Threat Modeling Report

Created on 25/05/2025 3:46:55 PM
Threat Model Name:
Owner:
Reviewer:
Contributors:
Description:
Assumptions:
External Dependencies:

Threat Model Summary:

Not Started	85
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	85
Total Migrated	0

Diagram: Diagram 1

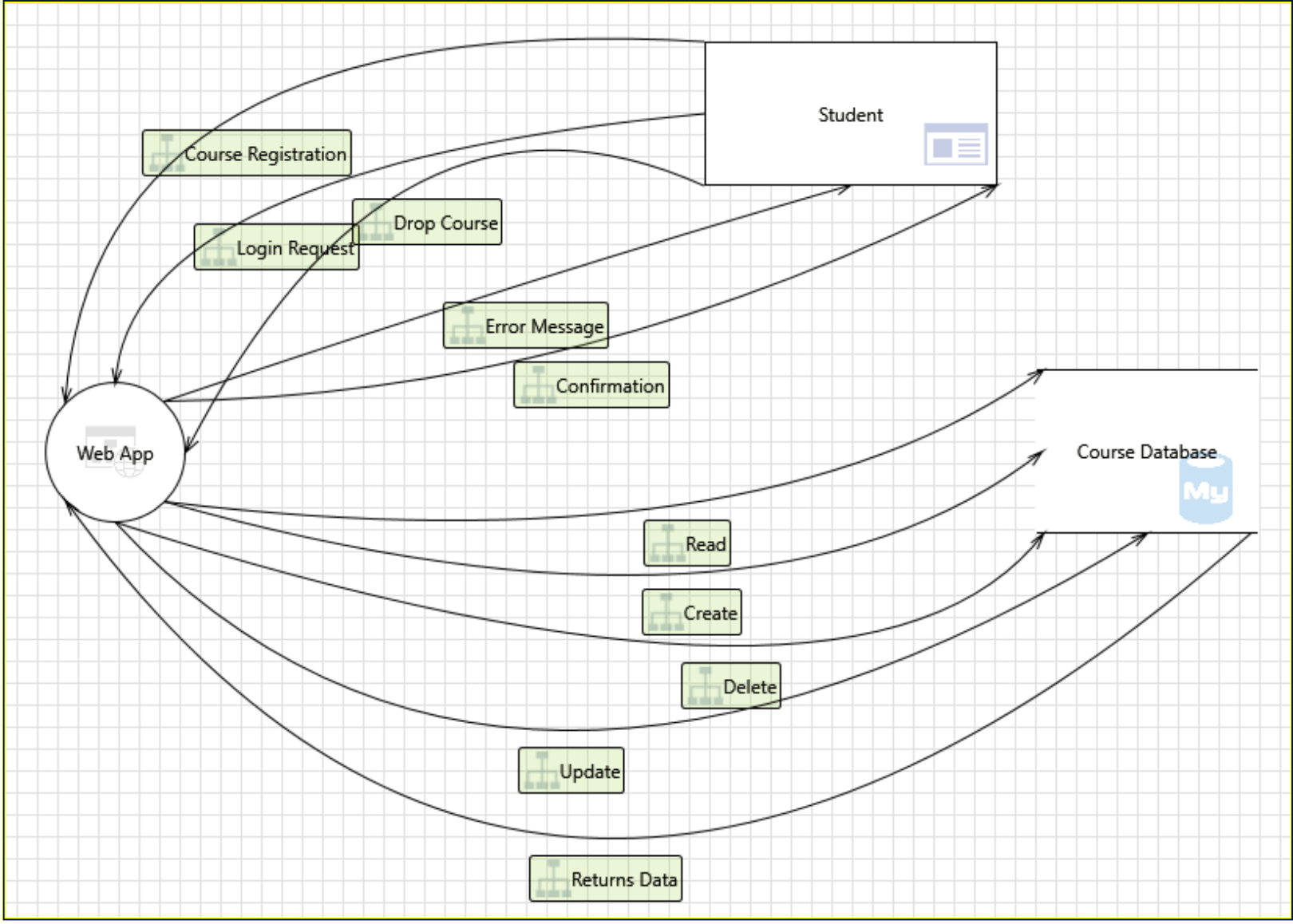
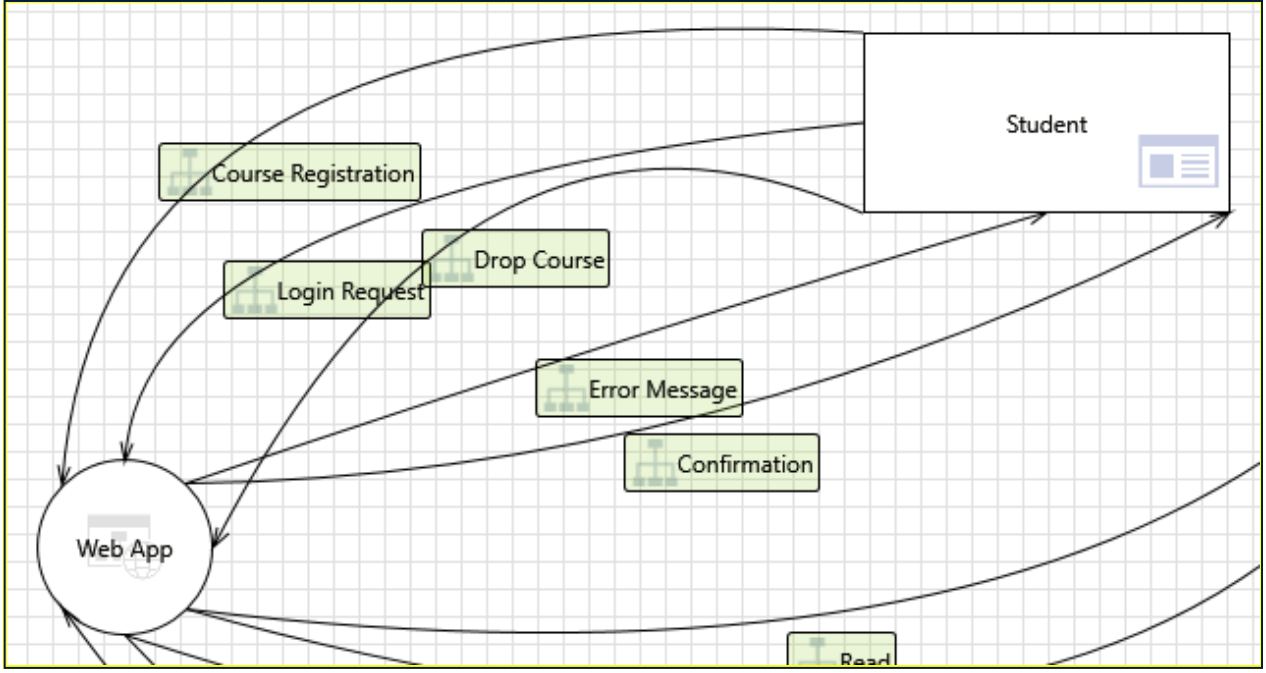


Diagram 1 Diagram Summary:

Not Started	85
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	85
Total Migrated	0

Interaction: Course Registration



1. An adversary can perform action on behalf of other user due to lack of controls against cross domain requests [State: Not Started] [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 Started] [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 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

4. 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

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):	Use Multi-Factor Authentication (2FA) for login. 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 Student 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: 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]

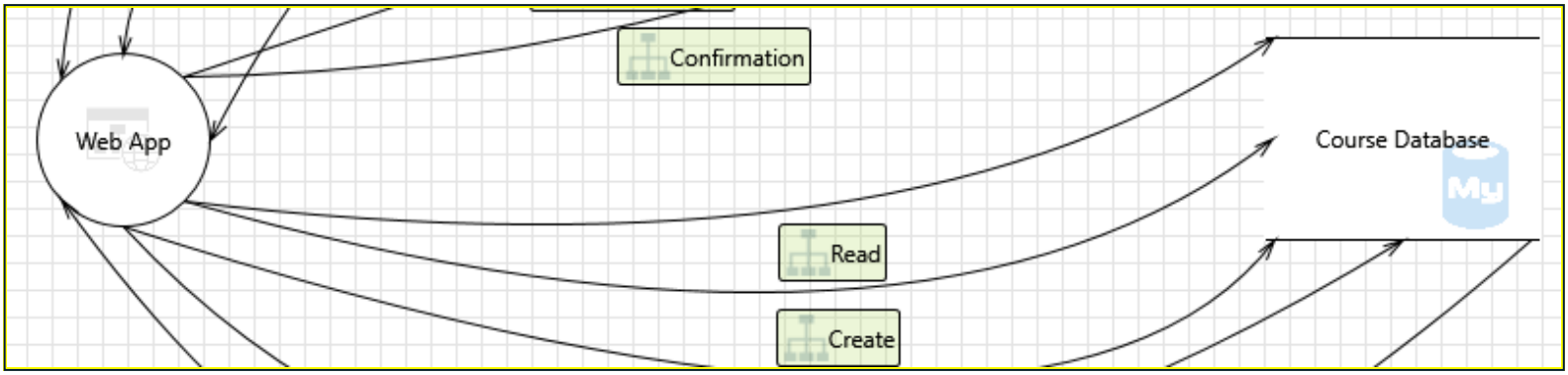
Category:	Tampering
Description:	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated 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: Create



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

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Course Database instances due to weak network security configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict access to Azure MySQL DB instances by configuring server-level firewall rules to only permit connections from selected IP addresses where possible. Refer: https://aka.ms/tmt-th150
SDL Phase:	Implementation

23. An adversary may read and/or tamper with the data transmitted to Azure MySQL DB due to weak configuration [State: Not Started] [Priority: High]

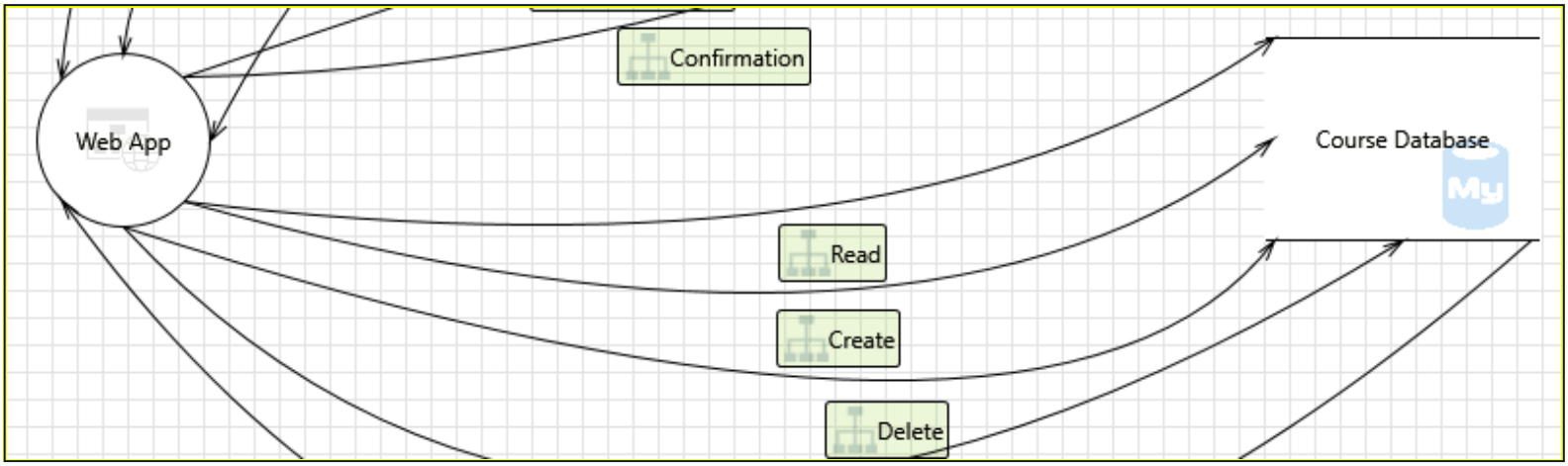
Category:	Tampering
Description:	An adversary may read and/or tamper with the data transmitted to Course Database due to weak configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enforce communication between clients and Azure MySQL DB to be over SSL/TLS by enabling the Enforce SSL connection feature on the server. Check that the connection strings used to connect to MySQL databases have the right configuration (e.g. ssl = true or sslmode=require or sslmode=true are set). Refer: https://aka.ms/tmt-th151a Configure MySQL server to use a verifiable SSL certificate (needed for SSL/TLS communication). Refer: https://aka.ms/tmt-th151b
SDL Phase:	Implementation

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

Category:	Elevation of Privileges
Description:	An adversary can gain long term, persistent access to Course Database instance through the compromise of local user account password(s).
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to rotate user account passwords (e.g. those used in connection strings) regularly, in accordance with your organization's policies. Store secrets in a secret storage solution (e.g. Azure Key Vault).

SDL Phase:	Implementation

Interaction: Delete



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

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Course Database instances due to weak network security configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict access to Azure MySQL DB instances by configuring server-level firewall rules to only permit connections from selected IP addresses where possible. Refer: https://aka.ms/tmt-th150
SDL Phase:	Implementation

26. An adversary may read and/or tamper with the data transmitted to Azure MySQL DB due to weak configuration [State: Not Started] [Priority: High]

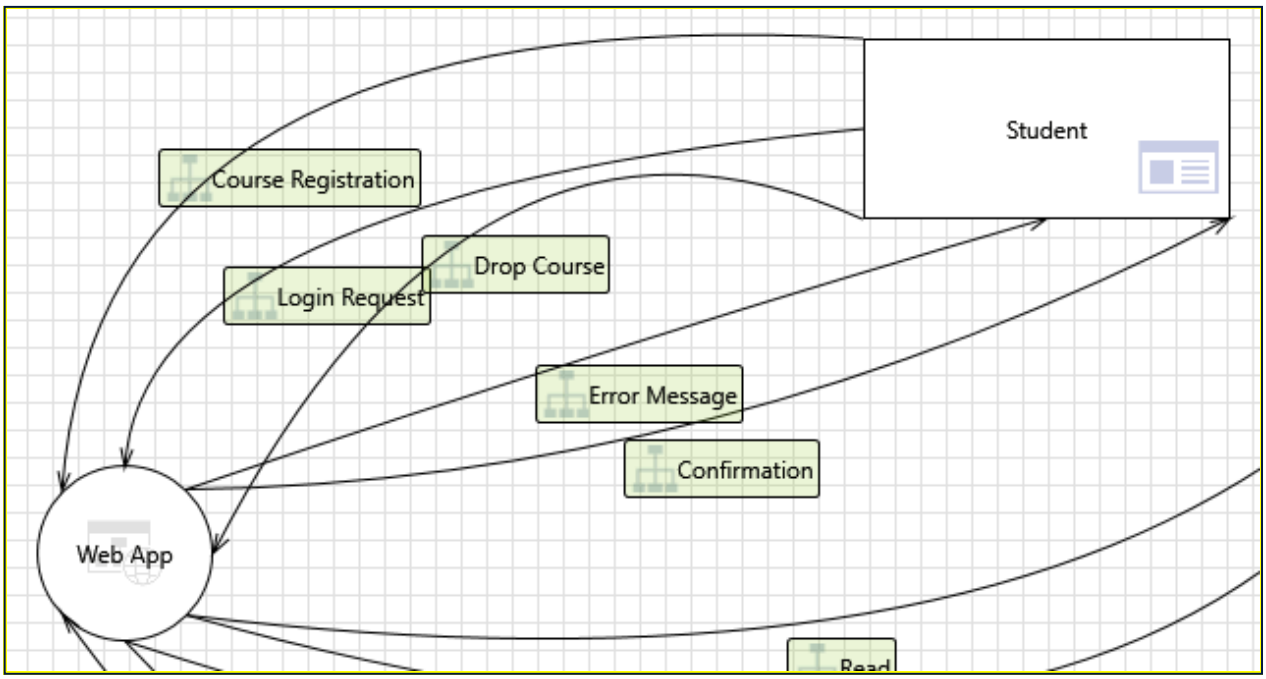
Category:	Tampering
Description:	An adversary may read and/or tamper with the data transmitted to Course Database due to weak configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enforce communication between clients and Azure MySQL DB to be over SSL/TLS by enabling the Enforce SSL connection feature on the server. Check that the connection strings used to connect to MySQL databases have the right configuration (e.g. ssl = true or sslmode=require or sslmode=true are set). Refer: https://aka.ms/tmt-th151a Configure MySQL server to use a verifiable SSL certificate (needed for SSL/TLS communication). Refer: https://aka.ms/tmt-th151b
SDL Phase:	Implementation

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

Category:	Elevation of Privileges
Description:	An adversary can gain long term, persistent access to Course Database instance through the compromise of local user account password(s).
Justification:	<no mitigation provided>

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

Interaction: Drop Course



28. 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

29. 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

30. 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

31. 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

32. 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

33. 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

34. 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

35. 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

36. 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

37. 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

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

39. An adversary can perform action on behalf of other user due to lack of controls against cross domain requests [State: Not Started] [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

40. 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

41. 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):	Use Multi-Factor Authentication (2FA) for login. 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

42. 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

43. 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

44. An adversary may spoof Student 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

45. 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: <a href="https://aka.ms/tmtconfigmgmt#js-

	trusted">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: 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

46. 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

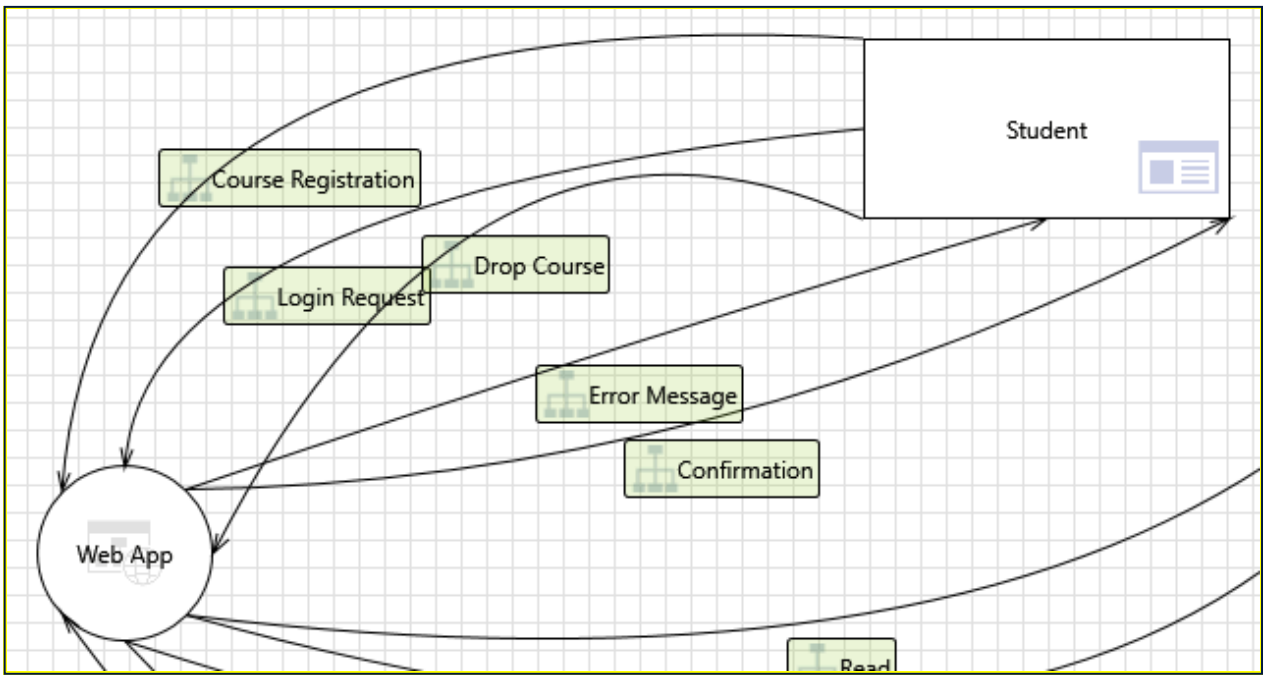
47. An adversary can gain access to sensitive data by performing SQL injection through Web App [State: Not Started] [Priority: High]

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

48. 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: Login Request



49. An adversary can perform action on behalf of other user due to lack of controls against cross domain requests [State: Not Started] [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

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

51. 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

52. 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: <a

	href="https://aka.ms/tmtauditlog#log-restricted-access">https://aka.ms/tmtauditlog#log-restricted-access
SDL Phase:	Implementation

53. 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

54. 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

55. 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

56. 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

57. 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

58. 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

59. 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

60. 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

61. 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

62. 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: <a href="https://aka.ms/tmtauthn#step-up-

	adaptive-authn">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

63. 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

64. 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):	Use Multi-Factor Authentication (2FA) for login. 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

65. An adversary may spoof Student 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

66. 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: 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

67. 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

68. 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

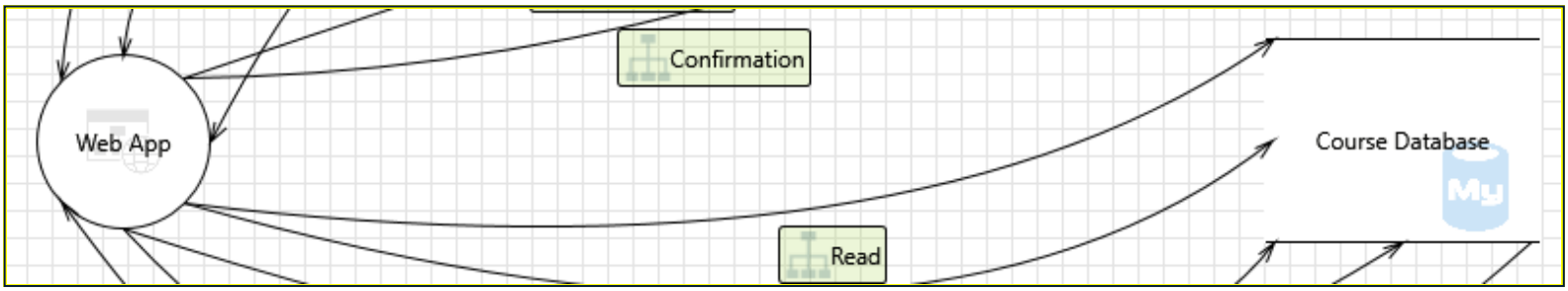
executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.

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

69. 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: Read



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

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Course Database instances due to weak network security configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Restrict access to Azure MySQL DB instances by configuring server-level firewall rules to only permit connections from selected IP addresses where possible. Refer: https://aka.ms/tmt-th150
SDL Phase:	Implementation

71. An adversary may read and/or tamper with the data transmitted to Azure MySQL DB due to weak configuration [State: Not Started] [Priority: High]

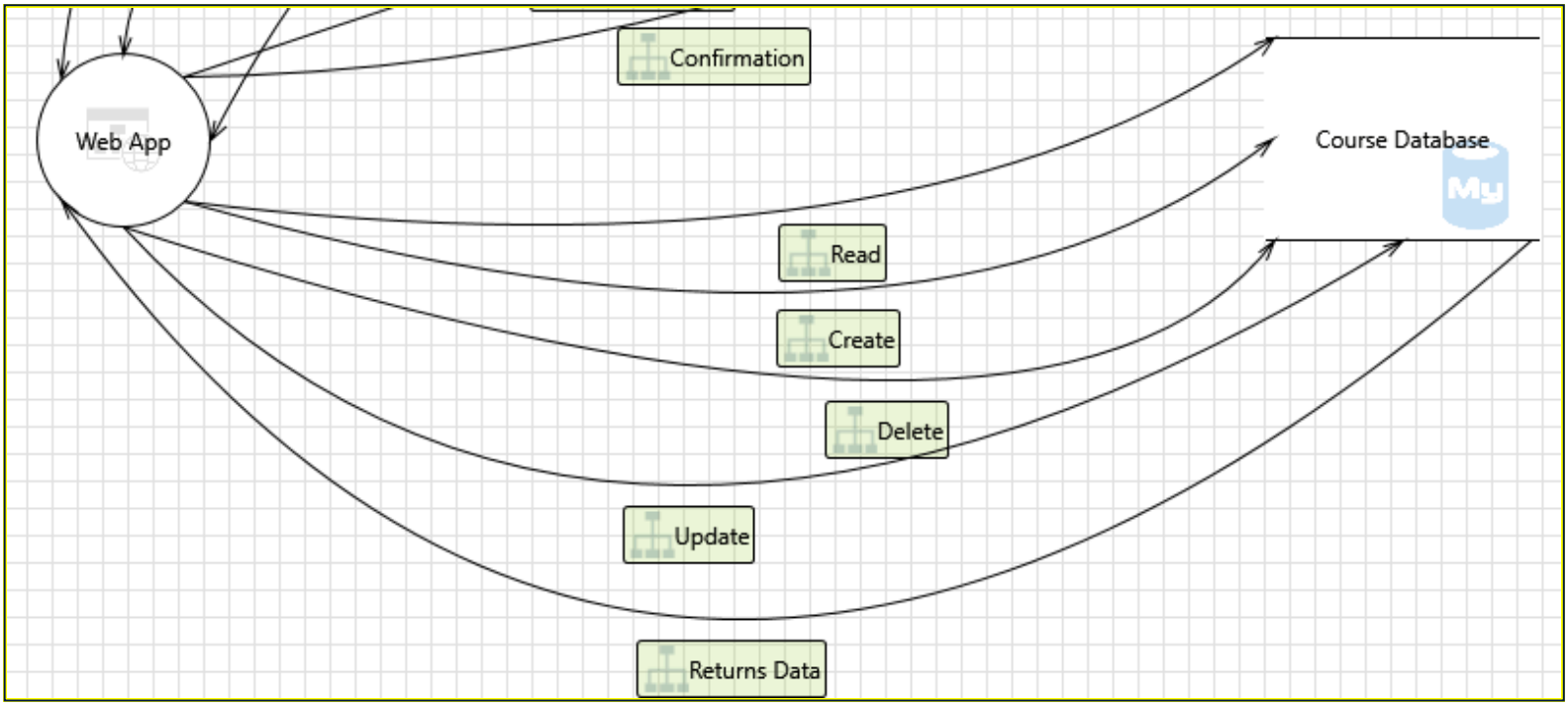
Category:	Tampering
Description:	An adversary may read and/or tamper with the data transmitted to Course Database due to weak configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enforce communication between clients and Azure MySQL DB to be over SSL/TLS by enabling the Enforce SSL connection feature on the server. Check that the connection strings used to connect to MySQL databases have the right configuration (e.g. ssl = true or sslmode=require or

	sslmode=true are set). Refer: https://aka.ms/tmt-th151a Configure MySQL server to use a verifiable SSL certificate (needed for SSL/TLS communication). Refer: https://aka.ms/tmt-th151b
SDL Phase:	Implementation

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

Category:	Elevation of Privileges
Description:	An adversary can gain long term, persistent access to Course Database instance through the compromise of local user account password(s).
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to rotate user account passwords (e.g. those used in connection strings) regularly, in accordance with your organization's policies. Store secrets in a secret storage solution (e.g. Azure Key Vault).
SDL Phase:	Implementation

Interaction: Returns Data



73. 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

74. 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

75. An adversary may spoof Course Database 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

76. 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

77. 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

78. 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

79. 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

80. 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

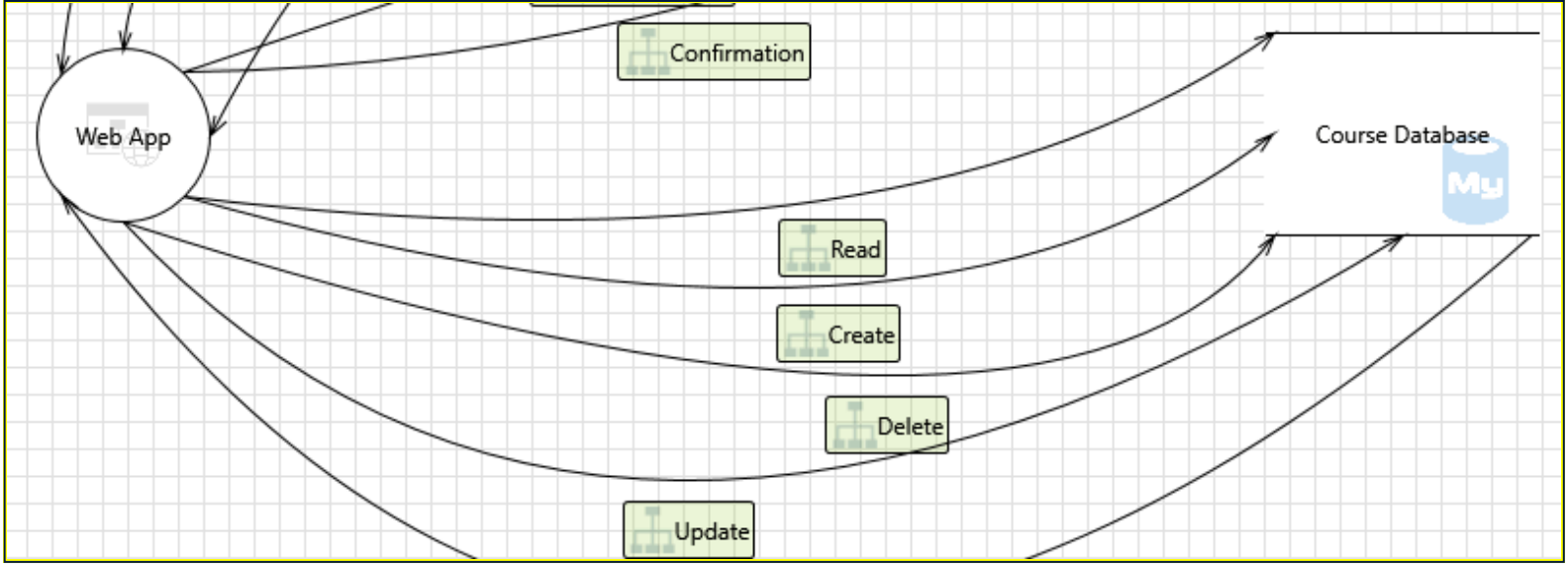
81. An adversary may gain access to sensitive data from log files [State: Not Started] [Priority: High]

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

82. 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

Interaction: Update



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

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Course Database instances due to weak network security configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Implement role-based access control and input validation Restrict access to Azure MySQL DB instances by configuring server-level firewall rules to only permit connections from selected IP addresses where possible. Refer: https://aka.ms/tmt-th150
SDL Phase:	Implementation

84. An adversary may read and/or tamper with the data transmitted to Azure MySQL DB due to weak configuration [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary may read and/or tamper with the data transmitted to Course Database due to weak configuration.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Enforce communication between clients and Azure MySQL DB to be over SSL/TLS by enabling the Enforce SSL connection feature on the server. Check that the connection strings used to connect to MySQL databases have the right configuration (e.g. ssl = true or sslmode=require or sslmode=true are set). Refer: https://aka.ms/tmt-th151a Configure MySQL server to use a verifiable SSL certificate (needed for SSL/TLS communication). Refer: https://aka.ms/tmt-th151b
SDL Phase:	Implementation

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

Category:	Elevation of Privileges
Description:	An adversary can gain long term, persistent access to Course Database instance through the compromise of local user account password(s).
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to rotate user account passwords (e.g. those used in connection strings) regularly, in accordance with your organization's policies. Store secrets in a secret storage solution (e.g. Azure Key Vault).

SDL Phase:	Implementation