Automatic License Plate Recognition (ALPR) System Assessment for Team2

Team3 PURPLE

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

Table of Contents

Version	Date	Status	Author
1.0	2022.07.14	1st release	team3

1. Plan

1.1 Roles & Responsibilities

- Team Lead
  - Kibong Song (송기봉)
- Vulnerability Assessment and Documentation
  - Youngmi Choi (최영미), Seungkyu Lee (이승규)
- Reverse Engineering
  - Seongsik Kim (김성식), Haenggi Lee (이행기)
- Penetration Testing
  - Kibong Song (송기봉), Gyunggui Moon (문경귀)
- Mentor
  - David Belasco (데이비드)

1.2 Communication Ground Rules

1. All team members will treat each other with respect and will always provide accurate and truthful information.
2. Meetings should start and end on time, and an agenda should be prepared in advance and distributed to all participants; stay on topic.
3. Mentor and team members share and discuss the progress through online video meetings once a week.

1.3 Communication System and tools

1. Team3 will use the google drive to share all documents with other team members at the same time.
2. Team members communicate with mentors in real time using slack.

1.4 Timeline

The expected timeline and deliverables of the assessment of team2 will be achieved over a period of approximately two weeks.

1. week1 (From July 4th to July 8th)
   a. review all artifactories of team2
   b. establish the vulnerability assessment plan
2. week2 (From July 11th to July 14th)
   a. start looking for vulnerabilities
   b. prepare the assessment report for found vulnerability

1.5 Target discovery and reconnaissance

Our target is ALPR system of team2.
We collect and review publicly available information about the target. Such information can be found in Team2's github(https://github.com/hyecahn/2022-security-specialist-team2).

1.6 Tools & Techniques

1. reverse engineering techniques
    a. Runtime analysis plan to perform using the tool to monitor the network activity between the server and the client.
        i. Wireshark
    b. Runtime analysis plan to perform using the tool to analyze a encrypted DB
        i. db_dump
        ii. Hex View
    c. Since the source code of the other team has been provided, static analysis will be performed through code review.
        i. We plan to focus on logic error, Insecure configuration, cryptographic vulnerability and Input validation error
2. penetration testing techniques
    a. We plan to exploit the client app uses popular open source software
        i. OWASP ZAP: target to Django
        ii. Metasploitable in Kali: target to opencv-python
    b. We plan to use input validation for various inputs.
3. Research
    a. We plan to research some specific DB since the client app uses a specific version of open source software.
        i. https://www.exploit-db.com/
        ii. https://www.cvedetails.com/
4. Fuzz testing
    a. We plan to use fuzzing tools to mutate the sample JPEG images of license plates
        i. Dumb fuzzing with ZZUF
        ii. JPEG fuzzer https://github.com/h0mbre/Fuzzing
5. vulnerability evaluation
    a. We select CVSS as the vulnerability evaluation model.

1.7 Effort Allocation

In order to produce meaningful results within limited resources, it was decided to analyze the areas with high threat risk first.

From the attacker's point of view, the security goal and threat modeling results provided by team2 were first reviewed.
Based on the risk assessment provided by Team 2, efforts for critical/high-grade requirements were first determined to analyze the security flaws of high-risk requirements.

SR-01, critical : The system shall allow an officer to access the ALPR system through a secure web interface by two factor authentication.
- secure web interface (https, certification by CA)
- two factor authentication

SR-03, high : The system shall grant the admin user to access and modify configuration files
- access and modify the configuration file

SR-04, high : The Plate DB must be encrypted data
- encrypted plates DB

2. Assessment

2.1 Executive Summary

Assessment target : ALPR system implemented by team2.

The ALPR system was analyzed using techniques and tools discussed in the planning stage, and eight security vulnerabilities were found.

Priority	Vulnerability	Impact	Recommendation
High (CVSS 8.7)	(VID-1) If we can set request.session.error, then we have access to add a new user in the same privilege as super user.	An attacker can compromise user credential information.	Fix improper error handling; Apply least privileged access control
High (CVSS 7.5)	(VID-4) The key value used for DB encryption in the Lookup server is the ASCII string '2Team_AhnLab'; it's hard-coded and predictable.	An attacker can disclose personal identifiable information.	Remove the hard-coded key value; Change to a stronger one and store it in a secure place
High (CVSS 7.9)	(VID-8) 0 can be entered as the maximum user setting value in 'server.conf'. This causes the server to either return an internal error or hang.	An attacker can tamper configuration information to stop the service.	Check setting input value whether it is within the normal range before use
Medium (CVSS 6.5)	(VID-2) If the user id is '..' then a video or image file cannot be uploaded.	Path traversal allows unauthorized users to access files.	Recommend that you use user ID as user folder path.
High (CVSS 7.3)	(VID-3) If you rename the sample video file to a long name and upload it, an internal error occurs the moment the license plate is recognized.	An attacker can trigger malfunction of various components by providing malformed data.	recommend limiting the file length in the upload function.
High (CVSS 7.6)	(VID-5) If you upload an image file with comments as 'Ahnlab', it gives a	An attacker can trigger malfunction of various components by providing malformed	Remove unnecessary code through code review

	shutdown command to the server.	data.	
High (CVSS 7.6)	(VID-6) It is possible to delete files uploaded by another user.	Uploaded contents can be deleted by anyone who logged in the system.	Apply access control so that users can only control their own data
Medium (CVSS 5.8)	(VID-7) Using the user's session id, you can login without password or 2 factor authentication.	An attacker can use another user's session ID to log in and use the system without a separate authentication process.	Set session validity period for user authentication In case of django, the session validity period can be set through the COOKIE_AGE value (default : 2 weeks)

The found vulnerability was able to damage 9 of the 10 assets identified by team2.

By exploiting the vulnerability of VID-01,The attacker can use the system like a regular user by adding multiple unauthorized IDs and passwords. In this way, the attacker can modulate the information(ID, Password) in the user DB.

- AID-06 : ID
- AID-07 : Password

By exploiting the vulnerability of VID-04, all information of the plate DB has been discovered and DB tampering is possible, so damage can be applied to five assets identified by team2.

- AID-01 : License Plate Number
- AID-02 : Status
- AID-03 : Owner Name
- AID-04 : Owner Date of Birth
- AID-05 : Owner Street Address/ Location
- AID-10 :Owner City, State and Zip Code

By exploiting the vulnerability of VID-08, It is possible to stop the system by changing the configuration value that affects the system operation.

- AID-08 : Configuration File

In conclusion, we confirmed the results of compromising the declared security goal.

2.2 Executive Constraint

- validation of Certificates is excluded from the analysis.
- The analysis of the simultaneous connection of multi-user was excluded as an issue of the executable environment.

2.3 Evaluation Narrative

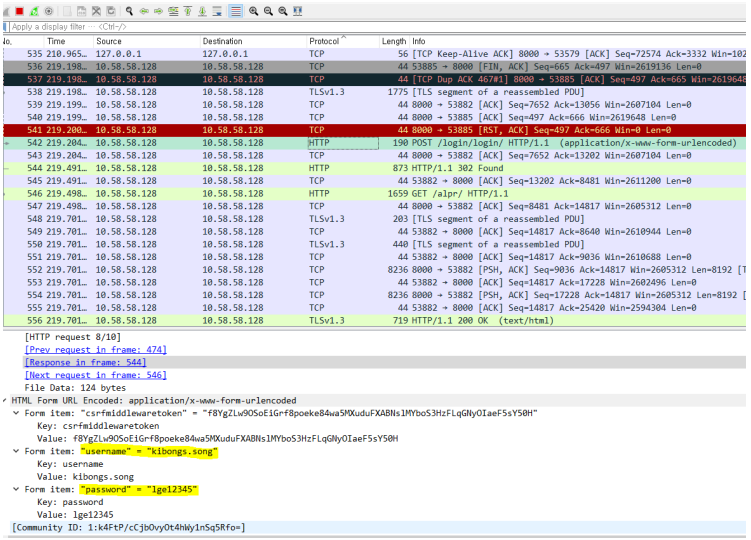
The critical/high risks of the threats derived at the time of development were analyzed first.

1. **secure web interface between client and server**
    a. runtime analysis
        i. techniques & tool : wireshark
        ii. rationale : Commonly used wireshark is used to verify TLS application and packet verification between networks

iii. activities :
    1. Capture packets with wireshark while running server and client
    2. Check client and server communication packets
iv. result : TLS was applied.
v. found vulnerability ID : n/a

2. two factor authentication

a. runtime analysis
i. techniques & tool : wireshark
ii. rationale : Commonly used wireshark is used to verify TLS application and packet verification between networks
iii. activities :
    1. Capture packets with wireshark while running server and client
    2. Check client and server communication packets
iv. result : User ID/Password exposed in login packet transmission (POST) between web interface and client. But, we can't prove how to check other users' network packets, so we excluded it from the vulnerability.



v. found vulnerability ID : n/a

b. runtime analysis
i. techniques & tool : cookies
ii. rationale : Authenticates users through sessions. Therefore, to check the cookie information to see if the mitigation for the weak point of the session is applied.
iii. activities :
    1. multi user login
    2. Check session id value for each user through developer tools in chrome
    3. Enter other user session id values in cookies to access the webpage
iv. result : Can use another user's session ID to log in and use the system without a separate authentication process
v. found vulnerability ID : VID-7

3. access and modify the configuration file

a. static analysis
i. techniques & tool : Code Review
ii. rationale :
    1. Use CodeReview to check input validation
iii. activities :
    1. Conduct a code review on the part of the server source code that handles the change in the value of the configuration file “server.conf”.

2. Verify that a value is assigned to a variable without validation of the value entered by the user
3. Check if UI processes invalid input
4. Check the range of inputs in the UI, but make sure that not all invalids are checked
5. Click the update button without input values(MaxUser, Confidence Level)
6. An internal server error(500) occurs

iv. result :
1. the MaxUser value is set to 0 in “server.conf” file and
2. The ALPR system will not work until the server operator notices this and changes it to the correct value.

v. found vulnerability : VID-8

b. runtime analysis
i. techniques & tool : DB Tool (DB Browser for sql)
ii. rationale : As a way to classify admin accounts in Phase.1, a flag for classifying admins is added to the user db table. Assuming that other teams have done the same, searched the db table.
iii. activities :
1. Check whether user DB file password is applied
2. Check whether user DB data encryption is applied
iv. result : User DB file was not encrypted, and Admin flag among user db data is not encrypted. Can acquire Admin privileges by changing the Admin flag. but, we can't prove how to access the DB, so we excluded it from the vulnerability
v. found vulnerability ID : n/a

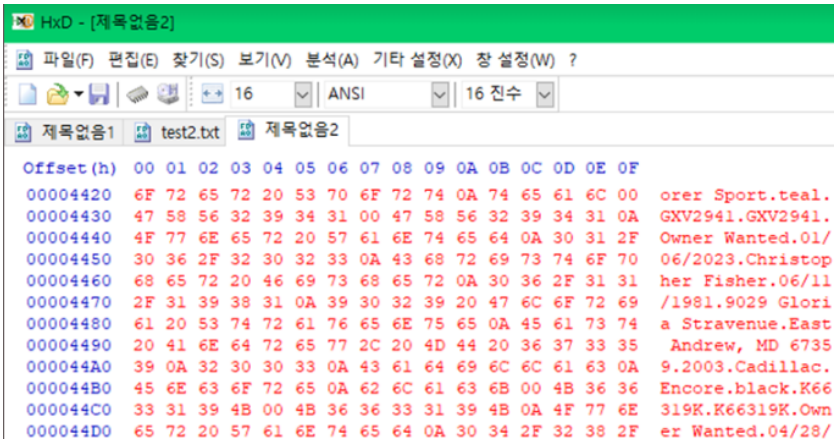
4. encrypted plates DB

a. static analysis
i. techniques & tool : code review, IDA, db_dump, Hex View
ii. rationale :
1. Use IDA to determine the key value used for encryption
2. Use db_dump, Hex View for DB lookup and data parsing
iii. activities :
1. Open licenseplate.db to verify that it is encrypted
2. Decompile Server.exe to IDA to find out the key
3. Found out key is "2Team An Lab"

```
86  else if ( (*(unsigned int (__fastcall **)(__int64, int *))(v57 + 824))(v57, v66)
87          || (v66[0] = 1, (*(unsigned int (__fastcall **)(__int64, const char *, __int64))(v57
88                                                                                            v57,
89                                                                                            code,
90                                                                                            1164)) )
91  {
92    printf("DB Encrypt Error\n");
93    result = 0xFFFFFFFFi64;
94  }
```

```
.rdata:000000014000DB30 ; const char code[12]
.rdata:000000014000DB30 code            db '2', 'T', 'e', 'a', 'm', '_', 'A', 'h', 'n', 'L', 'a'
.rdata:000000014000DB30                                         ; DATA XREF: ProcessClient(void *)+B9↑o
.rdata:000000014000DB30                 db 'b'
.rdata:000000014000DB3C                 align 20h
```

4. After db_dump (decrypt with key value input) for DB decoding and query, plate DB information is found with HexView



iv. result : All personal information in the plate DB was leaked by decrypting the encrypted DB.

v. found vulnerability ID : VID-4

After that, the part randomly selected by our team was analyzed.

1. plate image/video recognition and processing

a. static analysis
   i. techniques & tool : Code Review
   ii. rationale : Use CodeReview to identify vulnerabilities in ALPR system code
   iii. activities :
      1. Make a small image with width and height less than or equal to 1
      2. Upload it
      3. Then we have access to add a new user in the same privilege as admin
   iv. result :
      1. The ‘Create’ button is displayed when uploading an image
   v. found vulnerability ID : VID-1

b. static analysis
   i. techniques & tool : Code Review
   ii. rationale : Use CodeReview to identify vulnerabilities in client code
   iii. activities :
      1. Make a small image with width and height less than or equal to 1
      2. Upload it
      3. Then we have access to add a new user in the same privilege as admin
      4. Add a new user - ID is '..'
      5. Try to upload a video or image file
      6. Get an error from Django framework
   iv. result : If the user id is '..' then a video or image file cannot be uploaded
   v. found vulnerability ID : VID-2

c. static analysis
   i. techniques & tool : Code Review
   ii. rationale : Use CodeReview to identify vulnerabilities in ALPR system code
   iii. activities :
      1. Check if there is any code that can force the lookup server to shutdown on server-side
      2. Check if there is any code that sends the ‘shutdown’ command on the client-side
      3. Send shutdown message when the comment of image is the same as app_name(in this case: Ahnlab)
   iv. result :
      1. Add comment to image using ‘imagemagick’ library

2. The lookup server is shutdown when uploading the image created in step 1

v. **found vulnerability ID : VID-5**

d. static analysis

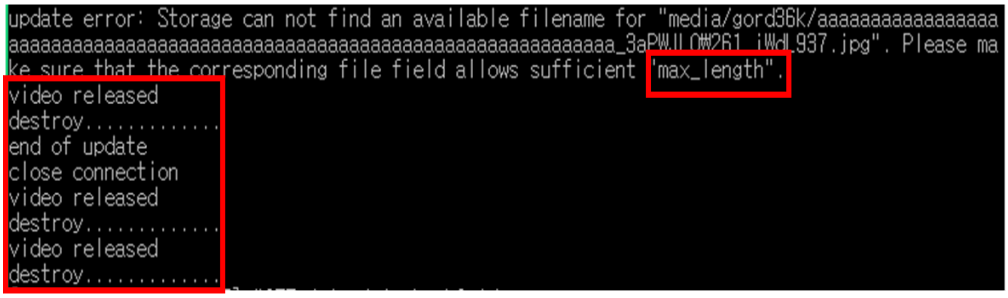
i. techniques & tool : Code Review

ii. rationale : Use CodeReview to check input validation

iii. activities :

1. Use code review about File upload and how to call the ALPR program with the arguments.
2. Check no exception routine about length of file name.
3. Upload aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaa.avi file to the server.
4. There are some warning messages and changed file name by force.

5. Play the file and then, happen certain exceptions & stop when to recognize the first plate. The system tries to store isolated plate image. Total path is very long so, It is crashed.



iv. result : There is no input validation about upload file type and length.

v. **found vulnerability ID : VID-3**

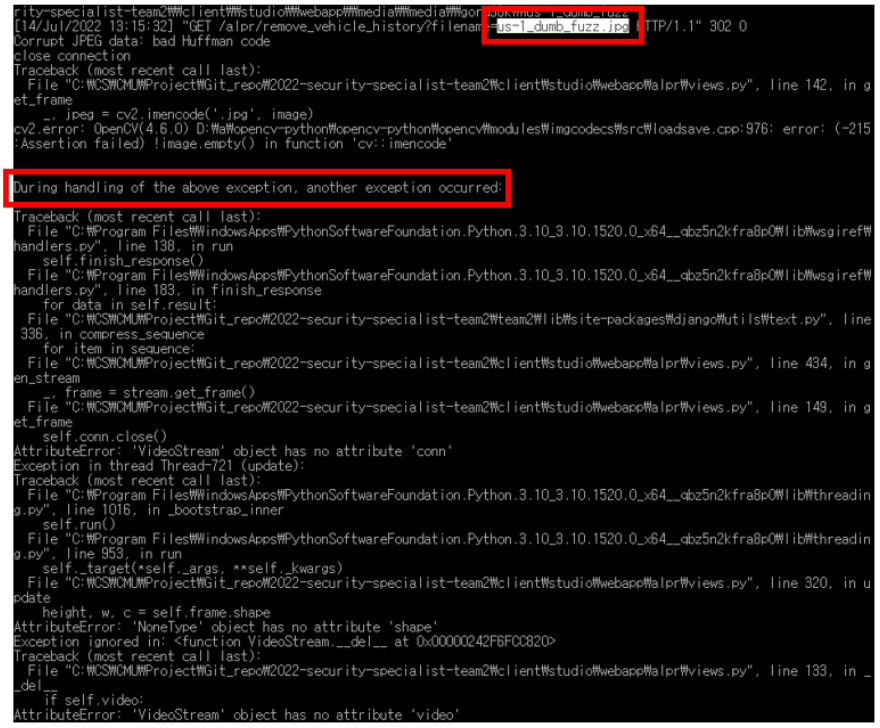
e. Fuzz testing

i. techniques & tool : zzuf, JPEG fuzzer

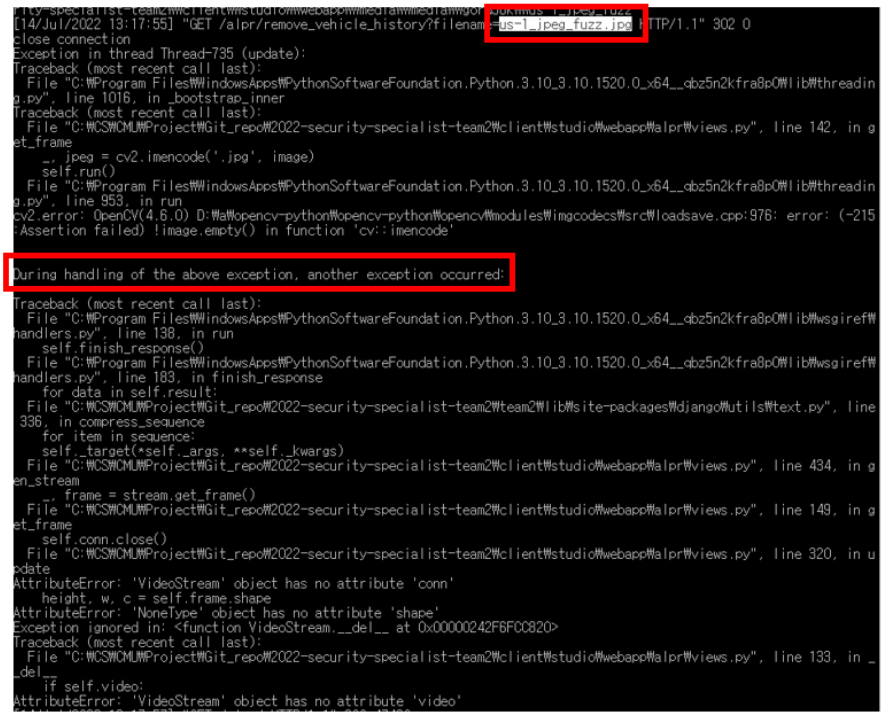
ii. rationale : Users can input various images into the OpenALPR library. Professor Jeff recommended the application of fuzzing when evaluating projects.

iii. activities :

1. Dumb fuzzing: Mutate a sample license plate image(us-1.jpg) using zzuf and upload it.



2. JPEG fuzzing: Mutate a sample license plate image(us-1.jpg) using a tool which subtly alters valid JPEGs and upload it
   JPEG fuzzer: https://github.com/h0mbre/Fuzzing

3. 

iv. result : Just caught in opencv library assertions upon both fuzzing
v. found vulnerability ID : n/a

2. Django

a. runtime analysis

i. techniques & tool : General browsers
ii. rationale : for sending GET request
iii. activities : file delete GET requests were made from result of ZAP active scan, then send them through the browser. e.g. https://ahnlab2.lge.com:8000/alpr/remove?id=xx
iv. result : Contents uploaded by another user could be deleted. If deletion failed due to absence of target, Server error (500) returned.
v. found vulnerability ID : VID-6

b. vulnerability scanning
i. techniques & tool : OWASP ZAP
ii. rationale : ZAP provides “spider scan” as crawling and “active scan” which tries to find out known vulnerabilities.
iii. activities : run “active scan” for ahnlab2.lge.com:8080
iv. result : Can get directory list provided by server
v. found vulnerability ID : n/a

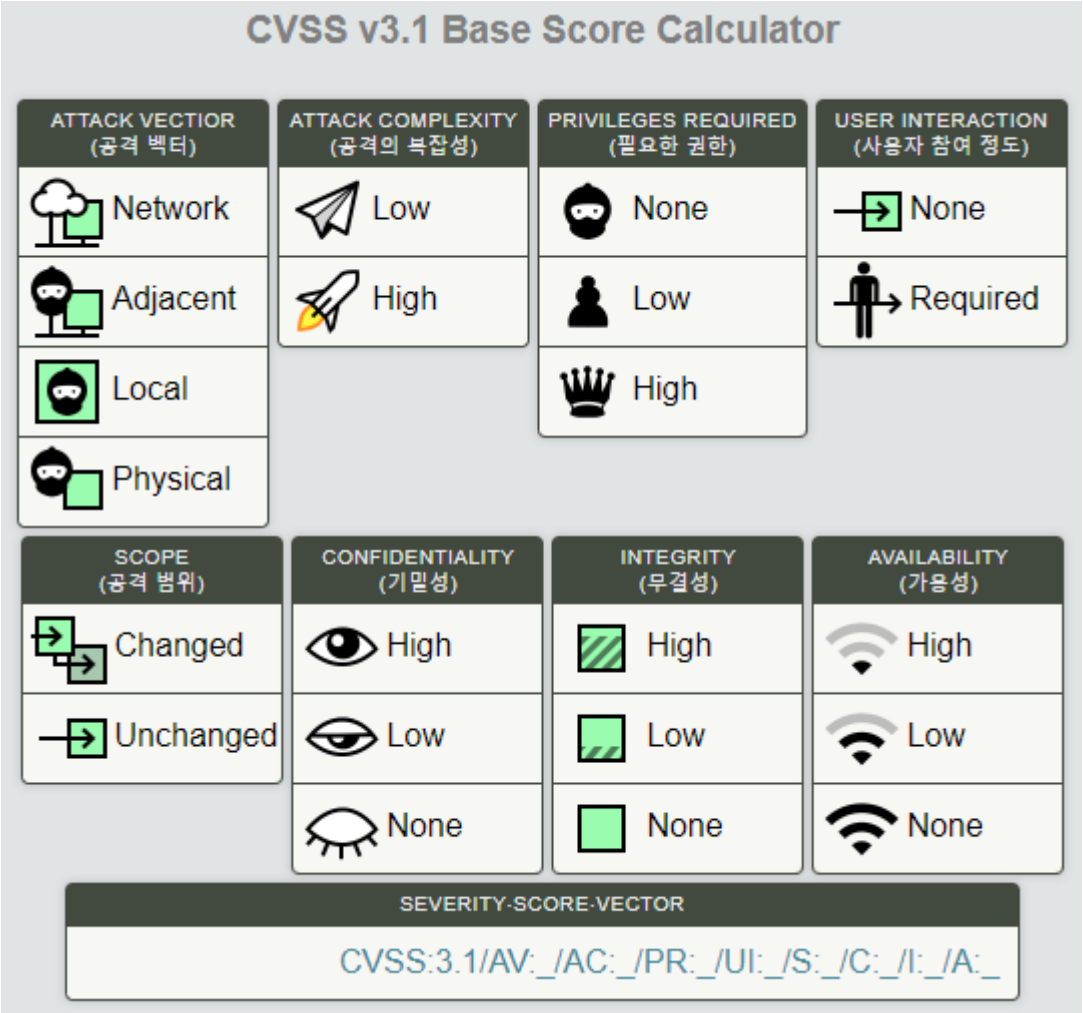
c. research Exploit DB and Vulnerability DB
i. techniques & tool : research
ii. rationale : The client app uses a specific version of open source software.
iii. activities :
1. https://www.exploit-db.com/ : target to Django 4.0.5 and opencv-python 4.6.0.66
2. https://www.cvedetails.com/ : target to Django 4.0.5 and opencv-python 4.6.0.66
iv. result : Known vulnerabilities are already patched to Django 4.0.5 and opencv-python 4.6.0.66 E(they are latest)
v. found vulnerability ID : n/a

3. opencv-python
a. penetration testing
i. techniques & tool : Metasploitable in Kali
ii. rationale : The client app uses popular open source software.
iii. activities : Search the metasploit database for an exploit in opencv-python
iv. result : No search result returned.
v. found vulnerability ID : n/a

2.4 Evaluation Result

In conclusion, We found eight vulnerabilities and evaluated them with CVSS scoring methodology.

We used a tool that automatically calculates scores.
https://itschool-info-lab.github.io/cvss/



You can find the found vulnerabilities in detail on the "Team3_purple_ALPR_Phase.2_Assessment_for_Team2_Found_Vulnerability_list.pdf".

3. Retrospect

While developing the project in phase 1, We tried to apply what I learned in class to the project if possible, but when we started implementation, we were more focused on implementing the function. It was difficult to maintain a security mindset throughout the development lifecycle.
If I do this task again in the future, I would like to try injecting various vulnerabilities by further analyzing the attack surface.
In phase 2, we tried to analyze and approach them using the artifactory provided by other teams. However, under the pressure to discover security flaws in a short period of time, it was difficult to consider the various tools and methodologies learned in the class, and it seemed to focus only on reverse engineering with provided source codes.

Below is a short comment from the team3 members at the end of the CMU class.

- It is difficult to tell security relevant failures from other functional failures. (Haenggi, Brian)
- It requires more experience to apply the attack techniques we have practiced in the course. (Seungkyu, Kibong)
- Choosing secure OSS packages reduces the risks from unintentional vulnerabilities. (Seongsik, Haenggi)
- Threat modeling is not only a set of technical activities, but also an opportunity to collaborate to build more secure systems. (Youngmi, Brian)

4. Deliverables

4.1 Found vulnerability list

Team3_purple_ALPR_Phase.2_Assessment_for_Team2_Found_Vulnerability_list.pdf