Threat Modeling Report

Created on 21/12/2023 04:42:38 p. m.
Threat Model Name:
Owner:
Reviewer:
Contributors:
Description:
Assumptions:
External Dependencies:

Threat Model Summary:

Not Started	147
Not Applicable	8
Needs Investigation	1
Mitigation Implemented	0
Total	156
Total Migrated	0

Diagram: Diagram 1

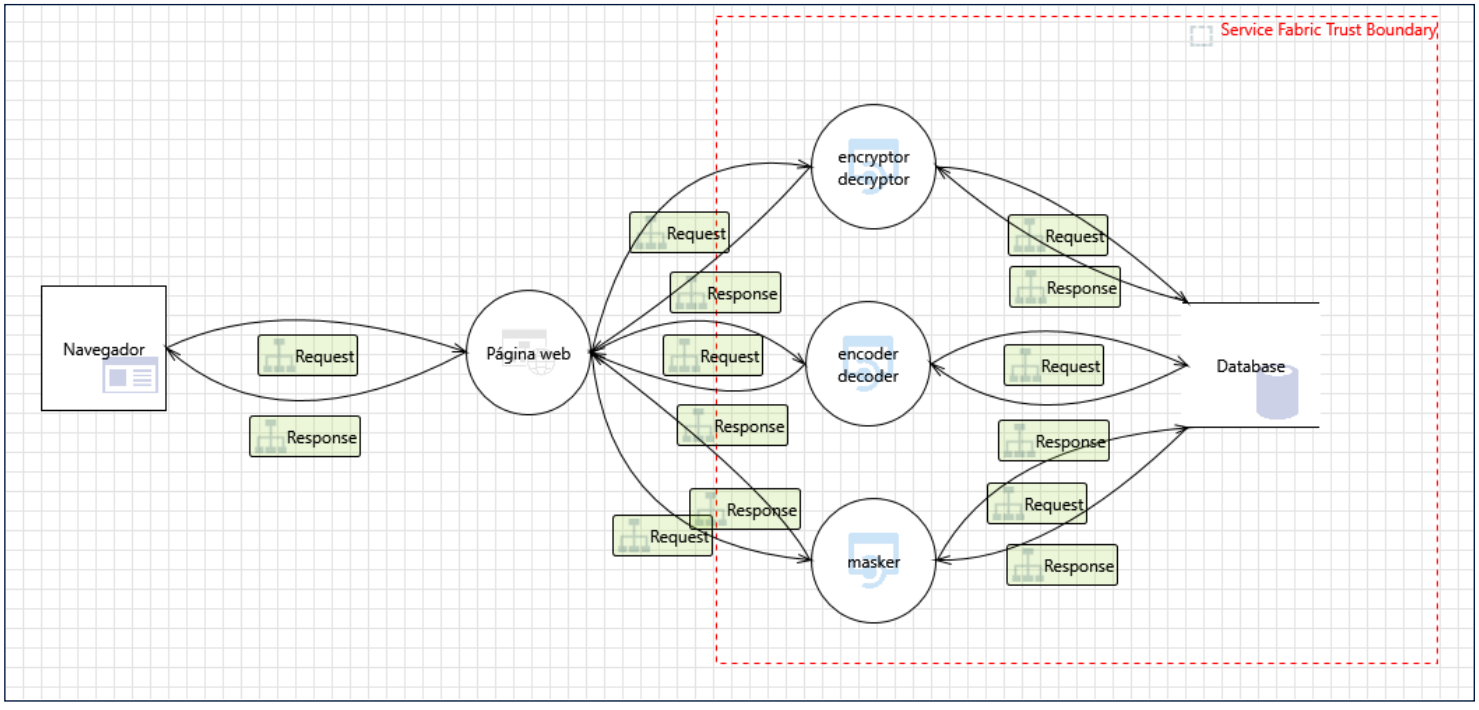
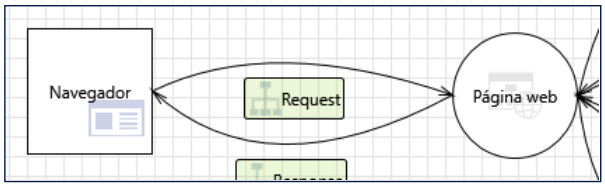


Diagram 1 Diagram Summary:

Not Started	147
Not Applicable	8
Needs Investigation	1
Mitigation Implemented	0
Total	156
Total Migrated	0

Interaction: Request



1. An adversary can perform action on behalf of other user due to lack of controls against cross domain requests [State: Needs Investigation] [Priority: High]

Category:	Denial of Service
Description:	Failure to restrict requests originating from third party domains may result in unauthorized actions or access of data
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that authenticated ASP.NET pages incorporate UI Redressing or clickjacking defences. Refer: https://aka.ms/tmtconfigmgmt#ui-defenses Ensure that only trusted origins are allowed if CORS is enabled on ASP.NET Web Applications. Refer: https://aka.ms/tmtconfigmgmt#cors-aspnet Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET web pages. Refer: https://aka.ms/tmtsmgmt#csrf-asp
SDL Phase:	Implementation

2. An adversary may bypass critical steps or perform actions on behalf of other users (victims) due to improper validation logic [State: Not Applicable] [Priority: High]

Category:	Elevation of Privileges
Description:	Failure to restrict the privileges and access rights to the application to individuals who require the privileges or access rights may result into unauthorized use of data due to inappropriate rights settings and validation.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that administrative interfaces are appropriately locked down. Refer: https://aka.ms/tmtauthn#admin-interface-lockdown Enforce sequential step order when processing business logic flows. Refer: https://aka.ms/tmtauthz#sequential-logic Ensure that proper authorization is in place and principle of least privileges is followed. Refer: https://aka.ms/tmtauthz#principle-least-privilege Business logic and resource access authorization decisions should not be based on incoming request parameters. Refer: https://aka.ms/tmtauthz#logic-request-parameters Ensure that content and resources are not enumerable or accessible via forceful browsing. Refer: https://aka.ms/tmtauthz#enumerable-browsing
SDL Phase:	Implementation

3. An adversary can reverse weakly encrypted or hashed content [State: Not Applicable] [Priority: High]

Category:	Information Disclosure
Description:	An adversary can reverse weakly encrypted or hashed content
Justification:	Aparte del cifrado la información se encripta
Possible Mitigation(s):	Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Use only approved symmetric block ciphers and key lengths. Refer: https://aka.ms/tmtcrypto#cipher-length Use approved block cipher modes and initialization vectors for symmetric ciphers. Refer: https://aka.ms/tmtcrypto#vector-ciphers Use approved asymmetric algorithms, key lengths, and padding. Refer: https://aka.ms/tmtcrypto#padding Use approved random number generators. Refer: https://aka.ms/tmtcrypto#numgen Do not use symmetric stream ciphers. Refer: https://aka.ms/tmtcrypto#stream-ciphers Use approved MAC/HMAC/keyed hash algorithms. Refer: https://aka.ms/tmtcrypto#mac-hash Use only approved cryptographic hash functions. Refer: https://aka.ms/tmtcrypto#hash-functions Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls
SDL Phase:	Implementation

4. An adversary may gain access to sensitive data from log files [State: Not Applicable] [Priority: High]

Category:	Information Disclosure
Description:	An adversary may gain access to sensitive data from log files

Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that the application does not log sensitive user data. Refer: https://aka.ms/tmtauditlog#log-sensitive-data Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access
SDL Phase:	Implementation

5. An adversary may gain access to unmasked sensitive data such as credit card numbers [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary may gain access to unmasked sensitive data such as credit card numbers
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that sensitive data displayed on the user screen is masked. Refer: https://aka.ms/tmtdata#data-mask
SDL Phase:	Implementation

6. An adversary can gain access to certain pages or the site as a whole. [State: Not Started] [Priority: Medium]

Category:	Information Disclosure
Description:	Robots.txt is often found in your site's root directory and exists to regulate the bots that crawl your site. This is where you can grant or deny permission to all or some specific search engine robots to access certain pages or your site as a whole. The standard for this file was developed in 1994 and is known as the Robots Exclusion Standard or Robots Exclusion Protocol. Detailed info about the robots.txt protocol can be found at robotstxt.org.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that administrative interfaces are appropriately locked down. Refer: https://aka.ms/tmtauthn#admin-interface-lockdown
SDL Phase:	Implementation

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

Category:	Information Disclosure
Description:	An adversary may conduct man in the middle attack and downgrade TLS connection to clear text protocol, or forcing browser communication to pass through a proxy server that he controls. This may happen because the application may use mixed content or HTTP Strict Transport Security policy is not ensured.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Applications available over HTTPS must use secure cookies. Refer: https://aka.ms/tmtsmgmt#https-secure-cookies Enable HTTP Strict Transport Security (HSTS). Refer: https://aka.ms/tmtcommsec#http-hsts
SDL Phase:	Implementation

8. An adversary can gain access to sensitive information 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):	Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Exceptions should fail safely. Refer: https://aka.ms/tmtxmgmt#fail ASP.NET applications must disable tracing and debugging prior to deployment. Refer: https://aka.ms/tmtconfigmgmt#trace-deploy Implement controls to prevent username enumeration. Refer: https://aka.ms/tmtauthn#controls-username-enum
SDL Phase:	Implementation

9. An adversary may gain access to sensitive data from uncleared browser cache [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary may gain access to sensitive data from uncleared browser cache

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that sensitive content is not cached on the browser. Refer: https://aka.ms/tmtdata#cache-browser

SDL Phase: Implementation

10. Attacker can deny the malicious act and remove the attack foot prints leading to repudiation issues [State: Not Started] [Priority: Medium]

Category: Repudiation

Description: Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that auditing and logging is enforced on the application. Refer: https://aka.ms/tmtauditlog#auditing Ensure that log rotation and separation are in place. Refer: https://aka.ms/tmtauditlog#log-rotation Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access Ensure that User Management Events are Logged. Refer: https://aka.ms/tmtauditlog#user-management

SDL Phase: Implementation

11. An adversary can get access to a user's session due to improper logout and timeout [State: Not Started] [Priority: High]

Category: Spoofing

Description: The session cookies is the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token he would be able to access all user data and perform all actions on behalf of user.

Justification: <no mitigation provided>

Possible Mitigation(s): Set up session for inactivity lifetime. Refer: https://aka.ms/tmtsmgmt#inactivity-lifetime Implement proper logout from the application. Refer: https://aka.ms/tmtsmgmt#proper-app-logout

SDL Phase: Implementation

12. An adversary can get access to a user's session due to insecure coding practices [State: Not Started] [Priority: High]

Category: Spoofing

Description: The session cookies is the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token he would be able to access all user data and perform all actions on behalf of user.

Justification: <no mitigation provided>

Possible Mitigation(s): Enable ValidateRequest attribute on ASP.NET Pages. Refer: https://aka.ms/tmtconfigmgmt#validate-aspnet Encode untrusted web output prior to rendering. Refer: https://aka.ms/tmtinputval#rendering Avoid using Html.Raw in Razor views. Refer: https://aka.ms/tmtinputval#html-razor Sanitization should be applied on form fields that accept all characters e.g, rich text editor . Refer: https://aka.ms/tmtinputval#richtext Do not assign DOM elements to sinks that do not have inbuilt encoding . Refer: https://aka.ms/tmtinputval#inbuilt-encode

SDL Phase: Implementation

13. An adversary can spoof the target web application due to insecure TLS certificate configuration [State: Not Started] [Priority: High]

Category: Spoofing

Description: Ensure that TLS certificate parameters are configured with correct values

Justification: <no mitigation provided>

Possible Mitigation(s): Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls

SDL Phase: Implementation

14. An adversary can steal sensitive data like user credentials [State: Not Started] [Priority: High]

Category: Spoofing

Description: Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal

credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor,

Justification: <no mitigation provided>

Possible Mitigation(s): Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs. Refer: https://aka.ms/tmtdata#autocomplete-input Perform input validation and filtering on all string type Model properties. Refer: https://aka.ms/tmtinputval#typemodel Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe Enable step up or adaptive authentication. Refer: https://aka.ms/tmtauthn#step-up-adaptive-authn Implement forgot password functionalities securely. Refer: https://aka.ms/tmtauthn#forgot-pword-fxn Ensure that password and account policy are implemented. Refer: https://aka.ms/tmtauthn#pword-account-policy Implement input validation on all string type parameters accepted by Controller methods. Refer: https://aka.ms/tmtinputval#string-method

SDL Phase: Implementation

15. Attackers can steal user session cookies due to insecure cookie attributes [State: Not Started] [Priority: High]

Category: Spoofing

Description: The session cookies is the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token he would be able to access all user data and perform all actions on behalf of user.

Justification: <no mitigation provided>

Possible Mitigation(s): Applications available over HTTPS must use secure cookies. Refer: https://aka.ms/tmtsmgmt#https-secure-cookies All http based application should specify http only for cookie definition. Refer: https://aka.ms/tmtsmgmt#cookie-definition

SDL Phase: Implementation

16. An adversary can create a fake website and launch phishing attacks [State: Not Started] [Priority: High]

Category: Spoofing

Description: Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication

Justification: <no mitigation provided>

Possible Mitigation(s): Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls Ensure that authenticated ASP.NET pages incorporate UI Redressing or clickjacking defences. Refer: https://aka.ms/tmtconfigmgmt#ui-defenses Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe

SDL Phase: Implementation

17. An adversary may spoof Navegador and gain access to Web Application [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): Consider using a standard authentication mechanism to authenticate to Web Application. Refer: https://aka.ms/tmtauthn#standard-authn-web-app

SDL Phase: Design

18. An adversary can deface the target web application by injecting malicious code or uploading dangerous files [State: Not Started] [Priority: High]

Category: Tampering

Description: Website defacement is an attack on a website where the attacker changes the visual appearance of the site or a webpage.

Justification: <no mitigation provided>

Possible Mitigation(s): Implement Content Security Policy (CSP), and disable inline javascript. Refer: https://aka.ms/tmtconfigmgmt#csp-js Enable browser's XSS filter. Refer: https://aka.ms/tmtconfigmgmt#xss-filter Access third party javascripts from trusted sources only. Refer: https://aka.ms/tmtconfigmgmt#js-trusted Enable ValidateRequest attribute on ASP.NET Pages. Refer: https://aka.ms/tmtconfigmgmt#validate-aspnet Ensure that each page that could contain user controllable content opts out of automatic MIME sniffing . Refer: https://aka.ms/tmtinputval#out-sniffing Use locally-hosted latest versions of JavaScript libraries . Refer: <a

href="https://aka.ms/tmtconfigmgmt#local-js">https://aka.ms/tmtconfigmgmt#local-js Ensure appropriate controls are in place when accepting files from users. Refer: https://aka.ms/tmtinputval#controls-users Disable automatic MIME sniffing. Refer: https://aka.ms/tmtconfigmgmt#mime-sniff Encode untrusted web output prior to rendering. Refer: https://aka.ms/tmtinputval#rendering Perform input validation and filtering on all string type Model properties. Refer: https://aka.ms/tmtinputval#typemodel Ensure that the system has inbuilt defences against misuse. Refer: https://aka.ms/tmtauditlog#inbuilt-defenses Enable HTTP Strict Transport Security (HSTS). Refer: https://aka.ms/tmtcommsec#http-hsts Implement input validation on all string type parameters accepted by Controller methods. Refer: https://aka.ms/tmtinputval#string-method Avoid using Html.Raw in Razor views. Refer: https://aka.ms/tmtinputval#html-razor Sanitization should be applied on form fields that accept all characters e.g, rich text editor . Refer: https://aka.ms/tmtinputval#richtext Do not assign DOM elements to sinks that do not have inbuilt encoding . Refer: https://aka.ms/tmtinputval#inbuilt-encode

SDL Phase: Implementation

19. An attacker steals messages off the network and replays them in order to steal a user's session [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An attacker steals messages off the network and replays them in order to steal a user's session
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Implementation

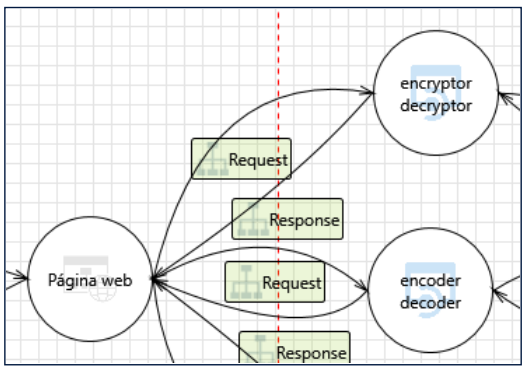
20. An adversary can gain access to sensitive data by performing SQL injection through Web App [State: Not Started] [Priority: High]

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 Application for data access. Refer: https://aka.ms/tmtinputval#typesafe
SDL Phase:	Implementation

21. An adversary can gain access to sensitive data stored in Web App's config files [State: Not Started] [Priority: High]

Category:	Tampering
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 App's configuration files that contain sensitive data. Refer: https://aka.ms/tmtdata#encrypt-data
SDL Phase:	Implementation

Interaction: Request



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

Category:	Elevation of Privileges
Description:	An adversary may gain unauthorized access to Web API due to poor access control checks
Justification:	No hay un sistema de roles
Possible Mitigation(s):	Implement proper authorization mechanism in ASP.NET Web API. Refer: https://aka.ms/tmtauthz#authz-aspnet
SDL Phase:	Implementation

23. An adversary may gain unauthorized access to Service Fabric cluster operations [State: Not Applicable] [Priority: High]

Category:	Elevation of Privileges
Description:	If RBAC is not implemented on Service Fabric, clients may have over-privileged access on the fabric's cluster operations
Justification:	No hay un sistema de roles
Possible Mitigation(s):	Restrict client's access to cluster operations using RBAC. Refer: https://aka.ms/tmtauthz#cluster-rbac
SDL Phase:	Design

24. 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

25. 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

26. An adversary can gain access to unencrypted secrets in Service Fabric applications [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. If secrets are not encrypted, an adversary who can gain access to them can abuse them.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Encrypt secrets in Service Fabric applications. Refer: https://aka.ms/tmtdata#fabric-apps
SDL Phase:	Implementation

27. 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

28. 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

29. An adversary may gain unauthorized access to resources in Service Fabric [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If a service fabric cluster is not secured, it allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict anonymous access to Service Fabric Cluster. Refer: https://aka.ms/tmtauthn#anon-access-cluster
SDL Phase:	Implementation

30. An adversary can spoof a node and access Service Fabric cluster [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If the same certificate that is used for node-to-node security is used for client-to-node security, it will be easy for an adversary to spoof and join a new node, in case the client-to-node certificate (which is often stored locally) is compromised
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate. Refer: https://aka.ms/tmtauthn#fabric-cn-nn
SDL Phase:	Implementation

31. An adversary can potentially spoof a client if weaker client authentication channels are used [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Azure AD authentication provides better control on identity management and hence it is a better alternative to authenticate clients to Service Fabric
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use AAD to authenticate clients to service fabric clusters. Refer: https://aka.ms/tmtauthn#aad-client-fabric
SDL Phase:	Design

32. An adversary can spoof a node in Service Fabric cluster by using stolen certificates [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If self-signed or test certificates are stolen, it would be difficult to revoke them. An adversary can use stolen certificates and continue to get access to Service Fabric cluster.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA). Refer: https://aka.ms/tmtauthn#fabric-cert-ca
SDL Phase:	Design

33. An adversary may spoof Página 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

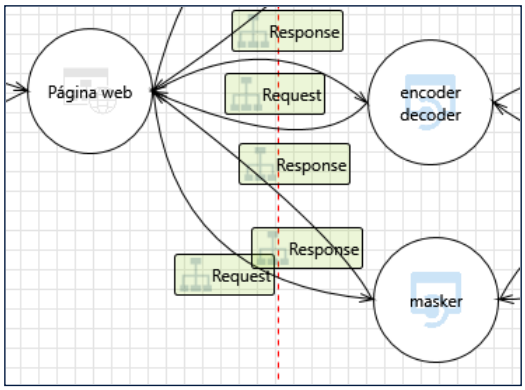
34. An adversary may inject malicious inputs into an API and affect downstream processes [State: Not Started] [Priority: High]

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

35. 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



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

Category:	Elevation of Privileges
Description:	An adversary may gain unauthorized access to Web API due to poor access control checks
Justification:	No hay un sistema de roles
Possible Mitigation(s):	Implement proper authorization mechanism in ASP.NET Web API. Refer: https://aka.ms/tmtauthz#authz-aspnet
SDL Phase:	Implementation

37. An adversary may gain unauthorized access to Service Fabric cluster operations [State: Not Applicable] [Priority: High]

Category:	Elevation of Privileges
Description:	If RBAC is not implemented on Service Fabric, clients may have over-privileged access on the fabric's cluster operations
Justification:	No hay un sistema de roles
Possible Mitigation(s):	Restrict client's access to cluster operations using RBAC. Refer: https://aka.ms/tmtauthz#cluster-rbac
SDL Phase:	Design

38. 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

39. 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

40. An adversary can gain access to unencrypted secrets in Service Fabric applications [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. If secrets are not encrypted, an adversary who can gain access to them can abuse them.

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt secrets in Service Fabric applications. Refer: https://aka.ms/tmtdata#fabric-apps

SDL Phase: Implementation

41. 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

42. 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

43. An adversary may gain unauthorized access to resources in Service Fabric [State: Not Started] [Priority: High]

Category: Spoofing

Description: If a service fabric cluster is not secured, it allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.

Justification: <no mitigation provided>

Possible Mitigation(s): Restrict anonymous access to Service Fabric Cluster. Refer: https://aka.ms/tmtauthn#anon-access-cluster

SDL Phase: Implementation

44. An adversary can spoof a node and access Service Fabric cluster [State: Not Started] [Priority: High]

Category: Spoofing

Description: If the same certificate that is used for node-to-node security is used for client-to-node security, it will be easy for an adversary to spoof and join a new node, in case the client-to-node certificate (which is often stored locally) is compromised

Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate. Refer: https://aka.ms/tmtauthn#fabric-cn-nn
SDL Phase:	Implementation

45. An adversary can potentially spoof a client if weaker client authentication channels are used [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Azure AD authentication provides better control on identity management and hence it is a better alternative to authenticate clients to Service Fabric
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use AAD to authenticate clients to service fabric clusters. Refer: https://aka.ms/tmtauthn#aad-client-fabric
SDL Phase:	Design

46. An adversary can spoof a node in Service Fabric cluster by using stolen certificates [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If self-signed or test certificates are stolen, it would be difficult to revoke them. An adversary can use stolen certificates and continue to get access to Service Fabric cluster.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA). Refer: https://aka.ms/tmtauthn#fabric-cert-ca
SDL Phase:	Design

47. An adversary may spoof Página 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

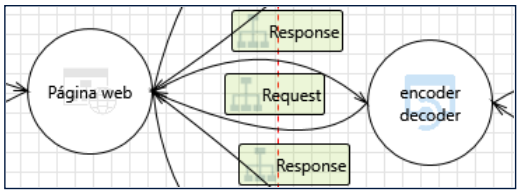
48. An adversary may inject malicious inputs into an API and affect downstream processes [State: Not Started] [Priority: High]

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

49. 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



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

Category:	Elevation of Privileges
Description:	An adversary may gain unauthorized access to Web API due to poor access control checks
Justification:	No hay un sistema de roles
Possible Mitigation(s):	Implement proper authorization mechanism in ASP.NET Web API. Refer: https://aka.ms/tmtauthz#authz-aspnet
SDL Phase:	Implementation

51. An adversary may gain unauthorized access to Service Fabric cluster operations [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	If RBAC is not implemented on Service Fabric, clients may have over-privileged access on the fabric's cluster operations
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict client's access to cluster operations using RBAC. Refer: https://aka.ms/tmtauthz#cluster-rbac
SDL Phase:	Design

52. 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

53. 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

54. An adversary can gain access to unencrypted secrets in Service Fabric applications [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. If secrets are not encrypted, an adversary who can gain access to them can abuse them.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Encrypt secrets in Service Fabric applications. Refer: https://aka.ms/tmtdata#fabric-apps
SDL Phase:	Implementation

55. 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

56. 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

57. An adversary may gain unauthorized access to resources in Service Fabric [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If a service fabric cluster is not secured, it allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict anonymous access to Service Fabric Cluster. Refer: https://aka.ms/tmtauthn#anon-access-cluster
SDL Phase:	Implementation

58. An adversary can spoof a node and access Service Fabric cluster [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If the same certificate that is used for node-to-node security is used for client-to-node security, it will be easy for an adversary to spoof and join a new node, in case the client-to-node certificate (which is often stored locally) is compromised
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate. Refer: https://aka.ms/tmtauthn#fabric-cn-nn
SDL Phase:	Implementation

59. An adversary can potentially spoof a client if weaker client authentication channels are used [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Azure AD authentication provides better control on identity management and hence it is a better alternative to authenticate clients to Service Fabric
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use AAD to authenticate clients to service fabric clusters. Refer: https://aka.ms/tmtauthn#aad-client-fabric
SDL Phase:	Design

60. An adversary can spoof a node in Service Fabric cluster by using stolen certificates [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If self-signed or test certificates are stolen, it would be difficult to revoke them. An adversary can use stolen certificates and continue to get access to Service Fabric cluster.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA). Refer: https://aka.ms/tmtauthn#fabric-cert-ca
SDL Phase:	Design

61. An adversary may spoof Página 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

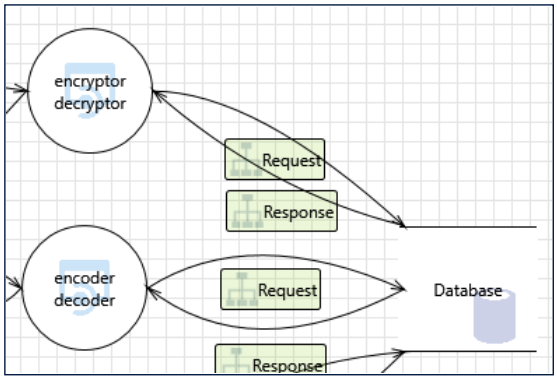
62. An adversary may inject malicious inputs into an API and affect downstream processes [State: Not Started] [Priority: High]

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

63. 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



64. An adversary may leverage the lack of monitoring systems and trigger anomalous traffic to database [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary may leverage the lack of intrusion detection and prevention of anomalous database activities and trigger anomalous traffic to database
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection
SDL Phase:	Design

65. An adversary can tamper critical database securables and deny the action [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary can tamper critical database securables and deny the action
Justification:	<no mitigation provided>

Possible Mitigation(s):	Add digital signature to critical database securables. Refer: https://aka.ms/tmtcrypto#securables-db
SDL Phase:	Design

66. An adversary can deny actions on database due to lack of auditing [State: Not Started] [Priority: Medium]

Category:	Repudiation
Description:	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities
SDL Phase:	Implementation

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

Category:	Information Disclosure
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 login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection Do not use dynamic queries in stored procedures. Refer: https://aka.ms/tmtinputval#stored-proc
SDL Phase:	Implementation

68. An adversary can gain access to sensitive PII or HBI data in database [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Additional controls like Transparent Data Encryption, Column Level Encryption, EKM etc. provide additional protection mechanism to high value PII or HBI data.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use strong encryption algorithms to encrypt data in the database. Refer: https://aka.ms/tmtcrypto#strong-db Ensure that sensitive data in database columns is encrypted. Refer: https://aka.ms/tmtdata#db-encrypted Ensure that database-level encryption (TDE) is enabled. Refer: https://aka.ms/tmtdata#tde-enabled Ensure that database backups are encrypted. Refer: https://aka.ms/tmtdata#backup Use SQL server EKM to protect encryption keys. Refer: https://aka.ms/tmtcrypto#ekm-keys Use AlwaysEncrypted feature if encryption keys should not be revealed to Database engine. Refer: https://aka.ms/tmtcrypto#keys-engine
SDL Phase:	Implementation

69. An adversary can gain unauthorized access to database due to loose authorization rules [State: Not Started] [Priority: High]

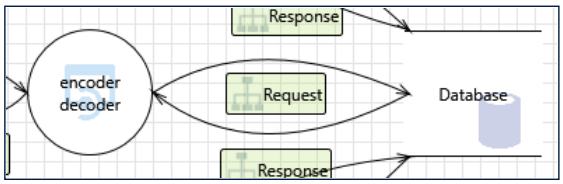
Category:	Elevation of Privileges
Description:	Database access should be configured with roles and privilege based on least privilege and need to know principle.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Implement Row Level Security RLS to prevent tenants from accessing each others data. Refer: https://aka.ms/tmtauthz#rls-tenants Sysadmin role should only have valid necessary users . Refer: https://aka.ms/tmtauthz#sysadmin-users
SDL Phase:	Implementation

70. An adversary can gain unauthorized access to database due to lack of network access protection [State: Not Started] [Priority: High]

Category:	Elevation of Privileges

Description:	If there is no restriction at network or host firewall level, to access the database then anyone can attempt to connect to the database from an unauthorized location
Justification:	<no mitigation provided>
Possible Mitigation(s):	Configure a Windows Firewall for Database Engine Access. Refer: https://aka.ms/tmtconfigmgmt#firewall-db
SDL Phase:	Implementation

Interaction: Request



71. An adversary can gain unauthorized access to database due to lack of network access protection [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	If there is no restriction at network or host firewall level, to access the database then anyone can attempt to connect to the database from an unauthorized location
Justification:	<no mitigation provided>
Possible Mitigation(s):	Configure a Windows Firewall for Database Engine Access. Refer: https://aka.ms/tmtconfigmgmt#firewall-db
SDL Phase:	Implementation

72. An adversary can gain unauthorized access to database due to loose authorization rules [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	Database access should be configured with roles and privilege based on least privilege and need to know principle.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Implement Row Level Security RLS to prevent tenants from accessing each others data. Refer: https://aka.ms/tmtauthz#rls-tenants Sysadmin role should only have valid necessary users . Refer: https://aka.ms/tmtauthz#sysadmin-users
SDL Phase:	Implementation

73. An adversary can gain access to sensitive PII or HBI data in database [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Additional controls like Transparent Data Encryption, Column Level Encryption, EKM etc. provide additional protection mechanism to high value PII or HBI data.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use strong encryption algorithms to encrypt data in the database. Refer: https://aka.ms/tmtcrypto#strong-db Ensure that sensitive data in database columns is encrypted. Refer: https://aka.ms/tmtdata#db-encrypted Ensure that database-level encryption (TDE) is enabled. Refer: https://aka.ms/tmtdata#tde-enabled Ensure that database backups are encrypted. Refer: https://aka.ms/tmtdata#backup Use SQL server EKM to protect encryption keys. Refer: https://aka.ms/tmtcrypto#ekm-keys Use AlwaysEncrypted feature if encryption keys should not be revealed to Database engine. Refer: https://aka.ms/tmtcrypto#keys-engine
SDL Phase:	Implementation

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

Category:	Information Disclosure
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 login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection Do not use dynamic queries in stored procedures. Refer: https://aka.ms/tmtinputval#stored-proc
SDL Phase:	Implementation

75. An adversary can deny actions on database due to lack of auditing [State: Not Started] [Priority: Medium]

Category:	Repudiation
Description:	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities
SDL Phase:	Implementation

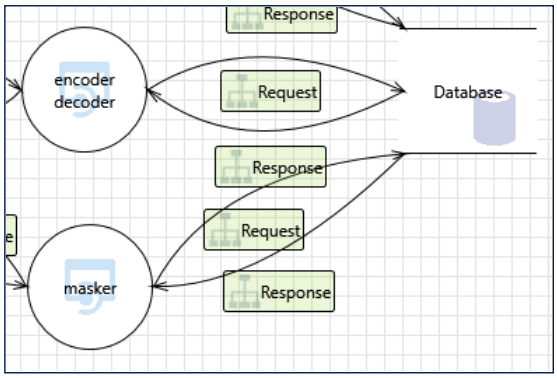
76. An adversary can tamper critical database securables and deny the action [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary can tamper critical database securables and deny the action
Justification:	<no mitigation provided>
Possible Mitigation(s):	Add digital signature to critical database securables. Refer: https://aka.ms/tmtcrypto#securables-db
SDL Phase:	Design

77. An adversary may leverage the lack of monitoring systems and trigger anomalous traffic to database [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary may leverage the lack of intrusion detection and prevention of anomalous database activities and trigger anomalous traffic to database
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection
SDL Phase:	Design

Interaction: Request



78. An adversary can gain unauthorized access to database due to lack of network access protection [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	If there is no restriction at network or host firewall level, to access the database then anyone can attempt to connect to the database from an unauthorized location
Justification:	<no mitigation provided>

Possible Mitigation(s):	Configure a Windows Firewall for Database Engine Access. Refer: https://aka.ms/tmtconfigmgmt#firewall-db
SDL Phase:	Implementation

79. An adversary can gain unauthorized access to database due to loose authorization rules [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	Database access should be configured with roles and privilege based on least privilege and need to know principle.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Implement Row Level Security RLS to prevent tenants from accessing each others data. Refer: https://aka.ms/tmtauthz#rls-tenants Sysadmin role should only have valid necessary users . Refer: https://aka.ms/tmtauthz#sysadmin-users
SDL Phase:	Implementation

80. An adversary can gain access to sensitive PII or HBI data in database [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Additional controls like Transparent Data Encryption, Column Level Encryption, EKM etc. provide additional protection mechanism to high value PII or HBI data.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use strong encryption algorithms to encrypt data in the database. Refer: https://aka.ms/tmtcrypto#strong-db Ensure that sensitive data in database columns is encrypted. Refer: https://aka.ms/tmtdata#db-encrypted Ensure that database-level encryption (TDE) is enabled. Refer: https://aka.ms/tmtdata#tde-enabled Ensure that database backups are encrypted. Refer: https://aka.ms/tmtdata#backup Use SQL server EKM to protect encryption keys. Refer: https://aka.ms/tmtcrypto#ekm-keys Use AlwaysEncrypted feature if encryption keys should not be revealed to Database engine. Refer: https://aka.ms/tmtcrypto#keys-engine
SDL Phase:	Implementation

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

Category:	Information Disclosure
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 login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection Do not use dynamic queries in stored procedures. Refer: https://aka.ms/tmtinputval#stored-proc
SDL Phase:	Implementation

82. An adversary can deny actions on database due to lack of auditing [State: Not Started] [Priority: Medium]

Category:	Repudiation
Description:	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities
SDL Phase:	Implementation

83. An adversary can tamper critical database securables and deny the action [State: Not Started] [Priority: High]

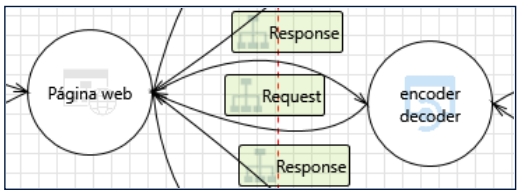
Category:	Tampering

Description:	An adversary can tamper critical database securables and deny the action
Justification:	<no mitigation provided>
Possible Mitigation(s):	Add digital signature to critical database securables. Refer: https://aka.ms/tmtcrypto#securables-db
SDL Phase:	Design

84. An adversary may leverage the lack of monitoring systems and trigger anomalous traffic to database [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary may leverage the lack of intrusion detection and prevention of anomalous database activities and trigger anomalous traffic to database
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection
SDL Phase:	Design

Interaction: Response



85. An adversary may gain unauthorized access to Service Fabric cluster operations [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	If RBAC is not implemented on Service Fabric, clients may have over-privileged access on the fabric's cluster operations
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict client's access to cluster operations using RBAC. Refer: https://aka.ms/tmtauthz#cluster-rbac
SDL Phase:	Design

86. An adversary can reverse weakly encrypted or hashed content [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary can reverse weakly encrypted or hashed content
Justification:	<no mitigation provided>
Possible Mitigation(s):	Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Use only approved symmetric block ciphers and key lengths. Refer: https://aka.ms/tmtcrypto#cipher-length Use approved block cipher modes and initialization vectors for symmetric ciphers. Refer: https://aka.ms/tmtcrypto#vector-ciphers Use approved asymmetric algorithms, key lengths, and padding. Refer: https://aka.ms/tmtcrypto#padding Use approved random number generators. Refer: https://aka.ms/tmtcrypto#numgen Do not use symmetric stream ciphers. Refer: https://aka.ms/tmtcrypto#stream-ciphers Use approved MAC/HMAC/keyed hash algorithms. Refer: https://aka.ms/tmtcrypto#mac-hash Use only approved cryptographic hash functions. Refer: https://aka.ms/tmtcrypto#hash-functions Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls
SDL Phase:	Implementation

87. An adversary may gain access to sensitive data from log files [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary may gain access to sensitive data from log files

Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that the application does not log sensitive user data. Refer: https://aka.ms/tmtauditlog#log-sensitive-data Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access
SDL Phase:	Implementation

88. An adversary can gain access to unencrypted secrets in Service Fabric applications [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. If secrets are not encrypted, an adversary who can gain access to them can abuse them.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Encrypt secrets in Service Fabric applications. Refer: https://aka.ms/tmtdata#fabric-apps
SDL Phase:	Implementation

89. An adversary can gain access to sensitive information 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):	Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Exceptions should fail safely. Refer: https://aka.ms/tmtxmgmt#fail ASP.NET applications must disable tracing and debugging prior to deployment. Refer: https://aka.ms/tmtconfigmgmt#trace-deploy Implement controls to prevent username enumeration. Refer: https://aka.ms/tmtauthn#controls-username-enum
SDL Phase:	Implementation

90. Attacker can deny the malicious act and remove the attack foot prints leading to repudiation issues [State: Not Started] [Priority: Medium]

Category:	Repudiation
Description:	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that auditing and logging is enforced on the application. Refer: https://aka.ms/tmtauditlog#auditing Ensure that log rotation and separation are in place. Refer: https://aka.ms/tmtauditlog#log-rotation Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access Ensure that User Management Events are Logged. Refer: https://aka.ms/tmtauditlog#user-management
SDL Phase:	Implementation

91. An adversary can spoof the target web application due to insecure TLS certificate configuration [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Ensure that TLS certificate parameters are configured with correct values
Justification:	<no mitigation provided>
Possible Mitigation(s):	Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls
SDL Phase:	Implementation

92. An adversary may gain unauthorized access to resources in Service Fabric [State: Not Started] [Priority: High]

Category:	Spoofing

Description:	If a service fabric cluster is not secured, it allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict anonymous access to Service Fabric Cluster. Refer: https://aka.ms/tmtauthn#anon-access-cluster
SDL Phase:	Implementation

93. An adversary can spoof a node and access Service Fabric cluster [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If the same certificate that is used for node-to-node security is used for client-to-node security, it will be easy for an adversary to spoof and join a new node, in case the client-to-node certificate (which is often stored locally) is compromised
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate. Refer: https://aka.ms/tmtauthn#fabric-cn-nn
SDL Phase:	Implementation

94. An adversary can steal sensitive data like user credentials [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor,
Justification:	<no mitigation provided>
Possible Mitigation(s):	Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs. Refer: https://aka.ms/tmtdata#autocomplete-input Perform input validation and filtering on all string type Model properties. Refer: https://aka.ms/tmtinputval#typemodel Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe Enable step up or adaptive authentication. Refer: https://aka.ms/tmtauthn#step-up-adaptive-authn Implement forgot password functionalities securely. Refer: https://aka.ms/tmtauthn#forgot-pword-fxn Ensure that password and account policy are implemented. Refer: https://aka.ms/tmtauthn#pword-account-policy Implement input validation on all string type parameters accepted by Controller methods. Refer: https://aka.ms/tmtinputval#string-method
SDL Phase:	Implementation

95. An adversary can potentially spoof a client if weaker client authentication channels are used [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Azure AD authentication provides better control on identity management and hence it is a better alternative to authenticate clients to Service Fabric
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use AAD to authenticate clients to service fabric clusters. Refer: https://aka.ms/tmtauthn#aad-client-fabric
SDL Phase:	Design

96. An adversary can spoof a node in Service Fabric cluster by using stolen certificates [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If self-signed or test certificates are stolen, it would be difficult to revoke them. An adversary can use stolen certificates and continue to get access to Service Fabric cluster.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA). Refer: https://aka.ms/tmtauthn#fabric-cert-ca
SDL Phase:	Design

97. An adversary can create a fake website and launch phishing attacks [State: Not Started] [Priority: High]

Category:	Spoofing

Description:	Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication
Justification:	<no mitigation provided>
Possible Mitigation(s):	Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls Ensure that authenticated ASP.NET pages incorporate UI Redressing or clickjacking defences. Refer: https://aka.ms/tmtconfigmgmt#ui-defenses Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe
SDL Phase:	Implementation

98. An adversary may spoof encoder decoder and gain access to Web Application [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):	Consider using a standard authentication mechanism to authenticate to Web Application. Refer: https://aka.ms/tmtauthn#standard-authn-web-app
SDL Phase:	Design

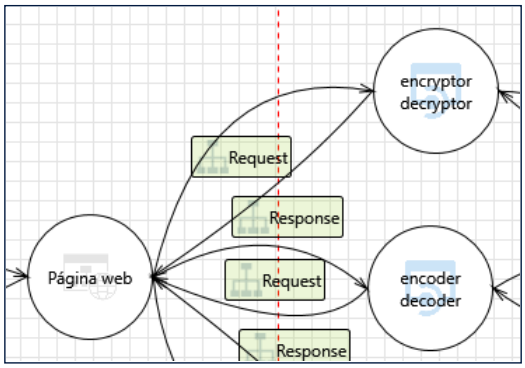
99. An adversary can gain access to sensitive data by performing SQL injection through Web App [State: Not Started] [Priority: High]

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 Application for data access. Refer: https://aka.ms/tmtinputval#typesafe
SDL Phase:	Implementation

100. An adversary can gain access to sensitive data stored in Web App's config files [State: Not Started] [Priority: High]

Category:	Tampering
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 App's configuration files that contain sensitive data. Refer: https://aka.ms/tmtdata#encrypt-data
SDL Phase:	Implementation

Interaction: Response



101. An adversary may gain unauthorized access to Service Fabric cluster operations [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	If RBAC is not implemented on Service Fabric, clients may have over-privileged access on the fabric's cluster operations

Justification: <no mitigation provided>

Possible Mitigation(s): Restrict client's access to cluster operations using RBAC. Refer: https://aka.ms/tmtauthz#cluster-rbac

SDL Phase: Design

102. An adversary can reverse weakly encrypted or hashed content [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary can reverse weakly encrypted or hashed content

Justification: <no mitigation provided>

Possible Mitigation(s): Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Use only approved symmetric block ciphers and key lengths. Refer: https://aka.ms/tmtcrypto#cipher-length Use approved block cipher modes and initialization vectors for symmetric ciphers. Refer: https://aka.ms/tmtcrypto#vector-ciphers Use approved asymmetric algorithms, key lengths, and padding. Refer: https://aka.ms/tmtcrypto#padding Use approved random number generators. Refer: https://aka.ms/tmtcrypto#numgen Do not use symmetric stream ciphers. Refer: https://aka.ms/tmtcrypto#stream-ciphers Use approved MAC/HMAC/keyed hash algorithms. Refer: https://aka.ms/tmtcrypto#mac-hash Use only approved cryptographic hash functions. Refer: https://aka.ms/tmtcrypto#hash-functions Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls

SDL Phase: Implementation

103. An adversary may gain access to sensitive data from log files [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary may gain access to sensitive data from log files

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that the application does not log sensitive user data. Refer: https://aka.ms/tmtauditlog#log-sensitive-data Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access

SDL Phase: Implementation

104. An adversary can gain access to unencrypted secrets in Service Fabric applications [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. If secrets are not encrypted, an adversary who can gain access to them can abuse them.

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt secrets in Service Fabric applications. Refer: https://aka.ms/tmtdata#fabric-apps

SDL Phase: Implementation

105. An adversary can gain access to sensitive information 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): Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Exceptions should fail safely. Refer: https://aka.ms/tmtxmgmt#fail ASP.NET applications must disable tracing and debugging prior to deployment. Refer: https://aka.ms/tmtconfigmgmt#trace-deploy Implement controls to prevent username enumeration. Refer: https://aka.ms/tmtauthn#controls-username-enum

SDL Phase: Implementation

106. Attacker can deny the malicious act and remove the attack foot prints leading to repudiation issues [State: Not Started] [Priority: Medium]

Category: Repudiation

Description: Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that auditing and logging is enforced on the application. Refer: https://aka.ms/tmtauditlog#auditing Ensure that log rotation and separation are in place. Refer: https://aka.ms/tmtauditlog#log-rotation Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access Ensure that User Management Events are Logged. Refer: https://aka.ms/tmtauditlog#user-management

SDL Phase: Implementation

107. An adversary can spoof the target web application due to insecure TLS certificate configuration [State: Not Started] [Priority: High]

Category: Spoofing

Description: Ensure that TLS certificate parameters are configured with correct values

Justification: <no mitigation provided>

Possible Mitigation(s): Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls

SDL Phase: Implementation

108. An adversary may gain unauthorized access to resources in Service Fabric [State: Not Started] [Priority: High]

Category: Spoofing

Description: If a service fabric cluster is not secured, it allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.

Justification: <no mitigation provided>

Possible Mitigation(s): Restrict anonymous access to Service Fabric Cluster. Refer: https://aka.ms/tmtauthn#anon-access-cluster

SDL Phase: Implementation

109. An adversary can spoof a node and access Service Fabric cluster [State: Not Started] [Priority: High]

Category: Spoofing

Description: If the same certificate that is used for node-to-node security is used for client-to-node security, it will be easy for an adversary to spoof and join a new node, in case the client-to-node certificate (which is often stored locally) is compromised

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate. Refer: https://aka.ms/tmtauthn#fabric-cn-nn

SDL Phase: Implementation

110. An adversary can steal sensitive data like user credentials [State: Not Started] [Priority: High]

Category: Spoofing

Description: Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor,

Justification: <no mitigation provided>

Possible Mitigation(s): Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs. Refer: https://aka.ms/tmtdata#autocomplete-input Perform input validation and filtering on all string type Model properties. Refer: https://aka.ms/tmtinputval#typemodel Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe Enable step up or adaptive authentication. Refer: https://aka.ms/tmtauthn#step-up-adaptive-authn Implement forgot password functionalities securely. Refer: https://aka.ms/tmtauthn#forgot-pword-fxn Ensure that password and account policy are implemented. Refer: <a

href="https://aka.ms/tmtauthn#pword-account-policy">https://aka.ms/tmtauthn#pword-account-policy Implement input validation on all string type parameters accepted by Controller methods. Refer: https://aka.ms/tmtinputval#string-method

SDL Phase: Implementation

111. An adversary can potentially spoof a client if weaker client authentication channels are used [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Azure AD authentication provides better control on identity management and hence it is a better alternative to authenticate clients to Service Fabric
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use AAD to authenticate clients to service fabric clusters. Refer: https://aka.ms/tmtauthn#aad-client-fabric
SDL Phase:	Design

112. An adversary can spoof a node in Service Fabric cluster by using stolen certificates [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If self-signed or test certificates are stolen, it would be difficult to revoke them. An adversary can use stolen certificates and continue to get access to Service Fabric cluster.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA). Refer: https://aka.ms/tmtauthn#fabric-cert-ca
SDL Phase:	Design

113. An adversary can create a fake website and launch phishing attacks [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication
Justification:	<no mitigation provided>
Possible Mitigation(s):	Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls Ensure that authenticated ASP.NET pages incorporate UI Redressing or clickjacking defences. Refer: https://aka.ms/tmtconfigmgmt#ui-defenses Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe
SDL Phase:	Implementation

114. An adversary may spoof encryptor decryptor and gain access to Web Application [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):	Consider using a standard authentication mechanism to authenticate to Web Application. Refer: https://aka.ms/tmtauthn#standard-authn-web-app
SDL Phase:	Design

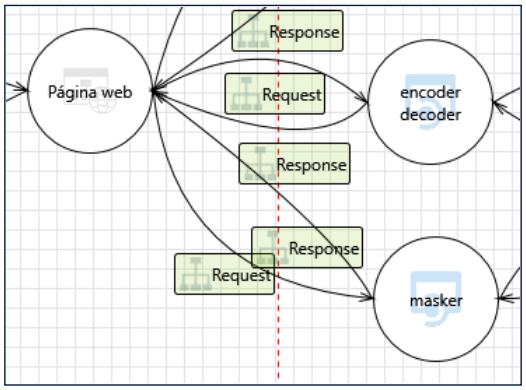
115. An adversary can gain access to sensitive data by performing SQL injection through Web App [State: Not Started] [Priority: High]

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 Application for data access. Refer: https://aka.ms/tmtinputval#typesafe
SDL Phase:	Implementation

116. An adversary can gain access to sensitive data stored in Web App's config files [State: Not Started] [Priority: High]

Category:	Tampering
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 App's configuration files that contain sensitive data. Refer: https://aka.ms/tmtdata#encrypt-data
SDL Phase:	Implementation

Interaction: Response



117. An adversary may gain unauthorized access to Service Fabric cluster operations [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	If RBAC is not implemented on Service Fabric, clients may have over-privileged access on the fabric's cluster operations
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict client's access to cluster operations using RBAC. Refer: https://aka.ms/tmtauthz#cluster-rbac
SDL Phase:	Design

118. An adversary can reverse weakly encrypted or hashed content [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary can reverse weakly encrypted or hashed content
Justification:	<no mitigation provided>
Possible Mitigation(s):	Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Use only approved symmetric block ciphers and key lengths. Refer: https://aka.ms/tmtcrypto#cipher-length Use approved block cipher modes and initialization vectors for symmetric ciphers. Refer: https://aka.ms/tmtcrypto#vector-ciphers Use approved asymmetric algorithms, key lengths, and padding. Refer: https://aka.ms/tmtcrypto#padding Use approved random number generators. Refer: https://aka.ms/tmtcrypto#numgen Do not use symmetric stream ciphers. Refer: https://aka.ms/tmtcrypto#stream-ciphers Use approved MAC/HMAC/keyed hash algorithms. Refer: https://aka.ms/tmtcrypto#mac-hash Use only approved cryptographic hash functions. Refer: https://aka.ms/tmtcrypto#hash-functions Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls
SDL Phase:	Implementation

119. An adversary may gain access to sensitive data from log files [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary may gain access to sensitive data from log files
Justification:	<no mitigation provided>

Possible Mitigation(s):	Ensure that the application does not log sensitive user data. Refer: https://aka.ms/tmtauditlog#log-sensitive-data Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access
SDL Phase:	Implementation

120. An adversary can gain access to unencrypted secrets in Service Fabric applications [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. If secrets are not encrypted, an adversary who can gain access to them can abuse them.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Encrypt secrets in Service Fabric applications. Refer: https://aka.ms/tmtdata#fabric-apps
SDL Phase:	Implementation

121. An adversary can gain access to sensitive information 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):	Do not expose security details in error messages. Refer: https://aka.ms/tmtxmgmt#messages Implement Default error handling page. Refer: https://aka.ms/tmtxmgmt#default Set Deployment Method to Retail in IIS. Refer: https://aka.ms/tmtxmgmt#deployment Exceptions should fail safely. Refer: https://aka.ms/tmtxmgmt#fail ASP.NET applications must disable tracing and debugging prior to deployment. Refer: https://aka.ms/tmtconfigmgmt#trace-deploy Implement controls to prevent username enumeration. Refer: https://aka.ms/tmtauthn#controls-username-enum
SDL Phase:	Implementation

122. Attacker can deny the malicious act and remove the attack foot prints leading to repudiation issues [State: Not Started] [Priority: Medium]

Category:	Repudiation
Description:	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that auditing and logging is enforced on the application. Refer: https://aka.ms/tmtauditlog#auditing Ensure that log rotation and separation are in place. Refer: https://aka.ms/tmtauditlog#log-rotation Ensure that Audit and Log Files have Restricted Access. Refer: https://aka.ms/tmtauditlog#log-restricted-access Ensure that User Management Events are Logged. Refer: https://aka.ms/tmtauditlog#user-management
SDL Phase:	Implementation

123. An adversary can spoof the target web application due to insecure TLS certificate configuration [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Ensure that TLS certificate parameters are configured with correct values
Justification:	<no mitigation provided>
Possible Mitigation(s):	Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls
SDL Phase:	Implementation

124. An adversary may gain unauthorized access to resources in Service Fabric [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	If a service fabric cluster is not secured, it allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.

Justification: <no mitigation provided>

Possible Mitigation(s): Restrict anonymous access to Service Fabric Cluster. Refer: https://aka.ms/tmtauthn#anon-access-cluster

SDL Phase: Implementation

125. An adversary can spoof a node and access Service Fabric cluster [State: Not Started] [Priority: High]

Category: Spoofing

Description: If the same certificate that is used for node-to-node security is used for client-to-node security, it will be easy for an adversary to spoof and join a new node, in case the client-to-node certificate (which is often stored locally) is compromised

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate. Refer: https://aka.ms/tmtauthn#fabric-cn-nn

SDL Phase: Implementation

126. An adversary can steal sensitive data like user credentials [State: Not Started] [Priority: High]

Category: Spoofing

Description: Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor,

Justification: <no mitigation provided>

Possible Mitigation(s): Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs. Refer: https://aka.ms/tmtdata#autocomplete-input Perform input validation and filtering on all string type Model properties. Refer: https://aka.ms/tmtinputval#typemodel Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe Enable step up or adaptive authentication. Refer: https://aka.ms/tmtauthn#step-up-adaptive-authn Implement forgot password functionalities securely. Refer: https://aka.ms/tmtauthn#forgot-pword-fxn Ensure that password and account policy are implemented. Refer: https://aka.ms/tmtauthn#pword-account-policy Implement input validation on all string type parameters accepted by Controller methods. Refer: https://aka.ms/tmtinputval#string-method

SDL Phase: Implementation

127. An adversary can potentially spoof a client if weaker client authentication channels are used [State: Not Started] [Priority: High]

Category: Spoofing

Description: Azure AD authentication provides better control on identity management and hence it is a better alternative to authenticate clients to Service Fabric

Justification: <no mitigation provided>

Possible Mitigation(s): Use AAD to authenticate clients to service fabric clusters. Refer: https://aka.ms/tmtauthn#aad-client-fabric

SDL Phase: Design

128. An adversary can spoof a node in Service Fabric cluster by using stolen certificates [State: Not Started] [Priority: High]

Category: Spoofing

Description: If self-signed or test certificates are stolen, it would be difficult to revoke them. An adversary can use stolen certificates and continue to get access to Service Fabric cluster.

Justification: <no mitigation provided>

Possible Mitigation(s): Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA). Refer: https://aka.ms/tmtauthn#fabric-cert-ca

SDL Phase: Design

129. An adversary can create a fake website and launch phishing attacks [State: Not Started] [Priority: High]

Category: Spoofing

Description: Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication

Justification:	<no mitigation provided>
Possible Mitigation(s):	Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections. Refer: https://aka.ms/tmtcommsec#x509-ssltls Ensure that authenticated ASP.NET pages incorporate UI Redressing or clickjacking defences. Refer: https://aka.ms/tmtconfigmgmt#ui-defenses Validate all redirects within the application are closed or done safely. Refer: https://aka.ms/tmtinputval#redirect-safe
SDL Phase:	Implementation

130. An adversary may spoof masker and gain access to Web Application [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):	Consider using a standard authentication mechanism to authenticate to Web Application. Refer: https://aka.ms/tmtauthn#standard-authn-web-app
SDL Phase:	Design

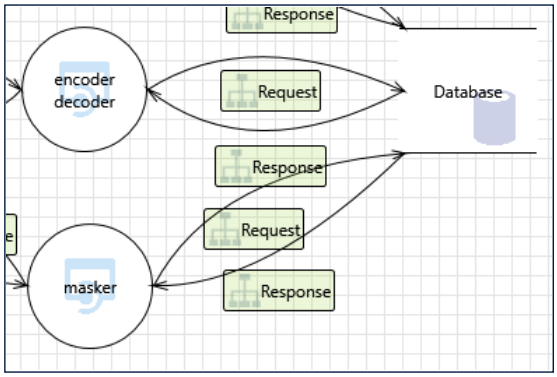
131. An adversary can gain access to sensitive data by performing SQL injection through Web App [State: Not Started] [Priority: High]

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 Application for data access. Refer: https://aka.ms/tmtinputval#typesafe
SDL Phase:	Implementation

132. An adversary can gain access to sensitive data stored in Web App's config files [State: Not Started] [Priority: High]

Category:	Tampering
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 App's configuration files that contain sensitive data. Refer: https://aka.ms/tmtdata#encrypt-data
SDL Phase:	Implementation

Interaction: Response



133. 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

134. 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

135. 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

136. 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

137. 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

138. An adversary may spoof 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

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

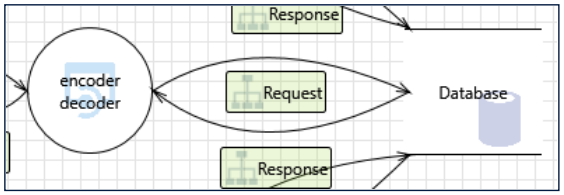
Category:	Tampering
Description:	An adversary may inject malicious inputs into an API and affect downstream processes
Justification:	<no mitigation provided>

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

140. 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: Response



141. 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

142. 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

143. 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

144. 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

145. 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

146. An adversary may spoof 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

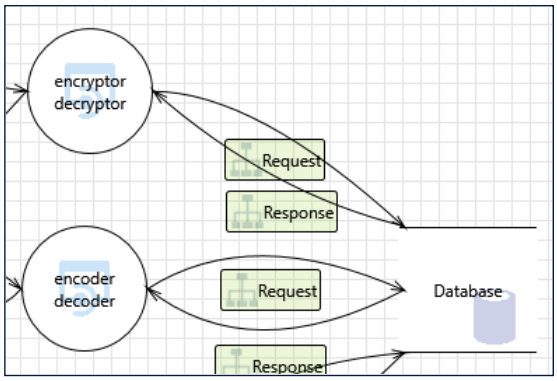
147. An adversary may inject malicious inputs into an API and affect downstream processes [State: Not Started] [Priority: High]

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

148. 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: Response



149. 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

150. 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

151. An adversary may spoof 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

152. 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

153. 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

154. 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

155. 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

156. 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