DAT 235

IoT & Security

Prosjekt DAT235 - Del 1

Author:
Bendik Egenes Dyrli
Nikolai Kjærem Ellingsen
Jens Martin Håsæther
Mathias Solheim Jansen

Supervisor:
Geir Myrdahl Køien

2017

Contents

List of Figures

Chapter 1

Introduction

In this assignment we were tasked with putting our planned system to test by implementing a mock setup of it to see if it would actually work and conduct tests on the system.
The goal of doing this was creating a revised plan that could serve to point out improvements to the ”new” or revised plan.
This allows us to check for errors and other unforeseen elements that could ultimately improve the end installation, and reduce post-installation problems.

1.1 MASSE ROT - skal fjernes

- **Threat Model (Mathias)**
  - Mye tekst
  - forbedringer?

  - Funktionalitet?
  - Sikkerhet?
- **DONE and DONE**
  - Create the code (Bendik)
  - Flytskjema (Jens)
  - Sikkerhetstegning (Nikolai)
  - Diverse Endringer (Mathias)
  - Timer? /copypasta? (Nikolai)
  - Forbedringer? (Delvis utsatt)
  - Fant vi noen bugs? (Bendik)
  - tthere is no such things as bugs, only stupid hoomans
  - **Dokumentèr Design (Jens)**
    - Forklar flytskejma etc
  - Revidert Design Dokument? (Delvis Utsatt)

Chapter 2

Pi setup

The pis were installed with raspian streach lite september 2017.
Once booted, the default password was changed and hostname changed to something more identifiable.
To further enchance security, we decided to use serial connection instead of WiFi.
We cut power to the Wifi antenna by running:

```
iwconfig wlan0 txpower off
```

Then we started the bluetooth setup betwen the broker and tanks.
We ran the following commands in order:

```
bluetoothctl

agent on

default-agent

scan on #Only needed to see other devices

pair XXX #Where XXX is the devices MAC address

discoverable on #Only needed to be seen by other devices

trust XXX #Makes the devices pair automatically
```

An alternative to this is paring them together automaticaly by MAC address, but this was deemed too unsafe:

```
/etc/rc.local

echo "connect AA:BB:CC:DD:EE:FF \nquit" | bluetoothctl
```

Last but not least, we installed paho with

```
pip install paho-mqtt
```

At this point we were ready to upload our python scripts and start testing.

2.1 Documenting Design

Our current setup assumes that hardware(water intake, and the amount) is not handled by us.

We start the system off by opening the input valve to allow water to flow inside the system.
Once the water has reached a desired level the input valve for the water will then be shut off.
Once the input valve has been shut off we will then activate the dispenser to add the mix/-solvent to the mix.
It will then proceed to heat the mix up to a desired temperature (in this case between 80 and 90 degrees).
When the mixture has reached this temperature it will start a timer of 100 seconds.
While this timer goes the heater system will make sure that the temperature remains in the desired range.
Once the timer then has reached 0 it will open the output valve which will run until the boiling tank is empty.

There will then be a reset of the code to allow the process to start over again.

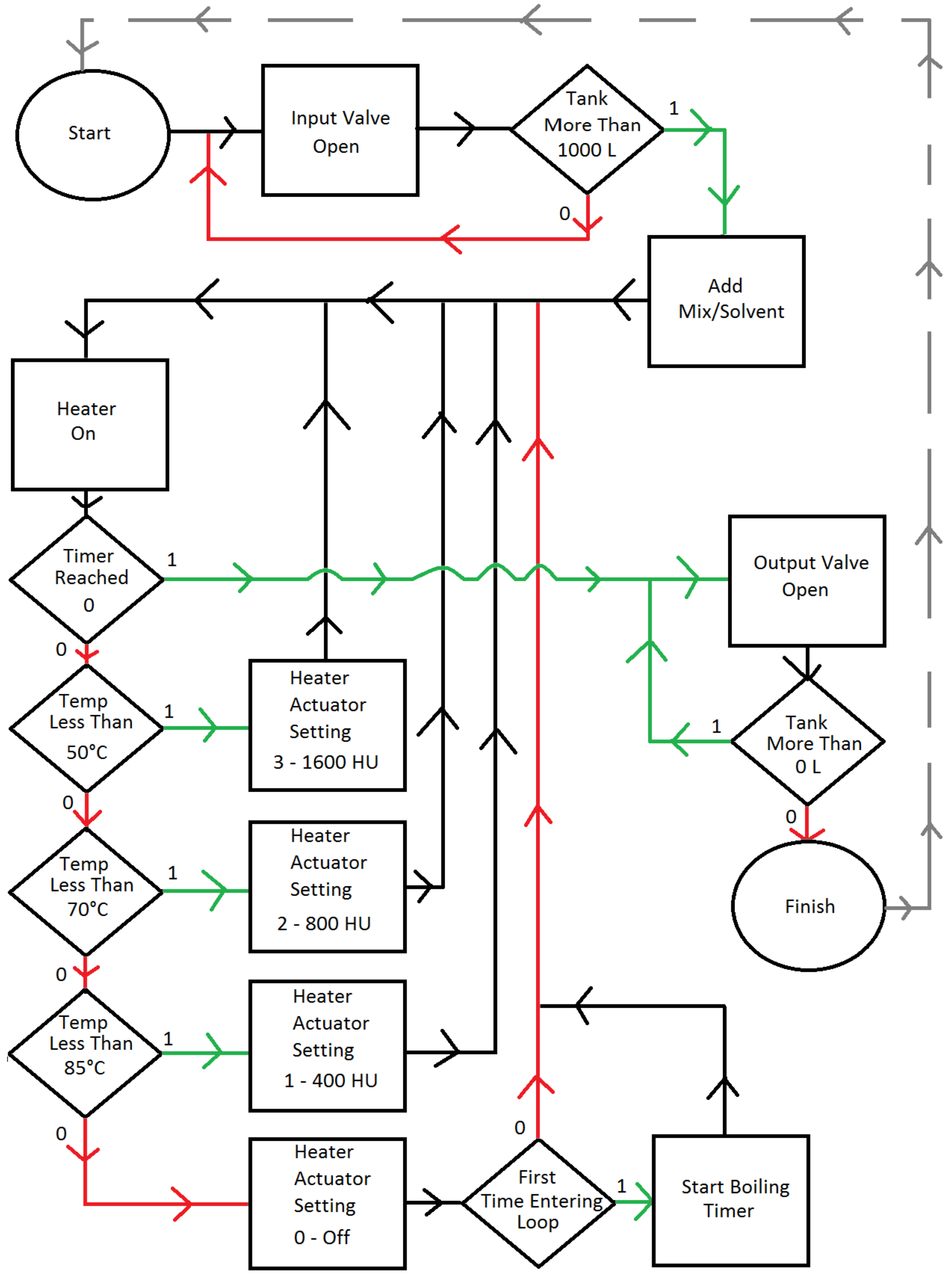


Figure 2.1: Flowchart for the Boiling Process

Chapter 3

Security outline

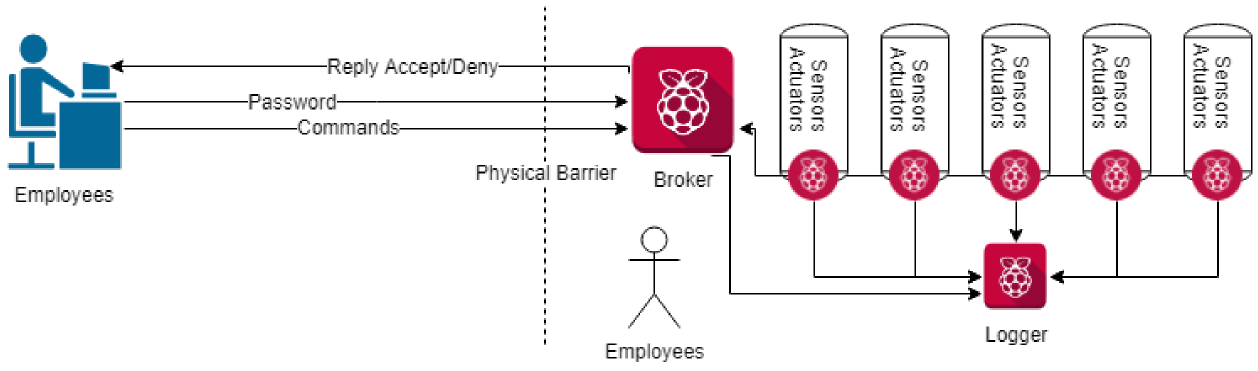


Figure 3.1: Security Outline

The system will be based on Raspian and communicate with each other locally with Bluetooth 4.1.
The broke will communicate across the internet with employees.
Inputs will be sanitized and since the networks will mostly be kept locally, information disclosure and spoofing will be partially mitigated.

Physical tampering will be mitigated by the business owners and will for the most part be out of scope in this task.

In general the raspberry pis will be locked in metal containers attached to the tanks.
The logger and broker will also be locked and contained within near vicinity to the other pis to enable Bluetooth connection.

The tanks themselves will be secured by employees during working hours and with cameras connected to securitas or similar at night.

As a continuation of the security measures, training will be provided to employees.
This will reduce potential user error as well as provide faster error handling.

Since only employees will have access to the network and keys to interface from outside, this is of great importance.
Likewise denial of service will be partially mitigated due to a list of trusted sources being allowed to interface with the broker.

Chapter 4

Threat modeling

The full report can be found in the TMProsjektPhase1 appendix.
In this threat model we have discussed most of the threats to our system.
As is discussed in the improvements part we had not enough time to finish every aspect of the threat modeling tool, ideally things like authorization would be implemented. [Appendix G]

When it comes to the design it is suggested to look through the Threat model report. In the report everything is set to priority high, however this is a bug, and if needed can be seen in the actual tm file.[Appendix G]

In short terms, the system should be able to deffend its self agains all most every type of attack, I/O is sanitized, Authorization should have been implemented.
Everything is sent over secure connections. In addition to this we would also want to encrypt all data before its leaving the device.
However due to short time this could not be done.

Chapter 5

Bugs & Improvement

From the phase 0 and the assigned task we have done a few changes.
We have changed the design to run everything on one computer this is because of little time.
The group had only a weekend to do the task due to a bigger project in another class.
In a real situation this would not have been possible and the group would have to use real sensors and controllers like raspberry Pi, the program however has been written in Python, so the code could be transferred to a raspberry Pi.

In the threat model in Phase 0, we wanted to implement a type of authorization like user login.
This is to prevent unwanted people or programs to take control and/or destroy the system.
In a real setup this would have been a must and if we had time enough we would ofcourse have implemented it as well.
There was not enough time to implement https either, which would have been done in a real situation.
In this project we assumed that all actuators are working under ideal conditions, with no "transfer states".
For an example a heater that is turned on would give the preset heating units instantly, and that the input valve stops directly at the preset amount of liters.
In a real world when closing the input valve there would be a little bit extra water in the tank.
The heating unit would also take some time to reach the desired heating effect.
In a real situation this would have to be taken into accountability.

We had planned that this project would take around 30 man-hours, but because of our short amount of time, this has been drastically reduced.
If we had implemented everything we had in mind, we would probably be a lot closer to the planned time.

Chapter 6

Setup

This setup is estimated to take 30 hours divided among 4 people.
Which will be completed within 7 days with the schedule below.
As seen below, the setup should only take 5 days.
The reasoning behind this is to allocate time to write the report.
It will also functioning as a buffer to potential mishaps and/or complications.

Systems may be dropped or simplified if deemed too time consuming within the projects time frame.
This will be included in Phase 1.

The physical security of the hardware will be included if time permits it, but this is deemed out of scope for now.

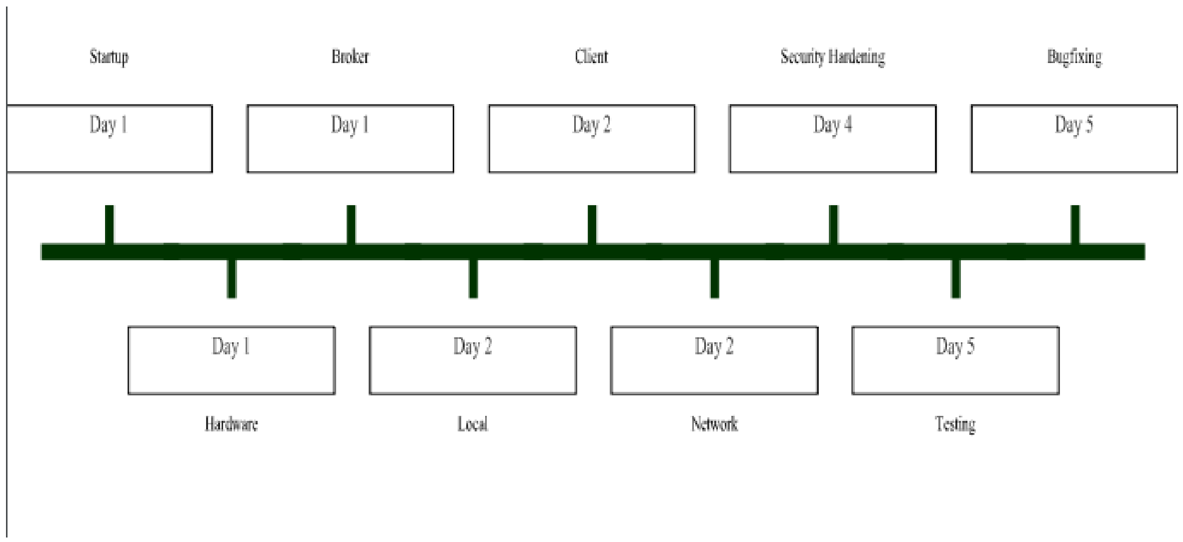


Figure 6.1: Timeline

Chapter 7

Conclusion

Our system functions as we interpreted the task.
As mentioned in Phase 0, many systems were dropped or modified to function within the little time we had left.
Some corners were cut, but as a whole we are happy with the results.
If we had more time, we would have followed our Phase 0 more closely.
Actual real life testing would be one of the main points we would like to address.
Overall thanks to diligent work, we believe that our system meets the requirements of the task.

Appendix A

Appendix

A.1

```
from actuator import heater, inputValve, outputValve, tankCapacity, dispenser
from sensor import levelIndicator, temperaturSensor
from misc import config

def Testing():
        from client import client
        print "\n### TESTING actuator ###"
        client("\n")
        actuatorTest = [heater(3), inputValve(True), inputValve(), outputValve(True), outpu
        for key in actuatorTest:
                client(key)

        print "\n### TESTING sensor ###"
        client("\n")
        sensorTest = [levelIndicator(420), temperaturSensor(-100)]
        for key in sensorTest:
                client(key)

if __name__ == '__main__':
        Testing()
```

A.2

```
from misc import config
import paho.mqtt.client as mqtt

def client(msg):
        client = mqtt.Client()
        client.connect(config("MQTT","Address"),config("MQTT","Port"),config("MQTT","Timeo
        client.publish("topic/test", msg)
        client.disconnect()
```

A.3

```
import paho.mqtt.client as mqtt
from misc import config

def server():
        def on_connect(client, userdata, flags, rc):
                print("Connected with result code "+str(rc))
                client.subscribe("topic/test")

        def on_message(client, userdata, msg):
                print msg.payload.decode()

        client = mqtt.Client()
        client.connect(config("MQTT","Address"), config("MQTT","Port"), config("MQTT","Tim
        client.on_connect = on_connect
        client.on_message = on_message
        client.loop_forever()

if __name__ == '__main__':
        print ("\n### Starting Server ###")
        server()
```

A.4

```
from misc import checkInputs
from sys import exit

import logging
logging.basicConfig(level=logging.INFO)

def heater(heat):
        logger = logging.getLogger('HEATER')
        if checkInputs(heat, int,":HEATER"):
                heating = {0:'OFF', 1:'400', 2:'800', 3:'1600'}
                logger.info("{:s}".format([(key,value) for key, value in heating.items() i
                return("{:s}".format([(key,value) for key, value in heating.items() if heat

def inputValve(liquid=False):
        logger = logging.getLogger('INPUT')
        #TODO: 50liter per second
        if checkInputs(liquid, bool,":INPUT"):
                if liquid:
                        logger.info("Valve␣is␣Open")
                        return("Valve␣is␣Open")

                elif liquid is False:
                        logger.error("Valve␣is␣Closed")
                        return("Valve␣is␣Closed")

def outputValve(liquid=False):
        #TODO: 100 liter per second
        logger = logging.getLogger('OUTPUT')
        if checkInputs(liquid, bool,":OUTPUT"):
                if liquid:
                        logger.info("Valve␣is␣Open")
                        return("Valve␣is␣Open")

                elif liquid is False:
                        logger.error("Valve␣is␣Closed")
                        return("Valve␣is␣Closed")

def tankCapacity(capacity):
        logger = logging.getLogger('TANK_CAPACITY')
        if checkInputs(capacity, int,":TANK"):
                if capacity == 1200:
                        logger.info("Max␣capacity")
                        return("Max␣capacity")

                elif capacity in range(1000, 1101):
                        logger.info("Operational␣'Boiling'".format(capacity))
                        return("Operational␣'Boiling'".format(capacity))



                elif capacity in range(0,1000) or capacity in range(1101, 1200):
                        logger.info("{:d}␣Liters".format(capacity))
                        return("{:d}␣Liters".format(capacity))

                else:
                        logger.error("Please␣check␣tankCapacity␣sensor")
                        return("Please␣check␣tankCapacity␣sensor")

def dispenser(n, coldWater = True):
        logger = logging.getLogger('DISPENSER')
        if checkInputs(n, int,":DISPENSER"):
                if coldWater:
                        logger.info("Mixing")
                        return("Mixing")
        else:
                logger.error("Please␣check␣the␣dispenser")
                return("Please␣check␣the␣dispenser")
```

A.5

```
from misc import checkInputs
import logging

logging.basicConfig(level=logging.INFO)

def levelIndicator(level):
        logger = logging.getLogger('SENSOR')
        if checkInputs(level, int,":LEVEL"):
                if level in range(1200):
                        logger.info("Fill␣Level␣{0:d}␣Liters␣".format(level))
                        return("Fill␣Level␣{0:d}␣Liters␣".format(level))
                else:
                        logger.error("Please␣check␣Level␣Indicator␣Sensor")
                        return("Please␣check␣Level␣Indicator␣Sensor")

def temperaturSensor(SensorInput):
        SensorInput = int(SensorInput)
        logger = logging.getLogger('SENSOR')
        if checkInputs(SensorInput, int,":TEMPERATURE"):
                if SensorInput == int(-100):
                        logger.info("Empty␣Tank...")
                        return("Empty␣Tank...")

                elif int(SensorInput) in range(101):
                        logger.info("{:d}␣Celsius".format(SensorInput))
                        return("{:d}␣Celsius".format(SensorInput))

                else:
                    logger.error("Please check Temperature sensor")
                    return("Please check Temperature sensor")
```

A.6

```
{
        "MQTT":{
                "Address":"datapor.no",
                "Port":1883,
                "Timeout":60
        }
}
```

A.7

Threat Modeling Report

Created on 20.11.2017 14.32.53

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	4
Needs Investigation	4
Mitigation Implemented	118
Total	126
Total Migrated	0

Diagram: Diagram 1

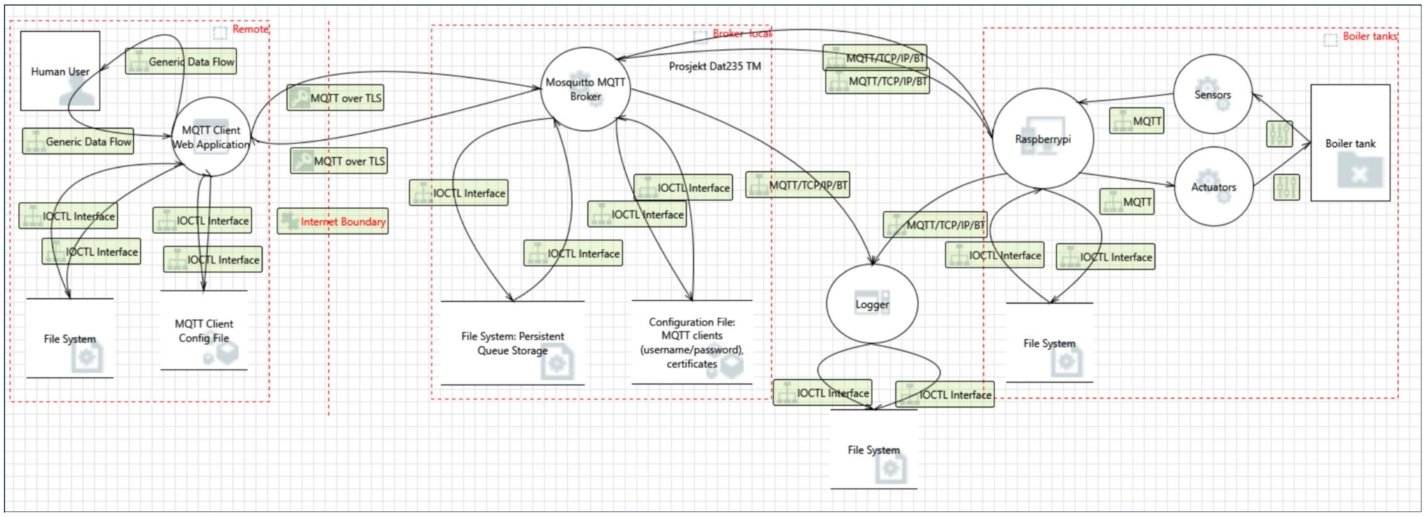


Diagram 1 Diagram Summary:

Not Started	0
Not Applicable	4
Needs Investigation	4
Mitigation Implemented	118
Total	126
Total Migrated	0

Threat(s) Not Associated With an Interaction:

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

Category:	Information Disclosure
Description:	Improper data protection of Device Storage can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification:	Authentication

2. Spoofing of Source Data Store Device Storage [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	Device Storage may be spoofed by an attacker and this may lead to incorrect data delivered to Paho Python MQTT Client. Consider using a standard authentication mechanism to identify the source data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

3. Potential Excessive Resource Consumption for Paho Python MQTT Client or Device Storage [State: Mitigation Implemented] [Priority: High]

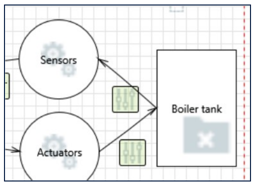
Category:	Denial Of Service
Description:	Does Paho Python MQTT Client or Device Storage take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification:	Resource management

4. Spoofing of Destination Data Store Device Storage [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing

Description: Device Storage may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Device Storage. Consider using a standard authentication mechanism to identify the destination data store.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction:



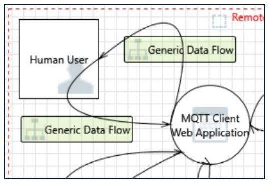
5. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Sensors may be able to impersonate the context of Boiler tank in order to gain additional privilege.
Justification: Authentication and filtering.

6. Spoofing the Boiler tank External Entity [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Boiler tank may be spoofed by an attacker and this may lead to unauthorized access to Sensors. Consider using a standard authentication mechanism to identify the external entity.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: Generic Data Flow



7. Spoofing the Human User External Entity [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Human User may be spoofed by an attacker and this may lead to unauthorized access to MQTT Client Web Application. Consider using a standard authentication mechanism to identify the external entity.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

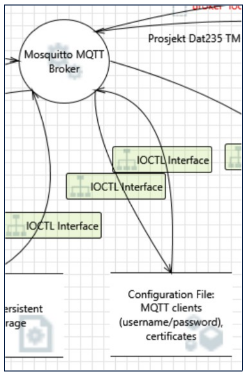
8. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'MQTT Client Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Sanitize inputs

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

Category: Elevation Of Privilege
Description: MQTT Client Web Application may be able to impersonate the context of Human User in order to gain additional privilege.
Justification: Authentication

Interaction: IOCTL Interface



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

Category: Information Disclosure
Description: Improper data protection of Configuration File: MQTT clients (username/password), certificates can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification: Authentication

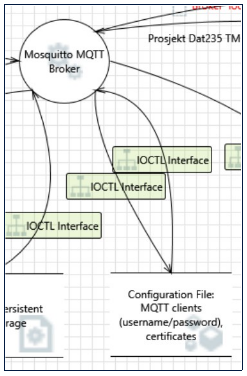
11. Spoofing of Source Data Store Configuration File: MQTT clients (username/password), certificates [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Configuration File: MQTT clients (username/password), certificates may be spoofed by an attacker and this may lead to incorrect data delivered to Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the source data store.

Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



12. Potential Excessive Resource Consumption for Mosquitto MQTT Broker or Configuration File: MQTT clients (username/password), certificates [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service

Description: Does Mosquitto MQTT Broker or Configuration File: MQTT clients (username/password), certificates 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: Resource management

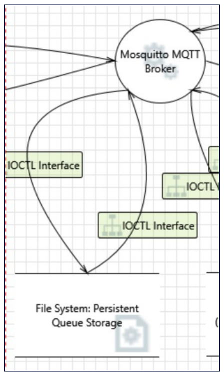
13. Spoofing of Destination Data Store Configuration File: MQTT clients (username/password), certificates [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: Configuration File: MQTT clients (username/password), certificates may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Configuration File: MQTT clients (username/password), certificates. Consider using a standard authentication mechanism to identify the destination data store.

Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



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

Category: Information Disclosure

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

Justification: Authentication

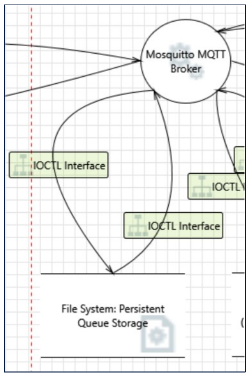
15. Spoofing of Source Data Store File System: Persistent Queue Storage [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: File System: Persistent Queue Storage may be spoofed by an attacker and this may lead to incorrect data delivered to Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the source data store.

Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



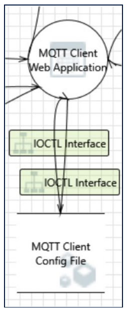
16. Potential Excessive Resource Consumption for Mosquitto MQTT Broker or File System: Persistent Queue Storage [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does Mosquitto MQTT Broker or File System: Persistent Queue 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:	Resource management

17. Spoofing of Destination Data Store File System: Persistent Queue Storage [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System: Persistent Queue Storage may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of File System: Persistent Queue Storage. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



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

Category:	Information Disclosure
Description:	Improper data protection of MQTT Client Config File can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification:	Authorization

19. Persistent Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'MQTT Client Web Application' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'MQTT Client Config File' inputs and output.
Justification:	Sanitize in-/outputs

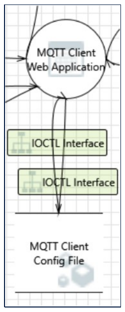
20. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'MQTT Client Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	Sanitize inputs

21. Spoofing of Source Data Store MQTT Client Config File [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	MQTT Client Config File may be spoofed by an attacker and this may lead to incorrect data delivered to MQTT Client Web Application. Consider using a standard authentication mechanism to identify the source data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



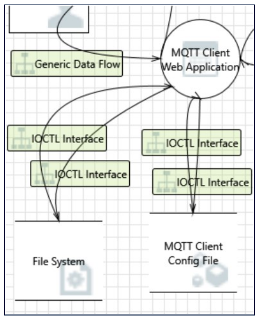
22. Potential Excessive Resource Consumption for MQTT Client Web Application or MQTT Client Config File [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does MQTT Client Web Application or MQTT Client Config 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:	Resource management

23. Spoofing of Destination Data Store MQTT Client Config File [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	MQTT Client Config File may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of MQTT Client Config File. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



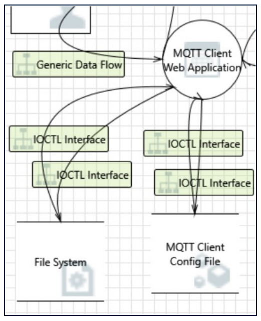
24. Spoofing of Destination Data Store File System [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of File System. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

25. Potential Excessive Resource Consumption for MQTT Client Web Application or File System [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does MQTT Client Web Application or File System 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:	Resource managament

Interaction: IOCTL Interface



26. Spoofing of Source Data Store File System [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System may be spoofed by an attacker and this may lead to incorrect data delivered to MQTT Client Web Application. Consider using a standard authentication mechanism to identify the source data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

27. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'MQTT Client Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.

Justification: Sanitize inputs

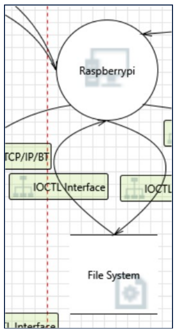
28. Persistent Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'MQTT Client Web Application' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'File System' inputs and output.
Justification:	Sanitize in-/outputs

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

Category:	Information Disclosure
Description:	Improper data protection of File System can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification:	Authentication

Interaction: IOCTL Interface



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

Category:	Information Disclosure
Description:	Improper data protection of File System can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification:	Only certain users will be able to read and write to files in the system, the files will also be stored in a encrypted way.

31. Persistent Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'Raspberrypi' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'File System' inputs and output.
Justification:	The Raspberrypi should sanitize both inputs and outputs

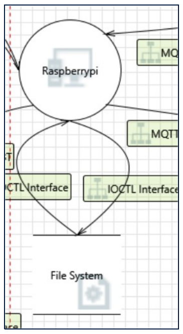
32. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'Raspberrypi' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	The Raspberry should sanitize input, the input should never come from a new source as well as that the input should be pretty much the same every time. except for the values

33. Spoofing of Source Data Store File System [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System may be spoofed by an attacker and this may lead to incorrect data delivered to Raspberrypi. Consider using a standard authentication mechanism to identify the source data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



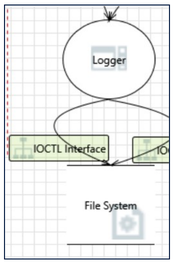
34. Potential Excessive Resource Consumption for Raspberrypi or File System [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does Raspberrypi or File System 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:	it is important with an efficient resource management and a maximum time so that it doesent get stuck in a deadlock.

35. Spoofing of Destination Data Store File System [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of File System. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



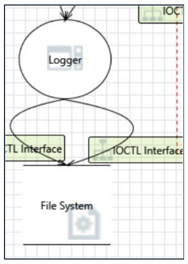
36. Potential Excessive Resource Consumption for Logger or File System [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does Logger or File System 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:	Resource management

37. Spoofing of Destination Data Store File System [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of File System. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: IOCTL Interface



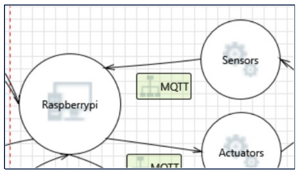
38. Potential Excessive Resource Consumption for Logger or File System [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Does Logger or File System 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:	Resource management

39. Spoofing of Destination Data Store File System [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	File System may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of File System. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: MQTT



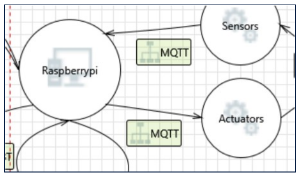
40. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	Raspberrypi may be able to impersonate the context of Sensors in order to gain additional privilege.
Justification:	Authentication and filtering.

41. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	The web server 'Raspberrypi' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	The Raspberry should sanitize input, the input should never come from a new source as well as that the input should be pretty much the same every time. except for the values

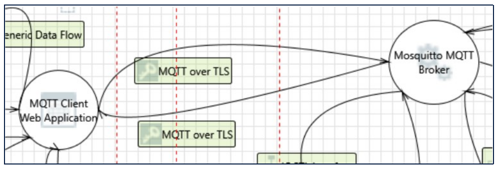
Interaction: MQTT



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

Category:	Elevation Of Privilege
Description:	Actuators may be able to impersonate the context of Raspberrypi in order to gain additional privilege.
Justification:	Authentication and filtering.

Interaction: MQTT over TLS



43. Spoofing the MQTT Client Web Application Process [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	MQTT Client Web Application may be spoofed by an attacker and this may lead to unauthorized access to Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the source process.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

44. Potential Data Repudiation by Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Repudiation
Description:	Mosquitto MQTT Broker 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:	Logging

45. Potential Process Crash or Stop for Mosquitto MQTT Broker [State: Not Applicable] [Priority: High]

Category:	Denial Of Service
Description:	Mosquitto MQTT Broker crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification:	restart?

46. Data Flow Generic Data Flow 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:	Redundancy

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

Category:	Elevation Of Privilege
Description:	Mosquitto MQTT Broker may be able to impersonate the context of MQTT Client Web Application in order to gain additional privilege.
Justification:	Authentication

48. Mosquitto MQTT Broker May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	MQTT Client Web Application may be able to remotely execute code for Mosquitto MQTT Broker.
Justification:	Disallow remote execution of abusable code

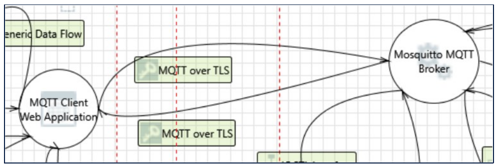
49. Elevation by Changing the Execution Flow in Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	An attacker may pass data into Mosquitto MQTT Broker in order to change the flow of program execution within Mosquitto MQTT Broker to the attacker's choosing.
Justification:	Authentication should easily solve this problem

50. 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:	Integrate additional piece of secret payload in authentication, alternatively user training

Interaction: MQTT over TLS



51. Spoofing the Mosquitto MQTT Broker Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Mosquitto MQTT Broker may be spoofed by an attacker and this may lead to unauthorized access to MQTT Client Web Application. Consider using a standard authentication mechanism to identify the source process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

52. Mosquitto MQTT Broker Process Memory Tampered [State: Needs Investigation] [Priority: High]

Category: Tampering
Description: If Mosquitto MQTT Broker is given access to memory, such as shared memory or pointers, or is given the ability to control what MQTT Client Web Application executes (for example, passing back a function pointer.), then Mosquitto MQTT Broker can tamper with MQTT Client Web Application. 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: not sure, needs further investigation

53. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'MQTT Client Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Sanitize inputs

54. Potential Data Repudiation by MQTT Client Web Application [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: MQTT Client Web Application 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: Logging

55. Potential Process Crash or Stop for MQTT Client Web Application [State: Not Applicable] [Priority: High]

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

56. Data Flow Generic Data Flow 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: Redundancy

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

Category: Elevation Of Privilege
Description: MQTT Client Web Application may be able to impersonate the context of Mosquitto MQTT Broker in order to gain additional privilege.
Justification: Authentication

58. MQTT Client Web Application May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Mosquitto MQTT Broker may be able to remotely execute code for MQTT Client Web Application.
Justification: Disallow remote execution of abusable code

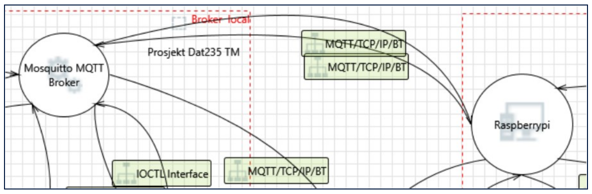
59. Elevation by Changing the Execution Flow in MQTT Client Web Application [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into MQTT Client Web Application in order to change the flow of program execution within MQTT Client Web Application to the attacker's choosing.
Justification: Authentication should easily solve this problem

60. 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: Integrate additional piece of secret payload in authentication, alternatively user training

Interaction: MQTT/TCP/IP/BT



61. Potential Process Crash or Stop for Paho Python MQTT Client [State: Not Applicable] [Priority: High]

Category: Denial Of Service
Description: Paho Python MQTT Client crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

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

Category: Information Disclosure
Description: Data flowing across MQTT/TCP/IP 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: Encryption

63. Potential Data Repudiation by Paho Python MQTT Client [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Paho Python MQTT Client 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: Logging

64. Mosquitto MQTT Broker Process Memory Tampered [State: Needs Investigation] [Priority: High]

Category: Tampering
Description: If Mosquitto MQTT Broker is given access to memory, such as shared memory or pointers, or is given the ability to control what Paho Python MQTT Client executes (for example, passing back a function pointer.), then Mosquitto MQTT Broker can tamper with Paho Python MQTT Client. 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: Pass data?, not sure, needs further investigation

65. Potential Lack of Input Validation for Paho Python MQTT Client [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Data flowing across MQTT/TCP/IP may be tampered with by an attacker. This may lead to a denial of service attack against Paho Python MQTT Client or an elevation of privilege attack against Paho Python MQTT Client or an information disclosure by Paho Python MQTT Client. 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: Input validation

66. Spoofing the Paho Python MQTT Client Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Paho Python MQTT Client may be spoofed by an attacker and this may lead to information disclosure by Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the destination process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

67. Spoofing the Mosquitto MQTT Broker Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Mosquitto MQTT Broker may be spoofed by an attacker and this may lead to unauthorized access to Paho Python MQTT Client. Consider using a standard authentication mechanism to identify the source process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

68. Data Flow Generic Data Flow 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: Redundancy

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

Category: Elevation Of Privilege
Description: Paho Python MQTT Client may be able to impersonate the context of Mosquitto MQTT Broker in order to gain additional privilege.
Justification: Authentication

70. Paho Python MQTT Client May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Mosquitto MQTT Broker may be able to remotely execute code for Paho Python MQTT Client.
Justification: Disallow remote execution of abusable code

71. Elevation by Changing the Execution Flow in Paho Python MQTT Client [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Paho Python MQTT Client in order to change the flow of program execution within Paho Python MQTT Client to the attacker's choosing.
Justification: Authentication should easily solve this problem

72. 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: Integrate additional piece of secret payload in authentication, alternatively user training

73. Elevation by Changing the Execution Flow in Raspberrypi [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Raspberrypi in order to change the flow of program execution within Raspberrypi to the attacker's choosing.
Justification: An Authentication mechanism should easily solve this problem.

74. Raspberrypi May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege

Description: Mosquitto MQTT Broker may be able to remotely execute code for Raspberrypi.
Justification: Authentication with a strong secure login, so that only certain users is able to execute a code remotely. Anyone else should be denied.

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

Category: Elevation Of Privilege
Description: Raspberrypi may be able to impersonate the context of Mosquitto MQTT Broker in order to gain additional privilege.
Justification: Authentication and filtering.

76. Data Flow MQTT/TCP/IP 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: Redundancy make sure that only one device is allowed to connect a certain times in a certain timeline

77. Potential Process Crash or Stop for Raspberrypi [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: Raspberrypi crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: if it happens a team has to be sent to make sure that the raspberry is still alive and potentially restarting it or swap it.

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

Category: Information Disclosure
Description: Data flowing across MQTT/TCP/IP/BT 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: The data will be encrypted before sending it, therefor this will not be a problem

79. Potential Data Repudiation by Raspberrypi [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Raspberrypi 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: With a proper logging system which records both time and data will atleast pinpoint any doubts.

80. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'Raspberrypi' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: The Raspberry should sanitize input, the input should never come from a new source as well as that the input should be pretty much the same every time. except for the values

81. Potential Lack of Input Validation for Raspberrypi [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Data flowing across MQTT/TCP/IP/BT may be tampered with by an attacker. This may lead to a denial of service attack against Raspberrypi or an elevation of privilege attack against Raspberrypi or an information disclosure by Raspberrypi. 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: validation of input

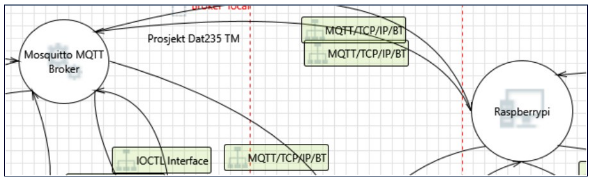
82. Spoofing the Raspberrypi Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Raspberrypi may be spoofed by an attacker and this may lead to information disclosure by Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the destination process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

83. Spoofing the Mosquitto MQTT Broker Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Mosquitto MQTT Broker may be spoofed by an attacker and this may lead to unauthorized access to Raspberrypi. Consider using a standard authentication mechanism to identify the source process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: MQTT/TCP/IP/BT



84. Spoofing the Paho Python MQTT Client Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Paho Python MQTT Client may be spoofed by an attacker and this may lead to unauthorized access to Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the source process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

85. Spoofing the Mosquitto MQTT Broker Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Mosquitto MQTT Broker may be spoofed by an attacker and this may lead to information disclosure by Paho Python MQTT Client. Consider using a standard authentication mechanism to identify the destination process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

86. Potential Lack of Input Validation for Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description:	Data flowing across MQTT/TCP/IP may be tampered with by an attacker. This may lead to a denial of service attack against Mosquitto MQTT Broker or an elevation of privilege attack against Mosquitto MQTT Broker or an information disclosure by Mosquitto MQTT Broker. 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:	Input validation

87. Potential Data Repudiation by Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Repudiation
Description:	Mosquitto MQTT Broker 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:	Logging

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

Category:	Information Disclosure
Description:	Data flowing across MQTT/TCP/IP 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:	Encryption

89. Potential Process Crash or Stop for Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Mosquitto MQTT Broker crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification:	restart? send a team to check it out.

90. Data Flow Generic Data Flow 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:	Redundancy

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

Category:	Elevation Of Privilege
Description:	Mosquitto MQTT Broker may be able to impersonate the context of Paho Python MQTT Client in order to gain additional privilege.
Justification:	Authentication

92. Mosquitto MQTT Broker May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	Paho Python MQTT Client may be able to remotely execute code for Mosquitto MQTT Broker.
Justification:	Disallow remote execution of abusable code

93. Elevation by Changing the Execution Flow in Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	An attacker may pass data into Mosquitto MQTT Broker in order to change the flow of program execution within Mosquitto MQTT Broker to the attacker's choosing.
Justification:	Authentication should easily solve this problem

94. 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:	Integrate additional piece of secret payload in authentication, alternatively user training

95. Elevation by Changing the Execution Flow in Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	An attacker may pass data into Mosquitto MQTT Broker in order to change the flow of program execution within Mosquitto MQTT Broker to the attacker's choosing.
Justification:	Authentication should easily solve this problem

96. Mosquitto MQTT Broker May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	Raspberrypi may be able to remotely execute code for Mosquitto MQTT Broker.
Justification:	Raspberrypi should never be able to remotely execute code in any cases at all.

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

Category:	Elevation Of Privilege
Description:	Mosquitto MQTT Broker may be able to impersonate the context of Raspberrypi in order to gain additional privilege.
Justification:	Authentication and filtering.

98. Data Flow MQTT/TCP/IP 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:	Redundancy make sure that only one device is allowed to connect a certain times in a certain timeline

99. Potential Process Crash or Stop for Mosquitto MQTT Broker [State: Not Applicable] [Priority: High]

Category:	Denial Of Service
Description:	Mosquitto MQTT Broker crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification:	restart? send a team to check it out.

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

Category:	Information Disclosure
Description:	Data flowing across MQTT/TCP/IP/BT 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:	The data will be encrypted before sending it, therefor this will not be a problem

101. Potential Data Repudiation by Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Repudiation
Description:	Mosquitto MQTT Broker 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:	With a proper logging system which records both time and data will atleast pinpoint any doubts.

102. Potential Lack of Input Validation for Mosquitto MQTT Broker [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	Data flowing across MQTT/TCP/IP/BT may be tampered with by an attacker. This may lead to a denial of service attack against Mosquitto MQTT Broker or an elevation of privilege attack against Mosquitto MQTT Broker or an information disclosure by Mosquitto MQTT Broker. 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:	validation of input

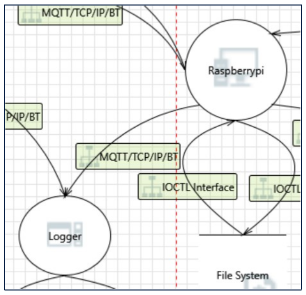
103. Spoofing the Mosquitto MQTT Broker Process [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	Mosquitto MQTT Broker may be spoofed by an attacker and this may lead to information disclosure by Raspberrypi. Consider using a standard authentication mechanism to identify the destination process.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

104. Spoofing the Raspberrypi Process [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	Raspberrypi may be spoofed by an attacker and this may lead to unauthorized access to Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the source process.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

Interaction: MQTT/TCP/IP/BT



105. Spoofing the Raspberrypi Process [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	Raspberrypi may be spoofed by an attacker and this may lead to unauthorized access to Logger. Consider using a standard authentication mechanism to identify the source process.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

106. Spoofing the Logger Process [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	Logger may be spoofed by an attacker and this may lead to information disclosure by Raspberrypi. Consider using a standard authentication mechanism to identify the destination process.
Justification:	This will not be a problem as long as there is an authentication mechanism with a strong login in place

107. Potential Lack of Input Validation for Logger [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	Data flowing across MQTT/TCP/IP/BT may be tampered with by an attacker. This may lead to a denial of service attack against Logger or an elevation of privilege attack against Logger or an information disclosure by Logger. 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:	Input validation

108. Raspberrypi Process Memory Tampered [State: Needs Investigation] [Priority: High]

Category:	Tampering
Description:	If Raspberrypi is given access to memory, such as shared memory or pointers, or is given the ability to control what Logger executes (for example, passing back a function pointer.), then Raspberrypi can tamper with Logger. 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:	not sure, needs further investigation

109. Potential Data Repudiation by Logger [State: Mitigation Implemented] [Priority: High]

Category:	Repudiation
Description:	Logger 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:	extensive logging

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

Category:	Information Disclosure
Description:	Data flowing across MQTT/TCP/IP/BT 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: The data will be encrypted before sending it, therefor this will not be a problem

111. Potential Process Crash or Stop for Logger [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: Logger crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: restart? send a team to check it out.

112. Data Flow Generic Data Flow 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: Redundancy make sure that only one device is allowed to connect a certain times in a certain timeline

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

Category: Elevation Of Privilege
Description: Logger may be able to impersonate the context of Raspberrypi in order to gain additional privilege.
Justification: Authentication and filtering.

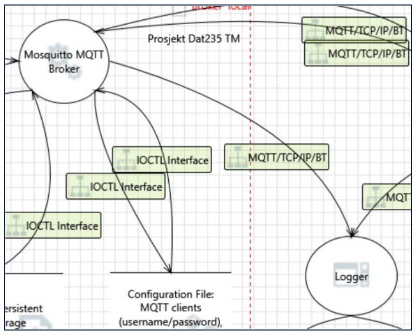
114. Logger May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Raspberrypi may be able to remotely execute code for Logger.
Justification: Authentication and filtering.

115. Elevation by Changing the Execution Flow in Logger [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Logger in order to change the flow of program execution within Logger to the attacker's choosing.
Justification: Authentication should easily solve this problem

Interaction: MQTT/TCP/IP/BT



116. Spoofing the Mosquitto MQTT Broker Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Mosquitto MQTT Broker may be spoofed by an attacker and this may lead to unauthorized access to Logger. Consider using a standard authentication mechanism to identify the source process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

117. Spoofing the Logger Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: Logger may be spoofed by an attacker and this may lead to information disclosure by Mosquitto MQTT Broker. Consider using a standard authentication mechanism to identify the destination process.
Justification: This will not be a problem as long as there is an authentication mechanism with a strong login in place

118. Potential Lack of Input Validation for Logger [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: Data flowing across MQTT/TCP/IP/BT may be tampered with by an attacker. This may lead to a denial of service attack against Logger or an elevation of privilege attack against Logger or an information disclosure by Logger. 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: Input validation

119. Mosquitto MQTT Broker Process Memory Tampered [State: Needs Investigation] [Priority: High]

Category: Tampering
Description: If Mosquitto MQTT Broker is given access to memory, such as shared memory or pointers, or is given the ability to control what Logger executes (for example, passing back a function pointer.), then Mosquitto MQTT Broker can tamper with Logger. 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: not sure, needs further investigation

120. Potential Data Repudiation by Logger [State: Mitigation Implemented] [Priority: High]

Category: Repudiation
Description: Logger 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: extensive logging

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

Category: Information Disclosure

Description:	Data flowing across MQTT/TCP/IP/BT 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:	The data will be encrypted before sending it, therefor this will not be a problem

122. Potential Process Crash or Stop for Logger [State: Mitigation Implemented] [Priority: High]

Category:	Denial Of Service
Description:	Logger crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification:	restart? send a team to check it out.

123. Data Flow MQTT/TCP/IP/BT 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:	Redundancy make sure that only one device is allowed to connect a certain times in a certain timeline

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

Category:	Elevation Of Privilege
Description:	Logger may be able to impersonate the context of Mosquitto MQTT Broker in order to gain additional privilege.
Justification:	Authentication and filtering.

125. Logger May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	Mosquitto MQTT Broker may be able to remotely execute code for Logger.
Justification:	Authentication and filtering.

126. Elevation by Changing the Execution Flow in Logger [State: Mitigation Implemented] [Priority: High]

Category:	Elevation Of Privilege
Description:	An attacker may pass data into Logger in order to change the flow of program execution within Logger to the attacker's choosing.
Justification:	Authentication should easily solve this problem