

Security Analysis of Classmate System

Phase#2

CMU X LGE Security Specialist Course
TEAM5 (DEFENSE)

PURPOSE

Phase #1: We conducted threat modeling and mitigation to develop a safe program by analyzing and evaluating the threats related to the security of the ALPR program

Phase #2 In order to discover and analyze the vulnerabilities of the classmate program, find out how to use various security tools, check the effectiveness, and prove as exploit

Our Team



**Jinhwan
Kim**

Threat Modeling
Risk Assessment
Validation

Surface Analysis
Code Review



**Paul
Lim**

Client App
Development
Client Security

Static Analysis
Reverse Engineering



**Jongsoo
Oh**

Architecture
System/Security
Requirement

Design Analysis
Overall
Architecture



**Sangwook
Lee**

2FA (OTP)
DB, Mail
management

Static Analysis
Runtime Analysis



**Minyong
Ha**

Server Security
JWT
Cryptography
Authentication
/Cryptography
Analysis



**Dawoon
Park**

Server Development
Server Security
TLS, Certificate

Reverse Engineering
Code Review

Schedule for Phase #2

	Mon	Tue	Wed	Thu	Fri
Week 1	Analyze Customer requirement		Define System requirement		
Week 2	Define Secure Requirement & Mitigation				
	Threat modeling & Risk assessment				
			Implement Client / Server		
Week 3	Make test case				
	Implement & integration Client / Server				Presentation 1
Week 4					
	Overall Design Review		Surface/Static/Runtime Analysis		
Week 5					
	Surface/Static/Runtime Analysis				
	Proven vulnerability & POC				Presentation 2

Table of Contents

01

Summarization of Phase #1

Remind activity for secure
development

02

Analysis

Secure analysis of classmate
System

03

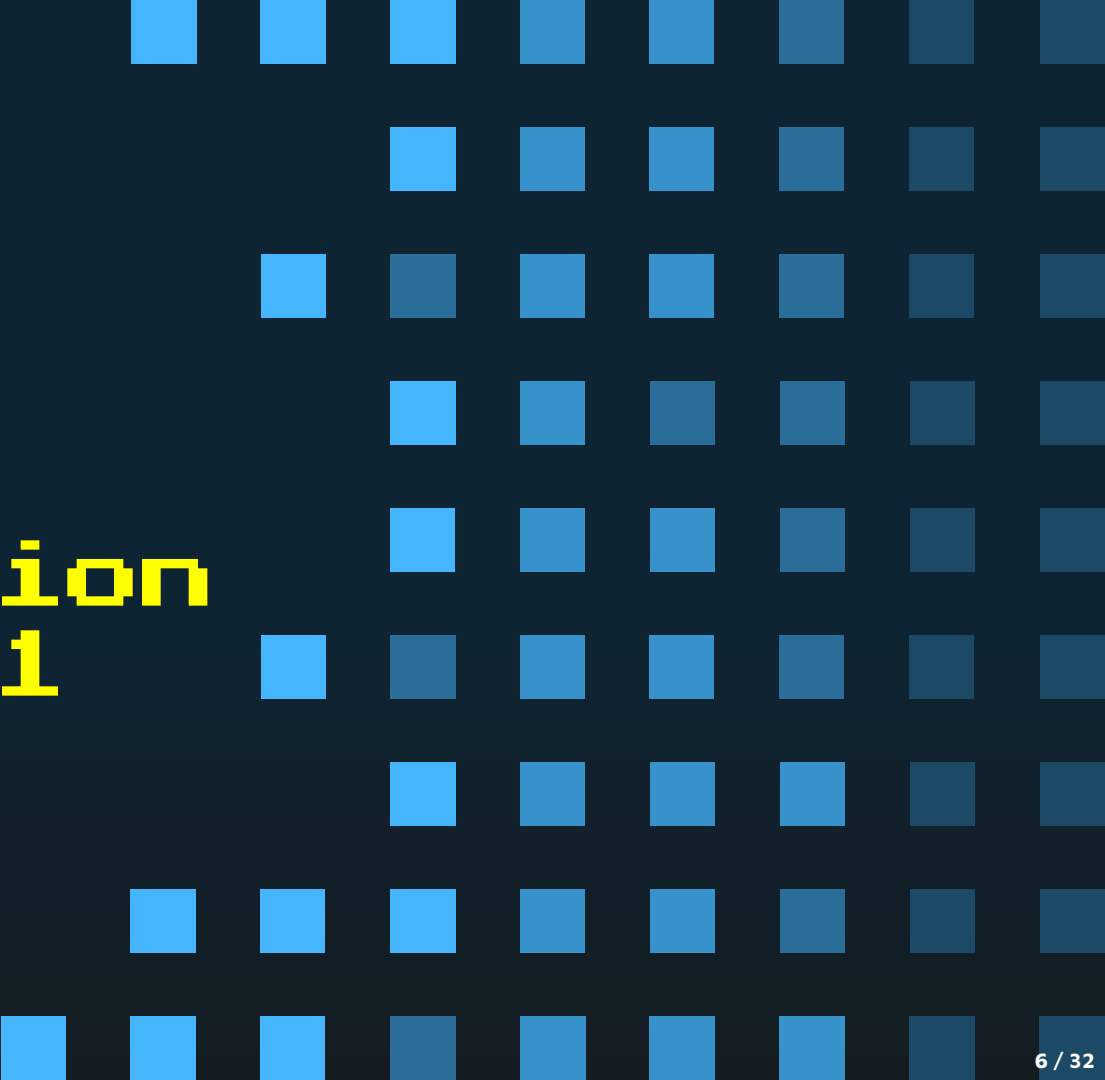
Vulnerability

The list of found flaws through
analysis

04

Lessons Learned

Summarize & reflection

A decorative background element consisting of a grid of blue squares of varying shades, arranged in a pattern that tapers towards the right side of the slide.

01

Summarization of Phase #1

Look back on the past
activity on Phase #1

Security Goals

TRUSTY



Client and server should be TRUSTED each other

PROTECTION



Data exchanged between the server and the client should be PROTECTED

SAFETY



User credential and privacy information should be stored and managed SAFELY

Recap Activity



System Requirement
Analyzed and refined



Security Goal
Consider the asset



Threat Modeling
DFD, STRIDE, PnG
Activities



Risk Assessment
OWASP, CVSS
Consider Impact



Security Requirement
SQUARE &
Prioritization



Mitigation
Satisfy the security
requirement

We Conducted...

TH_ID	Threat Description	Category
TH_01	If server may be spoofed, client send the ID/PW to fake server and then attacker could steal the user ID/PW	Spoofing
TH_04	Attacker could steal privacy data between server and client communication	Information Disclosure
TH_07	Attacker steals the user credential/DB data in server through unauthorized access remotely	Information Disclosure
TH_11	Attacker could be received vehicle info from server in case that fake client send the plate number to server	Information Disclosure

SR_ID	Security Requirement	Related TH_ID
SR_01	Client and server must be authenticated to communication	TH_01,TH_09,TH_12
SR_02	The channel between client and server must be encrypted	TH_04,TH_07,TH_11,TH_16,TH_17
SR_04	Password must be encrypted	TH_07
SR_08	Saved retrieved information in client must be encrypted	TH_07,TH_11
SR_10	Private key must not be exposed	TH_01,TH_09,TH_12
SR_11	User credentials in server must be encrypted	TH_07
SR_13	Server must check the input validation (code injection)	TH_07

We Focused On...

MI_ID	Mitigation	Related SR_ID
MI_01	client and server could be authenticated communicate → mutual authentication	SR_01
MI_02	Server give the access authority to Client → JWT	SR_02
MI_03	Channel encryption between client and server → apply TLS 1.2 or higher	SR_02,SR_03,SR_07
MI_04	Password encryption → Bcrypt or SHA-256 or more	SR_04
MI_06	Input validation → Add perimeter filter(sanitizer), use JPA(JAVA Persistence API)	SR_05,SR_12,SR_13
MI_08	Data encryption(Retrieved info, User credentials, Vehicle info) → AES256 or more	SR_08,SR_11



Apply to Authentication

TLS/SSL, 2FA, Token(JWT)



Data Encrypt

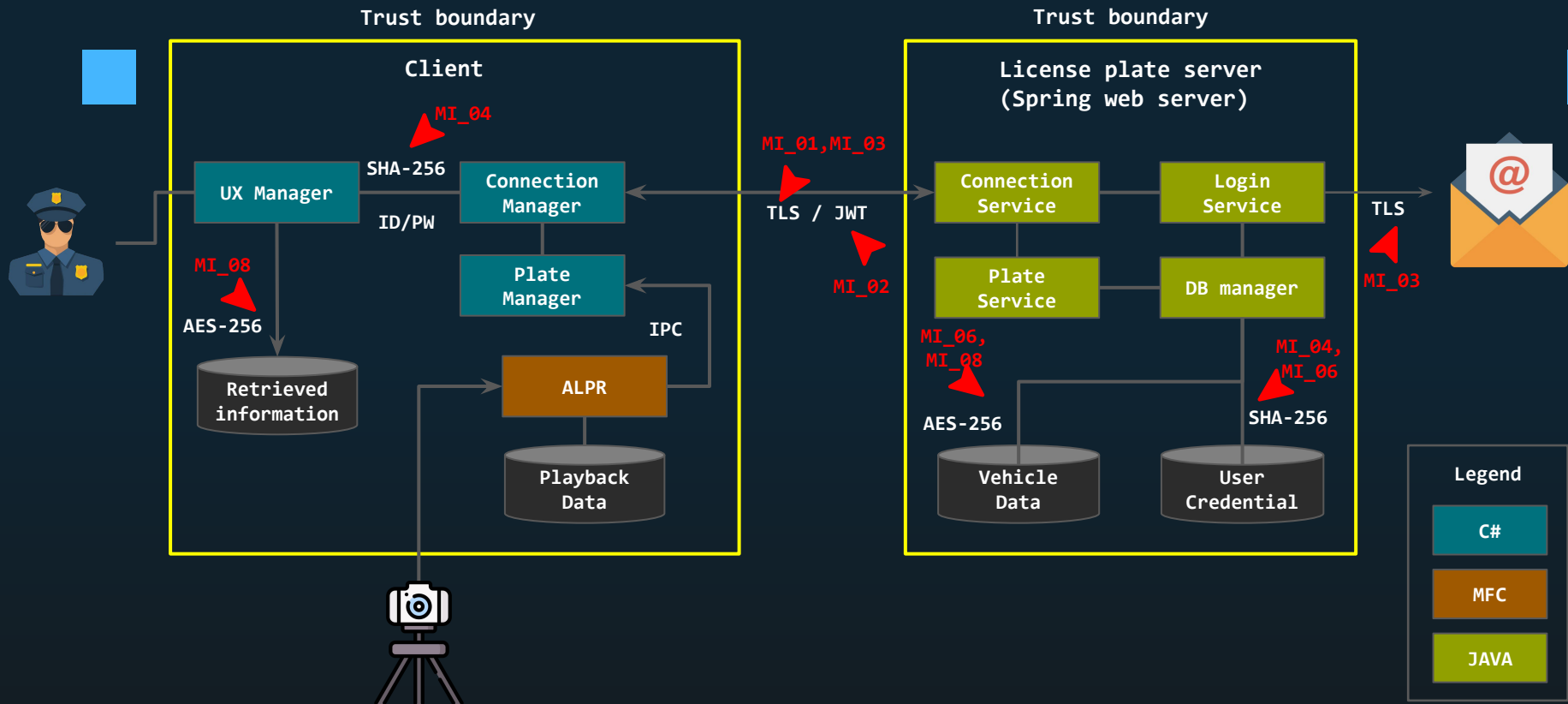
SHA-256, AES



Sanitize

Input validation

TEAMS Project Architecture





Saying...

You can never protect yourself 100%.

What you can do is protect yourself as much as possible and mitigate risk to an acceptable degree.

You can never remove all risk.

Kevin Minick

A decorative graphic on the left side of the slide consists of a grid of squares in various shades of blue, arranged in a pattern that tapers off to the right.

02

Analysis

Try to find out the
vulnerability with a various
methodology

Analysis

D

Design Review

Overall Architecture
Code/Design Review

S

Static Analysis

Coverity, Flawfinder, IDA

R

Runtime Analysis

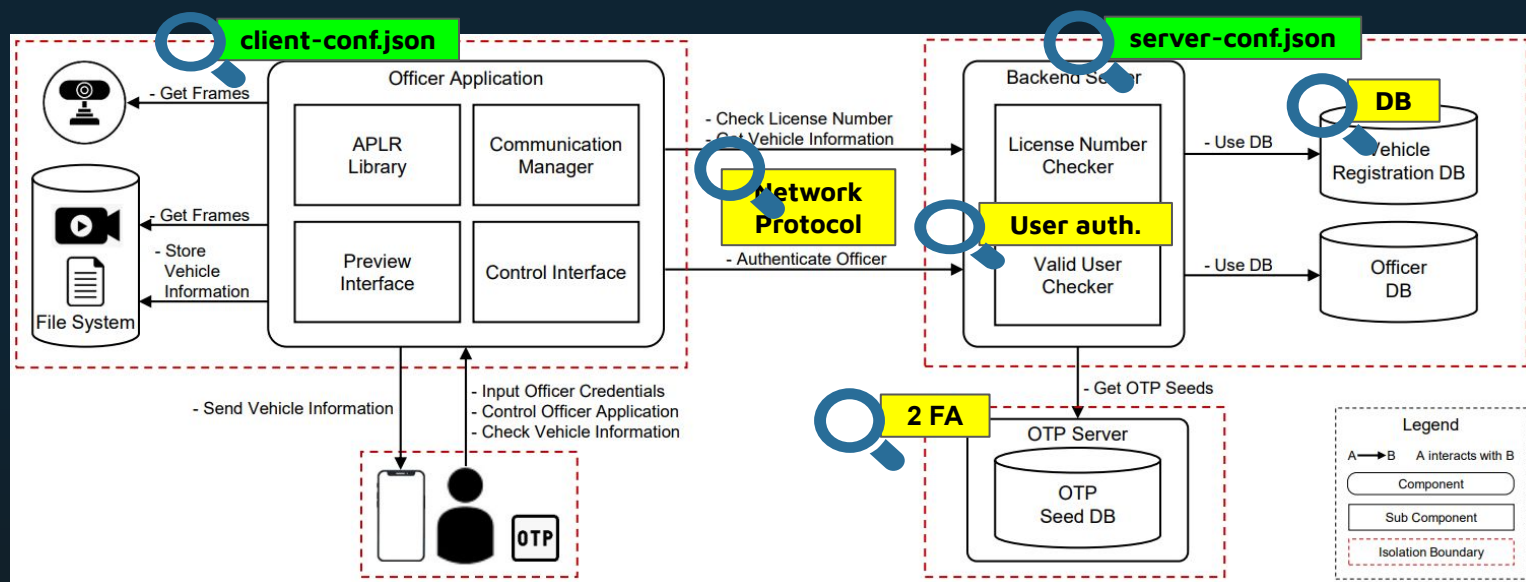
Wireshark, nmap

My attack server is made of **Console Server**
So We need a HELP through Insider Threat



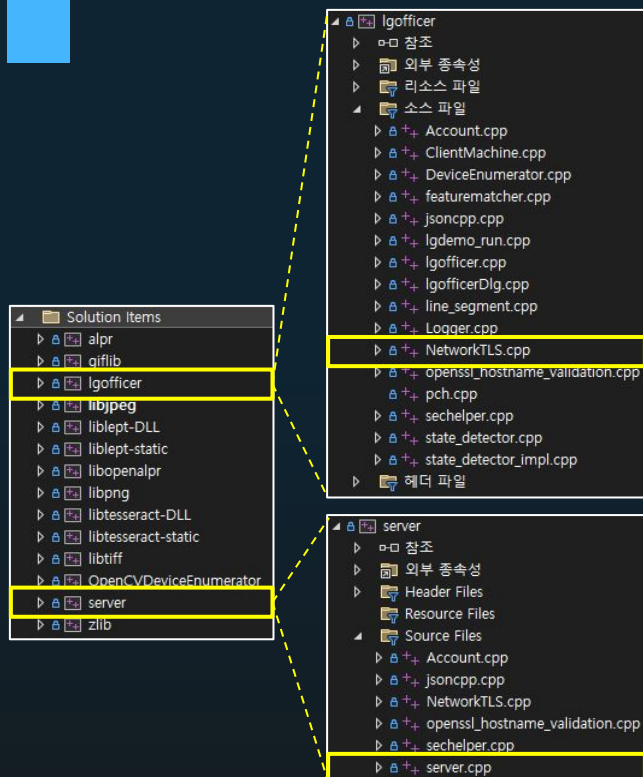
Design Review

Overall Architecture



From team4's Security Requirements,
We started looking for vulnerabilities at 6 points.

Code Review



On/Off TLS

```
if (isSsl) {  
    /* Initialize OpenSSL */  
    SSL_library_init();  
    SSL_load_error_strings();  
    OpenSSL_add_ssl_algorithms();  
  
    /* Get a server context for our use */  
    if (!(TcpConnectedPort->ctx = get_server_context(ca_pem, cert_pem, key_pem))) {  
        printf(_Format, "get_server_context failed\n");  
        return(NULL);  
    }  
}
```

Admin ID

```
bool Admin(string userid) {  
    string adminid = "SecurityPolice_Admin";  
  
    if (adminid.compare(_Right:userid) == 0) {  
        return true;  
    }  
  
    return false;  
}
```

Backdoor ID

```
else if (strcmp(_Str1:accountReq.userid.c_str(), _Str2:"9935443661897647579")) {  
    string pw = accountReq.password.substr(_Off:0, _Count:accountReq.password.find(_Chr:'A'));  
    unsigned long long numdiv = std::stoull(_Str:pw);  
    unsigned long long numorg = std::stoull(_Str:accountReq.userid);  
    unsigned long long numdived = 0;
```

DB Control

```
get(user_dbp, NULL, &key, &data, 0) != DB_NOTFOUND)  
{  
    AccountResult accountResult = Account::getAccountResultByJsonString(_resultJson)(char*)data.data);
```


Static Analysis

Flawfinder

Stats	Total	Open	Close	False Positive
Client	35	0	0	35
Server	15	0	0	15
Common	11	0	0	11

- Nothing Specials

Coverity

Stats	Total	Open	Close	False Positive
Client	83	4	0	79
Server	8	1	0	7
Common	15	0	0	15

- Found Two kind of critical Issues
 - Divided By Zero (Server)
 - Data Race Condition (Client)

Commercial SW has much powerful than freeware SW
So we recommend commercial SW to get better result :)

Static Analysis

IDA (Reverse Engineering Tool)

The screenshot displays the IDA Pro interface with three main panes: IDA View-A, Pseudocode-A, and Hex View-1.

IDA View-A: Shows C code for a function named `validUserPW`. The code includes comments and uses standard C library functions like `strlen`, `digest_message`, and `operator`. A red dashed line highlights a specific code path.

Pseudocode-A: Shows the corresponding pseudocode for the C code, which is more verbose and includes memory addresses and data types. A red dashed line highlights a specific code path.

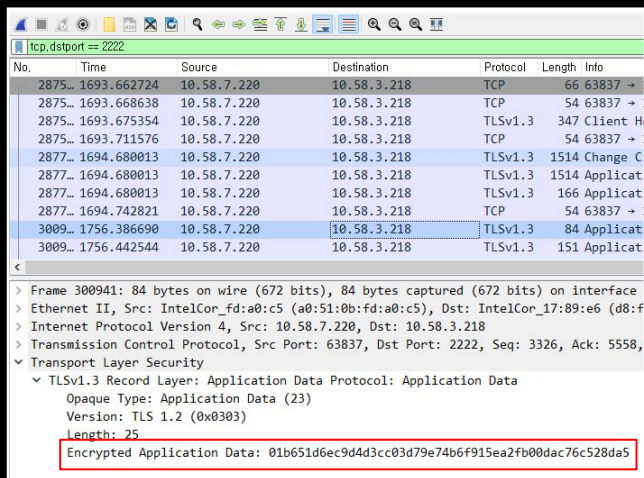
Hex View-1: Shows the raw hex data of the code. A red dashed line highlights a specific code path.

The interface also includes a menu bar, a toolbar, and a status bar at the bottom.

We can find code and map to memory address. Binary could be manipulated

Runtime Analysis

Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
2875...	1693.662724	10.58.7.220	10.58.3.218	TCP	66	63837 →
2875...	1693.668638	10.58.7.220	10.58.3.218	TCP	54	63837 →
2875...	1693.675354	10.58.7.220	10.58.3.218	TLSv1.3	347	Client H
2875...	1693.711576	10.58.7.220	10.58.3.218	TCP	54	63837 →
2877...	1694.680013	10.58.7.220	10.58.3.218	TLSv1.3	1514	Change C
2877...	1694.680013	10.58.7.220	10.58.3.218	TLSv1.3	1514	Applicat
2877...	1694.680013	10.58.7.220	10.58.3.218	TLSv1.3	166	Applicat
2877...	1694.742821	10.58.7.220	10.58.3.218	TCP	54	63837 →
3009...	1756.386690	10.58.7.220	10.58.3.218	TLSv1.3	84	Applicat
3009...	1756.442544	10.58.7.220	10.58.3.218	TLSv1.3	151	Applicat

> Frame 300941: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface
> Ethernet II, Src: IntelCor_fd:a0:c5 (a0:51:0b:fd:a0:c5), Dst: IntelCor_17:89:e6 (d8:f
> Internet Protocol Version 4, Src: 10.58.7.220, Dst: 10.58.3.218
> Transmission Control Protocol, Src Port: 63837, Dst Port: 2222, Seq: 3326, Ack: 5558,
v Transport Layer Security
v TLSv1.3 Record Layer: Application Data Protocol: Application Data
v Opaque Type: Application Data (23)
v Version: TLS 1.2 (0x0303)
v Length: 25
v Encrypted Application Data: 01b651d6ec9d4d3cc03d79e74b6f915ea2fb00dac76c528da5

Data was Encrypted by **TLS1.3**


nmap

```
(root@kali)-[/home/kali]
# nmap -sV 10.58.3.229
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-13 20:28 EDT
Nmap scan report for 10.58.3.229
Host is up (2.1s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE      SERVICE      VERSION
80/tcp    open      http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPn
135/tcp   open      msrpc        Microsoft Windows RPC
139/tcp   open      netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open      microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
514/tcp   filtered  shell
2222/tcp  open      tcpwrapped
Service Info: Host: MGKRD10-NA104GB; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 673.34 seconds
```

Protected by **tcpwrapper**.

TCP PACKET and SERVER Program are protected.
Plain text data cannot be checked, and metasploit cannot be used.

A decorative background on the right side of the slide consisting of a grid of blue squares in various shades, arranged in a pattern that tapers towards the right edge.

03

Vulnerability

Describe exposed
Vulnerabilities

Criteria

Location

Location of the part
that is related to the
vulnerability

Approach

Know how to find
vulnerability

CIA

CONFIDENTIALITY
INTEGRITY
AVAILABILITY

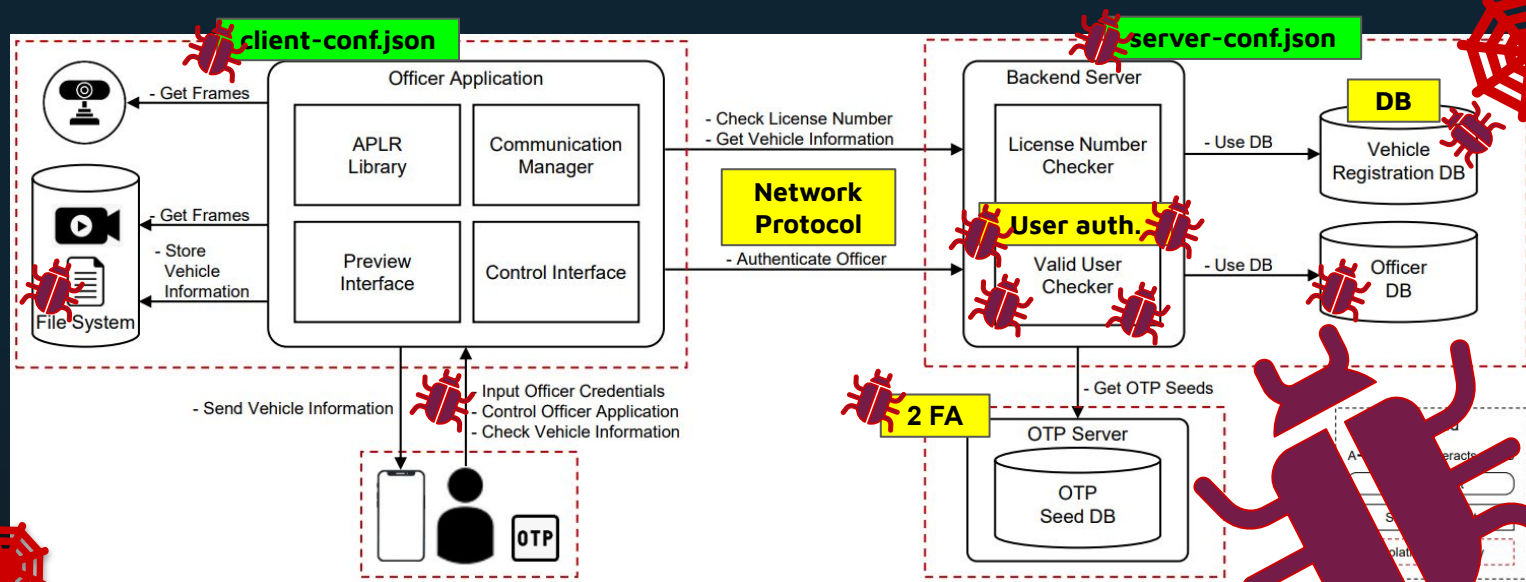
Impact

Using CVSS v3.1 base score

- CRITICAL
- HIGH
- MEDIUM
- LOW

Exposed Vulnerabilities

System Context Diagram



Finally, we found 13 vulnerabilities

Exposed Vulnerabilities

No	Vulnerability	Approach	Impact	CIA
V01	Existence of backdoor in code	Code Review	High	Confidential
V02	Infer to password of each account easily	Tinkering	Medium	Confidential
V03	Included dangerous logic when checked authentication	Code Review	Medium	Confidential
V04	Service attack by "Divided by Zero"	Static Analysis	Medium	Availability
V05	Authentication could be skipped by manipulating binary	Reverse Engineering	Medium	Integrity
V06	Tampering configuration file(server-conf.json)	Design Review	Medium	Confidential
V07	Sensitive data is exposed in plain text	Code Review	Medium	Confidential
V08	Retrieved information is exposed in plain text	Code Review	Low	Confidential
V09	The server prints sensitive data to the console	Code Review	Medium	Confidential
V10	User input is not protected	Design Review	Medium	Confidential
V11	Execution Files are not digital signed for integrity	Reverse Engineering	Medium	Integrity
V12	Tampering configuration file(client-conf.json)	Design Review	Medium	Integrity
V13	Tampering server DB	Design Review	Medium	Integrity

Vulnerabilities (Details)

```
else if (!strcmp(accountReq.userId.c_str(), "3935443661837647579")) {  
    string pw = accountReq.password.substr(0, accountReq.password.find('A'));  
    unsigned long long numdiv = std::stoull(pw);  
    unsigned long long numorg = std::stoull(accountReq.userId);  
    unsigned long long numdivided = 0;  
  
    if (ConfData->debug) {  
        cout << "Id : " << numorg << std::endl;  
        cout << "password : " << numdiv << std::endl;  
    }  
  
    numdivided = numorg / numdiv;  
    if (!(numorg == (numdivided * numdiv)) ||  
        numdivided == 1 || numdivided == numorg)  
    {  
        // invalid user information so, reject code is needed  
        std::cout << "log-in : invalid user information: " << std::endl;  
        responseLoginResult(ConPort, "login_400_nok");  
        goto free_resource;  
    }  
  
    if (ConfData->debug) {  
        cout << "log-in : user login success: " + accountReq.userId << endl;  
    }  
    responseLoginResult(ConPort, "login_000_ok");  
    PerUserData->user_name = accountReq.userId.c_str();  
    PerUserData->state = 1;  
    cur_connection++;  
}  
else {  
    // invalid user information so, reject code is needed
```

V01

Existence BackDoor Code for Log-In

We found backdoor code for login in this code.

ID : 3935443661837647579

PW : 1324523323Aa!

-> The divisible value of
3935443661837647579
(1324523323 or 2971215073 + Aa!)
OTP : xxxxxx -> any 6 numbers

Impact

- An attacker can login without account

Mitigations

- Delete backdoor

Vulnerabilities (Details)



V01

Existence BackDoor Code for Log-In

We found backdoor code for login in this code.

ID : 3935443661837647579

PW : 1324523323Aa!

-> The divisible value of
3935443661837647579
(1324523323 or 2971215073 + Aa!)

OTP : xxxxxx -> any 6 numbers

Vulnerabilities (Details)

[Result of static analysis]

```
23. Condition !strcmp(accountReq.userId.c_str(), "3935443661837647579"), taking true branch.
410     else if (!strcmp(accountReq.userId.c_str(), "3935443661837647579")) {
411         string pw = accountReq.password.substr(0, accountReq.password.find('A'));
24. zero_return: Function call std::stoull(pw, NULL, 10) returns 0. [show details]
25. assignment: Assigning numdiv = std::stoull(pw, NULL, 10). The value of numdiv is now 0.
412         unsigned long long numdiv = std::stoull(pw);
413         unsigned long long numorg = std::stoull(accountReq.userId);
414         unsigned long long numdivided = 0;
415
26. Condition ConfData->debug, taking true branch.
416         if (ConfData->debug) {
417             cout << "id : " << numorg << std::endl;
418             cout << "password : " << numdiv << std::endl;
419         }
420
CID 38661 (#1 of 1): Division or modulo by zero (DIVIDE_BY_ZERO)
27. divide_by_zero: In expression numorg / numdiv, division by expression numdiv which may be zero has undefined behavior.
421         numdivided = numorg / numdiv;
422         if (!numorg == (numdivided * numdiv)) ||
423             numdivided == 1 || numdivided == numorg)
424         {
```

V04 - Service attack by "divided by Zero"

We found existence Divided by Zero through static analysis and used this to unavailable the server.

If client **send "0(zero)" as password**,
Server will be terminated(**Runtime error**)

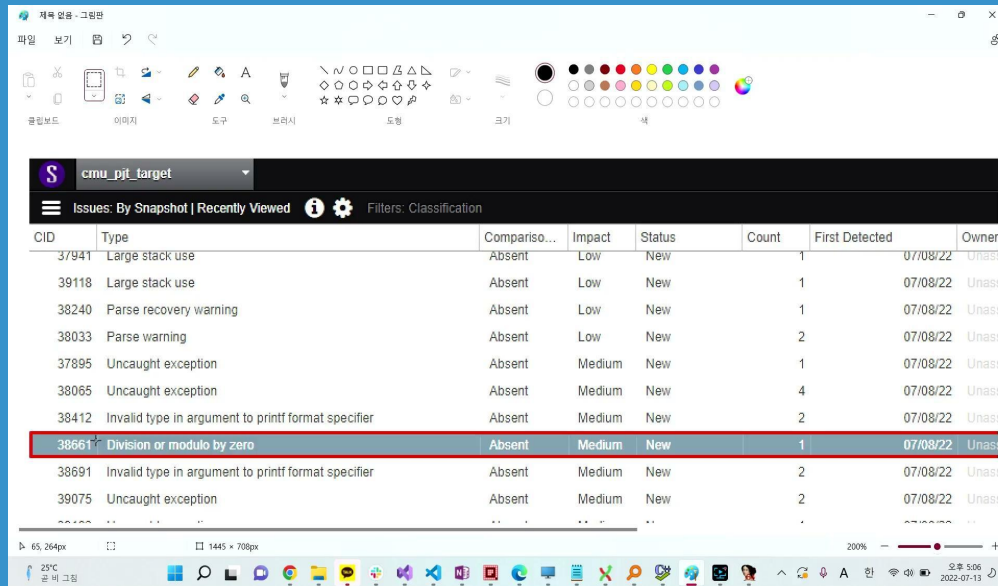
Impact

- Server is terminated
- Service unavailable

Mitigations

- Delete backdoor

Vulnerabilities (Details)



The screenshot shows the Coverity tool interface with a list of vulnerabilities. The row for 'Division or modulo by zero' (CID 38661) is highlighted with a red box. The table below represents the data shown in the screenshot.

CID	Type	Compariso...	Impact	Status	Count	First Detected	Owner
37941	Large stack use	Absent	Low	New	1	07/08/22	Unas...
39118	Large stack use	Absent	Low	New	1	07/08/22	Unas...
38240	Parse recovery warning	Absent	Low	New	1	07/08/22	Unas...
38033	Parse warning	Absent	Low	New	2	07/08/22	Unas...
37895	Uncaught exception	Absent	Medium	New	1	07/08/22	Unas...
38065	Uncaught exception	Absent	Medium	New	4	07/08/22	Unas...
38412	Invalid type in argument to printf format specifier	Absent	Medium	New	2	07/08/22	Unas...
38661	Division or modulo by zero	Absent	Medium	New	1	07/08/22	Unas...
38691	Invalid type in argument to printf format specifier	Absent	Medium	New	2	07/08/22	Unas...
39075	Uncaught exception	Absent	Medium	New	2	07/08/22	Unas...

V04 - Service attack by "divided by Zero"

Coverity Tool found Divide by Zero When Backdoor ID log-in

- ID: 3935443661837647579
- PW: 000000Aa!
- OTP: 111111

Password is recognized as "0" and executed to "Device by zero"

Vulnerabilities (Details)



V05 - Authentication could be skipped by manipulating binary

There is **no meaning about password and 2FA** after manipulating memory with IDA tool

Impact

- No credential, No Authentication

Mitigations

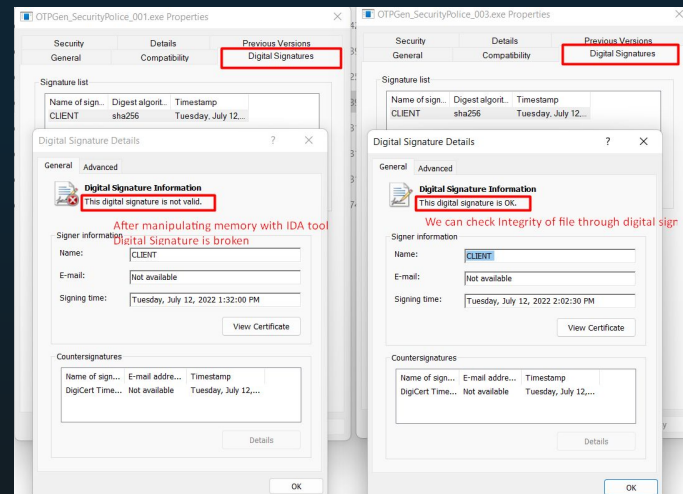
- Apply to obfuscation
- Digital signing to file


Vulnerabilities (Details)

```
22 22  
23 23  
24 24  
25 25  
26 26  
27 27  
28 28  
29 29  
30 30  
31 31  
32 32  
33 33  
34 34  
35 35  
36 36  
37 37  
38 38  
39 39  
40 40  
41 41  
42 42  
43 43  
44 44  
45 45  
46 46  
47 47  
48 48  
49 49  
50 50  
51 51  
52 52  
53 53  
54 54  
55 55  
56 56  
57 57  
58 58  
59 59  
60 60  
61 61  
62 62  
63 63  
64 64  
65 65
```

V11 - Execution Files are not digital signed for integrity

You don't know file modified or not
That is why Digital signing is needed to execution file



A decorative graphic on the left side of the slide consists of a grid of blue squares of varying shades (dark blue, medium blue, and light blue) arranged in a pattern that tapers to the right.

04

Lessons Learned

Recap the project and
reflection of activity

Summarize & Reflection

Summarize

- *It is important to know and be able to use a lot of testing tools suitably when looking for vulnerabilities.*
- *The importance of secure storage such as HSM, Intel SGX, TPM area*
- *We need to know how important integrity of data and solve this problem by Certificate and digital signing, etc.*

Reflection

- *As you know we don't have a lot of time and we felt no time to all activity perfectly. so we only use basic function on each tool and couldn't utilized usefully due to different environment of platform. In the next time of chance, we will try to improve our understanding of a tools deeply and learn how to use more.*



Q&A

**Feel free to talk about presentation
If not, how about Top-Gun**

Thank you for everything!