Threat Modeling Report

Created on 5/11/2023 8:10:10 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	195
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	195
Total Migrated	0

Diagram: Diagram 1

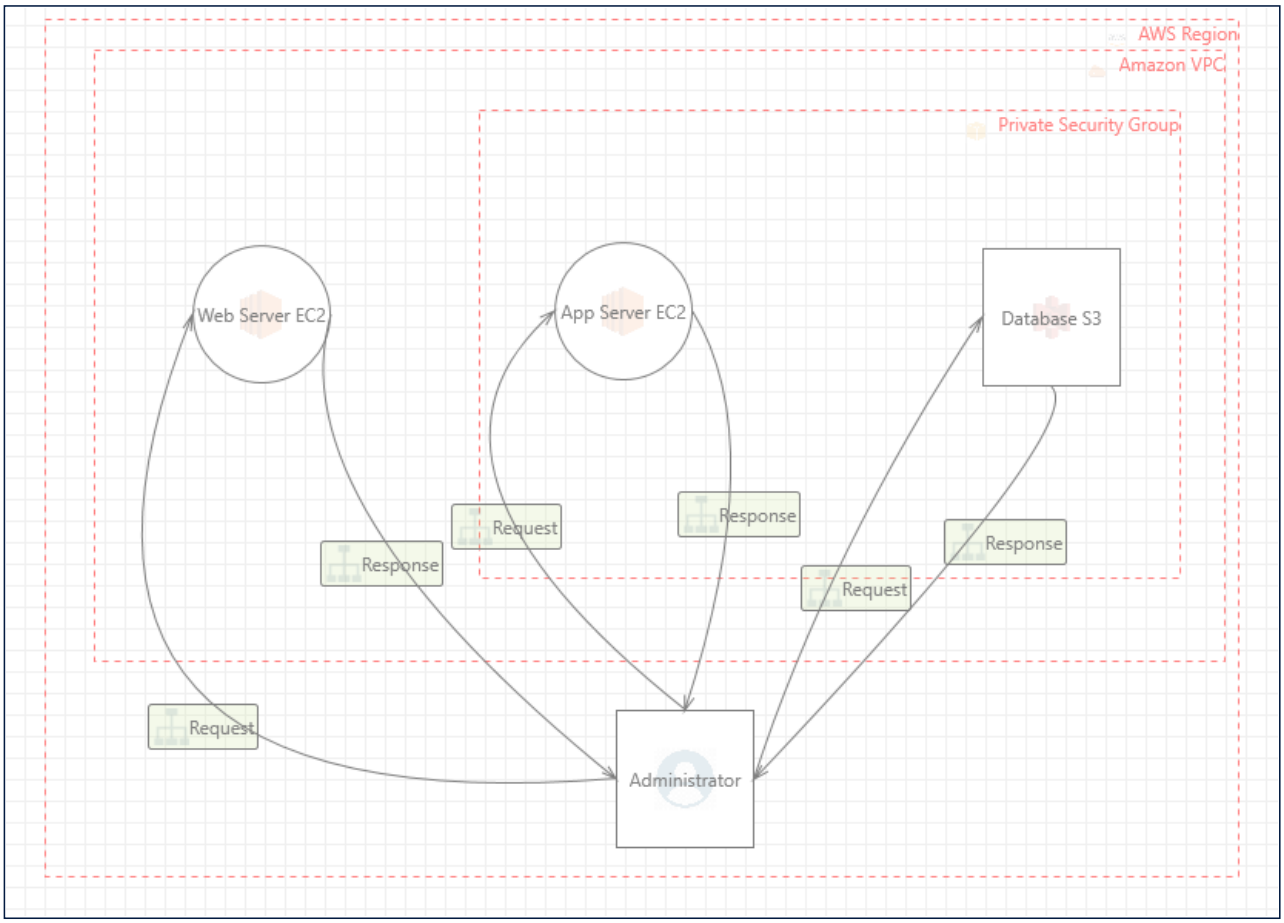
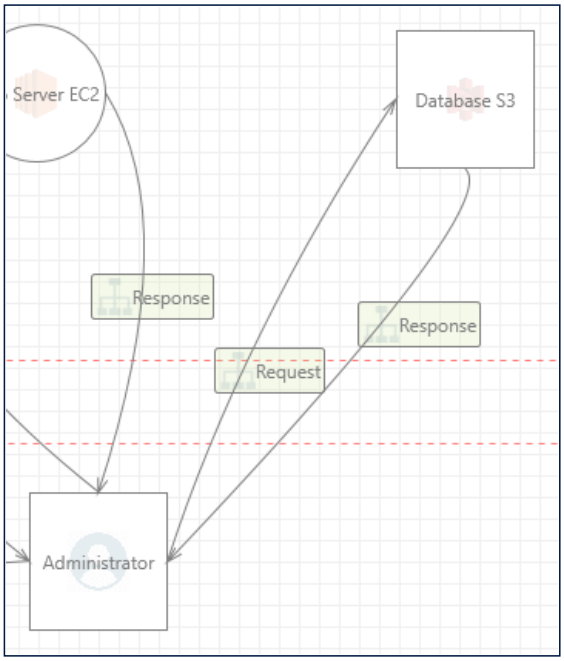


Diagram 1 Diagram Summary:

Not Started	88
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	88
Total Migrated	0

Interaction: Request



1. AWS Storage encryption [State: Not Started] [Priority: High]

Category:	Storage
Description:	Ensure AWS Storage sensitive information encrypted at Rest to prevent data leakage, unauthorized access or compromise
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure the sensitive information is encrypted in Database S3 2. Ensure encryption keys stored in AWS Key Management Service (AWS KMS)
Comments:	

2. Amazon Macie usage [State: Not Started] [Priority: High]

Category:	Storage
Description:	Usage of Amazon Macie allows to prevent sensitive data (such as PII, access or API keys) leakage and unauthoried access on S3 bucket level
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that AWS Amazon Macier is used for Database S3 scan and its report reviewed on a regular basis
Comments:	

3. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists
Comments:	

4. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege
Comments:	

5. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis
Comments:	

6. AWS WAF [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

7. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

8. Security groups best practices [State: Not Started] [Priority: High]

Category: Security Boundary

Description: Ensure VPC Security groups is configured correctly to segregate access to various VPC resources to minimize attack surface and to set different rules for different classes of instances

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: Example of correct configuration: 1. The group for the web servers would have only port 443 (HTTPS) open to the Internet 2. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 3. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 4. All three groups would permit administrative access on port 22 (SSH), but only from Corporate network

Comments:

9. An adversary may try to access, modify, create or delete information stored in Amazon S3 [State: Not Started] [Priority: High]

Category: Denial of Service

Description: An adversary may try to access, modify, create or delete information stored in Amazon S3

Justification: <no mitigation provided>

Possible Mitigation(s):

SDL Phase: Design

Action Items:

Comments:

10. An adversary can gain unauthorized access to Database S3 due to weak access control restrictions [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description: An adversary can gain unauthorized access to Database S3 due to weak access control restrictions

Justification: <no mitigation provided>

Possible Mitigation(s): Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b

SDL Phase: Implementation

Action Items:

Comments:

11. A compromised access key may permit an adversary to have more access than intended to an Database S3 instance [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	A compromised access key may permit an adversary to have over-privileged access to an Database S3 instance
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use resource (SAS like) tokens (derived using master keys) to connect to Cosmos DB instances whenever possible. Scope the resource tokens to permit only the privileges necessary (e.g. read-only). Store secrets in a secret storage solution (e.g. Azure Key Vault). Refer: https://aka.ms/tmt-th54
SDL Phase:	Design
Action Items:	
Comments:	

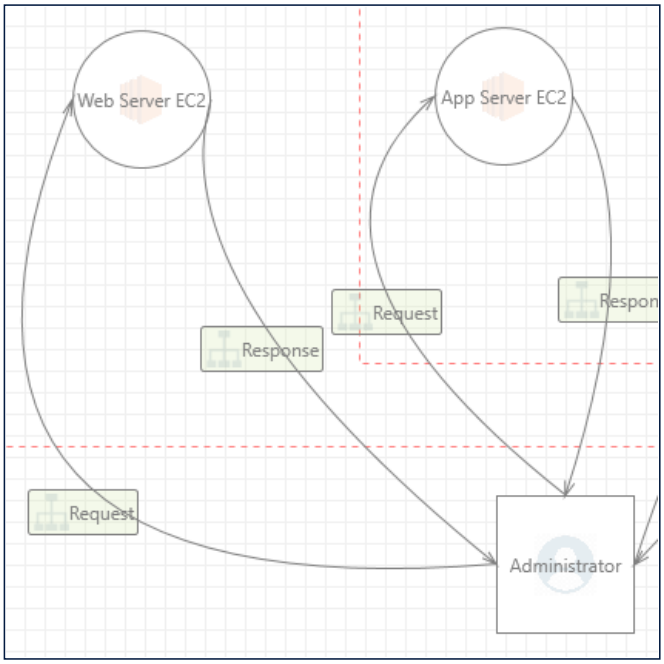
12. A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

13. A malicious user can deny they made a change to Database S3 [State: Not Started] [Priority: High]

Category:	Repudiation
Description:	This is due to the Last Modified By field being overwritten on each save
Justification:	<no mitigation provided>
Possible Mitigation(s):	Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities
SDL Phase:	Design
Action Items:	
Comments:	

Interaction: Request



14. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category:	Compute
Description:	Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:	

15. Amazon EC2 API Gateway [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Dont expose EC2 to the Internet directly and if exposed only allow invocation through API Gateway
Comments:	

16. EC2 Metadata protection [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design

Action Items: 1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance

Comments:

17. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category: Network Boundary

Description: In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists

Comments:

18. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category: Network Boundary

Description: When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege

Comments:

19. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis

Comments:

20. AWS WAF [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

21. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

22. SSH Port Exposure [State: Not Started] [Priority: High]

Category: Compute

Description: SSH Port 22 is Open to the Internet

Justification: <no mitigation provided>

Possible Mitigation(s): Close SSH Port 22

SDL Phase: Design

Action Items:

Comments:

23. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category: Compute

Description: HTTP Port 80 is Open to the Internet

Justification: <no mitigation provided>

Possible Mitigation(s): Consider placing Firewalls to block unwanted traffic.

SDL Phase: Design

Action Items:

Comments:

24. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category: Compute

Description: HTTPS Port 443 Open

Justification: <no mitigation provided>

Possible Mitigation(s): Consider placing a Firewall to block unwanted traffic.

SDL Phase: Design

Action Items:

Comments:

25. An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack
Justification:	<no mitigation provided>
Possible Mitigation(s):	Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c
SDL Phase:	Implementation
Action Items:	
Comments:	

26. An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions
Justification:	<no mitigation provided>
Possible Mitigation(s):	Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase:	Implementation
Action Items:	
Comments:	

27. An adversary may gain unauthorized access to privileged features on Administrator [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

29. 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

Action Items:

Comments:

30. 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

Action Items:

Comments:

31. An adversary having access to Web Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary having access to Web Server EC2 may read sensitive clear-text data

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.

SDL Phase: Design

Action Items:

Comments:

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

Action Items:

Comments:

33. A malicious user can deny they made a change to Web Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation

Description: This is due to the Last Modified By field being overwritten on each save

Justification: <no mitigation provided>

Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities

SDL Phase: Design

Action Items:

Comments:

34. 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

Action Items:

Comments:

35. 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: <a href="https://aka.ms/tmtconfigmgmt#xss-

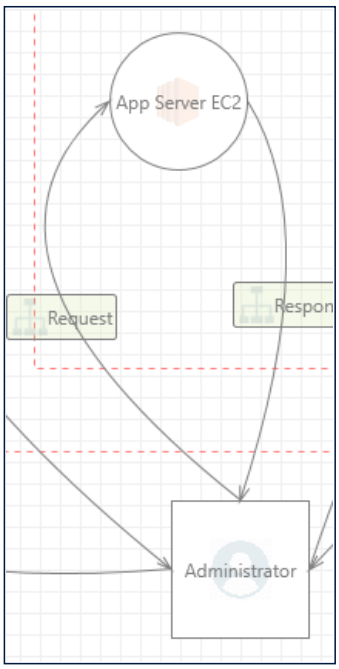
filter">https://aka.ms/tmtconfigmgmt#xss-filter Access third party javascripts from trusted sources only. Refer: https://aka.ms/tmtconfigmgmt#js-trusted Enable ValidateRequest attribute on ASP.NET Pages. Refer: https://aka.ms/tmtconfigmgmt#validate-aspnet Ensure that each page that could contain user 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

Action Items:

Comments:

Interaction: Request



36. EC2 Metadata protection [State: Not Started] [Priority: High]

Category: Compute

Description: Don’t expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with “Authentication as a Service” providers like Cognito or Okta.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance

Comments:

37. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category: Compute

Description: Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis

Comments:

38. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category: Network Boundary

Description: In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists

Comments:

39. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category: Network Boundary

Description: When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege

Comments:

40. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis

Comments:

41. AWS WAF [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

42. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

43. Security groups best practices [State: Not Started] [Priority: High]

Category: Security Boundary

Description: Ensure VPC Security groups is configured correctly to segregate access to various VPC resources to minimize attack surface and to set different rules for different classes of instances

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: Example of correct configuration: 1. The group for the web servers would have only port 443 (HTTPS) open to the Internet 2. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 3. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 4. All three groups would permit administrative access on port 22 (SSH), but only from Corporate network

Comments:

44. SSH Port Exposure [State: Not Started] [Priority: High]

Category: Compute

Description: SSH Port 22 is Open to the Internet
Justification: <no mitigation provided>
Possible Mitigation(s): Close SSH Port 22
SDL Phase: Design
Action Items:
Comments:

45. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category: Compute
Description: HTTP Port 80 is Open to the Internet
Justification: <no mitigation provided>
Possible Mitigation(s): Consider placing Firewalls to block unwanted traffic.
SDL Phase: Design
Action Items:
Comments:

46. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category: Compute
Description: HTTPS Port 443 Open
Justification: <no mitigation provided>
Possible Mitigation(s): Consider placing a Firewall to block unwanted traffic.
SDL Phase: Design
Action Items:
Comments:

47. An adversary may block access to the application or API hosted on App Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category: Denial of Service
Description: An adversary may block access to the application or API hosted on App Server EC2 through a denial of service attack
Justification: <no mitigation provided>
Possible Mitigation(s): Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c
SDL Phase: Implementation
Action Items:
Comments:

48. An adversary can gain unauthorized access to App Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category: Elevation of Privileges
Description: An adversary can gain unauthorized access to App Server EC2 due to weak access control restrictions
Justification: <no mitigation provided>
Possible Mitigation(s): Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase: Implementation

Action Items:

Comments:

49. An adversary may gain unauthorized access to privileged features on Administrator [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

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
Action Items:	
Comments:	

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: https://aka.ms/tmtauditlog#log-restricted-access
SDL Phase: Implementation
Action Items:
Comments:

53. An adversary having access to App Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: An adversary having access to App Server EC2 may read sensitive clear-text data
Justification: <no mitigation provided>
Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.
SDL Phase: Design
Action Items:
Comments:

54. 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
Action Items:
Comments:

55. A malicious user can deny they made a change to App Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation
Description: This is due to the Last Modified By field being overwritten on each save
Justification: <no mitigation provided>
Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities
SDL Phase: Design
Action Items:
Comments:

56. 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

Action Items:

Comments:

57. 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>

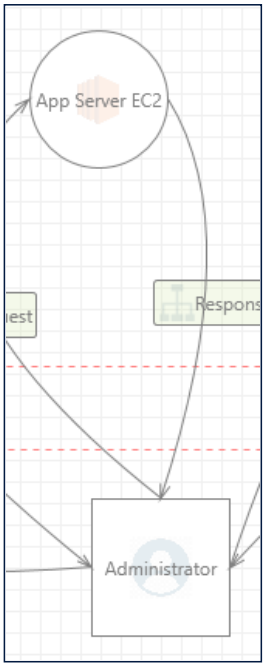
Possible Mitigation(s): Implement Content Security Policy (CSP), and disable inline javascript. Refer: https://aka.ms/tmtconfigmgmt#csp-js Enable browser's XSS filter. Refer: https://aka.ms/tmtconfigmgmt#xss-filter Access third party javascripts from trusted sources only. Refer: https://aka.ms/tmtconfigmgmt#js-trusted 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

Action Items:

Comments:

Interaction: Response



58. EC2 Metadata protection [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance
Comments:	

59. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category:	Compute
Description:	Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:	

60. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design

Action Items: 1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists

Comments:

61. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category: Network Boundary

Description: When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege

Comments:

62. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis

Comments:

63. AWS WAF [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

64. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and

can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

65. Security groups best practices [State: Not Started] [Priority: High]

Category: Security Boundary

Description: Ensure VPC Security groups is configured correctly to segregate access to various VPC resources to minimize attack surface and to set different rules for different classes of instances

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: Example of correct configuration: 1. The group for the web servers would have only port 443 (HTTPS) open to the Internet 2. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 3. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 4. All three groups would permit administrative access on port 22 (SSH), but only from Corporate network

Comments:

66. SSH Port Exposure [State: Not Started] [Priority: High]

Category: Compute

Description: SSH Port 22 is Open to the Internet

Justification: <no mitigation provided>

Possible Mitigation(s): Close SSH Port 22

SDL Phase: Design

Action Items:

Comments:

67. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category: Compute

Description: HTTP Port 80 is Open to the Internet

Justification: <no mitigation provided>

Possible Mitigation(s): Consider placing Firewalls to block unwanted traffic.

SDL Phase: Design

Action Items:

Comments:

68. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category: Compute

Description: HTTPS Port 443 Open

Justification: <no mitigation provided>

Possible Mitigation(s): Consider placing a Firewall to block unwanted traffic.

SDL Phase: Design

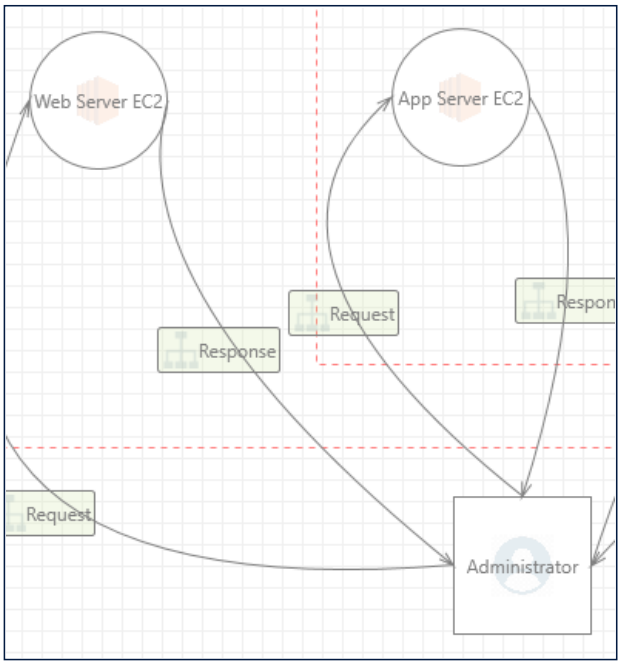
Action Items:

Comments:

69. 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
Action Items:	
Comments:	

Interaction: Response



70. EC2 Metadata protection [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don’t expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with “Authentication as a Service” providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you’re using in every EC2 instance
Comments:	

71. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category:	Compute
Description:	Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:	

72. Amazon EC2 API Gateway [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Dont expose EC2 to the Internet directly and if exposed only allow invocation through API Gateway
Comments:	

73. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists
Comments:	

74. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege
Comments:	

75. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis
Comments:	

76. AWS WAF [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port
Comments:	

77. AWS Shield usage [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html
Comments:	

78. SSH Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	SSH Port 22 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Close SSH Port 22
SDL Phase:	Design
Action Items:	
Comments:	

79. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category:	Compute
Description:	HTTP Port 80 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Consider placing Firewalls to block unwanted traffic.
SDL Phase:	Design
Action Items:	
Comments:	

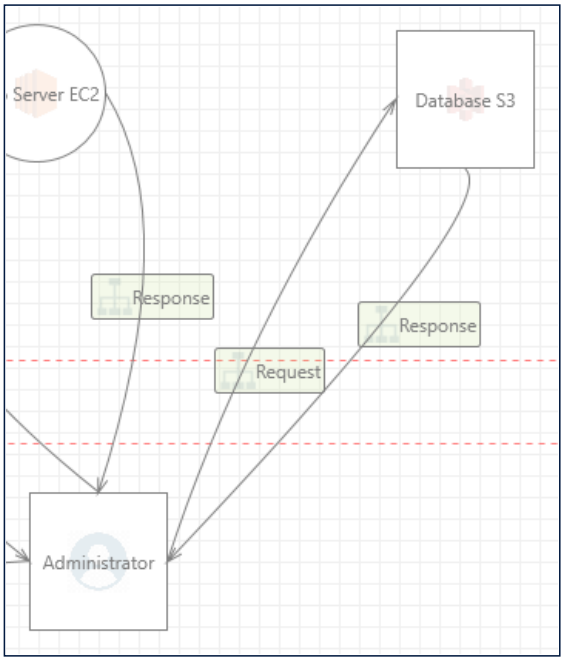
80. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	HTTPS Port 443 Open
Justification:	<no mitigation provided>
Possible Mitigation(s):	Consider placing a Firewall to block unwanted traffic.
SDL Phase:	Design
Action Items:	
Comments:	

81. 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
Action Items:	
Comments:	

Interaction: Response



82. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists
Comments:	

83. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege
Comments:	

84. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category:	Network Boundary
Description:	AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis

Comments:

85. AWS WAF [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

86. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

87. Security groups best practices [State: Not Started] [Priority: High]

Category: Security Boundary

Description: Ensure VPC Security groups is configured correctly to segregate access to various VPC resources to minimize attack surface and to set different rules for different classes of instances

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: Example of correct configuration: 1. The group for the web servers would have only port 443 (HTTPS) open to the Internet 2. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 3. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 4. All three groups would permit administrative access on port 22 (SSH), but only from Corporate network

Comments:

88. 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
Action Items:	
Comments:	

Diagram: Diagram 2

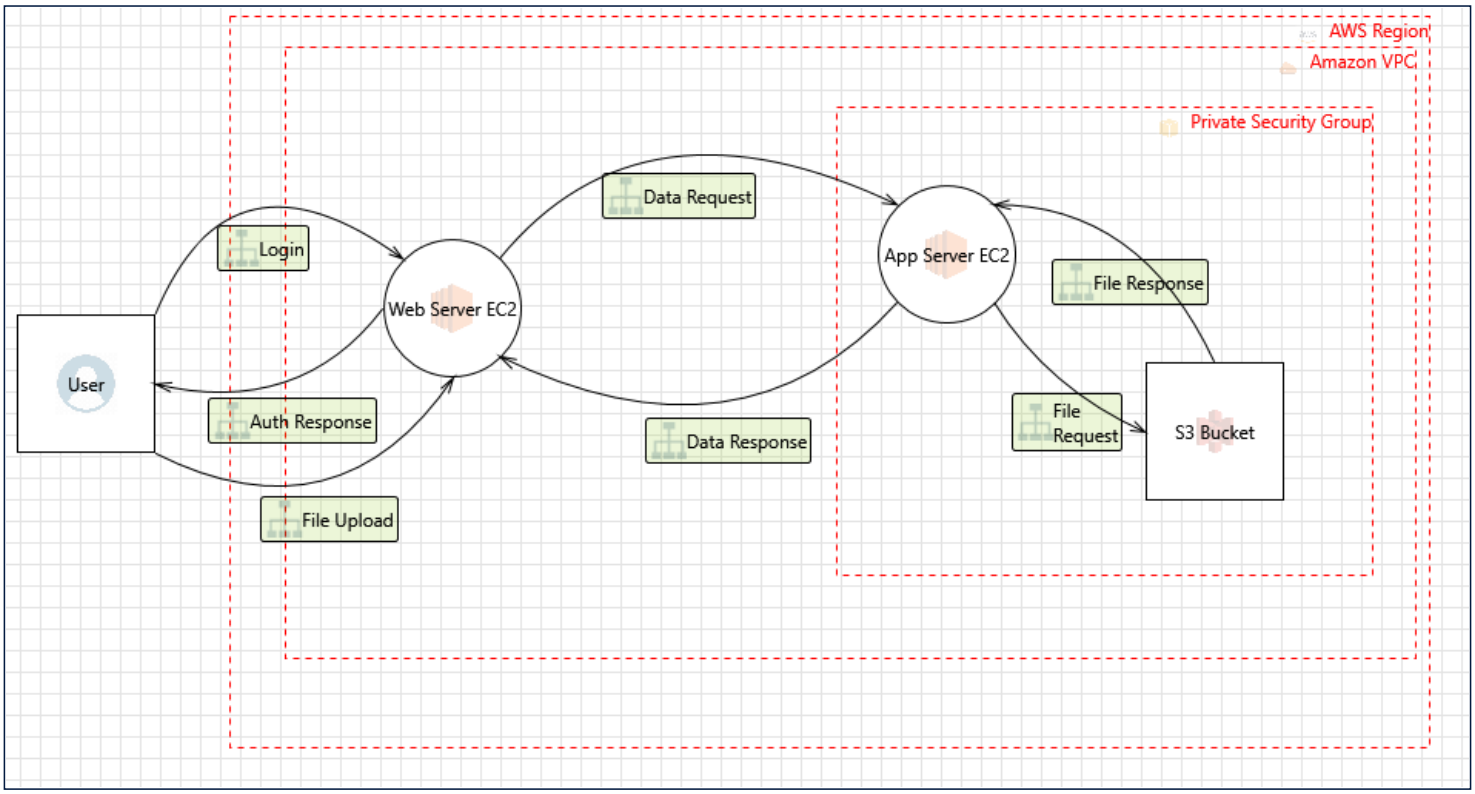
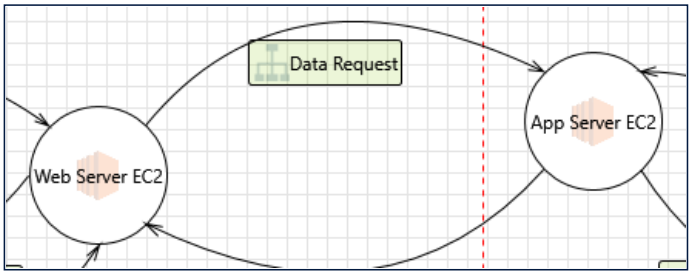


Diagram 2 Diagram Summary:

Not Started	107
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	107
Total Migrated	0

Interaction: Data Request



89. EC2 Metadata protection [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance
Comments:	

90. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category:	Compute
Description:	Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:	

91. Amazon EC2 API Gateway [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Dont expose EC2 to the Internet directly and if exposed only allow invocation through API Gateway
Comments:	

92. SSH Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	SSH Port 22 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Close SSH Port 22
SDL Phase:	Design

Action Items:

Comments:

93. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category:	Compute
Description:	HTTP Port 80 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Consider placing Firewalls to block unwanted traffic.
SDL Phase:	Design
Action Items:	
Comments:	

94. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	HTTPS Port 443 Open
Justification:	<no mitigation provided>
Possible Mitigation(s):	Consider placing a Firewall to block unwanted traffic.
SDL Phase:	Design
Action Items:	
Comments:	

95. Security groups best practices [State: Not Started] [Priority: High]

Category:	Security Boundary
Description:	Ensure VPC Security groups is configured correctly to segregate access to various VPC resources to minimize attack surface and to set different rules for different classes of instances
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	Example of correct configuration: 1. The group for the web servers would have only port 443 (HTTPS) open to the Internet 2. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 3. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 4. All three groups would permit administrative access on port 22 (SSH), but only from Corporate network
Comments:	

96. An adversary may block access to the application or API hosted on App Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	An adversary may block access to the application or API hosted on App Server EC2 through a denial of service attack
Justification:	<no mitigation provided>
Possible Mitigation(s):	Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c
SDL Phase:	Implementation
Action Items:	
Comments:	

97. An adversary can gain unauthorized access to App Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to App Server EC2 due to weak access control restrictions
Justification:	<no mitigation provided>
Possible Mitigation(s):	Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase:	Implementation
Action Items:	
Comments:	

98. An adversary may gain unauthorized access to privileged features on Web Server EC2 [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

100. 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: <a href="https://aka.ms/tmtcrypto#mac-

hash">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

Action Items:

Comments:

101. 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

Action Items:

Comments:

102. An adversary having access to App Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary having access to App Server EC2 may read sensitive clear-text data

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.

SDL Phase: Design

Action Items:

Comments:

103. 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

Action Items:

Comments:

104. A malicious user can deny they made a change to App Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation

Description: This is due to the Last Modified By field being overwritten on each save

Justification: <no mitigation provided>

Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities

SDL Phase: Design

Action Items:

Comments:

105. 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

Action Items:

Comments:

106. 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

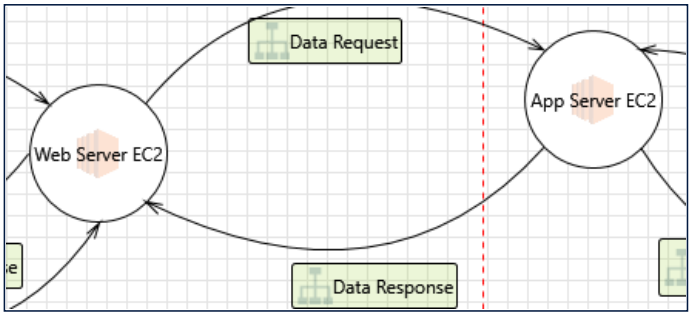
Controller methods. Refer: https://aka.ms/tmtinputval#string-method Avoid using Html.Raw in Razor views. Refer: https://aka.ms/tmtinputval#html-razor Sanitization should be applied on form fields that accept all characters e.g, rich text editor . Refer: https://aka.ms/tmtinputval#richtext Do not assign DOM elements to sinks that do not have inbuilt encoding . Refer: https://aka.ms/tmtinputval#inbuilt-encode

SDL Phase: Implementation

Action Items:

Comments:

Interaction: Data Response



107. EC2 Metadata protection [State: Not Started] [Priority: High]

Category: Compute

Description: Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance

Comments:

108. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category: Compute

Description: Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis

Comments:

109. Amazon EC2 API Gateway [State: Not Started] [Priority: High]

Category: Compute

Description: Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.

Justification: Security Requirement

Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Dont expose EC2 to the Internet directly and if exposed only allow invocation through API Gateway
Comments:

110. SSH Port Exposure [State: Not Started] [Priority: High]

Category: Compute
Description: SSH Port 22 is Open to the Internet
Justification: <no mitigation provided>
Possible Mitigation(s): Close SSH Port 22
SDL Phase: Design
Action Items:
Comments:

111. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category: Compute
Description: HTTP Port 80 is Open to the Internet
Justification: <no mitigation provided>
Possible Mitigation(s): Consider placing Firewalls to block unwanted traffic.
SDL Phase: Design
Action Items:
Comments:

112. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category: Compute
Description: HTTPS Port 443 Open
Justification: <no mitigation provided>
Possible Mitigation(s): Consider placing a Firewall to block unwanted traffic.
SDL Phase: Design
Action Items:
Comments:

113. Security groups best practices [State: Not Started] [Priority: High]

Category: Security Boundary
Description: Ensure VPC Security groups is configured correctly to segregate access to various VPC resources to minimize attack surface and to set different rules for different classes of instances
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: Example of correct configuration: 1. The group for the web servers would have only port 443 (HTTPS) open to the Internet 2. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 3. The group for the application servers would have port 8000 (application specific) accessible only to the web server group 4. All three groups would permit administrative access on port 22 (SSH), but only from Corporate network
Comments:

114. An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack
Justification:	<no mitigation provided>
Possible Mitigation(s):	Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c
SDL Phase:	Implementation
Action Items:	
Comments:	

115. An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions
Justification:	<no mitigation provided>
Possible Mitigation(s):	Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase:	Implementation
Action Items:	
Comments:	

116. An adversary may gain unauthorized access to privileged features on App Server EC2 [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Action Items:

Comments:

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

Action Items:

Comments:

120. An adversary having access to Web Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary having access to Web Server EC2 may read sensitive clear-text data

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.

SDL Phase: Design

Action Items:

Comments:

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

Action Items:

Comments:

122. A malicious user can deny they made a change to Web Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation

Description: This is due to the Last Modified By field being overwritten on each save

Justification: <no mitigation provided>

Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities

SDL Phase: Design

Action Items:

Comments:

123. 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

Action Items:

Comments:

124. 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: <a href="https://aka.ms/tmtconfigmgmt#xss-

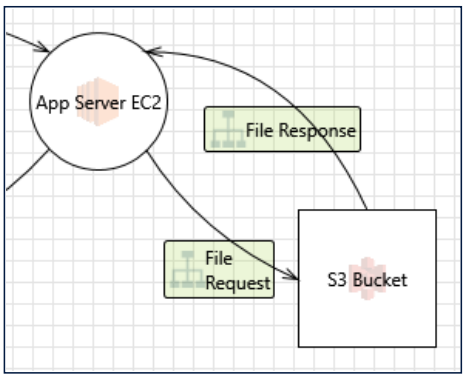
filter">https://aka.ms/tmtconfigmgmt#xss-filter Access third party javascripts from trusted sources only. Refer: https://aka.ms/tmtconfigmgmt#js-trusted Enable ValidateRequest attribute on ASP.NET Pages. Refer: https://aka.ms/tmtconfigmgmt#validate-aspnet Ensure that each page that could contain user 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

Action Items:

Comments:

Interaction: File Request



125. AWS Storage encryption [State: Not Started] [Priority: High]

Category: Storage

Description: Ensure AWS Storage sensitive information encrypted at Rest to prevent data leakage, unauthorized access or compromise

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure the sensitive information is encrypted in Database S3 2. Ensure encryption keys stored in AWS Key Management Service (AWS KMS)

Comments:

126. Amazon Macie usage [State: Not Started] [Priority: High]

Category: Storage

Description: Usage of Amazon Macie allows to prevent sensitive data (such as PII, access or API keys) leakage and unauthoried access on S3 bucket level
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure that AWS Amazon Macier is used for Database S3 scan and its report reviewed on a regular basis
Comments:

127. EC2 Metadata protection [State: Not Started] [Priority: High]

Category: Compute
Description: Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance
Comments:

128. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category: Compute
Description: Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:

129. SSH Port Exposure [State: Not Started] [Priority: High]

Category: Compute
Description: SSH Port 22 is Open to the Internet
Justification: <no mitigation provided>
Possible Mitigation(s): Close SSH Port 22
SDL Phase: Design
Action Items:
Comments:

130. An adversary may try to access, modify, create or delete information stored in Amazon S3 [State: Not Started] [Priority: High]

Category: Denial of Service
Description: An adversary may try to access, modify, create or delete information stored in Amazon S3
Justification: <no mitigation provided>
Possible Mitigation(s):
SDL Phase: Design
Action Items:
Comments:

131. An adversary can gain unauthorized access to S3 Bucket due to weak access control restrictions [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to S3 Bucket due to weak access control restrictions
Justification:	<no mitigation provided>
Possible Mitigation(s):	Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase:	Implementation
Action Items:	
Comments:	

132. A compromised access key may permit an adversary to have more access than intended to an S3 Bucket instance [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	A compromised access key may permit an adversary to have over-privileged access to an S3 Bucket instance
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use resource (SAS like) tokens (derived using master keys) to connect to Cosmos DB instances whenever possible. Scope the resource tokens to permit only the privileges necessary (e.g. read-only). Store secrets in a secret storage solution (e.g. Azure Key Vault). Refer: https://aka.ms/tmt-th54
SDL Phase:	Design
Action Items:	
Comments:	

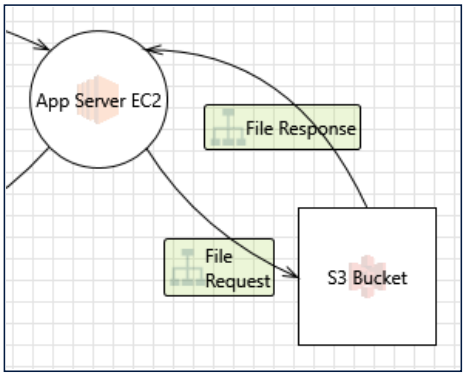
133. A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

134. A malicious user can deny they made a change to S3 Bucket [State: Not Started] [Priority: High]

Category:	Repudiation
Description:	This is due to the Last Modified By field being overwritten on each save
Justification:	<no mitigation provided>
Possible Mitigation(s):	Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities
SDL Phase:	Design
Action Items:	
Comments:	

Interaction: File Response



135. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category:	Compute
Description:	Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:	

136. EC2 Metadata protection [State: Not Started] [Priority: High]

Category:	Compute
Description:	Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification:	Security Requirement
Possible Mitigation(s):	
SDL Phase:	Design
Action Items:	1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance
Comments:	

137. SSH Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	SSH Port 22 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Close SSH Port 22
SDL Phase:	Design
Action Items:	
Comments:	

138. An adversary may block access to the application or API hosted on App Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	An adversary may block access to the application or API hosted on App Server EC2 through a denial of service attack
Justification:	<no mitigation provided>

Possible Mitigation(s):	Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c
SDL Phase:	Implementation
Action Items:	
Comments:	

139. An adversary can gain unauthorized access to App Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to App Server EC2 due to weak access control restrictions
Justification:	<no mitigation provided>
Possible Mitigation(s):	Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase:	Implementation
Action Items:	
Comments:	

140. An adversary may gain unauthorized access to privileged features on S3 Bucket [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

142. 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

Action Items:

Comments:

143. 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

Action Items:

Comments:

144. An adversary having access to App Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: An adversary having access to App Server EC2 may read sensitive clear-text data

Justification: <no mitigation provided>

Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.

SDL Phase: Design

Action Items:

Comments:

145. 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: <a href="https://aka.ms/tmtconfigmgmt#trace-

deploy">https://aka.ms/tmtconfigmgmt#trace-deploy Implement controls to prevent username enumeration. Refer: https://aka.ms/tmtauthn#controls-username-enum

SDL Phase: Implementation

Action Items:

Comments:

146. A malicious user can deny they made a change to App Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation

Description: This is due to the Last Modified By field being overwritten on each save

Justification: <no mitigation provided>

Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities

SDL Phase: Design

Action Items:

Comments:

147. 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

Action Items:

Comments:

148. 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

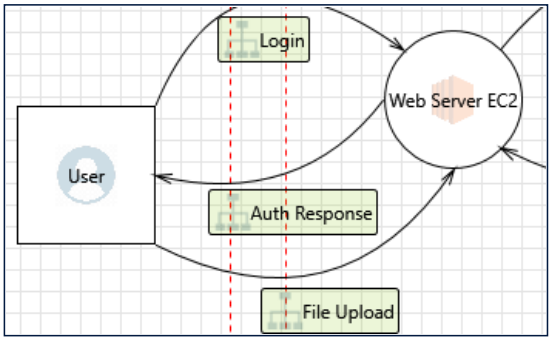
href="https://aka.ms/tmtconfigmgmt#local-js">https://aka.ms/tmtconfigmgmt#local-js Ensure appropriate controls are in place when accepting files from users. Refer: https://aka.ms/tmtinputval#controls-users Disable automatic MIME sniffing. Refer: https://aka.ms/tmtconfigmgmt#mime-sniff Encode untrusted 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

Action Items:

Comments:

Interaction: File Upload



149. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category: Compute

Description: HTTPS Port 443 Open

Justification: <no mitigation provided>

Possible Mitigation(s): Consider placing a Firewall to block unwanted traffic.

SDL Phase: Design

Action Items:

Comments:

150. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category: Compute

Description: HTTP Port 80 is Open to the Internet

Justification: <no mitigation provided>

Possible Mitigation(s): Consider placing Firewalls to block unwanted traffic.

SDL Phase: Design

Action Items:

Comments:

151. SSH Port Exposure [State: Not Started] [Priority: High]

Category: Compute
Description: SSH Port 22 is Open to the Internet
Justification: <no mitigation provided>
Possible Mitigation(s): Close SSH Port 22
SDL Phase: Design
Action Items:
Comments:

152. User Uploading Files [State: Not Started] [Priority: High]

Category: Data Flows
Description: User may try to upload malicious scripts to the Web Server
Justification: <no mitigation provided>
Possible Mitigation(s): Validate the type of file uploaded and do not provide executable permissions to any uploaded File.
SDL Phase: Design
Action Items:
Comments:

153. EC2 Metadata protection [State: Not Started] [Priority: High]

Category: Compute
Description: Don’t expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with “Authentication as a Service” providers like Cognito or Okta.
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you’re using in every EC2 instance
Comments:

154. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category: Compute
Description: Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:

155. Amazon EC2 API Gateway [State: Not Started] [Priority: High]

Category: Compute
Description: Don’t expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with “Authentication as a Service” providers like Cognito or Okta.
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design

Action Items: 1. Dont expose EC2 to the Internet directly and if exposed only allow invocation through API Gateway

Comments:

156. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category: Network Boundary

Description: In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists

Comments:

157. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category: Network Boundary

Description: When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege

Comments:

158. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis

Comments:

159. AWS WAF [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

160. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

161. An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category: Denial of Service

Description: An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack

Justification: <no mitigation provided>

Possible Mitigation(s): Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c

SDL Phase: Implementation

Action Items:

Comments:

162. An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description: An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions

Justification: <no mitigation provided>

Possible Mitigation(s): Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b

SDL Phase: Implementation

Action Items:

Comments:

163. An adversary may gain unauthorized access to privileged features on User [State: Not Started] [Priority: High]

Category: Elevation of Privileges

Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

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

Category:	Elevation of Privileges
Description:	A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification:	<no mitigation provided>
Possible Mitigation(s):	It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase:	Implementation
Action Items:	
Comments:	

165. 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
Action Items:	
Comments:	

166. 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
Action Items:
Comments:

167. An adversary having access to Web Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: An adversary having access to Web Server EC2 may read sensitive clear-text data
Justification: <no mitigation provided>
Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.
SDL Phase: Design
Action Items:
Comments:

168. 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
Action Items:
Comments:

169. A malicious user can deny they made a change to Web Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation
Description: This is due to the Last Modified By field being overwritten on each save
Justification: <no mitigation provided>
Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities
SDL Phase: Design
Action Items:
Comments:

170. 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

Action Items:

Comments:

171. 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>

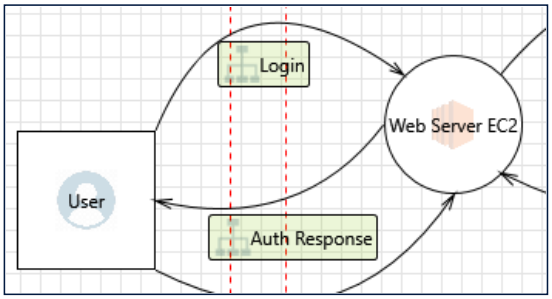
Possible Mitigation(s): Implement Content Security Policy (CSP), and disable inline javascript. Refer: https://aka.ms/tmtconfigmgmt#csp-js Enable browser's XSS filter. Refer: https://aka.ms/tmtconfigmgmt#xss-filter Access third party javascripts from trusted sources only. Refer: https://aka.ms/tmtconfigmgmt#js-trusted 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

Action Items:

Comments:

Interaction: Login



172. HTTPS Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	HTTPS Port 443 Open
Justification:	<no mitigation provided>
Possible Mitigation(s):	Consider placing a Firewall to block unwanted traffic.
SDL Phase:	Design
Action Items:	
Comments:	

173. HTTP Port Exposure [State: Not Started] [Priority: Low]

Category:	Compute
Description:	HTTP Port 80 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Consider placing Firewalls to block unwanted traffic.
SDL Phase:	Design
Action Items:	
Comments:	

174. SSH Port Exposure [State: Not Started] [Priority: High]

Category:	Compute
Description:	SSH Port 22 is Open to the Internet
Justification:	<no mitigation provided>
Possible Mitigation(s):	Close SSH Port 22
SDL Phase:	Design
Action Items:	
Comments:	

175. User Providing Text Input [State: Not Started] [Priority: High]

Category:	Data Flows
Description:	The user may try to provide malicious input and exploit possible command injection vulnerabilities
Justification:	<no mitigation provided>
Possible Mitigation(s):	Conduct Input Validation or Limit User Inputs
SDL Phase:	Design
Action Items:	
Comments:	

176. EC2 Metadata protection [State: Not Started] [Priority: High]

Category: Compute
Description: Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure AWS IMDSv2 is used: if the team had configured their compute instance to use IMDSv2, unauthorized access by the external threat actor would have been blocked 2. Monitor what version of IMDS you're using in every EC2 instance
Comments:

177. Amazon Inspector for EC2 instances [State: Not Started] [Priority: High]

Category: Compute
Description: Amazon Inspector checks for exposures and vulnerabilities at EC2 instance level
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure that Amazon Inspector is used for EC2 instances scan and its report reviewed on a regular basis
Comments:

178. Amazon EC2 API Gateway [State: Not Started] [Priority: High]

Category: Compute
Description: Don't expose EC2 to the wider Internet unless strictly needed. If they are exposed, do not expose them for direct invocation, but put them behind an API Gateway. API Gateways provide DDOS protection, rate limiting and simple integration with "Authentication as a Service" providers like Cognito or Okta.
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Dont expose EC2 to the Internet directly and if exposed only allow invocation through API Gateway
Comments:

179. AWS VPC Network access control lists (ACLs) [State: Not Started] [Priority: High]

Category: Network Boundary
Description: In case of AWS VPC misconfigution, a Threat Agent may easily connect to any internal AWS component or service
Justification: Security Requirement
Possible Mitigation(s):
SDL Phase: Design
Action Items: 1. Ensure AWS VPC Network access control lists (ACLs) has deny-by-default policy which rejects any external or internal connection by default and whitelist only allowed connection for application support or functionality exists
Comments:

180. AWS IAM Security Best practices [State: Not Started] [Priority: High]

Category: Network Boundary
Description: When users do not properly define and configure permissions attached to a service, privilege escalation could happen. This could allow a compromised low-privileged user to change the password of a high-privileged user. The same goes for

improperly configured role permission policies, which could allow a malicious user to create a new policy version that could in turn allow the changing of permissions in a policy. If role permissions are not securely set, the malicious user could be granted full administrator privileges. It is important to note that although it takes time to create a comprehensive list of roles that states which users and services in the architecture are allowed to pass, such a list helps ensure a more secure, misconfiguration-free system.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Locking away the AWS account root user access key 2. Creating individual AWS IAM users 3. Using groups to assign permissions to AWS IAM users 4. Using roles to delegate permissions 5. Granting least privilege

Comments:

181. AWS Trusted Advisor usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: AWS Trusted Advisor is a tool, that allows to evaluate AWS account security (such as SG unrestricted ports and access, IAM policies, MFA usage, etc) and optimization best practices

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure that AWS Trusted Advisor is used and its report reviewed on a regular basis

Comments:

182. AWS WAF [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A WAF or Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. Ensure AWS WAF is enabled to protect public accesible part of the application accessible from the internet for corresponding security group: for example, AWS WAF can be configured to inspect the traffic that is permitted to reach 443 port

Comments:

183. AWS Shield usage [State: Not Started] [Priority: High]

Category: Network Boundary

Description: A distributed denial of service (DDoS) attack is an attack in which multiple compromised systems attempt to flood a target, such as a network or web application, with traffic. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume.

Justification: Security Requirement

Possible Mitigation(s):

SDL Phase: Design

Action Items: 1. AWS provides two levels of protection against DDoS attacks: AWS Shield Standard and AWS Shield Advanced see https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

Comments:

184. An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	An adversary may block access to the application or API hosted on Web Server EC2 through a denial of service attack
Justification:	<no mitigation provided>
Possible Mitigation(s):	Network level denial of service mitigations are automatically enabled as part of the Azure platform (Basic Azure DDoS Protection). Refer: https://aka.ms/tmt-th165a. Implement application level throttling (e.g. per-user, per-session, per-API) to maintain service availability and protect against DoS attacks. Leverage Azure API Management for managing and protecting APIs. Refer: https://aka.ms/tmt-th165b. General throttling guidance, refer: https://aka.ms/tmt-th165c
SDL Phase:	Implementation
Action Items:	
Comments:	

185. An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary can gain unauthorized access to Web Server EC2 due to weak access control restrictions
Justification:	<no mitigation provided>
Possible Mitigation(s):	Grant limited access to objects in Azure Storage using SAS or SAP. It is recommended to scope SAS and SAP to permit only the necessary permissions over a short period of time. Refer: https://aka.ms/tmt-th17a and https://aka.ms/tmt-th17b
SDL Phase:	Implementation
Action Items:	
Comments:	

186. An adversary may gain unauthorized access to privileged features on User [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may get access to admin interface or privileged services like WiFi, SSH, File shares, FTP etc., on a device
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all admin interfaces are secured with strong credentials. Refer: https://aka.ms/tmtconfigmgmt#admin-strong
SDL Phase:	Implementation
Action Items:	
Comments:	

187. An adversary may trigger unauthorized commands on Web Server EC2 [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may leverage insufficient authorization checks on the device and execute unauthorized and sensitive commands remotely.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Perform authorization checks in the device if it supports various actions that require different permission levels. Refer: https://aka.ms/tmtauthz#device-permission
SDL Phase:	Design
Action Items:	
Comments:	

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

Category: Elevation of Privileges
Description: A compromised identity may permit more privileges than intended to an adversary due to weak permission and role assignments.
Justification: <no mitigation provided>
Possible Mitigation(s): It is recommended to review permission and role assignments to ensure the users are granted the least privileges necessary. Refer: https://aka.ms/tmt-th146
SDL Phase: Implementation
Action Items:
Comments:

189. 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
Action Items:
Comments:

190. 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
Action Items:
Comments:

191. An adversary having access to Web Server EC2 may read sensitive clear-text data [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: An adversary having access to Web Server EC2 may read sensitive clear-text data
Justification: <no mitigation provided>
Possible Mitigation(s): Encrypt sensitive data before storing it in Azure Document DB.
SDL Phase: Design

Action Items:

Comments:

192. 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

Action Items:

Comments:

193. A malicious user can deny they made a change to Web Server EC2 [State: Not Started] [Priority: High]

Category: Repudiation

Description: This is due to the Last Modified By field being overwritten on each save

Justification: <no mitigation provided>

Possible Mitigation(s): Identify sensitive entities in your solution and implement change auditing. Refer: https://aka.ms/tmtauditlog#sensitive-entities

SDL Phase: Design

Action Items:

Comments:

194. 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

Action Items:

Comments:

195. 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

Action Items:

Comments: