SQL Database [State: Not Started] [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.

6. Potential SQL Injection Vulnerability for SQL Database [State: Not Started] [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.

7. Potential Excessive Resource Consumption for Web Service or SQL Database [State: Not Started] [Priority: High]
   **Category**: Denial Of Service
   **Description**: Does Web Service 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.

8. Web Service Process Memory Tampered [State: Not Started] [Priority: High]
   **Category**: Tampering
   **Description**: If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

9. Cross Site Scripting [State: Not Started] [Priority: High]
   **Category**: Tampering
   **Description**: The web server 'Web Application' could be subject to a cross-site scripting attack because it does not sanitize untrusted input.

10. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Web Application may be able to impersonate the context of Web Service in order to gain additional privilege.

11. Web Application Process Memory Tampered [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: If Web Application is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Service executes (for example, passing back a function pointer.), then Web Application can tamper with Web Service. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

12. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Web Service may be able to impersonate the context of Web Application in order to gain additional privilege.

13. Spoofing of Destination Data Store SQL Database [State: Not Started] [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.

14. Potential SQL Injection Vulnerability for SQL Database [State: Not Started] [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.

15. Potential Excessive Resource Consumption for Web Service or SQL Database [State: Not Started] [Priority: High]
    **Category**: Denial Of Service
    **Description**: Does Web Service 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.

16. Spoofing of Source Data Store SQL Database [State: Not Started] [Priority: High]
    **Category**: Spoofing
    **Description**: SQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to Web Service. Consider using a standard authentication mechanism to identify the source data store.

17. Weak Access Control for a Resource [State: Not Started] [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.

18. Web Service Process Memory Tampered [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

19. Cross Site Scripting [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: The web server 'Web Application' could be subject to a cross-site scripting attack because it does not sanitize untrusted input.

20. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Web Application may be able to impersonate the context of Web Service in order to gain additional privilege.

21. Web Service Process Memory Tampered [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

22. Cross Site Scripting [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: The web server 'Web Application' could be subject to a cross-site scripting attack because it does not sanitize untrusted input.

23. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Web Application may be able to impersonate the context of Web Service in order to gain additional privilege.

24. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Audit & logging may be able to impersonate the context of Web Application in order to gain additional privilege.

25. Audit & logging Process Memory Tampered [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: If Audit & logging is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Audit & logging can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

26. Cross Site Scripting [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: The web server 'Web Application' could be subject to a cross-site scripting attack because it does not sanitize untrusted input.

27. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege

**Description**: Web Application may be able to impersonate the context of Audit & logging in order to gain additional privilege.

28. Elevation Using Impersonation [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Browser Client may be able to impersonate the context of Web Application in order to gain additional privilege.

29. Web Application Process Memory Tampered [State: Not Started] [Priority: High]
    **Category**: Tampering
    **Description**: If Web Application is given access to memory, such as shared memory or pointers, or is given the ability to control what Browser Client executes (for example, passing back a function pointer.), then Web Application can tamper with Browser Client. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

30. Spoofing the Web Application Process [State: Not Started] [Priority: High]
    **Category**: Spoofing
    **Description**: Web Application may be spoofed by an attacker and this may lead to unauthorized access to Browser Client. Consider using a standard authentication mechanism to identify the source process.

31. Potential Data Repudiation by Browser Client [State: Not Started] [Priority: High]
    **Category**: Repudiation
    **Description**: Browser Client claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

32. Potential Process Crash or Stop for Browser Client [State: Not Started] [Priority: High]
    **Category**: Denial Of Service
    **Description**: Browser Client crashes, halts, stops or runs slowly; in all cases violating an availability metric.

33. Data Flow HTTPS Is Potentially Interrupted [State: Not Started] [Priority: High]
    **Category**: Denial Of Service
    **Description**: An external agent interrupts data flowing across a trust boundary in either direction.

34. Browser Client May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: Web Application may be able to remotely execute code for Browser Client.

35. Elevation by Changing the Execution Flow in Browser Client [State: Not Started] [Priority: High]
    **Category**: Elevation Of Privilege
    **Description**: An attacker may pass data into Browser Client in order to change the flow of program execution within Browser Client to the attacker's choosing.

**Report**

# Threat Modeling Report

Created on 06-08-2022 22:11:13

**Threat Model Name:**

**Owner:**

**Reviewer:**

**Contributors:**

**Description:**

**Assumptions:**

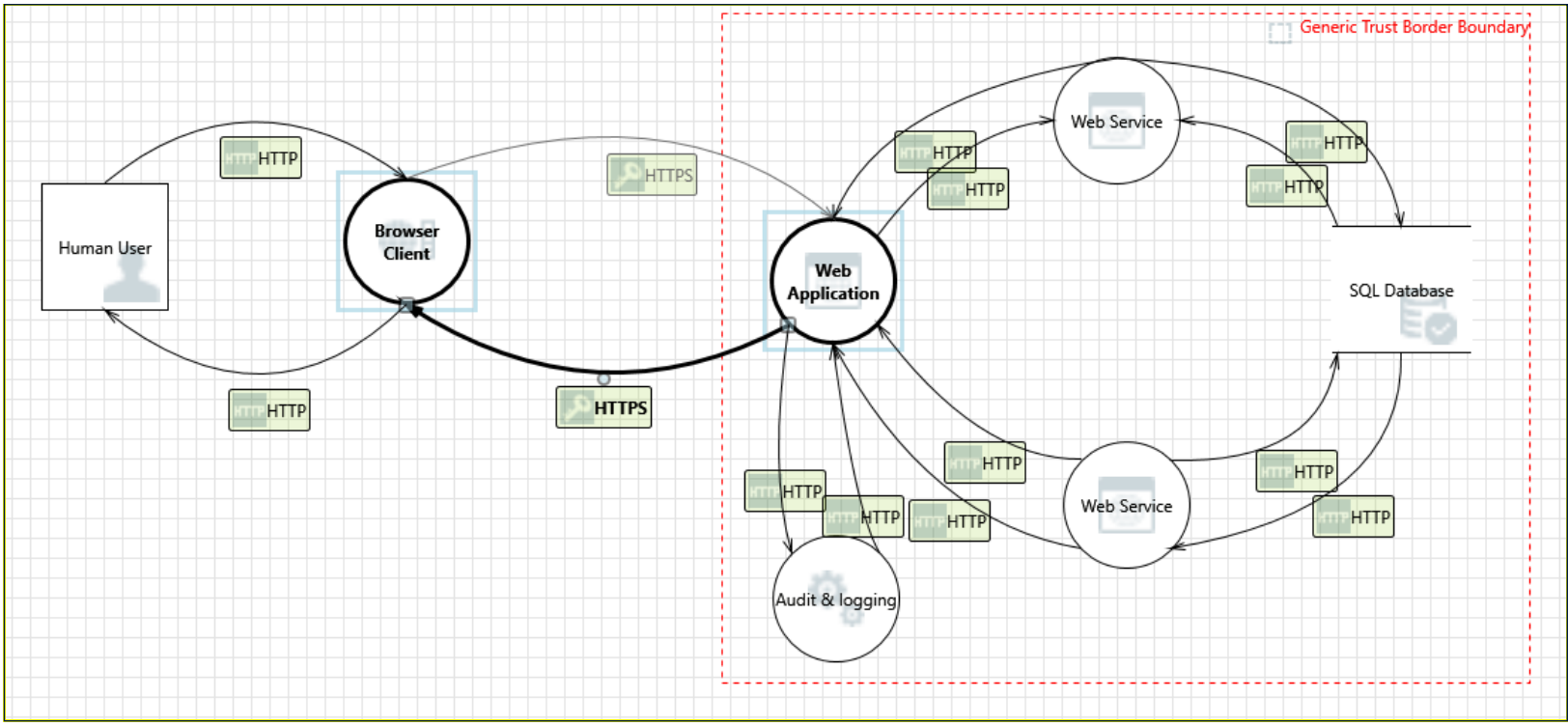
**External Dependencies:**

### Threat Model Summary:

| | |
|---|---|
| Not Started | 35 |
| Not Applicable | 0 |
| Needs Investigation | 0 |
| Mitigation Implemented | 0 |
| Total | 35 |
| Total Migrated | 0 |

---

## Diagram: Diagram 1




### Validation Messages:

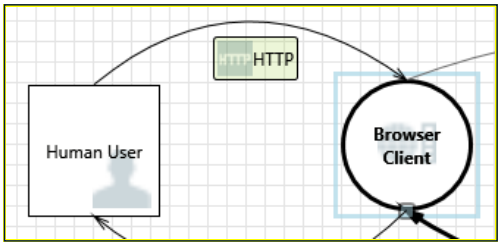
1. **Error:** The connector should be attached to two elements.

### Diagram 1 Diagram Summary:

| | |
|---|---|
| Not Started | 35 |
| Not Applicable | 0 |
| Needs Investigation | 0 |
| Mitigation Implemented | 0 |
| Total | 35 |

| | |
|---|---|
| Total Migrated | 0 |

## Interaction: HTTP



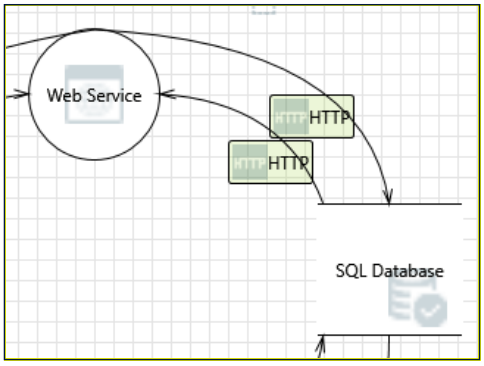

**1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]**

**Category:** Spoofing
**Description:** Human User may be spoofed by an attacker and this may lead to unauthorized access to Browser Client. Consider using a standard authentication mechanism to identify the external entity.
**Justification:** <no mitigation provided>

**2. Elevation Using Impersonation [State: Not Started] [Priority: High]**

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

## Interaction: HTTP



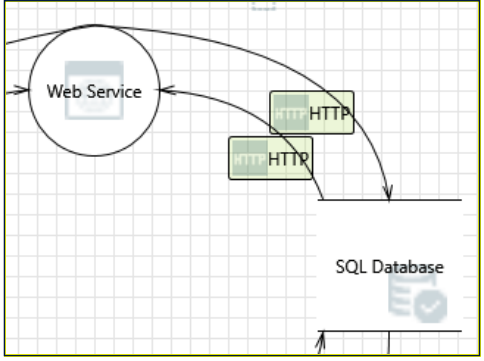

**3. Weak Access Control for a Resource [State: Not Started] [Priority: High]**

**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>

**4. Spoofing of Source Data Store SQL Database [State: Not Started] [Priority: High]**

**Category:** Spoofing
**Description:** SQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to Web Service. Consider using a standard authentication mechanism to identify the source data store.
**Justification:** <no mitigation provided>

## Interaction: HTTP




**5. Spoofing of Destination Data Store SQL Database [State: Not Started] [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>

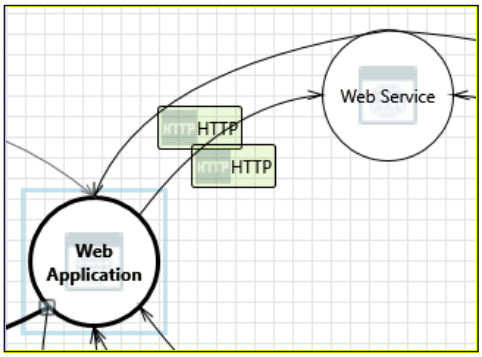
**6. Potential SQL Injection Vulnerability for SQL Database [State: Not Started] [Priority: High]**

**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>

**7. Potential Excessive Resource Consumption for Web Service or SQL Database [State: Not Started] [Priority: High]**

**Category:** Denial Of Service
**Description:** Does Web Service 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: HTTP




**8. Web Service Process Memory Tampered [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
**Justification:** <no mitigation provided>

**9. Cross Site Scripting [State: Not Started] [Priority: High]**

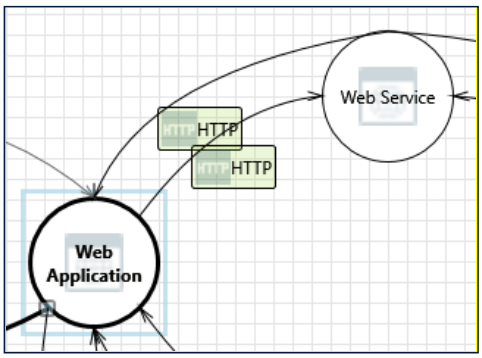
**Category:** Tampering
**Description:** The web server 'Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
**Justification:** <no mitigation provided>

**10. Elevation Using Impersonation [State: Not Started] [Priority: High]**

**Category:** Elevation Of Privilege

**Description:** Web Application may be able to impersonate the context of Web Service in order to gain additional privilege.
**Justification:** <no mitigation provided>

## Interaction: HTTP



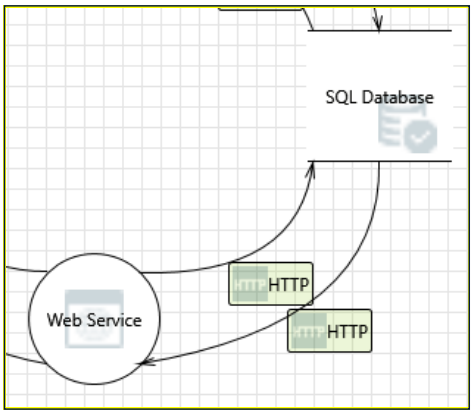

**11. Web Application Process Memory Tampered [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** If Web Application is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Service executes (for example, passing back a function pointer.), then Web Application can tamper with Web Service. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
**Justification:** <no mitigation provided>

**12. Elevation Using Impersonation [State: Not Started] [Priority: High]**

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

## Interaction: HTTP




**13. Spoofing of Destination Data Store SQL Database [State: Not Started] [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>

**14. Potential SQL Injection Vulnerability for SQL Database [State: Not Started] [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>

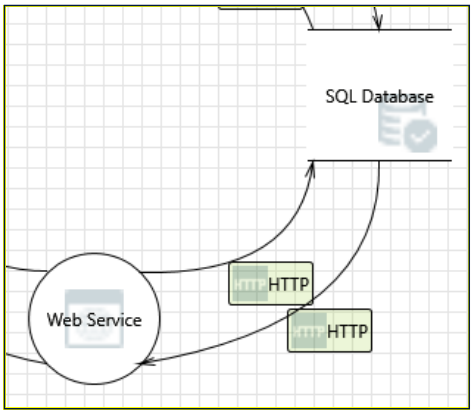
**15. Potential Excessive Resource Consumption for Web Service or SQL Database [State: Not Started] [Priority: High]**

**Category:** Denial Of Service
**Description:** Does Web Service 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: HTTP




**16. Spoofing of Source Data Store SQL Database [State: Not Started] [Priority: High]**

**Category:** Spoofing

**Description:** SQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to Web Service. Consider using a standard authentication mechanism to identify the source data store.

**Justification:** <no mitigation provided>

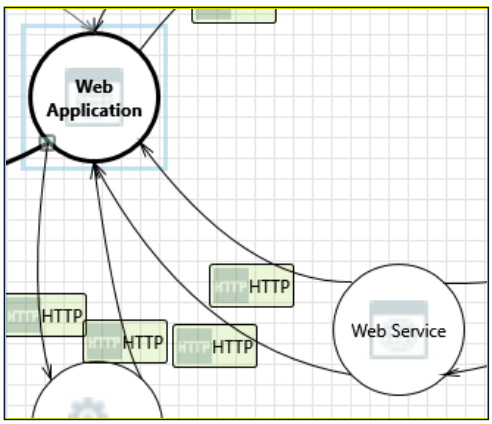
**17. Weak Access Control for a Resource [State: Not Started] [Priority: High]**

**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>

## Interaction: HTTP




**18. Web Service Process Memory Tampered [State: Not Started] [Priority: High]**

**Category:** Tampering

**Description:** If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

**Justification:** <no mitigation provided>

**19. Cross Site Scripting [State: Not Started] [Priority: High]**

**Category:** Tampering

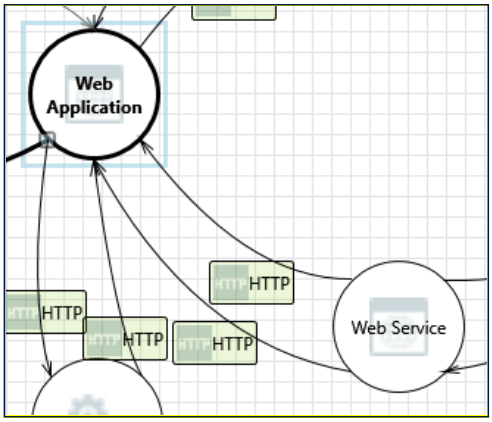
**Description:** The web server 'Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.

**Justification:** <no mitigation provided>

**20. Elevation Using Impersonation [State: Not Started] [Priority: High]**

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

## Interaction: HTTP




**21. Web Service Process Memory Tampered [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
**Justification:** <no mitigation provided>

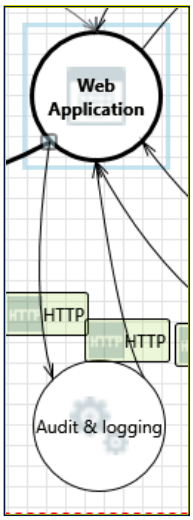
**22. Cross Site Scripting [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** The web server 'Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
**Justification:** <no mitigation provided>

**23. Elevation Using Impersonation [State: Not Started] [Priority: High]**

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

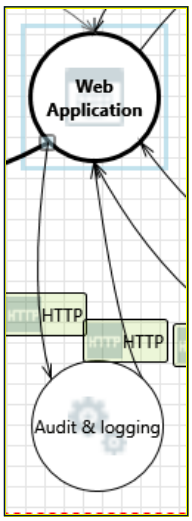
## Interaction: HTTP




**24. Elevation Using Impersonation [State: Not Started] [Priority: High]**

**Category:** Elevation Of Privilege
**Description:** Audit & logging may be able to impersonate the context of Web Application in order to gain additional privilege.
**Justification:** <no mitigation provided>

## Interaction: HTTP




**25. Audit & logging Process Memory Tampered [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** If Audit & logging is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Audit & logging can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
**Justification:** <no mitigation provided>

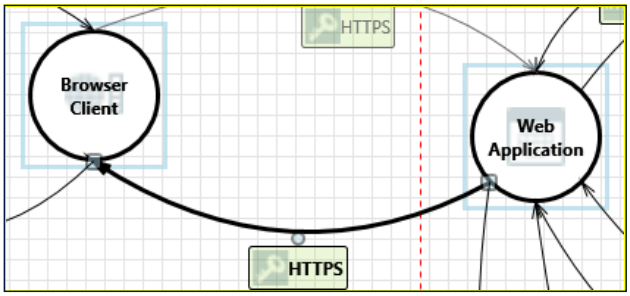
**26. Cross Site Scripting [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** The web server 'Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
**Justification:** <no mitigation provided>

**27. Elevation Using Impersonation [State: Not Started] [Priority: High]**

**Category:** Elevation Of Privilege
**Description:** Web Application may be able to impersonate the context of Audit & logging in order to gain additional privilege.
**Justification:** <no mitigation provided>

## Interaction: HTTPS




**28. Elevation Using Impersonation [State: Not Started] [Priority: High]**

**Category:** Elevation Of Privilege
**Description:** Browser Client may be able to impersonate the context of Web Application in order to gain additional privilege.
**Justification:** <no mitigation provided>

**29. Web Application Process Memory Tampered [State: Not Started] [Priority: High]**

**Category:** Tampering
**Description:** If Web Application is given access to memory, such as shared memory or pointers, or is given the ability to control what Browser Client executes (for example, passing back a function pointer.), then Web Application can tamper with Browser Client. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
**Justification:** <no mitigation provided>

**30. Spoofing the Web Application Process [State: Not Started] [Priority: High]**

**Category:** Spoofing
**Description:** Web Application may be spoofed by an attacker and this may lead to unauthorized access to Browser Client. Consider using a standard authentication mechanism to identify the source process.
**Justification:** <no mitigation provided>

**31. Potential Data Repudiation by Browser Client [State: Not Started] [Priority: High]**

**Category:** Repudiation
**Description:** Browser Client claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
**Justification:** <no mitigation provided>

**32. Potential Process Crash or Stop for Browser Client [State: Not Started] [Priority: High]**

**Category:** Denial Of Service
**Description:** Browser Client crashes, halts, stops or runs slowly; in all cases violating an availability metric.
**Justification:** <no mitigation provided>

**33. Data Flow HTTPS Is Potentially Interrupted [State: Not Started] [Priority: High]**

**Category:** Denial Of Service
**Description:** An external agent interrupts data flowing across a trust boundary in either direction.
**Justification:** <no mitigation provided>

**34. Browser Client May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]**

**Category:** Elevation Of Privilege
**Description:** Web Application may be able to remotely execute code for Browser Client.
**Justification:** <no mitigation provided>

**35. Elevation by Changing the Execution Flow in Browser Client [State: Not Started] [Priority: High]**

**Category:** Elevation Of Privilege
**Description:** An attacker may pass data into Browser Client in order to change the flow of program execution within Browser Client to the attacker's choosing.
**Justification:** <no mitigation provided>

**Conclusion**:

Whichever tool is used, the threat modeling process should be repeated whenever the application, IT infrastructure or threat environment changes. That keeps the threat model current, as new threats emerge. Analyzing threats involves time and effort. It is not a checklist exercise, but it is better to find a vulnerability and fix it before hackers discover it, and threat modeling methodology is the best way to achieve this.