Threat Modeling Report

Created on 10/9/2022 11:54:16 AM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

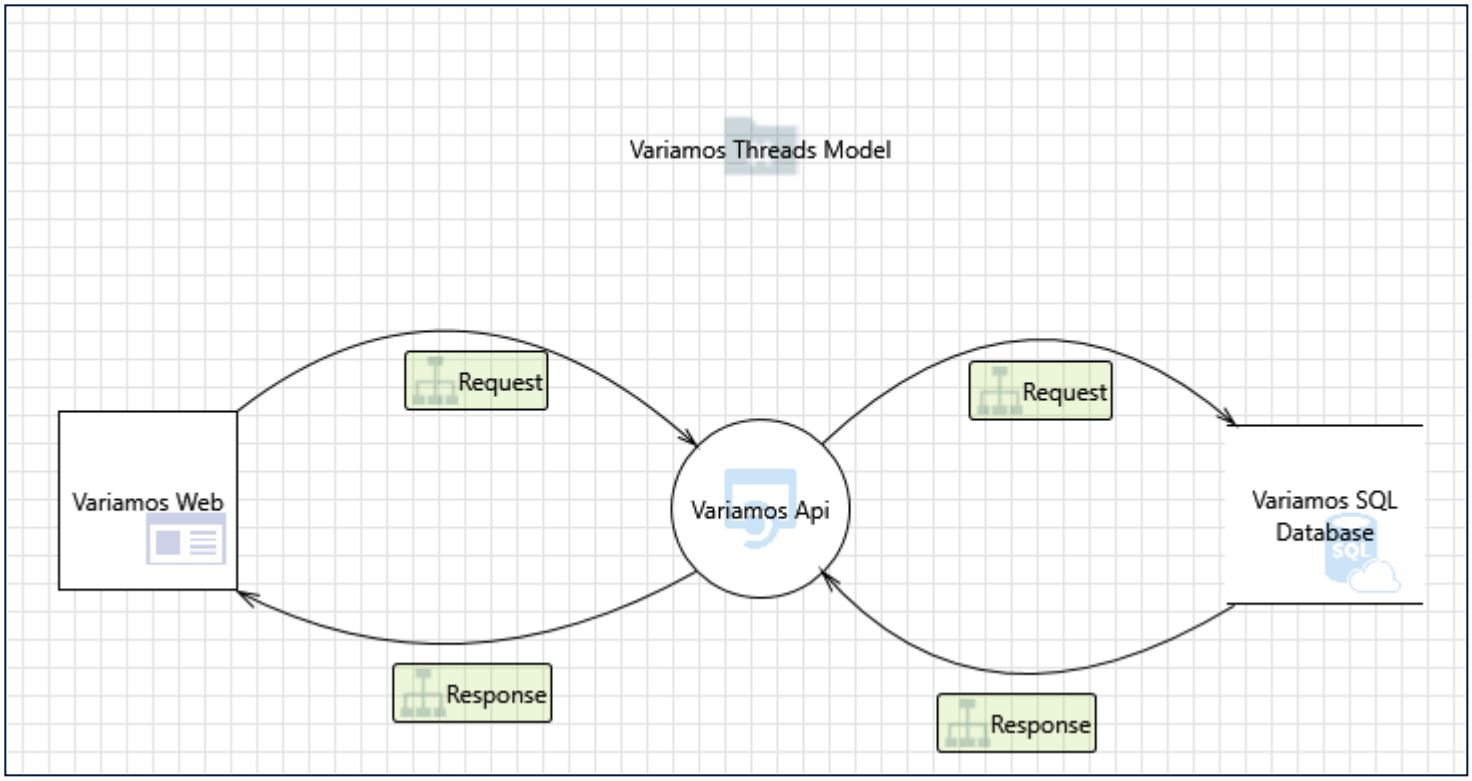
Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	26
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	26
Total Migrated	0

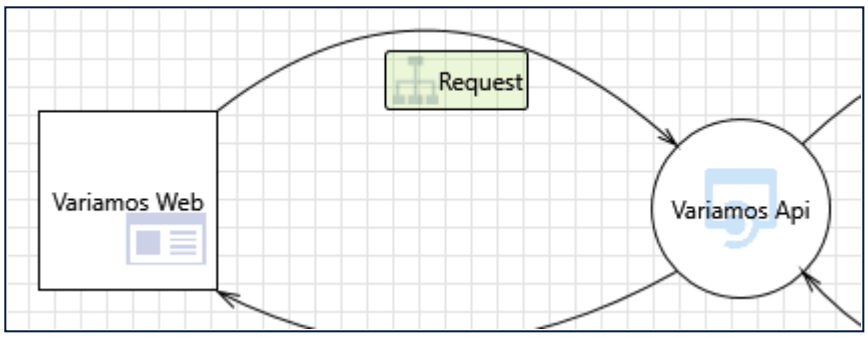
Diagram: VariamosThreadingDiagram



VariamosThreadingDiagram Diagram Summary:

Not Started	26
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	26
Total Migrated	0

Interaction: Request



1. An adversary may gain unauthorized access to Web API due to poor access control checks [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description: An adversary may gain unauthorized access to Web API due to poor access control checks

Justification: <no mitigation provided>

Possible Mitigation(s): Implement proper authorization mechanism in ASP.NET Web API. Refer: https://aka.ms/tmtauthz#authz-aspnet

SDL Phase: Implementation

2. An adversary can gain access to sensitive information from an API through error messages [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames - Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that proper exception handling is done in ASP.NET Web API. Refer: https://aka.ms/tmtxmgmt#exception

SDL Phase: Implementation

3. An adversary may retrieve sensitive data (e.g, auth tokens) persisted in browser storage [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary may retrieve sensitive data (e.g, auth tokens) persisted in browser storage

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that sensitive data relevant to Web API is not stored in browser's storage. Refer: https://aka.ms/tmtdata#api-browser

SDL Phase: Implementation

4. An adversary can gain access to sensitive data by sniffing traffic to Web API [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary can gain access to sensitive data by sniffing traffic to Web API

Justification: <no mitigation provided>

Possible Mitigation(s): Force all traffic to Web APIs over HTTPS connection. Refer: https://aka.ms/tmtcommsec#webapi-https

SDL Phase: Implementation

5. An adversary can gain access to sensitive data stored in Web API's config files [State: Not Started] [Priority: Medium]

Category:	Information Disclosure
Description:	An adversary can gain access to the config files. and if sensitive data is stored in it, it would be compromised.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Encrypt sections of Web API's configuration files that contain sensitive data. Refer: https://aka.ms/tmtconfigmgmt#config-sensitive
SDL Phase:	Implementation

6. Attacker can deny a malicious act on an API leading to repudiation issues [State: Not Started] [Priority: High]

Category:	Repudiation
Description:	Attacker can deny a malicious act on an API leading to repudiation issues
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that auditing and logging is enforced on Web API. Refer: https://aka.ms/tmtauditlog#logging-web-api
SDL Phase:	Design

7. An adversary can gain unauthorized access to API end points due to unrestricted cross domain requests [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can gain unauthorized access to API end points due to weak CORS configuration
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that only trusted origins are allowed if CORS is enabled on ASP.NET Web API. Refer: https://aka.ms/tmtconfigmgmt#cors-api Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET Web APIs. Refer: https://aka.ms/tmtsmgmt#csrf-api
SDL Phase:	Implementation

8. An adversary may spoof Variamos Web and gain access to Web API [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to the Web Application

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that standard authentication techniques are used to secure Web APIs. Refer: https://aka.ms/tmtauthn#authn-secure-api

SDL Phase: Design

9. An adversary may inject malicious inputs into an API and affect downstream processes [State: Not Started] [Priority: High]

Category: Tampering

Description: An adversary may inject malicious inputs into an API and affect downstream processes

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that model validation is done on Web API methods. Refer: https://aka.ms/tmtinputval#validation-api Implement input validation on all string type parameters accepted by Web API methods. Refer: https://aka.ms/tmtinputval#string-api

SDL Phase: Implementation

10. An adversary can gain access to sensitive data by performing SQL injection through Web API [State: Not Started] [Priority: High]

Category: Tampering

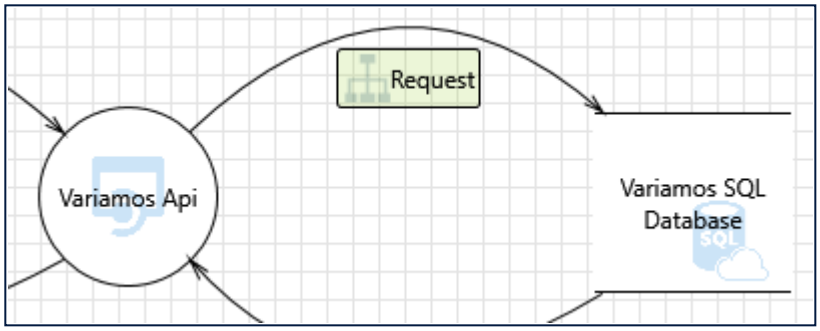
Description: SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that type-safe parameters are used in Web API for data access. Refer: https://aka.ms/tmtinputval#typesafe-api

SDL Phase: Implementation

Interaction: Request



11. An adversary can gain unauthorized access to Azure SQL database due to weak account policy [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	Due to poorly configured account policies, adversary can launch brute force attacks on Variamos SQL Database
Justification:	<no mitigation provided>
Possible Mitigation(s):	When possible use Azure Active Directory Authentication for connecting to SQL Database. Refer: https://aka.ms/tmt-th10a Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmt-th10b and https://aka.ms/tmt-th10c
SDL Phase:	Implementation

12. An adversary can gain unauthorized access to Azure SQL DB instances due to weak network security configuration. [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Azure SQL DB instances due to weak network security configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict access to Azure SQL Database instances by configuring server-level and database-level firewall rules to permit connections from selected networks (e.g. a virtual network or a custom set of IP addresses) where possible. Refer:https://aka.ms/tmt-th143
SDL Phase:	Implementation

13. An adversary can read confidential data due to weak connection string configuration [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary can read confidential data due to weak connection string configuration.
Justification:	<no mitigation provided>
Possible	Clients connecting to an Azure SQL Database instance using a connection string should

Mitigation(s): ensure encrypt=true and trustservercertificate=false are set. This configuration ensures that connections are encrypted only if there is a verifiable server certificate (otherwise the connection attempt fails). This helps protect against Man-In-The-Middle attacks. Refer: https://aka.ms/tmt-th144

SDL Phase: Implementation

14. An adversary having access to the storage container (e.g. physical access to the storage media) may be able to read sensitive data [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary having access to the storage container (e.g. physical access to the storage media) may be able to read sensitive data.

Justification: <no mitigation provided>

Possible Mitigation(s): Enable Transparent Data Encryption (TDE) on Azure SQL Database instances to have data encrypted at rest. Refer:https://aka.ms/tmt-th145a. Use the Always Encrypted feature to allow client applications to encrypt sensitive data before it is sent to the Azure SQL Database. Refer: https://aka.ms/tmt-th145b

SDL Phase: Implementation

15. A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description: A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.

Justification: <no mitigation provided>

Possible Mitigation(s): It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146

SDL Phase: Implementation

16. An adversary can deny actions performed on Variamos SQL Database due to a lack of auditing [State: Not Started] [Priority: Medium]

Category: Repudiation

Description: An adversary can deny actions performed on Variamos SQL Database due to a lack of auditing.

Justification: <no mitigation provided>

Possible Mitigation(s): Enable auditing on Azure SQL Database instances to track and log database events. After configuring and customizing the audited events, enable threat detection to receive alerts on

anomalous database activities indicating potential security threats. Refer: https://aka.ms/tmt-th147

SDL Phase: Design

17. An adversary can gain long term, persistent access to an Azure SQL DB instance through the compromise of local user account password(s) [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description: An adversary can gain long term, persistent access to an Azure SQL DB instance through the compromise of local user account password(s).

Justification: <no mitigation provided>

Possible Mitigation(s): It is recommended to rotate user account passwords (e.g. those used in connection strings) regularly, in accordance with your organization's policies. Store secrets in a secret storage solution (e.g. Azure Key Vault).

SDL Phase: Implementation

18. An adversary may abuse weak Variamos SQL Database configuration [State: Not Started] [Priority: High]

Category: Elevation of Privileges

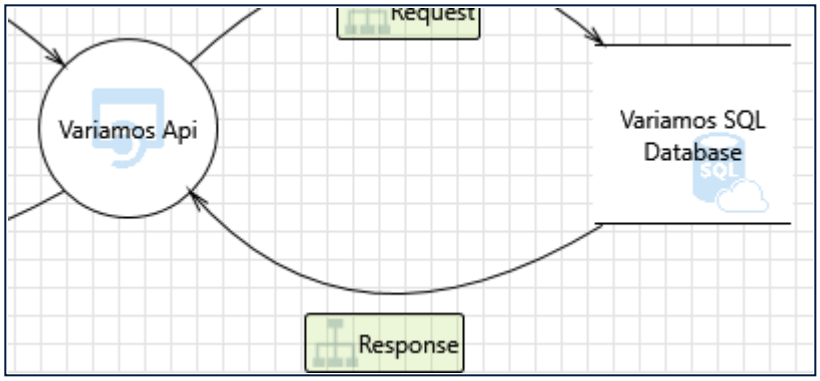
Description: An adversary may abuse weak Variamos SQL Database configuration.

Justification: <no mitigation provided>

Possible Mitigation(s): Enable SQL Vulnerability Assessment to gain visibility into the security posture of your Azure SQL Database instances. Acting on the assessment results help reduce attack surface and enhance your database security. Refer: https://aka.ms/tmt-th149

SDL Phase: Implementation

Interaction: Response



19. An adversary may gain unauthorized access to Web API due to poor access control checks [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description: An adversary may gain unauthorized access to Web API due to poor access control checks

Justification: <no mitigation provided>

Possible Mitigation(s): Implement proper authorization mechanism in ASP.NET Web API. Refer: https://aka.ms/tmtauthz#authz-aspnet

SDL Phase: Implementation

20. An adversary can gain access to sensitive information from an API through error messages [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames - Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that proper exception handling is done in ASP.NET Web API. Refer: https://aka.ms/tmtxmgmt#exception

SDL Phase: Implementation

21. An adversary can gain access to sensitive data by sniffing traffic to Web API [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary can gain access to sensitive data by sniffing traffic to Web API

Justification: <no mitigation provided>

Possible Mitigation(s): Force all traffic to Web APIs over HTTPS connection. Refer: https://aka.ms/tmtcommsec#webapi-https

SDL Phase: Implementation

22. An adversary can gain access to sensitive data stored in Web API's config files [State: Not Started] [Priority: Medium]

Category: Information Disclosure

Description: An adversary can gain access to the config files. and if sensitive data is stored in it, it would be compromised.

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt sections of Web API's configuration files that contain sensitive data. Refer: <a href="https://aka.ms/tmtconfigmgmt#config-

sensitive">https://aka.ms/tmtconfigmgmt#config-sensitive

SDL Phase: Implementation

23. Attacker can deny a malicious act on an API leading to repudiation issues [State: Not Started] [Priority: High]

Category: Repudiation

Description: Attacker can deny a malicious act on an API leading to repudiation issues

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that auditing and logging is enforced on Web API. Refer: https://aka.ms/tmtauditlog#logging-web-api

SDL Phase: Design

24. An adversary may spoof Variamos SQL Database and gain access to Web API [State: Not Started] [Priority: High]

Category: Spoofing

Description: If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to the Web Application

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that standard authentication techniques are used to secure Web APIs. Refer: https://aka.ms/tmtauthn#authn-secure-api

SDL Phase: Design

25. An adversary may inject malicious inputs into an API and affect downstream processes [State: Not Started] [Priority: High]

Category: Tampering

Description: An adversary may inject malicious inputs into an API and affect downstream processes

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that model validation is done on Web API methods. Refer: https://aka.ms/tmtinputval#validation-api Implement input validation on all string type parameters accepted by Web API methods. Refer: https://aka.ms/tmtinputval#string-api

SDL Phase: Implementation

26. An adversary can gain access to sensitive data by performing SQL injection through Web API [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. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that type-safe parameters are used in Web API for data access. Refer: https://aka.ms/tmtinputval#typesafe-api
SDL Phase:	Implementation