Security Review of Infant Incubator

Contributors: Isaiah Genis & Jamie Leach

System Overview

Security Report Motivations

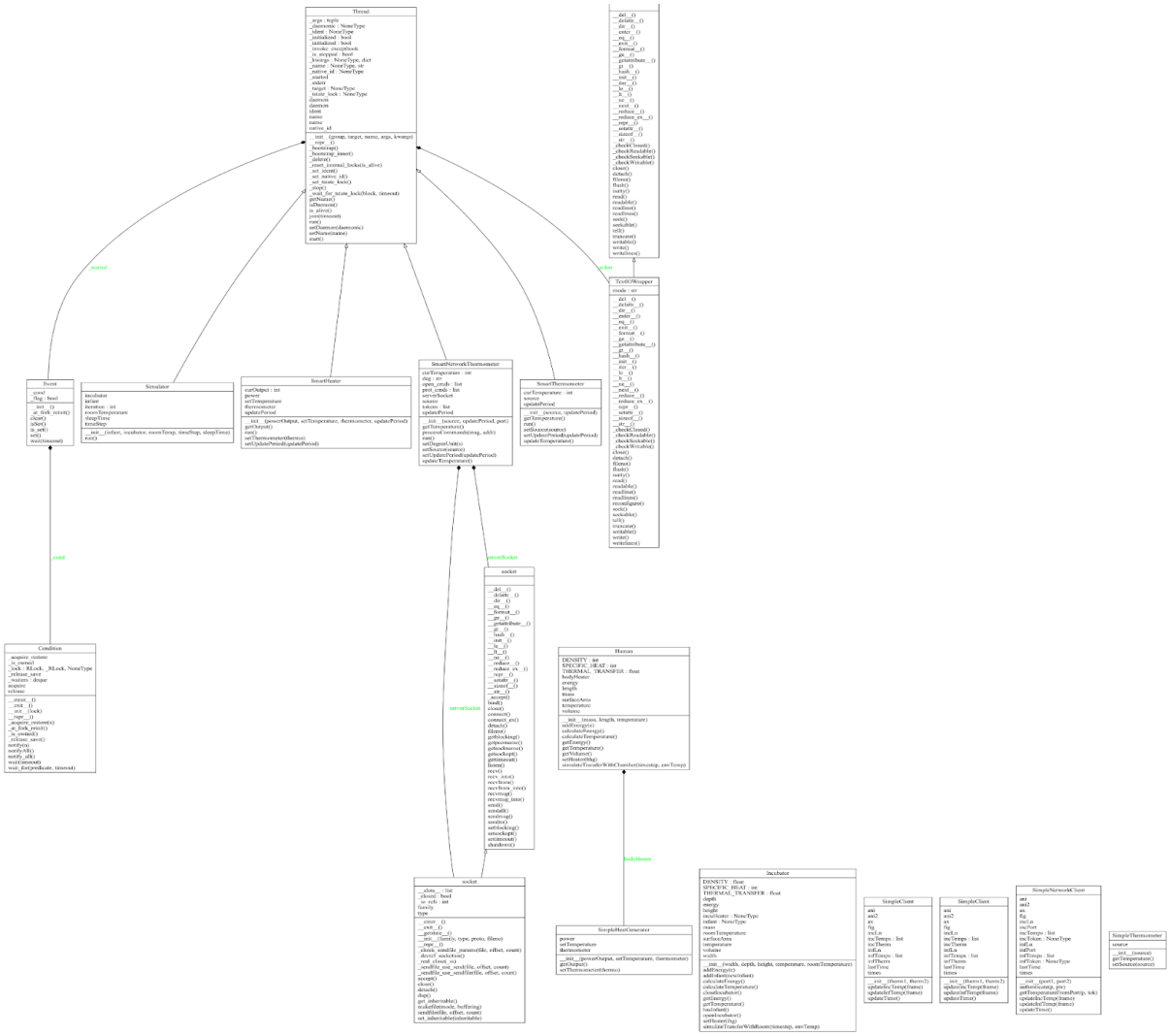
As security review analysts we understand that the Infant Incubator is: “designed to provide a safe, controlled space for infants to live while their vital organs develop and is intended to set the ideal temperature as well as the perfect amount of oxygen, humidity, and light needed to promote this growth” and that our work directly contributes to the safety of human life both infants and operators/users of the device. Additionally the developers of this project have tasked our team to ensure that we satisfy US Government's security regulations for medical devices and we are motivated to ensure these regulations are met. Currently many medical device security regulations are published by the FDA for overall compliance recommendations and ISO for international device standards and we have reviewed what materials are publicly available in order to inform the commendations, assessments and conclusions we provide.

System Summary

The Infant Incubator is an enclosed box filled with air wherein the bottom is an insulator pad that an infant would rest upon, and all other sides are plexiglass acrylic that borders the outside room. The system is multi-threaded and accounts for the latent energy present in the air, insulator, and infant, and calculates a heat transfer model between them. The heat transfer model must not only account for some constants we assume to be true, but also the specific heat of the air, the animal tissue, the human body, the plexiglass material, and the air density. It must also account for how heat transfers between them at specific rates. There are also specific use cases for how the chamber is operated; for example, it is likely empty to start, warmed up empty, then opened for a duration to place the infant inside as air is dispersed and later opened again to depressurize the system and remove the infant. All of these heat transfer and energy calculations are to be written in a multi-threaded way, which indicates various real-time constraints, race-conditions, and faults which must be accounted for as the system reads temperatures from a thermometer, calculates heat, and relays instructions to the insulator and the displays. Lastly, while the device can operate as a stand-alone system, it must also be able to be controlled and monitored over the network. This will then present various time delays, network setup, inherent network vulnerabilities, etc., and the system must be designed to handle errors and attacks. In case of these or in an emergency, the system must default to actions that in all regards prioritize the safety of the infant over the security of the system.

System Components

Class Diagram



Libraries in Use

Python Standard Library:

The Python standard library provides the core capabilities for python and all the built-in modules that provide access to system functionality like I/O.

Modules:

- Math Module This module provides access to the mathematical functions defined by the C standard.

        - infinc.py- Code Comments suggest this is imported to calculate the square root which takes a single parameter of the number you want the square root of.
        - SampleNetworkClient.py - used to calculate the floor of a number which is the largest integer less than or equal to the input.
    - Time Module This module provides various time-related functions.
        - Infinc.py - Code comments suggest this is imported for sleep function call which has a single parameter for the number of seconds to pause the execution of the thread.
        - SampleNetworkClient.py -
            - used to calculate the current time in seconds and time.time() is invoked with no parameters.
            - Time.localtime calculates the time object in the local timezone given the parameter of the time in seconds to convert.
            - Time.strftime is used to convert time object into a string. The parameters are the formatting string for how to represent the time and the time object to convert to a string.
    - Threading Module - This module provides high-level interfaces for threads and performing threaded operations.
        - Infinc.py - Comments and code indicate it is being used to create new Thread class objects. The Thread class represents an activity that is run in a separate thread of control. Thread is being passed 2 parameters in this case. The first is the target which is what will be executed on the thread and the second is if it is a daemon thread which means it will run in the background.
    - Socket Module This module provides access to the BSD socket interface
        - SampleNetworkClient.py
            - socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM). Family is the address and protocol family to use for the socket. Type represents the type of socket. In this case a datagram socket
            - Socket.sendto is used to send data over the socket and the parameters are the message to send and the destination address.
            - socket.recvfrom(1024) is used to receive data from the socket and the parameter is the number of bytes to receive.
    - String Module - common string operations and constants
        - SampleNetworkServer.py:
            - string.ascii_uppercase - constant with string representation of uppercase letters
            - string.ascii_lowercase - constant with string representation of lowercase letters
            - string.digits - constant with string representation of digits
    - OS Module - this is a platform independent way of using operating system functionality.

        - SampleNetworkServer.py
            - os.O_NONBLOCK - non-blocking flag for file descriptor. A blocking mode means that I/O system calls like read, or connect can be blocked by the system.
    - Fcntl module This Module provinces file and I/O control on file descriptors.
        - SampleNetworkServer.py
            - fcntl.fcntl(fd, cmd, arg=0) - performs cmd on file descriptor fd
            - fcntl.F_SETFL - operation to set file descriptor status flags. In our case is used to set non-blocking mode using os.O_NONBLOCK flag
    - Errno module - This module provides standard error codes.
        - SampleNetworkServer.py
            - errno.EWOULDBLOCK - error code, which appears when recv() call doesn't find any data, or if a send() call can't immediately dispose of the data
    - Random Module - This Module is used for pseudo-random number generation
        - SampleNetworkServer.py
            - random.choice - selects random element from the sequence

Matplotlib

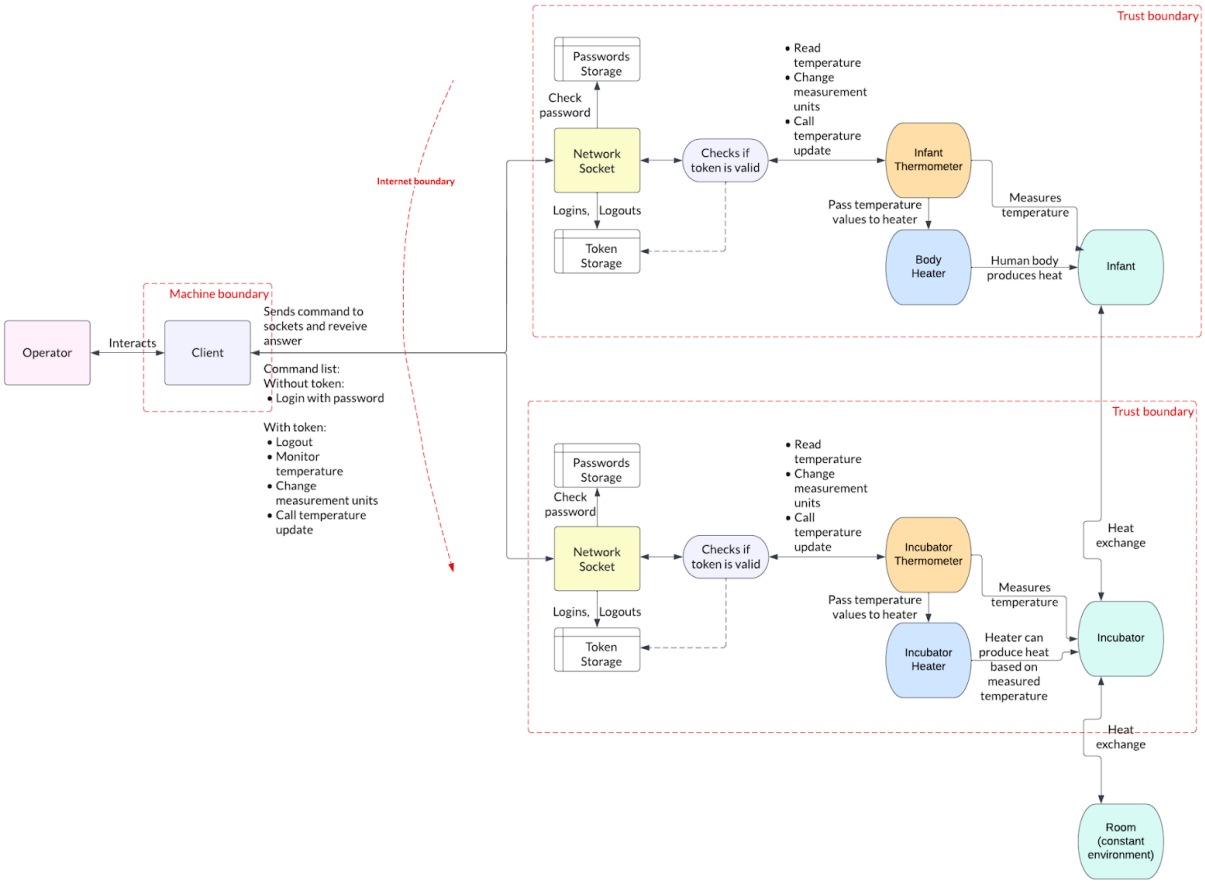
- Modules & Usage
    - Pyplot - This module is used to generate interactive plots and simple cases of programmatic plot generation
        - SampleNetowrkClient.py
            - plt.subplots() Create a figure and a set of subplots.
            - plt.plot - plots data. The parameters are the x values, the y values and the label for the line.
            - plt.title sets the plot title
            - plt.xticks - sets tick location and labels on the x-axis
            - plt.ylim - set limit to display on y-axis
            - plt.legend - place legend on the plot
            - plt.grid - control the grid on the plot
            - plt.show - start showing plot
    - Animation - used to create live animations in matplotlib
        - SampleNetworkClient.py, SampleNetworkServer.py, SampleClient.py
            - animation.FuncAnimation - this repeatedly updates an animation by calling the function provided over and over again. The parameters used are the figure monitor, the function to call, and the interval to delay between frames.

Security Requirement Recommendations

Requirement #	Requirement Component applies to: (Concept/Hardware)	Requirement Description	Is this requirement met in the final product?
1.	Access Control -User, Incubator, SimpleNetworkClient	Limit access to authorized authenticated users, processes, and devices ● Only approved Medical and Technical staff should be able to access the device ● Only approved temperature sensors should be able to interact with the incubator processor. ● Only Local Networks should be allowed to connect to the incubators without administrator approval	No
2.	Access Control - SimpleNetworkClient,SmartNetworkThermometer	To avoid confidentiality concerns, authentication measures such as usernames and/or passwords should be unique per device or user account.	No
3.	Access Control - Network Socket, SimpleNetworkClient,SmartNetworkThermometer	Manufacturer provided credentials must be rotated on first activation.	No
4.	System Usage - Medical Professional/User	To avoid unsafe operation, the incubator should not support adding infants while the incubator is occupied or closed.	No
5.	Access Control - Network Socket	All Incubator sessions should have a time limit established before requiring re-authentication. We recommend 60 minutes with up to 12 hours being the limit.	No
6.	Access Control - Network Socket	A Logout should result in the termination of any heater activity if no other tokens are valid.	No
7.	Input Validation - Client	All inputs should be validated for the appropriate encoding and reject non-compliant messages. In this case, it should be UTF-8.	No
8.	Input Validation - Client	Failed Authentication or validation should immediately stop the request's processing. The system should be deny-by-default.	No
9.	Auditability - Incubator	The incubator shall implement logging and define a set of required fields and events.	No
10.	Auditability - Incubator	Logging that must include the timestamp in UTC	No
11.	Auditability - Incubator	Logged events must include but not limited to: ● Authentication requests ● Logout requests ● temperature setting changes ● processing failures or other errors, ● adding or removing an infant. ● Network Connection Start/Stop/Loss	No
12.	System Configurations - Thermometer	To avoid harm to the infant the temperate range that can be set on the heater should be set to those considered medically safe and supported by the hardware.	No
13.	System Configurations - Incubator Heater	The power output of the Incubator should be limited based on the manufacturer's specifications to avoid damage to people or the equipment and medical requirements.	No
14.	System Configurations - Incubator/Thermometer/Heater	To avoid adverse effects on infant health and safety, the incubator should not run for longer than there is sufficient air volume for the infant to breathe inside the incubator.	No
15.	System Configurations - Incubator	In order to detect malfunctions and accurately be able to perform its function Incubator Update Interval shall not fall below once every 15 minutes.	No
16.	System Configurations - Incubator/Thermometer/Heater	To avoid data integrity issues all data should be stored and in a standardized unit as set by company policy. For the purposes of this writeup we shall say Kelvin. Conversion should occur at display/return to the user only.	No
17.	Media Protection - SmartNetworkClient, Incubator	To protect data confidentiality, all incubators should support encryption at rest using FIPS compliant Algorithms such as AES-256 or stronger.	No

18.	System Configurations -Vulnerability Management	The System must provide a mechanism to provide security updates to the system and device.	No
19.	Systems Communications - Network Socket, Incubator at whole	All Communications with the incubator should be over encrypted channels with forward secrecy. (TLS 1.3+)	No
20.	System Configurations	All Sensitive attributes should be stored and accessed securely and be able to be configured dynamically.	No
21.	Access Control - Network Socket	Authentication should be handled before processing any other commands.	No
22.	Systems Development - Incubator	All releases should be tested for functionality and security prior to publishing.	No
23.	Documentation - Incubator & Client	All releases should come with accompanying documentation highlighting the changes between versions as well as the documentation on how to securely use and configure their systems.	No
24.	Privacy	The Incubator and related services should not store or log PII or PHI. Unique identifiers should be used to identify users or infants.	
25.	System Communications	All requests received should return a status of their outcome back to their user without disclosing sensitive system errors or details.	No
26.	System Architecture	The system should be designed with maximum error/exception handling with logging to maintain system availability.Individual request processing and input errors should not result in a request failure not a process failure.	No
27.	System Communications and Architecture	The system should document expected usage rates and patterns and set thresholds for use based on these expectations. This includes items like request rate limiting, number of active tokens at a time, number of thermometers per incubator, number of users per incubator, etc.	No

Data Flow Diagram/Threat Model



Asset List

Asset #	Asset Name	Asset Description	Concrete or Abstract Asset?	Critical Asset or no?
1.	Infant	The child placed in an incubator	Both	ISC2 says human life is our number 1 priority.
2.	Incubator	The device being produced and the code associated with representing it.	Both	Critical
3.	SmartNetworkThermometer	The Smart Thermometer device and associated python code to represent it.	Both	Critical to functionality
4.	Token List	List of authenticated user tokens used to interact with the system	Abstract	Critical to functionality

5.	Auth Password	The shared password used to access the systems.	Abstract	Critical
6.	NetworkSocket	The Device/Class that allows network communications		Critical to functionality
7.	Room	The room containing the incubator	Concrete	Critical
8.	System Documentation	The internal and external documentation for the incubator systems	Concrete	Critical

Threat List

Threat #	Affected Component(s)	Threat Summary	Threat Category	Recommended Control
1.	SmartNetworkThermometer, Socket, SimpleNetworkClient	Sockets Flood Causing Crash or at very least legitimate requests are not processed timely if at all.	DoS	Implement Rate Limiting
2.	SmartNetworkThermometer	Session Flooding Causes Memory Resource Issues	DoS	Implement a limit on valid tokens per password. See Vuln 8.
3.	Incubator	Significant Temperature Manipulation Via changing units	Tampering	
4.	Infant, Incubator Heater	Harm Infant(OverHeat/Burn)	Tampering	See Requirements 12,13
5.	Infant, Incubator	Harm Infant(Suffocation)	DoS	See Requirement 14
6.	Infant, User, Incubator	Data Leak	Information Disclosure	See Requirement 17 & 19 for encryption and 23 for Privacy
7.	SampleNetworkServer,Smart NetworkThermometer, Incubator	Unauthenticated User Interacts with Incubator Privileged functions	Escalation of Privilege	See Requirement 2, 3 & 5 for access control recommendations
8.	Incubator	Theft of Device	Tampering	See Requirement 1 for Physical access control
9.	Incubator	Damage to Device	Tampering	Develop system Health Check and notification functionality if there are any failures. Also see Requirements 1 and 23.
10.	SmartHeater	Heater Fails/Breaks	Dos/Availability	Develop system Health Check and notification functionality if there are any failures.
11.	Socket,SimpleNetworkClient, SmartNetworkThermometer	Packet Sniffing - unencrypted communications leaks credentials and data.	Information Disclosure	Implement Requirement 19

12.	Incubator, SmartThermometer ,SmartNetworkThermometer, SmartHeater	Unrecoverable System Failure	Denial of Service	Implement Error Handling, Logging, and Diagnostic Capabilities
13.	Human	Misdiagnosis	Tampering	See requirements 1 and 23 and discuss with legal as part of terms of purchase/license agreement.
14.	Human	Mistreatment	Tampering	See requirements 1 and 23 and discuss with legal as part of terms of purchase/license agreement.

Risk Calculations

Risk #	Risk Name	Related Threat (if applicable)	Likelihood	Impact	Risk Level
1.	Risk of Inaccessibility, and potential injury	DoS SmartNetworkThermometer	3 - Medium	5 - System outage	Medium
2	Risk of potential injury	Tampering - Temperature Manipulation via unit change	4 - High	8 - Mistreatment/Diagnosis	High
3	Risk of potential infant injury	Tampering - Harm Infant via overheating	4 - High	9 - Potential Injury	Critical
4.	Risk of potential infant death	Denial of Service - Suffocation of Infant by insufficient oxygen into the incubator	3 - Medium	10 - Potential Death	High
5.	Privacy risk	Information Disclosure/Packet Sniffing - unencrypted communications leaks credentials and data.	3 - Medium	6 - Privacy Breach	Medium
6.	Risk of repudiation	Tampering - Ability to alter messages between client and the incubator with no way for either party to know. Missing Message Authentication Codes	4 - High	5 - Network Compromise	High
7.	Risk of unauthorized access	Escalation of Privs - Ability for Unauthenticated users to chain commands with invalid authentication requests allows them to execute privileged commands	4 - High	5 - Network Compromise	High
8.	Risk of system crashing	Input validation/Error handling	4 - High	5 - Network Compromise	High

9.	Risk of account compromise	Account Spoofing/Session Hijacking	4 - High	7 - User Compromised	High
10.	Out of date software risk	DoS - Inability to provide patches. There is no service to provide updates security or otherwise.	3 - Medium	4 - Equipment Damage	Medium
11.	Risk of network loss and inaccessibility	Dos -session flooding	3 - Medium	5 - Network Compromise	Medium
12.	Risk of man in the middle attacks	DNS cache poisoning	2 - Low	5 - Network Compromise	Medium
13.	Risk of social engineering	Phishing	2 - Low	2 - Reputational Damage	Low
14.	Risk of command injections	Injection attack	4 - High	5 - Network Compromise	High
15.	Loss of device risk	Theft of Device	3 - Medium	1 - Financial Loss	Medium
16.	Device damage risk	Tampering - Damage to device	4 - High	4 - Equipment Damage	High
17.	Poor User experience Risk	Application crashes due to exception handling errors	3- Medium	2 - Reputational Damage	Medium

Risk Analysis

RISK Justification/Value

For the infant incubator we will be using a manual assessment method to calculate a risk rating for the various threats identified within the threat model as seen in table 4. This risk value (RV) is computed by determining the attack potential value (APV), the attack success value (ASV)

and the impact value (IV), which correspond to common device attack surface characteristics, and adding them together:
RV = APV + ASV + IV
Once the RV is calculated, the risk is then classified as a: “Critical”, ‘‘High”, ‘‘Medium’’, and ‘‘Low’’ risk based on the sum, the higher the value the higher the risk.

Attack Potential Value

Attack potential value (APV) is comprised of factors that are needed in order to construct an attack or exploit against a system such as:

- time needed to identify or exploit a vulnerability
- experience required to run an exploit
- knowledge of the system needed to craft an exploit
- Access level needed in the system to execute an attack
- Current hardware/software defenses against scanning/vulnerabilities

In our work these factors are then rated on a scale from 0-5 and the overall attack potential value (AVP) is then defined as the sum of these factors as seen in table 1.

Attack Success Value

The attack success value (ASV) is a generalization of how likely an attack would succeed considering the evaluation performed in the AP value. ASV is obtained by taking the sum of the factors for AP and then classified into one of five levels: “Critical”, ‘‘High”, ‘‘Medium’’, ‘‘Low’’, and “Unlikely” as seen in table 2.

Impact Value

The impact value is simplification of the types of consequences incurred should an attack succeed. As safety is our primary concern any compromises to patient death/injury or misdiagnoses are rated as the highest possible impact. These are not mutually exclusive risks, and it is often common for an attack to cause one or many impacts to occur, for the sake of our

risk rating the highest impact will be taken over adding impact values together.

Attack Potential Value

Factor	Level	Value
Time	< 1 Day	5
	< 1 Week	4
	<1 Month	3
	<1 Year	2
	> 1 Year	1
	Undefined	0
Experience	Master	0
	Expert	1
	Adept	2
	Entry level	3
	Common	4
	None	5
System Knowledge	Classified	5
	Intimate	4
	Familiar	3
	Uncommon	2
	Public	1
	None	0
Accessibility	No restriction	5
	High	4
	Medium	3
	Low	2
	No access	1
	Off network	0
Defenses	No controls	5
	Low	4
	Medium	3
	High	2
	Critical	1

Table 1

Attack Success Value

AP Value	Attack Likelihood	ASV Rating
21 to 25	Critical	5
16 to 20	High	4
11 to 15	Medium	3
6 to 10	Low	2
1 to 6	Unlikely	1

Table 2

Impact Value

Risk	Value
Potential Death	10
Potential Injury	9
Mistreatment/diagnosis	8
User compromised	7
Privacy Breach	6
Network compromise/outages	5
Equipment Damage	4
Legal Impact	3
Reputational Damage	2
Financial loss	1
No Impact	0

Table 3

Risk Value

Rating	Total Sum
Critical	32 to 40
High	22 to 31
Medium	12 to 21
Low	2 to 11

Table 4

	APV					ASV	IV	RV	
Risk/Attack	**Time**	**Experience**	**System Knowledge**	**Accessibility**	**Defenses**	**Attack Likelihood**	**Impact/Risk**	**Sum**	**Rating**
DoS- Smart Network Thermometer	4	1	2	1	5	3	5	21	Medium

Tampering - Temperature Manipulation via unit change	5	2	4	3	4	4	8	30	High
Tampering - Harm Infant via overheating	4	4	4	3	4	4	9	32	Critical
DOS - Suffocation of Infant by insufficient oxygen into the incubator	4	1	2	1	5	3	10	26	High
Information Disclosure/Packet Sniffing - unencrypted communications leaks credentials and data.	3	1	3	2	3	3	6	21	Medium
Tampering - Ability to alter messages between client and the incubator with no way for either party to know. Missing Message Authentication Codes	5	2	4	3	3	4	5	26	High
Escalation of Privs - Ability for Unauthenticated users to chain commands with invalid authentication requests allows them to execute privileged commands	4	1	4	3	4	4	5	25	High
Input validation/Error handling	5	5	0	2	5	4	5	26	High
Account Spoofing/Session Hijacking	4	1	4	3	4	4	7	27	High
DoS - Inability to provide patches. There is no service	4	1	2	1	5	3	4	20	Medium

to provide updates security or otherwise.									
Dos -session flooding	4	1	2	1	5	3	5	21	Medium
DNS cache poisoning	4	0	1	1	4	2	5	17	Medium
Phishing	3	0	1	0	3	2	2	11	Low
Injection attack	5	1	4	4	5	4	5	28	High
Theft of Device	0	4	1	5	5	3	1	19	Medium
Tampering - Damage to device	5	2	4	3	3	4	4	25	High
Application crashes due to exception handling errors	0	5	3	3	4	3	2	20	Medium

Software Analysis (from Labs)

We performed manual static and dynamic analysis without the use of additional tools beyond python.

Vuln 1: Improper Password validation and secure-by-default handling.

Affected Component(s)

- SampleNetworkServer.py:SmartNetworkThermometer:processCommands

Description

A vulnerability was discovered in the ‘SampleNetworkServer.py’ file on lines 59-72 where the ‘processCommands’ function could allow a user/attacker to execute any command even if an invalid password was given to the AUTH command. This can compromise the incubator as it allows the system to be accessed by anyone and directly impacts any expectation of confidentiality.

Affected Requirement(s)

- Our Requirements 1,2,3,8
- NIST 800-171r2 Control: 3.1.1: Limit system access to authorized users, processes acting on behalf of authorized users, and devices (including other systems)
- NIST 800-171r2 Control: 3.1.20 (Verify and control/limit connections to and use of external systems)
- NIST 800-171r2 Control: 3.1.15 (Authorize remote execution of privileged commands and remote access to security-relevant information)
- CWE-390

Exploit of Bad Password Error Handling

```
# Exploit
import unittest
import socket

def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        return False
def run_specified_tests(tests:list):
    # Run only the tests in the specified classes
    test_classes_to_run = tests
    loader = unittest.TestLoader()
    suites_list = []
    for test_class in test_classes_to_run:
        suite = loader.loadTestsFromTestCase(test_class)
        suites_list.append(suite)
    big_suite = unittest.TestSuite(suites_list)
    runner = unittest.TextTestRunner()
    results = runner.run(big_suite)
    return results

class TestPassword(unittest.TestCase):
    # Vuln 1 Exploit Test
    def test_bad_password(self):
        s_rl = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
        input_request=b'AUTH basd;GET_TEMP'
        s_rl.sendto(input_request, ("127.0.0.1", 23456))
```

```
        output_responce, real_server_addr = s_rl.recvfrom(1024)
        output_responce=str(output_responce.strip(),"utf8")
        s_rl.close()
        # print(output_responce)
        self.assertFalse(is_number(output_responce),msg="Received Valid Temp when should
have been rejected for invalid auth.")
if __name__ == '__main__':
    run_specified_tests([TestPassword])
```

Recommended Fix

The Patch for this vulnerability is to add proper error handling of invalid auth tokens to the processCommands function of SampleNetworkServer.py. See lines 13-15 of the below method

```
# Patched Function Snippet
import string
import random

def processCommands(self, msg, addr) :
    cmds = msg.split(';')
    for c in cmds :
        cs = c.split(' ')
        if len(cs) == 2 : #should be either AUTH or LOGOUT
            if cs[0] == "AUTH":
                if cs[1] == "!Q#E%T&U8i6y4r2w" :
                    self.tokens.append(''.join(random.choice(string.ascii_uppercase + string.ascii_lowercase +
string.digits) for _ in range(16)))
                    self.serverSocket.sendto(self.tokens[-1].encode("utf-8"), addr)
                else:
                    self.serverSocket.sendto(b"Bad Token\n", addr)
                    return
                    #print (self.tokens[-1])
            elif cs[0] == "LOGOUT":
                if cs[1] in self.tokens :
                    self.tokens.remove(cs[1])
            else : #unknown command
                self.serverSocket.sendto(b"Invalid Command\n", addr)
        elif c == "SET_DEGF" :
            self.deg = "F"
        elif c == "SET_DEGC" :
            self.deg = "C"
        elif c == "SET_DEGK" :
            self.deg = "K"
        elif c == "GET_TEMP" :
            self.serverSocket.sendto(b"%f\n" % self.getTemperature(), addr)
```

```
elif c == "UPDATE_TEMP" :
    self.updateTemperature()
elif c :
    self.serverSocket.sendto(b"Invalid Command\n", addr)
```

Vuln 2: Ability to chain commands on Logout after a token is revoked.

Affected Component(s):

- SampleNetworkServer.py:SmartNetworkThermometer:processCommands

Description

A vulnerability was discovered in the 'SampleNetworkServer.py' file on lines 59-72 where the processCommands function does not suspend access to configurations and resources being utilized. The user maintains access to sensitive functions and information.
Calling the Logout command should immediately suspend my access to configurations and resources. Failing to exit allows the user to maintain access to sensitive functions and information.
This occurs on lines 59-72 in the processCommands function.

Affected Requirement(s)

- Our Requirements 1,6,8
- NIST 800-171r2 Control: 3.1.1: (Limit system access to authorized users, processes acting on behalf of authorized users, and devices (including other systems))
- NIST 800-171r2 Control: 3.1.20 (Verify and control/limit connections to and use of external systems)
- NIST 800-171r2 Control: 3.1.15 (Authorize remote execution of privileged commands and remote access to security-relevant information)
- NIST 80-66r2 5.1.4
- CWE-390

Exploit of Logout chaining

```
#Exploit
import unittest
import socket


class TestLogout(unittest.TestCase):
    # Logout Test
```

```
    def test_bad_logout(self):
        s_rl = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
        request_1 = b'AUTH !Q#E%T&U8i6y4r2w;'
        s_rl.sendto(request_1, ("127.0.0.1", 23456))
        token, real_server_addr = s_rl.recvfrom(1024)
        token = token.strip()
        # print(token[-1])
        request_2 = bytes(f"LOGOUT {token};GET_TEMP","utf8")
        s_rl.sendto(request_2, ("127.0.0.1", 23456))
        output_responce, real_server_addr = s_rl.recvfrom(1024)
        output_responce = str(output_responce.strip(),"utf8")
        s_rl.close()
        # print(output_responce)
        self.assertFalse(is_number(output_responce),msg="Received Valid Temp when should
have been rejected for invalid token.")
# res = unittest.main(argv=[''], verbosity=3, exit=False)
if __name__ == '__main__':
    run_specified_tests([TestLogout])
```

Recommended Fix

The Patch for this vulnerability is to check that before executing any of the commands that there is a successful AUTH completed first.

Example Code Fix

```
# Patched Function section.
def processCommands(self, msg, addr):
    cmds = msg.split(';')
    for c in cmds:
        cs = c.split(' ')
        if len(cs) == 2:  # should be either AUTH or LOGOUT
            if cs[0] == "AUTH":
                if cs[1] == "!Q#E%T&U8i6y4r2w":
                    self.tokens.append(''.join(
                        random.choice(string.ascii_uppercase + string.ascii_lowercase + string.digits) for
_ in
                        range(16)))
                    self.serverSocket.sendto(self.tokens[-1].encode("utf-8"), addr)
                else:
                    self.serverSocket.sendto(b"Bad Token\n", addr)
                    return
```

```
            # print (self.tokens[-1])
        elif cs[0] == "LOGOUT":
            if cs[1] in self.tokens:
                self.tokens.remove(cs[1])
                self.serverSocket.sendto(b"Logged Out\n", addr)
            else:
                self.serverSocket.sendto(b"Bad Token\n", addr)
            return
        else:  # unknown command
            self.serverSocket.sendto(b"Invalid Command\n", addr)
    elif c == "SET_DEGF":
        self.deg = "F"
    elif c == "SET_DEGC":
        self.deg = "C"
    elif c == "SET_DEGK":
        self.deg = "K"
    elif c == "GET_TEMP":
        self.serverSocket.sendto(b"%f\n" % self.getTemperature(), addr)
    elif c == "UPDATE_TEMP":
        self.updateTemperature()
    elif c:
        self.serverSocket.sendto(b"Invalid Command\n", addr)
```

Vuln 3: Inaccurate Incubator Temperature Reporting on unit change

Affected Component(s)

- SampleNetworkServer.py:SimpleClient:updateInfTemp
- SampleNetworkServer.py:SimpleClient:updateIncTemp

Description

A vulnerability was discovered in the ‘SampleNetworkServer.py’ file on lines 157 & 165 where the updateInfTemp and updateIncTemp variables use the same calculation for temperature regardless of what unit of measurement is chosen and causes the graph to not display properly as the values fall out of the visible range. This can compromise the incubator as it causes inconsistencies in the reporting of temperature values which leads to unintended temperatures in the incubator and endangers the safety of anyone placed into the machine while also making it hard for a medical professional to monitor.

Affected Requirements

- Our Requirements 16

Exploit

In [ ]:

```
#Exploit
import unittest
import socket

class TestUnitChange(unittest.TestCase):
    # Unit change Test
    def test_unit_change(self):
        s_rl = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
        request_1=b'AUTH !asd;GET_TEMP'
        s_rl.sendto(request_1, ("127.0.0.1", 23457))
        output_responce, real_server_addr = s_rl.recvfrom(1024)
        deg_k=float(output_responce.strip())
        # print(f"deg kelvin={deg_k}")
        input_request=b'AUTH sad;SET_DEGF'
        s_rl.sendto(input_request, ("127.0.0.1", 23457))
        # output_responce, real_server_addr = s_rl.recvfrom(1024)
        s_rl.sendto(request_1, ("127.0.0.1", 23457))
        output_responce, real_server_addr = s_rl.recvfrom(1024)
        deg_f=float(output_responce.strip())
        # print(f"deg f start={deg_f}")
        request_2=b'AUTH sad;SET_DEGC'
        # request_3=b'AUTH sad;GET_TEMP'
        s_rl.sendto(input_request, ("127.0.0.1", 23457))
        s_rl.sendto(input_request, ("127.0.0.1", 23457))
        s_rl.sendto(request_1, ("127.0.0.1", 23457))
        s_rl.sendto(request_2, ("127.0.0.1", 23457))
        s_rl.sendto(request_1, ("127.0.0.1", 23457))
        output_responce, real_server_addr = s_rl.recvfrom(1024)
        s_rl.close()
        deg_f2=float(output_responce.strip())
        # print(f"deg f2={deg_f2}")
        self.assertAlmostEqual(deg_f,deg_f2, places=0,msg="Temperature shouldn't change much
in time it takes to call SET_DEGF twice")
if __name__ == '__main__':
    run_specified_tests([TestUnitChange])
```

Vuln 4: Lack of ability to confirm set units.

Affected Component(s)

- SampleNetworkServer.py:SmartNetworkThermometer

Description

A vulnerability was discovered in the ‘SampleNetworkServer.py’ file not only on lines 79 & 80 where the GET_TEMP command does not return the temperature unit but also lines 73-78 where the ‘SET_DEGF, SET_DEGC, SET_DEGK’ commands to set the units does not return any sort of confirmation. This can compromise the integrity of the incubator’s configurations as an attacker or default temperature setting ‘self.deg’ could change the temperature unit which would not be displayed causing a medical professional to respond inappropriately.

Affected Requirements

- CWE-392
- Our Requirement 16

Recommended Fix

The remediation is to add code to the server that everytime the client requests the temperature, the units are returned as well. Further, commands that attempt to set the units should return a response confirming the change was successful.

Vuln 5: Vulnerable to Decode Exceptions

Affected Component(s)

- SampleNetworkServer.py:SmartNetworkThermometer:run

Description

A vulnerability was discovered in the ‘SampleNetworkServer.py’ file on line 88, where the run function is unable to handle any other type of encryption other than utf-8 encoded strings to the server. This can compromise the availability of the incubator itself as it will crash with a decoding exception on the server.

Affected Requirement(s)

- Our Requirements 25,26
- CWE-703
- CWE-390

Exploit of Decode Vuln

Exploit for Decode

```
import unittest
import socket
class TestDecode(unittest.TestCase):

    def test_decode_exception(self):
        s_rl = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
        s_rl.settimeout(3)
        request_1 = 'AUTH ρƆƆƆ;'.encode('utf-16')
        s_rl.sendto(request_1, ("127.0.0.1", 23456))
        try:
            token, real_server_addr = s_rl.recvfrom(1024)
            token = token.strip()
        except (socket.timeout, socket.gaierror) as err:
            self.fail(msg="No response to invalidly encoded auth leaves client hanging and causes
uncaught exception.")

if __name__ == '__main__':
    run_specified_tests([TestDecode])
```

Recommended Fix

The easiest fix would be to add under line 116 the ability to catch UnicodeDecodeError exceptions. When handling this exception, the server could send the client the message, "Bad Command" or something of that nature.

Example Code Fix

Updated Run Function

```
    def run(self) : # fthe running function
        while True :
            try :
                msg, addr = self.serverSocket.recvfrom(1024)
                msg = msg.decode("utf-8").strip()
                cmds = msg.split(' ')
                if len(cmds) == 1 : # protected commands case
                    semi = msg.find(';')
```

```
            if semi != -1 : #if we found the semicolon
                #print (msg)
                if msg[:semi] in self.tokens : #if its a valid token
                    self.processCommands(msg[semi+1:], addr)
                else :
                    self.serverSocket.sendto(b"Bad Token\n", addr)
            else :
                    self.serverSocket.sendto(b"Bad Command\n", addr)
        elif len(cmds) == 2 :
            if cmds[0] in self.open_cmds : #if its AUTH or LOGOUT
                self.processCommands(msg, addr)
            else :
                self.serverSocket.sendto(b"Authenticate First\n", addr)
        else :
            # otherwise bad command
            self.serverSocket.sendto(b"Bad Command\n", addr)

    except IOError as e :
        if e.errno == errno.EWOULDBLOCK :
            #do nothing
            pass
        else :
            #do nothing for now
            pass
        msg = ""
    # Handle Decode issues
    except UnicodeDecodeError:
        self.serverSocket.sendto(b"Bad Command, Invalid encoding\n", addr)


    self.updateTemperature()
    time.sleep(self.updatePeriod)
```

Vuln 6: Server is unable to safely catch Keyboard Interrupts.

Affected Component(s):

- SampleNetworkServer.py:SmartNetworkThermometer:run

Description

A vulnerability was discovered in the ‘SampleNetworkServer.py’ file on line 116 within the IOERROR handling when the server receives a SIGINT (CTRL + C) command, the server will crash unexpectedly. This can compromise the availability and stability of the incubator itself as the keystrokes/input are very common and could be performed by mistake; it will crash with an unhandled exception.

Affected Requirement(s)

- Our Requirements 26
- CWE-703
- CWE-390

Recommended Fix

The easiest fix would be to add under line 116 the ability to catch the Interruption & exceptions and instead implement a shutdown command. When handling this exception, the server could send the client the message, "Bad Command" or something of that nature.

Example Code Fix

Updated Run Function

```
def run(self) : #the running function
    while True :
        try :
            msg, addr = self.serverSocket.recvfrom(1024)
            msg = msg.decode("utf-8").strip()
            cmds = msg.split(' ')
            if len(cmds) == 1 : # protected commands case
                semi = msg.find(';')
                if semi != -1 : #if we found the semicolon
                    #print (msg)
                    if msg[:semi] in self.tokens : #if its a valid token
                        self.processCommands(msg[semi+1:], addr)
                    else :
                        self.serverSocket.sendto(b"Bad Token\n", addr)
                else :
                        self.serverSocket.sendto(b"Bad Command\n", addr)
            elif len(cmds) == 2 :
                if cmds[0] in self.open_cmds : #if its AUTH or LOGOUT
                    self.processCommands(msg, addr)
                else :
```

```
                self.serverSocket.sendto(b"Authenticate First\n", addr)
        else :
            # otherwise bad command
            self.serverSocket.sendto(b"Bad Command\n", addr)

    except IOError as e :
        if e.errno == errno.EWOULDBLOCK :
            #do nothing
            pass
        else :
            #do nothing for now
            pass
        msg = ""
    # Handle SIGINT
    except KeyboardInterrupt:
        self.serverSocket.sendto(b"Server received SIGINT, closing\n", addr)
        break

    self.updateTemperature()
    time.sleep(self.updatePeriod)
```

Vuln 7: Hardcoded Password

Affected Component(s):

- SampleNetworkServer.py:SmartNetworkThermometer:processCommands
- SampleNetworkClient:SimpleNetworkClient:updateInfTemp
- SampleNetworkClient:SimpleNetworkClient:updateIncTemp

Description

A vulnerability was discovered in the code , which contains a function that passes/stores a hardcoded password value. Hardcoded credentials generally pose a security risk to an application as an attacker can reverse engineer the code to obtain the password to use elsewhere. In this case an attacker would be able to impersonate/seize the client's account as their username/password has been exposed.

Affected Requirement(s)

- Our Requirements 1,2,3,20

Recommended Fix

Move all credentials and sensitive information out of code and store it using secure systems and pass it using configuration management. This will also facilitate easier credential rotation. We further advise that credentials should be scoped to a particular user and/or device.

Example Code Fix

```
parser= configparser.ConfigParser(strict=False, interpolation=None)
parser.read(filenames='config.ini')
password=parser['configs']["PASSWORD"]
```

Vuln 8: Token list can infinitely grow

Affected Component(s):

- SampleNetworkServer.py:SmartNetworkThermometer:processCommands

Description

A vulnerability was discovered in the ‘SampleNetworkServer.py’ part of the application which allows the array “self.tokens”, that is used to store generated token values for each authenticated session, to continuously grow if authenticated sessions continually occur without ‘logouts’. This vulnerability can lead to performance issues should the list become too long and could also lead to a buffer overflow if attacked. Plaintext session tokens continuously added to a list overall can compromise active sessions within the application as an attacker can easily identify which user/session is running on the system and could potentially terminate a client’s session and can also lead to a Man-in-the-middle attack, which allows an attacker to obtain an unencrypted token and use it to either impersonate and act on behalf of a legitimate user or to observe traffic between the server and the host.

Affected Requirement(s)

- Our Requirements 27
- CWE-390

Recommended Fix

Limit the number of tokens per valid authentication password and force older tokens to be logged out or expire prior to issuing new tokens.

Example Code Fix Snippet

```
def processCommands(self, msg, addr):
        print(msg)
        cmds = msg.split(';')
        for c in cmds:
            cs = c.split(' ')
            if len(cs) == 2:  # should be either AUTH or LOGOUT
                if cs[0] == "AUTH":
                    if cs[1] == self.hashed_password:
                        if len(self.tokens) > 9:  # Added Check for number of active tokens
                            print("Token Limit Reached")
                            msg = self.crypto.encrypt(b"Too many active tokens. Please Logout a
session.")
                            self.serverSocket.sendto(msg, addr)
                        else:
                            self.tokens.append(secrets.token_urlsafe(16))
                            self.serverSocket.sendto(self.crypto.encrypt(self.tokens[-1].encode('utf-8')),
addr)
                        # print (self.tokens[-1])
                elif cs[0] == "LOGOUT":
                    if cs[1] in self.tokens:
                        self.tokens.remove(cs[1])
```

Conclusions

In this report we have provided a table of device assets and a data flow diagram of the current products specifications and design. We have then used these to develop a threat model and then performed a risk analysis on this list of threats. It is our opinion that the current version of the product is not ready for launch as is. We can defend this conclusion by looking at the risk assessment, the identified vulnerabilities and the recommended changes.

For the vulnerabilities we have identified, each of them map to a core risk presented in the threat table and model as well as mapping to risk ratings/severities. Each vulnerability was found to cause serious, lasting and damaging impacts to both user safety and device/data integrity and are out of compliance with FDA postmarket guidelines for baseline device recommendations. For each of these vulnerabilities we have included a number of recommended vulnerability patches which we believe if implemented mitigate or even eliminate the specific threat and risk it is mapped to in this report and should bring the device in FDA compliance for that specific vulnerability. We have also provided recommended security requirements for this application which we believe should be considered when redesigning or patching the device going forward.

For the risk assessment we have taken each threat presented and have performed an analysis to gauge the severity of each against how likely these threats are to occur, how likely they are to succeed, and what impacts they may cause. From this assessment there were 8 high level risks and 1 critical risk identified which were assessed to potentially cause Equipment Damage, Network compromises/outages, User/account compromises, Mistreatment/diagnoses, Injury and death.

In conclusion, it is our opinion that this product is not ready for release given that it contains numerous vulnerabilities, could potentially cause injury or death and is out of compliance with government regulations. Furthermore should the company decide to proceed with launch we would have no ability to push updates that correct any of these known vulnerabilities or any other technical bugs that may be discovered without an extremely costly rollout process that has not been established.This product should goes back for a complete redesign so that it may incorporate as many vulnerability patch suggestions as possible before the system is ready for launch again.

References

- NIST SP 800-66r2 Implementing the Helath Insurance Portability and Accountability Act (HIPPA) Security Rule: A Cybersecurity Resource Guide
- Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions- Draft Guidance for Industry and Food and Drug Administration Staff
- FDA Premarket notification guideline: https://www.fda.gov/medical-devices/premarket-submissions-selecting-and-preparing-correct-submission/premarket-notification-510k
- FDA Postmarket regulation guideline: https://www.fda.gov/media/95862/download
- FDA Premarket Approval: https://www.fda.gov/medical-devices/premarket-submissions-selecting-and-preparing-correct-submission/premarket-approval-pma
- ISO guide for medical device risk management: https://www.iso.org/standard/72704.html
- Example for a certifying body https://www.iecee.org/dyn/ww w/f?p=106:48:0