Transport Layer Security (TLS) Implementation in Driver Monitoring Systems

Gede Prasidha Bhawarnawa - 13520004
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13520004@std.stei.itb.ac.id

***Abstract*—This paper aims to discuss the usage of transport layer security (TLS) in driver monitoring systems (DMS). The goal is to validate the identity of the system that calls the API that is used within the system between the subsystems. The algorithm proposed by the researcher is by using OpenSSL certificates and running the application with the SSL keyfile and certfile in uvicorn library. The result is a secure HTTPS for communication.**

Keywords—driver monitoring systems, OpenSSL, TLS, uvicorn, FastAPI.

I. Introduction

Some industries in the world have listed transportation as one of the core activities and integral to the businesses' general wellbeing. This is generally caused when the means of production and the means to process the core materials are distanced quite far away. The speed to move materials from the source or from semi-finished materials processing sites can very well be the determining factor in the business' success. One such example of a business that fits the notion is the mining industry.

Figure 1 Trucks in a Mining Area
(Source: ABC Net News, https://www.abc.net.au/news/2019-10-25/truck-driver-death-caused-by-mechanical-failure-report-says/11639992)

In the mining industry, trucks follow a relatively constant path every day. They go down to the excavation site to pick up the raw materials and then they go back uphill to deliver said raw materials for processing. It's a very long and challenging downhill and hike for the driver, due to the nature of the course, the environmental hazards themselves, and the state of their trucks and themselves. They may find it more difficult to control the truck when they're tired, when they're distracted, or worse, when they are exposed to the chemical hazards from working in the mining area. Any miscalculation made by anyone in the industry can cost both financial loss to the company and potential human loss to the drivers themselves. Hence, there's a need to monitor these drivers constantly.

In the beginning, driver monitoring was performed by using a peer system. Each truck would have multiple personnels operating it, so that each personnel can watch over each other. However, with this method also come some disciplinary and other safety infractions. Drivers would be caught smoking or playing with their phones or disregarding the mining company's dress code. At some point, companies would try to set up manned posts, but due to the nature of the mining track that can extend for hundreds of miles, it's impossible to check for each single infraction manually.

This is the beginning of the need for driver monitoring systems that can monitor the drivers' activities automatically.

II. Driver Monitoring System (DMS)

A driver monitoring system (DMS), or a driver attention monitor system in other terminologies, is a vehicle and driver safety assessment to assess the driver's alertness and warn the driver of unsafe behavior and/or unsafe conditions. While some terms point out that a DMS can take over the vehicle once it deemed the driver unsafe to continue driving, a DMS' application is determined case by case. In the mining example, it's safer to just alert the driver with either loud noise or light indicators compared to letting the DMS to take over the wheel, as the latter may cause worse incidents.

Figure 2 Driver Monitoring System Implementation
(Source : Lexus, https://mag.lexus.co.uk/lexus-car-safety-monitoring-systems/)

Just like the term itself, the actual implementation of a DMS can vary a lot and is really dependent on the expected usage. However, in general, a DMS consists of a hardware component and an integrated software system. The hardware component, at the very least, consists of a camera to consistently monitor the driver. It may also contain additional sensors that can help monitor the driver and vehicle's safety. Some examples include using a accelerometer to detect crashes or sudden abrupt stops, a sensor to detect poisonous gases, a sensor to detect the driver's heat signature by using infrared to detect sickness, etc.

Figure 3 ESP32 Cam for DMS
(Source: Lazada Indonesia, https://www.lazada.co.id/products/esp32-cam-esp32-cam-esp-32-camera-ov2640-wifi-module-esp32-kamera-i4996436847.html)

Other than the sensors and camera, one also integral part of a DMS is the controller or processor. Some notable examples are ESP32, Raspberry Pi, and Arduino Uno. This module is used to process the inputs from the sensors and determine whether the driver, the vehicle, or both are safe or unsafe. It's very possible to embed an image recognition machine learning model into these modules so that the module can be more accurate and explainable in determining whether the situation is safe or not.

The next important hardware module to be included is a Wi-Fi module. While this is not an absolute necessity as DMS can just be deployed locally without any connection to the cloud, in terms of the usage context within mining areas, for the supervisor to inspect the driver's activities from a safe distance without being in the same vehicle as the drivers, the DMS needs to send the data over to the cloud, which requires a Wi-Fi module. Considering that mining locations don't have the best signal connection, this module can be complemented with a signal booster module.

Last but not least, an important hardware module is an output module to notify the drivers of their infractions and unsafe behavior. This module can be either an LED display, a speaker, or just a designated LED bulb. While those modules can all work, generally speaking, the more information that can be passed on to the driver, the better. However, this may come at the expense of the driver's attention.

Other than the hardware component, another integral component is the integrated software system. At the very minimum, the software can just be an embedded computer vision model on the controller or microprocessor to detect anomalies in the driver and/or the vehicle. However, considering the context is to implement it on trucks carrying mining materials, there are two other components additional to the computer vision model: a dashboard and a communication channel/method.

A dashboard is a way for supervisors to see a visual summary of the activities of their subordinate drivers. A dashboard will show the activity logs of someone as recorded by the vehicle's DMS, their infractions, details of the infractions, and video proof for future accountability. Considering the scale of the mining areas, this will also increase the speed of a supervisor noticing an accident that involves their subordinates.

The next software component is a communication channel so that the DMS can send data retrieved from the sensors to the cloud and then to the dashboard. The communication from the cloud to the dashboard wouldn't be much of an issue as it can be performed using built-in libraries and internal IPs if deployed on the same cloud architecture. However, considering that there might be signal connection issues in the mining area, one of the possible solutions that gets implemented is that the driver is asked to provide Wi-Fi hotspot connection for the DMS connection. The problem here is that, because the driver's phone or tablet or any other mobile device is the one providing the connection, there's a possibility of the data getting changed midway when the data packet passes through the phone. This is why a reason exists to secure the connection method between the DMS and the cloud.

In English dictionary, there exists more than 171.146 words according to the Oxford English Dictionary with all sizes and length. Since this research will only focus on the Wordle game, we can filter the words to those that have a length of five letters. There are more than 158.000 five-letter words in English dictionary but Wordle does not use every word in the dictionary. Wordle has only 2.315 possible solution words and an additional 10.657 words that will not be a possible solution but still accepted as guesses.

Second thing to understand about the English vocabulary is the types of letters that are used to construct words. Commonly there are two types based on the sound that are represented: vowels and consonants. Vowel letters are “a”, “i", “u”, “e”, “o” with the remainder of the letters classified as consonants. Out of the 158.000 five-letter words in the English dictionary, there are less than a percent words that has no vowels, a total of twenty to be exact. With that being said, it is safe to asssume that vowels are almost guaranteed to exist in a secret word every time.

III. Greedy Algorithms and Implementation

Greedy algorithms are algorithms that solve problems by picking out the currently (local) best option available with the hope that it will lead to an global optimal solution. Greedy algorithms never reverses an earlier decision and only progresses forwards until the solution is reached, regardless the solution being most optimal or not.

Greedy algorithms are commonly used for optimization problems or problems that require an optimal solution. There are only two types of optimization problems: maximazation problems and minimization problems.

In a greedy algorithm, there are elements that are required for the functionality of the algorithm. In total there are six: candidate set, solution set, solution function, selection function, feasibility function, and objective function. Candidate set is a set which contents are filled with possible choices to pick at each step. Solution set contains candidates that have been chosen from previous steps. Solution function is a function that determines whether the previously mentioned solution set is enough to satisfy as a solution. Selection functions are functions that serve as a filter for the candidate set on a given step based on heuristic strategies. Feasibility function is a function that determines whether a candidate solution is feasible to be inserted into the solution set. Lastly, objective function is a function that serves as an optimization, either maximizing or minimizing the solution set.

In this research, the researcher’s final aim is to determine the best strategy to win a Wordle game. To do this, the researcher create a simulation of a Wordle game using Python programming language. To simulate the dictionary used in Wordle, the simulation uses a Python library called NLTK or Natural Language Toolkit. Using this library, the simulation can extract five-letter English words and save that list to be used later. It is important to note that some words in the NLTK library has upper-cased letters and since the simulation wants to mimic Wordle as close as possible, these words are eliminated from the list.

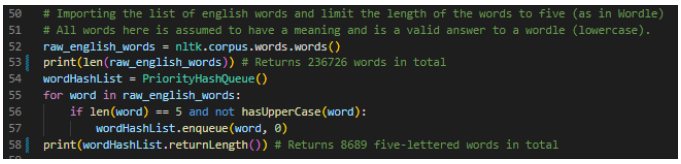

Figure 4 Importing Words to Simulation
(Source : Personal Library)

In total, there are 8.689 five-lettered words extracted from the NLTK library, which is almost four times the amount of possible solutions in Wordle dictionary. As a result, it may need more than six guesses to find the final answer and more computation time to guess the answer.

These 8.689 serve as members of the candidate set, as every single one of them has the possibility of being the final solution. As for the solution set, it is not necessary to store every word that has been attempted. In exchange, the solution set can be replaced with a list of five elements with each item stored sequentially and can be accessed directly or through random access. The reason being we only need to store the correct letters with the correct position in the set. Note that to input letters into the solution set, it is required to do sequential comparison between the candidate and the secret word or the solution. Therefore, the corresponding solution function can be pseudocoded as follows: “If every cell in the solution set is full, then the solution function returns true”. The feasibility function for this case must consult the current contents of the solution set and joins all contents of the list as a string, with empty cells replaced with regex wildcard (.). If a candidate pass the regex, then it passes the feasibility function.

For the optimization function, there will be two parameters used in this simulation: the computation time and the number of guesses made. This problems falls into the minimization problem category, as the strategies involved and used are expected to minimize the computation time and the number of guesses made.

IV. Wordle Heuristic Strategies and Results

There will be two different specific-use heuristic strategies that will be implemented and compared with each other for this simulation. There will also be two general-use heuristic strategies that will be used in every simulation.

The first general-use strategy is to test words that has the most variative vowels on the first guess. Those words are contained in a list named “first_words” as those will be the words that will be tested first. The contents of this list are the words "adieu", "audio", "auloi", "aurei", "louie", "miaou", "ouija", "ourie", and "uraei". All of the above letters has four different vowels and this strategy is expected to eliminate as many candidate members in the first guess.

The second general-use strategy is to use a priority queue. The reason for using this specific type of data structure is so that the simulation knows what word to be used for the next guess. As previously mentioned, greedy algorithms always attempts to pick the best option at each step. By using priority queue with degree of correctness as the priority number, the simulation may pick the best option at all times. The degree of correctness for each word is equal to the amount of misplaced

letters in the corresponding word and add that with the amount of correct letters times six. The reason six being used is that it's higher than five, which is the length of the word, and it avoids a problem where a word with five misplaced letters is used as a guess instead of a word with three correctly placed letters. To implement this in Python, the researcher construct a class using object-oriented programming. The researcher created two types of class: one to store word and priority number named HashMap and the other for the priority queue itself named PriorityHashQueue.

```python
class HashMap:
    # This class is used to create a hashmap
    def __init__(self, word, match_value):
        self.word = word
        self.match_value = match_value

    def get_word(self):
        return self.word

    def get_match_value(self):
        return self.match_value

```

Figure 5 HashMap Class Implementation
(Source : Personal Library)

```python
class PriorityHashQueue:
    def __init__(self):
        self.queue = []

    def returnWordIdx(self, index):
        return self.queue[index].get_word()

    def returnMatchValueIdx(self, index):
        return self.queue[index].get_match_value()

    def returnLength(self):
        return len(self.queue)

    def returnContents(self):
        for i in self.queue:
            print(i.get_word(), i.get_match_value())
        print()

    def returnRandomValue(self):
        return self.queue[random.randint(0, self.returnLength() - 1)]

    def returnTopValue(self):
        return self.queue[0]

    def enqueue(self, word, match_value):
        if (self.returnLength() == 0):
            self.queue.append(HashMap(word, match_value))
        else:
            newEntry = HashMap(word, match_value)
            for i in range(self.returnLength()):
                if (i == self.returnLength() - 1 or match_value == 0):
                    self.queue.append(newEntry)
                    break
                elif (match_value > self.queue[i].get_match_value()):
                    self.queue.insert(i, newEntry)
                    break

    def dequeue(self, index):
        return self.queue.pop(index)
```

Figure 6 PriorityHashQueue Class Implementation
(Source : Personal Library)

The first specific-use heuristic strategy is by using a "banned letters" list. This list contains letters, can be vowels or consonants, that have been confirmed to not exist in the secret word. The goal here is to eliminate as many words as possible with every guess. Theoritically speaking, this strategy is good, but has its flaws. Assuming that the secret word has five distinct letters, it means that there are 21 incorrect letters. To eliminate every word from the candidate list with at least one of the 21 incorrect letters, it requires at least 5 guesses, which is 5/6 of the total guesses the game provides.

```python
def filterWordsBannedLetters(word_list, banned_letters, baseline_answers, misplaced_letters):
    # This filter uses list of banned letters to eliminate words from the candidate list (word_list)
    wordHashList = PriorityHashQueue()
    try:
        for i in range(word_list.returnLength()):
            word = word_list.dequeue(0).get_word()
            isBanned = False
            matching_letters = 0  # Priority value score
            for letter in banned_letters:
                if letter in word:
                    isBanned = True
                    break
            if not isBanned:
                index = 0
                for letter in misplaced_letters:
                    if letter == baseline_answers[index]:
                        matching_letters += 6
                    elif letter in word:
                        matching_letters += 1
                    index += 1
                wordHashList.enqueue(word, matching_letters)
    except Exception as e:
        print(repr(e))
    return wordHashList
```

Figure 7 First Heuristic Strategy Implementation
(Source : Personal Library)

The second specific-use heuristic strategy is by using regular expression (regex). As previously mentioned, the feasibility function can extract the solution set into a regex with the empty cells replaced with a wildcard. The idea here is to eliminate every word in the candidate list that gets rejected from the regex test. In theory, there should be an improvement from the previous strategy because assuming that a five-letter word can be made from all 26 characters in the alphabet, regardless whether there exists a void in meaning or not, that means there are 26^5 words. Assuming the regex consists only 1 non-wildcard element, like /^…a.$/, this would make the candidate set contents size at a 26^4 words, a decrease of 96%.

```python
def filterWordsRegex(word_list, baseline_answers, misplaced_letters):
    # This filter uses regex to eliminate words from the candidate list (word_list)
    wordHashList = PriorityHashQueue()
    try:
        for i in range(word_list.returnLength()):
            word = word_list.dequeue(0).get_word()
            isBanned = False
            matching_letters = 0  # Priority value score
            if not re.match("".join(baseline_answers), word):
                isBanned = True
            if not isBanned:
                index = 0
                for letter in misplaced_letters:
                    if letter == baseline_answers[index]:
                        matching_letters += 6
                    elif letter in word:
                        matching_letters += 1
                    index += 1
                wordHashList.enqueue(word, matching_letters)
    except Exception as e:
        print(repr(e))
    return wordHashList
```

Figure 8 Second Heuristic Strategy Implementation
(Source : Personal Library)

The third specific-use heuristic strategy is by combining the first and the second strategy with the hopes of achieving better results than the result of one specific-use strategy alone.

```python
def filterWordsBannedLettersRegex(word_list, banned_letters, baseline_answers, misplaced_letters):
    # This filter uses regex and list of banned letters to eliminate words from the candidate list (word_list)
    wordHashList = PriorityHashQueue()
    try:
        for i in range(word_list.returnLength()):
            word = word_list.dequeue(0).get_word()
            isBanned = False
            matching_letters = 0  # Priority value score
            for letter in banned_letters:
                if letter in word:
                    isBanned = True
                    break
            if not isBanned and not re.match("".join(baseline_answers), word):
                isBanned = True
            if not isBanned:
                index = 0
                for letter in misplaced_letters:
                    if letter == baseline_answers[index]:
                        matching_letters += 6
                    elif letter in word:
                        matching_letters += 1
                    index += 1
                wordHashList.enqueue(word, matching_letters)
    except Exception as e:
        print(repr(e))
    return wordHashList
```

Figure 9 Third Heuristic Strategy Implementation
(Source : Personal Library)

The result can be shown with the table below for the time and the number of guesses required. Strategy 1 is for “banned letters” list, strategy 2 is for the regex method, and strategy 3 is the combination of the first two strategy

Word	**Strategy 1**	**Strategy 2**	**Strategy 3**
Cully	0.02678 s	0.07329 s	0.02742 s
Rever	0.05221 s	6.90094 s	0.03840 s
Slurp	0.08432 s	10.84957 s	0.10699 s
Abode	6.66904 s	0.11388 s	0.05123 s
Fenny	0.14661 s	14.55807 s	0.04644 s
Nokta	0.97845 s	2.11133 s	0.61277 s
Dodgy	0.03228 s	0.83489 s	0.06668 s
Maybe	2.52278 s	0.48531 s	0.16146 s
Average	1.31406 s	4.49091 s	0.13892 s

Table 1 Computation Time for Three Heuristic Strategies
(Source : Personal Library)

Word	**Strategy 1**	**Strategy 2**	**Strategy 3**
Cully	9 guesses	7 guesses	4 guesses
Rever	10 guesses	16 guesses	6 guesses
Slurp	6 guesses	16 guesses	5 guesses
Abode	11 guesses	12 guesses	8 guesses
Fenny	11 guesses	21 guesses	4 guesses
Nokta	11 guesses	11 guesses	5 guesses
Dodgy	5 guesses	21 guesses	5 guesses
Maybe	14 guesses	11 guesses	5 guesses
Average	9.625 guesses	14.375 guesses	5.25 guesses

Table 2 Number of Guesses for Three Heuristic Strategies
(Source : Personal Library)

It can be deduced from the two tables that the third strategy is better than the first two strategy alone. The reason why the first strategy fails is for the reason mentioned previously. There may exists a chance that many guesses have to be made until every candidate that contains the letters not in the final answer are eliminated from the set. This takes time and guess attempts.

Another reason that can make the first strategy slow is for cases where the answer word has more than one repetition of letters. Like the word “Fenny” that has two n. This makes it harder for strategy 1 to work efficiently because it focuses more on eliminating the wrong answer rather than finding the correct answer.

The reason why the second strategy doesn’t work efficiently, or worse than the first strategy in multiple cases, is that the strategy focuses too much on finding the correct answer rather than searching and eliminating the incorrect answers. In English, there are multiple words that follow vowel

patterns like CVCCV with C being a consonant letter and V being a vowel letter. This makes the next guess attempts purely guessing and adding the risk of longer computation time and guess attempts. It will also difficult the algorithm to reduce the length of candidate list later in the program and since priority queue enqueue and dequeue process uses linear scan, the longer the list, the longer the time it will take.

The reason why the third strategy is better is because it focuses on both finding the correct answer and eliminating the incorrect answers. By decreasing the length of the candidate set with the banned_letters list and finding the correct answer with regex, the program finds the ansswer more efficiently even in cases where there exists double letters or more in the answer word. Even though the dictionary has four times the word list compared to the official Wordle dictionary, the current algorithm still manages to, in average, answer the daily puzzle without running out of guesses.

V. SUMMARY

Wordle is a web-based word-puzzle game where the player has limited guesses on the answer word. Hints are given to the player in the type of letter coloring. Green if the letter is correctly placed. Yellow if the letter exists in the answer but is currently misplaced. Gray if the letter doesn't exist. This research successfully determines that by implementing a greedy algorithm with a priority queue, a "banned_letters" list, and a regular expression to identify the possible correct words and eliminate incorrect words from the word list using a simulation with Python, an optimal solution is reached, even for a dictionary that is significantly larger than the Wordle dictionary.

VIDEO LINK AT YOUTUBE

https://www.youtube.com/watch?v=vOTjM04gdeQ

GITHUB LINK

https://github.com/LordGedelicious/Wordle-Puzzle-Solver-using-Regex-and-Greedy-Algorithm-in-Python

REFERENCES

[1] Munir, Rinaldi. Algoritma Greedy (Bagian 1). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[2] Munir, Rinaldi. Algoritma Greedy (Bagian 2). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

[3] Munir, Rinaldi. Algoritma Greedy (Bagian 3). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd Edition, pp. 414-450, Massachustes Institute of Technology, 2009.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Gede Prasidha Bhawarnawa 13520004

NOTE: Mohon maaf pak Rinaldi, file Word saya tadi corrupt karena disk space saya habis dimakan oleh Docker Engine dan saya baru menyadari hal ini jam 11.30, tanggal 12 Juni 2024, saat akan melakukan submisi. Saya sedang dalam proses mengerjakan ulang makalah saya dan saya dengan ini menyatakan hukuman yang diberikan karena kelalaian saya akibat mengirimkan makalah melewati deadline.