Group 18

ISSEM

Summer 2022

8/16/22

Final - Infant Incubator

Alexander Lotero, Albert Yen, Carlos Thomas, Faisal Qasim

aal562@nyu.edu, acy238@nyu.edu, carlos.thomas@nyu.edu, fq2014@nyu.edu

1. **Security Requirements**

Write Security Requirements for the Infant Incubator.[1]

Requirement #	Requirement Component applies to (either component or overall system)	Requirement Description	Is this requirement met in the final product
1.	SampleNetworkServer.py SampleNetworkClient.py	Pseudo-random number generators must be secure and not predictable.	No
2.	SampleNetworkServer.py	Command execution must be prevented without successful authentication.	No
3.	Infinc.py SampleNetworkServer.py SampleNetworkClient.py	Input must not be accepted before proper validation.	No
4.	SampleNetworkServer.py SampleNetworkClient.py	Hardcoded credentials shall not be used.	No
5.	Overall	All data shall be encrypted in transit and at rest.	No
6.	Overall	The incubator, and incubator implementation, must meet all applicable regulatory security requirements.	Unknown

[1] Requirements seven through 24 are based on the Center for Internet Security (CIS) Critical Security Controls which can be found here: https://learn.cisecurity.org/control-download

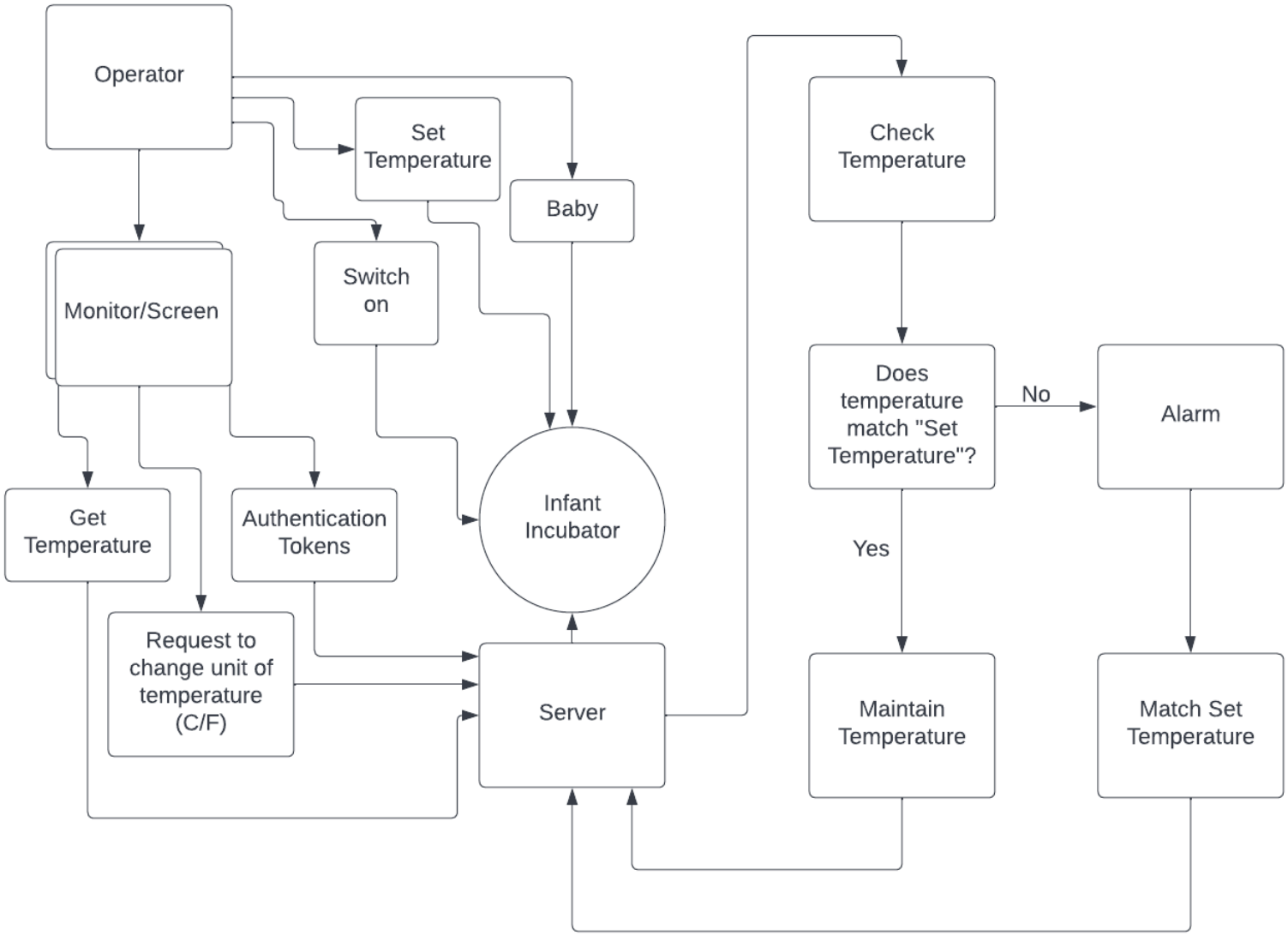
7.	Overall	Actively manage (inventory, track, and correct) all enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/Internet of Things (IoT) devices; and servers) connected to the infrastructure physically, virtually, remotely, and those within cloud environments, to accurately know the totality of assets that need to be monitored and protected within the enterprise. This will also support identifying unauthorized and unmanaged assets to remove or remediate.	Unknown
8.	Overall	Actively manage (inventory, track, and correct) all software (operating systems and applications) on the network so that only authorized software is installed and can execute, and that unauthorized and unmanaged software is found and prevented from installation or execution.	Unknown
9.	Overall	Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.	Unknown
10.	Overall	Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications).	Unknown

11.	Overall	Use processes and tools to assign and manage authorization to credentials for user accounts, including administrator accounts, as well as service accounts, to enterprise assets and software.	Unknown
12.	Overall	Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software.	Unknown
13.	Overall	Develop a plan to continuously assess and track vulnerabilities on all enterprise assets within the enterprise's infrastructure, in order to remediate, and minimize, the window of opportunity for attackers. Monitor public and private industry sources for new threat and vulnerability information.	Unknown
14.	Overall	Collect, alert, review, and retain audit logs of events that could help detect, understand, or recover from an attack.	Unknown
15.	Overall	Improve protections and detections of threats from email and web vectors, as these are opportunities for attackers to manipulate human behavior through direct engagement.	Unknown
16.	Overall	Prevent or control the installation, spread, and execution of malicious applications, code, or scripts on enterprise assets.	Unknown

17.	Overall	Establish and maintain data recovery practices sufficient to restore in-scope enterprise assets to a pre-incident and trusted state.	Unknown
18.	Overall	Establish, implement, and actively manage (track, report, correct) network devices, in order to prevent attackers from exploiting vulnerable network services and access points.	Unknown
19.	Overall	Operate processes and tooling to establish and maintain comprehensive network monitoring and defense against security threats across the enterprise's network infrastructure and user base.	Unknown
20.	Overall	Establish and maintain a security awareness program to influence behavior among the workforce to be security conscious and properly skilled to reduce cybersecurity risks to the enterprise.	Unknown
21.	Overall	Develop a process to evaluate service providers who hold sensitive data, or are responsible for an enterprise's critical IT platforms or processes, to ensure these providers are protecting those platforms and data appropriately.	Unknown
22.	Overall	Manage the security life cycle of in-house developed, hosted, or acquired software to	Unknown

		prevent, detect, and remediate security weaknesses before they can impact the enterprise.	
23.	Overall	Establish a program to develop and maintain an incident response capability (e.g., policies, plans, procedures, defined roles, training, and communications) to prepare, detect, and quickly respond to an attack.	Unknown
24.	Overall	Test the effectiveness and resiliency of enterprise assets through identifying and exploiting weaknesses in controls (people, processes, and technology), and simulating the objectives and actions of an attacker.	Unknown

2. Data Flow Diagram (DFD)



3. Asset List

Asset #	Asset Name	Asset Description	Concrete or Abstract Asset?	Critical Asset or no?
1.	Fan	Helps with purification and sanitization	Concrete	Yes
2.	Heater	Helps maintain infant's body temperature	Concrete	Yes
3.	Air Distributors	Distributes air evenly to ensure equal temperature within the incubator	Concrete	Yes
4.	Canopy	Clear covering that protects the baby from the outside germs and maintain warm environment	Concrete	Yes
5.	Filters	Helps clean the air	Concrete	Yes
6.	Hygrometer	Measures humidity within the incubator	Concrete	Yes
7.	Port Holes	Holes to reach the baby without having to	Concrete	Yes

		open the canopy every time a nurse needs to reach the patient. Helps with not contaminating the baby's environment		
8.	Infant	Baby in the the incubator	Concrete	Yes
9.	Server Hardware	System that manages the computing components of the incubator	Concrete	Yes
10.	Authenticated users token list	List of tokens of users that are allowed to login	Abstract	Yes
11.	Client accounts	List of user accounts allowed to login	Abstract	No
12.	Inlets	Manages oxygen, medications and IV fluids	Concrete	Yes
13.	Respiratory Tubing	Provides artificial oxygen to the baby	Concrete	Yes

4. **Threat List**

Threat #	Affected Component(s)	Threat Category (if applicable)	Threat Severity (if available)	Recommended Control
1.	infinc.py - infant incubator simulator, the incubator session and the energy/temperature	DOS - denial of service	Medium	Type check on input parameters for the function addEnergy.
2.	infinc.py - infant incubator simulator, the incubator session and the environmental temperature	DOS - denial of service	Medium	Type check on input parameters for the function simulateTransferWithChamber.
3.	SampleNetworkServer - the network server and personal data: its token list	Information disclosure	High	Avoid standard pseudo-random generators and use cryptographically safe generators.
4.	SampleNetworkClient - the network Client	Tampering	High	Avoid token in plain text and use cryptographically safe generators.

5. **Risk Analysis**

Take at least 3 threats from Threat List and analyze the Risk level (provide justification)

Risk #	Risk Name	Related Threat (if applicable)	Likelihood	Impact	Risk level
1.	Tampering or theft of login	Tampering	High	Severe	High - token in plain text is easy to find.

	credentials				
2.	Distributed DOS on all incubator simulators	DOS	High	Severe	High - it is easy to give a function incorrect parameters so that the parameters cause the function to glitch.
3.	Predict tokens and spoof credentials	Information disclosure	High	Severe	High - it is easy to guess the returned result of a standard pseudo-random generator.

6. **Software Analysis (from Labs)**

Take the results of your Lab 9 submission and include those (or more vulnerabilities) in the following table. No need to do further software analysis or coding.

Vulnerability #	Affected Component(s)	Description	Affected Requirement #	Recommended Fix
1. (#1 from lab 8)	SampleNetworkServer - “tokens,” the list of authenticated clients SampleNetworkClient - legitimate, authenticated clients	By using a standard pseudo-random number generator, token entries may be predictable in a context requiring unpredictability. It may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate a legitimate, authenticated user.	1, 7, 10	A recommendation for improvement is to implement a pseudo random number generator that is known to be secure (i.e. with greater bits of security). An example is the “secrets” library for Python.
2. (#2 from lab 8)	SampleNetworkServer - sensitive data on the server (e.g. current temperature)	By chaining commands onto the initial authentication attempt, clients can execute commands on the server (e.g. GET_TEMP) without successfully authenticating first. Malicious/chained command: "*AUTH wrongtoken;GET_TEMP*"	2, 7, 8, 10	When processing commands, the server should identify if an authentication request is among them, and if so, reject any additional commands until authentication is completed/successful (see lab 9).

Vulnerability #	Affected Component(s)	Description	Affected Requirement #	Recommended Fix
3. (#3 from lab 8) 4. (#4 from lab 8)	infinc.py - infant incubator simulator, the incubator session, and the energy/temperature SampleNetworkServer - availability of simulator data SampleNetworkClient - availability of simulator data	The *addEnergy(self, e)* [vuln. #3] function within "infinc.py" does not check if the given parameter "e" is an integer/float/or double, hence, if given a string, it cannot add "e" with self.energy because self.energy is an integer, and you cannot add a string with an integer and the program will fail and return a type error. Adversaries can consistently give invalid parameters and cause the incubator model to fail, which can lead to a denial of service attack. This same vulnerability can be seen in the *simulateTransferWithChamber()*, also found in "infinc.py" [vuln. #4].	3, 7, 10	To patch these two vulnerabilities, a check (if statement) should be added to the beginning of both of the vulnerable functions: *addEnergy()* and *simulateTransferWithChamber()*, to confirm that the stated parameters are of a compatible type (see lab 9).

Vulnerability #	Affected Component(s)	Description	Affected Requirement #	Recommended Fix
5. (#1 from labs 5 and 6)	SampleNetworkServer - confidentiality of server data and functions SampleNetworkClient - integrity of client account	According to MITRE CWE-259: “if hard-coded passwords are used, it is almost certain that malicious users will gain access through the account in question.”	4, 7, 10	We recommend an implementation of the python-dotenv library. Python-dotenv reads key-value pairs from a .env file and can set them as environment variables. This requires implementation on both the server and the client.
6. (#3 from labs 5 and 6)	SampleNetworkClient - confidentiality of the authentication token and integrity of client account SampleNetworkServer - confidentiality of server data and functions	The authentication token, in its current implementation in SampleNetworkClient.py, is sent to the server in the clear, unencrypted. Meaning, in order to become an authenticated user and gain all the privileges that that entails, all an attacker must do is intercept this packet.	5, 7, 10	To fix this plaintext authentication token vulnerability, an encryption scheme must be deployed. Our recommendation is to create a pre-shared symmetric key scheme, storing the appropriate keys on both the client and server machines, respectively.

7. Conclusions

Based on your consolidated analysis, answer the following question: In your opinion, is this product ready for launch as is, ready for launch with the recommended fixes, or needs complete redesign? Argue your case based on the previous sections.

To start, it is the opinion of this group that in its current state, before any of the recommendations stated above are implemented, this product is not ready to launch. This conclusion was not drawn only because of the high severity of the reported vulnerabilities, but also how easy many of them may be to discover. Take vulnerability number two from the software analysis table above. This vulnerability, in which clients can receive sensitive data and execute server commands without proper authentication, does not necessarily require a thorough code analysis to become apparent to a malicious actor. A client may just wish to expedite the command-issuing process, and to do so may choose to chain several commands together in their initial message to the server, i.e. the initial authentication request. In doing so, they may submit an authentication token with a typo in it, or in the case of an attacker, they may intentionally submit an incorrect token just to observe the server's behavior. In doing so, they will soon discover that, in its current state, authentication is not required so long as commands are chained to the authentication message. We make several assumptions within this hypothetical scenario, but it is all to demonstrate the ease with which this vulnerability could be discovered. It is because of this ease and the sheer number of identified vulnerabilities that we recommend that the product not be launched in its current state.

So, we have explained why the product should not be launched "as is," but the question remains: will it be ready for launch after the recommended fixes? Or does the product need a complete redesign? It might seem like the obvious opinion of a security architect that a delayed launch is a small price to pay for an improved security qualification. Nonetheless, it is the opinion of this group that the product should receive a complete redesign before release. In addition to the details of our findings reported above, we still suspect the presence of additional "somewhat-easy-to-discover" vulnerabilities that, with more time and manpower, should still be discovered and patched before launch. The testing and analysis performed up to this point does not match that of a proper penetration test in thoroughness of testing and in turn, results.

Our full recommendation would be to perform a complete redesign of the product with the advice of security engineers and architects taken at each step of the process. From there we would encourage a full penetration test on what becomes of the product following the

redesign, with patching performed on all discovered vulnerabilities within financial feasibility. These steps can help ensure that the product is launched with the best security stature possible within the organization’s resource constraints. The process will contribute to the viability of the product and the long-term reputation of the organization.