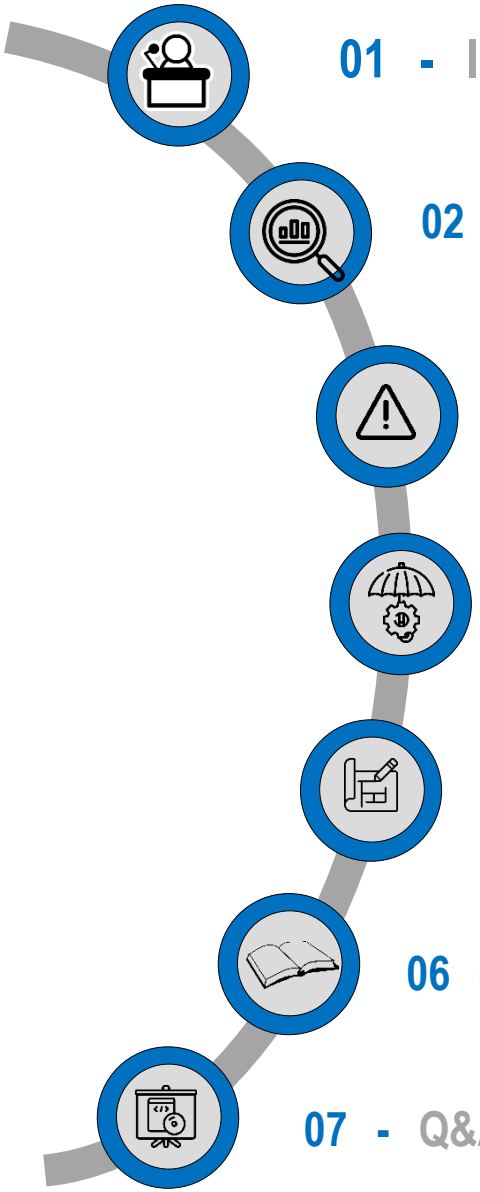

Phase 2
LGE Security Specialist 2022
Team 4 (S4Best)
15th July, 2022

Agenda



01 - Introduction

02 - Plans

03 - Summary phase1
- Retrospective
- Vulnerabilities

04 - Target of Evaluation
- Goals of phase2
- Targeted designs
- System understanding

05 - Vulnerability Analysis
- Vulnerability assessment strategy
- Reconnaissance
- Static analysis
- Fuzzing
- Design review
- Major results - Vulnerability list
- Recommended mitigations
- Representative Vulnerability

06 - Lessons Learned

07 - Q&A

1. Introduction

- Members & Roles

Members & Roles

- **Team 4 intro**

  Team name — S4Best (Security For Best)

  Motto — Do Best

- **Who we are…**

JG	(Jaegeun Lee)	{ Core engineering, Client }	{ Exploit Vulnerability }
SE	(Seongeun Kim)	{ System Architecture, Server side }	{ Design Review }
JH	(Jeongho Choi)	{ Authentication, Threat analysis }	{ Risk Assessment }
NS	(Namseok Kim)	{ DevOps, Team Leading }	{ Static Analysis }
CR	(Choongrae Kim)	{ Secure Architecture/Communication }	{ Design Review }
JW	(Jiwoong Eom)	{ Authentication, Risk analysis }	{ Exploit Vulnerability }

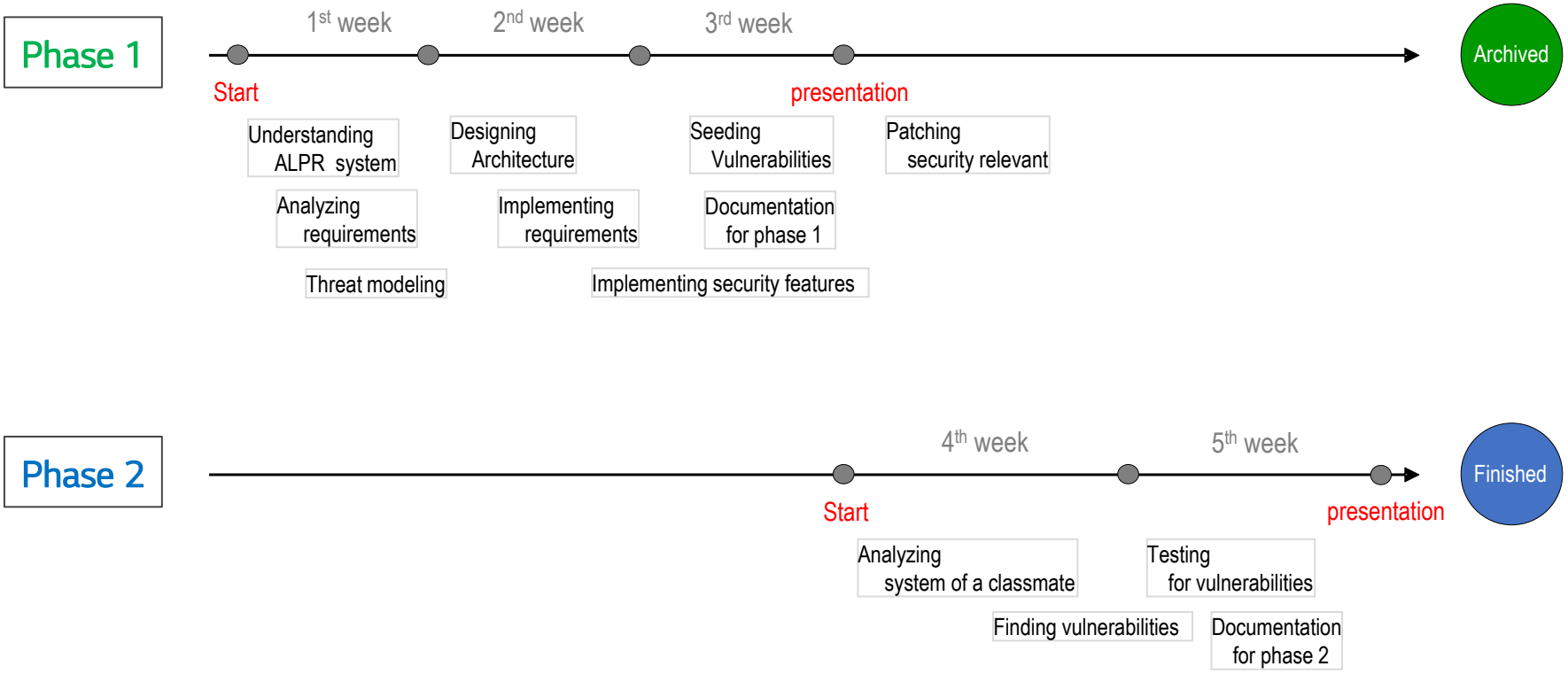
- **Special thanks to**

Clifford Huff as a mentor
Jeffrey Gennari as a professor
Nancy Mead as a professor
Myongsook Jin as a manager
Jihye Yoo as a manager
and
All of the participants for 2022 LGE security specialist

2. Plans

- Plans & Schedule performance

❑ Plans and schedule performance



3. Summarization of phase 1

- Retrospective
- others

❑ Primary requirements

1. Strengthen account policy
2. Secure communication
3. Input validation

REQ_CLI_01	Two Factor Authentication

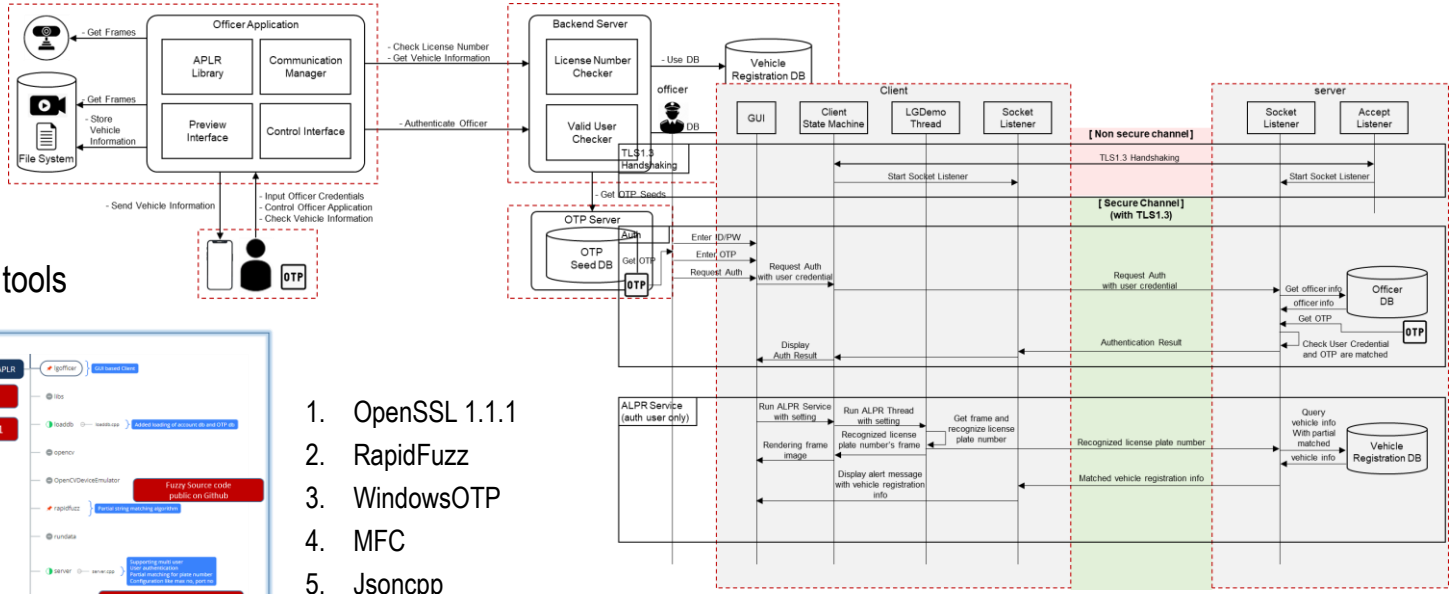
→ Officers can access highly privacy-sensitive information, such as criminal information.
Therefore, to prevent spoofing of officers, it is necessary to take a strengthened account policy.

REQ_CLI_05	Secure Communication between client and server

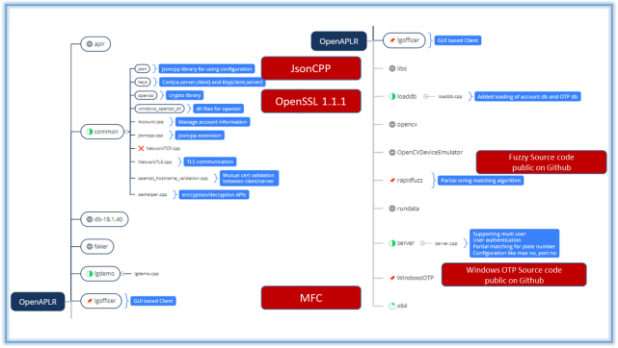
→ Spoofing/Sniffing of Server/Client are possible at the same time.
This can be protected through mutual authentication and message encryption between client and server.

❑ System Design and working

✓ Design, System working



✓ Libraries, frameworks and tools



1. OpenSSL 1.1.1
2. RapidFuzz
3. WindowsOTP
4. MFC
5. Jsoncpp

Vulnerabilities #1

❑ Threats

✓ The relative priority of each threat

TID-C5	User Spoofing - Weak passwords, password exposure

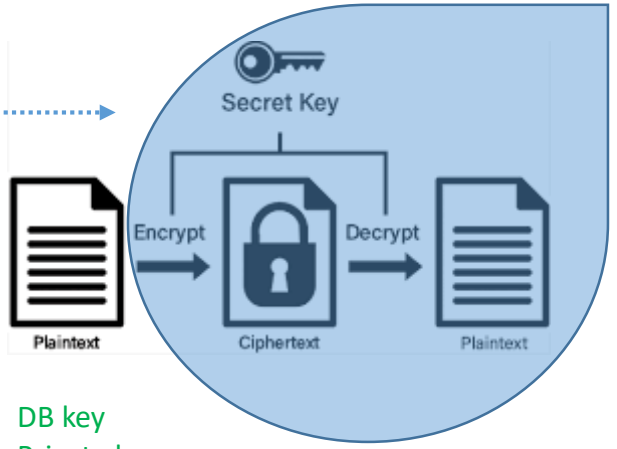
→ Officers can access highly privacy-sensitive information, such as criminal information.

TID-N7	Information Disclosure - Client-Server User ID/PW

→ Officer's ID/PW can be sniffed by monitoring network communication.

❑ Zero-day Vulnerabilities

● Not applied cryptography



➡ Patched!

- DB key
- Private key
- Alert data

Vulnerabilities #2

Seeded Vulnerabilities

✓10 vulnerabilities that we injected ➡ Refer to Team4_vulnerabilities.pdf for details

VID	STRIDE	Description	CVSS Score
VU1	D	Null pointer dereference could have occurred when an alert message is written in a log file after service is stopped	Overall CVSS Score: 3.3 Severity : Low
VU2	E	Backdoor - ID/PW based on factorization of big number	Overall CVSS Score: 7.3 Severity : High
VU3	T	Tampering configuration file results in TCP connection (not TLS)	Overall CVSS Score: 6.1 Severity : Medium
VU4	I	Sniffing ID/Password/OTP/Plate Information	Overall CVSS Score: 6.1 Severity : Medium
VU5	S	Problem of using the same TLS key: Self-signed certificate	Overall CVSS Score: 7.1 Severity : High
VU6	S	Buffer overflow of Alert.log data	Overall CVSS Score: 3.2 Severity : Low
VU7	I	Display id and password in the log of server-side	Overall CVSS Score: 6.3 Severity : Medium
VU9	E	Weak point of protocol about packet command processing	Overall CVSS Score: 7.5 Severity : High
VU10	T	Tampering with the file that is used for storing alert information (TOCTOU)	Overall CVSS Score: 7.6 Severity : High

Vulnerabilities #3

❑ Seeded Vulnerabilities

✓ Common items among all vulnerabilities

➡ Refer to Team4_vulnerabilities.pdf

VID	STRIDE	Description	Source Code	PoC	CVSS Score	Consequences	Impact
VU2	E	Backdoor - ID/PW based on factorization of big number - Backdoor for Developer ID = (Prime A) * (Prime B) PW = Prime A or Prime B ID/PW Authentication : if (ID / PW) == (Integer), then Authenticate For the number to be used as the ID, choose a number large enough to take a week or two with the current NotePC computing power	server.cpp line : 411~440	1. Attacker can check the ID for backdoor by analyzing shared codes. 2. Attacker can check the procedure of backdoor authentication by using shared codes. 3. Attacker gets the password by factorization of ID number 4. Attacker can infer the order of two prime numbers by considering the order of the ID number, and proper prime numbers can be selected sequentially by using - Brute-Force - A list of known prime numbers (http://compoasso.free.fr/primelistweb/page/prime/liste_online_en.php)	*CVSS Base Score: 5.9* *Impact Subscore: 3.6* *Exploitability Subscore: 2.2* *CVSS Temporal Score: 5.6* *CVSS Environmental Score: 7.3* *Modified Impact Subscore: 5.4* *Overall CVSS Score: 7.3* *Severity : High*	An attacker without access rights can gain access to the system
VU3	T	Tampering configuration file results in TCP connection (not TLS) - tcp connection is conducted through OpenTcpConnection function which we implemented - this function has argument const char* ca_pem, const char* cert_pem, const char* key_pem from configure file but when fail to read CA path from configure file, it returns NULL - when NULL is passed, it tries to TCP connection rather than TLS - it results in non-secure channel establishment	NetworkTLS.cpp line :261~263 line : 542~708 line : 745~758 line : 780~828 line : 851~860	1. restart the server by deleting the "server_cert_path", "server_key_path", "ca_cert_path" value to the server's config file 2. On the server side, it changes from tls socket communication to unprotected socket communication.	*CVSS Base Score: 5.7* *Impact Subscore: 5.2* *Exploitability Subscore: 0.5* *CVSS Temporal Score: 5.4* *CVSS Environmental Score: 6.1* *Modified Impact Subscore: 5.9* *Overall CVSS Score: 6.1* *Severity : Medium*	Mutual authentication based on PKI certificates is not supported.

4. Target of Evaluation

- Goals of phase2
- Targeted designs
- System understanding for security aspect

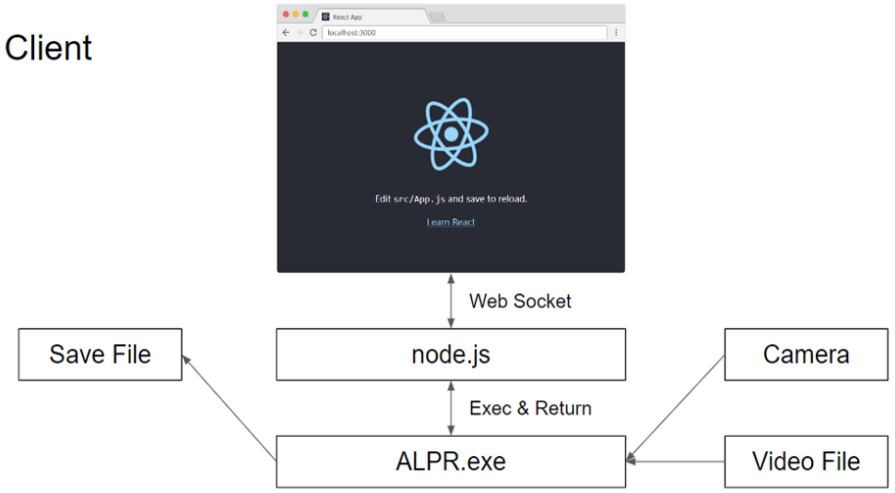
Goals of phase2

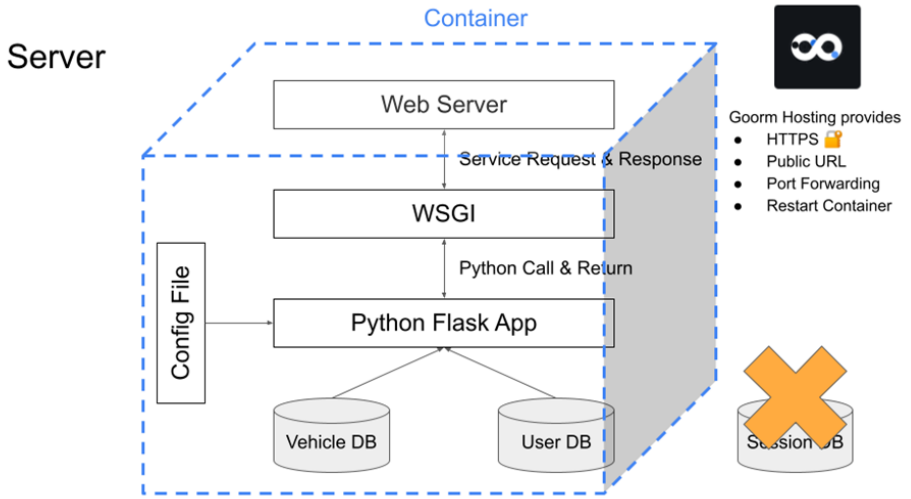
Security Goals for Team 3

- G-01 System should maintain confidentiality and integrity of databases .
- G-02 The confidentiality of the network communication should be maintained
- G-03 System needs to detect a breach of personal identifiable information of databases

Breaking **C**onfidentiality	Breaking **I**ntegrity	Breaking **A**vailibity
Steal the credentials and Plate information	**Pollute user DB**	**Make the system unavailable**

❑ Architecture Design

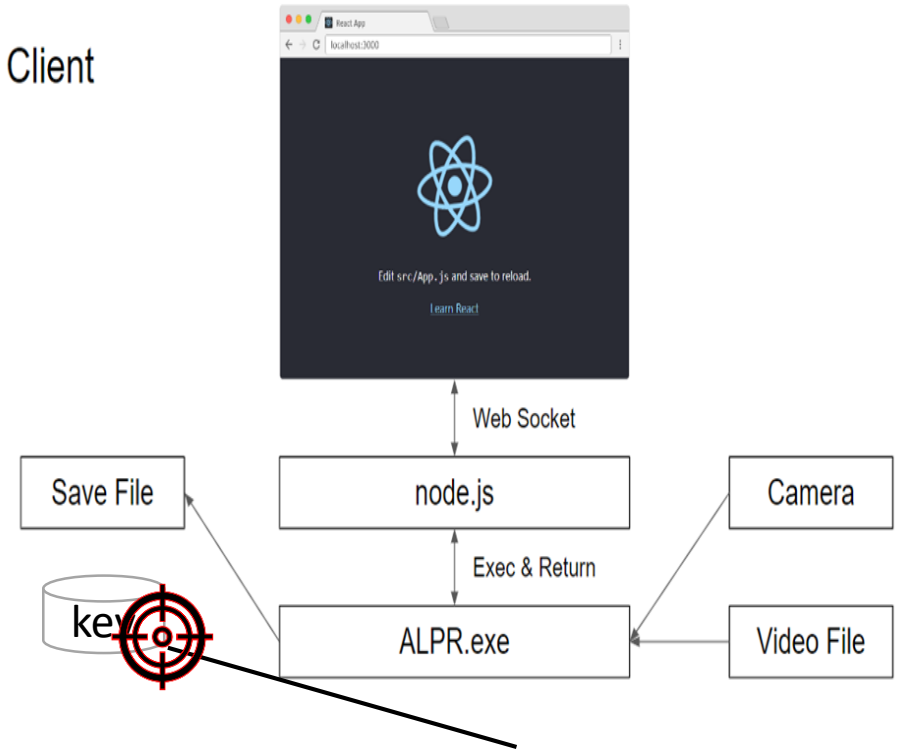


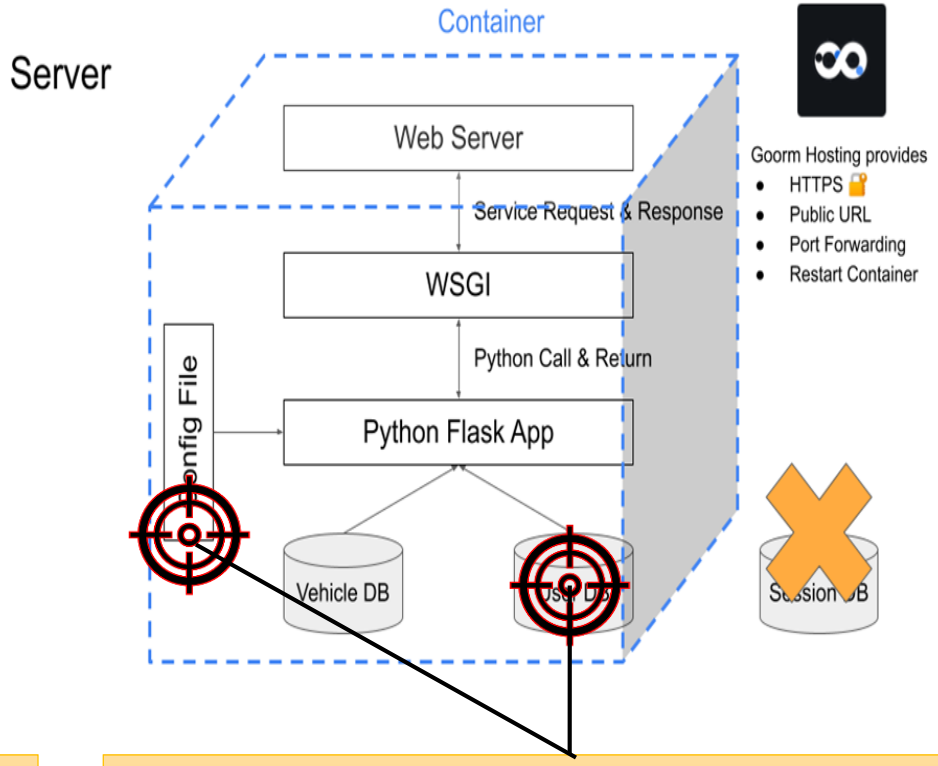


❑ First impressions

1. Wow!!! Why haven't we considered this approach?
2. Is it possible to attack?
3. It is going to be really hard to attack this system (Hosting Server, Container, TLS and etc.)

❑ Sensitive data protection

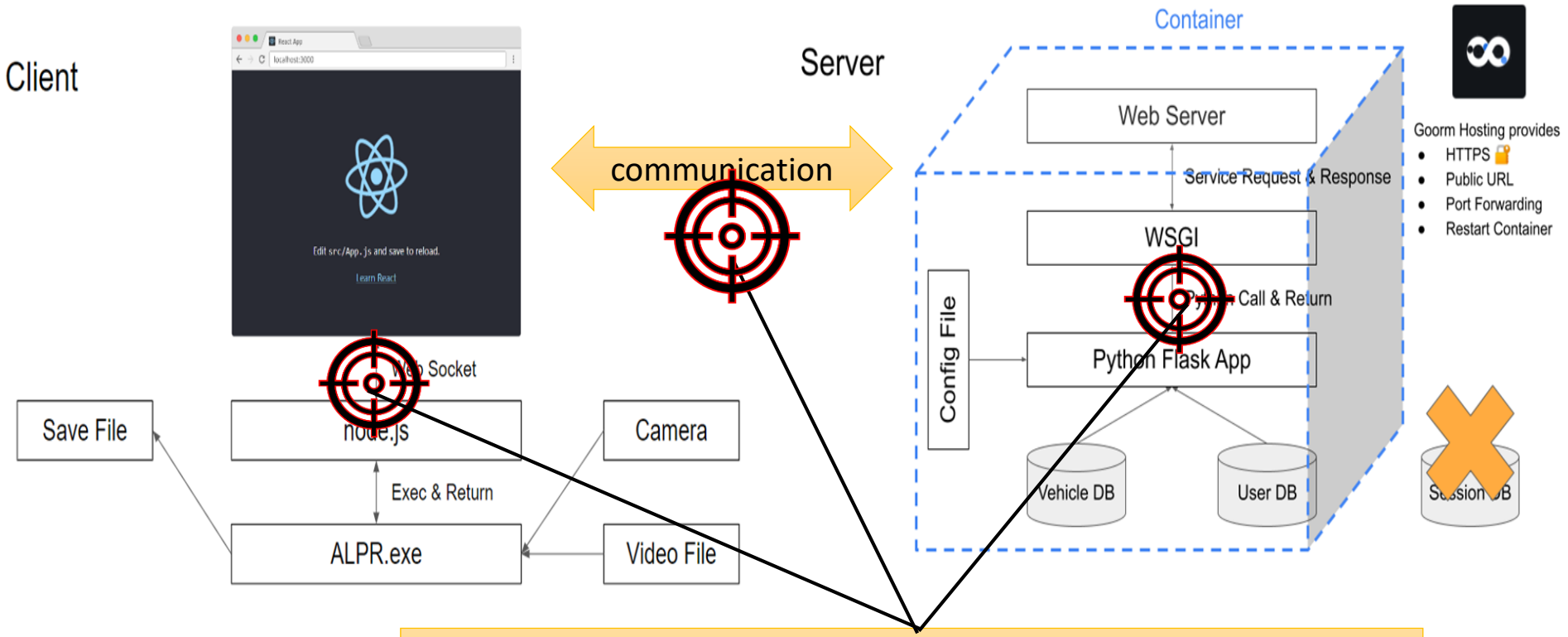




- No any mechanism to protect sensitive data such as private key and secure token
- Not Support to store alert information locally

- Support db column encryption for sensitive data
- Plaintext password is used for validation
- Config file is protected by container

❑ Secure Communication



1. Hosting Server provides both HTTPS and HTTP protocol
2. Do not support mutual authentication
3. Not support secure communication for internal commu nication both client and server

5. Vulnerabilities Analysis

- Vulnerability assessment strategy
- Reconnaissance
- Static analysis
- Fuzzing
- Design review
- Major results - Vulnerability list
- Recommended mitigations
- Representative Vulnerability

Vulnerability assessment strategy

- **Violation of Security Property – CIA**

  [CONFIDENTIALITY] - compromises confidentiality
  [INTEGRITY] - compromises integrity
  [AVAILABILITY] - compromises availability

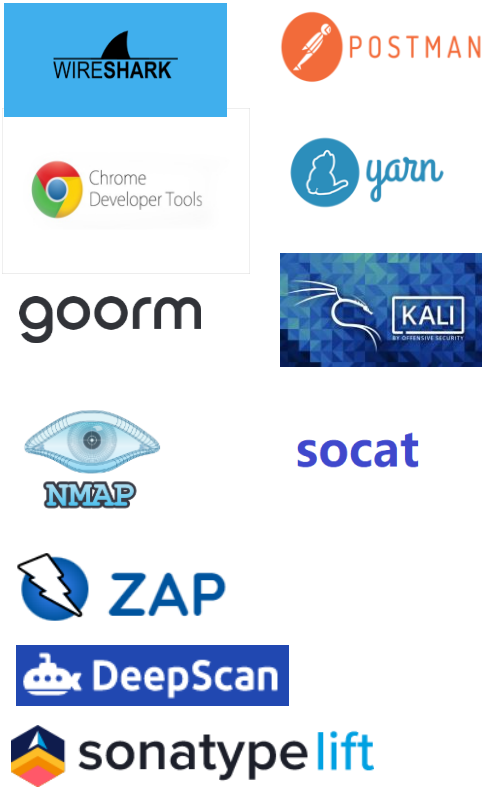
- **Scanning Approach**

  [STATIC] - done by running static analysis tools
  [FUZZING] - done by running fuzzing tools
  [REVIEW/DESIGN] - done by reviewing documents or code
  [REVIEW/CODE] - done by reviewing code itself

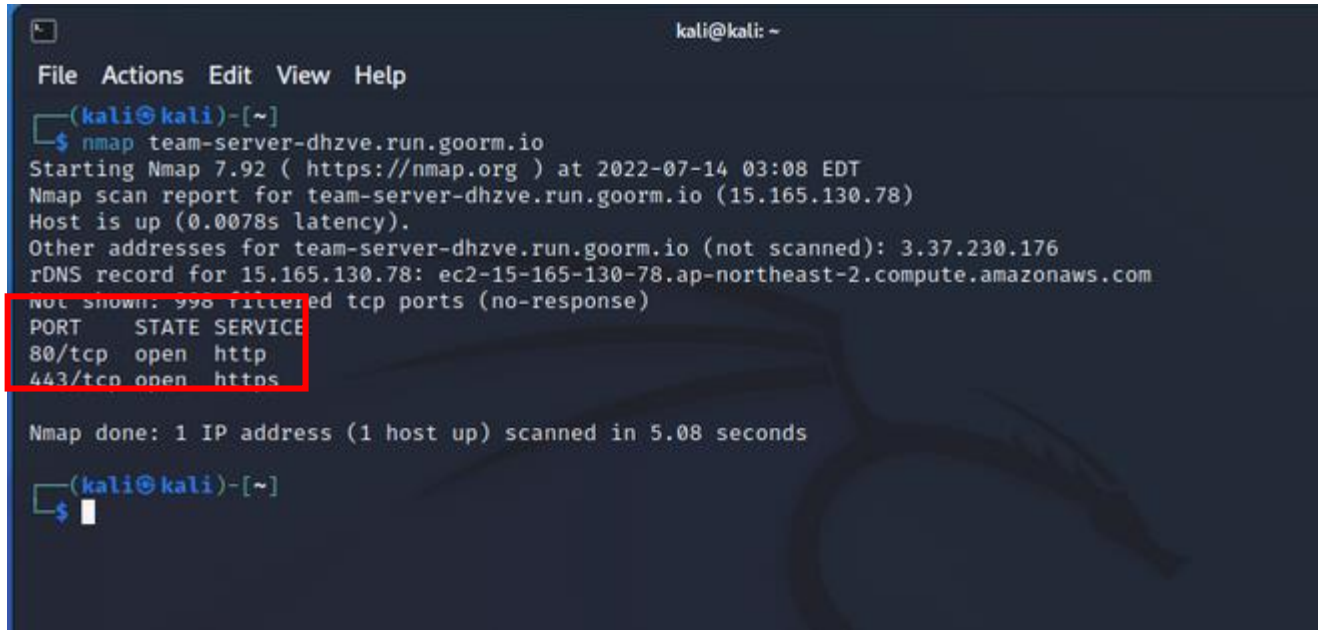
- **Technique (Exploit)**

  [SNIFFING] - sniffing packets over the network
  [SPOOFING] - so-called, man in the middle attack
  [BRUTEFORCE] - trying all possible input until success
  [CRAFTPACKET] - crafting and sending a customized packet
  [TAMPERING] - modifying system components for a purpose
  [WEBDEBBER] - sniffing credentials using web debugger.
  [NOSPECIFIED] - no special techniques specified

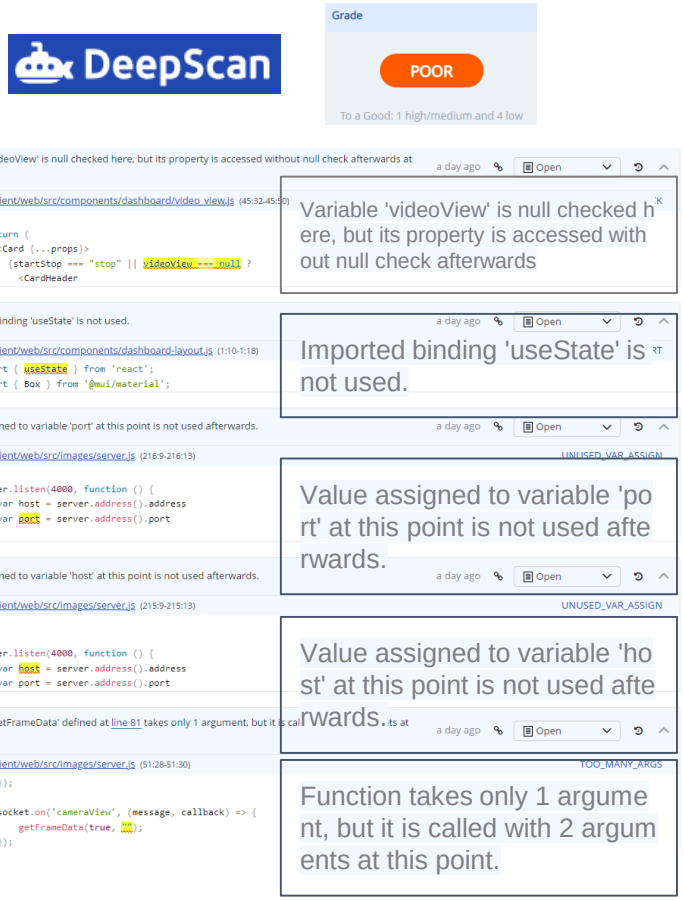
- **Tools**



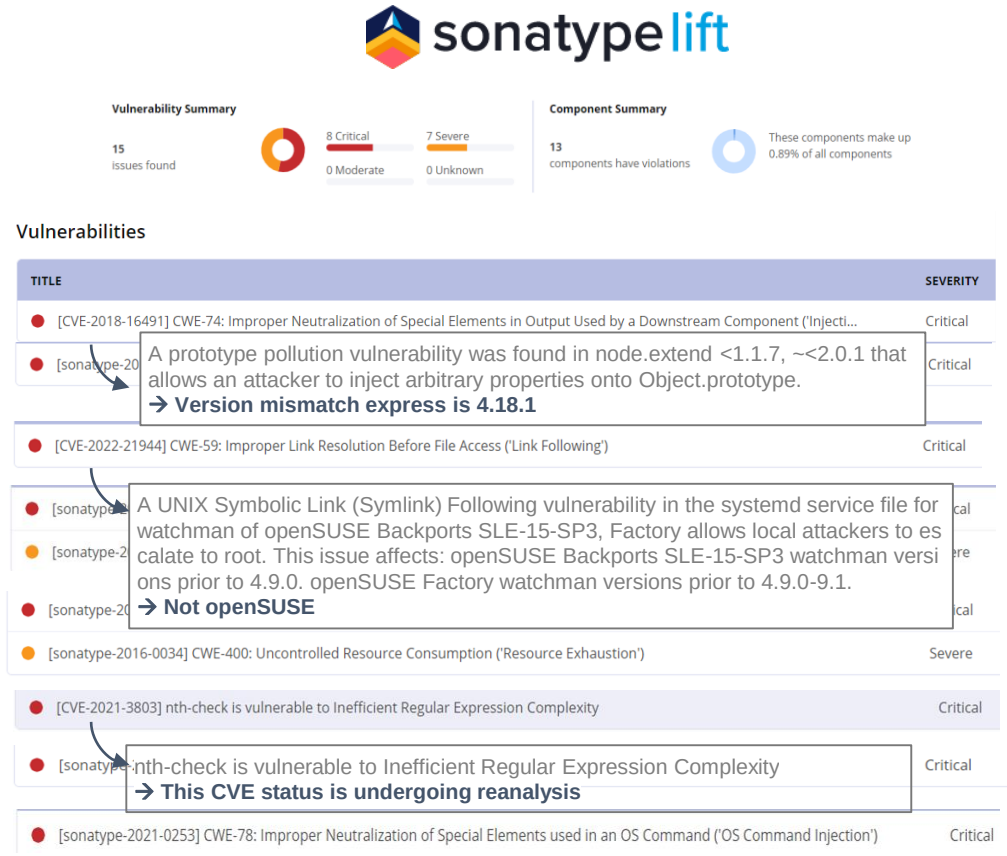
- Port scan with nmap on kali
  - target : team-server-dhzve.run.goorm.io (remote server)
  - there are two ports 80(http), 443(https)



- 2 different tools that we used (each tools have several lint engines like jslint)



✓ 6 issues, Poor grade

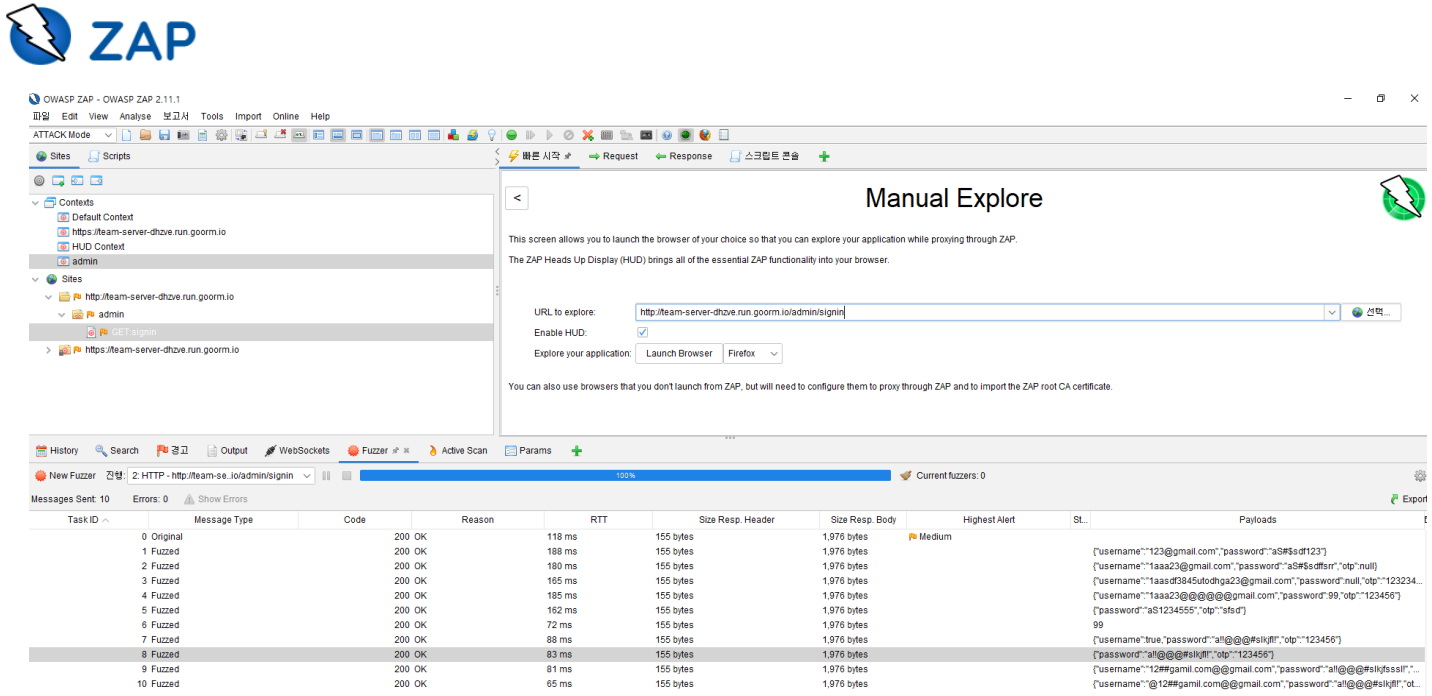


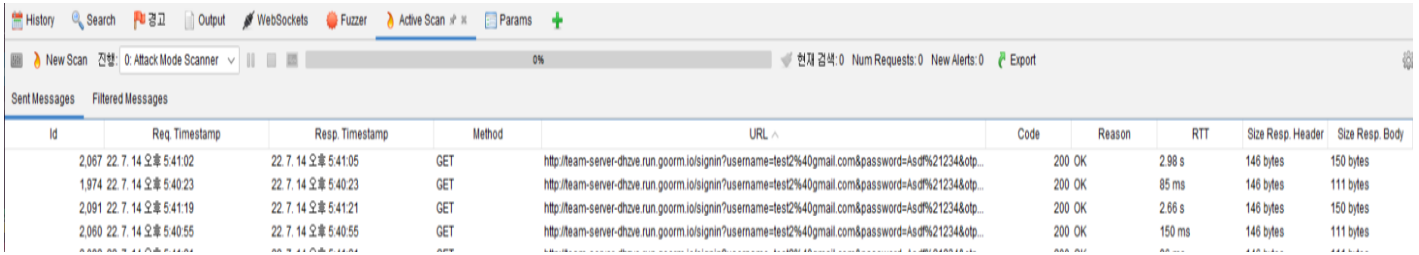
✓ 15 issues (8 critical / 7 severe)

✓ 13 components Violation

We could not find any critical issue in static analysis!

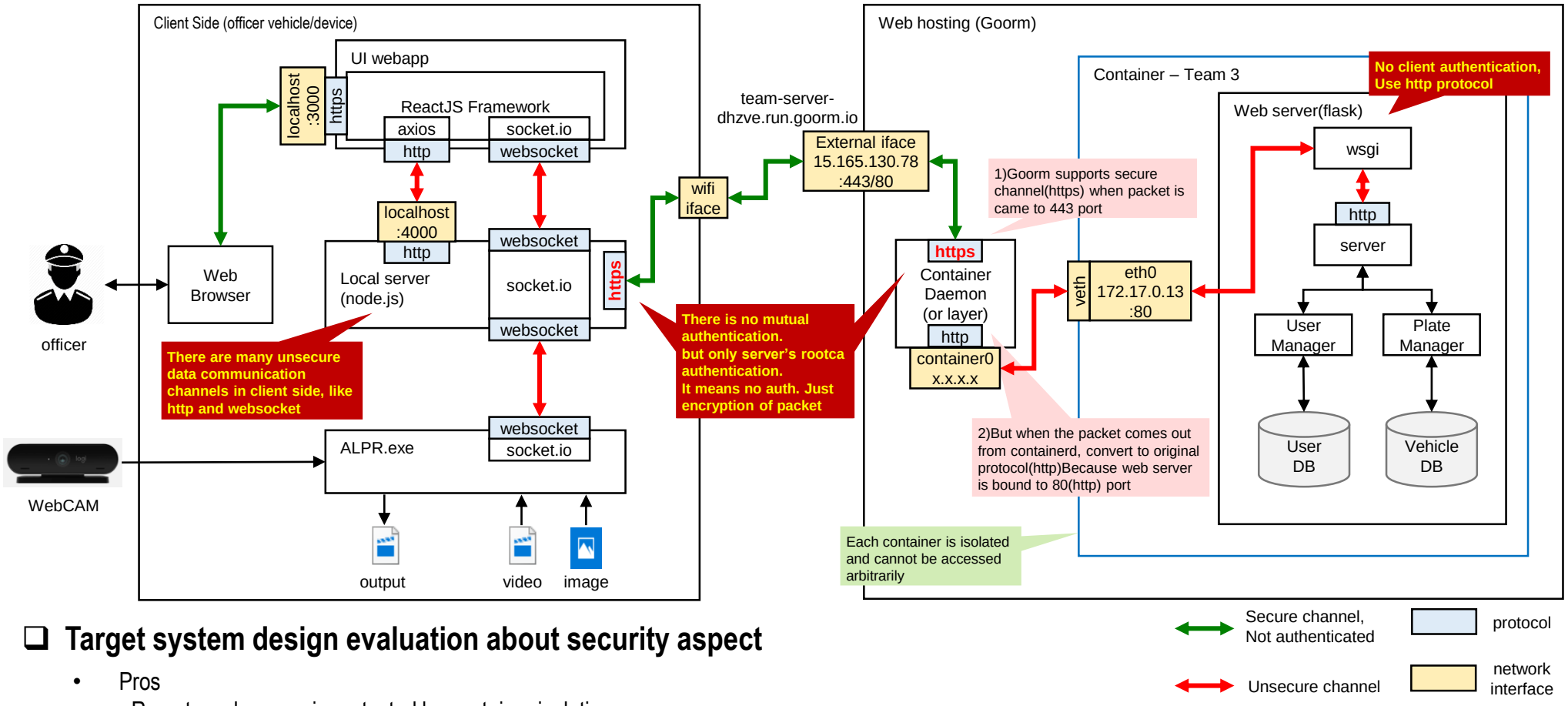
- Testing of Rest API using ZAP





We could not find any critical issue in Fuzz!

❑ Target system analysis for identifying module and data communication



❑ Target system design evaluation about security aspect

- Pros
  - Remote web server is protected by container isolation
    → **container protects data and prevents from illegal access**
- Cons
  - On client side, there are many unsecured data communication channel (http, websocket)
    → **Packets are easily sniffed by packet capturing tool**
  - Web server use http protocol natively, and packets are wrapped with https (it is supported by Groom hosting service)
    → **There is no authentication of server/client, it means that it is vulnerable to spoofing**

Major results - Vulnerability list

- We found **27 vulnerabilities** on target software!
- Classification of discovered vulnerabilities

STRIDE	VID	vulnerabilities	
		Client side	Server side
Spoofing	V08, V11, V13, V15, V16, V18, V22, V23, V24, V25, V26	• Server connection and plate information inquiry are possible with an spoofing client app (ex. jwt token) • using phishing of fake client web site, get a user credential • Available to change https to http by tempering the client's server.js file .	• An attacker can retrieve plate information with a newly added user account by the stolen administrator credentials. • Remote server spoofing available - The external server provided by goorm does not support mut ual authentication using tls based on pki.
Tampering	V12, V13, V25, V26, V27	• A TOCTOU attack is possible because there is a time gap between file selection and file playback	• Illegal user account can be added by obtaining administrator privileges(ex. Admin cookie). - team-server-dhzve.run.goorm.io/admin/signup
Repudiation	V14, V21, V25, V26		• no log records for login or logout with an administrator or general account
Information Di sclosure	V01, V02, V03, V05, V06, V07, V09, V10, V15, V17, V18, V19, V20, V22, V23, V24, V25, V26	▪ User Credentials are disclosure by wireshark sniffing or web debugger tool, User Interface - User Name, Password, OTP, Token, Plate Information	• User names and OTP seeds are disclosure by anonymously accessible API - "team-server-dhzve.run.goorm.io/admin/log" - "team-server-dhzve.run.goorm.io/auth/<user>" • Admin cookie i s disclosure by wireshark sniffing or web debugger tool
DoS	V04	• App crash due to non input validation	
Elevation of P rivilege	V11, V12, V13, V16,	• User privileges can be obtained by using token	• Administrator privileges can be obtained by below API - team-server-dhzve.run.goorm.io/admin/signin

❑ Consequences Recommendations

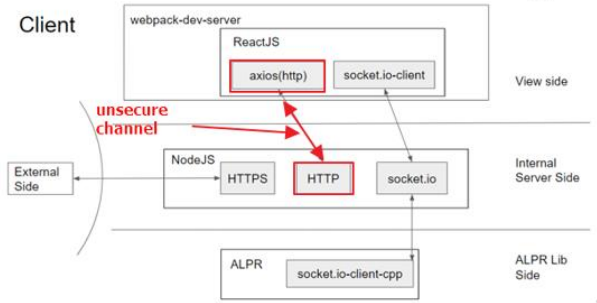
STRIDE	VID	vulnerabilities	
		Consequences / Impact	Recommendations
Spoofing	V08, V11, V13, V15, V16, V18, V22, V23, V24, V25, V26	• An attacker can acquire the plate information • An Attacker can add new account arbitrarily • An attacker can steal admin or general account information by fake client app • An attacker can steal admin or general account information by fake remote server using DNS spoofing • An attacker can sniff all information by changing HTTPS to HTTP by ta mpering with the internal server of the client app	• Using the PKI with SSL/TLS • Using JWT token revoke mechanism • No multiple access with one account
Tampering	V12, V13, V25, V26, V27	• An attacker can pollute the user db by adding illegal users • An attacker can be possible to disturb normal plate inquiry by replacing t he plate image	• Using the PKI with SSL/TLS • Using ACLs/permissions
Repudiation	V14, V21, V25, V26	• The attacker accesses the server with the stolen account, but the server does not record. So, it is impossible to track the attacker	• Logging
Information Di sclosure	V01, V02, V03, V05, V06, V07, V09, V10, V15, V17, V18, V19, V20, V22, V23, V24, V25, V26	▪ An attacker can steal general user credentials such as user name, password, otp seed, token and use this credentials to acquire the plate information ▪ An attacker can steal Administrator`s credentials such as admin name, password, otp seed, cookie and use this admin credentials to add new illegal account	• Minimize interfaces that can expose information disclosure • Provide the client interface in the form of WebApp. - Eliminate exposed IPC channels by operating in one process - Block external web debug tools at the WebApp level • Add access permissions to the API provided to the administrator • Using Javascript Obfuscation technology
DoS	V04	• An attacker can disable the use of the client app	• Check to input validation on the client side • Using file input filters
Elevation of Pr ivilege	V11, V12, V13, V16,	• An attacker can acquire the plate information • An Attacker can add new account arbitrarily	• Eliminate account information disclosure by using role based access controls for server API. • Minimize attackable URLs such as "team-server-dhzve.run.goorm .io/auth/<user>" • When the number of failed logins is exceeded, the account is lock ed

Representative Vulnerability – Sniffing User Credential

Description	[V01] UI Webapp in client side send user credential as raw string, not encrypted		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/DESIGN]	Exploit technique	[SNIFFING]

Vulnerabilities

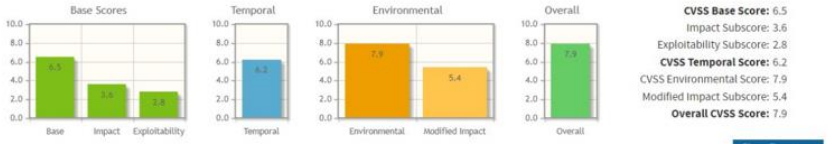
1. In client side, there are three parts; webapp for UI, local nodejs server and ALPR
2. To communicate between webapp UI and local server, HTTP which is not secured channel is used
3. It causes information disclosure, because data is not encrypted



CVSS Score	7.9	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



Consequence / Impact analysis

The officer's credential is not only for ALPR service. It may be SSO(Single Sign On) account for police office. So, this officer's credential can be used for following attack like penetrate police office server and connect to police office portal and then steal target data and so on. In addition, store this user credential and use this for another site's dictionary attack. Because people sometimes use same ID/PW on various site. Also, attacker can sell user credential to illegal group and make money

Recommended mitigation

Use secure communication channel protocol like HTTPS(above TLS version 1.2) between module, instead of HTTP

Tools needed	Wireshark(Packet capture)

Relative component / source code

Local Server - UI webapp

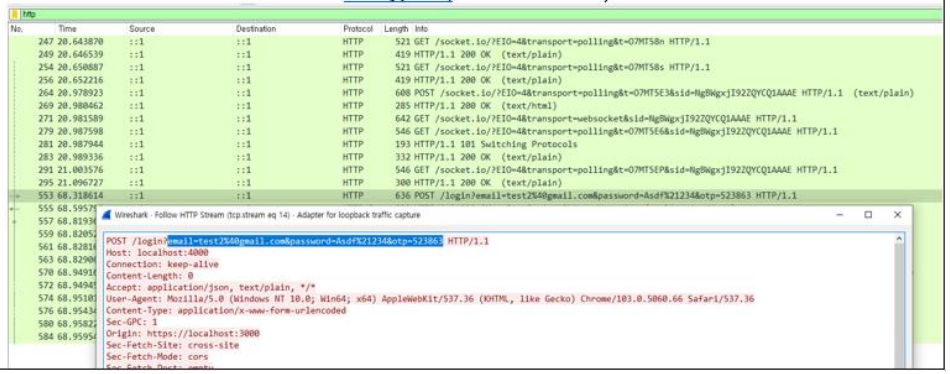
Proof of concept / How to attack

Constraints:

- Attacker can access client device and run Wireshark for packet capturing

Procedure:

1. Run Wireshark for capturing packets and bind to localhost
2. Connect to officer UI webpage(https://localhost:3000)
3. Attempt to login with ID/PW/OTP which are provided
4. Check that user credential is not encrypted(HTTP Protocol) in Wireshark



Representative Vulnerability – Server Spoofing(1)

Description	[V22] Remote server spoofing		
CIA	[Availability] [Confidentiality]	Attack vector	[NETWORK]
Approach	[REVIEW/DESIGN]	Exploit technique	[SPOOFING]

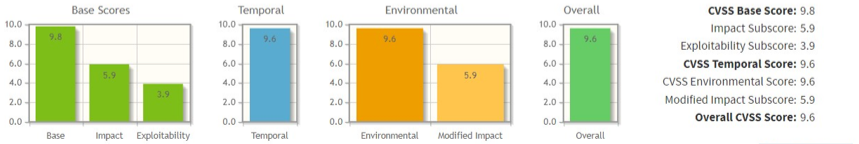
Vulnerabilities

1. Local server in client side and remote server do not proceed with mutual authentication.
2. The remote server is implemented with http, but Goorm hosting service(COTS) supports https(TLS) when packet is came into 443 port. And convert to http packet because remote server in container is bounded at 80 port(HTTP). It means that client authentication is omitted
3. Local server uses HTTPS/TLS1.2, but dose not authenticate server's certificate, just verify root CA of server certificate if this certificate is issued from authorized agency. It means that there is no authentication of server identification and encrypt packet on external network only.
4. If attacker can redirect remote server URL to attacker's URL/IP, server spoofing could be done without any additional procedure

CVSS Score	9.6	Severity	Critical

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



Consequence / Impact analysis

Spoofing server can occur stealing sensitive data (e.g. user credential) and it is sold to illegal group for making money or used for following attack. Secondly, officer cannot execute law enforcement. It means that ALPR service is completely down (all stop) as long as DNS spoofing is worked. It damages to police office(government) reputation critically.

Recommended mitigation

Apply mutual authentication procedure between server and client like TLS(above 1.2)

Tools needed	Goorm hosting service

Relative component / source code

Local Server in client side

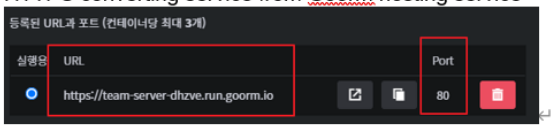
(there is no additional authentication procedure or option, just request with https)

client/web/src/images/server.js

```
app.post('/login', function(req, res){
    let email = req.query.email;
    let password = req.query.password;
    let otp = req.query.otp;

    const url = host +"/signin?username="+email+"&password="+password+"&otp="+otp;
    const request = https.request(url, (response) => {
        let data = '';
        response.on('data', (chunk) => {
            data = data + chunk.toString();
        });
```

HTTPS converting service from Goorm hosting service



HTTP bind in remote server(server.py)

```
app = Flask(__name__)

if __name__ == '__main__':
    app = Flask(__name__)
    app.config['DEBUG'] = True
    app.config['SQLALCHEMY_DATABASE_URI'] = config._DB_PATH_
    app.config["SECRET_KEY"] = config._SESSION_SECRET_KEY_

    db.init_app(app)
    if not os.path.isfile('./server.db'):
        with app.app_context():
            db.create_all()
            createdb()

    app.register_blueprint(usermanagement.user_management)
    app.register_blueprint(platequery.plate_query)

    app.run(host='0.0.0.0', port=80)
```

Representative Vulnerability – Server Spoofing(2)

Proof of concept / How to attack

Constraints:

1. Attacker has capability for DNS spoofing in external network (DNS cache poisoning)
2. Attacker should have cert which is not self-signed
   → we use goorm hosting service for certificate which issued from root CA (real attackers prepare their site with root CA and IP)
   → But It works with container, cannot access directly remote server in ours container with URL or IP
   → So verifying server spoofing, we test with URL modification in code like below,

```
//const host = "https://team-server-dhzve.run.goorm.io";
const host = "https://con-t-ythyc.run.goorm.io";
```

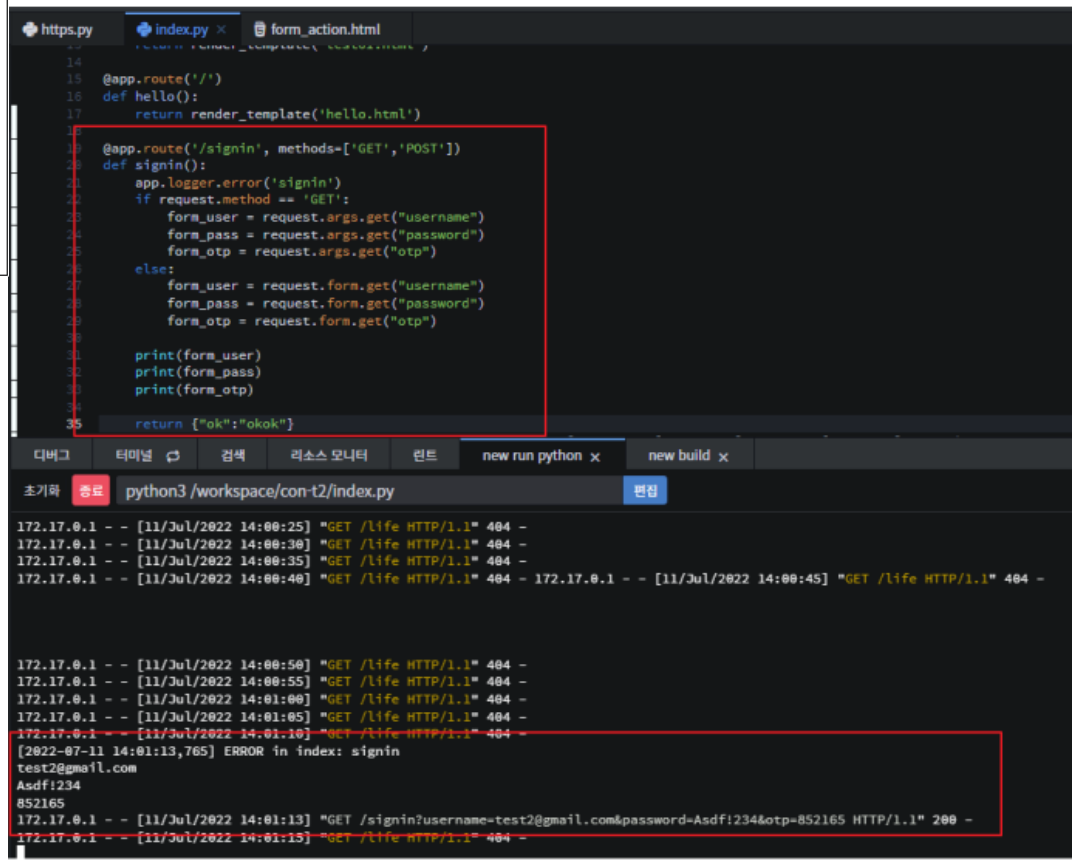
(below is attacker's)

⇨ Assume that DNS spoofing is applied for team3's server URL by modifying URL in source code

Procedure:

1. make fake server with python flask and run in goorm hosting service
2. connect localhost:3000 and try to login
3. server is spoofed
4. check if user credential is received or not in attacker's server
5. check if ALPR service doses not worked well

faker server's impl - signin method : sniffing user credential



without processing signin, we just sniffing
http get method and user credential is passed

6. Lessons learned

- Lessons learned

❑ Lessons Learned

- ✓ Thinking like an attacker!
- ✓ Not all vulnerabilities can be eliminated. However, it should be possible to remove as much as possible at each stage of the SDL cycle.
- ✓ Security issues also arise where there is no direct relationship with the security requirements.
  It seems to occur mainly in weak design, implementation, and the interface between modules.
- ✓ Software requirement is easily implemented with COTS, opensource and so on.
  But without deep consideration about threat analysis, security requirement and design, It can occur tremendous vulnerabilities
- ✓ By finding vulnerabilities in target software, we could find hidden security issue and mitigation reversely

Lessons learned 2/2

❑ Learned methodologies, techniques and tools

- ✓ Threat modeling using STRIDE → To identify and eliminate potential vulnerabilities
- ✓ Understanding into the most common security risks so that we could use the findings of the report as part of our class-mates security practices Using OWASP
- ✓ How much using code-review and utilizing some tools is important
- ✓ Utilize wireshark, Chrome inspector and a few types of static analysis tools

Thank You