Threat Modeling Report

Created on 2025-11-27 8:49:22 PM
Threat Model Name:
Owner:
Reviewer:
Contributors:
Description:
Assumptions:
External Dependencies:

Threat Model Summary:

Not Started	33
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	33
Total Migrated	0

Diagram: Diagram 1

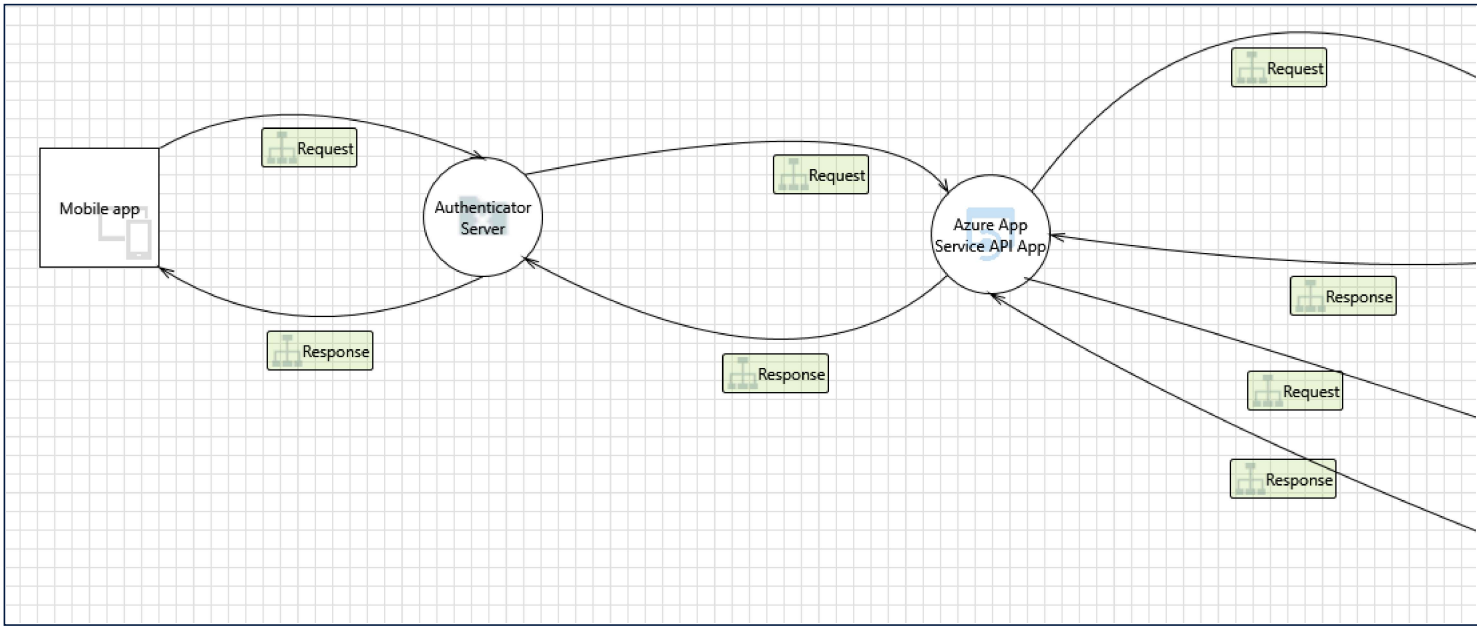
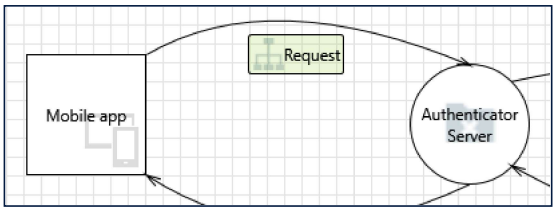


Diagram 1 Diagram Summary:

Not Started	33
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	33
Total Migrated	0

Interaction: Request



1. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Override the default Identity Server token cache with a scalable alternative. Refer: https://aka.ms/tmtauthn#override-token
SDL Phase:	Design

2. An adversary may jail break into a mobile device and gain elevated privileges [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may jail break into a mobile device and gain elevated privileges
Justification:	<no mitigation provided>
Possible Mitigation(s):	Implement implicit jailbreak or rooting detection. Refer: https://aka.ms/tmtauthz#rooting-detection
SDL Phase:	Design

3. An adversary may sniff the data sent from Identity Server [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all traffic to Identity Server is over HTTPS connection. Refer: https://aka.ms/tmtcommsec#identity-https
SDL Phase:	Design

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

Category:	Information Disclosure
Description:	An adversary can gain access to sensitive data by sniffing traffic from Mobile client
Justification:	<no mitigation provided>
Possible Mitigation(s):	Implement Certificate Pinning. Refer: https://aka.ms/tmtcommsec#cert-pinning
SDL Phase:	Implementation

5. An adversary can gain sensitive data from mobile device [State: Not Started] [Priority: High]

Category:	Information Disclosure
Description:	If application saves sensitive PII or HBI data on phone SD card or local storage, then it ay get stolen.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Encrypt sensitive or PII data written to phones local storage. Refer: https://aka.ms/tmtdata#pii-phones
SDL Phase:	Implementation

6. An adversary can bypass authentication due to non-standard Identity Server authentication schemes [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can bypass authentication due to non-standard Identity Server authentication schemes
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use standard authentication scenarios supported by Identity Server. Refer: https://aka.ms/tmtauthn#standard-authn-id

SDL Phase: Design

7. An adversary can get access to a user's session due to improper logout from Identity Server [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can get access to a user's session due to improper logout from Identity Server
Justification:	<no mitigation provided>
Possible Mitigation(s):	Implement proper logout when using Identity Server. Refer: https://aka.ms/tmtsmgmt#proper-logout
SDL Phase:	Implementation

8. An adversary may issue valid tokens if Identity server's signing keys are compromised [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that signing keys are rolled over when using Identity Server. Refer: https://aka.ms/tmtcrypto#rolled-server
SDL Phase:	Design

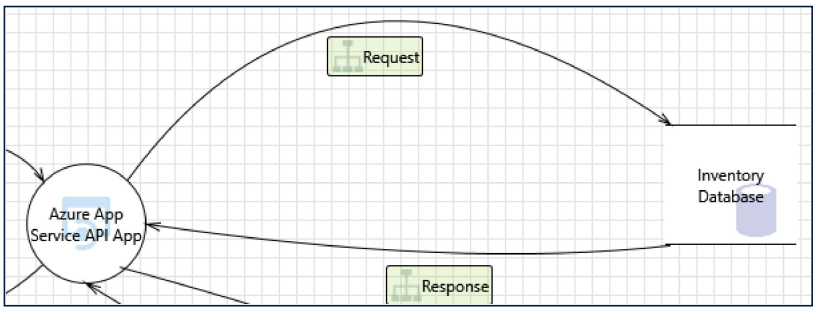
9. An adversary may guess the client id and secrets of registered applications and impersonate them [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary may guess the client id and secrets of registered applications and impersonate them
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that cryptographically strong client id, client secret are used in Identity Server. Refer: https://aka.ms/tmtcrypto#client-server
SDL Phase:	Implementation

10. An adversary can reverse engineer and tamper binaries [State: Not Started] [Priority: High]

Category:	Tampering
Description:	An adversary can use various tools, reverse engineer binaries and abuse them by tampering
Justification:	<no mitigation provided>
Possible Mitigation(s):	Obfuscate generated binaries before distributing to end users. Refer: https://aka.ms/tmtdata#binaries-end
SDL Phase:	Design

Interaction: Request



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

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

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

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

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

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

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

Category:	Information Disclosure
Description:	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that login auditing is enabled on SQL Server. Refer: https://aka.ms/tmtauditlog#identify-sensitive-entities Ensure that least-privileged accounts are used to connect to Database server. Refer: https://aka.ms/tmtauthz#privileged-server Enable Threat detection on Azure SQL database. Refer: https://aka.ms/tmtauditlog#threat-detection Do not use dynamic queries in stored procedures. Refer: https://aka.ms/tmtinputval#stored-proc
SDL Phase:	Implementation

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

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

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

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

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

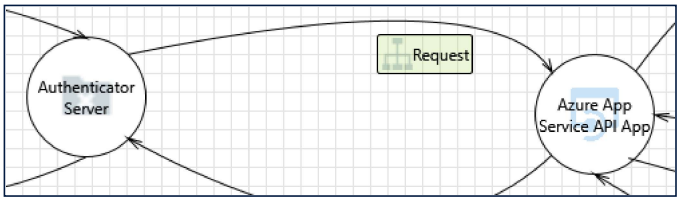
Category:	Elevation of Privileges

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

18. An adversary can tamper SSIS packages and cause undesirable consequences [State: Not Started] [Priority: High]

Category:	Tampering
Description:	The source of a package is the individual or organization that created the package. Running a package from an unknown or untrusted source might be risky.
Justification:	<no mitigation provided>
Possible Mitigation(s):	SSIS packages should be encrypted and digitally signed . Refer: https://aka.ms/tmtcrypto#ssis-signed
SDL Phase:	Design

Interaction: Request



19. An adversary may block access to the application or API hosted on Azure App Service API App 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 Azure App Service API App 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

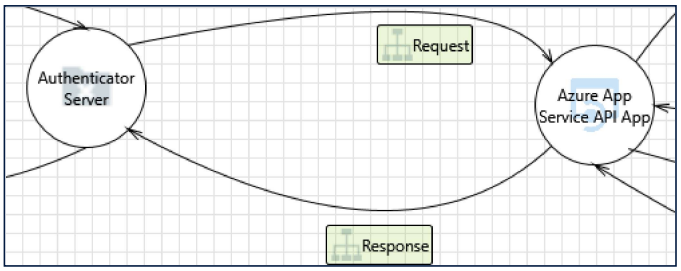
20. An adversary may gain long term persistent access to related resources through the compromise of an application identity [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may gain long term persistent access to related resources through the compromise of an application identity
Justification:	<no mitigation provided>
Possible Mitigation(s):	Store secrets in secret storage solutions where possible, and rotate secrets on a regular cadence. Use Managed Service Identity to create a managed app identity on Azure Active Directory and use it to access AAD-protected resources. Refer: https://aka.ms/tmt-th166
SDL Phase:	Implementation

21. An adversary may perform action(s) on behalf of another user due to lack of controls against cross domain requests [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may perform action(s) on behalf of another user due to lack of controls against cross domain requests
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that only trusted origins are allowed if CORS is being used. Refer: https://aka.ms/tmt-th176
SDL Phase:	Implementation

Interaction: Response



22. An adversary may guess the client id and secrets of registered applications and impersonate them [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary may guess the client id and secrets of registered applications and impersonate them
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that cryptographically strong client id, client secret are used in Identity Server. Refer: https://aka.ms/tmtcrypto#client-server
SDL Phase:	Implementation

23. An adversary may issue valid tokens if Identity server's signing keys are compromised [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that signing keys are rolled over when using Identity Server. Refer: https://aka.ms/tmtcrypto#rolled-server
SDL Phase:	Design

24. An adversary can get access to a user's session due to improper logout from Identity Server [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can get access to a user's session due to improper logout from Identity Server
Justification:	<no mitigation provided>
Possible Mitigation(s):	Implement proper logout when using Identity Server. Refer: https://aka.ms/tmtsmgmt#proper-logout
SDL Phase:	Implementation

25. An adversary can bypass authentication due to non-standard Identity Server authentication schemes [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	An adversary can bypass authentication due to non-standard Identity Server authentication schemes
Justification:	<no mitigation provided>
Possible Mitigation(s):	Use standard authentication scenarios supported by Identity Server. Refer: https://aka.ms/tmtauthn#standard-authn-id
SDL Phase:	Design

26. An adversary may sniff the data sent from Identity Server [State: Not Started] [Priority: High]

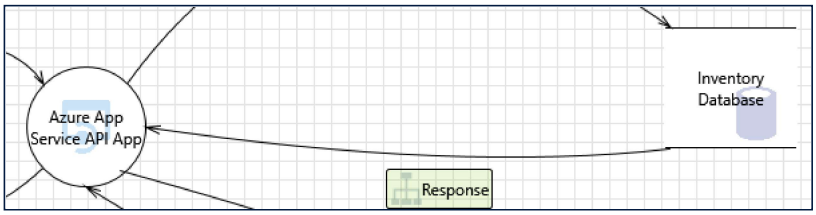
Category:	Information Disclosure
Description:	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that all traffic to Identity Server is over HTTPS connection. Refer: https://aka.ms/tmtcommsec#identity-https
SDL Phase:	Design

27. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS [State: Not Started] [Priority: High]

Category:	Denial of Service
Description:	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.

Justification:	<no mitigation provided>
Possible Mitigation(s):	Override the default Identity Server token cache with a scalable alternative. Refer: https://aka.ms/tmtauthn#override-token
SDL Phase:	Design

Interaction: Response



28. An adversary may perform action(s) on behalf of another user due to lack of controls against cross domain requests [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may perform action(s) on behalf of another user due to lack of controls against cross domain requests
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that only trusted origins are allowed if CORS is being used. Refer: https://aka.ms/tmt-th176
SDL Phase:	Implementation

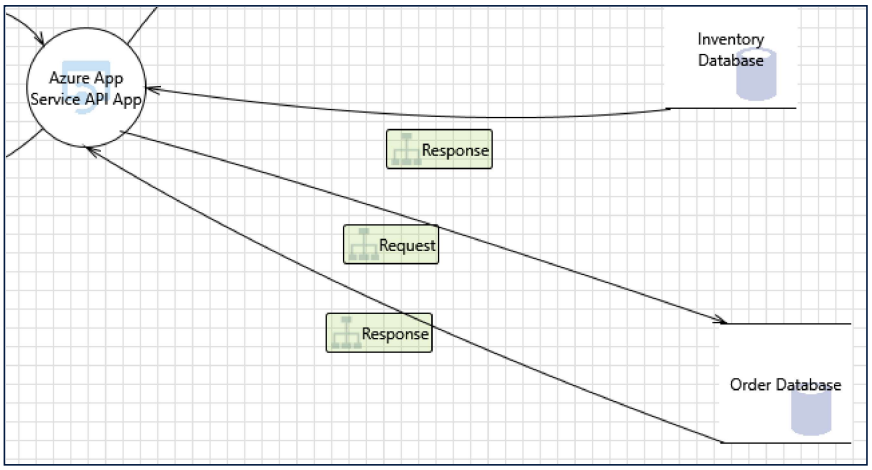
29. An adversary may gain long term persistent access to related resources through the compromise of an application identity [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may gain long term persistent access to related resources through the compromise of an application identity
Justification:	<no mitigation provided>
Possible Mitigation(s):	Store secrets in secret storage solutions where possible, and rotate secrets on a regular cadence. Use Managed Service Identity to create a managed app identity on Azure Active Directory and use it to access AAD-protected resources. Refer: https://aka.ms/tmt-th166
SDL Phase:	Implementation

30. An adversary may block access to the application or API hosted on Azure App Service API App 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 Azure App Service API App 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

Interaction: Response



31. An adversary may block access to the application or API hosted on Azure App Service API App 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 Azure App Service API App 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

32. An adversary may gain long term persistent access to related resources through the compromise of an application identity [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may gain long term persistent access to related resources through the compromise of an application identity
Justification:	<no mitigation provided>
Possible Mitigation(s):	Store secrets in secret storage solutions where possible, and rotate secrets on a regular cadence. Use Managed Service Identity to create a managed app identity on Azure Active Directory and use it to access AAD-protected resources. Refer: https://aka.ms/tmt-th166
SDL Phase:	Implementation

33. An adversary may perform action(s) on behalf of another user due to lack of controls against cross domain requests [State: Not Started] [Priority: High]

Category:	Elevation of Privileges
Description:	An adversary may perform action(s) on behalf of another user due to lack of controls against cross domain requests
Justification:	<no mitigation provided>
Possible Mitigation(s):	Ensure that only trusted origins are allowed if CORS is being used. Refer: https://aka.ms/tmt-th176
SDL Phase:	Implementation