CANCombat
Bridging the Gap Between CAN-Bus Attacks and Defenses

Gaurav Sarraf, Manav Hirani, Shubhashri Chavan, and Jay Patel
Syracuse University, gsarraf, mshirani, schava03, jpatel25@syr.edu

Abstract - **The CAN-Bus protocol, developed by Bosch in the 1980s, is a pillar in automotive communication. This protocol, which is essential to contemporary and autonomous cars, ensures effective data exchange between several sensors and control systems, fostering reliable and real-time vehicular operations. However, the CAN-Bus' increased interconnectedness ultimately results in serious security flaws. In " CANCombat: Offense to Defense with Encryption & IDS," we explore the real dangers posed by an unprotected CAN system, outline strategies to strengthen its defenses, and design a unique Intrusion Detection System (IDS) specifically designed to detect CAN protocol abnormalities. Due to difficulties in obtaining actual testing equipment, we use ICSim to simulate an automobile environment in our analyses.**

Keywords – CAN-protocol, IDS, Defense, AI

INTRODUCTION

Fundamentally, CAN-Bus differs from conventional wire systems in that any device or node may connect with other nodes without needing a centralized computer. A message-based protocol is used by devices to communicate with one another; each message is identified uniquely so that devices can determine how relevant it is. For vehicle-critical systems like brakes and airbags, fault tolerance is essential, and the CAN-Bus architecture promotes robustness and fault tolerance. Additionally, the simplified design increases efficiency and reduces vehicle weight by eliminating extra wire because all devices share a single communication connection. Our in-depth investigation of the security aspects of CAN-Bus is made possible by this intrinsic structure and its ramifications.

The importance of CAN-Bus in the age of driverless cars cannot be emphasized. These vehicles' advanced sensors, including cameras and LiDAR, generate vast data. This data is supported by CAN-Bus, which ensures smooth coordination and communication between various components. The dependability of this protocol is essential for making real-time decisions and for the security of self-driving automobiles. Additionally, CAN-Bus supports smooth operations by integrating new automotive technologies, such as emergency braking and adaptive cruise control. However, as automation increases, CAN-Bus security flaws become more apparent. Malicious actors compromising vehicle functionality over the CAN network are potential hazards, highlighting the urgent need for strong security measures. Furthermore, CAN-Bus is pivotal in diagnostics, granting maintenance teams invaluable insights into system health and performance.[1]

Particularly in the age of autonomous driving, the Controller Area Network (CAN) protocol, essential to vehicle communications, poses significant security issues. The protocol has fundamental problems since it lacks integrated authentication and encryption, making it vulnerable to dangers like replay attacks and message injection. Additionally, as vehicles become more networked, the possibility of remote vulnerabilities increases, making current automobiles vulnerable to possible cyberattacks. [2] Physical flaws combined with the spread of faults throughout the CAN network can result in cascade failures that endanger other road users and passengers in moving vehicles. We examine these flaws in-depth, providing analysis and recommendations to strengthen our autonomous future's vital vehicular communication infrastructure.

BACKGROUND

I. CAN Protocol and ICSim

The CAN bus (Controller Area Network) is a standard for vehicle buses that enables microcontrollers and devices to communicate with each other without needing a host computer. [3] Originally developed to reduce the amount of copper wiring in cars through a multiplexing approach, it's a message-based protocol applicable in various contexts beyond automobiles. Each device sends data within a frame serially. If multiple devices transmit simultaneously, the one with the highest priority can proceed while the rest yield. All devices, including the sender, receive these frames. The CAN protocol includes four standard frame types:

- Data Frame: This is exclusively used for sending data.
- Remote Frame: This frame requests a data frame from a specific node. When a node sends out a remote frame with a particular identifier, any node that matches this identifier immediately responds with a data frame.
- Error Frame: This frame indicates an error in the transmitted frame.

- Overload Frame: This is used to postpone the beginning of the following message if the CAN controller hasn't completed processing the current message.

In contemporary automotive systems, CAN buses play a pivotal role in the interconnectivity of vehicular controllers. These controllers are tasked with acquiring and disseminating various data points, such as sensor readings, vehicle states, informational parameters, and calibration details. [4] The architecture of each vehicle's network is uniquely tailored by the manufacturer, ensuring the data transmitted is specific and localized to the car in question. This localized nature of data transmission within CAN buses negates the necessity for adherence to protocols beyond the physical layer. Consequently, when interfacing with a vehicle's CAN bus, one encounters a substantial volume of data. However, the interpretability of this data is often obscured due to its proprietary nature and lack of explicit content descriptors within the data bytes. This type of data, primarily used for real-time intra-controller communication, is continuously present and is thus referred to as "Normal Mode Messages."
ICSim (Instrumented Control System Simulator) is an advanced simulation tool for emulating Industrial Control Systems (ICS). ICS is integral in managing and operating critical infrastructure, encompassing sectors like power grids, water treatment facilities, and various transportation systems.

A noteworthy application of ICSim lies in its utility for car hacking research, which facilitates the simulation of a car's Electronic Control Units (ECUs). By creating a virtual environment to mimic these ECUs, ICSim enables researchers to identify potential vulnerabilities and assess the robustness of security measures against car hacking threats.
The relevance of ICSim extends to its interaction with the Controller Area Network (CAN) protocol. CAN, a message-based network protocol, was initially developed by Robert Bosch GmbH for vehicular applications. [5] This protocol is pivotal in the automotive industry for the communication between various electronic components within vehicles. The integration of ICSim with CAN protocol simulations allows for a comprehensive exploration of the network's resilience to cyber-attacks and the effectiveness of various defense mechanisms. Moreover, a crucial component in this research framework, CAN-utils, emerges as an essential toolset. Developed originally by Robert Bosch GmbH, the creators of the CAN protocol, CAN-utils are a collection of open-source drivers and a networking stack integrated into the Linux kernel, courtesy of a contribution from Volkswagen Research. This set of tools enables the manipulation and monitoring of CAN network traffic, offering valuable insights into the network's communication patterns and potential security flaws.[5]

The utility of CAN-utils in conjunction with ICSim is further augmented when applied in a segmented terminal environment. Tools like 'candump' and 'cansniffer' provide distinct functionalities; 'candump' is helpful for packet tracking in the CAN network, though it can be complex to decipher, while 'cansniffer' is preferred for its ability to display differences in CAN data content, offering a more nuanced view of the network traffic.[6] When employed with ICSim, these tools facilitate a comprehensive and practical approach to exploring and securing vehicular networks.
Accessing the CAN bus in an actual vehicle requires a nuanced understanding of the vehicle's specific network architecture, as designed by the manufacturer. Given the proprietary nature of the data transmitted across these networks, interfacing with a vehicle's CAN bus presents unique challenges. Typically, the process involves connecting to the OBD-II (On-Board Diagnostics) port, which is standard in most modern cars. This port is a gateway to the vehicle's various electronic systems, including the CAN bus. (Figure 1)

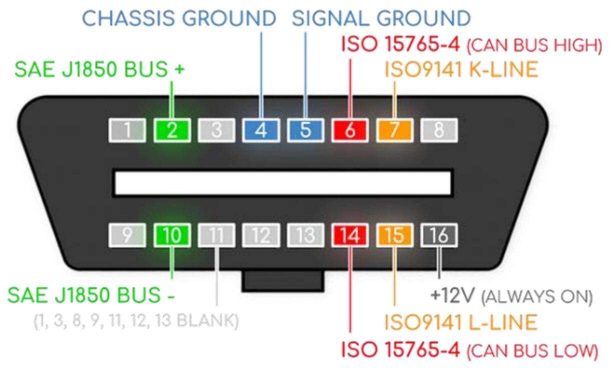


FIGURE I
MODERN OBD PORT USING CAN PROTOCOL

Moreover, the continuous nature of data transmission in the CAN bus, characterized by 'Normal Mode Messages,' implies that data is constantly being exchanged among the controllers if they are active. [7] This aspect is crucial for researchers and technicians seeking to monitor or analyze the network traffic in real time.

II. CAN Security - Attacks

Various attack vectors can be exploited in the context of CAN bus security within automotive systems, emphasizing the importance of robust security measures. Tools like ICSim and CAN-utils are instrumental in simulating and analyzing these threats in a controlled environment. [8] Using simulation tools like ICSim and practical toolkits like CAN-utils, our study explores the following attack vectors in a safe and controlled manner. This exploration is crucial for developing more secure automotive systems, ensuring the integrity and safety of vehicular communication networks. This approach enhances our understanding of CAN bus vulnerabilities but also aids in developing advanced defensive strategies against such cyber threats.

Eavesdropping, a passive attack, involves the unauthorized interception of messages on the CAN bus. Since CAN does not inherently encrypt its communications, an attacker with physical access can readily monitor and record the data traffic. [9] This vulnerability can lead to the exposure of sensitive vehicle operational data. Message injection is another significant threat, where an attacker introduces unauthorized messages onto the CAN bus. This can lead to erroneous data being fed to the vehicle's systems, potentially causing malfunctions, or triggering unintended actions. CAN-utils can simulate such scenarios, allowing us to assess the impact and devise countermeasures.[10]

Replay attacks involve the capture of legitimate messages and their subsequent retransmission. This type of attack can be used to mimic standard operational commands, leading to unauthorized control of vehicle functions. Fuzzing is a technique where the system is bombarded with random, invalid, or unexpected data inputs to test its robustness. In the CAN bus context, this can help identify vulnerabilities that might cause the system to crash, behave unpredictably, or expose security flaws. Finally, spoofing in the CAN bus involves forging messages to impersonate legitimate nodes. This can disrupt the normal communication flow, leading to potential safety hazards.

III. CAN Security – Intrusion Detection Systems

Implementing Intrusion Detection Systems (IDS) in the CAN bus environment is critical to enhancing vehicular network security. In this context, the dataset provided by the OCsLab Honeypot Security offers a valuable resource for developing and testing IDS solutions. This dataset includes a comprehensive collection of standard and unconventional CAN traffic patterns essential for training and validating IDS algorithms. Effective IDS for CAN buses must be capable of distinguishing between legitimate traffic and potential threats, such as those outlined previously, including eavesdropping, message injection, replay attacks, fuzzing, and spoofing. The dataset provides real-world examples of these attack vectors, enabling researchers to develop IDS solutions that are accurate and responsive to various threats.[11] Using ICSim and CAN-utils with this dataset can further enhance IDS development. ICSim allows for the simulation of CAN network environments under multiple conditions, including attack scenarios. This enables the testing of IDS in a controlled yet realistic setting. On the other hand, CAN-utils offer tools for manipulating and monitoring CAN network traffic, which is crucial for understanding the patterns and behaviors an IDS must detect and respond to.

Additionally, the implementation of IDS in CAN environments must consider the unique characteristics of vehicular networks, such as their real-time operational requirements and the need for minimal latency. The IDS solutions developed using these resources should be efficient and lightweight, ensuring they do not impede the normal functioning of the vehicle while providing adequate protection against cyber threats.

ATTACK AND DEFENSE

The Controller Area Network (CAN) protocol, central to the communication systems within modern vehicles, is subject to a range of potential cyber threats that necessitate a comprehensive understanding of both offensive and defensive strategies. This section first outlines a structured approach to exploring these vulnerabilities, utilizing CAN-utils for executing attacks and ICSim for simulation purposes. We will cover each of the attacks in finer detail later. Furthermore, we will also discuss preventive measures to save the environment from these attacks.

The first phase, Sniffing and Eavesdropping, involves passive monitoring of CAN traffic. Here, we aim to gain insights into the communication patterns and data transmitted over the network without actively altering the traffic. This step is crucial for understanding the baseline of regular network activity and forms the foundation for more complex attack strategies.
Following this, Fuzzing represents a technique where the network is bombarded with random, malformed, or unexpected data packets. This method tests the resilience of the CAN network to handle aberrant or non-standard data inputs, potentially revealing weaknesses or triggering system malfunctions.

The Replay attack phase involves capturing legitimate CAN messages and retransmitting them. This could lead to unauthorized control of vehicle functions if the system fails to distinguish between original and replayed messages. This step tests the system's ability to detect and mitigate replayed traffic, often a critical vulnerability in many network systems. Message Injection is another acute phase, where new or altered messages are injected into the CAN network. This attack tests the system's ability to detect and respond to unauthorized commands, which could lead to hazardous malfunctions or manipulation of vehicle behavior.

Finally, Spoofing involves forging messages to impersonate legitimate nodes on the CAN network. This attack assesses the network's susceptibility to identity-based threats, where we assume control over critical vehicle functions by masquerading as a legitimate entity.[12]

Throughout these phases, CAN-utils serve as the primary toolkit for executing these attack vectors, offering a range of functionalities to manipulate and analyze the CAN network traffic. Concurrently, ICSim provides a safe and controlled simulation environment, allowing for the detailed observation and analysis of these attacks' impacts on a virtual representation of a vehicle's CAN network. This structured approach lays the groundwork for in-depth exploration and understanding of the vulnerabilities inherent in the CAN protocol. It forms the basis for developing robust defense mechanisms against such cyber threats.

I. Sniffing & Eavesdropping

The primary goal of sniffing in the CAN bus is to capture and interpret the traffic to gain insights into the network's operations and potential vulnerabilities. This process involves focusing on several critical fields within each CAN frame, which are crucial for understanding the communication dynamics and possible points of exploitation.

The first significant field is the Arbitration Identifier, critical in message prioritization and collision resolution within the CAN network. This identifier determines the message priority (with lower values having higher priority) and indicates the message's intended purpose or destination. Understanding the patterns of these identifiers can reveal critical insights into how the network prioritizes and manages its communications.[11]

The Data Length Code (DLC) is another crucial field, indicating the length of the data field in bytes. This information is essential for understanding the structure of the messages and ensuring that the data is interpreted correctly. Anomalies in the DLC can signify unusual or potentially malicious activities within the network. The Data Field, containing the actual data being transmitted, is the core component of the CAN frame. Analyzing this field helps understand the content of the communications, which is vital for identifying standard operational patterns and potential malicious payloads. Other fields, such as the Control Field, which includes control bits and an acknowledgment slot, and the CRC (Cyclic Redundancy Check) Field, responsible for error checking, are also necessary. These fields ensure the integrity and acknowledgment of the messages, and their analysis can help detect errors or malicious modifications in the network traffic.

By using CAN-utils, we effectively capture and analyze these fields in real time, while ICSim allows for the simulation of various network conditions and attack scenarios. This dual approach is instrumental in identifying and understanding potential vulnerabilities in the CAN bus system, providing a foundation for developing robust security measures. [3] Through careful analysis of these fields, we gain a comprehensive understanding of the network's communication patterns, paving the way for identifying and mitigating potential security risks.

Sniffing and eavesdropping in a real car's CAN bus network can have disastrous consequences, primarily due to the critical role this network plays in controlling various vehicular functions. By intercepting and analyzing the data traffic, an attacker could access sensitive information about the vehicle's operational state or, more alarmingly, uncover vulnerabilities that could be exploited to manipulate vital vehicle controls. Such unauthorized access could compromise the safety systems, alter driving behavior, or even remotely control critical functions, posing a significant risk to the vehicle's and its occupants' safety. This emphasizes the necessity for robust security measures in automotive systems to prevent such potential exploits.

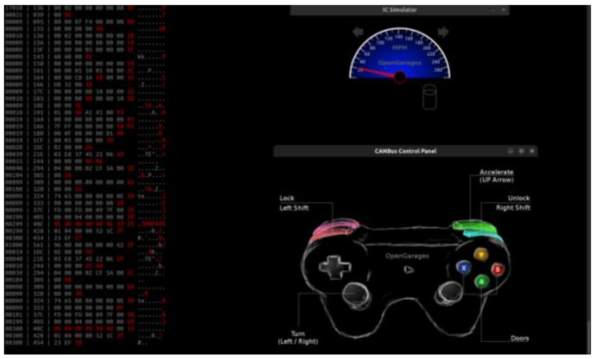


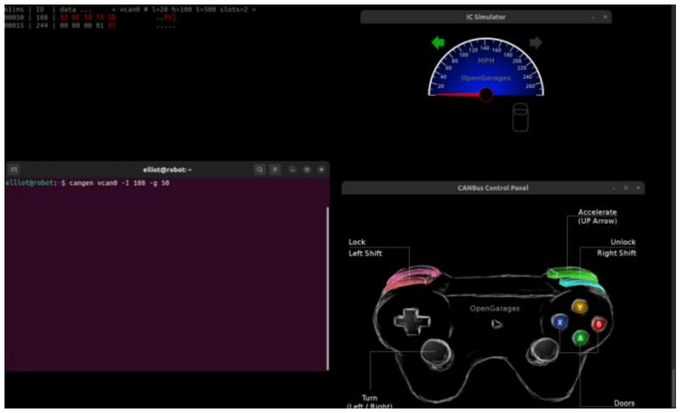
FIGURE 2
Analyzing CAN Bus Frames: Unveiling the Secrets of ICSIM's CAN Traffic

II. Fuzzing

This fuzzing process is critical for identifying weaknesses in the CAN bus system. It tests the network's ability to handle and isolate erroneous data without disrupting regular operations. For instance, a robust system should be able to detect and disregard a malformed message without any negative impact on the vehicle's functionality. However, if the system fails to handle these weird messages appropriately, it could lead to unintended behavior, such as triggering false alarms, turning off critical functions, or causing erratic responses in the vehicle's operational systems.[12]

The ‘cangen’ utility in CAN-utils is instrumental for this purpose. It allows for creating and sending CAN frames with varying degrees of randomness or specific, tailored anomalies. These frames can differ in several aspects, such as the Arbitration Identifier, Data Length Code (DLC), and the data field itself. By altering these fields in unpredictable ways, ‘cangen’ can simulate a range of irregular scenarios that a vehicle's CAN network might encounter.

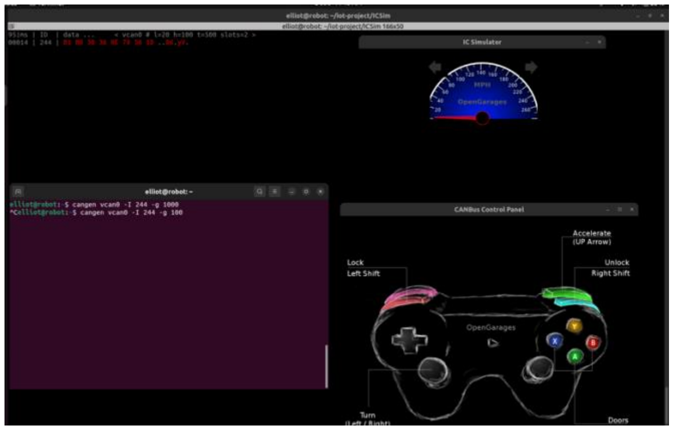
From an attacker's perspective, successful fuzzing can reveal vulnerabilities that could be exploited to disrupt or take control of vehicular functions. [13] By observing how the system reacts to these fuzzed messages, an attacker can gain insights into potential points of failure or ways to trigger specific responses. This information is invaluable for devising targeted attacks that could compromise the vehicle's safety and security.



```
61|ms | ID  | data ...      < vcan0 # l=20 h=100 t=500 slots=2 >
00050 | 188 | [illegible]            ..[illegible]
00015 | 244 | 00 00 00 01 [illegible]         .....
```

```
elliot@robot:~$ cangen vcan0 -I 188 -g 50
```

FIGURE 3
Turn Signal Fuzzing with Cangen in ICSIM



```
95|ms | ID  | data ...      < vcan0 # l=20 h=100 t=500 slots=2 >
00014 | 244 | [illegible] ..[illegible].y?.
```

```
elliot@robot:~$ cangen vcan0 -I 244 -g 1000
^Celliot@robot:~$ cangen vcan0 -I 244 -g 100
```

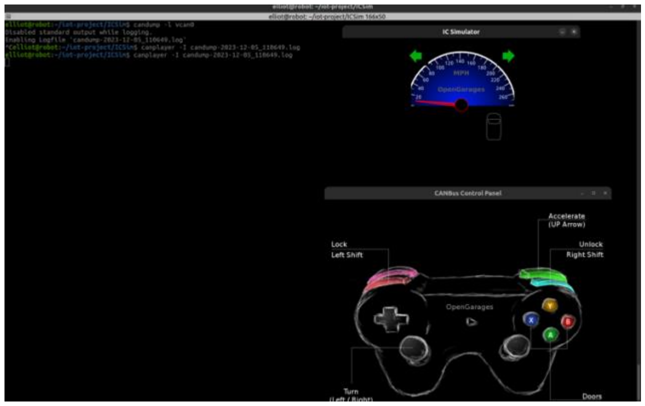
FIGURE 4
Tachometer Fuzzing with Cangen in ICSIM: Exploring Weaknesses in Simulation

III. Replay Attack

In a CAN bus environment, executing a replay attack involves capturing specific CAN frames and then retransmitting them to observe or manipulate the network's behavior. The process begins with identifying a relevant CAN message, which is then sent onto the bus using a command like cansend can0 <arb_id>#<data>. This method is particularly effective when the vehicle's engine is non-idle, as idle engine states may not yield noticeable changes upon message replay.

However, more than merely sending the packet once is required due to the continuous transmission of original messages by the vehicle's ECUs. To overcome this, one strategy is to transmit the spoofed packets at a higher frequency than the actual messages, as demonstrated by the while true; do cansend can0 <arb_id>#<data>; sleep 0.002; done command. [14] Another approach involves monitoring the bus and sending the spoofed packet immediately after detecting the targeted original packet, using a command like candump can0 | grep " <arb_id> " | while read line; do cansend can0 <arb_id>#<data>; done. ICSim for a replay attack entails logging the frames with candump -l vcan0 while simulating actions like throttle increase or turn signal activation. The logged frames are saved in a file and then used as input for the player to replay the captured frames. This is executed with a command like canplayer -I candump-<date>.log, leading to the simulation of the previously recorded actions, such as the activation of turn signals or changes in the speedometer.

Such a replay attack in a real-world scenario can be significantly impactful. It allows an attacker to mimic legitimate commands, potentially leading to unauthorized control over various aspects of the vehicle's functionality. This highlights the importance of securing vehicular networks against such vulnerabilities, ensuring the safety and integrity of the CAN bus system.



```
elliot@robot:~/iot-project/ICSim$ candump -l vcan0
Disabled standard output while logging.
Enabling Logfile 'candump-2023-12-05_110649.log'
^Celliot@robot:~/iot-project/ICSim$ canplayer -I candump-2023-12-05_110649.log
elliot@robot:~/iot-project/ICSim$ canplayer -I candump-2023-12-05_110649.log
```

FIGURE 5
Replay Attack on ICSIM: Manipulating CAN Bus Traffic with Candump and Canplayer

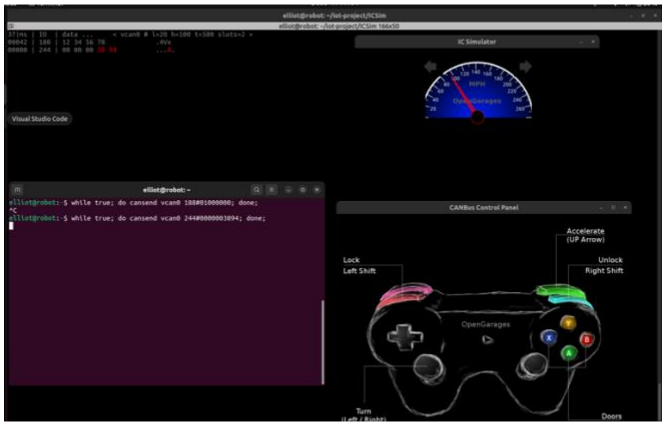
IV. Message Injection & Spoofing

Message injection and spoofing within a CAN bus environment, particularly in the context of vehicular networks, are formidable attack vectors that can lead to profound safety implications. Utilizing CAN-utils for the execution of these attacks and observing the outcomes through ICSim, attackers can manipulate various vehicle functions by injecting or spoofing legitimate CAN messages. In the process of message injection, attackers create and send their own CAN frames onto the network. Tools like cansend in CAN-utils facilitate this by allowing attackers to specify both the arbitration ID and the data payload. For instance, an attacker might use an arbitration ID like 188 or 40C, which is known to control specific vehicle functions. By injecting messages with these IDs, one can illicitly command the vehicle to perform actions like changing the indicator signal, modifying the tachometer readings, or even opening and closing doors.[15]
Spoofing takes this further by impersonating legitimate messages on the CAN bus. This is done by observing and replicating the format of authentic messages, then transmitting these disguised packets onto the network to mislead the system. This attack can lead to even more hazardous situations, as the vehicle's systems might be

tricked into believing that these spoofed commands are coming from a legitimate source.

For instance, using cangen from CAN-utils, an attacker can generate and send messages that mimic the structure of legitimate CAN frames controlling the turn signals or door mechanisms. The impact of such spoofed messages is directly observable in ICSim, which provides a real-time simulation of how the vehicle's systems would respond to these malicious inputs.

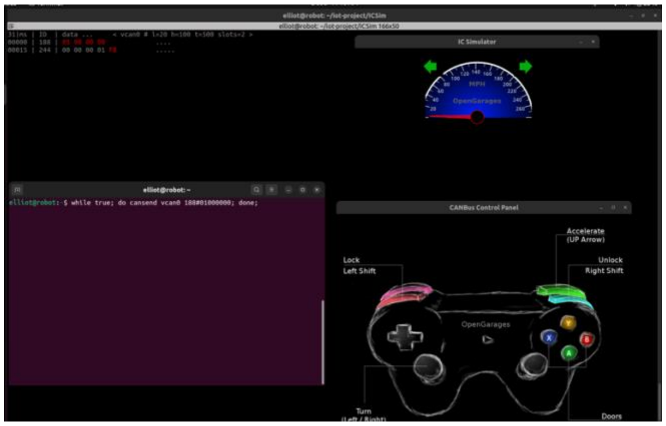
The potential consequences of successful message injection and spoofing in an actual vehicle are dire. By controlling crucial elements like door locks or manipulating the vehicle's display systems, such as the tachometer or turn signals, an attacker can compromise the safety and security of the car and its occupants. This underscores the critical need for robust security measures in designing and implementing vehicular communication systems to safeguard against such malicious activities.



```
37|ms | ID  | data ...     < vcan0 # l=20 h=100 t=500 slots=2 >
00042 | 188 | 12 34 56 78           .4Vx
00000 | 244 | 00 00 00 38 94         ...8.
```

```
elliot@robot:~$ while true; do cansend vcan0 188#01000000; done;
^C
elliot@robot:~$ while true; do cansend vcan0 244#0000003894; done;
```

FIGURE 6
Spoofing and Message Injection Attack: Manipulating Tachometer Data in ICSIM with Cansend



```
31|ms | ID  | data ...     < vcan0 # l=20 h=100 t=500 slots=2 >
00000 | 188 | 01 00 00 00            ....
00015 | 244 | 00 00 00 01 F8         .....
```

```
elliot@robot:~$ while true; do cansend vcan0 188#01000000; done;
```

FIGURE 7
Spoofing and Message Injection Attack: Using Cansend to Manipulate Turn Signals in ICSIM

V. Enhancing Controller Area Network (CAN) Security with XOR Encryption and Randomization Using Seed Value

XOR encryption and randomization with a seed value are fundamental techniques employed to enhance the security and privacy of Controller Area Network (CAN) communication. In a CAN network, where messages are broadcast to all nodes, protecting sensitive data and ensuring that only authorized nodes can access and interpret it is essential. XOR encryption is a bitwise operation that combines data with a secret key to produce ciphertext. In CAN communication, this key is shared among authorized nodes. When a message is transmitted, it undergoes XOR encryption using the key. This process ensures that even if an unauthorized node intercepts the message, it cannot decipher its content without access to the key.

Randomization is equally critical in CAN security. Attackers can exploit predictable patterns in communication. To mitigate this risk, a seed value is utilized. The seed is a starting point for generating a sequence of pseudo-random numbers. Authorized nodes share this seed to generate random numbers for two primary purposes. Firstly, the seed value is employed to randomize the arbitration ID of CAN messages. The arbitration ID is a crucial component that determines message priority and recipient. By randomly changing the ID using the seed, attackers find it difficult to predict which messages are critical. Secondly, the seed is used for data randomization. The data payload of CAN messages is subjected to randomization, ensuring that even if the same message is transmitted multiple times, it appears different to potential eavesdroppers.

The advantages of XOR encryption and randomization are numerous. They provide confidentiality, security, data integrity, and real-time adaptability. Messages remain confidential to authorized nodes, and the randomized content thwarts attackers. Data integrity is maintained, and authorized nodes can adapt to real-time changes in the encryption and randomization process. However, implementing these techniques comes with challenges. Key management is vital to ensure authorized nodes can decrypt messages. Seed synchronization among authorized nodes is necessary for generating consistent random sequences. There's also a slight overhead in terms of message transmission and processing. While XOR encryption is a simple method, it may offer a different level of security than advanced encryption algorithms.

In summary, XOR encryption and randomization with a seed value are valuable tools for enhancing CAN communication security, particularly in resource-constrained environments. Proper implementation and management of keys and seeds are essential to ensure the effectiveness of these techniques. Organizations can safeguard their CAN networks and protect sensitive data from threats by employing XOR encryption and randomization.

INTRUSION DETECTION SYSTEM

As we enter an era where vehicular systems are not just conveyances but sophisticated nodes in an ever-expanding tapestry of the Internet of Things (IoT), the security of automotive networks has become a critical concern. The Controller Area Network (CAN) Bus system, an industry-standard communication protocol that interlinks various electronic control units (ECUs) within a vehicle, is central to these networks. The CAN Bus is a harbinger of efficiency and a testament to the advancement in automotive technology, facilitating rapid and reliable communication. However, this interconnectivity also opens new avenues for cyber vulnerabilities, making vehicles the potential targets of sophisticated cyber-attacks.

Implementing Intrusion Detection Systems (IDS) is a pivotal defense mechanism in this landscape. These systems act as the sentinels of the CAN Bus, meticulously monitoring network traffic for signs of intrusion and ensuring the integrity of vehicular communication systems. With the stakes as high as they are—where a breach can compromise personal data, vehicular control, and passenger safety—the development and deployment of an effective IDS cannot be understated. Therefore, it is imperative to explore the nuanced requirements of IDS in the context of CAN Bus systems to understand their role, functioning, and criticality in preserving vehicular cybersecurity.

This section aims to dissect the complexity of IDS requirements within the specialized environment of the CAN Bus. Through a comprehensive examination of empirical research and the application of theoretical constructs, we will elucidate the essential features and functionalities that an IDS must embody to counteract the burgeoning cyber threats in modern vehicular networks effectively.

I. IDS Requirements for CAN Bus

In the rapidly evolving landscape of Controller Area Network (CAN) systems, the imperative for robust Intrusion Detection Systems (IDS) has never been more pronounced. The CAN Bus, a linchpin in vehicular communication systems, necessitates a multi-faceted approach to security, particularly in the face of escalating cyber threats. This section delineates the essential requirements for an IDS tailored to CAN Bus environments, drawing on empirical research and theoretical constructs.

- Real-Time Monitoring: An effective IDS for CAN Bus must perform real-time monitoring to detect anomalies as they occur. This includes continuously scanning network traffic and immediately identifying unusual patterns that could indicate a security breach.
- Low False Positive Rate: Given the critical nature of vehicular systems, an IDS must have a low rate of false positives to avoid unnecessary disruptions. Implementing advanced algorithms that can learn and adapt to the network's regular operational pattern is essential.
- Scalability: As vehicular networks grow in complexity, the IDS must be scalable to accommodate an increasing number of nodes and evolving types of communication.
- Minimal Performance Impact: The IDS should be designed to consume minimal system resources, ensuring that the vehicle's operational performance is not compromised.
- Forensic Capabilities: After an attack is detected, the IDS should have capabilities to log relevant data that can be used for forensic analysis to understand the attack's nature and potentially identify the perpetrator.

II. Anomaly Detection in IDS

Anomaly detection is a cornerstone of effective IDS, particularly in CAN Bus systems where behaviour can be highly predictable.

- **Baseline Establishment:** A critical first step in anomaly detection is establishing a baseline of everyday activities. This enables the IDS to recognize deviations that may signify an intrusion attempt.
- **Machine Learning Techniques:** Leveraging machine learning techniques can enhance anomaly detection by enabling the system to evolve and respond to new, previously unseen threats.

III. Signature and Behavior-based IDS

Combining signature and behavior-based detection methods can improve the robustness of IDS.

- **Signature Detection:** This method relies on known patterns of malicious activity. The IDS must regularly update the latest threat signatures to maintain effectiveness.
- **Behaviour-based Detection:** Unlike signature detection, behaviour-based methods do not require prior knowledge of attack patterns. These systems build a model of normal behaviour and then monitor for deviations from this model.

IV. IDS Response and Alert System

The response mechanism of an IDS is critical to its overall effectiveness.

- **Alert Severity Levels:** The system should categorize alerts by severity level to help prioritize the response actions.
- **Automated Response Protocols:** In some cases, an automatic response may be necessary to prevent immediate damage, such as temporarily isolating affected nodes from the network.

IDS – IMPLEMENTATION

I. Dataset and Experiment Setup

The foundation of this study is grounded in the comprehensive analysis of the OTIDS dataset, meticulously sourced from in-vehicle CAN Bus traffic. This dataset is a rich mix of various operational and intrusive scenarios meticulously captured to reflect the multifaceted nature of vehicular network communications. Below, we detail the dataset's composition and the setup of our experimental framework.

The OTIDS dataset, integral to our research, is derived from a KIA SOUL vehicle's CAN Bus system. This dataset is unique in its encompassment of normal operational states and various cyber-attack scenarios, providing a robust platform for testing the efficacy of our proposed intrusion detection method. The dataset is divided into four distinct categories:

- **DoS (Denial of Service) Attack:** This subset includes data where messages with the CAN ID '0x000' are injected at a high frequency, simulating a DoS attack scenario.
- **Fuzzy Attack:** This portion contains data with spoofed CAN IDs and DATA values, representing a scenario where an attacker injects random, nonsensical messages into the network.
- **Impersonation Attack:** This segment features data where the CAN messages impersonate a node, specifically with an arbitration ID of '0x164', to mimic legitimate network traffic.
- **Attack-Free State:** This part of the dataset represents regular CAN Bus traffic, free from intrusive activities.

The dataset's composition, as outlined below (Table 1), provides a quantitative overview of the data samples across different states:

TABLE I
SUMMARY OF DATASE

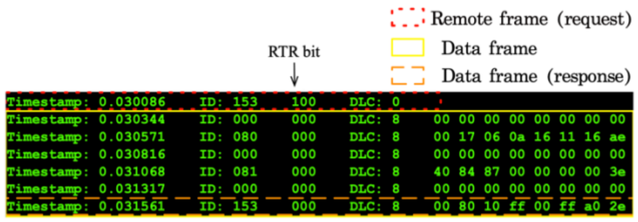
ID	Attack Type	Number of Messages
1	DoS Attack	656,579
2	Fuzzy Attack	591,990
3	Impersonation Attack	995,472
4	Attack free state	2,369,868

A Python environment was created for the experiment setup with libraries such as Pandas for data manipulation, Matplotlib, Seaborn for visualization, and Scikit-learn for machine learning. A function was defined to process each file, extract the relevant information, and label the data accordingly. The combined dataset was then loaded into a Pandas DataFrame for further analysis.

This detailed overview of the OTIDS dataset and the experimental setup provides the foundational framework for our subsequent analysis and model development. The diverse and extensive nature of the dataset ensures a comprehensive evaluation of the proposed intrusion detection system's efficacy.

II. Exploratory Data Analysis

Our analysis primarily focuses on the following data attributes: Timestamp, CAN ID, DLC (Data Length Code), and DATA[0-7] byte values. This initial exploratory phase involves scrutinizing these attributes to identify patterns indicative of attacks and regular traffic. This phase is essential to understand the dataset's dynamics, particularly the nuances between normal and attack states.



```
Timestamp: 0.030086    ID: 153    100    DLC: 0
Timestamp: 0.030344    ID: 000    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp: 0.030571    ID: 080    000    DLC: 8    00 17 06 0a 16 11 16 ae
Timestamp: 0.030816    ID: 000    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp: 0.031068    ID: 081    000    DLC: 8    40 84 87 00 00 00 00 3e
Timestamp: 0.031317    ID: 000    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp: 0.031561    ID: 153    000    DLC: 8    00 80 10 ff 00 ff a0 2e
```

FIGURE 8
DOS ATTACK

A critical component of our exploratory data analysis is the in-depth examination of DoS (Denial of Service) attacks, Fuzzy attacks, and Impersonation attack patterns within the OTIDS dataset. An illustrative example of such an attack, depicted in Figure 8, provides valuable insights into the modus operandi of DoS intrusions in the CAN Bus environment.

In the highlighted instance of a DoS attack, the initial record represents a remote frame characterized by specific identifiers — notably, the third index value set at 100 and a Data Length Code (DLC) of 0. This pattern is indicative of a request message in the CAN Bus network. Following this, the concluding record in the sequence is identified as a corresponding response message. The analysis of these records sheds light on the communication pattern typical of a DoS attack, where the rapid, repeated transmission of remote frames with specific identifiers and low DLC values overwhelms the network, hindering regular communication.

As part of our exploratory data analysis, the detailed investigation of such patterns is instrumental in identifying key characteristics and signatures of different attack types. This analysis not only aids in understanding the nature of the attacks but also plays a vital role in the subsequent stages of feature engineering and model development, where these identified patterns are used to train the IDS to recognize and react to similar intrusive behaviors.

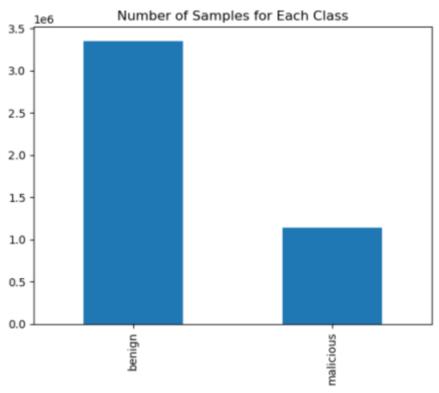


FIGURE 9
DISTRIBUTION OF SAMPLES

The distribution of the classes (benign and malicious) was visualized using a bar chart, as seen in Figure 9, providing insight into the class balance. This was achieved by mapping the labels for all attack samples to 1 and all attack-free samples to 0. Data types of each column were assessed to ensure compatibility with machine learning algorithms. Further analysis included checking for missing values and investigating the distribution and relationship of features through a correlation matrix.

III. Feature Engineering

Feature engineering is a critical step in preparing our dataset for machine learning. It involves creating new features from the existing data to improve the model's ability to learn from the data. In the context of our CAN Bus Intrusion Detection System, the following steps were taken to engineer features:

- **Conversion of Hexadecimal to Decimal:** The raw CAN Bus data payloads, originally in hexadecimal format, were converted into decimal format. This conversion is crucial as it transforms the data into a numerical format that machine learning algorithms can interpret. Each message payload contains up to 8 bytes of data, split into eight separate features (Data_0 to Data_7). Each feature represents one byte of the CAN message, allowing the model to assess each byte's significance independently.
- **Time Feature Engineering:** Timestamp values from the data were used to create a new feature called Time Interval. This feature captures the time difference between consecutive CAN messages, which can indicate network traffic behavior and potentially reveal patterns associated with malicious activities. The time intervals were converted to milliseconds to standardize the measurements and potentially improve the model's sensitivity to rapid successions of messages, which could signify an attack.
- **Statistical Features:** Rolling statistical features were calculated for each of the 8 data byte features. These included:
  1. The mean over a rolling window helps to smooth out fluctuations and capture longer-term trends in the data.
  2. The standard deviation over the same window measures variability or dispersion in the data.

  These features help identify patterns that deviate from what is observed in benign conditions, as attacks might manifest as anomalies in these statistical calculations.
- **Derived ID Feature:** The ID feature of the CAN message, which denotes the identifier for the message type being sent, was summed over a rolling window. This derived feature may help identify unusual patterns of message IDs over time, which could indicate spoofing or replay attacks.[1]
- **Feature Scaling and Normalization**: While not explicitly shown in the code, scaling, and normalization are crucial steps in feature engineering, significantly when the range of values varies widely. This ensures that all features contribute equally to the model's performance. For instance, byte data might be normalized between 0 and 255, and time intervals might be scaled based on their distribution.
- **Feature Selection:** Before model training, a selection process may be employed to choose which features are most relevant to the model. This can be done using techniques such as:
  - Correlation analysis to identify and remove features that are highly correlated with one another, reducing redundancy. Importance metrics from preliminary model runs to identify elements that contribute most to the model's predictive power.
  - Dimensionality Reduction: Advanced techniques such as Principal Component Analysis (PCA) could reduce the feature space's dimensionality while preserving the data's variance. This can be particularly beneficial in complex datasets with many features.

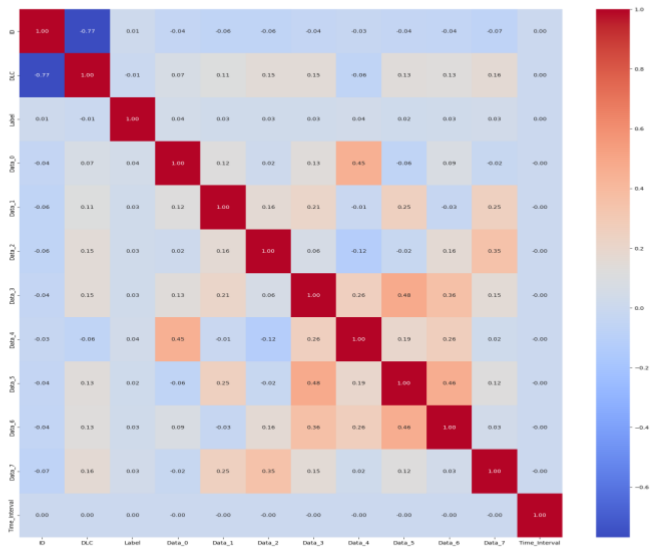


FIGURE 10
HEATMAP OF DIFFERENT FEATURES

Figure 10 presents a heatmap that visually represents the correlation matrix for the features within the CAN Bus dataset. The correlation coefficients range from -1 to 1, as indicated by the color spectrum on the right, where deep red signifies a strong positive correlation, deep blue suggests a strong negative correlation, and white represents no correlation. Features such as **ID** and **Label** show a notable positive correlation, which may imply that specific message IDs are more frequently associated with malicious activity. On the other hand, the data bytes (**Data_0** through **Data_7**) exhibit varying degrees of correlation with the label, suggesting differential predictive power among them. Interestingly, **Data_3** and **Data_4** have a higher positive correlation with the label, which indicates a stronger relationship with malicious instances. The feature **Time Interval** also displays a noticeable negative correlation with several data bytes, hinting at potential patterns in message timing that could be characteristic of ordinary versus abnormal traffic. This heatmap is a powerful tool for identifying relationships between features and will guide the feature selection process to improve the machine learning models' accuracy.[15]

The outcome of the feature engineering process is a dataset enriched with informative and relevant features that comprehensively represent the underlying data patterns. These engineered features are expected to enhance the machine learning model's ability to distinguish between benign and malicious traffic patterns in the CAN Bus data.

IV. Model Selection, Training and Evaluation

In this implementation phase, we focus on selecting appropriate machine learning models to classify the data into 'benign' and 'malicious' categories, training these models on our dataset, and evaluating their performance. Based on our analysis, two classifiers were employed:

1. **Decision Tree Classifier:**

- A Decision Tree Classifier was initialized with the Gini impurity criterion. This choice is suitable for its simplicity and effectiveness in binary classification tasks.
- No maximum depth was set, allowing the tree to expand until all leaves were pure or contained less than the minimum split samples.
- The Decision Tree was trained on 70% of the data, which was randomly split from the dataset to form the training set, with the remaining 30% serving as the test set.
- Performance metrics, including accuracy, precision, recall, F1 score, and ROC AUC score, were computed to evaluate the model. These metrics provide a comprehensive view of the model's performance, balancing the trade-off between true and false positives.
- A confusion matrix was plotted to visualize the performance of the Decision Tree Classifier, offering insight into the true positives, false negatives, and true negatives.
- The ROC curve was also plotted, showing the trade-off between the actual and false positive rates. The area under the curve (AUC) provides a single metric that summarizes the model's performance across all thresholds.

2. **Random Forest Classifier:**

- Following the Decision Tree, a Random Forest Classifier was trained as an ensemble method to improve predictive performance and control over-fitting. The Random Forest model leverages the power of multiple decision trees to produce a more robust classifier.
- The Random Forest was set up with 100 trees and a maximum depth of 10, with other parameters kept at their default values. These hyperparameters can be adjusted based on model performance and computational efficiency in future iterations.[18]
- Like the Decision Tree, the Random Forest Classifier was trained on the same train-test split of the data.
- The performance of the Random Forest was evaluated using the same metrics as the Decision Tree to maintain consistency in evaluation. The Random Forest is expected to perform better than a single Decision Tree due to its ensemble nature, which typically improves generalization.
- A confusion matrix for the Random Forest Classifier was created to assess its performance further.
- The ROC curve for the Random Forest was plotted, with the AUC score summarizing the model's capability to distinguish between the classes.

The mean absolute error was calculated for both models to provide another measure of model performance. This metric averages the fundamental difference between predicted and actual values, offering a straightforward interpretation of the average error magnitude.

V. Model Performance Analysis

Decision Tree's performance was first analyzed, revealing its accuracy, precision, recall, F1 score, and ROC AUC. These metrics are crucial for understanding not only how often the model is correct (accuracy) but also how reliable its predictions are (precision), how well it captures positive instances (recall), and the balance between precision and recall (F1 score).[17]

The Random Forest Classifier's performance was similarly assessed, with the expectation of an improvement over the Decision Tree due to its ensemble approach.

The mean absolute error for both models indicates the average prediction error, which is essential for assessing the impact of misclassifications in real-world applications.

Figure 5 depicts the confusion matrices for the Decision Tree (left) and Random Forest Classifier (right), clearly comparing the predictive performances between the two models on the CAN Bus dataset. In the Decision Tree matrix, we observe that the model correctly predicted 177,227 instances of the benign class (True Label: 0) and 56,546 cases of the malicious class (True Label: 1) while misclassifying

3,065 benign instances as malicious and 3,162 malicious instances as benign. Conversely, the Random Forest Classifier shows an improvement in the correct predictions for the innocent class with 175,934 cases and a slight decrease in the true positive rate for the malicious class with 58,495 accurate predictions. [16] However, it misclassified fewer benign instances (1,213) as malicious but more malicious instances (4,358) as benign compared to the Decision Tree.

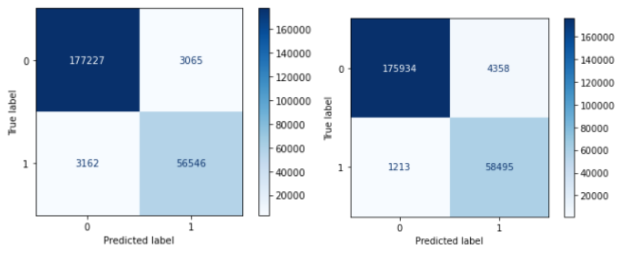


FIGURE 11
CONFUSION MATRIX FOR
DECISION TREE (LEFT) AND RANDOM FOREST CLASSIFIER (RIGHT)

VI. Discussion of Findings

The exploration of Intrusion Detection System (IDS) capabilities through Machine Learning algorithms has culminated in a decision to employ the Random Forest (RF) Classifier as the model of choice for the final training on the entire CAN Bus dataset.

TABLE 2
CLASSIFICATION REPORT

	Precision	Recall	F-1 Score	Support			
			Benign	0.96	1.00	0.98	670595
			Malicious	0.99	0.87	0.93	227352
Accuracy				97%		897937	
Macro Average	0.98		0.94	0.95		897937	
			Weighted Average	0.97	0.97	0.97	897937

As seen in Table 2, the performance metrics post-training indicate a robust model that exhibits high accuracy in classifying traffic as benign or malicious. Key Points from the RF Classifier's Performance Metrics:

- **High Precision for Malicious Class (1):** The precision of 0.99 for the malicious class is particularly noteworthy. This metric signifies that when the model predicts an instance to be malicious, it is correct 99% of the time. Such a high precision rate is crucial in the context of IDS, as it minimizes the possibility of overlooking actual threats, thereby ensuring that potential intrusions are given due attention.[19]
- **High Recall for Benign Class (0):** The recall for the benign class is 1.00, indicating that the model successfully identified all benign instances in the dataset. This is significant as it prevents the scenario where normal operations might be flagged as malicious, which could lead to unnecessary investigations or disruptions in the network's functioning.
- **Good F1-Score Across Classes:** The F1-score, a harmonic mean of precision and recall, stands at 0.98 for the benign class and 0.93 for the malicious class, suggesting a balanced performance between precision and recall. This balance is essential because it assures that the model is accurate and reliable in its predictions.
- **High Overall Accuracy:** With an overall accuracy of 0.97, the model demonstrates high consistency in predicting the correct labels across the dataset. This reinforces the model's capability to be deployed in real-world scenarios, where the cost of misclassification can be high.[20]
- **Macro and Weighted Averages:** The macro average F1-score of 0.95 reflects the model's performance across classes without considering the imbalance in class distribution. The weighted average F1-score of 0.97 evaluates the class imbalance, providing a metric more reflective of the model's predictive power on the dataset as it is constituted.

CONCLUSION

In conclusion, our exploration of the CAN bus protocol through various attack vectors—sniffing and eavesdropping, fuzzing, replay attacks, message injection, and spoofing—has illuminated significant vulnerabilities within vehicular networks, particularly in the context of self-driving cars. These attacks demonstrate the critical need for secure vehicle communication systems to prevent unauthorized control and ensure passenger safety.

As a countermeasure to these threats, we have implemented encryption within the CAN protocol, providing security to the data packets in transit. This encryption helps to prevent unauthorized access and manipulation of the vehicle's network by obscuring the content of the communications, thus hindering the efforts of potential attackers.

Furthermore, we have leveraged the CAN Intrusion Dataset provided by OCS Lab to develop a lightweight, high-accuracy Intrusion Detection System (IDS). This IDS serves as a vigilant network monitor, capable of detecting anomalies indicative of a cyber threat. Using this dataset, which includes a wide range of attack scenarios, our IDS has been trained to identify and respond to potential intrusions with high precision, thereby serving as an effective remediation tool.

We aim to refine our IDS for future work by integrating advanced machine learning algorithms that can adapt to evolving attack methods. Additionally, we will explore the integration of IDS within the encryption framework to create a comprehensive defense strategy that offers both proactive and reactive security measures. The continuous advancement of attack techniques necessitates ongoing research and development in vehicular cybersecurity. We aim to avoid potential threats by constantly improving the encryption and intrusion detection capabilities, ensuring our vehicular networks remain secure in an increasingly connected world.

References

[1] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," 2014 International Conference on the Internet of Things (IOT), Cambridge, MA, USA, 2014, pp. 13-18, doi: 10.1109/IOT.2014.7030108.

[2] W. A. Farag, "CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms," 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), Sharjah, United Arab Emirates, 2017, pp. 1-5, doi: 10.1109/ICMSAO.2017.7934878.

[3] M. Gmiden, M. H. Gmiden and H. Trabelsi, "An intrusion detection method for securing in-vehicle CAN bus," 2016 17th International Conference on Sciences and Techniques of

[4] Automatic Control and Computer Engineering (STA), Sousse, Tunisia, 2016, pp. 176-180, doi: 10.1109/STA.2016.7952095.

[5] M. Bozdal, M. Samie and I. Jennions, "A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions," 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Southend, UK, 2018, pp. 201-205, doi: 10.1109/iCCECOME.2018.8658720.

[6] B. Lampe and W. Meng, "IDS for CAN: A Practical Intrusion Detection System for CAN Bus Security," GLOBECOM 2022 - 2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 1782-1787, doi: 10.1109/GLOBECOM48099.2022.10001536.

[7] L. Dariz, M. Selvatici, M. Ruggeri, G. Costantino, and F. Martinelli, "Trade-off analysis of safety and security in CAN bus communication," 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), Naples, Italy, 2017, pp. 226-231, doi: 10.1109/MTITS.2017.8005670.

[8] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of CAN Bus Security Challenges," Sensors, vol. 20, no. 8, p. 2364, Apr. 2020, doi: 10.3390/s20082364.

[9] Zhouyan Deng, Jiajia Liu, Yijie Xun, Junman Qin, "IdentifierIDS: A Practical Voltage-Based Intrusion Detection System for Real In-Vehicle Networks", IEEE Transactions on Information Forensics and Security, vol.19, pp.661-676, 2024.

[10] Ouidad Saber, Tomader Mazri, "In-Vehicle Intrusion Detection based on Machine Learning," 2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM), pp.1-6, 2023.

[11] Amit Chougule, Kartik Agrawal, Vinay Chamola, "SCAN-GAN: Generative Adversarial Network Based Synthetic Data Generation Technique for Controller Area Network," IEEE Internet of Things Magazine, vol.6, no.3, pp.126-130, 2023.

[12] Brooke Lampe, Weizhi Meng, "Intrusion Detection in the Automotive Domain: A Comprehensive Review," IEEE Communications Surveys & Tutorials, vol.25, no.4, pp.2356-2426, 2023.

[13] Avu Bharath Kumar, Manwinder Singh, "Enhancing Intrusion Detection in Autonomous Vehicles Using Tree-Based Machine Learning Techniques," 2023 3rd Asian Conference on Innovation in Technology (ASIANCON), pp.1-7, 2023.

[14] Yan Meng, Jiachun Li, Fazhong Liu, Shaofeng Li, Haotian Hu, Haojin Zhu, "GB-IDS: An Intrusion Detection

[15] System for CAN Bus Based on Graph Analysis", 2023 IEEE/CIC International Conference on Communications in China (ICCC), pp.1-6, 2023.

[16] Amani Ibraheem, Zhengguo Sheng, George Parisis, Daxin Tian, "Network Tomography-based Anomaly Detection and Localisation in Centralised In-Vehicle Network," 2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS), pp.1-6, 2023.

[17] Jiangjiang Zhang, Bei Gong, Muhammad Waqas, Shanshan Tu, Sheng Chen, "Many-Objective Optimization Based Intrusion Detection for in-Vehicle Network Security", IEEE Transactions on Intelligent Transportation Systems, vol.24, no.12, pp.15051-15065, 2023.

[18] OCS Lab. (n.d.). CAN Intrusion Dataset. Retrieved from https://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset
Lee, H., Jeong, S. H., & Kim, H. K. (2018). OTIDS: A novel intrusion detection system for the in-vehicle networks using a remote frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST 2017) (pp. 57-66). Institute of Electrical and Electronics Engineers Inc.. https://doi.org/10.1109/PST.2017.00017

[19] A. L. Toledo and X. Wang, "Robust detection of MAC layer denial-of-service attacks in CSMA/CA wireless networks," IEEE Transactions on Information Forensics and Security, vol. 3, no. 3, pp. 347–358, 2008.

[20] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks - practical examples and selected short-term countermeasures," in Proc. of SAFECOMP, 2008.
H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in Proc. of ICOIN, 2016.