Threat Modeling Report

Created on 11/7/2018 10:37:52 PM

Threat Model Name: lvl_1_publish_orders

Owner: Fantastic Four Team

Reviewer:

Contributors: Neil Thorne, Emily Mays, Sanjar Hamidi, Pat Peters

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	15
Needs Investigation	11
Mitigation Implemented	59
Total	85
Total Migrated	0

Diagram:

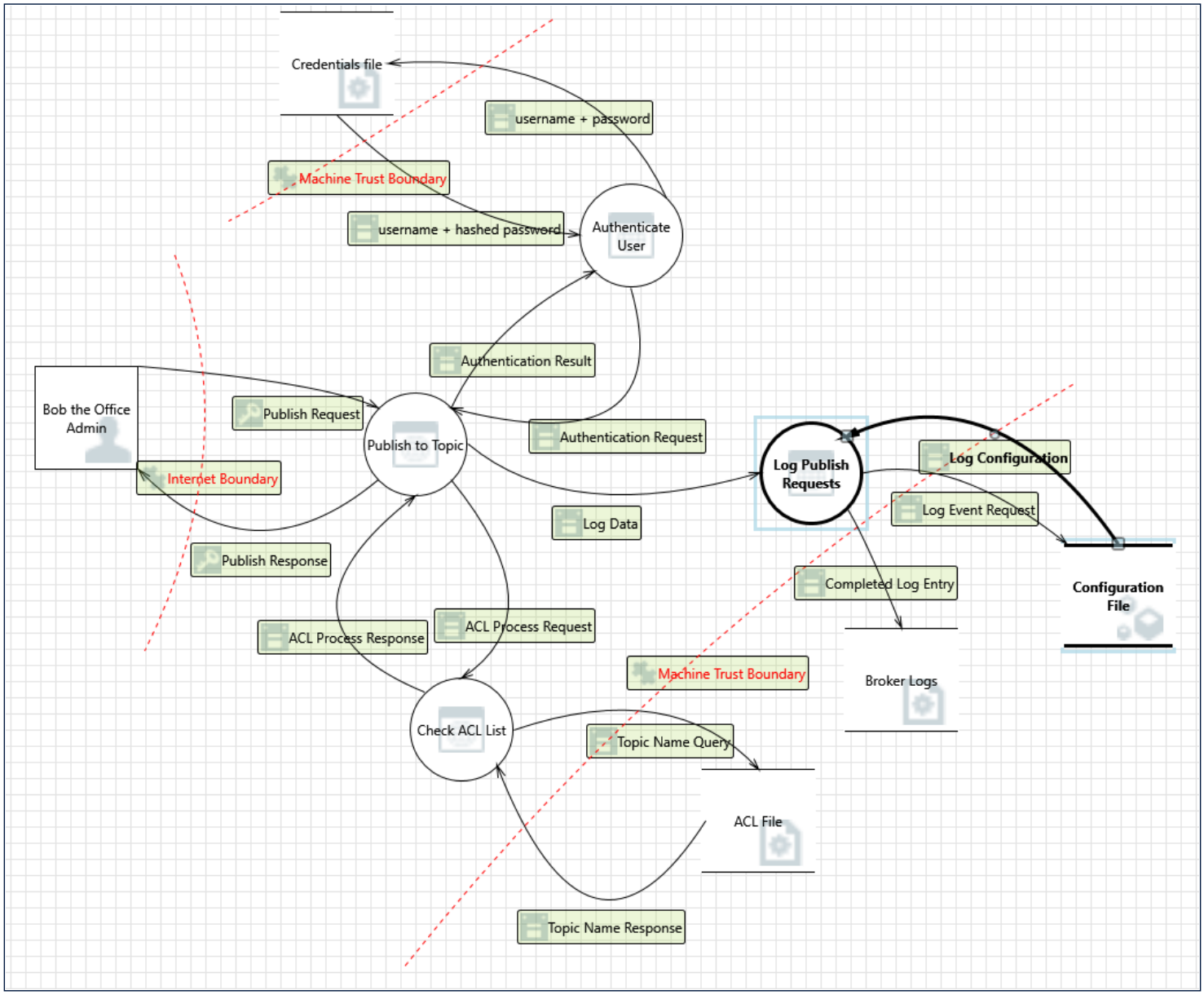
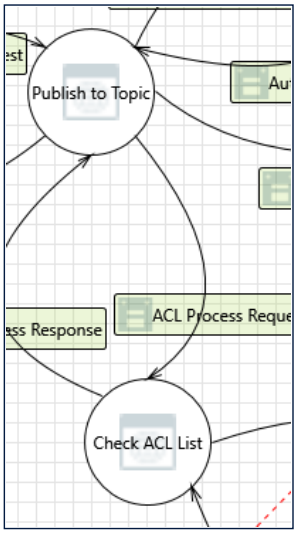


Diagram Summary:

Not Started	0
Not Applicable	15
Needs Investigation	11
Mitigation Implemented	59
Total	85
Total Migrated	0

Interaction: ACL Process Request



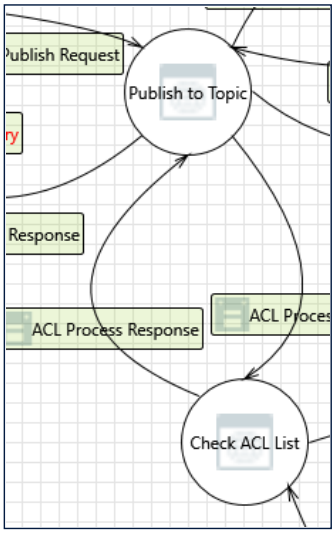
1. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: Check ACL List may be able to impersonate the context of Publish to Topic in order to gain additional privilege.

Justification: Currently all internal processes for a mosquitto broker instance run under the same user, so there are no additional privileges to be gained between these two processes.

Interaction: ACL Process Response



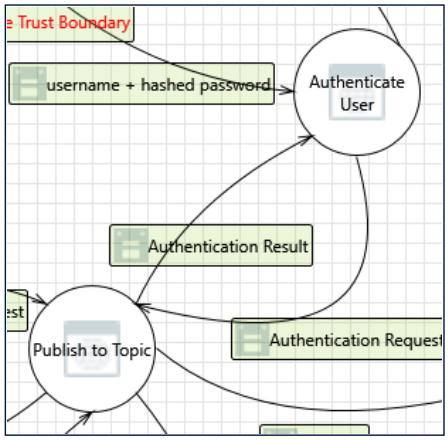
2. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: Publish to Topic may be able to impersonate the context of Check ACL List in order to gain additional privilege.

Justification: Currently all internal processes for a mosquitto broker instance run under the same user, so there are no additional privileges to be gained between these two processes.

Interaction: Authentication Request



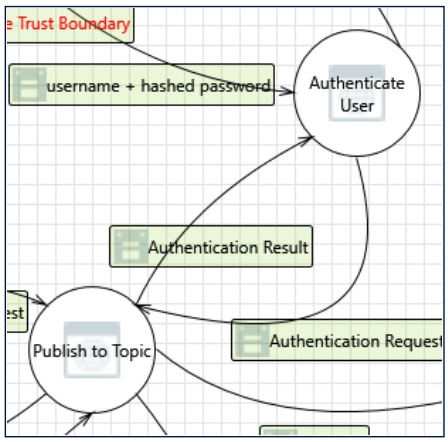
3. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: Publish to Topic may be able to impersonate the context of Authenticate User in order to gain additional privilege.

Justification: Currently all internal processes for a mosquitto broker instance run under the same user, so there are no additional privileges to be gained between these two processes.

Interaction: Authentication Result



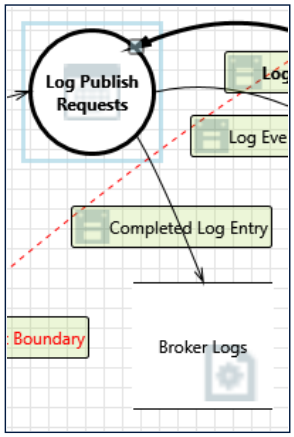
4. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: Authenticate User may be able to impersonate the context of Publish to Topic in order to gain additional privilege.

Justification: Currently all internal processes for a mosquitto broker instance run under the same user, so there are no additional privileges to be gained between these two processes.

Interaction: Completed Log Entry



5. Spoofing the Log Publish Requests Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Log Publish Requests may be spoofed by an attacker and this may lead to unauthorized access to Broker Logs. Consider using a standard authentication mechanism to identify the source process.
Justification: ACLs and dual factor authentication with keys can prevent the broker logs from being spoofed.

6. Spoofing of Destination Data Store Broker Logs [State: Not Applicable] [Priority: High]

Category: Spoofing
Description: Broker Logs may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Broker Logs. Consider using a standard authentication mechanism to identify the destination data store.
Justification: All data from the process being written to the broker log in this instance is already provided to the user in the ack or denial by the broker. Because of this, this is not applicable.

7. The Broker Logs Data Store Could Be Corrupted [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Data flowing across Completed Log Entry may be tampered with by an attacker. This may lead to corruption of Broker Logs. Ensure the integrity of the data flow to the data store.
Justification: Integrity checking of the data gathered to be placed into the logs with a combination of dual factor authentication with a pre-shared key, can help mitigate this sort of attack.

8. Data Store Denies Broker Logs Potentially Writing Data [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Broker Logs 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: When enabled, logging of the post data, post request, post user, post connection and post acknowledgement/denial are tracked, which would audit the source, time, and summary of the received data.

9. Data Flow Sniffing [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Data flowing across Completed Log Entry 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: When writing to a Log file, log data can only be read by users logged in to the host machine of the broker, log files cannot be read from within the Mosquitto Broker instance. When writing to the $SYS topic, all log data can be read by any users that aren't blocked from an ACL that is implemented. In this case, the use of SSL/TLS connections for writing to the $SYS topic would mitigate this threat.

10. Weak Credential Transit [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a

message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice.

Justification: While Client IDs are sent over the Completed Log entry, no user credentials are included in the log data.

11. Potential Excessive Resource Consumption for Log Publish Requests or Broker Logs [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: Does Log Publish Requests or Broker Logs 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: Memory and resource control is used in mosquitto.c, and logging.c to prevent the application from running out of control in system memory. All utilized memory for an individual log entry is immediately freed once the request is completed, which sets a very low requirement for memory when handling bulk amounts of logging.

12. Data Flow Completed Log Entry Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Completed log entries are sent one way and there is no response from the Broker Log that the entry was received.

13. Data Store Inaccessible [State: Mitigation Implemented] [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: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

14. Risks from Logging [State: Not Applicable] [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 data is ever read from the log files by any mosquitto processes. Log data is only placed in the log files, not read. Because of this, this threat is not applicable.

15. Lower Trusted Subject Updates Logs [State: Needs Investigation] [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: When logging is set to write to the $SYS topic, anyone who's access is not prevented by an ACL can write to the $SYS topic, which would effectively include writing to the logs. When logging is set to write to a file, the file is locked from writing by the processes for the broker, and is only written to by the user running the broker (either the mosquitto user, or root)

16. Data Logs from an Unknown Source [State: Needs Investigation] [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: When logs are written to the $SYS topic, ACLs must be implemented to only allow the broker system to write to the logs, otherwise anyone that can post to the broker can post to the logs. When logs are written to a file destination, only the broker service can write to the logs from within the mosquitto lifecycle. Outside of mosquitto, anyone can write to or read from the log files that has write access to the log file. Writing to the log file is locked while the broker is operational though.

17. Insufficient Auditing [State: Needs Investigation] [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: The entire post and subscribe request and ping lifecycles can be logged if enabled. This includes initial request from the client, authentication and validation by credential files and ACLs, dissemination of the post if acceptable, and returned acknowledgement of the post if successful or returned denial if post is not acceptable. Currently post message data is not stored in the logs, which would be very beneficial if under scrutiny.

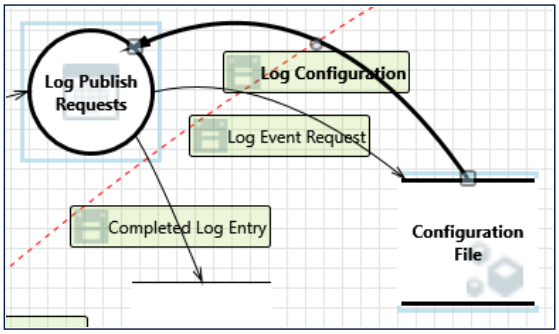
18. Potential Weak Protections for Audit Data [State: Needs Investigation] [Priority: High]

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: There currently is no reference monitor for the logs, or control flow of reading and writing for the broker logs.

Interaction: Log Configuration



19. Spoofing the Log Publish Requests Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Log Publish Requests may be spoofed by an attacker and this may lead to information disclosure by Configuration File. Consider using a standard authentication mechanism to identify the destination process.

Justification: ACLs and dual factor authentication with keys can prevent the broker logs from being spoofed.

20. Spoofing of Source Data Store Configuration File [State: Not Applicable] [Priority: High]

Category: Spoofing

Description: Configuration File may be spoofed by an attacker and this may lead to incorrect data delivered to Log Publish Requests. Consider using a standard authentication mechanism to identify the source data store.

Justification: Since the config file only can be accessed from the host machine for the broker, Operating System level authentication and authorization can prevent this from being an issue. Since this is outside the scope of the OSS, this is marked Not Applicable

21. Potential Data Repudiation by Log Publish Requests [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: Log Publish Requests 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: (When Authentication is enabled) the broker config already handles rejecting post requests that are missing data from the original post request. This rejection is also written to the logs as an audit for security or troubleshooting practices.

22. Weak Access Control for a Resource [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Improper data protection of Configuration File can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: Configuration file can only be read by users logged in to the host machine of the broker, config files cannot be read from within the Mosquitto Broker instance

23. Potential Process Crash or Stop for Log Publish Requests [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: Log Publish Requests crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: If a reference from the config to the logging request is interrupted, the error would be logged in the Broker logs and exited with an error message. The Publish request would continue without an error message since this is internal to the broker, but the error message would be logged.

24. Data Flow Binary Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: If a reference from the config to the logging request process is interrupted, the error would be logged in the Broker logs and exited with an error message. The Publish request would continue without an error message since this is internal to the broker, but the error message would be logged.

25. Data Store Inaccessible [State: Mitigation Implemented] [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: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

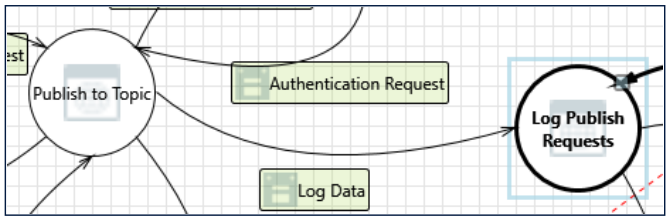
26. Log Publish Requests May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Configuration File may be able to remotely execute code for Log Publish Requests.
Justification: The memory management used in mosquitto.c, handle_publish.c, and handle_subscribe.c prevent Buffer Overflow attacks, and sanitize the input data when being published or requested to subscribe.

27. Elevation by Changing the Execution Flow in Log Publish Requests [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Log Publish Requests in order to change the flow of program execution within Log Publish Requests to the attacker's choosing.
Justification: This process is unavaible to users accessing from the broker, and is only available to users logged into the host system of the broker.

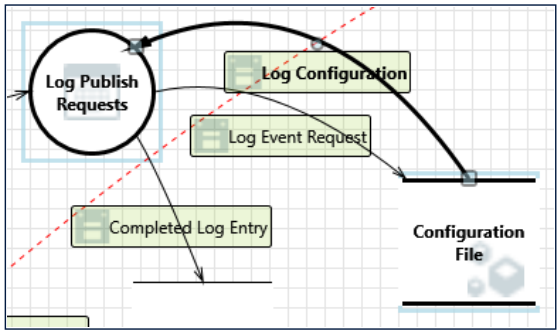
Interaction: Log Data



28. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: Log Publish Requests may be able to impersonate the context of Publish to Topic in order to gain additional privilege.
Justification: Currently all internal processes for a mosquitto broker instance run under the same user, so there are no additional privileges to be gained between these two processes.

Interaction: Log Event Request



29. Spoofing the Log Publish Requests Process [State: Not Applicable] [Priority: High]

Category: Spoofing

Description: Log Publish Requests may be spoofed by an attacker and this may lead to unauthorized access to Configuration File. Consider using a standard authentication mechanism to identify the source process.

Justification: Since the config file only can be accessed from the host machine for the broker, Operating System level authentication and authorization can prevent this from being an issue. Since this is outside the scope of the OSS, this is marked Not Applicable

30. Spoofing of Destination Data Store Configuration File [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Configuration File may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Configuration File. Consider using a standard authentication mechanism to identify the destination data store.

Justification: Logging of the reloading of the config file can flag this sort of issue and prevent it from going undetected.

31. The Configuration File Data Store Could Be Corrupted [State: Needs Investigation] [Priority: High]

Category: Tampering

Description: Data flowing across Log Event Request may be tampered with by an attacker. This may lead to corruption of Configuration File. Ensure the integrity of the data flow to the data store.

Justification: Integrity checking of the config file before the config is reloaded by a SIGHUP comand can prevent this sort of attack. (Config tampering would not take affect until the config is reloaded)

32. Data Store Denies Configuration File Potentially Writing Data [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: Configuration File 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: Configuration file data can be rewritten, but cannot take effect without the reloading of the config through a SIGHUP command which is logged by the broker and by (most) host operating systems.

33. Data Flow Sniffing [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Data flowing across Log Event Request 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: When writing to a Log file, log data can only be read by users logged in to the host machine of the broker, log files cannot be read from within the Mosquitto Broker instance. When writing to the $SYS topic, all log data can be read by any users that aren't blocked from an ACL that is implemented.

34. Weak Credential Transit [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice.

Justification: No login credentials are included in the Log data written to either a log file or the $SYS topic. This request only pings for the configuration

data of the logs.

35. Potential Excessive Resource Consumption for Log Publish Requests or Configuration File [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: Does Log Publish Requests or Configuration File 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: Memory and resource control is used in mosquitto.c, handle_publish.c and handle_subscribe.c to prevent the application from running out of control in system memory. All utilized memory for an individual post is immediately freed once the request is completed, which sets a very low requirement for memory when handling bulk amounts of messages.

36. Data Flow Binary Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: If a reference to the config from the logging request is interrupted, the error would be logged in the Broker logs and exited with an error message. The Publish request would continue without an error message since this is internal to the broker, but the error message would be logged.

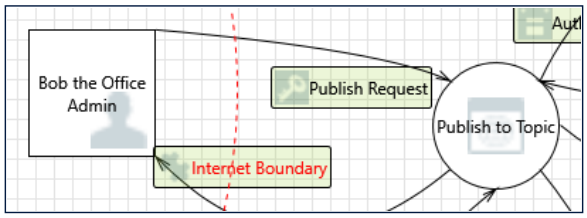
37. Data Store Inaccessible [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

Interaction: Publish Request



38. Spoofing the Bob the Office Admin External Entity [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Bob the Office Admin may be spoofed by an attacker and this may lead to unauthorized access to Publish to Topic. Consider using a standard authentication mechanism to identify the external entity.

Justification: Two level authentication with Access Control Lists limited by IP address and Public keys would prevent this type of attack

39. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege

Description: Publish to Topic may be able to impersonate the context of Bob the Office Admin in order to gain additional privilege.

Justification: Validation of Input by mosquitto.c combined with ACLs for priviliged topics and pre-shared keys for dual factor authentication can mitigate this threat.

40. Potential Data Repudiation by Publish to Topic [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: Publish to Topic 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: When enabled, logging of the post data, post request, post user, post connection and post acknowledgement/denial are tracked, which would audit the source, time, and summary of the received data.

41. Potential Process Crash or Stop for Publish to Topic [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: Publish to Topic crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: ACLs limiting publish and subscribe access, combined with limiting payload sizes with the MQTT protocol and managing memory usage and disposal through handle_publish.c and memory_mosq.c can mitigate this from becoming an issue.

42. Data Flow Publish Request Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: If a QoS level is set for the message being published, then the message can be resent automatically until a connectionack response is brought back from the broker.

43. Publish to Topic May be Subject to Elevation of Privilege Using Remote Code Execution [State: Needs Investigation] [Priority: High]

Category: Elevation Of Privilege
Description: Bob the Office Admin may be able to remotely execute code for Publish to Topic.
Justification: Input sanitization and validation, combined with dual factor authentication and ACLs can mitigate this threat

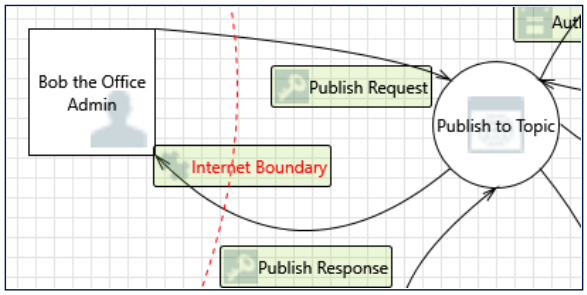
44. Elevation by Changing the Execution Flow in Publish to Topic [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Publish to Topic in order to change the flow of program execution within Publish to Topic to the attacker's choosing.
Justification: The front door of mosquitto immediately determines if the connection request is to publish or subscribe to topics. If the request does not match one of these parameters, the request packets are immediately dropped, which would prevent the ability of the attacker to access different processes within the broker implementation.

45. Cross Site Request Forgery [State: Mitigation Implemented] [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: When implemented, mosquitto still requires authentication with a username and password in every publish and subscribe request, which makes this type of attack not possible.

Interaction: Publish Response



46. Spoofing of the Bob the Office Admin External Destination Entity [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Bob the Office Admin may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Bob the Office Admin. Consider using a standard authentication mechanism to identify the external entity.

Justification: Two level authentication with Access Control Lists that limit by IP address and Public keys would prevent this type of attack

47. External Entity Bob the Office Admin Potentially Denies Receiving Data [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: Bob the Office Admin 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: When enabled, logging of the post acceptance/denial and acknowledgement/refusal are logged to act as an audit for this type of issue.

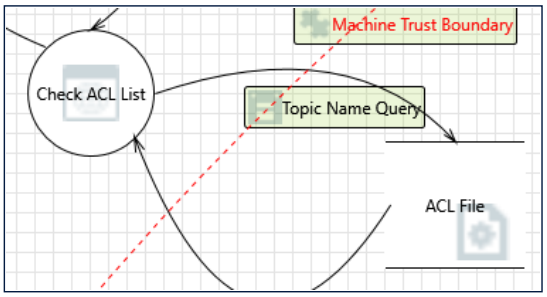
48. Data Flow Publish Response Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: If a QoS level is set for the message being published, then the message can be resent by the Office Admin automatically until a connectionack response is brought back from the broker.

Interaction: Topic Name Query



49. Spoofing of Destination Data Store ACL File [State: Not Applicable] [Priority: High]

Category: Spoofing

Description: ACL File may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of ACL File. Consider using a standard authentication mechanism to identify the destination data store.

Justification: Since ACL files only can be accessed from the host machine for the broker, Operating System level authentication and authorization can prevent this from being an issue.

50. Potential Excessive Resource Consumption for Check ACL List or ACL File [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: Does Check ACL List or ACL File 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: Memory and resource consumption are handled primarily through memory_mosq.c and throughout mosquitto.c. All publish requests managed through mosquitto.c and handle_publish.c free the used memory resource as soon as the publish request is either completed or errors out. MQTT limits on payload size (enforced through mosquitto)

51. Data Store Inaccessible [State: Mitigation Implemented] [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: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

52. Data Flow Topic Name Query Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

53. Weak Credential Transit [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice.

Justification: Credentials are transported as salted hashes through the processes, so if credentials were in this request, they would be indeciperable.

54. Data Flow Sniffing [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Data flowing across Topic Name Query 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: This process is closed off to outside access, and is only available to people logged into the host system of the broker.

55. Data Store Denies ACL File Potentially Writing Data [State: Needs Investigation] [Priority: High]

Category: Repudiation

Description: ACL File 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: Currently there is no process for writing new data to the ACL, they are written outside of the program and updates are applied when the config is reloaded. Currently I don't believe there is any logging to the changes of data within the ACL

56. The ACL File Data Store Could Be Corrupted [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: Data flowing across Topic Name Query may be tampered with by an attacker. This may lead to corruption of ACL File. Ensure the integrity of the data flow to the data store.

Justification: Validation of input by the process that checks the ACL and hashing of the ACL file values can mitigate this attack.

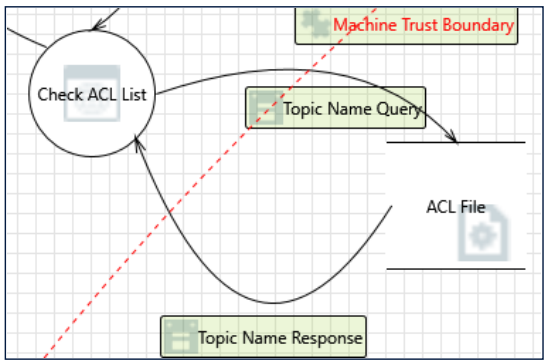
57. Spoofing the Check ACL List Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Check ACL List may be spoofed by an attacker and this may lead to unauthorized access to ACL File. Consider using a standard authentication mechanism to identify the source process.

Justification: Machine level trust boundaries with multiple users handling processes can mitigate this threat.

Interaction: Topic Name Response



58. Spoofing of Source Data Store ACL File [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: ACL File may be spoofed by an attacker and this may lead to incorrect data delivered to Check ACL List. Consider using a standard authentication mechanism to identify the source data store.
Justification: Two level authentication with Access Control Lists and Public keys would prevent this type of attack

59. Weak Access Control for a Resource [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Improper data protection of ACL File can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification: No users can access ACL files from within the mosquitto broker. Only users with access to the host system of the broker can access ACL files.

60. Elevation by Changing the Execution Flow in Check ACL List [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Check ACL List in order to change the flow of program execution within Check ACL List to the attacker's choosing.
Justification: This process is unavaible to users accessing from the broker, and is only available to users logged into the host system of the broker.

61. Check ACL List May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: ACL File may be able to remotely execute code for Check ACL List.
Justification: Input validation by the config checking the ACL file prevents invalid data from being read by the host system.

62. Data Store Inaccessible [State: Mitigation Implemented] [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: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

63. Data Flow Topic Name Response Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: If a response from the ACL is interrupted, and ACLs are outlined in the mosquitto instance config, then the connection attempt fails without and ACL check and an error is returned to Bob, the office admin.

64. Potential Process Crash or Stop for Check ACL List [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: Check ACL List crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: If the process for checking the ACL halts, the configuration specified in the mosquitto instance would fail due to the need of an ACL file location, which would then return an error message to the external interactor.

65. Potential Data Repudiation by Check ACL List [State: Not Applicable] [Priority: High]

Category: Repudiation
Description: Check ACL List 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: In order to not receive data from the other side of the trust boundary, that would mean a post topic was not defined, which would be handled as an error from the handle_publish.c file. There is no way the post would make it to this point in the processes without a post topic defined.

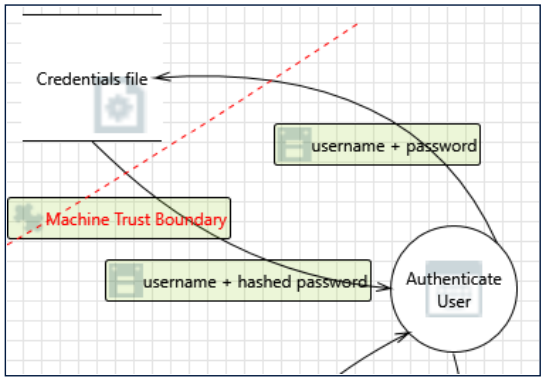
66. Spoofing the Check ACL List Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Check ACL List may be spoofed by an attacker and this may lead to information disclosure by ACL File. Consider using a standard

authentication mechanism to identify the destination process.

Justification: Machine level trust boundaries with multiple users handling processes can mitigate this threat.

Interaction: username + hashed password



67. Spoofing of Source Data Store Credentials file [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Credentials file may be spoofed by an attacker and this may lead to incorrect data delivered to Authenticate User. Consider using a standard authentication mechanism to identify the source data store.

Justification: Password hashing of the credential file can prevent credential spoofing from taking place.

68. Weak Access Control for a Resource [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Improper data protection of Credentials file can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: Credential files are currently inaccessible to connected clients from within the Mosquitto broker. Access to credential files is only authorized to users that logged in to the host system of the broker. Passwords are also salted hashes within the file that are unreadable outside of the mosquitto broker.

69. Spoofing the Authenticate User Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Authenticate User may be spoofed by an attacker and this may lead to information disclosure by Credentials file. Consider using a standard authentication mechanism to identify the destination process.

Justification: Password hashing of the credential file can prevent credential spoofing from taking place.

70. Potential Data Repudiation by Authenticate User [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: Authenticate User 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: (When Authentication is enabled) the broker config already handles rejecting post requests that are missing authentication data from the original post request.

71. Potential Process Crash or Stop for Authenticate User [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: Authenticate User crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: If authentication is enabled and the authentication process halts, the connection attempt is invalidated, and an error message is returned to the external interactor.

72. Data Flow username + hashed password Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

73. Data Store Inaccessible [State: Mitigation Implemented] [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: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.

74. Authenticate User May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: Credentials file may be able to remotely execute code for Authenticate User.

Justification: Data within credential files is stored in a salted hash that has no code to be executed from. This combined with the memory management used in mosquitto.c, handle_publish.c, and handle_subscribe.c prevent Buffer Overflow attacks, and sanitize the input data when being published or requested to subscribe.

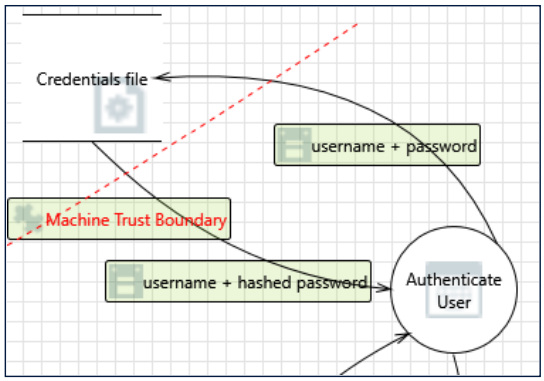
75. Elevation by Changing the Execution Flow in Authenticate User [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Authenticate User in order to change the flow of program execution within Authenticate User to the attacker's choosing.

Justification: This process is unavaible to users accessing from the broker, and is only available to users logged into the host system of the broker.

Interaction: username + password



76. Spoofing of Destination Data Store Credentials file [State: Not Applicable] [Priority: High]

Category: Spoofing

Description: Credentials file may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Credentials file. Consider using a standard authentication mechanism to identify the destination data store.

Justification: Since ACL files only can be accessed from the host machine for the broker, Operating System level authentication and authorization can prevent this from being an issue. Since this is outside the scope of the OSS, this is marked Not Applicable

77. Weak Credential Storage [State: Mitigation Implemented] [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: This is currently implemented by the mosquitto password utility, usernames are stored, but a salted hash of the password is stored to be used when authenticating users.

78. Potential Excessive Resource Consumption for Authenticate User or Credentials file [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: Does Authenticate User or Credentials file 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: Memory and resource control is used in mosquitto.c to prevent the application from running out of control in system memory. All utilized memory for an individual log entry is immediately freed once the request is completed, which sets a very low requirement for memory when handling bulk amounts of logging. It is currently unknown if resource requests deadlock credential files or not.

79. Spoofing the Authenticate User Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Authenticate User may be spoofed by an attacker and this may lead to unauthorized access to Credentials file. Consider using a standard authentication mechanism to identify the source process.
Justification: Password hashing of the credential file can prevent credential spoofing from taking place.

80. The Credentials file Data Store Could Be Corrupted [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Data flowing across username + password may be tampered with by an attacker. This may lead to corruption of Credentials file. Ensure the integrity of the data flow to the data store.
Justification: Validation of input by the authentication process and hashing of the credentials file values can mitigate this attack.

81. Data Store Denies Credentials file Potentially Writing Data [State: Needs Investigation] [Priority: High]

Category: Repudiation
Description: Credentials file 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: The writing of data to the Credentials file is currently done by a utility that is only accessible on the host machine of the broker. Currently it is not known if logging of credential changes takes place.

82. Data Flow Sniffing [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Data flowing across username + password 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: Credentials are hashed before storage which protects them from being deciphered when sniffed by an attacker.

83. Weak Credential Transit [State: Needs Investigation] [Priority: High]

Category: Information Disclosure
Description: Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice.
Justification: Credentials are stored as salted hashes, though it is currently unknown how strong the encryption is for said credentials.

84. Data Flow username + password Is Potentially Interrupted [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: A lack of a response from the credential file to the process would result in the processing reporting an error and returning authentication errors to Bob the Ofiice Admin

85. Data Store Inaccessible [State: Mitigation Implemented] [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: Requiring authentication using separate user accounts on either side of the trust boundary that are specific to the broker, can prevent external agents from potentially locking or preventing access to the given data store.