AI/ML Solution Against Cybersecurity Issues in Connected and Autonomous Vehicles (CAVs)

Master of Science in Internetworking

Capstone Project

Presented by

Hansen Zhang

Under the supervision of

Sandeep Kaur

Fall 2022 - Winter 2023

Table of Contents

1. Abstract

With the rapid development of Connected and Autonomous Vehicles (CAVs), critical challenges appear as well. Cybersecurity in CAVs is one of the most significant issues in the upcoming decades. This is a capstone project report for the MINT program essentially aims to solve Control Area Network (CAN) cybersecurity issues by gathering valid network message datasets M-CAN Intrusion Dataset (M-CAN) dataset to complete an Artificial Intelligence/Machine Learning (AI/ML) Model that facilitates the Intrusion Detect System (IDS). The outcome helps detect and classify cyberattacks through CAN inside of CAVs data communication. And the conclusion is made up of the comparison of the J48 decision tree classification ML algorithm and the Naïve Bayes classification ML algorithm. Both have pros and cons compared with each other. This project overall covers several concepts involving Connected and Autonomous Vehicles (CAVs), the Internet of Things (IoT), 5G data communication, Controller Area Network (CAN) protocol, Cyberattacks, Cybersecurity, and AI/ML.

The project report is composed of the following 5 sections:

1. Relative background knowledge was described in the Introduction section to help viewers better comprehend the project topic and understand part of the fundamental mechanism of CAVs related to this project.
2. The Literature Review section provides an analysis of the current condition and procedure of cybersecurity of CAVs research so far. The purpose is to acknowledge the research progress and establish the context and significance of the study and provide a basis for the next two steps, which are the methodology and results of the project.
3. The methodology section decomposes the procedure of M-CAN dataset selection, data analysis, dataset preprocessing, model implementation, and model performance analysis procedure.
4. The discussion part describes the challenges and issues that were met during the project and some opinions related to this project.
5. A conclusion was made in the last section.

2. Introduction

2.1 Types of cyberattacks

As humans increasingly rely on digital technology, their information leaves a trail everywhere on the internet. This growing dependence has also led to a substantial increase in the risk of cyberattacks. Data from Comparitech reports that over 71 million people become victims of cybercrime each year, and the number continues to rise. (Howarth, 2022) The Live Cyber Threat Map (Figure 1) shows that millions of cyberattacks occur daily. Consequently, the reality of cyber threats is much graver than it appears if individuals do not take adequate precautions.

Figure 1 (Live Cyber Threat Map, 2023)

However, these millions of cyberattacks are so specific and various. Recognizing and identifying their malicious actions requires a thorough comprehension of their underlying intention patterns. And how to design a comprehensive defense system against these threats is critical.

A common method that hackers use to penetrate connected devices is by attacking computer ports and protocols. **Ports** are a transport layer concept that serves as a designated entry and exit points for network connections. They are based on software and managed by the device's operating system, with each port designed for a specific service or process. Devices can exchange different types of data through various ports on the internet. (Anonymous, What is a computer port? | Ports in networking, 2023) And before the topic steps further. Another term **Protocol** should be introduced as well. In data communication, Protocols describe rules and standards in digital format that govern the exchange of data between devices. These rules and standards define how data are successfully transmitted, received, and processed, ensuring that data are

accurately and efficiently transmitted and received. (Wesley Chai)

Fortinet, one of the most professional commercial cybersecurity companies, introduces that cyberattacks could mainly be classified as 20 types (Types of Cyber Attacks, 2023), some of which are mutually relevant and duplicate, to make it more concise and understandable for further research, they are simplified into 14 types in this report:

1. Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks
   These two attacks are basically the same type of cyberattack that make a network resource or website unavailable to its intended users by occupying it with massive traffic from multiple sources. And commonly that some ports are especially targeted for DDoS attacks, such as Port 80 for TCP, and Ports 443, 8080, and 8443 for HTTP and HTTPS. (Myers, 2014) The difference between DdoS and DoS is that DoS is a point-to-point attack, and DdoS is an advanced attack to spread out the sources to augment efficiency and make hackers less trackable. (DoS and DDoS Attacks: What are Their Differences?, 2023) DDoS and DoS brutally destroy the function of the service and are easily detected.

2. Man-in-the-middle (MITM) Attacks
   It's a kind of cyberattack that refers to a data breach that invades and captures the data in the middle of the data communication route. It often occurs on TCP Port 80 for HTTP or TCP Port 443 for HTTPS. The hacker meticulously inserts a malicious device into the interaction between two sides and spying or even manipulates data without being noticed by the attacked users.

3. Phishing Attacks Attacks and derived Phishing Attacks
   This common cyberattack frequently occurs in email by camouflaging the website into a trusted and safe website to gather sensitive personal information, such as digital property or stored passwords from its target. Moreover, some phishing attack lures innocent users to download malware for further harmful manipulation. And there are two derived phishing attacks named Whale-phishing and Spear-phishing. Whale-phishing specifically aims at larger organizations or companies. Spear-phishing has one specific target and only aims at that individual.

4. Ransomware
   In the field of cybersecurity, ransomware is an internet version of ransomware in the real world. The attacker threats the victim's device or system to pay a ransom to get the permission of the electrical "hostage" back, for example, database or server control permission. As soon as the payment has been sent, the attacker provides instructions regarding how the target can regain control of their computer. The name "ransomware" is appropriate because the malware demands a ransom from victims.

5. Password Attacks

Password attacks mean breaking through access by illegally gaining passwords. It could occur in social engineering form, like password trading and Shoulder surfing. Or in a technical perspective, there are more methods, such as:

a. Brute Force Attack – Redundantly attempting every possible combination of characters until the correct password is found.
b. Dictionary Attack – Using the professional predefined list of words as passwords and trying each one until the correct password is found.
c. Password Reuse – Using a list of previously obtained passwords to attempt accessing the system.

Etc. (Rees, 2022)

6. SQL Injection Attacks

   SQL Injection is a type of cyberattack that exploits vulnerabilities in a website's database by injecting malicious SQL code into a web form input field. It usually aims at the database port, usually TCP port 1433 (Gupta, 2019) for Microsoft SQL Server, and TCP port 3306 for MySQL (MySQL Port Reference Tables, 2023). The attack does not attack a specific port, but rather through the port and invade the database server itself, exploiting vulnerabilities in the database software and tricking the database into executing unauthorized commands, such as extracting or modifying sensitive data. This can result in significant damage to the website and its users, such as theft of sensitive information and disruption of service.

7. URL Interpretation Attacks

   URL Interpretation is the manual process of analyzing the Uniform Resource Locator's (URL) components such as protocol, hostname, path, and query parameters, to determine the resource being requested by a client. (Parashar, 2022) If the permission of some websites is not regularized, hackers can illegally access them by manually changing the URL components.

8. DNS Spoofing Attacks

   DNS Spoofing, also known as DNS cache poisoning, is a type of cyberattack in which a malicious actor alters the mapping of a domain name to a different IP address, redirecting traffic intended for one website to another. This allows the attacker to intercept sensitive information, such as login credentials or other sensitive data, and can also result in the delivery of malicious content to the user. (Anonymous, What Is DNS Spoofing and How Can You Prevent It?, 2022)

9. Insider Threats

   Insider Threats refer to security incidents that occur as a result of actions taken by individuals within an organization, such as employees, contractors, or vendors, who have access to the organization's sensitive information and systems. (Froehlich, 2023) These individuals can pose a threat either deliberately through malicious intent or accidentally through human error. Insider threats can lead to data breaches, theft of confidential information, and damage to an organization's reputation and

bottom line.

10. Trojan Horses Attacks

    A Trojan Horse is a type of malicious software that disguises itself as a legitimate program but actually allows an attacker to gain unauthorized access to a victim's computer. The attacker can then use the compromised system to steal sensitive information, install additional malware, or carry out other malicious activities. Trojans can be delivered through various means, such as email attachments, infected software downloads, or malicious websites. (Trojan Horse Virus, n.d.)

11. Drive-by Attacks

    Drive-by Attacks are a type of attack in which a user's computer is infected with malware simply by visiting a compromised website. The attacker infects the website with malicious code, which then exploits vulnerabilities in the user's web browser or plug-ins to install malware on their computer without their knowledge. (What is a drive-by attack?, n.d.) Drive-by attacks can lead to the theft of sensitive information, the installation of additional malware, or the use of the infected computer as part of a larger attack.

12. XSS Attacks

    Cross-Site Scripting (XSS) attacks are a kind of attack that targets at implementation vulnerability that allows an attacker to inject malicious code into a normal and trusted website. (KirstenS, n.d.)When users visit the websites as usual, they will be passively executed by the browser. XSS attacks can be used to steal sensitive information, such as login credentials or other sensitive data, or to carry out other malicious activities, such as launching phishing attacks or distributing malware. XSS attacks can be defended and prevented by adding proper input validation and encoding, as well as by implementing a Content Security Policy (CSP) to restrict the execution of malicious code on the website.

13. Birthday Attacks

    A Birthday Attack is a type of cryptographic attack that takes advantage of the mathematical properties of hash functions. Hash functions are used to convert input data into a fixed-length output, called a hash, that can be used for verification purposes. (Birthday attack, n.d.) A Birthday Attack takes advantage of the fact that it is possible to find two different inputs that produce the same hash value, known as a hash collision. This can allow an attacker to create two different messages that produce the same hash, thereby tricking a system into accepting a malicious message as valid.

14. Malware Attacks

    Malware attacks occur through implementing malicious software to harm the target’s computer systems, steal sensitive information, or carry out other harmful actions. Malware can be delivered through various means, such as email

attachments, infected software downloads, or malicious websites. Once installed on a computer, malware can carry out its intended actions, such as stealing sensitive information, altering or destroying data, or disrupting the system's normal operation.

After recognizing the majority of cyberattacks, how do we recognize and defend them in reality? To protect the safety of network and data communication, Cybersecurity appears. It is the practice of protecting critical data and digital and physical systems from cyberattacks. And it has expanded to one of the most popular topics worldwide. Nowadays cyberattacks can penetrate every field once the related information could be converted into digital form on a modern device.

2.2 What is Cybersecurity

After knowing what is the cyberattack, Cybersecurity is the next concept necessary for this project. Literally, it is the technical methodology for defending against cyberattacks. And the ways to defend against cyberattacks are various. They mostly involve in hardware design, software design, social awareness, and regular defend updates:

1. Firewalls: An independent or inner network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules. It could be designed as software, hardware, or hybrid.
2. Encryption and decryption: Encryption is a process of converting original text into a string of unreadable text to protect the data from unauthorized access. Meanwhile, decryption is the process to convert the chaotic text back to the original data. The combination of encryption and decryption could increase the safety of the data transmission process.
3. Antivirus software: A kind of software designed to detect, prevent, and then remove malware, virus, and malicious software to protect the system.
4. Intrusion detection and prevention systems (IDS/IPS): Network security technologies that monitor network traffic for suspicious activity and block or alert on potential threats.
5. Access control: Restricting access to resources, such as data and networks, to authorized individuals. For example, creating a blacklist or whitelist for network access or adding passwords to gain access permission.
6. Regular software updates: Installing software updates that address security vulnerabilities.
7. User education and awareness training: Educating employees about safe computing practices and the importance of cybersecurity.

This project focuses on the data link layer communication attack defense by implementing a Machine Learning/Artificial Intelligent Model to predict and detect unusual network behaviors as a partition of IDS. **IDS** generally stands for Intrusion Detection System. It is specifically designated to constantly scrutinize the network connection and detect malicious or abnormal behaviors. It can be either in hardware form or in software form. As well as the issues are detected, the IDS generates feedback

or alarms for the administrator to investigate. (Rogers, 2015)

2.3 Connected and Autonomous Vehicles (CAVs)

CAVs are rapidly gaining attention and popularity in the current technological landscape. These vehicles are capable to operate without human intervention and are equipped with advanced sensors, cameras, and software that allow them to navigate roads and make decisions on their own. However, a vehicle defined as an autonomous vehicle is unnecessarily to be fully self-driven. In 2014, the Society of Automobile Engineers (SAE) technically classified the autonomous vehicle levels into 6 levels (SAE, 2021):

- Level 0: No automation.
  The human is responsible for all driving tasks, including steering, accelerating, breaking, etc. And all driving procedures are done manually.
- Level 1: Driver Assistance
  The human is assisted with single automated systems on driving, including monitoring speed, steering assistance OR braking and acceleration assistance. The driver is still responsible for the majority of driving tasks and must always be ready to take over the vehicle whenever necessary.
- Level 2: Partial Automation
  This level utilizes Advanced Driving Assistance Systems (ADAS). The vehicle can fully automate vehicular tasks such as acceleration, steering, and breaking in certain circumstances. But it does not guarantee safety in all kinds of situations, the driver still needs to observe and supervise the vehicle and remain active to take over the vehicle at all times.
- Level 3: Conditional Automation
  The Level 3 vehicles are capable of automatically detecting environmental factors of driving procedure. They can adjust their driving decision according to the situation. At this level, the human does not need to supervise the driving procedure all the time. But the chance of taking control back to humans is still available and required whenever a special situation, such as an emergency, a system failure, or a traffic incident occurs.
- Level 4: High Automation
  At this level, the vehicle abandons the manual steering wheel and pedals. And a human driver is no longer needed. The AVs can make the right decision at any time. But this level only represents public transportation, such as driverless taxis or buses that routinely travels back and forth alone certain route.
- Level 5: Full Automation
  The ultimate level indicates the vehicle can automatically go everywhere in all situations itself without human supervision and manipulation.

Therefore, as the categorization shows, the vehicles in levels 1 to 5 are all supposed to be AVs.

Whether people are aware or not, the concept of autonomous vehicles has been around for almost a century. Back in 1925, before World War II occurred, an American called Francis P.Houdina tested the world's first autonomous vehicle called the “American Wonder” in practical. The “American Wonder” achieved wireless remote control on the steering wheel, clutch, brake function, and other components via radio waves. (Felton, 2017) 14 years later, Between 1939 to 1960, Norman Bel Geddes and General Motors achieved the first self-driving electric car supported by a radio-controlled electromagnetic field in reality on the road. (Guy, 2021) However, back in that time, the self-driving vehicle could simply drive alone on a few given routes embedded with wires. Even though these prototypes decades ago could be classified as Level 4 or even Level 5 AVs, there were countless restrictions to make them run in the given area and it was impossible to allow them to run on the public road. Hence the development of autonomous vehicles has encountered a bottleneck since then.

Until recent years there has been a significant acceleration in the pace of development. This sudden breakthrough can be attributed to several factors, including advancements in technology, increasing investment in the field, and a gradually growing demand for safer and more efficient modes of transportation. As TechCrunch reported, there are over 1400 level 2 AVs in the US in 2022. (Kopestinsky, 2022)

2.4 The Internet of Things (IoT)

The Internet of Things (IoT) has been a major contributor to the growth and evolution of autonomous vehicles. IoT essentially gives physical objects a digital identity by assigning metadata, such as an address, date, authors, data size, and attributes like size, number, weight, and color, to them. With the integration of networks and software, these physical objects can communicate and exchange information with other digital entities, resulting in a unique virtual identity. This interconnected system of physical and digital entities has provided numerous benefits to the development of autonomous vehicles:

1. Mapping and Navigation systems:

   The development of high-resolution mapping systems, which provide comprehensively dynamic information about road infrastructures and environment, which improves the convenient navigation system from starting point to destination. For instance, nowadays commercial map apps and software on cellphones or vehicle inner control panels such as Google Maps. They provide several significant benefits to autonomous vehicles, including: (Deny, 2021)

   - Smart Navigation: Nowadays’ Google Maps provides real-time mapping data accurately and comprehensively. And such is essential for autonomous vehicles to navigate and reach their destination. The maps include information on road geometry, road signs, traffic signals, and other critical information needed for autonomous vehicles to make safe and efficient driving decisions.

(Figure 2)

- Real-time Traffic data: Google Maps provides real-time traffic data, including traffic speeds, congestion, and construction information. This information is critical for autonomous vehicles to make informed decisions about routing and avoid traffic delays.
- Local knowledge: Google Maps has detailed knowledge of local landmarks, businesses, and other points of interest, which can be used by autonomous vehicles to provide passengers with a more personalized and interactive experience.
- Efficient routing: Google Maps uses machine learning algorithms to provide efficient routing recommendations, taking into account real-time traffic information and other factors. This fruitfully assists autonomous vehicles to minimize travel time and fuel consumption and reduces their carbon footprint. Meanwhile, the abruption of the driving procedure could be fixed with immediate re-optimizing routing, which always leads the vehicles to the reachable destination.

Figure 2 Google Map

- Integration with other Google services: Google Maps integrates with other Google services, such as Google Earth and Google Street View, to provide a comprehensive and interactive mapping experience. This can be used to provide autonomous vehicles with detailed information about their surroundings, including road conditions, traffic patterns, and points of interest.

Therefore, an integral mapping system is a critical component of the autonomous vehicle ecosystem, providing essential navigation, traffic, and local knowledge to support safe and efficient autonomous driving. By leveraging the extensive mapping and location data provided by Google Maps, autonomous vehicles can make informed driving decisions, provide a more personalized experience to

passengers, and minimize travel time and fuel consumption.

2. Global Positioning System (GPS):
   The GPS is a network of satellites that observe and send the accurate location (e.g. latitude, longitude) and time information to vehicles on the Earth. (Anonymous, How Self-driving Cars Work: Sensor Systems, 2021) By connecting GPS devices to the network, vehicles and smart devices can correspond and share real-time information. Furthermore, thank the benefits of GPS, IoT enables autonomous vehicles to even communicate with other elements on the road, such as road infrastructure, traffic signals, and other vehicles. This communication allows the vehicles to have a clear understanding of their environment and make informed decisions for increased safety.

3. High-resolution surrounding cameras: However, even though mapping systems provide plenty of benefits, in the perspective of autonomous vehicles, the technology above is far from usable for mature level 3 or even higher vehicles. One critical shortage is geometrical precision. It is reported by the U.S. government that the current Global Positioning System (GPS) accuracy is 4.9 ft. (U.S. Government, 2022) Such ranging accuracy accumulated in traffic involving hundreds of vehicles could result in critical traffic accidents without human operation in autonomous vehicles.

   Therefore, as the investment and development continued, technologies focused on higher resolution and more accurate object observation is under exploitation. In 2018, Google posted an official demonstration on YouTube for Waymo 360° Experience (Figure 3). It demonstrated the new technology applied on AVs that utilizes LiDAR to gather pixel-size real-time detailed pictures from all directions and exchange them with vehicle sensors to optimize route planning. It drastically increases the traffic observation precision and analysis ability against complicated conditions from meters to centimeters. (Waymo, 2019) For example, in Figure 3 and Figure 4, the Waymo 360° Experience dynamically scans surrounding vehicles, pedestrians, traffic lights and traffic lines, etc. And the quantified information related to traffic and driving is sent to the autonomous driving system for analyzing the traffic condition and making an instant decision. By utilizing Artificial Intelligence, the autonomous vehicle can accumulate the acquired traffic information and instantly simulate drivers to send vehicular orders to the car. Meanwhile, the increased processing power of computers and the development of powerful artificial intelligence algorithms significantly promoted road navigation technology to the next generation as well.

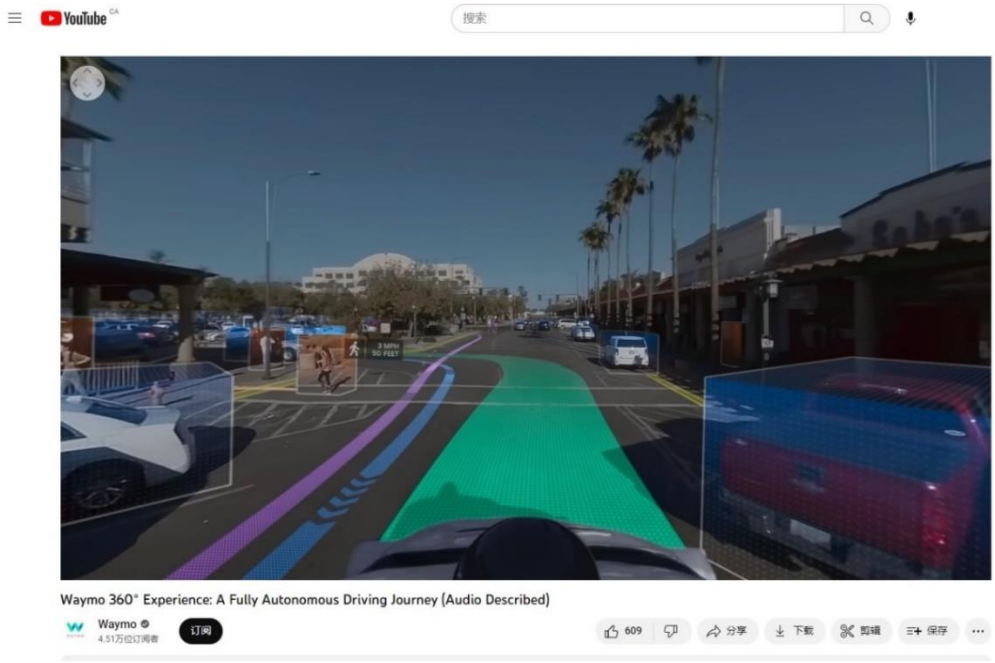

Figure 3 Waymo 360° Scanning Nearby Vehicles and Routes

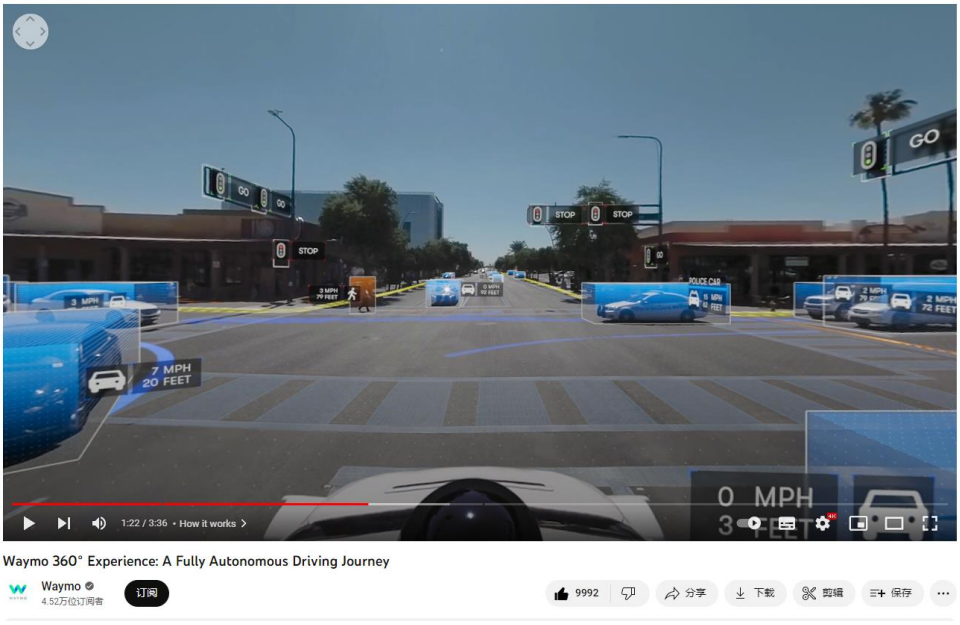

Figure 4 Waymo 360° Scanning Nearby Traffic Lights and Traffic Lines

4. Big Data Analysis: Since the development of IoT, vast amounts of data generated by devices and autonomous vehicles. The growing population of complicated data provides a potential value for analyzing the performance and reliability of autonomous vehicles. As reported by Florian Götz in 2021, each autonomous vehicle in America can generate 434 TB data to 5894 TB of data annually. (Götz, 2021)

2.5 How does 5G benefit CAVs?

The second critically important factor has been the development of data communication which supports an exponentially increased amount of data transportation. In 2023, 5G, which stands for the 5th generation mobile network, plays a significant role in the CAV field by supporting higher multi-Gbps peak data speeds, ultra-low latency, more reliability, massive network capacity, increased availability, and a more uniform user experience to more users. As the Huawei 5G Security White Paper indicates, the benefits are concluded as below (Huawei, 2021):

1. Improved data transfer speed and reduced delay: 5G technology offers substantial advancements in data transfer speed and reduction of delays compared to previous cellular network generations.

   The role of Ultra-Reliable and Low-Latency Communications (URLLC) provided by 5G technology is crucial for autonomous vehicles. In high-speed driving scenarios, milliseconds can make a critical difference in the outcome of an incident. The reduced latency of data transportation allows autonomous vehicles to make decisions faster, thereby potentially reducing casualties on the roads.

   These improvements enhance the ability of autonomous vehicles to receive and analyze information in real-time, resulting in more precise and effective decision-making.

2. Improved communication: 5G allows for more reliable communication between vehicles and other road users.

   Massive Machine-Type Communications (mMTC) is the second potential 5G enhancement that facilitates CAVs, especially Vehicle-to-everything technology (V2X). mMTC is designed to handle a massive number of connected devices, such as traffic lights, infrastructure, and other vehicles. This makes it well-suited for deployment in large-scale autonomous vehicle fleets. mMTC enables cost-effective communication and data transfer for autonomous vehicles, as it uses low power and requires less complex network infrastructure compared to other communication technologies.

   Such comprehensive data sharing can lead to improved safety and coordination on the roads.

3. Increased capacity: 5G offers greater capacity for data transfer, which is important for autonomous vehicles as they generate and use large amounts of data for navigation, decision-making, and communication.
   5G provides Enhanced Mobile Broadband (eMBB) for services with gigantic data

transportation requirements, such as high-definition video, and high-resolution real-time graphical information transportation.

4. Enhanced security: Compared to 2G, 3G, and 4G networks, 5G provides multiple enhancements to cybersecurity, such as enhanced user privacy protection, better-roaming security, stronger air interface security, and Enhanced Service-Based Architecture (SBA) security, etc (Figure 5). This is important as autonomous vehicles handle sensitive information, such as passenger data, and need to be protected from cyber threats.

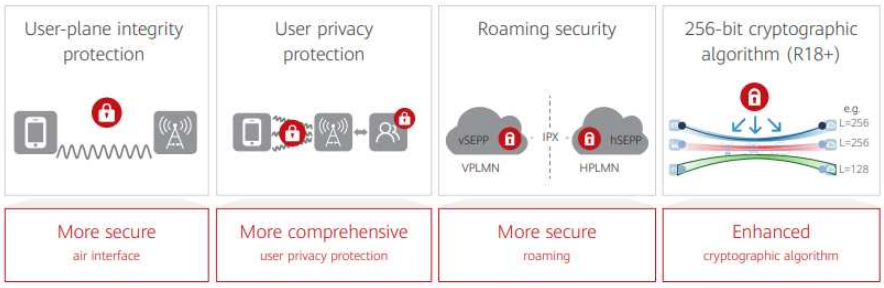

Figure 5 5G security hardening features (Huawei, 2021)

2.6 Autonomous Vehicles and Cyber Security

In recent years, there has also been a growing demand for safer and more efficient modes of transportation, which has further accelerated the development of autonomous vehicles. With the increasing concern about traffic accidents caused by human error and the desire for more friendly transportation options, there has been a growing interest in developing autonomous vehicles as a solution. But there are still plenty of incoming challenges and problems in the future.

In the 21st century, accompanied by the development of digital technology, commercial automotive companies continuously ameliorate vehicular products with digital accessories, such as digital control panels (Figure 6), external cameras (Figure 7), and remote-control software such as smart car control apps (Figure 8). Thank to the pervasion of digital technology, these upgrades effectively facilitate our traveling with consistently simplified driving skill requirement and better route navigation. However, just like what was mentioned above, those conveniences brought threats and breakthroughs to digital privacy as well.

Figure 6 speedometer, dashboard, vehicle, car, control panel, technology, drive, control (Webpage·2019·Pixnio)

Figure 7 The Best Backup Camera and Displays (Webpage·2023·Smirniotis)

Figure 8 Smart car control app mobile interface vector template. (Webpage Smartphone application page purple design layout)

As we can see in Figure 1 above, even though the current network systems provide a fortified defense system for network security, there are still millions of cyberattacks happening every day. It is because there is no perfect network system. Cybersecurity could only defend against known and detected threats and generate specific solutions against them. Hence, attacks are always one step further against cybersecurity. Therefore, the war between cybersecurity and cyberattack will never stop.

During internet usage in daily life, cyberattacks might not be extremely critical, for example capturing a personal email address for spamming email or defrauding money from a bank account, but when it happens on a vehicle speeding on the flyover, the aftermath would be mortal. In 2015, two security researchers called Charlie Miller and Chris Valasek astonished the world by remotely hacking into the vehicle's control system through a flaw in the car entertainment system called Uconnect. The successful hack gathered permission inside the car's internal Controller Area Network (CAN) to send commands to engines and wheels freely. (Rashid, 2018) Fortunately, this malicious incident did not cause any casualties. But finally, Fiat Chrysler, the related manufacture company of the hacked vehicle, urgently recalled 1.4 million vehicles and was fined $105 billion by the National Highway Traffic and Safety Administration. (Korosec, 2022)

Through this case, we can find CAN is one of the important terms in the cybersecurity of a vehicle, then what is CAN? And why back in 2015, it could be the very first invaded entry for a vehicle?

2.7 CAN Protocol Overview

Defined by the JavaTPoint website, Controller Area Network (CAN) is a message-based communication protocol implemented for microcontrollers and devices inside of CAVs to correspond with each other without the need for a central host. (CAN (Controller Area Network) protocol, 2011) CAN cover the majority of electronic gears inside of the vehicle. Basically, vehicular components that were controlled electrically such as Wiper controller, CD player, windows controller, lighting controller, seatbelt controllers and indicator, engines, cruise controller, and antilock brakes, all above require CAN buses to communicate with their related gear to achieve specific tasks. (Parikh, 2023) And since the vehicle is decomposed into numerous digital individuals in the vehicle and can communicate via CAN, CAN is also one of the most fundamental products derived from IoT technologies. Meanwhile, there are also other two substitutions called Local Interconnected Network (LIN) and FlexRay. But according to the article of Jim Harrison, they have different pros. and cons (Harrison, 2020)(shown in table 1). Therefore, they could not fully replace the other two.

Names	Pros.	Cons.
CAN	Easy implement	Slower than Flexray
LIN	Cheap	Slowest speed/bandwidth
FlexRay	High data rates Low delay	Costly Much more complicated design

Table 1 Pros & Cons of different network

According to "the Introduction to the Controller Area Network (CAN)" written by Steve Corrigan, the CAN protocol provides services in the physical and data link layers

of the OSI model. In the physical layer, the vehicle usually uses simple 25Kbps to 1Mbps twisted-pair wires to facilitate the CAN bus. But currently, the technology has already supported 2Mbps data speed. (Harrison, 2020) In the data link layer, CAN uses a broadcast method to transmit frames on the wire. (Corrigan, 2016) And CAN bus refers to the physical wire or network that connects devices using the CAN protocol.

And “The CAN Bus Protocol Tutorial” published by the Kvaser website makes a comprehensive explanation to the learner about the fundamental principle of CAN protocol.

First of all, all the messages communicated in the CAN bus could be heard by all nodes that are connected due to the broadcast data transport method. But the physical layer implementation helps filter the messages so that every node is only responsible to react to specific target messages with unique identifiers. (The CAN Bus Protocol Tutorial, 2023)

Exclusively, the CAN protocol message is a message-based protocol. (Parikh, 2023) It does not contain a special start point address or destination address like an address-based protocol frame, whereas it uses another field called Identifier in the CAN frame to indicate identity and help the nodes to be distinguished from each other. (The CAN Bus Protocol Tutorial, 2023) And the identifier will be introduced later in the frame structure part.

And essentially, the CAN protocol contains 4 types of messages according to the report of J.S. Freudenberg (J. A. Cook, 2008):

1. **Data Frame**
   The data frame is the message that contains broadcast data information to the destination. This could be sent from controllers such as engine controllers or Wiper controllers.

2. **Remote Frame**
   Remote Frame is a request message sent by a transmitter to acquire data from a specific node. It could be sent from a speedometer.

3. **Error frame**
   Error frame is used literally by any nodes for error detection in the bus.

4. **Overload frame**
   The overload frame is utilized to inject an extra delay between data or remote frames.

2.8 CAN Frame Structure

There are two types of CAN Frame structures. One is called the Standard CAN Data frame, which is CAN 2.0A; whereas another one is the Extended CAN Data frame, which is CAN 2.0B.

Figure 9 Standard CAN structure (CAN 2.0A): 11-Bit Identifier (Corrigan, 2016)

Steven Corrigan describes the basic standard CAN frame structure in figure 9 shown above. And explanations of fields inside of a CAN frame are shown below according to Corrigan's research (Corrigan, 2016) and relative online reports (reference will be noted after the specific knowledge):

SOF: start of frame (SOF), which indicates the start point of the frame. It's a single dominant bit.

Identifier: The Standard CAN 11-bit identifier. This component describes the priority of the message, and it also indicates the identity of the message. Messages that are sent from the same node or contain the same type of content have their own identity and priority. The lower the binary value, the higher its priority. For example, if the identifier is all 0s, it indicates that this message has the highest priority, and all other messages that are less important than this should delay and wait in line.

RTR: The remote transmission request (RTR). It contains only one bit and is used to define the type of frame. And this bit holds prominence when data is necessary from another node.

IDE: Identifier Extension (IDE). This field indicates whether this CAN frame contains an extended format, which is an Extended CAN message.

r0: Reserved bit. It currently obtains no value and is just in case saved for future extension and usage.

DLC: The Data Length Code (DLC). A four-bit data shows how many bytes of useful data are transmitted in this frame.

Data payload: This is the meaningful content contained in each frame of CAN. This part of messages is used for communication between controllers and gears. And the length could vary from 1 byte to 8 bytes. As mentioned above, its length

is forecasted in the DLC field.

CRC: The Cyclic redundancy check (CRC). It is a 15-bit checksum plus a 1-bit delimiter for error checking. Used for checking the integrity and completion of data transmission.

ACK: Node Acknowledges. This is a 1-bit field that indicates if the message is error-free when it reaches the destination. And also followed by another 1-bit delimiter.

EOF: End-of-Frame. This 7-bit field indicates the completion of a CAN frame.

IFS: Interframe Space. The last field of CAN reveals how long this received CAN frame takes to be moved by the controller to its next appropriate buffer area.

Meanwhile, the Extended CAN structure is extended from the Standard CAN structure with additional two fields and replaced RTR with SRR:

Figure 10 Extended CAN structure (CAN 2.0B): 29-Bit Identifier (Corrigan, 2016)

SRR: The substitute remote request. Replacing RTR in CAN 2.0A, this bit represents a placeholder in CAN 2.0B.

IDE: Identifier Extension. The usage of IDE here is the same as IDE in CAN 2.0A. But instead of showing a dominant bit 0, CAN 2.0B shows a recessive bit 1 to indicate this is an Extended CAN frame.

r1: An additional reserved bit after r0. Reserved for future use.

2.9 CAN Cyberattack Types and Impacts

And to understand how cyberattack incurs the malfunction of CAN, Identifier, DLC and Data are three essential fields required for this project.

And according to the research, many typical cyberattacks could occur in CAN, such as DoS attacks, Fuzzy attacks, and Impersonation attacks.

1. DoS attacks:
   Denial of Service attacks in CAN is adding a malicious node inside of CAN and frequently sending the 0s messages with the highest priority to interfere and delay all other useful messages. Such a phenomenon is also known as packet flooding.

In this case, if DoS attacks show up, the controlled gears such as windows, break, engines, or high-beam headlights will not receive any messages and fall or delay to respond to the orders. It can cause a critical traffic accident if DoS attacks occur on a driving CAV.

For example, in Figure 11 when the Malicious Node C keeps sending 0s messages with the highest priority, the bus will always deliver these messages at first and all the others have to wait in line.

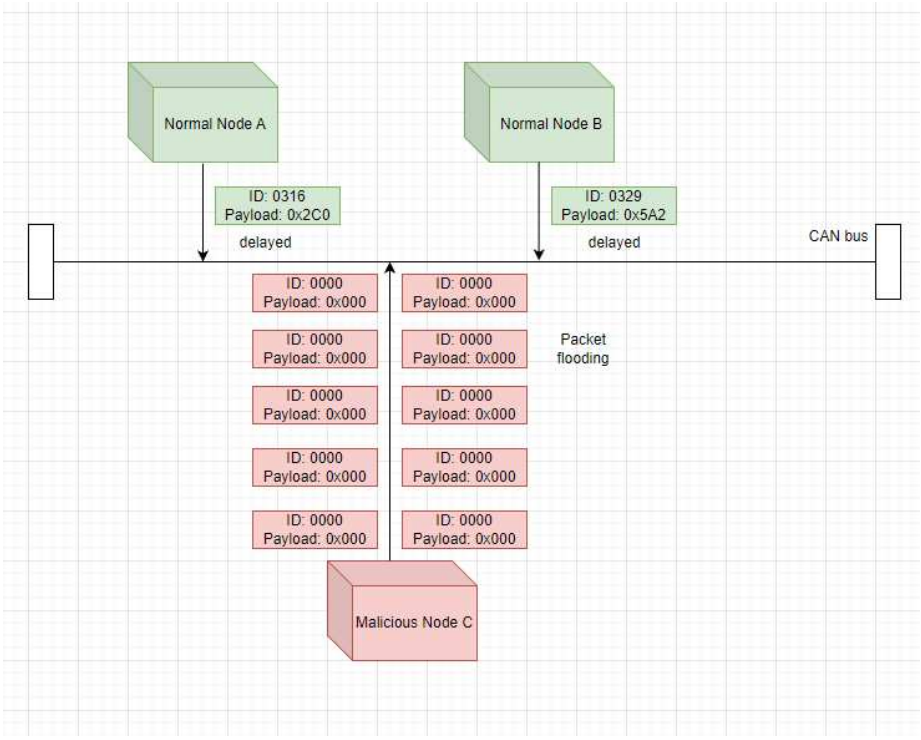

Figure 11 DoS attack in CAN drew in Draw.IO

2. Fuzzy attacks:
   In CAN bus, Fuzzy attacks could cause error actions of behavior or termination of the controller system. It is caused by injecting messages of spoofed random CAN ID and DATA values. And if the invalid message is received and is unable to be executed, the system will abandon the message or result in shutting down.

   In Figure 12, abnormal node C uses a spoofed random CAN ID to send an illegal message regardless of the rule of any nodes. And the CAN ID is possible to be the same as the legitimate CAN ID but contains unusual DLC and payload. In this case, if the data in the payload can not be processed, it will cause a malfunction of the related gear if the message fails to get abandoned.

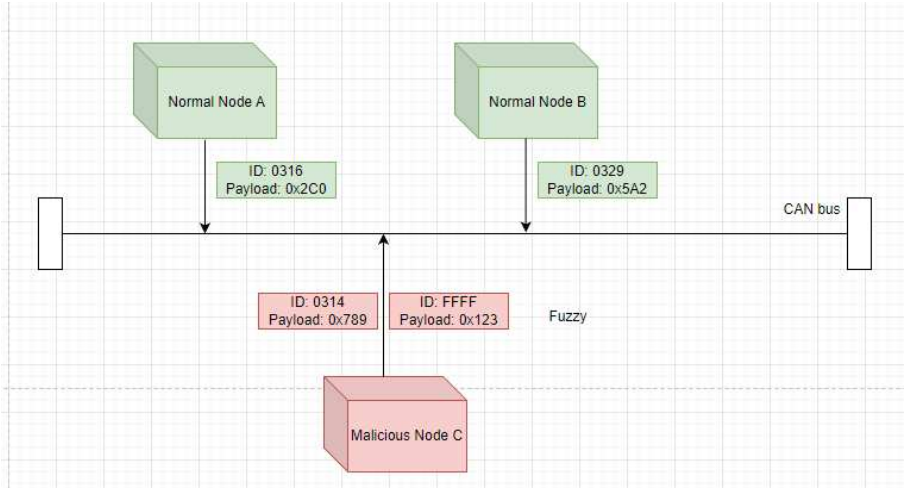

Figure 12 Fuzzy attack in CAN drew in Draw.IO

3. Impersonation Attack

   In a CAN network, this type of attack will mimic a legitimate node with a real CAN ID and send malicious messages to take over the original node and gain access to sensitive data or control of the CAV.

   In Figure 13, if normal node A is the original Speedometer on the dashboard of CAVs, and the malicious Node C mimics the Speedometer to keep sending legitimate messages, the hacker will be able to acquire the speed of the CAV and similarly gain even more sensitive data.

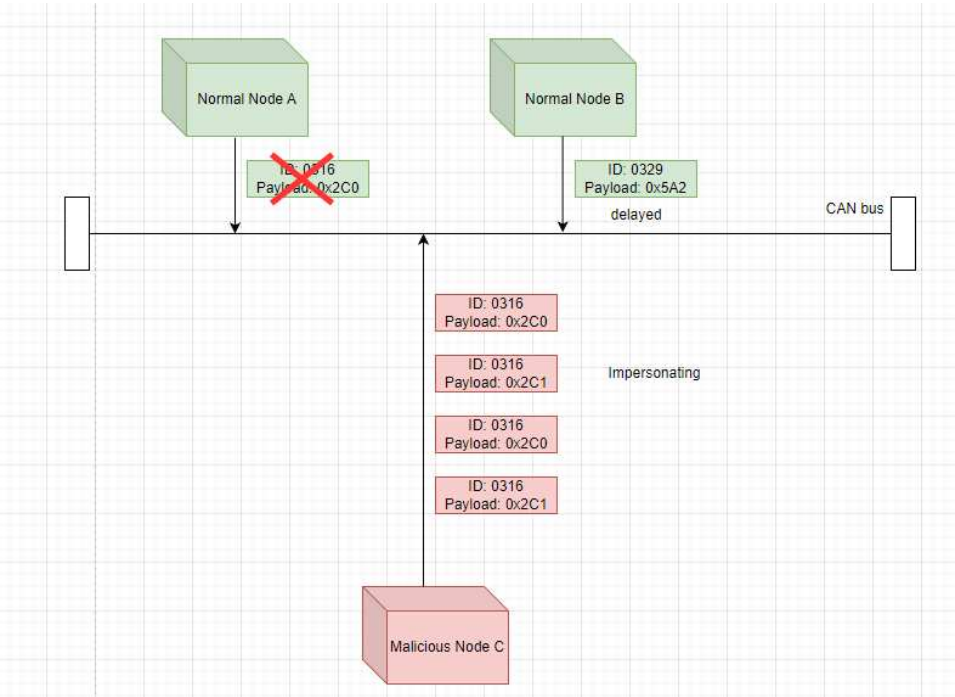

Figure 13 Impersonation attack in CAN drew in Draw.IO

4. Spoofing attack:

Hackers may cast a Spoofing attack on the CAN bus to inject falsifying messages of certain legitimate CAN IDs related to a certain function (Figure 14). It can mislead the function with wrong action or indication to CAVs to cause car accidents. messages to. Spoofing attacks are technically similar to Impersonation attacks. But the purposes are different. For example, the spoofing attack on the acceleration controller intends to send messages to let the car speed up on the highway will cause a vehicle to exceed the speed limit and hit another car.

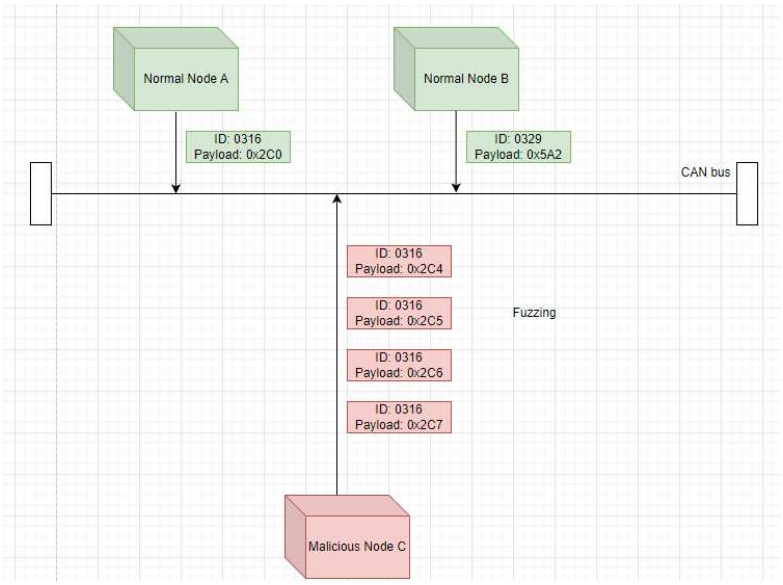

Figure 14 Spoofing attack from Node C in CAN drew in Draw.IO

Meanwhile, there are also some other types such as spoof attacks, and malfunction attacks.

Therefore, to protect the CAVs from these malicious threats, using AI/ML model to ameliorate IDS is critically imperative.

3. Related Works

Like what was discussed, in 2015, the first well-known car intrusion incidence happened on the CAN bus vehicle. And side with the rapid development of autonomous vehicles, how to detect and defend against attacks on CAN protocol is also growing into a popular topic in the field. AI/ML approaches toward the cybersecurity of CAVs are one of the most effective methodologies to fortify the IDS of CAVs. Many researchers from 2015 got in touch with this area. There has generated numerous datasets collected for CAV data communications.

3.1 Related Organizations and Datasets

HCR Lab

A Korean security lab called Hacking and Countermeasure Research Lab (HCR Lab) was established in 2010 and dedicated to improving data-driven security based on machine learning and data mining technology to generate and categorize hacking phenomena from a tremendous amount of data. In the field of Cybersecurity of CAVs, HCR Lab contributes to sharing real-world datasets for researchers to explore and improve the methodology of CAV's defense system. And HCR Lab currently is the only organization in the world that shares real-world datasets of cyberattacks with the public for free. (WELCOME TO HCRL, 2022) The conditions of each dataset are shown in Table 2.

Datasets related to CAVs cyberattacks that are shared by HCR Lab are below:

1. Car-Hacking Dataset
   A collection of CAN messages collected via the On-Board Diagnostics-II (OBD-II) port from a real vehicle. These datasets focus on continuous Controller Area Network (CAN) data transportation. It contains 4 separate datasets and each one records a continuous 30 to 40-minute of CAN traffic. Each intrusion occurred for 3 to 5 seconds.

2. CAN-intrusion-dataset (OTIDS)
   OTIDS datasets were documented by tracking the CAN frames via the OBD-II port from a running car while being attacked by malicious message injection.

3. Survival Analysis Dataset for automobile IDS
   This dataset targets to recognize malicious CAN frames without semantic knowledge of the CAN ID function. It was extracted from three different models of vehicles.

4. CAN-FD Intrusion Dataset
   A collection of CAN-FD (CAN with Flexible Data rate) messages contains malicious attacks recorded from a one-hour round-trip around Korean University.

5. M-CAN Intrusion Dataset
   This dataset contains extracted real M-CAN messages. M-CAN is a bus-type topology that is used for navigation and in-vehicle multimedia communication device modules. This data was extracted from the Genesis g80, and the attack messages were subsequently injected. The normal dataset was collected in a one-hour round drive around Korea University. And some of them are saved for M-CAN Intrusion Dataset, the others are used for B-CAN Intrusion Dataset.

6. B-CAN Intrusion Dataset

   This dataset is a B-CAN dataset containing attack messages. B-CAN is a network connecting BCM lights, power windows, and smart key modules. This data was extracted from the Genesis g80, and the attack messages were subsequently injected.

7. TOW-IDS: Automotive Ethernet Intrusion Dataset

   The dataset consists of three types of IVN data, namely AVTP, gPTP, and UDP, with the latter being converted from CAN messages. The collected data has been divided into two sets. One set contains normal driving data, while the other set includes abnormal driving data resulting from a simulated attack. The abnormal traffic is based on five predefined attack scenarios.

Dataset Name	**# MESSAGES**	**# NORMAL MESSAGES**	**# INJECTED MESSAGES**	**Percentage of Injected Messages(%)**	**Attack Types**
Car-Hacking Dataset	17558462	15226830	2331517	13.28	DoS Attacks Fuzzy Attacks Spoofing the drive gear Spoofing the revolutions per minute (RPM) gauge
CAN-intrusion-dataset (OTIDS)	4613909	NA	NA	NA	DoS Attacks Fuzzy Attacks Impersonation Attacks
Survival Analysis Dataset for automobile IDS	1735840	1552526	183314	10.56	Flooding Fuzzy Malfunction
CAN-FD Intrusion Dataset	7120602	5490129	1630473	22.90	Flooding Fuzzing Malfunction
M-CAN Intrusion Dataset	2952620	2452620	500000	16.93	DoS Fuzzy
B-CAN Intrusion Dataset	Unknown	NA	NA	NA	DoS Fuzzy

TOW-IDS:	10588348	10208689	379659	3.59	Frame injection attacks PTP sync attacks Switch (MAC Flooding) attacks CAN DoS attacks CAN replay attacks

Table 2. CAVs Intrusion Datasets Abstraction from HCR Lab

8. KU-CISC2017-OTIDS
   Meanwhile, there is another dataset called KU-CISC2017-OTIDS, but since its description was totally in Korean, hereby we don't consider exploiting it furthermore.

9. CAV-KDD99
   This dataset is fully named CAV communication-based cyber-attack data set. It is a simulated CAV dataset generated following UK CAV principles and based on a UML (Unified Modeling Language)-based CAV cybersecurity framework. It simulated malicious behaviors on Sensors (LiDAR, Radar, Camera), GNSS devices, and vehicle systems (OBD, CAN-bus, power system) but not in a real vehicle. It is derived from the KDD99 dataset, which is a well-known benchmark for online intrusion or attack detection datasets. KDD99 and CAV-KDD contain 4 major types of attacks (Table 3) but 14 subtypes (Figure 15).

Dataset Name	**# MESSAGES**	**# NORMAL MESSAGES**	**# INJECTED MESSAGES**	**Percentage of Injected Messages(%)**	**Attack Types**
KDD99	5209458	1033372	4176086	80.16	Probing attacks DoS attacks User-to-Root attacks Remote-to-Local attacks
CAV-KDD	213578	135745	77833	36.44	Probing attacks DoS attacks User-to-Root attacks Remote-to-Local attacks

Table 3. KDD99 and CAV-KDD Dataset Abstraction

			10% KDD99 Training Data Set	CAV-KDD Training Data Set	10% KDD99 Testing Data Set	CAV-KDD Testing Data Set
	0	NORMAL	97278	58716	60593	47913
PROBE	1	ipsweep	1247	341	306	155
	2	nmap	231	158	84	80
DOS	3	mailbomb	/	/	5000	308
	4	neptune	107201	12281	58001	20332
	5	pod	264	40	87	45
	6	smurf	280790	199	164091	936
	7	teardrop	979	199	12	12
	8	udpstorm	/	/	2	2
U2R	9	buffer_overflow	30	5	22	22
	10	httptunnel	/	/	158	146
R2L	11	ftp_write	8	8	3	3
	12	guess_passwd	53	53	4367	1302
	13	worm	/	/	2	2
	14	xsnoop	/	/	4	4

Figure 15 Amount of sub-attack types in KDD99 and CAV-KDD

3.2 Related Research

In the past few years, there are many papers and research published in this field. Two papers related to the CAV-KDD dataset were published:

1. “Machine Learning-Based Detection for Cyber Security Attacks on Connected and Autonomous Vehicles” published in August 2020 used Naïve Bayes and J48 Decision tree machine learning algorithms to separately implement two successful models with an accuracy rate of over 99%. And J48 beat Naïve Bayes Algorithm on the time cost. It takes less than 1 second to predict malicious behaviors in the provided test dataset. (Qiyi He, 2020)

2. “In-vehicle network intrusion detection using deep convolutional neural network (DCNN)” published by Song, Hyun Min, Jiyoung Woo, and Huy Kang Kim in 2020 in South Korea utilized a deep convolutional neural network machine learning algorithm to analyze Car-Hacking Dataset. And conclusively, the prediction rate also achieved 99.8%.

3. “Intrusion Detection System for Internet of Vehicles Based on Ensemble Learning and CNN” published in 2022 by Anlun Luo in China implemented an IDS ML model with Convolutional Neural Networks as well achieved a 100% accuracy rate and the detection time from 1.0ms to 2.8ms. This research trained Car-Hacking Dataset that was mentioned above. However, this research utilized many other ML convolutional neural network (CNN) algorithms such as including Xception, VGG19, Inception, MobileNet, and DenseNet, and all of them reached a 100% accuracy rate. But sometimes when the detection rate reaches 100%, it might result in overfitting.

4. “A Deep Learning Perspective on Connected Automated Vehicle (CAV) Cybersecurity and Threat Intelligence” written by Manoj Basnet and Mohd. Hasan Ali in The University of Memphis in September 2021 focused on the Neptune and Smurf attacks in the CAV-KDD dataset and implemented an “end-to-end deep convolutional neural network Long Short-Term Memory Networks (CNN-LSTM)” ML architecture to enhance IDS of CAVs. The prediction rate also reached 99%.

5. A graduate student in the Master of Science in Internetworking program at the University of Alberta (UofA) called Kaur, Jaskiran submitted a report “Research, Implementation and Security Analysis of Connected and Autonomous Vehicles (CAVs) using Machine Learning Algorithms” in 2022. In this project, he implemented ML classification models with Naive Bayes and J48 against the CAV-K99 dataset. The accuracy rate of both algorithms is almost 95%. (Kaur, 2022)

Meanwhile, HCR Lab also held many CAVs Intrusion ML model implementation challenges. In 2019 it collected in-vehicle network traffic data of HYUNDAI Sonata, KIA Soul, and CHEVROLET Spark for supporting In-Vehicle Network Intrusion Detection Challenge. A dataset with flooding attacks, fuzzy attacks, malfunction attacks, and replay attacks was officially provides for the challengers to build their work. HCR Lab held another challenge in 2020 called “Car Hacking: Attack & Defense Challenge” as well. And another dataset that contained flooding attacks, spoofing attacks, replay attacks, and fuzzy attacks was also provided. This dataset is collected from a CAN network traffic of an Avante CN7 vehicle.

In conclusion, there has been a great amount of research in this field, but there are still many datasets and explorations required for enhancing the detect ability of IDS against many different types of attacks toward CAVs. And as with the rapid development of CAVs, the vulnerability will be more critical than before when more and more electronic adjunctions are added using CAN protocol.

4. Methodology and Implementation Procedure

4.1 Project Implementation Tools

In this project, we will utilize WEKA (Waikato Environment for Knowledge Analysis), which is a free machine learning software under the GNU General Public License. And we also use Microsoft Visual Studio 2019 to write C# programs.

4.2 Target Clarification

Contributing to IDS by implementing an ML model to detect CAN abnormal messages based on datasets mentioned in Section III.

Firstly, it is important to clarify the project target and the type of ML model. According to all the research so far in the field of Cybersecurity IDS of CAVs, this project is to build a model to predict attack-type given messages containing numerical attributes. Therefore, the model is better to be supervised learning. And since the dependent variables are discrete as either **True (is cyberattack)** or **False (is not cyberattack)**. Or perhaps multiple **cyberattack names or normal messages**. Therefore, the ML model is designated to be a **binary classification** model or a **multiclass classification model.**

4.3 Dataset selection

Secondly, it is critical to choose valid datasets among multiple choices for future research.

To check out whether a dataset is qualified for machine learning model implementation, it is significant to observe and analyze several characteristics of datasets to make sure the effectiveness of data so that the validity of the model can be guaranteed.

The following 8 characteristics would help ensure the quality of a dataset.

- **Source Reliability**
  Check out whether the datasets are from a believable organization or individual. If they were made up of unknown sources and the quality of the dataset could not be verified, all future work for the project could be meaningless.

- **Data Reliability**
  Explore whether datasets were collected legally. This is also significant to make sure the reliability of the data. Reviewing the methodology used to collect the data and the sampling techniques could be effective ways.

- **Data Format**
  Guarantee that the datasets are collected and recorded in an effective format such as “CSV”, or “pcap” format. And make sure annotated with an understandable README file. Here “pcap” format is a special file format regulated in the field of network communication. It is especially used for capturing network traffics. This step could ensure the dataset is eligible to be imported into the machine learning software or program easily for future work.

- **Data Completeness**
  Scrutinize and ensure that the dataset contains all the necessary attributes and variables that are significant for implementing the model. Make sure the cleansing of the dataset would not interfere with the research task.

- **Data Accuracy**

Check whether all the data were recorded correctly and effectively. Compare datasets with other similar datasets or official reports such as textbooks or academic reports to guarantee accuracy.

- **Data Consistency**
  This factor determines if the dataset contains any invalid or mistakes, for instance, missing values, duplicated records, outliers, etc. Usually, it could not be 100% consistent. And if the amount of inconsistent data is not critical, the dataset should be well-cleansed before being utilized for the model implementation.

According to the exploration above in Section III Related Works, these datasets are all collected from authorized individuals or organizations. But CAV-KDD99 is not generated from a real CAV but is technically derived from a general cyberattack dataset. Meanwhile, it is not able to be acquired online without permission. Therefore, it is not considered for this project. Next, according to the observation of the CAN-Intrusion Dataset (OTIDS), it lacks the required dependent variables. CAN message in the dataset did not indicate whether it was attack free or malicious. Therefore, CAN-Intrusion is not considered as well. Meanwhile, since there was already multiple successful research on the Car-Hacking dataset, we will leave it for the current status.

Finally, **M-CAN Intrusion Dataset** and **Survival Analysis Dataset for automobile IDS** are chosen as satisfied research datasets. They both came from the authorized organization HCRL and were recorded from real CAVs. They contain enough consistent data, and the file format for both is eligible. The reason the B-CAN dataset is not considered is that the dataset is highly similar to M-CAN dataset in attack types and data characteristics. And the OTIDS was explored by other researchers. Meanwhile, The Car-Hacking dataset is already researched. And CAN-FD Intrusion dataset contains data collected from CAN with the Flexible Data, and the protocol structure is different from CAN2.0A or CAN2.0B.

4.4 M-CAN Raw Datasets Data Analysis

The target dataset is made up of 3 separate datasets, the *g80_mcan_ddos_data* dataset*, the g80_mcan_fuzzing_data* dataset*,* and *the g80_mcan_normal_data* dataset. They are all collected via the On-Board Diagnostics-II (OBD-II) port from a real autonomous vehicle. To make the research clear, OBD-II is an on-board self-diagnostic equipment required on vehicles that allow telematics devices to process and convert vehicular information into digital data. It is used for detecting real-time vehicle speed, and engine revolutions. These datasets focus on continuous Controller Area Network (CAN) data transportation. The *g80_mcan_normal_data dataset* contains normal CAN frames during a 56-minute normal driving (Figure 16), The other two datasets contain malicious attack records that are injected into the local computer due to the concern of actual traffic danger. And it is reasonable to inject not on a driving vehicle because it is

illegal and dangerous. But the conditions of injected attacks are the same as the real attacks on a vehicle. (M-CAN INTRUSION DATASET, 2020)

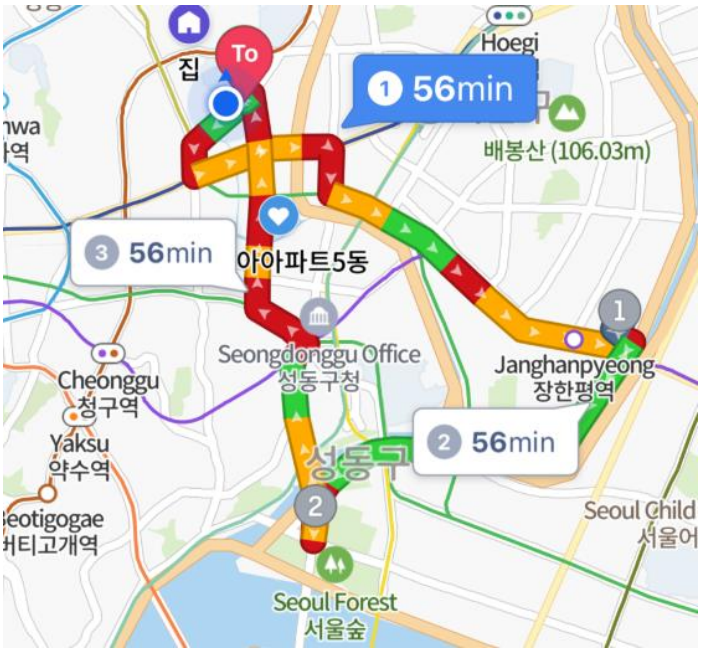

Figure 16 Route of the vehicle for collecting the *g80_mcan_normal_data* dataset

These datasets have 5 consistent attributes shown in Table 4 and Figure 12:

# attribute	Attribute Name	Value	Value Description(unit)
1	Timestamp	millisecond	instant time of the message (s)
2	CAN ID	Hexadecimal	identifier of CAN message in HEX (ex. 043f)
3	DLC	[0~8]	number of data bytes, from 0 to 8
4	DATA Payload	[0~7][Hex]	Data Payload (byte)
5	Label	1 or 0	1 = Injected; 0 = Normal

Table 4 M-CAN dataset attributes

1	Timestamp	ID	DLC	Payload	label
1048459	1953.426	1.00E+01	8	F1 31 35 0	0
1048460	1953.427	000002BB	8	36 6A 02 1	0
1048461	1953.427	1.00E+02	8	00 00 00 0	0
1048462	1953.427	000002BC	8	ED C2 8D (	0
1048463	1953.427	000002BD	8	F0 0A C2 2	0
1048464	1953.428	000002BE	8	FE FD FB 4	0
1048465	1953.428	0000042B	8	00 00 4A 0	0
1048466	1953.432	61	8	74 DE 21 0	0
1048467	1953.432	62	8	02 02 00 0	0
1048468	1953.433	63	8	00 00 05 0	0
1048469	1953.433	64	8	00 00 00 0	0
1048470	1953.433	000003C1	8	B7 D0 10 0	0
1048471	1953.433	000003C2	8	41 00 10 4	0
1048472	1953.434	156	8	07 C6 A6 1	0
1048473	1953.435	157	8	00 00 00 0	0
1048474	1953.436	161	8	C8 8F F9 0	0
1048475	1953.436	162	8	FF FC 01 0	0
1048476	1953.436	000001A1	8	03 9B F9 C	0
1048477	1953.436	000001A2	8	00 00 00 0	0
1048478	1953.437	000001A3	8	FF F3 3F 0	0
1048479	1953.437	000001A4	8	FE 07 00 0	0
1048480	1953.437	000000A1	8	E4 4E 21 E	0

Figure 17 Manifestation of g80_bcan_ddos_data dataset

4.5 Attacks in M-CAN Dataset

4.5.1 DDoS Attack

The malicious attacks are messages containing “0x00000000” CAN ID, which owns the highest priority (Figure 18).

486511	1684.42	0	8	00 00 00 0	1
486512	1684.421	0	8	00 00 00 0	1
486513	1684.421	0	8	00 00 00 0	1
486514	1684.421	0	8	00 00 00 0	1
486515	1684.421	0	8	00 00 00 0	1
486516	1684.422	0	8	00 00 00 0	1
486517	1684.422	0	8	00 00 00 0	1
486518	1684.422	0	8	00 00 00 0	1
486519	1684.422	0	8	00 00 00 0	1
486520	1684.423	0	8	00 00 00 0	1
486521	1684.423	0	8	00 00 00 0	1
486522	1684.423	0	8	00 00 00 0	1

Figure 18 Manifestation of recorded DDoS attacks in M-CAN

4.5.2 Fuzzing Attack

In digital format, the fuzzing attack performs as a message with a random ID and payload (figure 19). It cannot be detected directly with human eyes. But actually,

the random message payload disobeys the rule of specific CAN ID.

For example, in Figure 20, there does not exist a real CAN ID 0x00000D9E. All message from this CAN ID is malicious. And in figure 21, two malicious attacks from CAN ID 0x00000A74 contains different DLC and different length of the payload. And Figure 22 compares regular messages with malicious messages from the same CAN ID. Even though the attack generated a legal CAN ID by randomization, the DLC and payload violate the rule of Node with CAN ID 183. Hence, it could cause a malfunction of the vehicle in a real situation.

5	673	5	47 E9 FD 05 CC	1	
5	000002D2	8	00 C0 00 00 1E 40 00 00	0	
5	00000A38	8	2B 06 1F 60 9F DB 76 9E	1	
5	000008B0	4	29 71 20 AA	1	
5	00000B76	8	30 00 6D 15 BF C8 2F 97	1	
5	00000D9E	5	F9 CB 22 F5 44	1	
5	0000019E	4	E6 8C 86 61	1	
5	00000C72	5	49 D9 3B 6F 04	1	
5	00000C82	5	31 89 F5 5C 82	1	
5	0000094F	6	1A 50 3E B9 D5 31	1	
5	000005DB	7	85 64 5F 41 38 FF 30	1	
5	00000A43	8	74 9F 5E EB 64 7B 2F BF	1	
5	000006D5	5	56 01 D4 6B 06	1	

Figure 19 Digital manifestation of fuzzing attacks in M-CAN

528137	2823.988	00000D9E	4	4B 58 4C F1	1

Figure 20 Digital manifestation of fuzzing attack with invalid CAN ID

885157	1593.672	00000A74	3	CB 3D 04	1
885158	1782.146	00000A74	7	BE 3D AD F1 E5 7	1
885159	1831.913	00000A74	6	55 3F 5B 89 3D D	1
885160	1896.427	00000A74	8	2C 9A 2F 61 0F E	1
885161	1915.726	00000A74	8	6A 44 EB 05 0E A	1
885162	2179.954	00000A74	5	63 DD 8B 7F 91	1
885163	2241.131	00000A74	6	C0 41 6D D3 18 6	1
885164	2241.14	00000A74	7	1A E8 49 C8 98 A	1
885165	2321.814	00000A74	8	09 27 8C 01 AB 1	1
885166	2321.874	00000A74	6	36 8B B5 51 0F A	1
885167	2377.143	00000A74	3	F5 53 32	1
885168	2399.39	00000A74	5	6E 57 33 01 06	1
885169	2408.198	00000A74	3	BC 6B AF	1
885170	2411.532	00000A74	4	EE F2 CB 20	1
885171	2467.934	00000A74	5	13 F8 A0 7C 1D	1
885172	2518.781	00000A74	8	35 87 DB 94 77 7	1
885173	2641.834	00000A74	4	04 B0 28 5F	1
885174	2660.69	00000A74	8	4F 4C D6 24 91 4	1
885175	2662.574	00000A74	5	02 4A 76 F5 F4	1

Figure 21 Digital manifestation of fuzzing attacks with different DLC and payload length

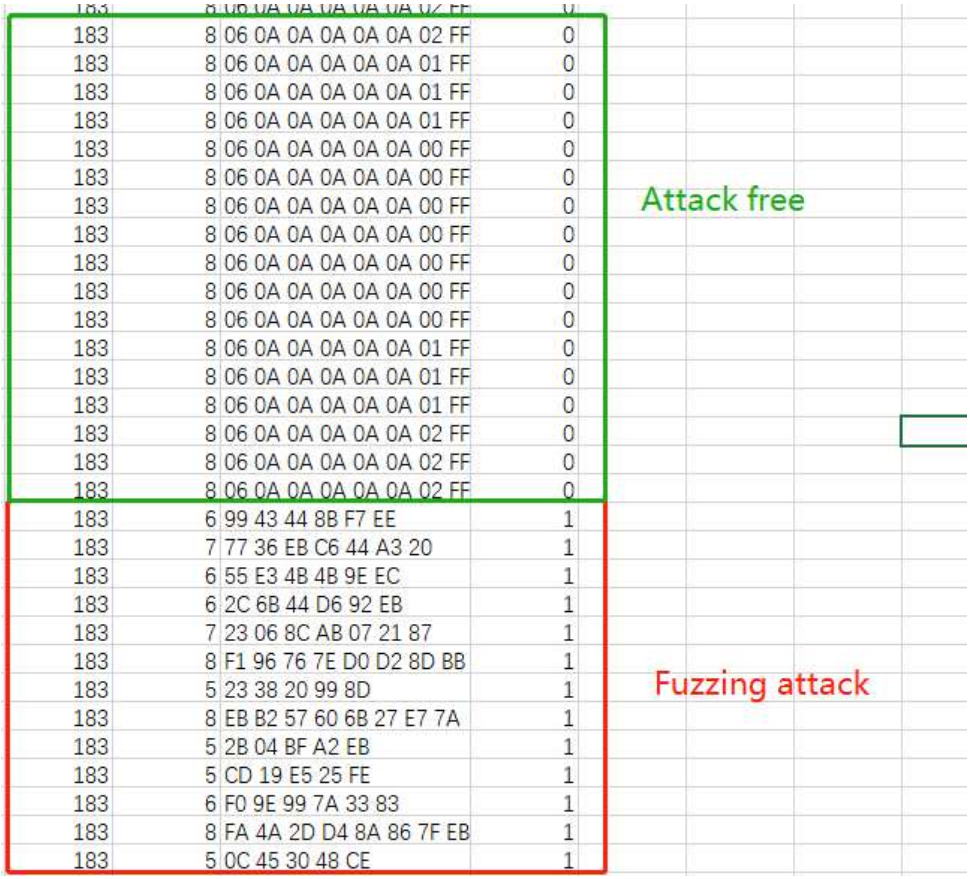

Figure 22 Attack-free messages vs fuzzing attacks with the same CAN ID.

4.6 M-CAN Data Preprocessing

Firstly, since these three datasets are separated, we need to mix them for the implementation of one ML model.

To do this, I implemented a C# program called “CombineM-CAN Dataset” in Microsoft Visual Studio 2019 (Figures 23.1 and 23.2). The whole C# project is in the “CombineM-CANDataset” folder.

The essential idea is firstly adjusting the label of 1s to 2s in g80_mcan_fuzzing_data so that after the combination of CSV files, the label of fuzzing attack and DDoS attack can be still distinct. After the adjustment, implement another function to scan all the data samples and copy them into the “combined_dataset.txt” file. Due to the inaccessibility of the original dataset file in function, I used Bash to convert three CSV files into txt files so that they can be accessed via the C# project before implementation. And to monitor the combination procedure for avoiding mistakes, I did multiple real-time row counts in the code.

```
static void Main(string[] args)
{
    string normal_data = @"E:\Capstone Project\used M-CAN Intrusion Dataset\g80_mcan_normal_data_new.txt";
    string fuzzing_data = @"E:\Capstone Project\used M-CAN Intrusion Dataset\g80_mcan_fuzzing_data.txt";
    string ddos_data = @"E:\Capstone Project\used M-CAN Intrusion Dataset\g80_mcan_ddos_data.txt";
    int normal_row = CountRowsInFile(normal_data);
    int fuzzing_row = CountRowsInFile(fuzzing_data);
    int ddos_row = CountRowsInFile(ddos_data);
    Console.WriteLine("The number of normal rows in original normal dataset:" + normal_row);
    Console.WriteLine("The number of ddos rows in original ddos dataset:" + ddos_row);
    Console.WriteLine("The number of fuzzing rows in original fuzzing dataset:" + fuzzing_row);

    string new_fuzzing_data = @"E:\Capstone Project\used M-CAN Intrusion Dataset\g80_mcan_fuzzing_data_without
    string new_ddos_data = @"E:\Capstone Project\used M-CAN Intrusion Dataset\g80_mcan_ddos_data_without_attri
    DeleteAndSaveFirstRow(fuzzing_data, new_fuzzing_data);
    DeleteAndSaveFirstRow(ddos_data, new_ddos_data);
    //int new_normal_row = CountRowsInFile(normal_data);
    //int new_fuzzing_row = CountRowsInFile(new_fuzzing_data);
    //int new_ddos_row = CountRowsInFile(new_ddos_data);
    //Console.WriteLine("The number of normal rows:" + new_normal_row);
    //Console.WriteLine("The number of ddos rows:" + new_ddos_row);
    //Console.WriteLine("The number of fuzzing rows:" + new_fuzzing_row);

    // Replace the dependent value 1.0 with 2.0 as fuzzing attack for each sample
    ModifyLastThreeCharacters(new_fuzzing_data);
    string combined_data = @"E:\Capstone Project\used M-CAN Intrusion Dataset\g80_mcan_combined_data.txt";

    // Combine three datasets into combined_data file
    CombineTextFiles(normal_data, new_fuzzing_data, new_ddos_data, combined_data);

    string type_ddos = "1.0";
    string type_fuzzing = "2.0";
```

Figure 23.1 main function of C# project

```
string type_ddos = "1.0";
string type_fuzzing = "2.0";
// 0.0 and 0 both represent normal type
string type_normal = "0.0";
string type_normal2 = ",0";

int normal_count = CountMessages(combined_data, type_normal);
normal_count += CountMessages(combined_data, type_normal2);
int ddos_count = CountMessages(combined_data, type_ddos);
int fuzzing_count = CountMessages(combined_data, type_fuzzing);
int total_count = CountMessages(combined_data);

Console.WriteLine("The number of normal messages in combined file:" + normal_count);
Console.WriteLine("The number of ddos messages in combined file:" + ddos_count);
Console.WriteLine("The number of fuzzing messages in combined file:" + fuzzing_count);
Console.WriteLine("The number of total messages in combined file:" + total_count);
```

Figure 23.2 main function of C# project

And the result after the combination is shown in figure 24.

```
The number of normal messages:2452620
The number of ddos messages:400000
The number of fuzzing messages:100000
The number of total messages:2952621
Error messages: 0
```

Figure 24 results of the combined dataset.

According to the original description of the dataset, the DDoS attack was injected every

0.25 millisecond for a total of 4 seconds, which means there should be 400000 DDoS attacks. And the fuzzing attack was injected every 0.1 milliseconds for 1 second. And the number of fuzzing attacks should be 100000 in this dataset. Therefore, the preprocessed result (Figure 24) shows that the number of DDoS attacks, fuzzing attacks, and normal messages are all correct.

Next, this project utilizes Weka 3.8.6 as the academic tool for implementing the ML model. After the data preprocess manually, the combined dataset was successfully opened in Weka. The list of independent attributes and dependent attributes is shown in Figure 26. And the distribution of attack type is shown in Figure 27.

Figure 26 list of attributes in the dataset

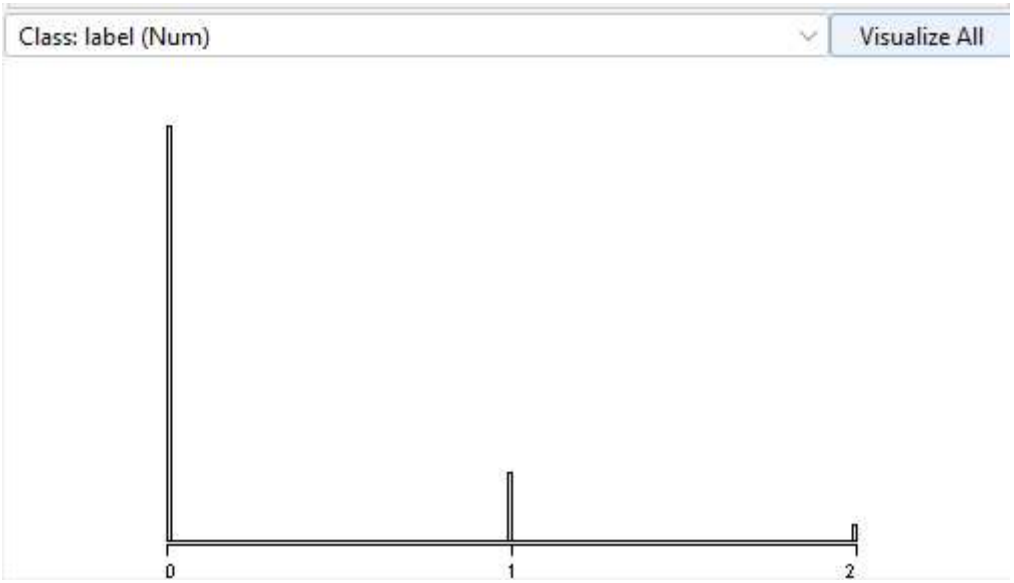

Figure 27 Distribution of two attacks and attack-free messages

Next, according to the performance of attack type correlated to different independent attributes. The Timestamp attribute should be irrelevant to the prediction of message type, it could be removed first (Figure 28).

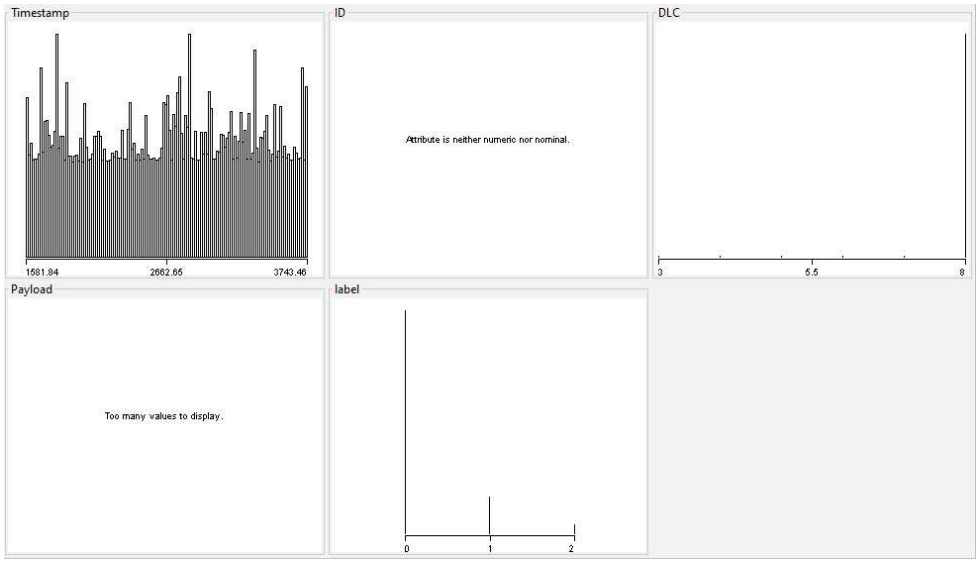

Figure 28 Raw Data attributes correlation to Label

Even though the attributes are numerical and in hexadecimal format and some are categorical, they do not indicate any numerical correlation in machine learning. Therefore, the data should be preprocessed additionally.

Therefore, we choose the “StringToNominal” unsupervised attribute Filter on the CAN ID attribute and Payload attributes and the “NumericalToNominal” unsupervised attribute Filter on the DLC attribute. And finally, we apply the “NumericalToNominal” unsupervised attribute Filter on the label attribute to convert the attack type into a discrete categorical form.

After the preprocessing, all data are nominal and are eligible for future classification. (Figure 29)

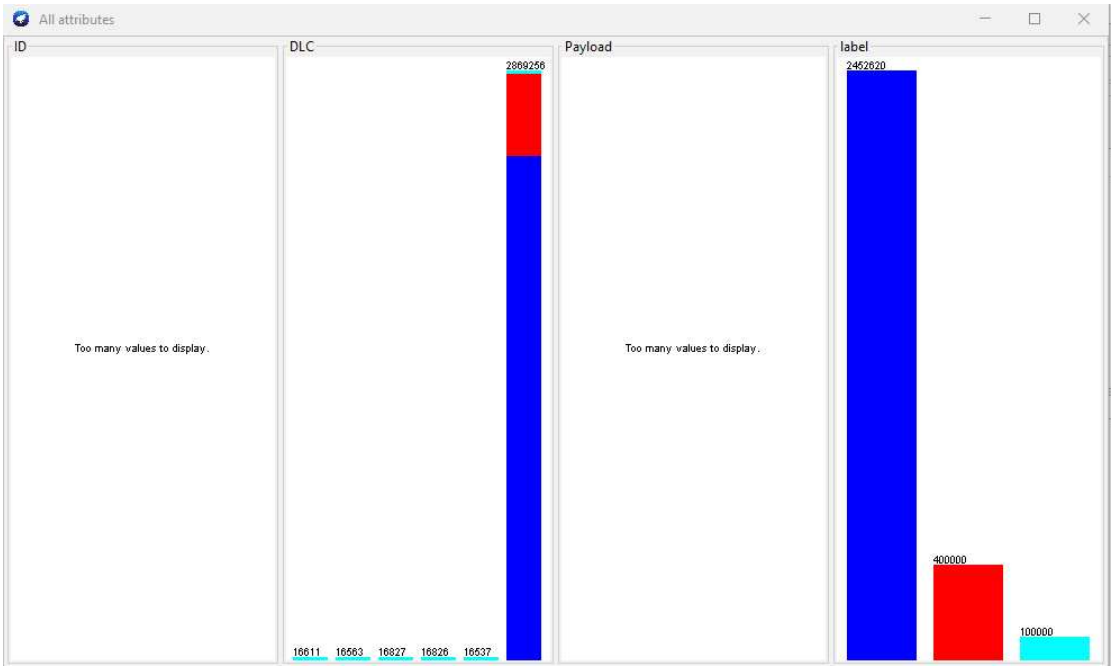

Figure 29 Visualization of attributes in nominal format correlated to Label

Next, since the dataset is combined manually by the program. The DDoS attacks are located in the middle of the dataset and the Fuzzing attacks spread at the end of the dataset. We have to randomize the dataset before the dataset splits during the Classification model implementation procedure.

Therefore, to shuffle the dataset, we apply the “Randomize” unsupervised instance Filter (Figure 30.1 and 30.2) 5 times and check the outcome via the “Remove Percentage” unsupervised instance filter.

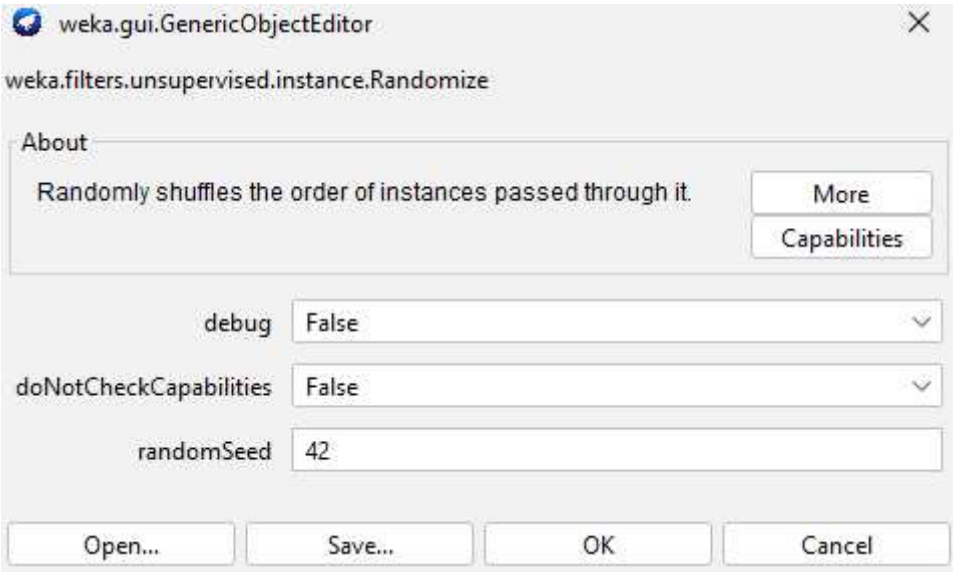

Figure 30.1 Randomize Filter Configuration

Figure 30.2 Randomize Logs

And to examine the randomized result, we next use the "RemovePercentage" unsupervised instance filter and manually split the dataset in the middle to see the number of attack types distribution. The result is shown in Figure 31. And after examination, we undo the result for the completeness of future research.

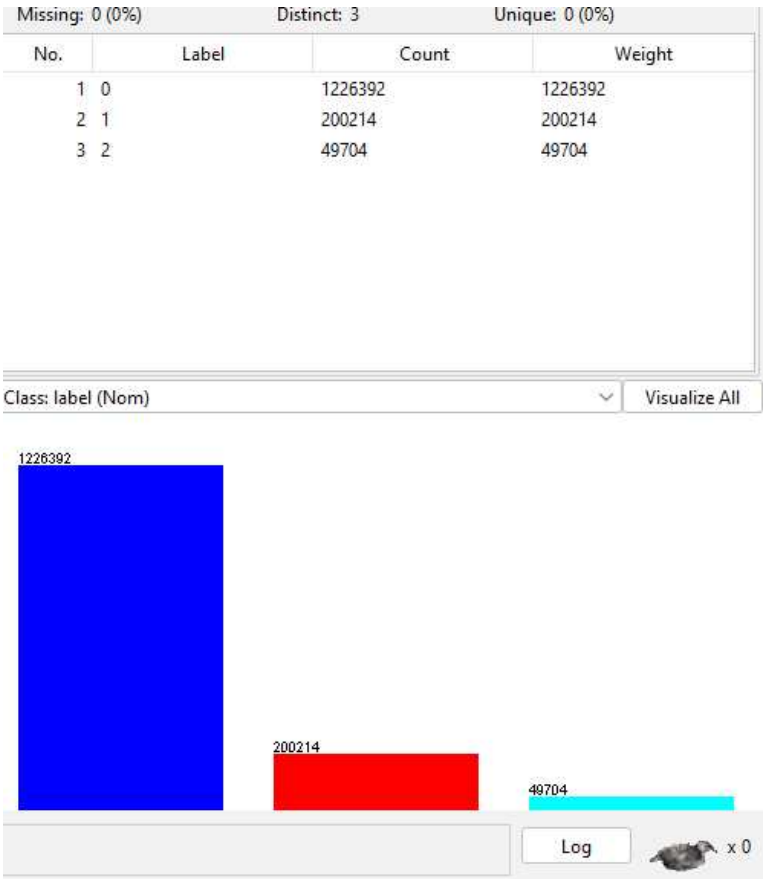

Figure 31 attack type distribution after removal.

4.7 M-CAN Dataset Post-preprocessed Analysis

Firstly, we visualize sample distribution to check the correlation among different attributes to the dependent result. According to Figures 32, 33 and 34, and 35, we can find that the DDoS attacks data are highly concentrated on CAN ID 000. The DLC and payload of DDoS attacks also highly focus on one exact position on the plots. It proves the phenomenon of DDoS attacks described before. But if we scrutinize attack-free data and Fuzzing data, the plots visualize many characteristics that are hidden behind the messages. According to figure 32 and 35, the regulated CAN attack-free messages shows that only a minority of CAN IDs are valid, and they are highly focused on a continuous range. And since the normal CAN messages obey standard rules for different CAN IDs, the visualized output also shows the regulated distribution. And all attack-free messages have DLC with 8. But Fuzzing attacks contain CAN IDs, DLC, and Payload that spread everywhere.

Figure 32 Visualization of CAN ID vs Attack Types

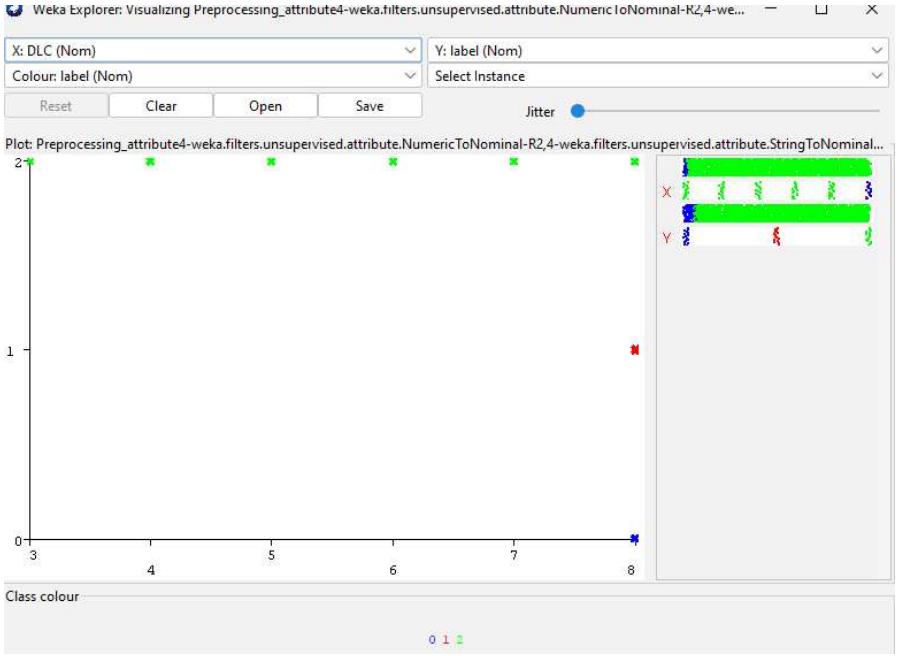

Figure 33 Visualization of DLC vs Attack Types

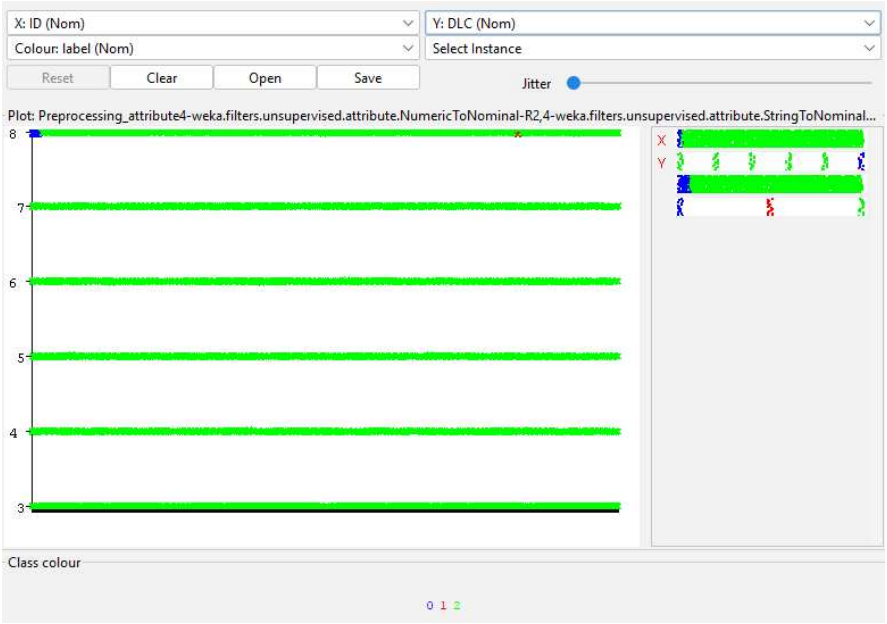

Figure 34 ID vs DLC vs Attack Types

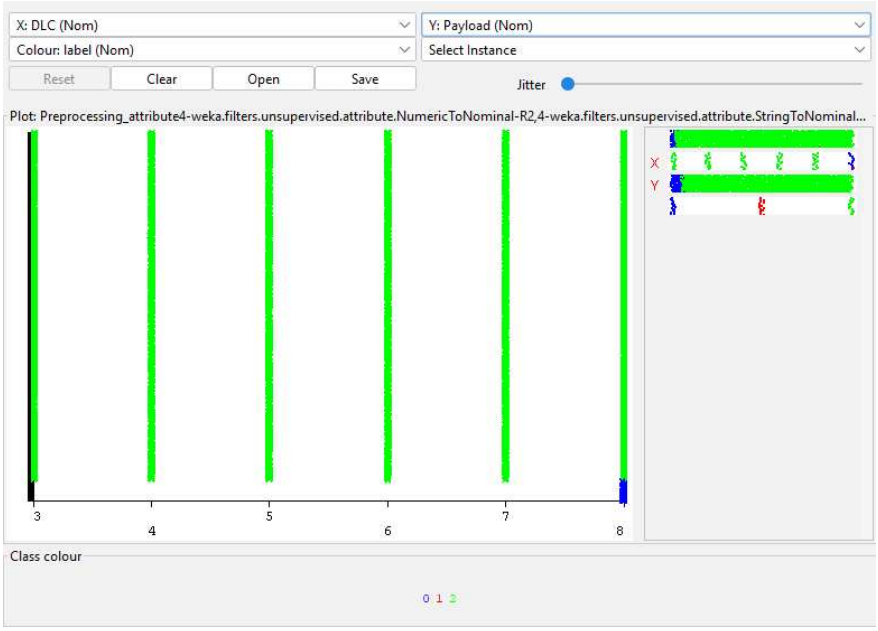

Figure 35 DLC vs Payload vs Attack Types

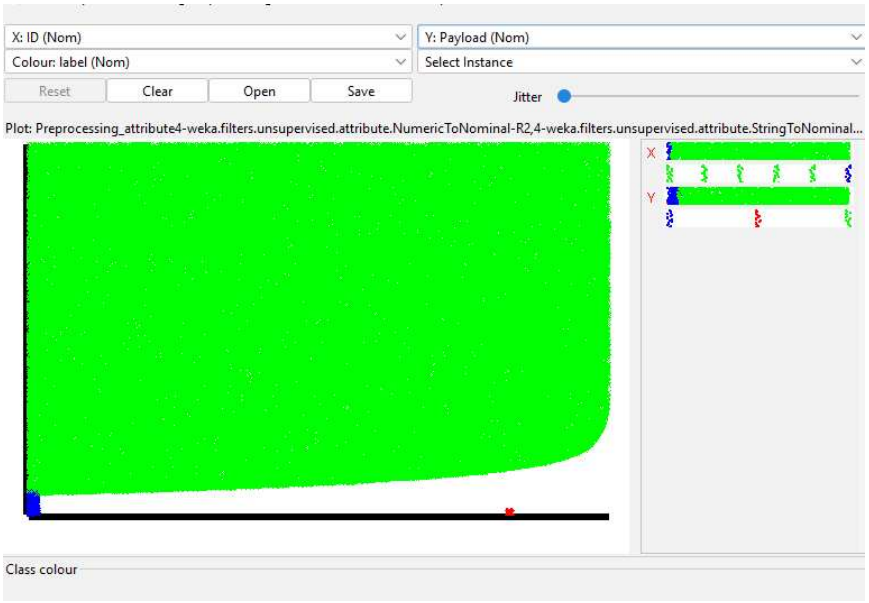

Figure 36 CAN ID vs Payload vs Attack Types

4.8 M-CAN Dataset Split

And we will attempt 2dataset-split methods for improving model performance.

4.8.1 10-Folds Cross Validation

In 10-Folds cross-validation, the dataset is split into 10 smaller subsets, called "folds." The model is trained on several of these folds and evaluated on the

remaining fold, and this process is repeated multiple times, with different combinations of folds used each time. The performance metric, such as accuracy or mean squared error, is averaged across all the iterations of training and testing.

4.8.2 Percentage split at 60% of the dataset

In Percentage split, the dataset is simply split into two subsets, the first 60% of data will be categorized as a training set and the last 40% of data will be used as a test set.

And the amounts of each attack type in the M-CAN training and testing sets are shown in Table 5.

Attack name	Training set	Test set
Attack-free	1471396	981224
DDoS attack	240424	159576
Fuzzing attack	59752	40248

Table 5 Amounts of attacks in the training set and test set

4.9 M-CAN Model Training and Evaluation

This project explores the J48 decision tree and Bayes Neural Network utilization on the M-CAN dataset.

4.9.1 J48 Decision Tree Classifier

J48 decision tree utilizes the C4.5 technique implemented in Java to achieve the decision tree algorithm.

Firstly, the J48 decision tree with the following configurations (Figure 37) is applied to build the Classification model. Since the attributes number is only 4 including dependent attribute and the phenomenon of attacks are obvious and easy to be comprehended. Therefore, the number of folds is maintained in default and the number of minimal number objects is saved as 2 to test first. Since batch prediction is not performed, so the batchSize is not modified.

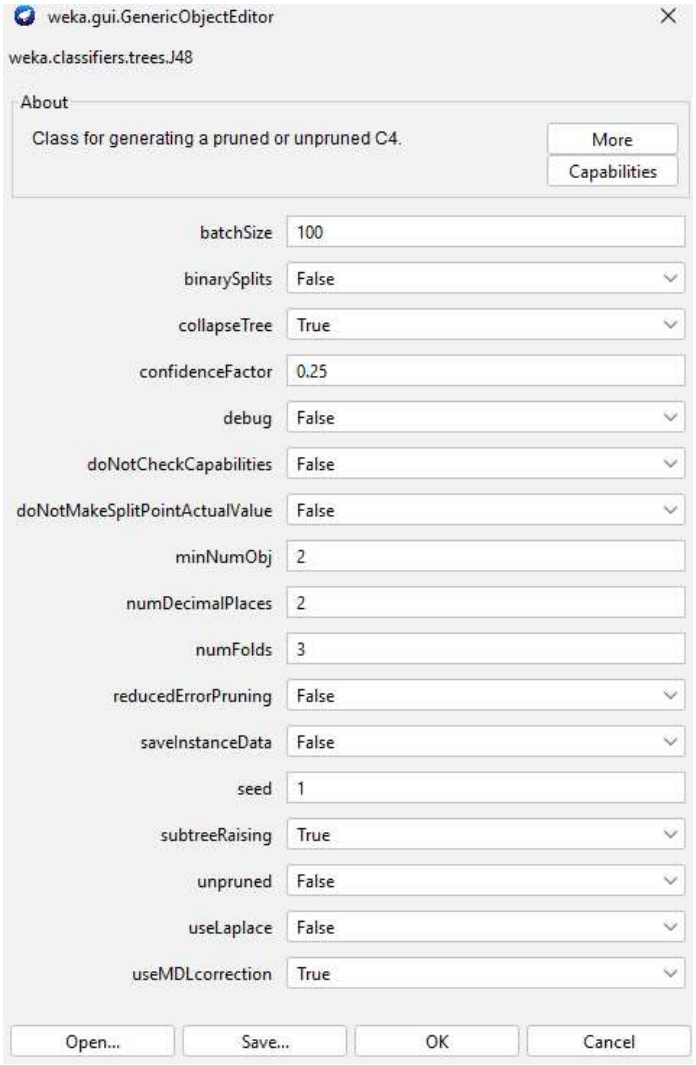

Figure 37 J48 Decision Tree 1 Configuration

```
Correctly Classified Instances     2952215               99.9863 %
Incorrectly Classified Instances       405                0.0137 %
Kappa statistic                          0.9995
Mean absolute error                      0.0001
Root mean squared error                  0.0096
Relative absolute error                  0.0725 %
Root relative squared error              3.0691 %
Total Number of Instances          2952620

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.001    1.000      1.000    1.000      1.000    1.000     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 0.996    0.000    1.000      0.996    0.998      0.998    0.999     0.998     2
Weighted Avg.    1.000    0.001    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

       a       b       c   <-- classified as
 2452620       0       0 |       a = 0
       0  400000       0 |       b = 1
     401       4   99595 |       c = 2
```

Figure 38.1 J48 Decision Tree 1 with 10-fold cross-validation split performance

```
Time taken to build model: 4.63 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 1.19 seconds

=== Summary ===

Correctly Classified Instances     1180875               99.9854 %
Incorrectly Classified Instances       173                0.0146 %
Kappa statistic                          0.9995
Mean absolute error                      0.0001
Root mean squared error                  0.0099
Relative absolute error                  0.0761 %
Root relative squared error              3.1715 %
Total Number of Instances          1181048

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 0.996    0.000    1.000      0.996    0.998      0.998    0.999     0.998     2
Weighted Avg.    1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000

=== Confusion Matrix ===

      a      b      c   <-- classified as
 980966      0      0 |      a = 0
      0 160242      0 |      b = 1
    172      1  39667 |      c = 2
```

Figure 38.2 J48 Decision Tree 1 model with 60% split performance

The two results of the J48 decision tree with 2 minimum number of instances per leaf and 3 number of folds performance are shown in Figures 38.1 and 38.2. And by using a stopwatch, the cross-validation takes 96s for the whole process of building and testing, but J48 only takes 25s.

Next, some small adjustments are applied to the J48 classifier that the minNumObj to 10change numFolds to 10, and the performance are shown in Figure 39.1 and 39.2. Or only the minNumObj is changed to 3 and result in the output shown in Figure 40.1 and 40.2. Meanwhile, if only the numFolds are changed to 4, the result is depicted in Figure 41.1 and Figure 41.2. Conclusively, The outputs all show equivalent or worse performance. Therefore, an overcomplicated tree structure does not fit this dataset. What needs to be concerned about in the performance output is that there are hundreds of fuzzing attacks are recognized as normal messages in both models.

```
Time taken to build model: 3.58 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances     2951256               99.9538 %
Incorrectly Classified Instances      1364                0.0462 %
Kappa statistic                          0.9984
Mean absolute error                      0.0005
Root mean squared error                  0.0175
Relative absolute error                  0.2839 %
Root relative squared error              5.6359 %
Total Number of Instances          2952620

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.003    0.999      1.000    1.000      0.998    0.999     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 0.986    0.000    1.000      0.986    0.993      0.993    0.997     0.990     2
Weighted Avg.    1.000    0.002    1.000      1.000    1.000      0.998    0.999     0.999

=== Confusion Matrix ===

       a       b       c   <-- classified as
 2452620       0       0 |       a = 0
       0  400000       0 |       b = 1
    1342      22   98636 |       c = 2
```

Figure 39.1 J48 Decision Tree model 2 with 10-fold cross-validation performance

```
=== Summary ===

Correctly Classified Instances     1180514               99.9548 %
Incorrectly Classified Instances       534                0.0452 %
Kappa statistic                          0.9984
Mean absolute error                      0.0006
Root mean squared error                  0.0174
Relative absolute error                  0.2896 %
Root relative squared error              5.5752 %
Total Number of Instances          1181048

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.003    0.999      1.000    1.000      0.998    0.999     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 0.987    0.000    1.000      0.987    0.993      0.993    0.996     0.990     2
Weighted Avg.    1.000    0.002    1.000      1.000    1.000      0.998    0.999     0.999

=== Confusion Matrix ===

      a      b      c   <-- classified as
 980966      0      0 |      a = 0
      0 160242      0 |      b = 1
    525      9  39306 |      c = 2
```

Figure 39.2 J48 Decision Tree model 2 with 60% split performance

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances     2952094               99.9822 %
Incorrectly Classified Instances       526                0.0178 %
Kappa statistic                          0.9994
Mean absolute error                      0.0002
Root mean squared error                  0.0109
Relative absolute error                  0.0955 %
Root relative squared error              3.4983 %
Total Number of Instances          2952620

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000     0
                1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                0.995    0.000    1.000      0.995    0.997      0.997    0.999     0.997     2
Weighted Avg.   1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000

=== Confusion Matrix ===

       a       b       c   <-- classified as
 2452620       0       0 |       a = 0
       0  400000       0 |       b = 1
     514      12   99474 |       c = 2
```

Figure 40.1 J48 Decision Tree model 3 with 10-fold cross-validation performance

```
Time taken to build model: 4.99 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 1.08 seconds

=== Summary ===

Correctly Classified Instances     1180861               99.9842 %
Incorrectly Classified Instances       187                0.0158 %
Kappa statistic                          0.9995
Mean absolute error                      0.0002
Root mean squared error                  0.0103
Relative absolute error                  0.0829 %
Root relative squared error              3.2975 %
Total Number of Instances          1181048

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000     0
                1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                0.995    0.000    1.000      0.995    0.998      0.998    0.999     0.998     2
Weighted Avg.   1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000

=== Confusion Matrix ===

      a      b      c   <-- classified as
 980966      0      0 |      a = 0
      0 160242      0 |      b = 1
    186      1  39653 |      c = 2
```

Figure 40.2 J48 Decision Tree model 3 with 60% split performance

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances     2952215               99.9863 %
Incorrectly Classified Instances       405                0.0137 %
Kappa statistic                          0.9995
Mean absolute error                      0.0001
Root mean squared error                  0.0096
Relative absolute error                  0.0725 %
Root relative squared error              3.0691 %
Total Number of Instances          2952620

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                1.000    0.001    1.000      1.000    1.000      1.000    1.000     1.000     0
                1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                0.996    0.000    1.000      0.996    0.998      0.998    0.999     0.998     2
Weighted Avg.   1.000    0.001    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

      a       b       c   <-- classified as
 2452620       0       0 |       a = 0
       0  400000       0 |       b = 1
     401       4   99595 |       c = 2
```

Figure 41.1 J48 Decision Tree model 3 with 10-fold cross-validation performance

```
Time taken to build model: 4.75 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 1.02 seconds

=== Summary ===

Correctly Classified Instances     1180875               99.9854 %
Incorrectly Classified Instances       173                0.0146 %
Kappa statistic                          0.9995
Mean absolute error                      0.0001
Root mean squared error                  0.0099
Relative absolute error                  0.0761 %
Root relative squared error              3.1715 %
Total Number of Instances          1181048

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000     0
                1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                0.996    0.000    1.000      0.996    0.998      0.998    0.999     0.998     2
Weighted Avg.   1.000    0.001    1.000      1.000    1.000      0.999    1.000     1.000

=== Confusion Matrix ===

      a      b      c   <-- classified as
 980966      0      0 |      a = 0
      0 160242      0 |      b = 1
    172      1  39667 |      c = 2
```

Figure 41.2 J48 Decision Tree model 3 with 60% split performance

4.9.2 Naïve Bayes Classifier

Next, the NaïveBayes classifier is utilized to build the model (Figure 42). The performance is shown in Figures 43.1 and 43.2. The performance shows greater accuracy and a lower FP rate than J48 overall. And the procedure time it takes to build a model with the percentage split is only 4s and for cross-validation is 29s approximately. Therefore, it is more efficient to build a Naïve Bayes model.

However, the test time it takes for Naïve Bayes is longer than J48. As we can see from Figures 38.2 and 40.2, the Naïve Bayes approximately need extra 0.2s to finish the test on 1.18m frames.

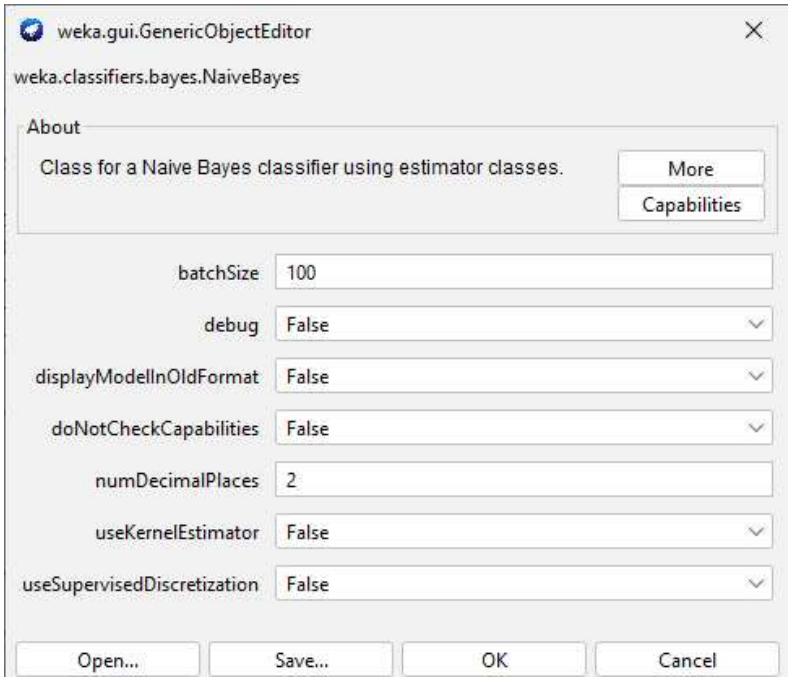

Figure 42 NaïveBayes Configurations

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances     2952403               99.9927 %
Incorrectly Classified Instances       217                0.0073 %
Kappa statistic                          0.9997
Mean absolute error                      0.0005
Root mean squared error                  0.01
Relative absolute error                  0.2715 %
Root relative squared error              3.2156 %
Total Number of Instances          2952620

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 0.998    0.000    1.000      0.998    0.999      0.999    1.000     1.000     2
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

       a       b       c   <-- classified as
 2452620       0       0 |       a = 0
       0  400000       0 |       b = 1
     214       3   99783 |       c = 2
```

Figure 43.1 NaïveBayes model with 10-fold cross-validation performance

```
=== Evaluation on test split ===

Time taken to test model on test split: 1.27 seconds

=== Summary ===

Correctly Classified Instances     1180963               99.9928 %
Incorrectly Classified Instances        85                0.0072 %
Kappa statistic                          0.9998
Mean absolute error                      0.0008
Root mean squared error                  0.0133
Relative absolute error                  0.4102 %
Root relative squared error              4.2599 %
Total Number of Instances          1181048

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 0.998    0.000    1.000      0.998    0.999      0.999    1.000     1.000     2
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

      a      b      c   <-- classified as
 980966      0      0 |      a = 0
      0 160242      0 |      b = 1
     84      1  39755 |      c = 2
```

Figure 43.2 NaïveBayes model with 60% split performance

5. Discussion

5.1 Hardware Requirement for ML Model Building

In this project, two 8GB RAM chips were used for extending the RAM to 16GB to support machine learning model implementation. In fact, the device's total RAM does not directly indicate available RAM to the project implementation since the device itself requires part of RAM to function normally. For example, 8GB RAM actually only supports 2GM JVM maximal memory for machine learning model implementation. And more than 8GB usage of machine learning model implementation could cause a memory exception under a 16 GB RAM environment (Figure 58). Even though the available memory maximum could be adjusted to larger, when RAM usage reaches 100%, it would be extremely slow and almost stop the whole device.

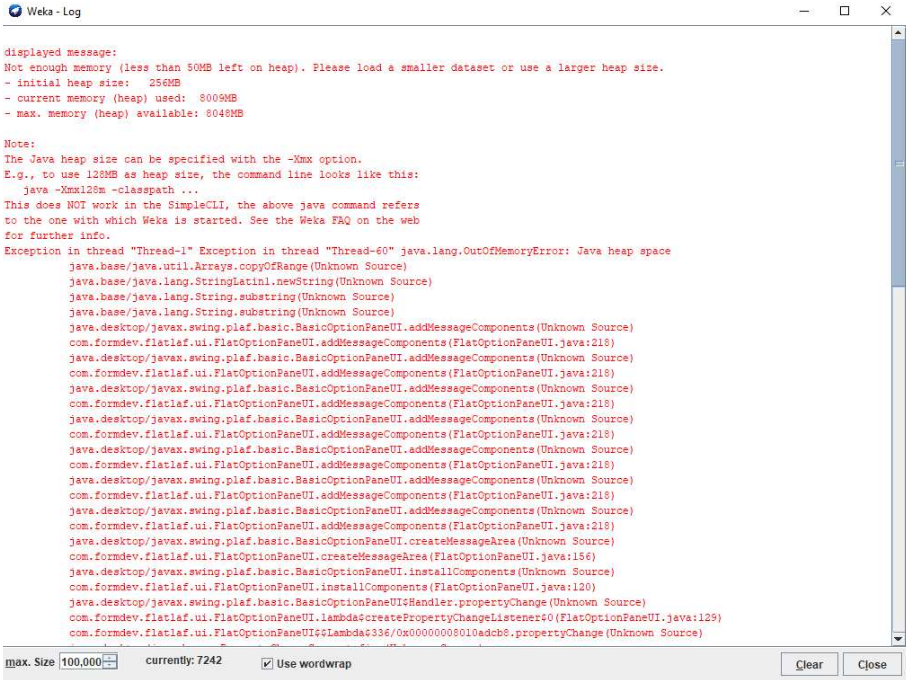

Figure 58 Memory Usage Exception in Weka

5.2 Balance between Time Cost and Accuracy

In "Machine Learning-Based Detection for Cyber Security Attacks on Connected and Autonomous Vehicles", the author mentioned "With almost the same accuracy, Naive Bayes needed a longer time to identify the attacks and, thus, Decision Tree was more efficient for CAV cyber security." (Qiyi He, 2020) In this article, the author met the same situation that the balance between the time cost to predict and accuracy must be made. And the fact that the balance between the time cost to predict and the accuracy rate is imperative is true. But the conclusion might depend on real situations.

Firstly, each SAE J1939 (the standard CAN message format) Message, takes 0.54 msec @ 250kbps and 0.27 msec @ 500 kbps to be transported. It indicates roughly 1852 messages are transported in 1s theoretically at 250kbps and 3704 at 500kbps.

$$1s/0.54ms \approx 1852 \qquad 1s/0.27ms \approx 3704$$

And average, in this real-world attack-free dataset, there are 817540 messages were collected from approximately 1581 seconds to 3743 seconds, which indicates 378 messages were sent per 1 second.

$$817540 \text{ messages} / (3743s - 1581s) \approx 378 \text{ message/s}$$

Therefore, more realistically, if the J48 decision tree is utilized, for every 1 second, it takes only 0.000406 seconds to predict one second real-time CAN data communication

as calculation shows below:

1.27s/1180875 messages ≈1.075 microseconds/message

378 * 1.075 microseconds ≈ 406.498766 microseconds ≈ 0.000406 seconds

And if we use Naïve Bayes,

1.02s/1180963 messages ≈ 863.7 nanoseconds/message

378 * 863.7 nanoseconds ≈ 326.4786 microseconds ≈ 0.000326 seconds
0.000406s - 0.000326s = 0.00008s

And the difference is only 0.00008s. Such an extremely small time-cost difference is less important than the FP (false positive) accuracy rate. And it is much more critical if the prediction goes wrong, especially when malicious messages are predicted to be attack-free.

But it also depends on how the design of IDS is in a more realistic situation. If the IDS configures multiple individual programs with different ML models to detect bus networks separately. In case the predicted amount for each model is small, then the accuracy is crucial and the gap between time cost could be ignored. Since one undetected malicious message could cause terrible aftermath. But if the IDS system is designated with just one comprehensive ML model to detect all CAN bus messages, the prediction time cost could still vary stupendously.

6. Future work

6.1 ML model upgrade to detect more types of cyberattacks

Currently, the number of available datasets for this field is still not big enough or contains more complicated kinds of attacks. But it might be because many datasets are restricted to the public, and CAVs companies only allow people to get in touch with it inside of the company Since of this, if more complex datasets and improved research devices could be provided, the research could move forward to implement a more powerful ML model to detect more types of malicious attacks based on this project. But spoofing attacks and Impersonation attacks according to the research have less explicit characteristics in nominal format. They act more similar to normal messages than DDoS and Fuzzing attacks. Therefore, the accuracy rate might change, and the FP rate might be higher. In that case, J48 and Naïve Bayes algorithms might become less practical.

6.2 Prevention of DDoS and Fuzzing Attacks

The prevention method could be achieved in the future. And the fundamental idea is shown in the steps below:

1. Extract and implement the model in JAVA.
2. Implement a function to detect and prevent attacks, and the pseudocode is shown below:

```
CyberattackDetectAndPreventwithJ48(String[] thisCANMessageList)
{
    Bool isMalicious = J48PredictionforCANMessage(thisCANMessageList);
    If (isMalicious)
    {
        Abandon(thisCANMessage);
    }
}
```

3. Prepare a device on the CAN bus, such as MITM Packet Squirrel or an OBD2 which can observe and drop detected malicious messages.
4. Deploy the JAVA program with ML attack detect and prevent function on the Packet Squirrel or an OBD2 software.

Nevertheless, the real deployment procedure must be more complicated than the theoretical design.

7. Conclusion

Since cyberattacks on CAVs are generated by devices with certain implemented characteristics, how well an ML model performs, depends on how efficiently and effectively it can catch the characteristics. Meanwhile, a strong research device with a decent Graphic card and large RAM size is essential to explore machine learning research. Especially RAM size is significant to the model implementation when the dataset is relatively large.

As per the research done in this project, the J48 classifier shows better performance on prediction accuracy, but the time cost to generate a model and predict a given data is less efficient. It also required more memory space for building the model. However, Naïve Bayes Classifier needs less prediction time cost and less memory occupation, whereas its prediction accuracy is worse than the J48 classifier. Conclusively, J48 and Naïve Bayes ML model have their own pros and cons when handling cyberattacks on CAN buses in CAVs, just like CAN, LIN and FlexRay has their own roles with different advantages and disadvantages in CAVs Network design. They both perform excellently, but which is better for practical utilization depends on the real situation. Ultimately, The Machine learning methodology is constructive against cyberattacks in the future, the attacks mostly have explicit digital characteristics and are easily captured once the

characters are extracted via machine learning models.

8. Glossary

ADAS - Advanced Driving Assistance Systems
AI/ML - Artificial Intelligence/Machine Learning
AVTP - Audio Video Transport Protocol
CAVs - Connected Autonomous Vehicles
CAN - Controller Area Networks
CNN - convolutional neural network
CRC - Cyclic redundancy check
DDoS – Distributed Denial of service attacks
DLC - Data Length Code
DNS - Domain Name System
DoS - Denial of Service attacks
eMBB - Enhanced Mobile Broadband
EOF - End-of-Frame
GPS - Global Positioning System
Gptp - generic Precision Time Protocol
HCR Lab - Hacking and Countermeasure Research Lab
HTTP - Hypertext Transfer Protocol
IDE - Identifier Extension
IDS - Invasion Detection Systems
IDS/IPS - Intrusion Detection and Prevention Systems
IFS - Interframe Space
IoT - Internet of Things
J48 Decision Tree Classifier
LIN - Local Interconnected Network
M-CAN - M-CAN Intrusion Dataset
MINT - Master of Science in Internetworking program
MITM - Man-in-the-middle Attacks
mMTC - Massive Machine-Type Communications
OBD-II - On-Board Diagnostics-II
OTIDS - CAN-intrusion-dataset
RPM - revolutions per minute
RTR - remote transmission request
SAE - the Society of Automotive Engineers
SBA - Service-Based Architecture
SOF - start of frame
SQL - Structured Query Language
SRR - substitute remote request
TCP – Transmission Control Protocol

TOW-IDS - Automotive Ethernet Intrusion Dataset
UDP - User Datagram Protocol
UML - Unified Modeling Language
UofA - University of Alberta
URLLC - Ultra-Reliable and Low-Latency Communications
URL - Uniform Resource Locator
V2X - Vehicle-to-everything
XSS - Cross-Site Scripting attacks
5G - the 5th generation mobile network

9. Bibliography

Anonymous. (2021, March 3). How Self-driving Cars Work: Sensor Systems. Retrieved from Udacity: https://www.udacity.com/blog/2021/03/how-self-driving-cars-work-sensor-systems.html#:~:text=Self%2Ddriving%20cars%20can%20use%20GPS%20to%20geolocate%20with%20numerical,around%20a%20five%2Dmeter%20radius.

Anonymous. (2022, May 20). What Is DNS Spoofing and How Can You Prevent It? Retrieved from Panda Security: https://www.pandasecurity.com/en/mediacenter/security/dns-spoofing/

Anonymous. (2023). What is a computer port? | Ports in networking. Retrieved from CloudFlare: https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/

Birthday attack. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Birthday_attack

CAN (Controller Area Network) protocol. (2011). Retrieved from JavaTPoint: https://www.javatpoint.com/can-protocol

Corrigan, S. (2016, May). Introduction to the Controller Area Network (CAN). Retrieved from Texas Instruments: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1676217283153&ref_url=https%253A%252F%252Fwww.google.com%252F#:~:text=The%20CAN%20communication%20protocol%20is,attempting%20to%20send%20a%20message.

Deny. (2021, Nov 10). Top 10 Benefits of Google Maps for Users. Retrieved from The Techrim: https://thetechrim.com/top-10-benefits-of-google-maps-for-users/

DoS and DDoS Attacks: What are Their Differences? (2023). Retrieved from Sunny Valley: https://www.sunnyvalley.io/docs/network-security-tutorials/dos-vs-ddos-attacks

Fanton, D. (2022, July 10). What is CAN bus and why is it so important? Retrieved from Onlogic: https://www.onlogic.com/company/io-hub/what-is-can-bus/#:~:text=CAN%20bus%20is%20a%20message,that%20transfers%20data%20between%20components.

Felton, R. (2017, February 14). The Man Who Tested The First Driverless Car In 1925

Had A Bizarre Feud With Harry Houdini. Retrieved from jalopnik: https://jalopnik.com/the-man-who-tested-the-first-driverless-car-in-1925-had-1792312207
Froehlich, A. (2023). insider threat. Retrieved from TechTarget: https://www.techtarget.com/searchsecurity/definition/insider-threat
Götz, F. (2021, January 22). The Data Deluge: What do we do with the data generated by AVs? Retrieved from Siemens: https://blogs.sw.siemens.com/polarion/the-data-deluge-what-do-we-do-with-the-data-generated-by-avs/
Gupta, R. (2019, June 17). Overview of SQL Server Ports. Retrieved from SQLShack: https://www.sqlshack.com/overview-of-sql-server-ports/#:~:text=We%20can%20define%20the%20port,to%20connect%20to%20SQL%20Server.
Guy, C. (2021, May 3). History of Self-Driving Cars | Autonomous Cars Present & Beyond. Retrieved from LelandWest: https://www.lelandwest.com/blog/listing.asp?2021/5/self-driving-cars-present-and-beyond
Harrison, J. (2020, April 7). HOW CAN BUS FLOURISHES IN IOT AND AUTOMATION DESIGNS. Retrieved from Controls: https://content.ccontrols.net/blog/how-can-bus-flourishes-in-iot-and-automation-designs
Howarth, J. (2022, December 21). The Ultimate List of Cyber Attack Stats (2023). Retrieved from Exploding Topics: https://explodingtopics.com/blog/cybersecurity-stats
Huawei. (2021, November). Huawei 5G Security White Paper. Retrieved from https://www-file.huawei.com/-/media/corp2020/pdf/trust-center/huawei-5g-security-white-paper-2021-en.pdf?la=en
J. A. Cook, J. S. (2008). Controller Area Network (CAN). Retrieved from https://www.eecs.umich.edu/courses/eecs461/doc/CAN_notes.pdf
KirstenS. (n.d.). Cross Site Scripting (XSS). Retrieved from OWASP: https://owasp.org/www-community/attacks/xss/
Kopestinsky, A. (2022, Sep 29). 25 Astonishing Self-Driving Car Statistics For 2022. Retrieved from https://policyadvice.net/insurance/insights/self-driving-car-statistics/: https://policyadvice.net/insurance/insights/self-driving-car-statistics/
Korosec, K. (2022, March 30). Legendary hackers Charlie Miller and Chris Valasek talk cybersecurity and autonomous vehicles at TC Sessions: Mobility 2022. Retrieved from TechChrunch: https://techcrunch.com/2022/03/30/legendary-hackers-charlie-miller-and-chris-valasek-talk-cybersecurity-and-autonomous-vehicles-at-tc-sessions-mobility-2022/
M-CAN INTRUSION DATASET. (2020). Retrieved from HCRL: https://ocslab.hksecurity.net/Datasets/m-can-intrusion-dataset
Myers, L. (2014, December). Guide to DDoS Attacks. Retrieved from Center for Internet Security: https://its.fsu.edu/sites/g/files/imported/storage/original/application/dfd40f8b7415faa8fc306afa529520da.pdf

MySQL Port Reference Tables. (2023). Retrieved from MySQL: https://dev.mysql.com/doc/mysql-port-reference/en/mysql-ports-reference-tables.html#:~:text=Port%203306%20is%20the%20default,such%20as%20mysqldump%20and%20mysqlpump.
Parashar, N. (2022, July 2). What is an URL Interpretation Attack? Retrieved from Medium: https://medium.com/@niitwork0921/what-is-an-url-interpretation-attack-dd4117b89f45
Parikh, B. (2023). CAN protocol: Understanding the controller area network. Retrieved from Engineers Garage.
Qiyi He, X. M. (2020). Machine Learning-Based Detection for Cyber Security. Mathematics.
Rashid, F. Y. (2018, May 25). HACKER HISTORY: THE TIME CHARLIE AND CHRIS HACKED A JEEP CHEROKEE. Retrieved from Decipher: https://duo.com/decipher/hacker-history-time-charlie-chris-hacked-jeep-cherokee
Rees, M. (2022, Nov 5). The 8 Most Common Types Of Password Attacks. Retrieved from Expert Insignts: https://expertinsights.com/insights/the-8-most-common-types-of-password-attacks/#:~:text=During%20this%20type%20of%20password,they've%20already%20stolen%20using
Rogers, L. (2015). Internet Security.
SAE. (2021, May 3). SAE Levels of Driving Automation™ Refined for Clarity and International Audience. Retrieved from Society of Automobile Engineers: https://www.sae.org/blog/sae-j3016-update
The CAN Bus Protocol Tutorial. (2023). Retrieved from Kvaser: https://www.kvaser.com/can-protocol-tutorial/
Trojan Horse Virus. (n.d.). Retrieved from Fortinet: https://www.fortinet.com/resources/cyberglossary/trojan-horse-virus
Types of Cyber Attacks. (2023). Retrieved from Fortinet: https://www.fortinet.com/resources/cyberglossary/types-of-cyber-attacks
U.S. Government. (2022, March 3). GPS Accuracy. Retrieved from GPS.gov: https://www.gps.gov/systems/gps/performance/accuracy/
Waymo (Director). (2019). Waymo 360° Experience: A Fully Autonomous Driving Journey (Audio Described) [Motion Picture].
WELCOME TO HCRL. (2022, Oct 8). Retrieved from Hacking and Countermeasure Research Lab: https://ocslab.hksecurity.net/welcome
Wesley Chai, W. C. (n.d.). network protocol. Retrieved from TechTarget: https://www.techtarget.com/searchnetworking/definition/protocol
What is a drive-by attack? (n.d.). Retrieved from Ericom: https://www.ericom.com/whatis/drive-by-attack/#:~:text=A%20drive%2Dby%20attack%2C%20also,device%2C%20running%20any%20operating%20system.