Threat Modeling Report

Created on 8/16/2025 1:26:22 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

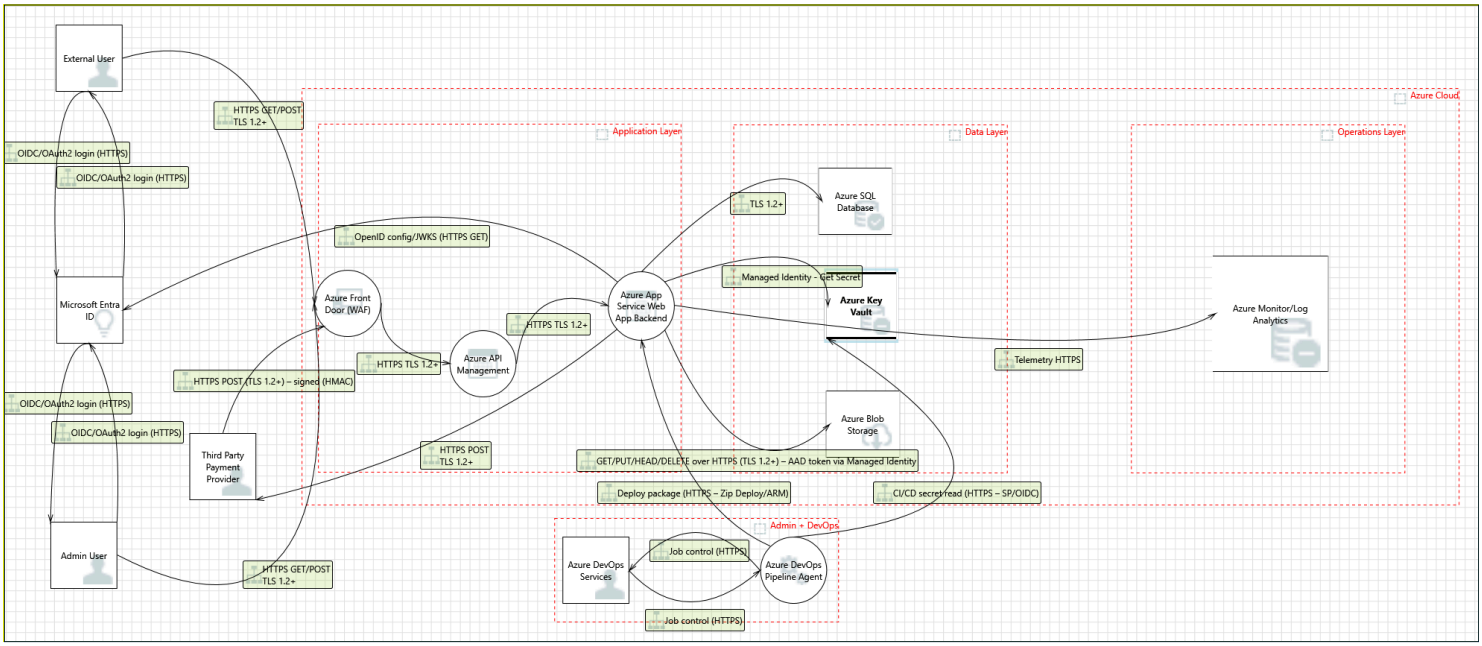
Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	128
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	128
Total Migrated	0

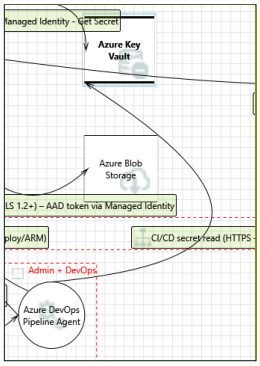
Diagram: Azure Web App



Azure Web App Diagram Summary:

Not Started	128
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	128
Total Migrated	0

Interaction: CI/CD secret read (HTTPS – SP/OIDC)



1. Spoofing of Destination Data Store Azure Key Vault [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Key Vault may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Azure Key Vault. Consider using a standard authentication mechanism to identify the destination data store.
Justification: <no mitigation provided>

2. The Azure Key Vault Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across CI/CD secret read (HTTPS – SP/OIDC) may be tampered with by an attacker. This may lead to corruption of Azure Key Vault. Ensure the integrity of the data flow to the data store.
Justification: <no mitigation provided>

3. Data Store Denies Azure Key Vault Potentially Writing Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Key Vault claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

4. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across CI/CD secret read (HTTPS – SP/OIDC) may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

5. Potential Excessive Resource Consumption for Azure DevOps Pipeline Agent or Azure Key Vault [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Does Azure DevOps Pipeline Agent or Azure Key Vault take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification: <no mitigation provided>

6. Data Flow CI/CD secret read (HTTPS – SP/OIDC) Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

7. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: <no mitigation provided>

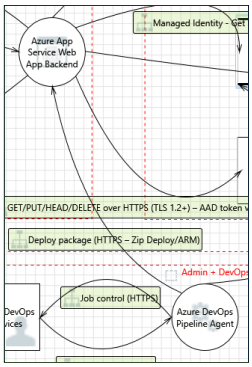
8. Authorization Bypass [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Can you access Azure Key Vault and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface.
Justification: <no mitigation provided>

9. Weak Credential Storage [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored
Justification: <no mitigation provided>

Interaction: Deploy package (HTTPS – Zip Deploy/ARM)



10. Replay Attacks [State: Not Started] [Priority: High]

Category: Tampering
Description: Packets or messages without sequence numbers or timestamps can be captured and replayed in a wide variety of ways. Implement or utilize an existing communication protocol that supports anti-replay techniques (investigate sequence numbers before timers) and strong integrity.
Justification: <no mitigation provided>

11. Potential Lack of Input Validation for Azure App Service Web App Backend [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across Deploy package (HTTPS – Zip Deploy/ARM) may be tampered with by an attacker. This may lead to a denial of service attack against Azure App Service Web App Backend or an elevation of privilege attack against Azure App Service Web App Backend or an information disclosure by Azure App Service Web App Backend. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.
Justification: <no mitigation provided>

12. Spoofing the Azure App Service Web App Backend Process [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure App Service Web App Backend may be spoofed by an attacker and this may lead to information disclosure by Azure DevOps Pipeline Agent. Consider using a standard authentication mechanism to identify the destination process.
Justification: <no mitigation provided>

13. Azure DevOps Pipeline Agent Process Memory Tampered [State: Not Started] [Priority: High]

Category: Tampering
Description: If Azure DevOps Pipeline Agent is given access to memory, such as shared memory or pointers, or is given the ability to control what Azure App Service Web App Backend executes (for example, passing back a function pointer.), then Azure DevOps Pipeline Agent can tamper with Azure App Service Web App Backend. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: <no mitigation provided>

14. Cross Site Scripting [State: Not Started] [Priority: High]

Category: Tampering
Description: The web server 'Azure App Service Web App Backend' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: <no mitigation provided>

15. Potential Data Repudiation by Azure App Service Web App Backend [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure App Service Web App Backend claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

16. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across Deploy package (HTTPS – Zip Deploy/ARM) may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

17. Potential Process Crash or Stop for Azure App Service Web App Backend [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Azure App Service Web App Backend crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

18. Data Flow Deploy package (HTTPS – Zip Deploy/ARM) Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

19. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure App Service Web App Backend may be able to impersonate the context of Azure DevOps Pipeline Agent in order to gain additional privilege.
Justification: <no mitigation provided>

20. Azure App Service Web App Backend May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure DevOps Pipeline Agent may be able to remotely execute code for Azure App Service Web App Backend.
Justification: <no mitigation provided>

21. Elevation by Changing the Execution Flow in Azure App Service Web App Backend [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Azure App Service Web App Backend in order to change the flow of program execution within Azure App Service Web App Backend to the attacker's choosing.
Justification: <no mitigation provided>

22. Cross Site Request Forgery [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting. The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: <no mitigation provided>

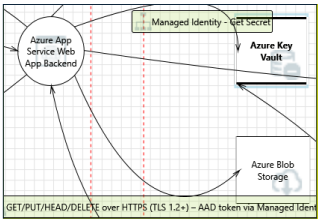
23. Collision Attacks [State: Not Started] [Priority: High]

Category: Tampering
Description: Attackers who can send a series of packets or messages may be able to overlap data. For example, packet 1 may be 100 bytes starting at offset 0. Packet 2 may be 100 bytes starting at offset 25. Packet 2 will overwrite 75 bytes of packet 1. Ensure you reassemble data before filtering it, and ensure you explicitly handle these sorts of cases.
Justification: <no mitigation provided>

24. Weak Authentication Scheme [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme.
Justification: <no mitigation provided>

Interaction: GET/PUT/HEAD/DELETE over HTTPS (TLS 1.2+) – AAD token via Managed Identity



25. Spoofing of Destination Data Store Azure Blob Storage [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Blob Storage may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Azure Blob Storage. Consider using a standard authentication mechanism to identify the destination data store.
Justification: <no mitigation provided>

26. The Azure Blob Storage Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across GET/PUT/HEAD/DELETE over HTTPS (TLS 1.2+) – AAD token via Managed Identity may be tampered with by an attacker. This may lead to corruption of Azure Blob Storage. Ensure the integrity of the data flow to the data store.
Justification: <no mitigation provided>

27. Data Store Denies Azure Blob Storage Potentially Writing Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Blob Storage claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

28. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across GET/PUT/HEAD/DELETE over HTTPS (TLS 1.2+) – AAD token via Managed Identity may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

29. Potential Excessive Resource Consumption for Azure App Service Web App Backend or Azure Blob Storage [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Does Azure App Service Web App Backend or Azure Blob Storage take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification: <no mitigation provided>

30. Data Flow GET/PUT/HEAD/DELETE over HTTPS (TLS 1.2+) – AAD token via Managed Identity Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

31. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: <no mitigation provided>

32. Authorization Bypass [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Can you access Azure Blob Storage and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface.
Justification: <no mitigation provided>

33. Risks from Logging [State: Not Started] [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources.
Justification: <no mitigation provided>

34. Lower Trusted Subject Updates Logs [State: Not Started] [Priority: High]

Category: Repudiation
Description: If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log.
Justification: <no mitigation provided>

35. Data Logs from an Unknown Source [State: Not Started] [Priority: High]

Category: Repudiation
Description: Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them.
Justification: <no mitigation provided>

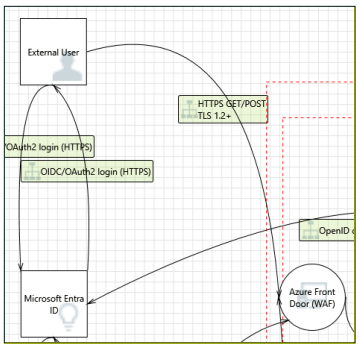
36. Insufficient Auditing [State: Not Started] [Priority: High]

Category: Repudiation
Description: Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle a repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data.
Justification: <no mitigation provided>

37. Potential Weak Protections for Audit Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect
Justification: <no mitigation provided>

Interaction: HTTPS GET/POST TLS 1.2+



38. Cross Site Request Forgery [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting. The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: <no mitigation provided>

39. Elevation by Changing the Execution Flow in Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Azure Front Door (WAF) in order to change the flow of program execution within Azure Front Door (WAF) to the attacker's choosing.
Justification: <no mitigation provided>

40. Azure Front Door (WAF) May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: External User may be able to remotely execute code for Azure Front Door (WAF).
Justification: <no mitigation provided>

41. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure Front Door (WAF) may be able to impersonate the context of External User in order to gain additional privilege.
Justification: <no mitigation provided>

42. Data Flow HTTPS GET/POST TLS 1.2+ Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

43. Potential Process Crash or Stop for Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Azure Front Door (WAF) crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

44. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across HTTPS GET/POST TLS 1.2+ may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

45. Potential Data Repudiation by Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Front Door (WAF) claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

46. Cross Site Scripting [State: Not Started] [Priority: High]

Category: Tampering
Description: The web server 'Azure Front Door (WAF)' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: <no mitigation provided>

47. Potential Lack of Input Validation for Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across HTTPS GET/POST TLS 1.2+ may be tampered with by an attacker. This may lead to a denial of service attack against Azure Front Door (WAF) or an elevation of privilege attack against Azure Front Door (WAF) or an information disclosure by Azure Front Door (WAF). Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.
Justification: <no mitigation provided>

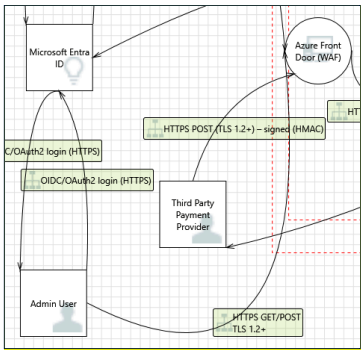
48. Spoofing the External User External Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: External User may be spoofed by an attacker and this may lead to unauthorized access to Azure Front Door (WAF). Consider using a standard authentication mechanism to identify the external entity.
Justification: <no mitigation provided>

49. Spoofing the Azure Front Door (WAF) Process [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Front Door (WAF) may be spoofed by an attacker and this may lead to information disclosure by External User. Consider using a standard authentication mechanism to identify the destination process.
Justification: <no mitigation provided>

Interaction: HTTPS GET/POST TLS 1.2+



50. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across HTTPS GET/POST TLS 1.2+ may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

51. Potential Data Repudiation by Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Front Door (WAF) claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

52. Cross Site Scripting [State: Not Started] [Priority: High]

Category: Tampering
Description: The web server 'Azure Front Door (WAF)' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: <no mitigation provided>

53. Potential Lack of Input Validation for Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across HTTPS GET/POST TLS 1.2+ may be tampered with by an attacker. This may lead to a denial of service attack against Azure Front Door (WAF) or an elevation of privilege attack against Azure Front Door (WAF) or an information disclosure by Azure Front Door (WAF). Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.
Justification: <no mitigation provided>

54. Spoofing the Admin User External Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: Admin User may be spoofed by an attacker and this may lead to unauthorized access to Azure Front Door (WAF). Consider using a standard authentication mechanism to identify the external entity.
Justification: <no mitigation provided>

55. Spoofing the Azure Front Door (WAF) Process [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Front Door (WAF) may be spoofed by an attacker and this may lead to information disclosure by Admin User. Consider using a standard authentication mechanism to identify the destination process.
Justification: <no mitigation provided>

56. Potential Process Crash or Stop for Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Azure Front Door (WAF) crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

57. Data Flow HTTPS GET/POST TLS 1.2+ Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

58. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure Front Door (WAF) may be able to impersonate the context of Admin User in order to gain additional privilege.
Justification: <no mitigation provided>

59. Azure Front Door (WAF) May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Admin User may be able to remotely execute code for Azure Front Door (WAF).
Justification: <no mitigation provided>

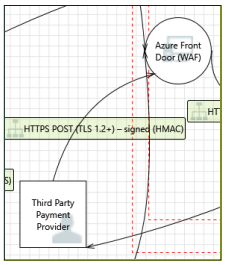
60. Elevation by Changing the Execution Flow in Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Azure Front Door (WAF) in order to change the flow of program execution within Azure Front Door (WAF) to the attacker's choosing.
Justification: <no mitigation provided>

61. Cross Site Request Forgery [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting. The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: <no mitigation provided>

Interaction: HTTPS POST (TLS 1.2+) – signed (HMAC)



62. Spoofing the Azure Front Door (WAF) Process [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Front Door (WAF) may be spoofed by an attacker and this may lead to information disclosure by Third Party Payment Provider. Consider using a standard authentication mechanism to identify the destination process.
Justification: <no mitigation provided>

63. Spoofing the Third Party Payment Provider External Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: Third Party Payment Provider may be spoofed by an attacker and this may lead to unauthorized access to Azure Front Door (WAF). Consider using a standard authentication mechanism to identify the external entity.
Justification: <no mitigation provided>

64. Potential Lack of Input Validation for Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across HTTPS POST (TLS 1.2+) – signed (HMAC) may be tampered with by an attacker. This may lead to a denial of service attack against Azure Front Door (WAF) or an elevation of privilege attack against Azure Front Door (WAF) or an information disclosure by Azure Front Door (WAF). Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.
Justification: <no mitigation provided>

65. Cross Site Scripting [State: Not Started] [Priority: High]

Category: Tampering
Description: The web server 'Azure Front Door (WAF)' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: <no mitigation provided>

66. Potential Data Repudiation by Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Front Door (WAF) claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

67. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across HTTPS POST (TLS 1.2+) – signed (HMAC) may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

68. Potential Process Crash or Stop for Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Azure Front Door (WAF) crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

69. Data Flow HTTPS POST (TLS 1.2+) – signed (HMAC) Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

70. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure Front Door (WAF) may be able to impersonate the context of Third Party Payment Provider in order to gain additional privilege.
Justification: <no mitigation provided>

71. Azure Front Door (WAF) May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Third Party Payment Provider may be able to remotely execute code for Azure Front Door (WAF).
Justification: <no mitigation provided>

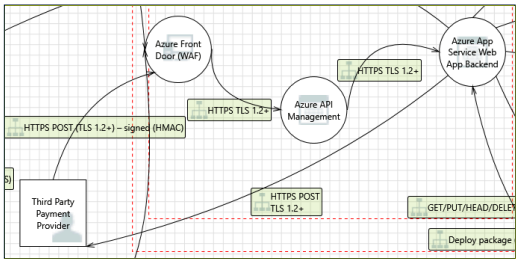
72. Elevation by Changing the Execution Flow in Azure Front Door (WAF) [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Azure Front Door (WAF) in order to change the flow of program execution within Azure Front Door (WAF) to the attacker's choosing.
Justification: <no mitigation provided>

73. Cross Site Request Forgery [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting. The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: <no mitigation provided>

Interaction: HTTPS POST TLS 1.2+



74. Spoofing of the Third Party Payment Provider External Destination Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: Third Party Payment Provider may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Third Party Payment Provider. Consider using a standard authentication mechanism to identify the external entity.
Justification: <no mitigation provided>

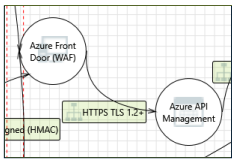
75. External Entity Third Party Payment Provider Potentially Denies Receiving Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Third Party Payment Provider claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

76. Data Flow HTTPS POST TLS 1.2+ Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

Interaction: HTTPS TLS 1.2+



77. Azure Front Door (WAF) Process Memory Tampered [State: Not Started] [Priority: High]

Category: Tampering
Description: If Azure Front Door (WAF) is given access to memory, such as shared memory or pointers, or is given the ability to control what Azure API Management executes (for example, passing back a function pointer.), then Azure Front Door (WAF) can tamper with Azure API Management. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: <no mitigation provided>

78. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure API Management may be able to impersonate the context of Azure Front Door (WAF) in order to gain additional privilege.
Justification: <no mitigation provided>

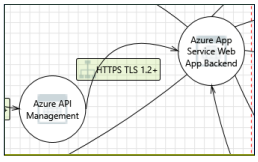
79. Collision Attacks [State: Not Started] [Priority: High]

Category: Tampering
Description: Attackers who can send a series of packets or messages may be able to overlap data. For example, packet 1 may be 100 bytes starting at offset 0. Packet 2 may be 100 bytes starting at offset 25. Packet 2 will overwrite 75 bytes of packet 1. Ensure you reassemble data before filtering it, and ensure you explicitly handle these sorts of cases.
Justification: <no mitigation provided>

80. Replay Attacks [State: Not Started] [Priority: High]

Category: Tampering
Description: Packets or messages without sequence numbers or timestamps can be captured and replayed in a wide variety of ways. Implement or utilize an existing communication protocol that supports anti-replay techniques (investigate sequence numbers before timers) and strong integrity.
Justification: <no mitigation provided>

Interaction: HTTPS TLS 1.2+



81. Azure API Management Process Memory Tampered [State: Not Started] [Priority: High]

Category: Tampering
Description: If Azure API Management is given access to memory, such as shared memory or pointers, or is given the ability to control what Azure App Service Web App Backend executes (for example, passing back a function pointer.), then Azure API Management can tamper with Azure App Service Web App Backend. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: <no mitigation provided>

82. Cross Site Scripting [State: Not Started] [Priority: High]

Category: Tampering

Description: The web server 'Azure App Service Web App Backend' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: <no mitigation provided>

83. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure App Service Web App Backend may be able to impersonate the context of Azure API Management in order to gain additional privilege.
Justification: <no mitigation provided>

84. Collision Attacks [State: Not Started] [Priority: High]

Category: Tampering
Description: Attackers who can send a series of packets or messages may be able to overlap data. For example, packet 1 may be 100 bytes starting at offset 0. Packet 2 may be 100 bytes starting at offset 25. Packet 2 will overwrite 75 bytes of packet 1. Ensure you reassemble data before filtering it, and ensure you explicitly handle these sorts of cases.
Justification: <no mitigation provided>

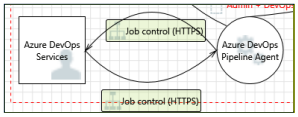
85. Replay Attacks [State: Not Started] [Priority: High]

Category: Tampering
Description: Packets or messages without sequence numbers or timestamps can be captured and replayed in a wide variety of ways. Implement or utilize an existing communication protocol that supports anti-replay techniques (investigate sequence numbers before timers) and strong integrity.
Justification: <no mitigation provided>

86. Weak Authentication Scheme [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme.
Justification: <no mitigation provided>

Interaction: Job control (HTTPS)



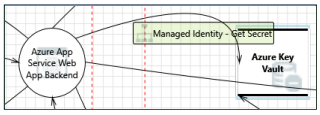
87. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Azure DevOps Pipeline Agent may be able to impersonate the context of Azure DevOps Services in order to gain additional privilege.
Justification: <no mitigation provided>

88. Spoofing the Azure DevOps Services External Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure DevOps Services may be spoofed by an attacker and this may lead to unauthorized access to Azure DevOps Pipeline Agent. Consider using a standard authentication mechanism to identify the external entity.
Justification: <no mitigation provided>

Interaction: Managed Identity - Get Secret



89. Spoofing of Destination Data Store Azure Key Vault [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Key Vault may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Azure Key Vault. Consider using a standard authentication mechanism to identify the destination data store.
Justification: <no mitigation provided>

90. The Azure Key Vault Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across Managed Identity - Get Secret may be tampered with by an attacker. This may lead to corruption of Azure Key Vault. Ensure the integrity of the data flow to the data store.
Justification: <no mitigation provided>

91. Data Store Denies Azure Key Vault Potentially Writing Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Key Vault claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

92. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across Managed Identity - Get Secret may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

93. Potential Excessive Resource Consumption for Azure App Service Web App Backend or Azure Key Vault [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Does Azure App Service Web App Backend or Azure Key Vault take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification: <no mitigation provided>

94. Data Flow Managed Identity - Get Secret Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

95. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: <no mitigation provided>

96. Authorization Bypass [State: Not Started] [Priority: High]

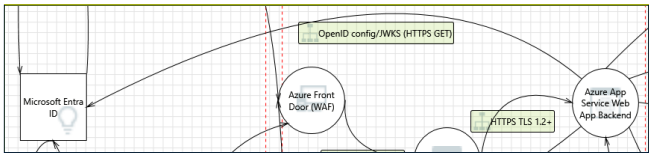
Category: Information Disclosure
Description: Can you access Azure Key Vault and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface.

Justification: <no mitigation provided>

97. Weak Credential Storage [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored
Justification: <no mitigation provided>

Interaction: OpenID config/JWKS (HTTPS GET)



98. Data Flow OpenID config/JWKS (HTTPS GET) Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

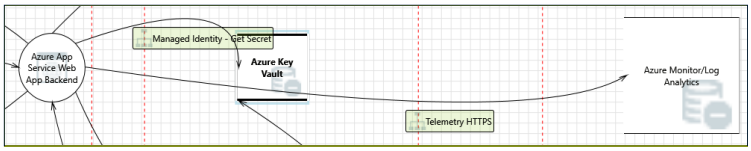
99. External Entity Microsoft Entra ID Potentially Denies Receiving Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Microsoft Entra ID claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

100. Spoofing of the Microsoft Entra ID External Destination Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: Microsoft Entra ID may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Microsoft Entra ID. Consider using a standard authentication mechanism to identify the external entity.
Justification: <no mitigation provided>

Interaction: Telemetry HTTPS



101. Spoofing of Destination Data Store Azure Monitor/Log Analytics [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure Monitor/Log Analytics may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Azure Monitor/Log Analytics. Consider using a standard authentication mechanism to identify the destination data store.
Justification: <no mitigation provided>

102. The Azure Monitor/Log Analytics Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across Telemetry HTTPS may be tampered with by an attacker. This may lead to corruption of Azure Monitor/Log Analytics. Ensure the integrity of the data flow to the data store.
Justification: <no mitigation provided>

103. Data Store Denies Azure Monitor/Log Analytics Potentially Writing Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure Monitor/Log Analytics claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

104. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across Telemetry HTTPS may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

105. Potential Excessive Resource Consumption for Azure App Service Web App Backend or Azure Monitor/Log Analytics [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Does Azure App Service Web App Backend or Azure Monitor/Log Analytics take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification: <no mitigation provided>

106. Data Flow Telemetry HTTPS Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

107. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: <no mitigation provided>

108. Authorization Bypass [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Can you access Azure Monitor/Log Analytics and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface.
Justification: <no mitigation provided>

109. Risks from Logging [State: Not Started] [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources.
Justification: <no mitigation provided>

110. Lower Trusted Subject Updates Logs [State: Not Started] [Priority: High]

Category: Repudiation

Description: If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log.
Justification: <no mitigation provided>

111. Data Logs from an Unknown Source [State: Not Started] [Priority: High]

Category: Repudiation
Description: Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them.
Justification: <no mitigation provided>

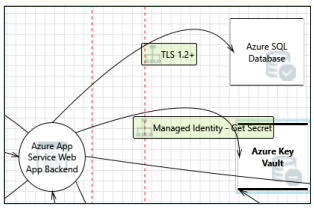
112. Insufficient Auditing [State: Not Started] [Priority: High]

Category: Repudiation
Description: Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle a repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data.
Justification: <no mitigation provided>

113. Potential Weak Protections for Audit Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect
Justification: <no mitigation provided>

Interaction: TLS 1.2+



114. Spoofing of Destination Data Store Azure SQL Database [State: Not Started] [Priority: High]

Category: Spoofing
Description: Azure SQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Azure SQL Database. Consider using a standard authentication mechanism to identify the destination data store.
Justification: <no mitigation provided>

115. Potential SQL Injection Vulnerability for Azure SQL Database [State: Not Started] [Priority: High]

Category: Tampering
Description: SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.
Justification: <no mitigation provided>

116. The Azure SQL Database Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across TLS 1.2+ may be tampered with by an attacker. This may lead to corruption of Azure SQL Database. Ensure the integrity of the data flow to the data store.
Justification: <no mitigation provided>

117. Data Store Denies Azure SQL Database Potentially Writing Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Azure SQL Database claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

118. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Data flowing across TLS 1.2+ may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification: <no mitigation provided>

119. Potential Excessive Resource Consumption for Azure App Service Web App Backend or Azure SQL Database [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: Does Azure App Service Web App Backend or Azure SQL Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification: <no mitigation provided>

120. Data Flow TLS 1.2+ Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

121. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: <no mitigation provided>

122. Lower Trusted Subject Updates Logs [State: Not Started] [Priority: High]

Category: Repudiation
Description: If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log.
Justification: <no mitigation provided>

123. Risks from Logging [State: Not Started] [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources.
Justification: <no mitigation provided>

124. Authorization Bypass [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Can you access Azure SQL Database and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface.
Justification: <no mitigation provided>

125. Data Logs from an Unknown Source [State: Not Started] [Priority: High]

Category: Repudiation

Description: Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them.
Justification: <no mitigation provided>

126. Insufficient Auditing [State: Not Started] [Priority: High]

Category: Repudiation
Description: Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle a repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data.
Justification: <no mitigation provided>

127. Potential Weak Protections for Audit Data [State: Not Started] [Priority: High]

Category: Repudiation
Description: Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect
Justification: <no mitigation provided>

128. Weak Credential Storage [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored
Justification: <no mitigation provided>