**Automatic License Plate Recognition (ALPR) System**

**Team3 PURPLE**

* Kibong Song
* Brian Moon
* Seungkyu Lee
* Haenggi Lee
* Seongsik Kim
* Youngmi Choi

**Table of Contents**

[**1.**](#_heading=h.30j0zll) **Terminology and Definitions 5**

[**2.**](#_heading=h.1fob9te) **Security goals 7**

[2.1 Business goals 7](#_heading=h.3znysh7)

[2.2 Security goals 7](#_heading=h.2et92p0)

[**3.**](#_heading=h.tyjcwt) **Customer Requirement 8**

[3.1 Client 8](#_heading=h.3dy6vkm)

[3.2 Server 8](#_heading=h.1t3h5sf)

[3.3 Quality Constraint 8](#_heading=h.4d34og8)

[**4.**](#_heading=h.2s8eyo1) **Solution & Environment 10**

[4.1 Server 10](#_heading=h.17dp8vu)

[4.2 Client 10](#_heading=h.3rdcrjn)

[**5.**](#_heading=h.26in1rg) **Software inventory 11**

[5.1 New feature 11](#_heading=h.lnxbz9)

[5.2 Reuse 11](#_heading=h.35nkun2)

[5.3 Open Source SW 11](#_heading=h.1ksv4uv)

[5.4 COTS 11](#_heading=h.44sinio)

[**6.**](#_heading=h.2jxsxqh) **Development Artifacts 12**

[6.1 use case 12](#_heading=h.z337ya)

[6.2 misuse case 12](#_heading=h.3j2qqm3)

[6.3 DFD 0 12](#_heading=h.1y810tw)

[**7.**](#_heading=h.4i7ojhp) **Threat Modeling 13**

[7.1 STRIDE 13](#_heading=h.2xcytpi)

[**8. Risk Assessment 14**](#_heading=h.1ci93xb)

[8.1 HEAVENS 14](#_heading=h.3whwml4)

[**9. Requirement Elicitation 15**](#_heading=h.2bn6wsx)

[**10. Requirement Inspection 16**](#_heading=h.qsh70q)

[10.1 checklist 16](#_heading=h.3as4poj)

[**11. Architecture Design 17**](#_heading=h.1pxezwc)

[11.1 Static View 17](#_heading=h.49x2ik5)

[11.2 Dynamic View 17](#_heading=h.2p2csry)

[**12. SW implementation 18**](#_heading=h.147n2zr)

[15.1 Server SW structure 18](#_heading=h.3o7alnk)

[15.2 Client SW structure 18](#_heading=h.23ckvvd)

[**13. SW vulnerability evaluation 19**](#_heading=h.ihv636)

[16.1 open source scanning 19](#_heading=h.32hioqz)

[16.2 secure coding rule 19](#_heading=h.1hmsyys)

[16.3 vulnerability report 19](#_heading=h.41mghml)

[**14. Test 20**](#_heading=h.2grqrue)

[14.1 Test methodology 20](#_heading=h.vx1227)

[14.2 Test Tool 20](#_heading=h.3fwokq0)

[14.2 Test Result 20](#_heading=h.1v1yuxt)

[**18. Test Bug 21**](#_heading=h.4f1mdlm)

[**19. Deliverables 22**](#_heading=h.2u6wntf)

[19.1 Source code 22](#_heading=h.19c6y18)

[19.2 Test cases 22](#_heading=h.3tbugp1)

[19.3 Threat modeling 22](#_heading=h.28h4qwu)

[19.4 Requirements 22](#_heading=h.nmf14n)

[19.5 Design document 22](#_heading=h.37m2jsg)

[19.6 Test bugs 22](#_heading=h.1mrcu09)

[19.7 Vulnerability report 22](#_heading=h.46r0co2)

[19.8 Seeded vulnerability list 22](#_heading=h.2lwamvv)

[**[Annex 1] SW Build Guide 23**](#_heading=h.111kx3o)

[**[Annex 2] User Manual 24**](#_heading=h.3l18frh)

| **Version** | **Date** | **Status** | **Author** |
| --- | --- | --- | --- |
| 1.0 | 2022.07.05 | 1st release | team3 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 1. Terminology and Definitions

This section provides a description of the terms and definition that will be used throughout this document.

| Term | Description |
| --- | --- |
| Abuse Use Case | Deliberate abuse of functional use cases in order to yield unintended results. |
| Accountability | The property that ensures that the actions of an entity may be traced uniquely to that entity. |
| Actor (Threat Agent) | Person who originates attacks, either with malice or by accident, taking advantage of vulnerabilities to create loss. |
| Application  Programming  Interface (API) | A source code interface that a computer system or program library provides to support requests for services to be made of it by a computer program [PCI HSM Security Req]. |
| Asset | An asset is a resource of value. It varies by perspective. To a business, an asset might be the availability of information, or the information itself, such as customer data. It might be intangible, such as a company's reputation. |
| Attack (Exploit) | An attack is an action taken that utilizes one or more vulnerabilities to realize a threat. |
| Attack Surface | Logical area (browser stack, infrastructure components, etc.) or physical area (hotel kiosk) that an attack may occur or originate from. |
| Attack Vector | Point and channel for which attacks travel over (card reader, form fields, network proxy, client browser, etc.). |
| Authenticity | The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator [NIST SP 800-137, CNSSI 4009]. |
| Authorization | The official management decision given by a senior organizational official to authorize operation of an information system and to explicitly accept the risk to organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, and the nation based on the implementation of an agreed-upon set of security controls [NIST SP 800-137, CNSSI 4009]. |
| Availability | Ensuring timely and reliable access to and use of information [NIST SP 800-137, 44 U.S.C., Sec. 3542].  Capability of a product to provide a stated function if demanded, under given conditions over its defined lifetime [ISO 26262-1]. |
| Confidentiality | Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information [NIST SP 800-137, 44 U.S.C., Sec. 3542]. |
| Countermeasures (Control) | Countermeasures address vulnerabilities to reduce the probability of attacks or the impacts of threats. They do not directly address threats; instead they address the factors that define the threats. |
| Impact | Value of damage possibly sustained via an attack. |
| Integrity | Guarding against improper information modification or destruction and includes ensuring information non-repudiation and authenticity [NIST SP 800-137, 44 U.S.C., Sec. 3542]. |
| Multi-tenant | An architecture in which a single computing resource is shared but logically isolated to serve multiple consumers [NIST.SP.500-322]. |
| Non-repudiation | The ability to provide proof of the integrity and origin of data. |
| Privacy | The ability to provide protection against personal data discovery and misuse of that information by other users [Common Criteria Part 2]. |
| Possession and/or control | the system and associated processes shall be designed, implemented, operated and maintained so as to prevent unauthorized control, manipulation or interference |
| Randomness | A random bit sequence could be interpreted as the result of the flips of an unbiased “fair” coin with sides that are labeled “0” and “1,” with each flip having a probability of exactly ½ of producing a “0” or “1.”  Furthermore, the flips are independent of each other: the result of any previous coin flip does not affect future coin flips. The unbiased “fair” coin is thus the perfect random bit stream generator, since the “0” and “1” values will be randomly distributed (and [0,1] uniformly distributed). All elements of the sequence are generated independently of each other, and the value of the next element in the sequence cannot be predicted, regardless of how many elements have already been produced [NIST 800-22]. |
| Safety | The design, implementation, operation and maintenance of the system and associated processes shall not jeopardize the health and safety of individuals, the environment or any associated assets.  Absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems [ISO 26262-1]. |
| Tampering | The ability to change data in transit or in a data store. |
| Threat | A threat is an undesired event. A potential occurrence often best described as an effect that might damage or compromise an asset or objective. It is relative to each site, industry, company and is more difficult to uniformly define. |
| Use Case | Describes the functional interaction between an application and its users. |
| Utility | The system and associated processes shall be designed, implemented, operated and maintained so that the utility of their assets is maintained throughout their life cycle |
| Security Use Case | Specifies the security requirements that the application shall successfully protect itself from its relevant security threats. |
| Vulnerability | Part of the information security infrastructure that could represent a weakness to attack in the absence of a control. |
| Sensitive Data | User Credentials  Social Security number or other identifying information  Credit card numbers or other financial information  Health information  Private keys or other data that could be used to decrypt encrypted information  System or application information that can be used to more effectively attack the application |
| ALPR system | Automatic License Plate Recognition System |

# 2. Customer Requirement

## 2.1 Client

#### 2.1.1 Functional Requirements

1. The system shall allow an officer to login and authenticate users locally and to the backend license plate database lookup. The system must use two factor authentication for sign on and user credentials must be protected.

2. The system should allow a law enforcement officer to select and save retrieved information locally.

3. The system should allow a law enforcement officer to send retrieved information to a mobile device, such as a mobile phone to use in the field.

4. The system should allow officers to configure computed camera / playback frames per second, average time per frame, jitter and frame number.

5. The system should allow the officer to choose between using a live camera and playback file in the UI.

#### 2.1.2 Non-Functional Requirements

1. The system shall allow an officer to access the ALPR system through a secure web interface.

2. Lost or compromised credentials must be handled in a reasonable way.

3. The system should provide secure communication between the client application and to the backend license plate database lookup system.

4. The system should read images from the vehicle camera or a playback file and identify license plates for evaluation.

5. The system should perform the ALPR function in real-time while maintaining a frame rate of at least 25fps.

6. The system should query the backend license plate server for details about the vehicle. The user must be alerted for vehicles that are stolen, the owner is wanted (criminal), or if it is a vehicle of interest (expired registration, unpaid tickets, owner is missing). Alerts must contain reason and vehicle make, model and color along with the isolated plate image and the recognized license plate number for operator comparison.

7. If a license plate does not generate an alert, then the user interface must display the last recognized plate image, the recognized license plate number and vehicle make, model and color so the operator can visually check if the plate matches the vehicle if desired.

8. The system should provide an area in the user interface that always contains the current camera /playback view.

9. The ability to detect network connectivity issues with the backend server within 5 seconds and automatically resolve the communication issue if possible.

10. The system should alert officers of any communication errors or failures.

11. The system must fetch vehicle information in no more than 10 seconds as officers are often making queries in real time.

## 2.2 Server

#### 2.2.1 Functional Requirements

1. Support license plate queries.

2. Authenticate remote laptop users.

3. Support configurable values via a configuration file

#### 2.2.2 Non-Functional Requirements

1. Ensure secure communication with the client applications.

2. Support multiple users.

3. Return the best match license plate if there is not an exact match that includes a configurable minimum confidence threshold to support a partial match.

4. Track the average number of queries per second for each user and overall queries per second, for all users.

5. Track the number partial matches and no matches for each user and all users

## 2.3 Common Information

1. Ensuring that all software in both applications are architected and coded to be secure and free of vulnerabilities.

2. Conduct proper fault/error detection, recovery and reporting.

3. Ensure the developed software adheres to the company coding standard and quality standards.

4. Ensure the developed software is adequately tested.

5. The application should allow a law enforcement officer to use a video feed/picture to identify a license plate and then query that license plate number to determine if there are any outstanding fines or warrants against the vehicle’s owner.

6. The proposed system should be a client server system where the client is an on-board computer in a police vehicle or mobile device carried by an officer.

Initial customer requirements have some parts that are ambiguous or unclear, and have been detailed through discussion with the customer (David Belasco).

In the next step, the security requirements will be elicited using the SQUARE methodology.

The SQUARE methodology to help organizations build security into the early stages of the production life cycle.

# 3. Security goals

## 3.1 Business goals

The system allows authorized users to make decisions based on the information provided by the image recognition web-based system.

## 3.2 Security goals

G-01 System should maintain confidentiality and integrity of databases when a system admin or an officer reads and writes databases.

G-02 The confidentiality of the network communication should be maintained when the server and a client communicate.

G-03 System needs to detect a breach of personal identifiable information of databases when an admin or an officer read and write databases.

# 4. System definition

## 4.1 Server

1. Hardware : Cloud hosting platform
2. Environment : WSGI server - it will run the web app based on python web framework
3. OS : Linux
4. SW module : plate query module, user management module, config module
5. data : access control list, vehicle plate information

## 4.2 Client for laptop

1. Hardware : laptop computer
2. Environment : ReactJS, NodeJS, SocketIO
3. OS : Windows
4. SW module : GUI, Connection Module
5. data : Video file, License Plate Image

# 5. Software inventory

## 5.1 Reuse

ALPR source code from CMU.

The source code provided by the customer does not proceed with the analysis to see if it meets the requirements.

## 5.2 Open Source SW

1. Server
   1. security goal
      1. The server system shall be available for use when needed.
      2. The confidentiality and integrity of the system’s DB shall be maintained.
      3. The server system should support basic security features.
   2. preliminary security requirements.
      1. All sensitive data should be encrypted.
      2. Whole user operations should be monitored.
   3. tradeoff analysis
      1. Additional time is required for encryption/decryption when serving data from the server.
      2. Need a database space for logging all user actions and an interface for monitoring.
   4. open source name (python package)
      1. click
      2. itsdangerous
      3. jinja2
      4. werkzeug
      5. flask
      6. flask-sqlalchemy
      7. sqlalchemy-utils
      8. pyotp
      9. pyjwt
      10. cryptography
      11. faker
      12. faker-vehicle
2. Client
   1. security goal
      1. The system shall allow an officer to access the ALPR system through a secure web interface.
      2. The system must use two factor authentication for sign on and user credentials must be protected.
      3. The system should provide secure communication between the client application and to the backend license plate database lookup system.
   2. preliminary security requirements
      1. The web interface uses HTTPS.
      2. Enter the google OTP number in the sign-in step.
      3. The communication method is HTTPS.
   3. tradeoff analysis
      1. n/a
   4. open source name (npm package)
      1. react
      2. react-bootstrap
      3. react-dom
      4. react-perfect-scrollbar
      5. react-router-dom
      6. react-scripts
      7. socket.io-client
      8. web-vitals
      9. yup
      10. axios
      11. formik
      12. http-proxy-middleware
      13. @mui/icon/ns-material
      14. @mui/material
      15. npm
      16. express
      17. path
      18. https
      19. http
      20. cors
      21. socket.io
      22. child\_process
      23. socket.io-client-cpp
      24. openalpr(provided by CMU)

## 5.3 COTS

Not used

# 6. Asset Identification

It was considered an important asset that influenced users to make informed decisions and derived assets.

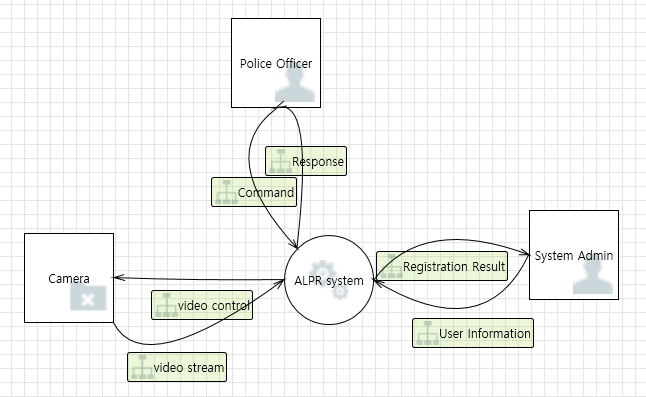
| **System** | **Assets** | **Security Characteristics** | **Damage Scenario** |
| --- | --- | --- | --- |
| External Entity | police officer | Authentication | Attackers may steal the user's ID, password, and OTP information, access the laptop client app, and assume the user's ID and password. |
| External Entity | admin | Integrity, Authorization | Attackers can acquire administrator rights through an escalation of privilege attack to compromise the DB or leak information. |
| Server | user DB | Integrity | Attackers can access the user DB and change the ID and password information. |
| Server | user DB | Confidentiality | Attackers can access user DB and obtain personal information. |
| Server | Drive/Car DB | Integrity | Attackers may access the Drive/Car DB to change vehicle information and owner information. |
| Server | Drive/Car DB | Confidentiality | Attackers can access the Drive/Car DB to obtain vehicle information and owner information. |
| Server | Drive/Car DB | Availability | Attackers can paralyze the ALPR system by performing excessive queries on the Drive/Car DB. |
| Client | Drive/Car Info. Storage | Integrity | Attackers may access Drive/Car Info. Storage to change vehicle information and owner information. |
| Client | Drive/Car Info. Storage | Confidentiality | Attackers can access Drive/Car Info. Storage to obtain vehicle information and owner information. |
| Network Interface | laptop client ↔ server | Integrity, Confidentiality | Attackers may cause a malfunction of the system by stealing and modulating data transferred between the client and the server. |
| Network Interface | laptop client ↔ server | Availability | Attackers can cause a network overload. |

## 

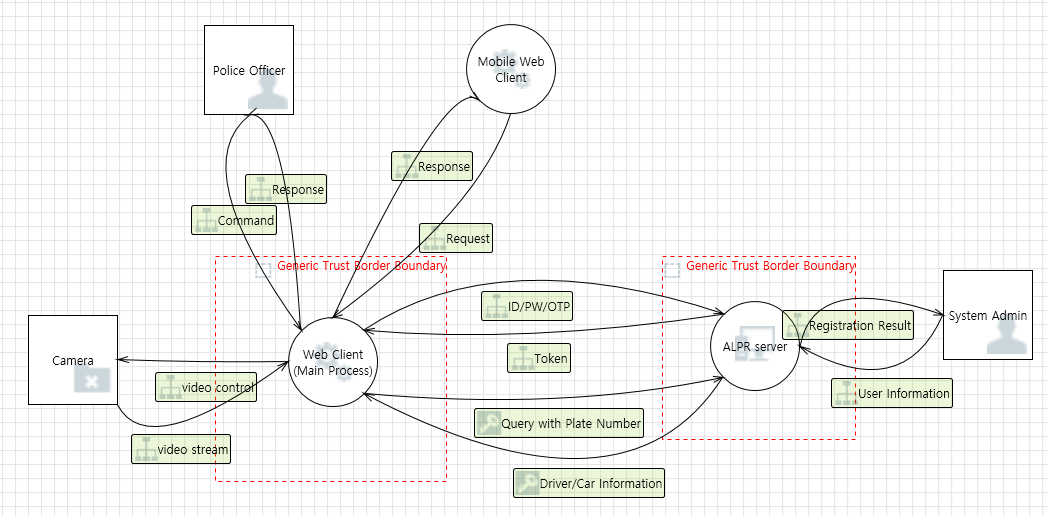
# 7. Threat Analysis

## 7.1 STRIDE

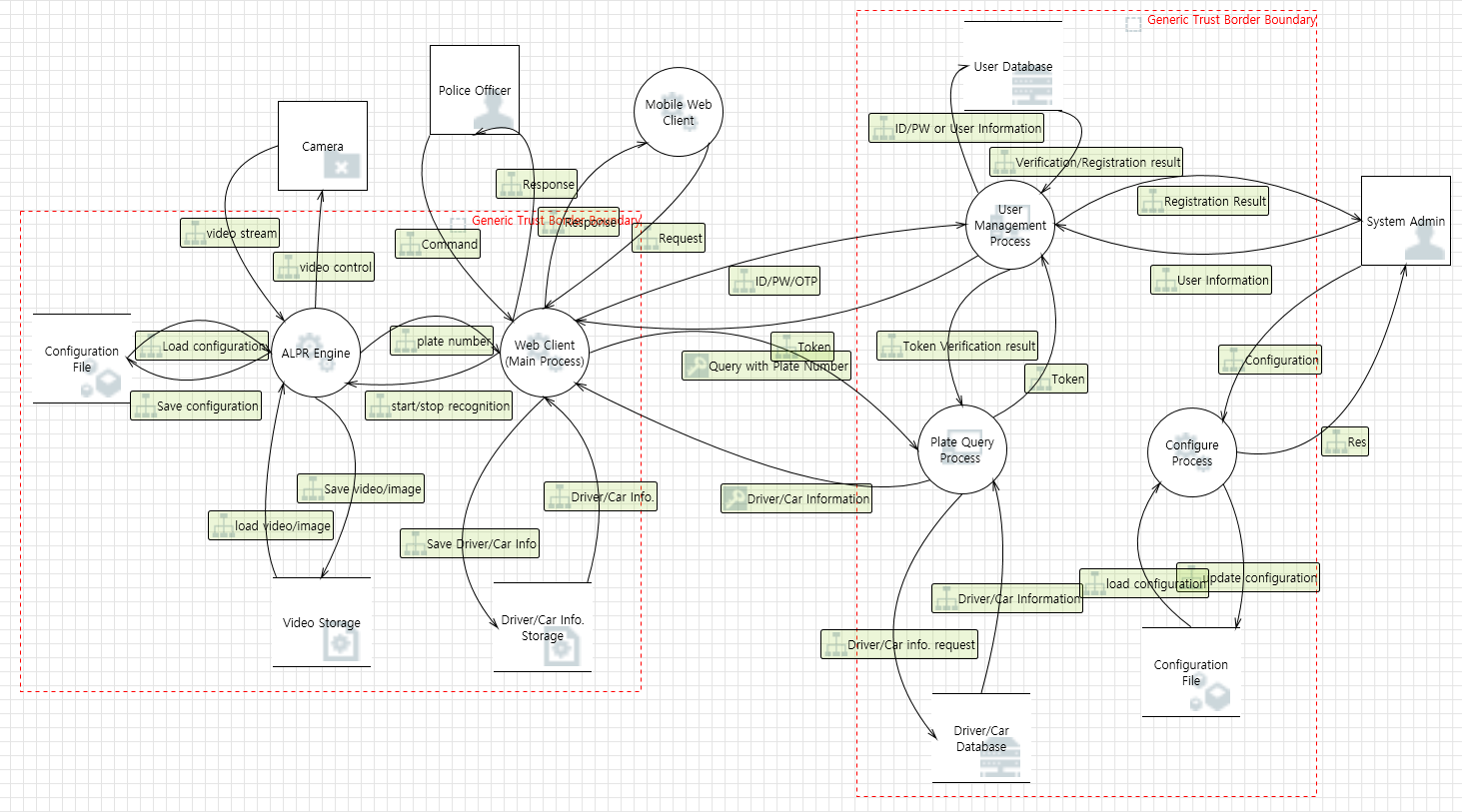
1. Development Artifacts
   1. DFD 0



* 1. DFD 1



* 1. DFD 2

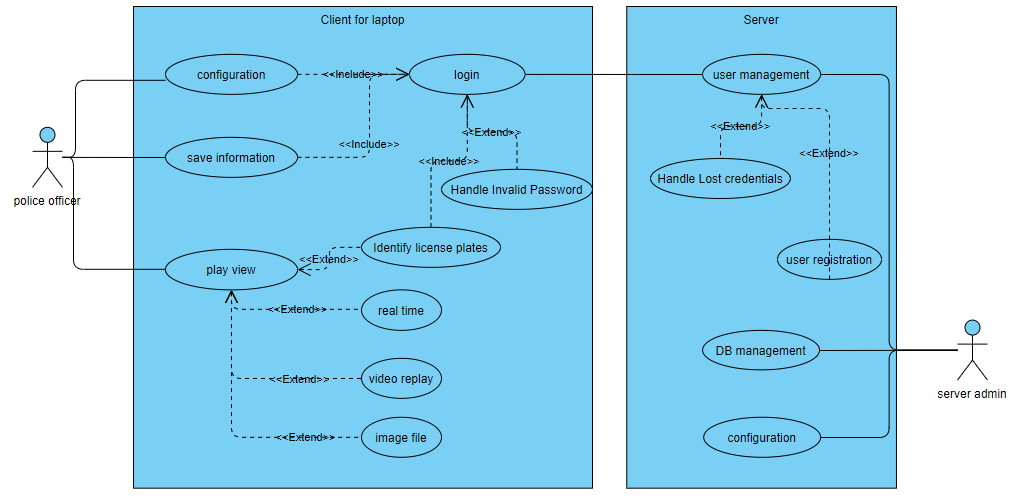


1. Threats list
   1. 145 threats were derived using Microsoft threat modeling tool.
   2. You can find the “All threats” in the “Team3\_purple\_ALPR\_ThreatModeling.xlsx" document in the deliverables.

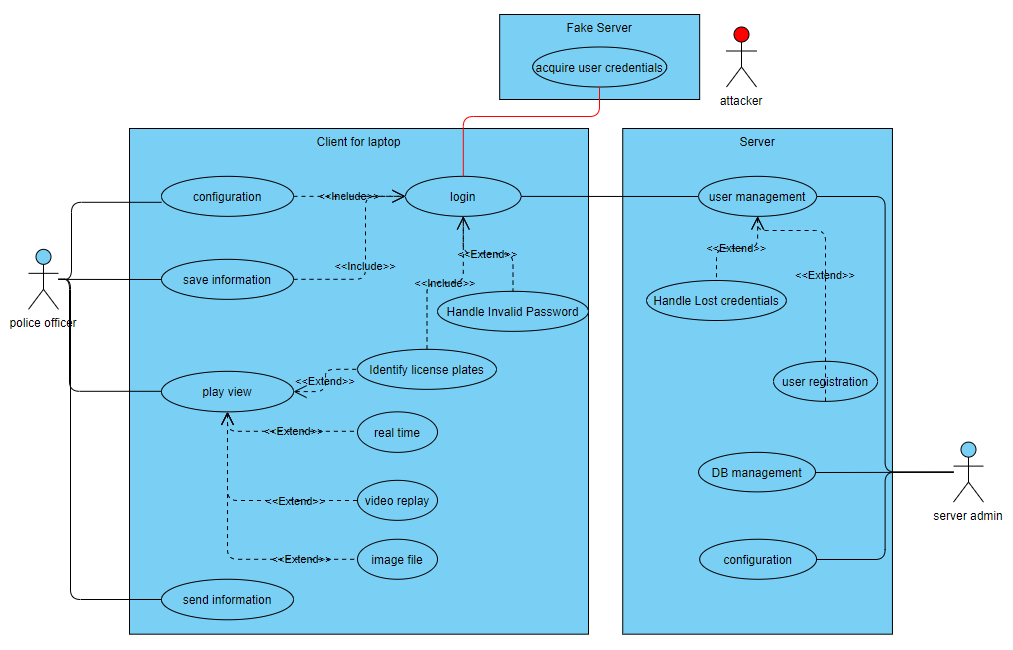
## 7.2 PnG

PnG was briefly used to derive threats from the attacker's point of view.

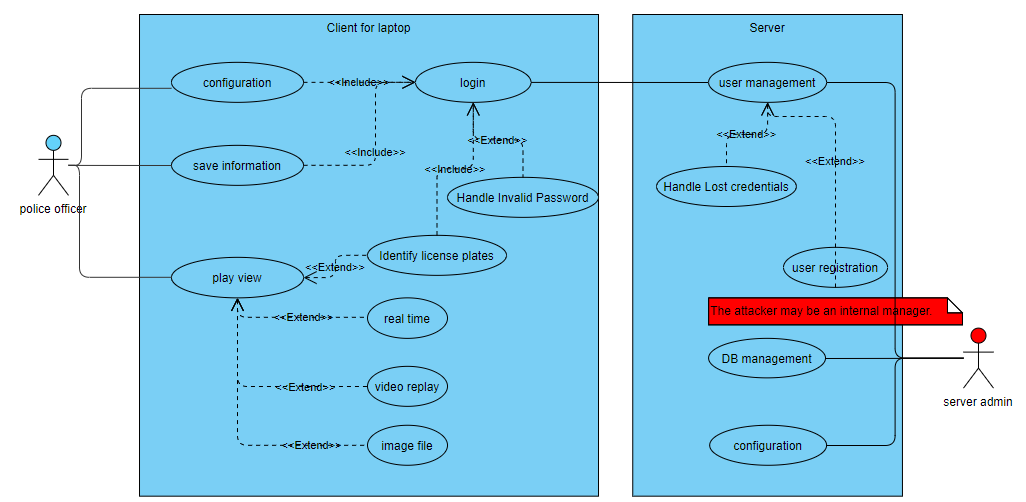
1. Development Artifacts
   1. use case diagram



* 1. misuse case diagram



1. PnG Type #1 : boring police officer
   1. motivations :
      1. I'm so free and bored these days. I want to create an incident or an accident.
   2. goals :
      1. I want to create a chaotic situation by malfunctioning the ALPR system used by road patrol officers.
   3. skills :
      1. I know how to distribute the client application.
      2. I have a friend who has experience in hacking the client/server system.
   4. misuse cases :
      1. boring police officer tells fellow police officers they need a client update and give them a fake server link
      2. All user account information accessed on the fake server is hijacked.
      3. The attacker accesses the client application with the acquired user account information to freely use and manipulate the desired information.
   5. threats :
      1. login spoofing



1. PnG Type #2 : Backend license plate server administrator without a sense of duty
   1. motivations :
      1. I want to get revenge on my ex-girlfriend after receiving a breakup notice from her.
      2. After the successful revenge of his girlfriend, there was a client who wanted to take small private revenge, so he earned extra income.
   2. goals :
      1. I want to make a fine by leaving a record of non-payment of the fine on the DB status of my ex-girlfriend's vehicle.
      2. I leave a false record of non-payment of fines in the requested vehicle DB.
   3. skills :
      1. Good knowledge of license plate server DB schema and how to change it
      2. Ability to delete DB access logs
   4. misuse cases :
      1. Falsely change the non-payment of fine information in the vehicle status information of the ex-girlfriend or client and remove change logs
   5. threats
      1. repudiation of authorized user

## 7.3 Critical threats selection

Since the threat affecting the asset identified in Section 6 is important, the critical threat was selected based on this.

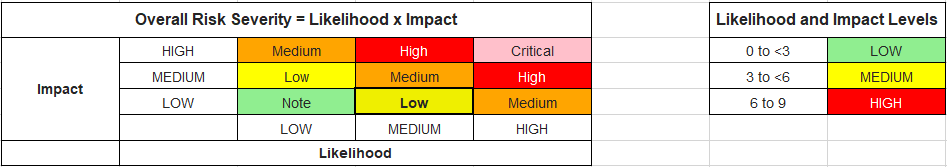
You can find the “Critical treats” in the “Team3\_purple\_ALPR\_ThreatModeling.xlsx" document in the deliverables.

In the case of an actual project, the entire project must be analyzed, but since it is a project for assignment with time constraints, some critical threats were selected and subsequent activities were carried out.

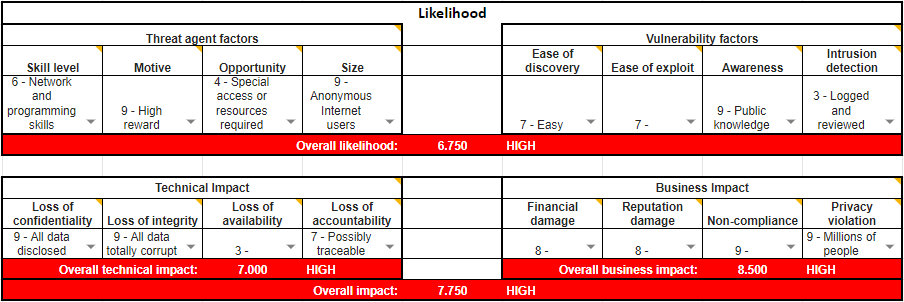
# 8. Security Risk Assessment

## 8.1 Risk Rating

We evaluated the risk of the asset inside the system among the identified assets using the OWASP method.

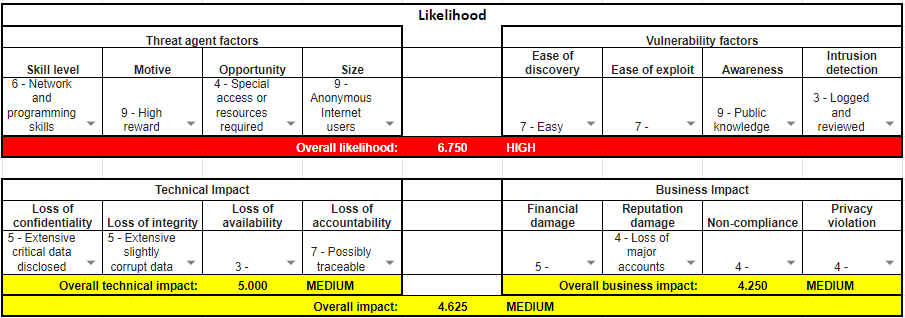


1. Drive/Car DB in server



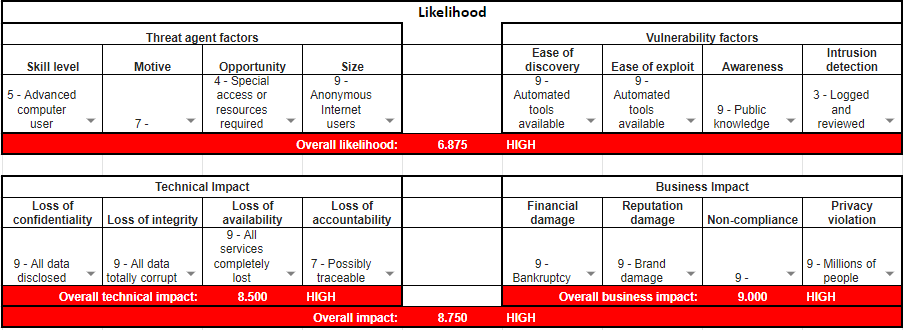
**Overall Risk Severity : CRITICAL**

1. User DB in server



**Overall Risk Severity : HIGH**

1. client ↔ server



**Overall Risk Severity : CRITICAL**

## 8.2 Threat Priority

1. The priority of selected threats is critical. This is because all of the critical threats selected above are DB managed by the server and query parts between the server and the client.
2. You can find the “Critical treats” in the “Team3\_purple\_ALPR\_ThreatModeling.xlsx" document in the deliverables.

# 9. Mitigating Threats

The mitigation measures did not devise new ones, but were used referring to existing ones.

In addition, they tried to use mitigation measures that could prevent multiple threats at once.

References) <https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-authentication>

<https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html>

You can find the “Critical threats” including mitigation in the “Team3\_purple\_ALPR\_ThreatModeling.xlsx" document in the deliverables.

10 security requirements (SR1 to SR10) were derived as threat modeling, and 10 requirement specifications (RS37 to RS47) applied with mitigation measures were prepared.

You can find the details in the “ALPR REQ. tracking” in the “Team3\_purple\_ALPR\_Requirements.xlsx” document in the deliverables.

# 10. Final product specification

## 10.1 Categorize requirements as to level

The requirement specifications are classified according to the following criteria.

| **Category** | **Meanings** |
| --- | --- |
| System level | information on the requirements for a system. |
| Software level | in-depth descriptions of the software that will be developed. |
| Technical constraints | the Non-Functional aspects of a system or component, such as restrictions on technology, resources or techniques to be used. |
| Business constraints | the Non-Functional aspects of a system or component, such as restrictions on business. |

## 10.2 Prioritize requirements

The priority of the requirements was classified as Essential, conditional, and optional, and was determined by discussing it with the stakeholder from a security perspective.

Requirements set as Optional were excluded from the scope of this development.

| **Priority** | **Meanings** |
| --- | --- |
| Essential | the product is not acceptable unless these requirements are satisfied |
| Conditional | would enhance the product, but the product is not unacceptable if absent |
| Optional | functions that may or may not be worthwhile |

## 10.3 Requirements inspection

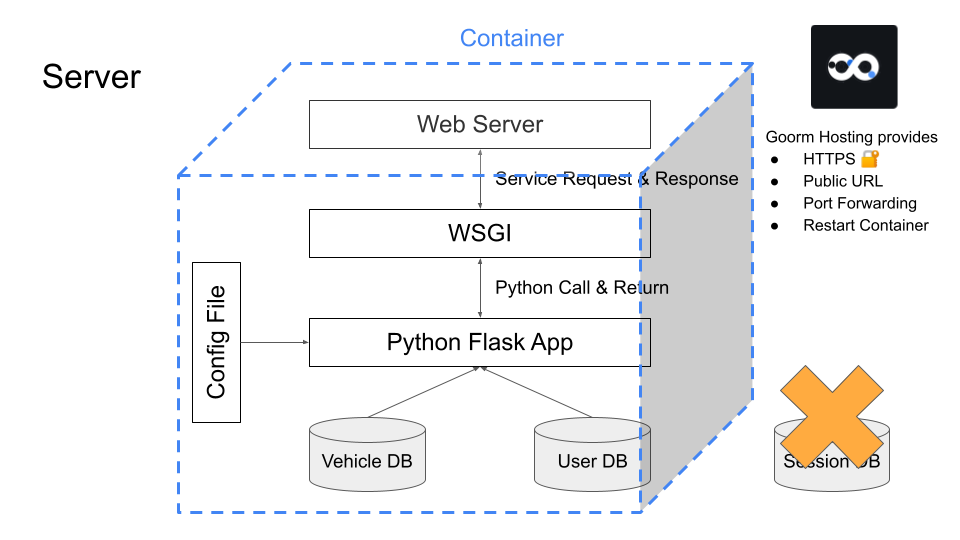
For requirement inspection, stakeholders gathered to conduct requirements walkthrough.

The checklist used for inspection can be found in the deliverables.

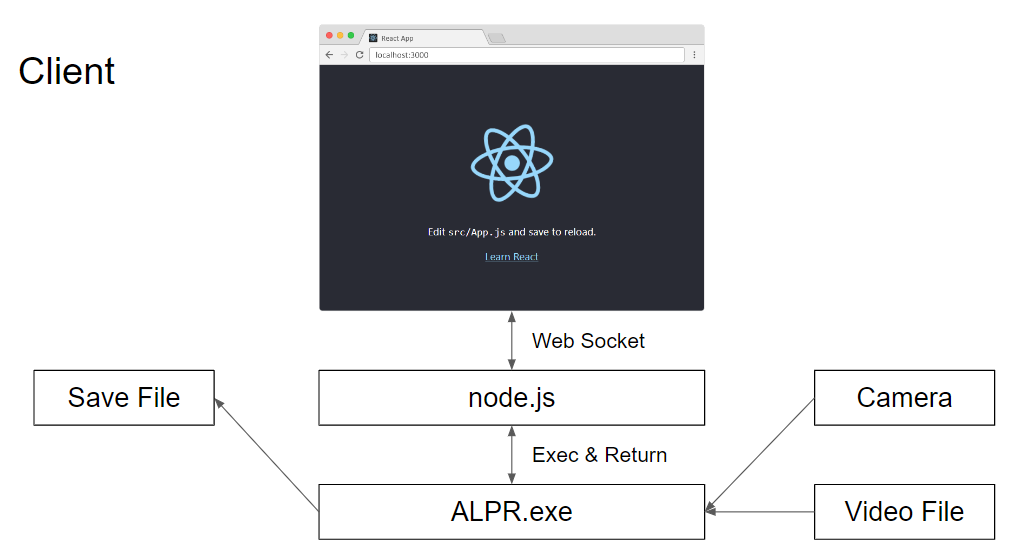
You can find the details in the “ALPR REQ. tracking” in the “Team3\_purple\_ALPR\_Requirements.xlsx” document in the deliverables.

# 11. Architecture Design

## 11.1 Server

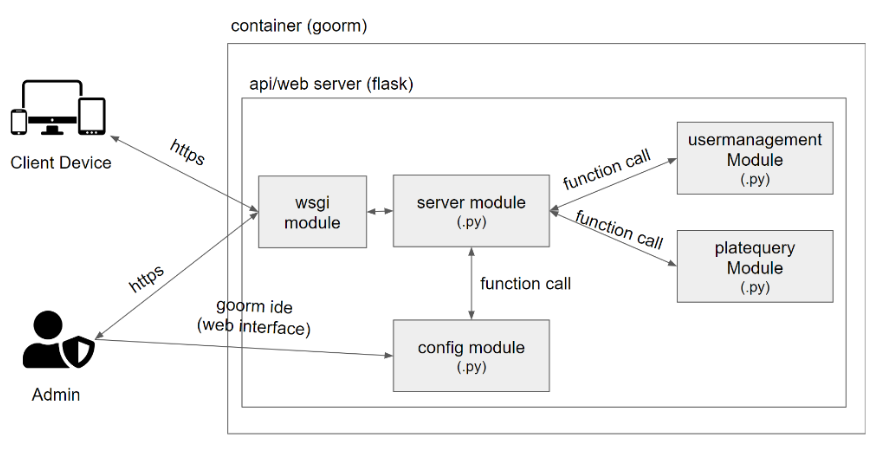


## 11.2 Client



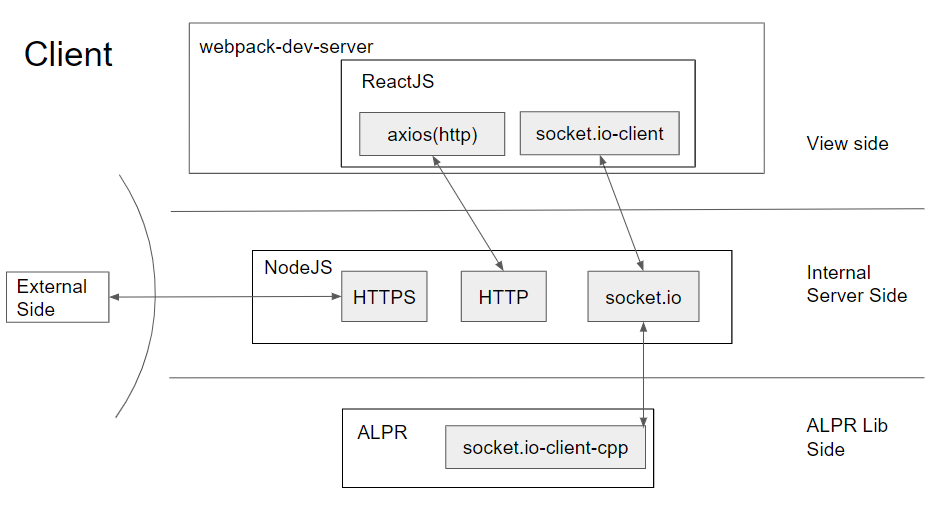
# 12. SW implementation

## 12.1 Server SW structure



## 

## 12.2 Client SW structure



# 13. SW vulnerability evaluation

## 13.1 open source scanning

OWASP Dependency Check result is below.

**Project: ALPR Project**

Scan Information (show less):

* *dependency-check version*: 7.1.1
* *Report Generated On*: Tue, 5 Jul 2022 12:27:13 +0900
* *Dependencies Scanned*: 40 (39 unique)
* *Vulnerable Dependencies*: 0
* *Vulnerabilities Found*: 0
* *Vulnerabilities Suppressed*: 0
* *NVD CVE Checked*: 2022-07-05T12:27:06
* *NVD CVE Modified*: 2022-07-05T09:00:01
* *VersionCheckOn*: 2022-06-16T08:22:01

**Analysis Exceptions**

Unable to read yarn audit output.

**Summary**

Display: Showing All Dependencies (click to show less)

| **Dependency** | **Vulnerability IDs** | **Package** | **Highest Severity** | **CVE Count** | **Confidence** | **Evidence Count** |
| --- | --- | --- | --- | --- | --- | --- |
| App.js |  |  |  | 0 |  | 0 |
| App.test.js |  |  |  | 0 |  | 0 |
| Newtonsoft.Json:8.0.3 |  | [pkg:nuget/Newtonsoft.Json@8.0.3](https://ossindex.sonatype.org/component/pkg:nuget/Newtonsoft.Json@8.0.3?utm_source=dependency-check&utm_medium=integration&utm_content=7.1.1) |  | 0 |  | 4 |
| OpenCVDeviceEnumerator.exe |  |  |  | 0 |  | 2 |
| TimeMem-1.0.exe | cpe:2.3:a:time\_project:time:1.0:\*:\*:\*:\*:\*:\*:\* |  |  | 0 | Low | 4 |
| alpr.exe |  |  |  | 0 |  | 2 |
| country\_info.js |  |  |  | 0 |  | 0 |
| dashboard-layout.js |  |  |  | 0 |  | 0 |
| display\_config.js |  |  |  | 0 |  | 0 |
| illegal\_plate.js |  |  |  | 0 |  | 0 |
| index.js |  |  |  | 0 |  | 0 |
| index.js |  |  |  | 0 |  | 0 |
| input\_source.js |  |  |  | 0 |  | 0 |
| libdb181.dll |  |  |  | 0 |  | 4 |
| liblept-DLL.dll |  |  |  | 0 |  | 2 |
| login.js |  |  |  | 0 |  | 0 |
| main.js |  |  |  | 0 |  | 0 |
| notfound.js |  |  |  | 0 |  | 0 |
| openalpr-opencv4.zip: AlprNet.csproj |  |  |  | 0 |  | 2 |
| openalpr-opencv4.zip: AlprNetGuiTest.csproj |  |  |  | 0 |  | 2 |
| openalpr-opencv4.zip: AlprNetTest.csproj |  |  |  | 0 |  | 2 |
| openalpr-opencv4.zip: packages.config |  |  |  | 0 |  | 0 |
| opencv\_videoio\_ffmpeg455\_64.dll |  |  |  | 0 |  | 2 |
| opencv\_videoio\_msmf455\_64.dll |  |  |  | 0 |  | 2 |
| opencv\_videoio\_msmf455\_64d.dll |  |  |  | 0 |  | 2 |
| opencv\_world455.dll | [cpe:2.3:a:opencv:opencv:455:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aopencv&cpe_product=cpe%3A%2F%3Aopencv%3Aopencv&cpe_version=cpe%3A%2F%3Aopencv%3Aopencv%3A455) |  |  | 0 | High | 4 |
| opencv\_world455d.zip: opencv\_world455d.dll | [cpe:2.3:a:opencv:opencv:455:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aopencv&cpe_product=cpe%3A%2F%3Aopencv%3Aopencv&cpe_version=cpe%3A%2F%3Aopencv%3Aopencv%3A455) |  |  | 0 | High | 4 |
| package.json |  |  |  | 0 |  | 0 |
| package.json |  |  |  | 0 |  | 0 |
| plate\_image.js |  |  |  | 0 |  | 0 |
| plate\_list.js |  |  |  | 0 |  | 0 |
| qrious.min.js |  |  |  | 0 |  | 0 |
| save\_option.js |  |  |  | 0 |  | 0 |
| server.js |  |  |  | 0 |  | 0 |
| setupProxy.js |  |  |  | 0 |  | 0 |
| severity-pill.js |  |  |  | 0 |  | 0 |
| ssl-config.js |  |  |  | 0 |  | 0 |
| video\_view.js |  |  |  | 0 |  | 0 |
| yarn.lock |  |  |  | 0 |  | 2 |

## 13.2 secure coding rule

FlawFinder result is below.

ANALYSIS SUMMARY:

Hits = 1922

Lines analyzed = 795750 in approximately 6.74 seconds (118107 lines/second)

Physical Source Lines of Code (SLOC) = 508562

Hits@level = [0] 2024 [1] 600 [2] 987 [3] 47 [4] 287 [5] 1

Hits@level+ = [0+] 3946 [1+] 1922 [2+] 1322 [3+] 335 [4+] 288 [5+] 1

Hits/KSLOC@level+ = [0+] 7.75913 [1+] 3.77928 [2+] 2.59949 [3+] 0.65872 [4+] 0.566303 [5+] 0.00196633

Minimum risk level = 1

[Team3-FlawFinder Result](https://docs.google.com/document/d/1PPxsssufcYDiLpko4zb8XAJWBPtV4OiVGl3R-19F3Og/edit?usp=sharing)

# 14. Test

## 14.1 Test methodology

We decided to perform a requirement-based test.

Test cases were created for requirements that are system level, software level, and whose priority is not optional.

## 14.2 Test Result & Bugs

1st Run

* test case pass % : 77% (14/18)
* test case NT(Not Tested) % : 10% (2/20)
* bug : 4

2nd Run

* test case pass % : 100% (18/18)
* test case NT(Not Tested) % : 10% (2/20)
* bug : 0

NT(Not Tested) means that a case in which a test case for a requirement has been created, but is not currently testable due to the reasons of the verification environment.

You can find the details in the “Requirement based Test” and “Test Bugs” in the “Team3\_purple\_ALPR\_Test.xlsx” document in the deliverables.

# 15. Deliverables

## 15.1 Source code

Team3\_purple\_ALPR\_source code\_package

## 15.2 Test cases/Test bugs

Team3\_purple\_ALPR\_Test.xlsx

## 15.3 Threat modeling

Team3\_purple\_ALPR\_ThreatModeling.xlsx

## 15.4 Risk rating

Team3\_purple\_ALPR\_RiskRating.xlsx

## 15.5 Requirements

Team3\_purple\_ALPR\_Requirements.xlsx

## 15.6 Seeded vulnerability list

Team3\_purple\_ALPR\_Seeded\_Vulnerability\_list.xlsx

# 16. Vulnerability Reporting

If you find any vulnerabilities or have questions about our ALPR system, please email the Purple Team.

**email : kibongs.song@gmail.com**