Course Project

Gate System (CCTV)

2021 LG Security Specialist
Team 2

Phase I

Secure Development

Team Charter (Phase I)

Role	Description	Members
Program Manager	Manage the project schedule & requirements and documentation	- *Gigwan Lee* - *Heejung Jeoung*
Architect	Responsible for the system architecture	- *Wonwoo Kim*
Implementation (Server)	Responsible for the server side (Jetson Nano) implementation	- *Wonwoo Kim* - *Bokyoung Ku* - *Heejung Jeoung*
Implementation (Client)	Responsible for the client side, UI.	- *Ukheon Jeong* - *Gigwan Lee*
Security	Responsible for the secure coding & function testing	- *Bokyoung Ku*
Mentor	Mentor	- *David Belasco*

- Contact info : lg-security-specialist-team2@googlegroups.com
- Github : https://github.com/jacob-ku/specialist-team2

Security Requirements (SQUARE-Lite)

2. Identify Assets and Security Goals

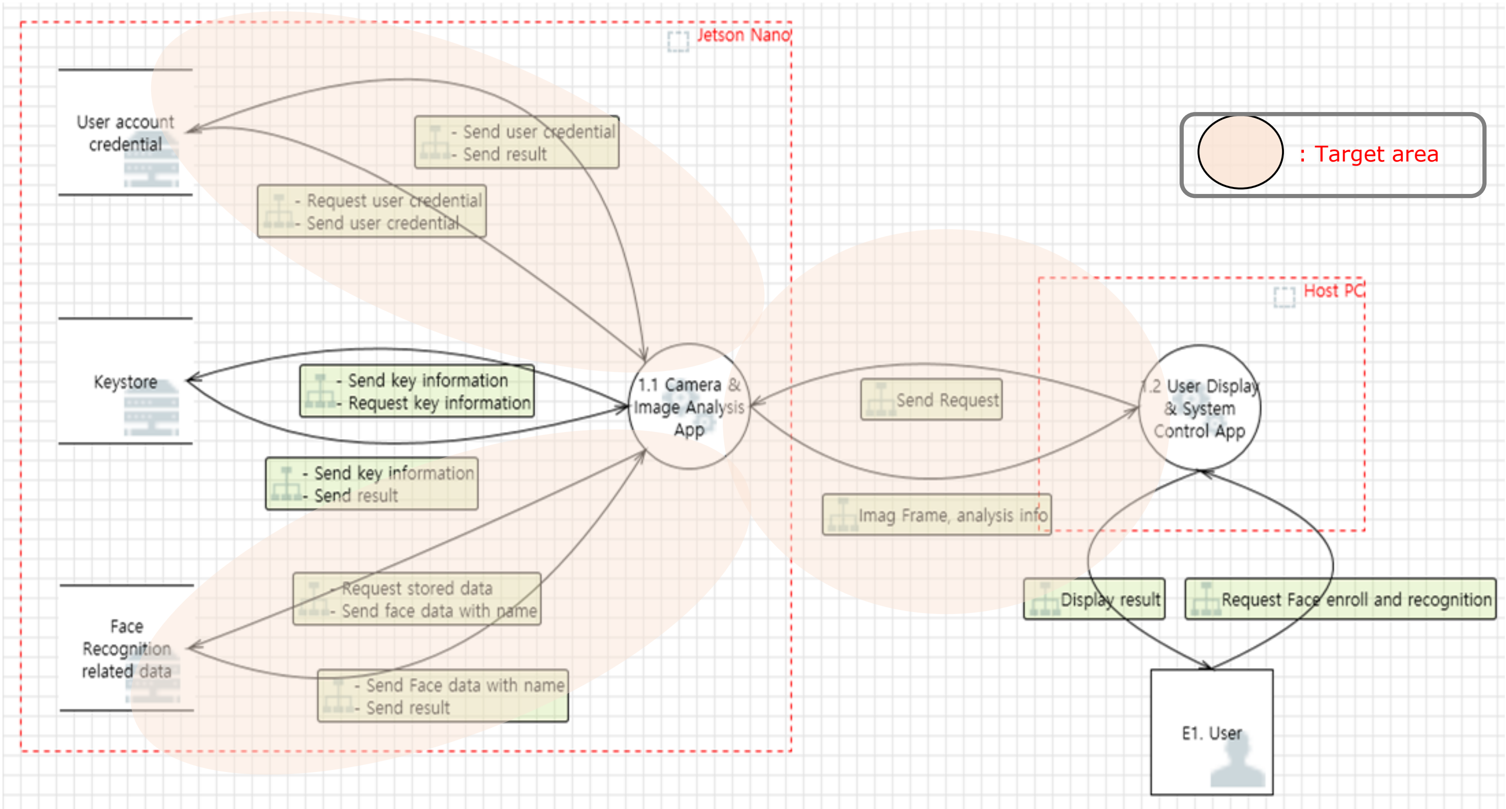
Goals	Contents
Business Goals	Provide a face recognition system to identify employees.
Security Goals	Recognized face images and image analyzed results which is personal/sensitive information must be protected while transmitting on the network.
	User credential and stored images have to be protected.
	Security weakness and vulnerabilities after launching the system must be minimized as much as possible.

Assets	Location
Captured face images (PII)	Transmitted over the network ,Stored in the server side storage
Added face images (PII)	Stored in the server side storage
Image analyzed results (PII)	Transmitted over the network
FaceNet trained model files, CNN (Convolutional Neural Network) trained model files	Stored in the server side storage
User credential	Transmitted over the network ,Stored in the server side storage

Security Requirements (SQUARE-Lite)

3. Perform risk assessment

ThreatModeling


Jetson Nano
Host PC
: Target area
User account credential
Keystore
Face Recognition related data
- Send user credential
- Send result
- Request user credential
- Send user credential
- Send key information
- Request key information
- Send key information
- Send result
- Request stored data
- Send face data with name
- Send Face data with name
- Send result
1.1 Camera & Image Analysis App
Send Request
Imag Frame, analysis info
1.2 User Display & System Control App
Display result
Request Face enroll and recognition
E1. User

Threats and Mitigation

Threat Scenario

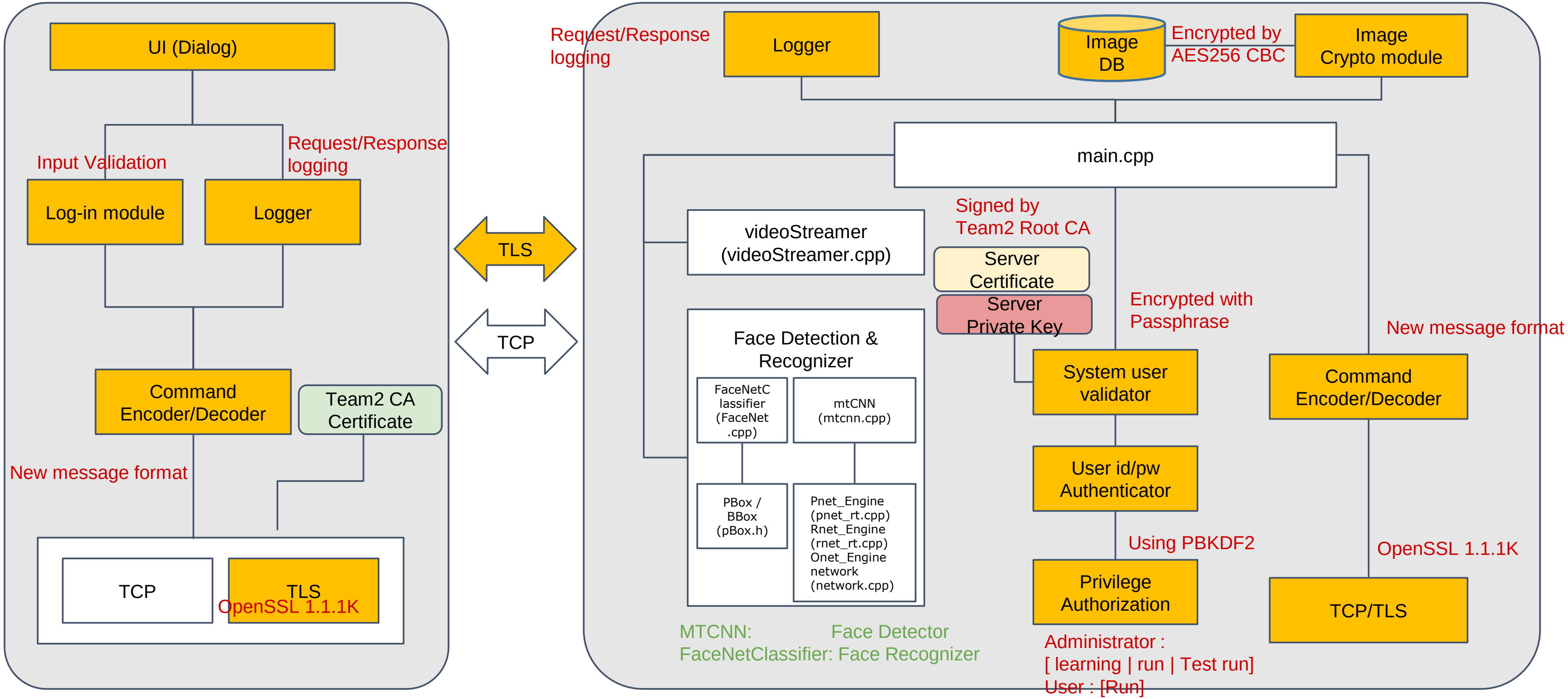
An unauthorized individual gains access to the GSS thru GSC and tries to add/modify/delete face images. The system detects the malicious behavior and prevents the unauthorized individual's actions.

Category	Threats	Mitigation
Information Disclosure	The stored face image related data on server side can be disclosed to an unauthorized user. The transmitting data on the communication between server /client can be disclosed to unauthorized user.	Provide encryption on the stored data on server side Provide encrypted the communication channel between server/client
Spoofing, Elevation of Privilege	Unauthorized user can run the program without any restriction.	Provide login functionality for user authentication/authorization
Tampering	Unauthorized user can manipulate the request/response	Validate requests/responses from each peer
Denial of Service	There is a limitation of resources on embedded application. Server application may not work properly due to massive request from clients.	Manage the connection between server/client
Repudiation	No logging feature for tracking the activities of the applications	Add logging on server/client

Implementation

Secure Architecture

Implement required enhancements to the system based on the security requirement elicited.



Input Validation

Request/Response logging

New message format

OpenSSL 1.1.1K

Request/Response logging

Encrypted by AES256 CBC

Signed by Team2 Root CA

Encrypted with Passphrase

New message format

Using PBKDF2

OpenSSL 1.1.1K

MTCNN: Face Detector
FaceNetClassifier: Face Recognizer

Administrator :
[learning | run | Test run]
User : [Run]

GSC (Gate System Client)

GSS (Gate System Server)

Security Evaluation

During the software development, we found some issues that need to be addressed in the source code. And some of them were fixed with secure coding.

static_analysis_FlawFinder

Static analysis by Flawfinder

Module	Found	Fixed	Result
GSC	5	5	● Change srand() to RAND_bytes()
GSS	57	3	● Support safe string API is required to prevent buffer overflow ● Static buffer is used to read encrypted file name. Therefore if the checking of file name length is insufficient, buffer overflow can be occurred.

TestCases

Test Case

Module	Total	Pass	Fail
GSC	23	23	0
GSS	46	43	3

Security Evaluation - Vulnerability List

Module	Category	Description
Client	DoS	Log file storage size checking is required to prevent denial of service.
	Insecure Configuration	Limiting the number of user login attempt is required to prevent brute force attack.
Server	Memory Corruption	Support safe string API is required to prevent buffer overflow
	Memory Corruption	Static buffer is used to read encrypted file name. Therefore if the checking of file name length is insufficient, buffer overflow can be occurred.
	DoS	Log file storage size checking is required to prevent denial of service.
	Protocol Error	When the server is running as non-secure mode and the client tries to connect to server as secure mode, it causes hang on both sides
Image Storage	DoS	Image file storage size checking is required to prevent denial of service.
Crypto	Insecure Configuration	Our program didn’t implement TLS mutual authentication. Therefore fake client can communicate with the server. This may lead to spoofing attack.
Face Recognition Model	Insecure Configuration	Model files for image recognition engine are not protected. This may lead information leakage.

Lesson & Learned

- ❑ Security area is new to me, I learned the process about enforcing security in software development.

- ❑ It was a good chance to apply what we have learned on the project.

- ❑ Taking enough time to consider security in the development process can only lead to safe software development.

- ❑ There are too many security consideration and features to implement the project and they were not fully implemented because I don’t have enough implementation experience of security and knowledge of security related libraries. Even if it is not sufficient, this project helped me have more security knowledge and experience.

- ❑ The good thing is we could discuss the project with variant perspectives on security because we are from different division with different domain knowledge.

- ❑ The one of the flawed approaches is that most programmers trust the source of the input and implicitly trust all data entering their application.

Phase II

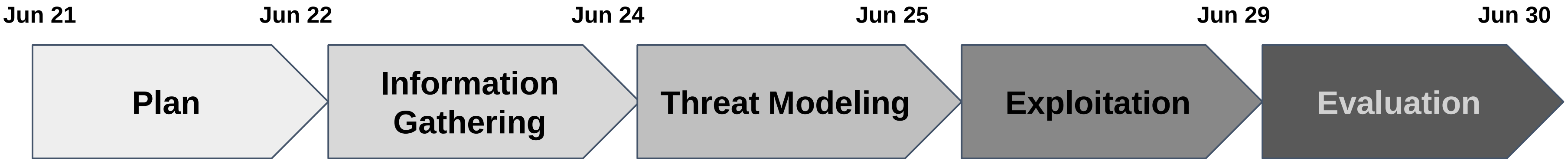
Security Analysis of Classmate System

Team Charter (Phase II)

Role	Description	Members
Program Manager	Assessment planning, documentation	● *Bokyoung Ku* ● *Heejung Jeoung*
Static Analysis	Responsible for static analysis	● *Heejung Jeoung* ● *Wonwoo Kim* ● *Gigwan Lee*
Review Artifacts (server)	Responsible for the server side artifacts	● *Wonwoo Kim* ● *Bokyoung Ku*
Review Artifacts (client)	Responsible for the client side artifacts	● *Ukheon Jeong* ● *Gigwan Lee*
Exploitation	Responsible for exploitation	● *Ukheon Jeong* ● *Gigwan Lee* ● *Bokyoung Ku* ● *Wonwoo Kim*
Mentor	Mentor	● *David Belasco*

- **Contact info : lg-security-specialist-team2@googlegroups.com**
- **Github : https://github.com/jacob-ku/specialist-team2**

Project Schedule (Phase II)



Date	Key Milestone	Task	Artifacts
Jun 21 ~ Jun 22	Plan	- Define roles	- Team Charter - Project Schedule
Jun 22 ~ Jun 24	Information Gathering	- Review Artifacts - Static analysis tool - Scanning the system	- Gathered information
Jun 24 ~ Jun 25	Threat Modeling	- Prioritize the assessment	- Expected threat list - Documentation
Jun 25 ~ Jun 29	Exploitation	- Run fuzzing tool - Perform penetration Test	- Expected vulnerability list - Documentation
Jun 29 ~ Jun 30	Evaluation	- Evaluate the vulnerabilities	- Project Final Report - Vulnerability Assessment Report

Information Gathering (1)

Review provided artifacts : architecture design document / configuration / source code review

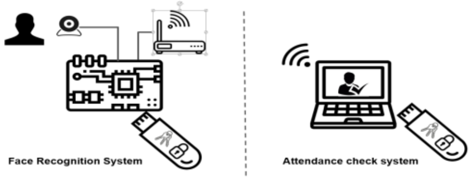
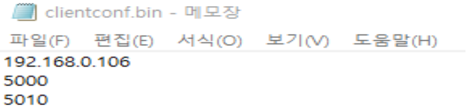
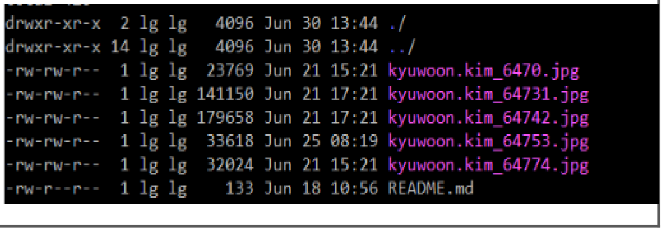
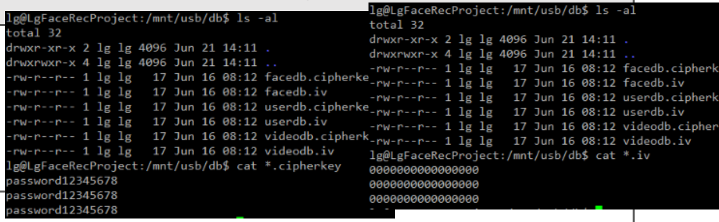
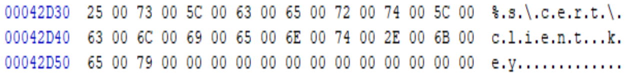
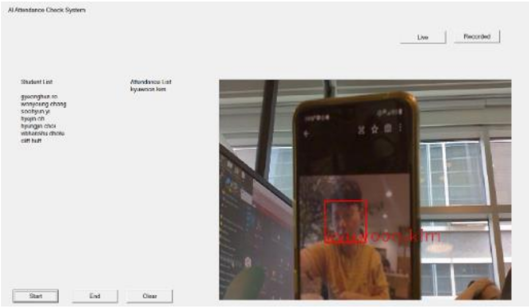
Module	Category	Finding	How
Client/Server	Repudiation	There is no way to track the activities of the systems when applications are terminated.	No logging files in client/server local storage
Server	Insecure Configuration	Since the key is stored in a USB, that may lead to insecure default behavior if malwares are in the USB	
Client	Information Disclosure	Server IP/port information is disclosed in client conf.bin and that may be a start of being the attacker's target.	 clientconf.bin - 메모장 파일(F) 편집(E) 서식(O) 보기(V) 도움말(H) 192.168.0.106 5000 5010

Image Storage	Information Disclosure/ Tampering	No encryption on image files on the local storage. Attackers can get the student information from the image file name or add/modify/delete the image files if the attacker has access to the system.	
			`drwxr-xr-x  2 lg lg   4096 Jun 30 13:44 ./`
			`drwxr-xr-x 14 lg lg   4096 Jun 30 13:44 ../`
			`-rw-rw-r--  1 lg lg  23769 Jun 21 15:21 kyuwoon.kim_6470.jpg`
			`-rw-rw-r--  1 lg lg 141150 Jun 21 17:21 kyuwoon.kim_64731.jpg`
			`-rw-rw-r--  1 lg lg 179658 Jun 21 17:21 kyuwoon.kim_64742.jpg`
			`-rw-rw-r--  1 lg lg  33618 Jun 25 08:19 kyuwoon.kim_64753.jpg`
			`-rw-rw-r--  1 lg lg  32024 Jun 21 15:21 kyuwoon.kim_64774.jpg`
			`-rw-r--r--  1 lg lg    133 Jun 18 10:56 README.md`

Information Gathering (2)

Review provided artifacts : architecture design document / configuration / source code review

Module	Category	Finding	How
Cryptography	Cryptographic Vulnerability	The key files for authentication have the same encryption key / IV.	
	Information Disclosure	we can check the path of the private key and the certificate partially/fully by Hex Editor	
Face Recognition	Logic Errors	The system cannot distinguish between the picture and the real person.	

```
lg@LgFaceRecProject:/mnt/usb/db$ ls -al
total 32
drwxr-xr-x 2 lg lg 4096 Jun 21 14:11 .
drwxrwxr-x 4 lg lg 4096 Jun 21 14:11 ..
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 facedb.cipherke
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 facedb.iv
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 userdb.cipherke
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 userdb.iv
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 videodb.cipherk
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 videodb.iv
lg@LgFaceRecProject:/mnt/usb/db$ cat *.cipherkey
password12345678
password12345678
password12345678
```

```
lg@LgFaceRecProject:/mnt/usb/db$ ls -al
total 32
drwxr-xr-x 2 lg lg 4096 Jun 21 14:11 .
drwxrwxr-x 4 lg lg 4096 Jun 21 14:11 ..
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 facedb.cipherk
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 facedb.iv
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 userdb.cipherk
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 userdb.iv
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 videodb.cipher
-rw-r--r-- 1 lg lg   17 Jun 16 08:12 videodb.iv
lg@LgFaceRecProject:/mnt/usb/db$ cat *.iv
0000000000000000
0000000000000000
0000000000000000
```

```
00042D30  25 00 73 00 5C 00 63 00 65 00 72 00 74 00 5C 00  %.s.\.c.e.r.t.\.
00042D40  63 00 6C 00 69 00 65 00 6E 00 74 00 2E 00 6B 00  c.l.i.e.n.t...k.
00042D50  65 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00  e.y.............
```

AI Attendance Check System

Student List

gyeonghun so
wonyoung chang
soohyun yi
hyojin oh
hyungjin choi
vishanshu dhole
cliff huff

Attendance List
kyuwoon kim

Information Gathering (3)

Static Analysis :

Team3_static_analysis_by_Team2

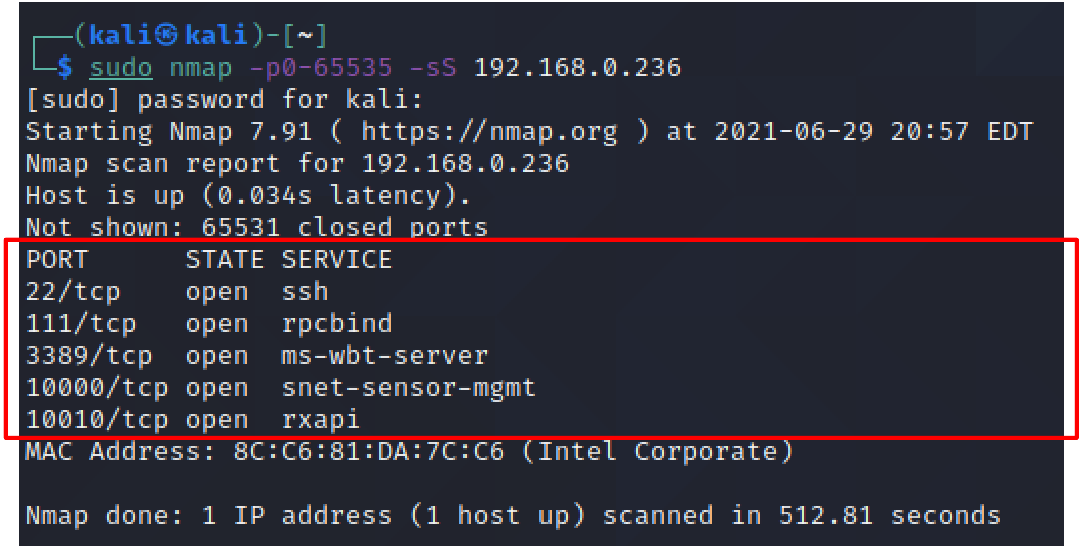
Tool	Target	Found	Summary
Flawfinder (https://dwheeler.com/flawfinder/)	Client	13	Using safe string API/handling buffer API is required to prevent buffer overflow
	Server	3	Checking buffer boundaries are required in face recognition module (Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20))
Trommel (https://github.com/CERTCC/trommel)	Client	39	Server IP/Port information is disclosed in clientconf.bin and gives hints to DoS attack.
	Server	25	keywords such as password/username/ssl/admin are detected and gives hints when reviewing source codes

Information Gathering (5)

System Scanning by nmap

Result of nmap scanning

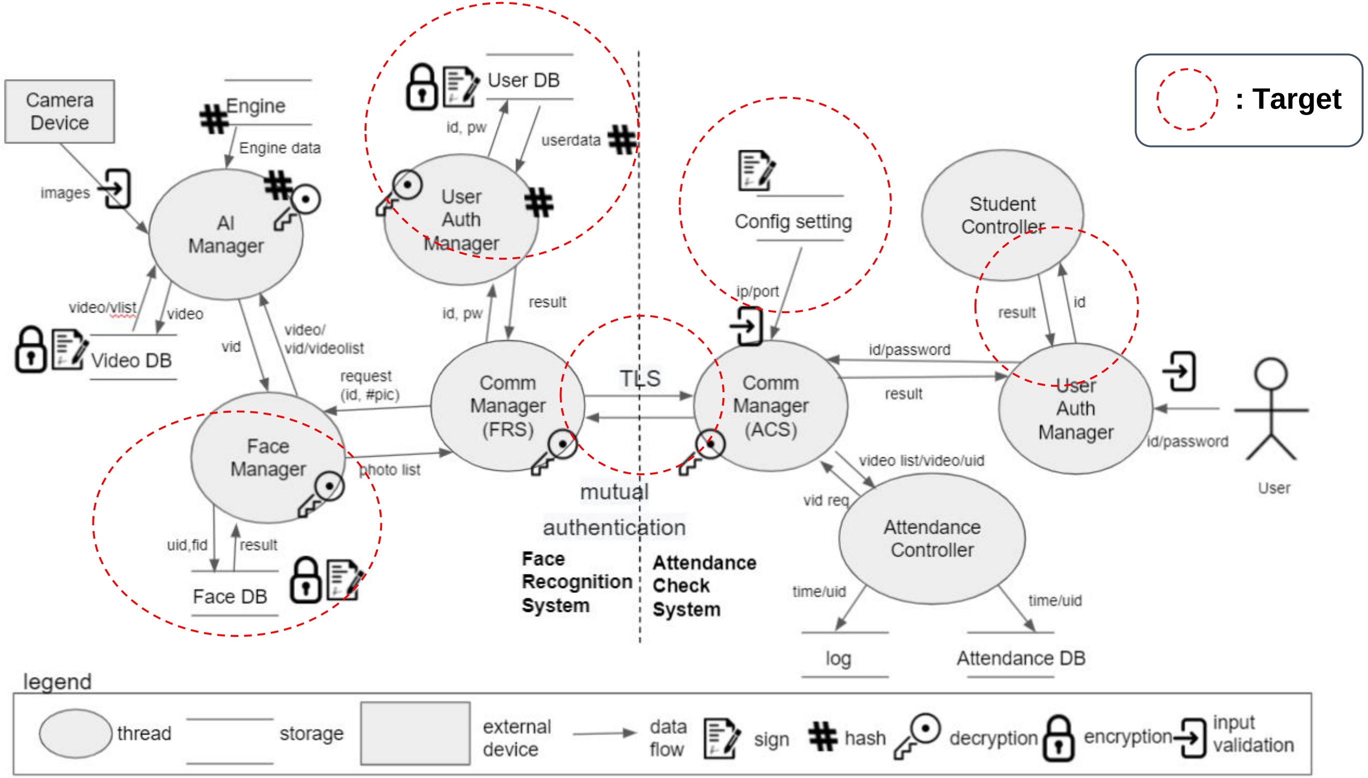
> *nmap -p0-65535 -sS 192.168.0.236*



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p0-65535 -sS 192.168.0.236
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-29 20:57 EDT
Nmap scan report for 192.168.0.236
Host is up (0.034s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
3389/tcp  open  ms-wbt-server
10000/tcp open  snet-sensor-mgmt
10010/tcp open  rxapi
MAC Address: 8C:C6:81:DA:7C:C6 (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 512.81 seconds
```

Threat Modeling (1)

Establish the scope of assessment and identify assets



Threat Modeling (2)

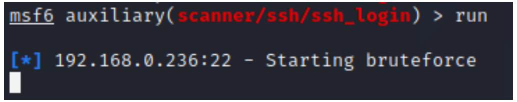
Identify the possible security risks through the analysis of assets, threats and vulnerabilities by their impacts and likelihood.

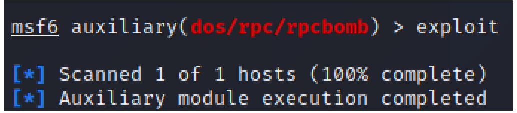
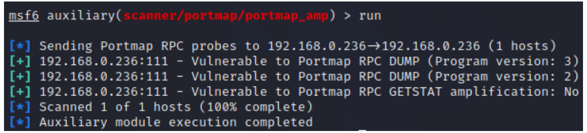
Module	Asset	Threat Category	Threat scenario	Impact Level
Client	Config Setting	Information Disclosure	Unauthorized user can open the configuration file (plain text file) of client and check the server information (IP/port). Based on the information, the attacker conducts the system scanning to gather more information.	High
	Transmitted data	Spoofing	Server may be spoofed by an attacker and the server may grant the unauthorized access of fake client.	High
Server	User DB	Information Disclosure	ID/PW can be disclosed by Brute Force attack.	High
		Tampering	Malicious input such as ID/PW can crash the server application	Medium
	Face DB	Tampering	The attackers can check the local storage of the server and get the student information from the image file name or add/modify/delete the image files.	High
	Transmitted data	Spoofing	Client may be spoofed by an attacker and this may lead to unauthorized access to the server.	High

Exploitation - opened port(1)

Attempt to identify potential threats and vulnerabilities throughout the services of the listening port.

Port	Service	Possible threats	Result

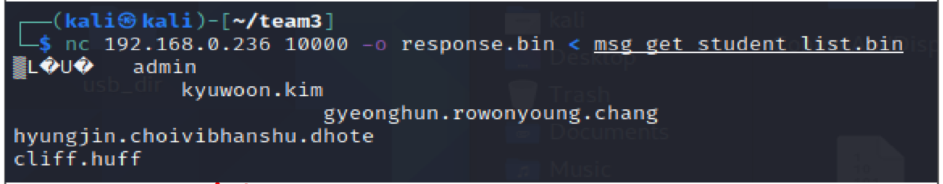
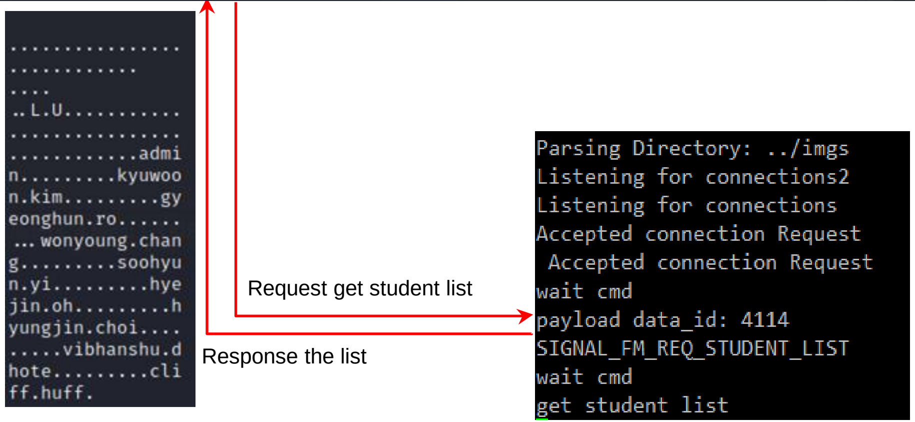
port 22	openSSH 7.6p1
	(latest 8.6p1)
		1) Known vulnerability (CVEs)
		2) Brute force attack to gain access
		a. Download pwned password list
		b. Do brute force attack using metasploit
			1) There are several CVEs but we can’t exploit that.
			2) We couldn’t get the success result during 3 days.
			
			`msf6 auxiliary(scanner/ssh/ssh_login) > run`
			`[*] 192.168.0.236:22 - Starting bruteforce`

port 111	rpcbind	1) Known vulnerability (CVEs)
		2) Exploit mapped service
			1) CVE-2017-8779 : DOS
			: Server frozen for a while after exploiting but we are not sure this is effective exploitation.
			
			`msf6 auxiliary(dos/rpc/rpcbomb) > exploit`
			`[*] Scanned 1 of 1 hosts (100% complete)`
			`[*] Auxiliary module execution completed`
			2) Got some info, but couldn’t get deeper in time.
			
			`msf6 auxiliary(scanner/portmap/portmap_amp) > run`
			`[*] Sending Portmap RPC probes to 192.168.0.236→192.168.0.236 (1 hosts)`
			`[+] 192.168.0.236:111 - Vulnerable to Portmap RPC DUMP (Program version: 3)`
			`[+] 192.168.0.236:111 - Vulnerable to Portmap RPC DUMP (Program version: 2)`
			`[+] 192.168.0.236:111 - Vulnerable to Portmap RPC GETSTAT amplification: No`
			`[*] Scanned 1 of 1 hosts (100% complete)`
			`[*] Auxiliary module execution completed`

port 3389	ms-wbt-server	1) Known vulnerability (CVEs) 2) Brute force attack to gain access	1) Recently, there are no known vulnerabilities in ms terminal server. 2) It is same with port 22. It’s security depends on user id/pw of the system.

Exploitation - opened port(2)

Attempt to identify potential threats and vulnerabilities throughout the services of the listening port.

Port	Service	Possible threats	Result
port 10000	Team3 TCP	1) Unknown connection with manipulated packet	1) Server sent student list by manipulated request. a. Sniffing packets when admin is logged in b. Create manipulated packet file (msg_get_student_list.bin) c. Send manipulated packet to TCP port(10000) using nc  

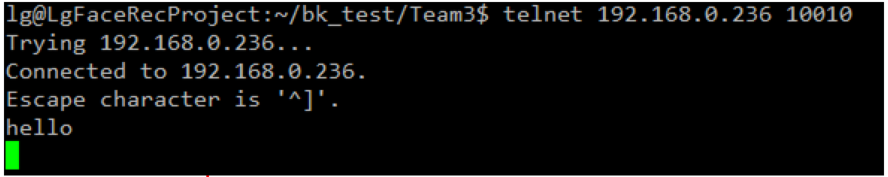
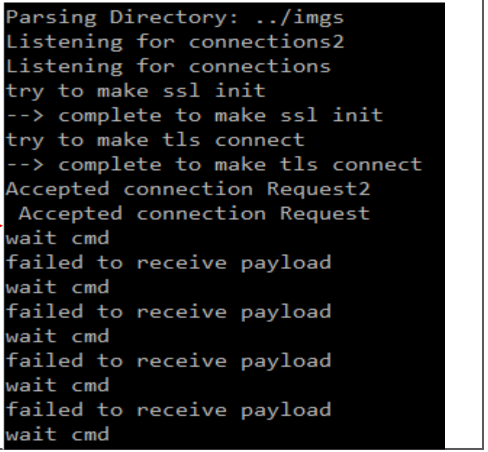
```
┌──(kali㉿kali)-[~/team3]
└─$ nc 192.168.0.236 10000 -o response.bin < msg_get_student_list.bin
▒L�U�    admin
             kyuwoon.kim
                          gyeonghun.rowonyoung.chang
hyungjin.choivibhanshu.dhote
cliff.huff
```

```
................
............
....
..L.U...........
................
............admi
n.........kyuwoo
n.kim.........gy
eonghun.ro......
...wonyoung.chan
g.........soohyu
n.yi.........hye
jin.oh.........h
yungjin.choi....
.....vibhanshu.d
hote.........cli
ff.huff.
```

```
Parsing Directory: ../imgs
Listening for connections2
Listening for connections
Accepted connection Request
 Accepted connection Request
wait cmd
payload data_id: 4114
SIGNAL_FM_REQ_STUDENT_LIST
wait cmd
get student list
```

Exploitation - opened port(3)

Attempt to identify potential threats and vulnerabilities throughout the services of the listening port.

Port	Service	Possible threats	Result
port 10010	Team3 TLS	1) Unknown connection with manipulated packet	1) Server program fall into an infinite loop a. Connect to TLS port(10010) using telnet b. Sent “hello”  lg@LgFaceRecProject:~/bk_test/Team3$ telnet 192.168.0.236 10010 Trying 192.168.0.236... Connected to 192.168.0.236. Escape character is '^]'. hello  Parsing Directory: ../imgs Listening for connections2 Listening for connections try to make ssl init --> complete to make ssl init try to make tls connect --> complete to make tls connect Accepted connection Request2 Accepted connection Request wait cmd failed to receive payload wait cmd failed to receive payload wait cmd failed to receive payload wait cmd failed to receive payload wait cmd

Exploitation - Fuzzing

Fuzzing - Client (1)

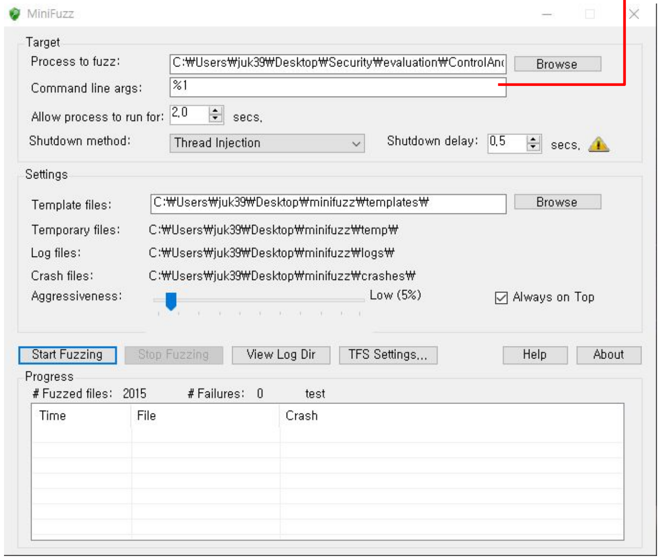
Tool	Target	Environment	Attack Scenario & Result
MiniFuzz	[Client, Server] ID & PW	● Windows 10 ● Visual Studio 2019	[Client] We had run 2,000 times as below but segmentation fault wasn’t found. 1. Rebuild the client after adding source code to use the program arguments 2. Generate randomly manipulated user id and password 3. Run the program in order to login using random id and password 4. Check the result of program execution 5. Repeat step 2-4 [Server] **No crash but after repeating 500 times of connection,the server cannot initialize SSL.** (Availability Issue)

Exploitation - Fuzzing

Fuzzing - Client (2)

```
UserAuthView::DlgProc()
{
    switch(message) {
        case WM_INITDIALOG:
            CString exeStr(__targv[0]);
            CString arguStr(__targv[1]);
            CString randStr, idStr, pwStr;

            int strIdx = arguStr.Find(_T("test-"));
            strIdx += strlen("test-");

            randStr = arguStr.Right(arguStr.GetLength() - strIdx);

            srand((unsigned int)(time(NULL));
            int random_number = rand() % randStr.GetLength();

            idStr = randStr.Left(random_number);
            pwStr = randStr.Right(randStr.GetLength() - random_number);

            SetDlgItemText(hWnd, IDC_USERNAME_EDIT, T2W(idStr.GetBuffer()));
            SetDlgItemText(hWnd, IDC_PASSWORD_EDIT, T2W(pwStr.GetBuffer()));
    }
}
```

Variable generation



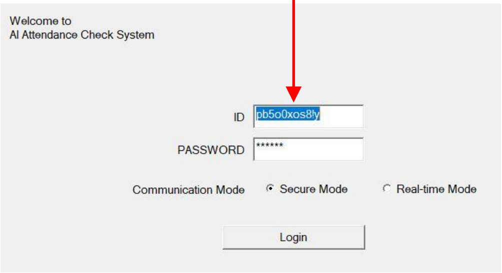
Target
Process to fuzz: C:₩Users₩juk39₩Desktop₩Security₩evaluation₩ControlAn(
Command line args: %1
Allow process to run for: 2.0 secs.
Shutdown method: Thread Injection
Shutdown delay: 0.5 secs.

Settings
Template files: C:₩Users₩juk39₩Desktop₩minifuzz₩templates₩
Temporary files: C:₩Users₩juk39₩Desktop₩minifuzz₩temp₩
Log files: C:₩Users₩juk39₩Desktop₩minifuzz₩logs₩
Crash files: C:₩Users₩juk39₩Desktop₩minifuzz₩crashes₩
Aggressiveness: Low (5%)
☑ Always on Top

Progress
Fuzzed files: 2015 # Failures: 0 test

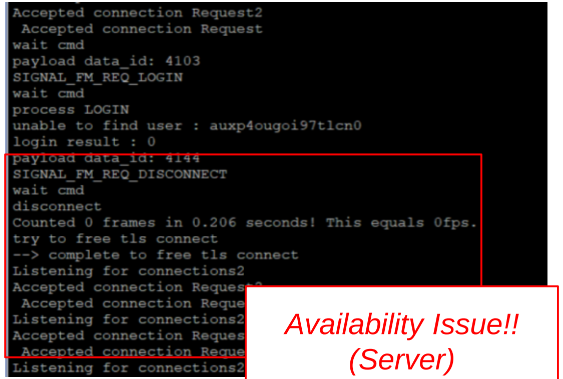
Time	File	Crash

Inputs and execution



Welcome to
AI Attendance Check System

ID pb5o0xos8ly
PASSWORD ******
Communication Mode ● Secure Mode ○ Real-time Mode



```
Accepted connection Request2
 Accepted connection Request
wait cmd
payload data_id: 4103
SIGNAL_FM_REQ_LOGIN
wait cmd
process LOGIN
unable to find user : auxp4ougoi97tlcn0
login result : 0
payload data_id: 4144
SIGNAL_FM_REQ_DISCONNECT
wait cmd
disconnect
Counted 0 frames in 0.206 seconds! This equals 0fps.
try to free tls connect
--> complete to free tls connect
Listening for connections2
Accepted connection Reques
 Accepted connection Reque
Listening for connections2
Accepted connection Reques
 Accepted connection Reque
Listening for connections2
```

Availability Issue!!
(Server)

Exploitation - Fuzzing

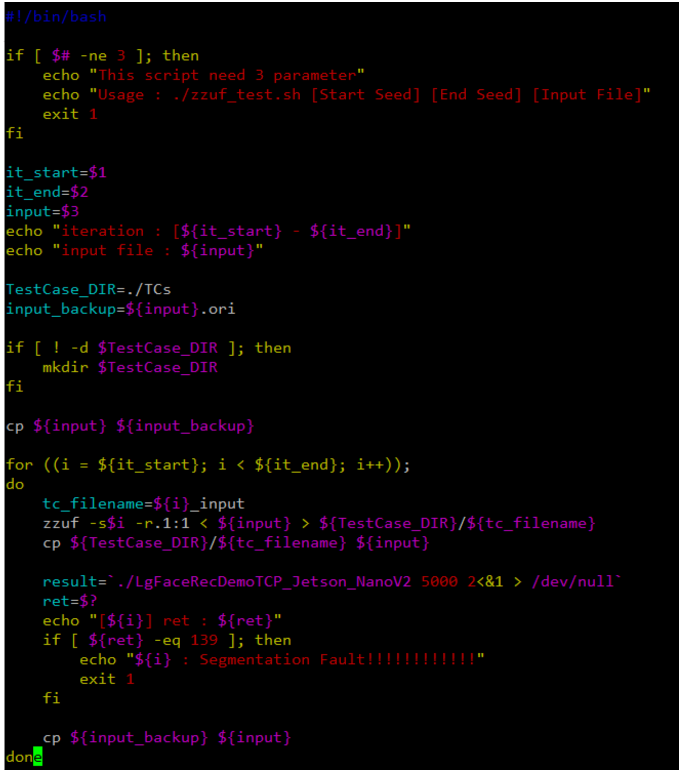
Fuzzing - Server(1)

Tool	Target	Environment	Attack Scenario & Result
AFL	[Server] User DB file	Jetson Nano	AFL doesn’t support coverage based fuzzing on ARM environment.
zzuf	[Server] User DB file	VM Kali Linux	**We had run over 30,000 times but segmentation fault wasn’t found**. 1. Rebuild program after removing source code related to face recognition 2. Generate randomly manipulated user db file 3. Run the program in order to read abnormal user db file 4. Check the result of program execution 5. Repeat step 2-4
	[Server] Registered Image file	Jetson Nano	**We had run over 10,000 times but segmentation fault wasn’t found.** 1. Rebuild program after removing source code related to socket 2. Generate randomly manipulated jpg file 3. Run the program in order to read abnormal jpg file 4. Check the result of program execution 5. Repeat step 2-4

Exploitation - Fuzzing

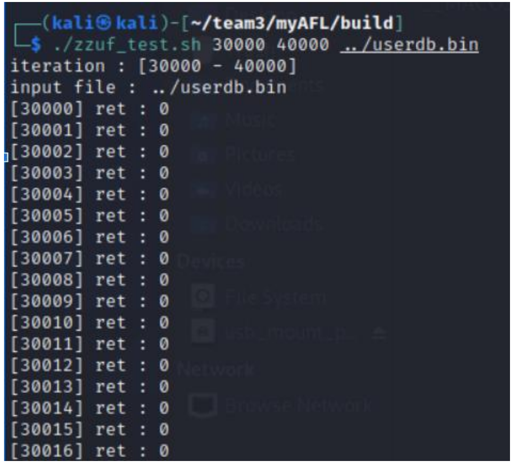
Fuzzing - Server(2)

1) Write the shell script



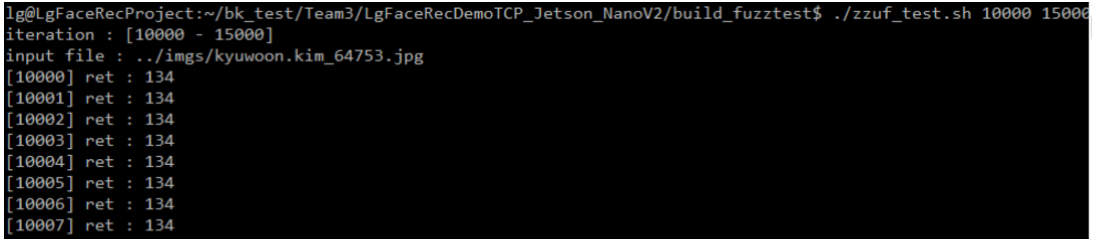
```
#!/bin/bash

if [ $# -ne 3 ]; then
    echo "This script need 3 parameter"
    echo "Usage : ./zzuf_test.sh [Start Seed] [End Seed] [Input File]"
    exit 1
fi

it_start=$1
it_end=$2
input=$3
echo "iteration : [${it_start} - ${it_end}]"
echo "input file : ${input}"

TestCase_DIR=./TCs
input_backup=${input}.ori

if [ ! -d $TestCase_DIR ]; then
    mkdir $TestCase_DIR
fi

cp ${input} ${input_backup}

for ((i = ${it_start}; i < ${it_end}; i++));
do
    tc_filename=${i}_input
    zzuf -s$i -r.1:1 < ${input} > ${TestCase_DIR}/${tc_filename}
    cp ${TestCase_DIR}/${tc_filename} ${input}

    result=`./LgFaceRecDemoTCP_Jetson_NanoV2 5000 2<&1 > /dev/null`
    ret=$?
    echo "[${i}] ret : ${ret}"
    if [ ${ret} -eq 139 ]; then
        echo "${i} : Segmentation Fault!!!!!!!!!!!!"
        exit 1
    fi

    cp ${input_backup} ${input}
done
```

2-1) Launch the server program with manipulated userdb.bin



```
┌──(kali㉿kali)-[~/team3/myAFL/build]
└─$ ./zzuf_test.sh 30000 40000 ../userdb.bin
iteration : [30000 - 40000]
input file : ../userdb.bin
[30000] ret : 0
[30001] ret : 0
[30002] ret : 0
[30003] ret : 0
[30004] ret : 0
[30005] ret : 0
[30006] ret : 0
[30007] ret : 0
[30008] ret : 0
[30009] ret : 0
[30010] ret : 0
[30011] ret : 0
[30012] ret : 0
[30013] ret : 0
[30014] ret : 0
[30015] ret : 0
[30016] ret : 0
```

2-2) Launch the server program with manipulated image file



```
lg@LgFaceRecProject:~/bk_test/Team3/LgFaceRecDemoTCP_Jetson_NanoV2/build_fuzztest$ ./zzuf_test.sh 10000 15000
iteration : [10000 - 15000]
input file : ../imgs/kyuwoon.kim_64753.jpg
[10000] ret : 134
[10001] ret : 134
[10002] ret : 134
[10003] ret : 134
[10004] ret : 134
[10005] ret : 134
[10006] ret : 134
[10007] ret : 134
```

Exploitation - Pen Testing

Penetration Testing with attack scenario(1)

- Manipulate the image files

Pre-condition

Attacker gained the access to the server system

Attack Scenario

1) A normal user login to the client.

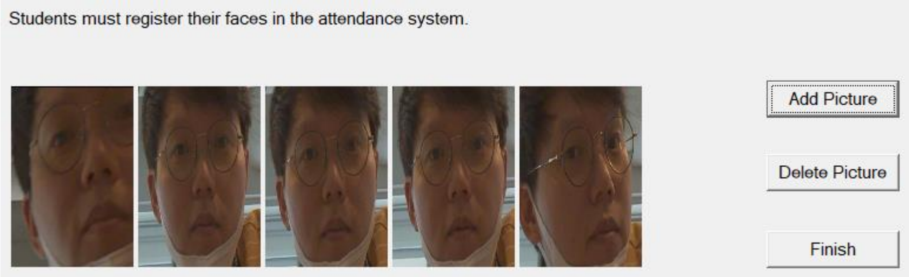
2) Add pictures

3) Log out

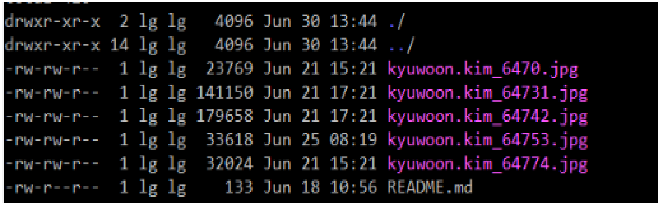
4) Then, the attacker replaces one of the images of the user with another one in the server storage.

Attack Result

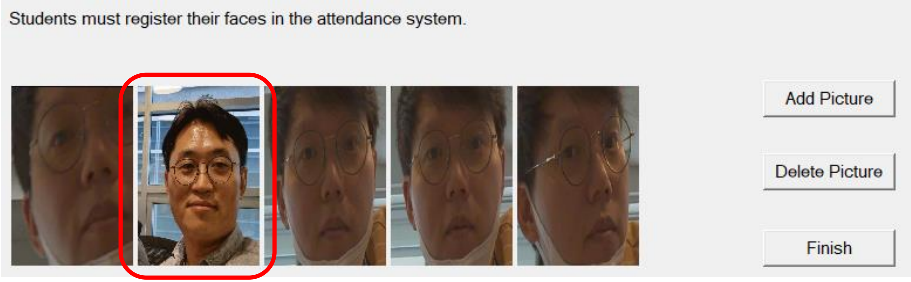
Unauthorized user can pass the attendance system



Students must register their faces in the attendance system.



```
drwxr-xr-x  2 lg lg   4096 Jun 30 13:44 ./
drwxr-xr-x 14 lg lg   4096 Jun 30 13:44 ../
-rw-rw-r--  1 lg lg  23769 Jun 21 15:21 kyuwoon.kim_6470.jpg
-rw-rw-r--  1 lg lg 141150 Jun 21 17:21 kyuwoon.kim_64731.jpg
-rw-rw-r--  1 lg lg 179658 Jun 21 17:21 kyuwoon.kim_64742.jpg
-rw-rw-r--  1 lg lg  33618 Jun 25 08:19 kyuwoon.kim_64753.jpg
-rw-rw-r--  1 lg lg  32024 Jun 21 15:21 kyuwoon.kim_64774.jpg
-rw-r--r--  1 lg lg    133 Jun 18 10:56 README.md
```



Students must register their faces in the attendance system.

Exploitation - Pen Testing

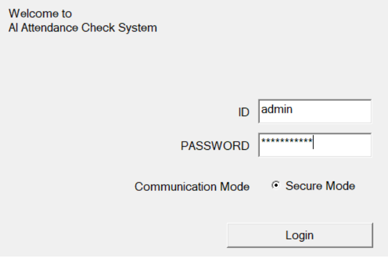
Penetration Testing with attack scenario(2)

- Attack Flow

Hacked!

[1] Man In The Middle attack

Attendance Check Client



Welcome to
AI Attendance Check System

ID admin

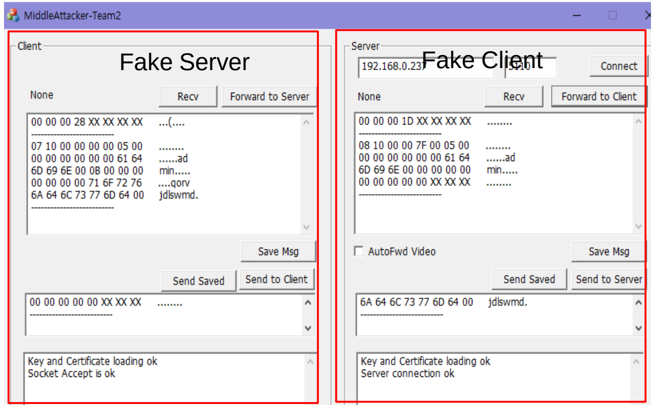
PASSWORD ***********

Communication Mode ● Secure Mode

- Network Configuration (clientconfig.bin)
- Client Key
- Client Certificate
- Root CA

ARP spoofing

MITM[1] Attacker



Client:

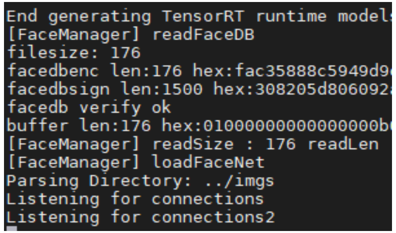
```
None

00 00 00 28 XX XX XX XX    ...(....
-----------------------
07 10 00 00 00 00 05 00    ........
00 00 00 00 00 00 61 64    ......ad
6D 69 6E 00 0B 00 00 00    min.....
00 00 00 00 71 6F 72 76    ....qorv
6A 64 6C 73 77 6D 64 00    jdlswmd.
-----------------------

00 00 00 00 00 XX XX XX    ........
-----------------------

Key and Certificate loading ok
Socket Accept is ok
```

Server: 192.168.0.237 5110

```
None

00 00 00 1D XX XX XX XX    ........
-----------------------
08 10 00 00 7F 00 05 00    ........
00 00 00 00 00 00 61 64    ......ad
6D 69 6E 00 00 00 00 00    min.....
00 00 00 00 00 XX XX XX    ........
-----------------------

□ AutoFwd Video

6A 64 6C 73 77 6D 64 00    jdlswmd.
-----------------------

Key and Certificate loading ok
Server connection ok
```

- Client Key
- Client Certificate

Attendance Check Server



```
End generating TensorRT runtime models
[FaceManager] readFaceDB
filesize: 176
facedbenc len:176 hex:fac35888c5949d9
facedbsign len:1500 hex:308205d806092
facedb verify ok
buffer len:176 hex:0100000000000000b
[FaceManager] readSize : 176 readLen
[FaceManager] loadFaceNet
Parsing Directory: ../imgs
Listening for connections
Listening for connections2
```

- Server Key
- Server Certificate
- Root CA
- AI Engines (Recognition /Detection)
- Database (Face, User)
- Video file for test-run

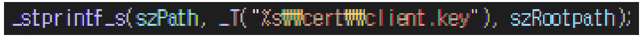
Hack the client PC and copy the keys and certificates for breaking the mutual TLS authentication
(Fortunately, client key file has no passphrase!!!)

Exploitation - Pen Testing

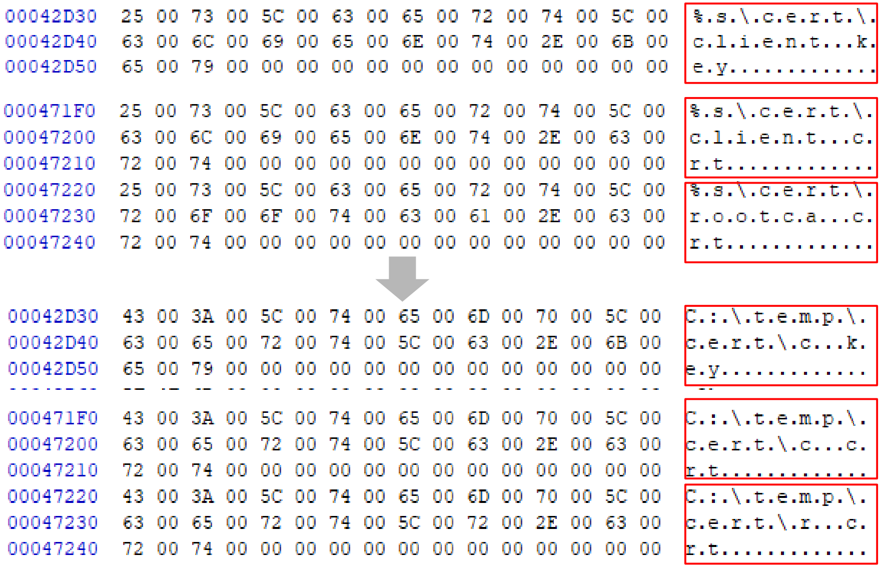
Penetration Testing with attack scenario(2)

■ **Change Key file location using hex editor:**

- Search keys from USB → Search keys from fixed disk storage of attacker's
  "%s\\cert\\client.key" → "C:\\temp\\cert\\c.key"



```
_stprintf_s(szPath, _T("%s₩₩cert₩₩client.key"), szRootpath);
```

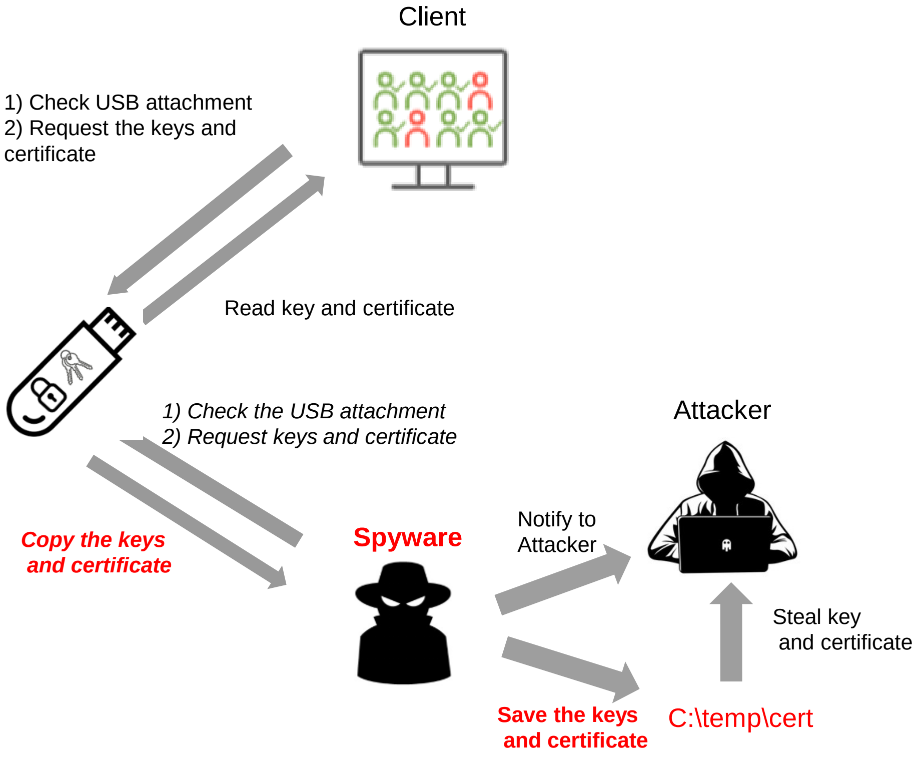


```
00042D30  25 00 73 00 5C 00 63 00 65 00 72 00 74 00 5C 00   %.s.\.c.e.r.t.\.
00042D40  63 00 6C 00 69 00 65 00 6E 00 74 00 2E 00 6B 00   c.l.i.e.n.t...k.
00042D50  65 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00   e.y.............

000471F0  25 00 73 00 5C 00 63 00 65 00 72 00 74 00 5C 00   %.s.\.c.e.r.t.\.
00047200  63 00 6C 00 69 00 65 00 6E 00 74 00 2E 00 63 00   c.l.i.e.n.t...c.
00047210  72 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00   r.t.............
00047220  25 00 73 00 5C 00 63 00 65 00 72 00 74 00 5C 00   %.s.\.c.e.r.t.\.
00047230  72 00 6F 00 6F 00 74 00 63 00 61 00 2E 00 63 00   r.o.o.t.c.a...c.
00047240  72 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00   r.t.............
```

```
00042D30  43 00 3A 00 5C 00 74 00 65 00 6D 00 70 00 5C 00   C.:.\.t.e.m.p.\.
00042D40  63 00 65 00 72 00 74 00 5C 00 63 00 2E 00 6B 00   c.e.r.t.\.c...k.
00042D50  65 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00   e.y.............

000471F0  43 00 3A 00 5C 00 74 00 65 00 6D 00 70 00 5C 00   C.:.\.t.e.m.p.\.
00047200  63 00 65 00 72 00 74 00 5C 00 63 00 2E 00 63 00   c.e.r.t.\.c...c.
00047210  72 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00   r.t.............
00047220  43 00 3A 00 5C 00 74 00 65 00 6D 00 70 00 5C 00   C.:.\.t.e.m.p.\.
00047230  63 00 65 00 72 00 74 00 5C 00 72 00 2E 00 63 00   c.e.r.t.\.r...c.
00047240  72 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00   r.t.............
```

■ **Run spyware on the client to sniff keys and certificates.**

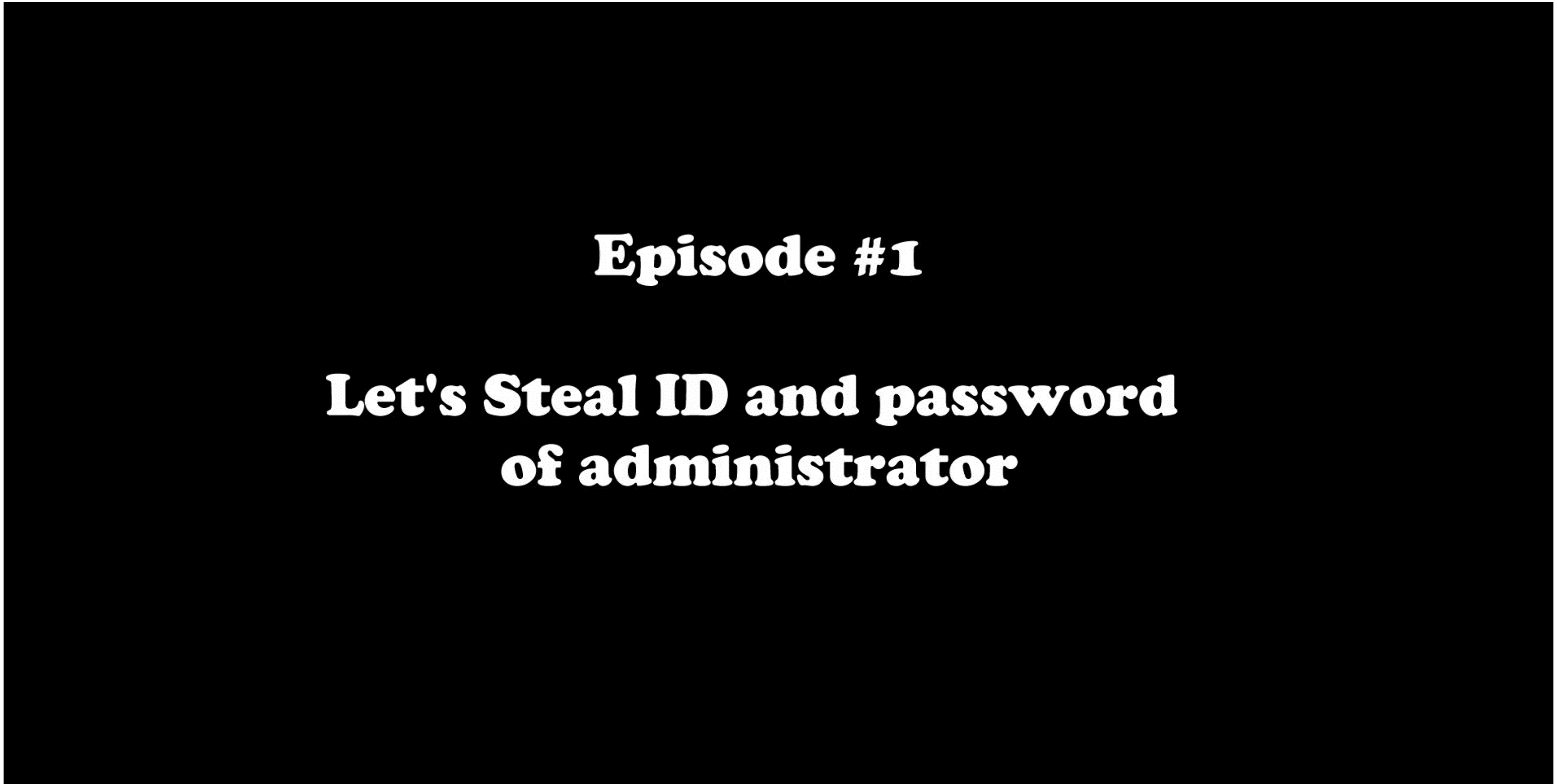
- When USB is attached, check if the key files exist
- If so, save key, certificate, and root CA to other location and notify to the attacker.



Client
1) Check USB attachment
2) Request the keys and certificate
Read key and certificate
1) Check the USB attachment
2) Request keys and certificate
Attacker
Copy the keys and certificate
Spyware
Notify to Attacker
Steal key and certificate
Save the keys and certificate
C:\temp\cert

Exploitation - Demonstration (Video)

Penetration Testing with attack scenario(2)



Episode #1

Let's Steal ID and password
of administrator

Exploitation - Pen Testing

Penetration Testing with attack scenario(2)

What we found!

1. Configuration data on client was not hidden from being viewed.
2. Private keys can be stolen because no encryption applied on the private keys
3. Authentication status is not managed on the server side for the requests.
4. No privilege checking for user account on the server side.

Evaluation

Vulnerability List

vulnerability_list

Severity	Count
Critical	5
High	9
Medium	16
Low	3
Total	**33**

Category	Count
Spoofing	5
Tampering	6
Repudiation	1
Information Disclosure	13
DoS	6
Elevation of Privilege	0
etc.	2
Total	**33**

Lesson & Learned

- ❑ I have learned that evaluating a project in security requires broad knowledge about security.

- ❑ Based on thinking about security vulnerabilities from the attacker’s point of view when analyzing the code, it seems that I can write code that is stronger for security.

- ❑ Before conducting MITM, I considered TLS has no attacker for the network security. However, after the attempt, MITM is a strong hacking technology than I expected. I could find more vulnerabilities of the server and client, and can plan the smart fuzzing with it.

- ❑ Based on what I learned in this course, I felt it was a challenge to find vulnerabilities in open source that are widely used around the world.

- ❑ "Easier" in this case results in less development time but more risk for the product and the end customer. Eliminate default credentials to secure all of your users.

Q & A

Q & A