# Automatic License Plate Recognition (ALPR) System

Assume that you and your team are working for Tartan Inc. You're readying a new remote camera application designed to support law enforcement. Specifically, you are working on a project to develop an Automated License Plate Recognition Management System, known as Tartan ALPR. The company wants to release the software as soon as possible, but there are concerns that the new software system is insecure, hard to use, and not fast enough. Before the software is released it must be completed using rigorous secure software development standards and evaluation.

## Learning Goals

We will use the Tartan ALPR application as a context for our project. Please note that the purpose of this project is not to make you an expert in image recognition web-based systems, or mobile development, but rather the context is being used as a model problem for you to practice software security techniques and methods presented in the course. Software development is necessary to fully explore these concepts, but programming is not the main focus. There are many dimensions to this project; the major themes include the following:

- Learning how to implement secure software development practices in the context of a team software project.
- Learning how to evaluate the security of an existing software system that you did not develop.
- Developing a plan to manage security on a software project.
- Gaining experience using industry standard software security tools during and after development.
- Learn to use data to validate decisions on a project.

There will be two phases for the project: development and assessment. We will require teams to inject vulnerabilities during development and award points for finding or not finding known issues. The details of the phases and scoring are described below.

The first phase of the project must be completed by the end of the third week of the course. The evaluation phase of the project will be carried out over the final two weeks of the course. Details of requirements for each phase are listed below.

## Phase 1: Development

Your team will develop a secure implementation of the ALPR system. Specifically, you must design and develop a system to meet the following notional requirements. 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.

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. The client application should communicate securely with a backend server that contains relevant information. The client application has the following basic requirements:

1. The system shall allow an officer to access the ALPR system through a secure web interface.
2. 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.
3. Lost or compromised credentials must be handled in a reasonable way.
4. The system should allow a law enforcement officer to select and save retrieved information locally.
5. 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.
6. The system should provide secure communication between the client application and to the backend license plate database lookup system.
7. The system should read images from the vehicle camera or a playback file and identify license plates for evaluation.
8. The system should perform the ALPR function in real-time while maintaining a frame rate of at least 25fps.
9. 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.
10. 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.
11. The system should provide an area in the user interface that always contains the current camera /playback view.
12. The system should allow officers to configure computed camera / playback frames per second, average time per frame, jitter and frame number.
13. The system should allow the officer to choose between using a live camera and playback file in the UI.
14. The ability to detect network connectivity issues with the backend server within 5 seconds and automatically resolve the communication issue if possible.
15. The system should alert officers of any communication errors or failures.
16. The system must fetch vehicle information in no more than 10 seconds as officers are often making queries in real time.

The server application shall provide the following functionality

1. Support license plate queries.
2. Ensure secure communication with the client applications.
3. Authenticate remote laptop users.
4. Support multiple users.
5. 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.
6. Track the average number of queries per second for each user and overall queries per second, for all users.
7. Track the number partial matches and no matches for each user and all users
8. Support configurable values via a configuration file.

The vehicle registration database on the server contains the following information for each vehicle:

| |
|---|
| License Plate Number |
| Status - Owner Wanted / Stolen / Unpaid Fines – Tow / No Wants/Warrants |
| Registration Expiration |
| Owner Name |
| Owner Date of Birth |
| Owner Street Address / Location |
| Owner City, State and Zip Code |
| Vehicle Year of Manufacture |
| Vehicle Make |
| Vehicle Model |
| Vehicle Color |

*Table 1: Vehicle Registration Database*

Aside from these requirements, there are a number of basic quality concerns that must be addressed during development.

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.

Deliverables
The following deliverables must be submitted as part of Phase 1

**Source Code, Test Cases, and Supporting Artifacts**
Your team must assemble a number of artifacts for delivery. Specifically, you must include all source code and for the main system and test cases for evaluation.

**Developer Artifacts and Documentation**
You must provide a developer guide to enable assessment of your solution. **The developer guide should allow an evaluation team to build and run your system.** Be sure to document major design decisions, especially if they impact security. Be sure to include any and all data collected during development. For example, open defects, external dependencies, and any shortcuts of technical debt incurred. You should also include all non-code artifacts in your delivery, such as requirements and design documentation.

**Vulnerability Reporting**
You must designate a point of contact to accept vulnerability reports and answer questions about the developed system. You must do this in accordance with best practices. You must make your team reporting policies available so evaluators know how to report problems.

**Seeded Vulnerability List**
You should provide a list of injected vulnerabilities only to your mentor and the program faculty (as a separate document). The list must include the following information about each vulnerability:

- How to locate it in the system
- The consequences of exploitation
- A simple proof of concept for exploitation (such as code snippet or sequence of steps).

Failure to provide this information and demonstrate that the seeded flaws are real flaws will result in a loss of points.

Presentation
At the end of phase 1 each team will present an overview of their application to bootstrap phase 2 of the project. Details on the presentation will be posted during the class.

## Phase 2: Evaluation
To ensure that no security concerns were overlooked, we will exchange projects to conduct a rigorous security evaluation of the created artifacts. The goal of this evaluation is to identify any and all vulnerabilities in the completed artifacts in such a way that you can make security recommendations to the original development team. This is a completely different project than in phase one. Accordingly, you will need to reorganize your team to accomplish the task. Your new team organization must be presented to your mentor as soon as phase two of the project begins.

Deliverables
The primary deliverable for this project is a security assessment report. The report must include the following sections:

1. Executive Summary: This summary should summarize the major results of your evaluation.
2. Evaluation Constraints: What, if any, mitigating factors were considered when deciding how to perform the evaluation.
3. Evaluation Narrative: This section should describe in detail the security evaluation activities performed in an organized and understandable way. Specifically, how did you:
    a. Decide which elements to evaluate
    b. Select security evaluation techniques; What techniques were selected and why?
    c. Verify results and prioritize found issues.

    This section should leave the reader with a clear sense of how you approached security evaluation.

4. Results, Conclusion, and Recommendations: The evaluation results and all supporting data are described in actionable terms. You should also describe a set of recommendations for the development team to improve security.

## Vulnerability Injection & Scoring

During the first phase of the project the developing team must purposefully inject **10 security flaws** into their system. The flaws can be design or implementation issues but each bug must have a clear security consequence (see below). These 10 flaws should be kept secret by the developing team.

At the end of the course the developing team and the assessing team will be judged on a point system according to the following rules.

- Each seeded vulnerability found during the security evaluation of the project will be worth 10 points to the team evaluating the system.
- Each seeded vulnerability not found during the security evaluation will be worth 10 points to the team that developed the system.
- If the mentors decide that a seeded vulnerability is not valid, then the developing team will be penalized 10 points (per vulnerability).
- If the evaluating team finds vulnerabilities that were not seeded by the developing team, but are judged by faculty to be actual vulnerabilities, then they will receive 10 points (per vulnerability).

At the end of course we will compute total scores and declare a project champion. In the event of a tie, the course mentors will vote on the champion from the teams involved in the tie.

## Vulnerability Information Table

The list of seeded flaws should be kept secret by the developing team and shared only with the team mentors via Canvas. Specifically, your team must complete and submit the **vulnerability information table** (shown below). Similarly, the evaluating team must fill out the same vulnerability information table for each found vulnerability to receive points for found problems. Each vulnerability found must include the following information:

| **Vuln. Num.** | **Summary** | **Location** | **Consequences/ Impact** | **CVSS Score** | **Proof of Concept** |
|---|---|---|---|---|---|
| *1..10* | *A brief description of the vulnerability.* | *Information needed to locate the vulnerability in the system, such as components, file(s), and lines of code.* | *What are the consequences of exploitation? For example, "Remote, unauthenticated code execution" and conduct "denial-of-service".* | *Use CVSS version 3[1] to compute the score.* | *Steps to trigger the vulnerability, either written in prose or as code to demonstrate exploitation.* |

*Table 2: Vulnerability Information Table*

[1] https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

The seeded and found vulnerabilities must be valid and exploitable. That is if it turns out that the seeded issue is not really a threat or there is no way for the assessing team to actually trigger the vulnerability, then the developing team will forfeit the points for that vulnerability.

### Vulnerability Information Table

You must complete and submit the vulnerability information table for all found vulnerabilities discovered during evaluation.

### Phase 2 Presentation

At the end of the class teams will present their assessment in terms of what security-related issues were found.

## Project Context

Law enforcement today frequently utilize Automated License Plate Recognition (ALPR) technologies to enhance their enforcement abilities, expand data collection, and to provide the capability for law enforcement to automatically compare vehicle license plates against lists of stolen, wanted, and other vehicles of interest while on patrol. The goal of this project is to design a robust and secure ALPR system for use by law enforcement for the project sponsor, Tartan Inc., that will launch this new product later this year. Key motivators for this project are:

- Select the best architecture/solution based on the results of this competition among the teams in this course.
- Ensure secure system communication with the backend server that will be accessed from the police vehicle via a 5G cellular network
- Fast and accurate license plate lookup and response times
- User interface that is intuitive that requires minimal human interaction.

### ALPR Overview

As vehicles pass through the ALPR camera field of view a picture is taken and a series of algorithms are performed on the image. See Figure 1. First the image is evaluated to determine if a plate is visible, and if so, the plate is then isolated from the image and the alphanumeric characters are converted into a computer readable format. Generally, there are a set of basic algorithms that the software performs to identify and read a license plate and the accuracy of the system is typically driven by the sophistication and complexity of each of these algorithms.

1. Detection - Finds potential license plate regions for further processing
2. Character Analysis - Finds character-sized "blobs" in the plate region
3. Plate Edges - Finds the edges/shape of the license plate
4. Deskew - Transforms the perspective to a straight-on camera view based on the typical license plate sizes
5. Character Segmentation – isolates characters so that they can be processed individually
6. Optical Character Recognition - Analyzes each character image and provides possible letters/confidences
7. Syntactical/Geometrical analysis – Check characters and positions against state-specific formats

In this project, each team will create a system to implement the software for an ALPR system that utilizes a laptop that will be located in the police vehicle and communicate via a network to a remote backend license plate database that records the vehicle owner, make, model, color and status.

All teams will interact and communicate their client related questions or concerns with a single Tartan Inc. point of contact.

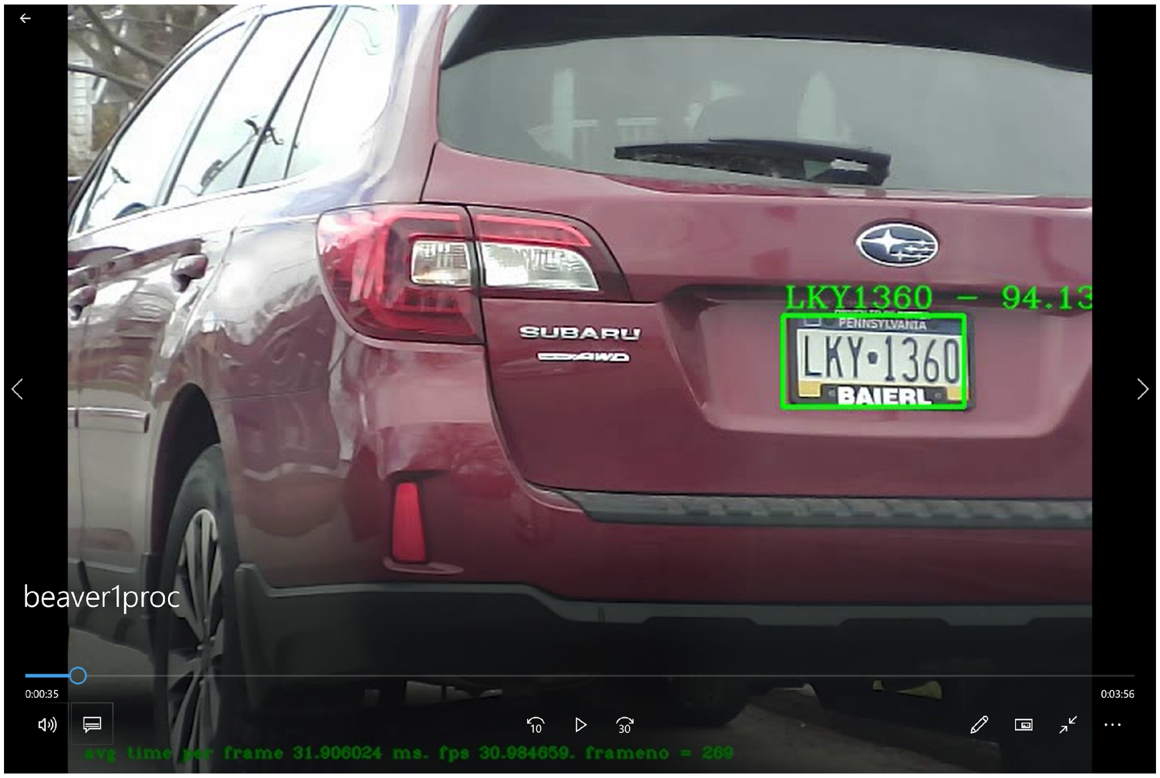

Figure 1: License Plate Recognition

## System Hardware

Teams will use their laptop to develop and execute both the user application and the backend license plate database server. Each application will be run on separate laptops and communicate via Wi-Fi.

## Required Software

Software will be developed using Visual Studio 2022 Community Edition running on Microsoft Windows 10/11 with OpenCV 4.5.5.

## Sample Code Archive

An example ALPR C++ application that uses OpenALPR (with all source code) is provided as shown in Figure 1. Please note this example application should NOT be considered an exemplar architecture or code. It merely provides a means for demonstrating how some basic features could be implemented and should not be used as an architecture to build upon or extend. Some example code may also not be implemented in the best or most efficient way, may not be complete.

## Assumptions and Hints

- The team may use any third-party software, but the choice of technology must be justified.
- Any programming language can be used.
- Assume that anything can fail… your design should account for failure and recovery to the greatest extent possible.
- Teams will use their own laptops for software development and to execute the user application and server software.