A Survey On Recent Attacks On CAN Networks And Its Countermeasures

Aniket Agarwal (40266485)
Concordia University
Montreal, Canada
aniketagarwal57@gmail.com

Kalyani Batle (40243967)
Concordia University
Montreal, Canada
kalyanibatle25@gmail.com

Teja Venkata Sai Manikanta Sirigidi (40265966)
Concordia University
Montreal, Canada
tejasirigidi@gmail.com

Geeshma Sri Sai Maddipati (40277629)
Concordia University
Montreal, Canada
geeshmamaddipati@gmail.com

Charan Tej Cheedella (40261465)
Concordia University
Montreal, Canada
cheedella.charantej@gmail.com

Sai Mahitha Bheemineni (40271856)
Concordia University
Montreal, Canada
saimahithabheemineni@gmail.com

Chandra Vamsi Sekhar Peketi (40277985)
Concordia University
Montreal, Canada
vamsisekhar2@gmail.com

Karunyamruta Subash Anguluri (40266020)
Concordia University
Montreal, Canada
subash.anguluri@gmail.com

Manikanta Chinthala (40271167)
Concordia University
Montreal, Canada
chinthalamanikanta6@gmail.com

Alaa Alsharif (40279078)
Concordia University
Montreal, Canada
alaa.alshariff@hotmail.com

Abstract— Today's automotive systems are becoming more intricate and interconnected, incorporating various protocols like CAN, LIN, MOST, and FlexRay. Among them CAN is a widely recognized vehicle bus standard utilized for in-vehicle networks. Due to its affordable price and adaptable design, which minimizes the need for a wiring harness, it is well-liked in industrial and automotive applications. CAN is a message-based protocol. All nodes can read the messages sent over the bus, and the packets contain no information about the sender or recipient of the messages. The automotive domain functions supported by the protocol include automatic start/stop, electric parking brakes, parking assistance, automatic lane detection, collision avoidance, and more. Advances in vehicle communication technology have made a variety of network-based risks easier to exploit. This paper focuses on the Controller Area Network (might) protocol and analyzes how it might be used to support malicious actions, such as DoS, Replay, Spoofing & message modification attacks on in-vehicle networks.

Keywords—CAN bus, DoS attack, Spoofing, Replay, Message modification attacks, ECU, Bus-off, Mitigations

I. INTRODUCTION

A. Overview of In-Vehicle CAN Network

CAN is a widely recognized bus standard utilized for in-vehicle networks. It is like the nervous system of a car, helping different parts communicate with each other. CAN is a message-based protocol for example, when a driver presses the accelerator, a message is generated and is passed through the CAN network to the engine telling it to speed up. The engine is a node in this case. All nodes can read the messages sent over the bus, and the packets contain no information about the sender or recipient of the messages. The automotive domain functions supported by the protocol include automatic start/stop, electric parking brakes, parking assistance, automatic lane detection, collision avoidance, etc.

A CAN bus system consists of two lines, CAN-high, and CAN-low, connected to all nodes in the system. These lines carry the differential message signal generated as the transmitted data. Each node has a Transceiver and a Controller. The Transceiver is responsible for converting the digital signal from the CAN controller to the differential signal to be transmitted on the bus and vice versa. The Controller, on the other hand, is responsible for creating the digital frames that contain the data to be transmitted to and from the node. The Controller also has an error-checking mechanism. Figure 1. Shows the general architecture of the CAN network.

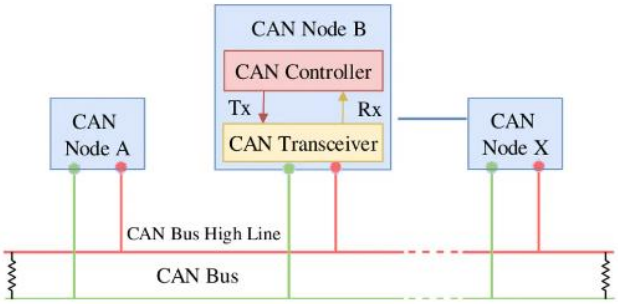


Figure 1: CAN Network Architecture.

CAN Bus Attack Interfaces: The attack surface in the connected car environment consists of telematics units, infotainment systems, the OBD-II port, and sensors. The attacker can inject messages into the network through direct

connections, such as OBD-II, or wirelessly via telematics as shown in Figure 2.

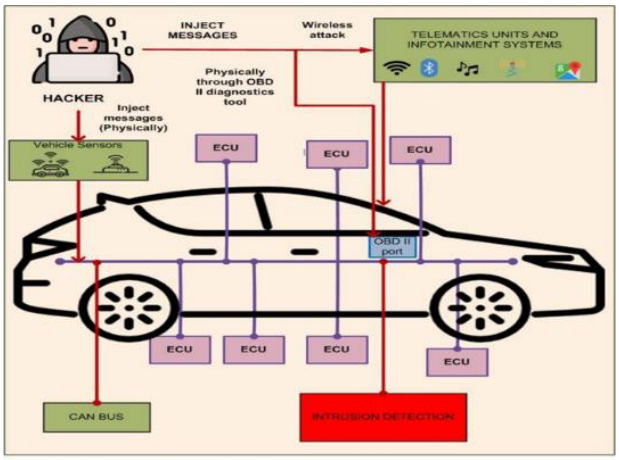


Figure 2: CAN Bus Attack Interfaces [20].

B. Purpose and Scope

With the increasing integration of CAN networks in modern vehicles and industrial automation systems, understanding the vulnerabilities and potential threats to these networks is crucial for ensuring their reliability, safety, and security. This survey aims to identify the various types of attacks that are possible on CAN networks and explore the existing countermeasures and mitigation strategies to protect CAN networks against such threats.

The scope of this survey involves the following areas:

1. Types of Attacks: This section discusses various attacks targeting CAN networks, including but not limited to denial of service (DoS), replay attacks, and physical attacks.
2. Countermeasures and Mitigation: This section reviews the existing countermeasures and mitigation strategies designed to protect CAN networks against attacks which include ML-based detection and IDS.
3. Future Directions: This section discusses the emerging trends and open challenges in securing CAN networks. It identifies areas for further research study and development to enhance the security of CAN-based systems.

II. BACKGROUND

A. Evolution Of Automotive Networks

One of the most important aspects of the automotive industry's continuous transformation has been the evolution of automotive networks, which have had to adapt to the demands and increasing complexity of modern automobiles. Back then, point-to-point wiring was the main way that vehicle electrical systems were implemented for features like lighting and ignition. Nevertheless, the necessity for a more complex system of controlling and coordinating among many electrical devices and modules emerged as cars gained additional features like air conditioning, motorized windows, and sophisticated safety systems.

B. Controller Area Network (CAN)

This necessity led to the introduction of the Controller Area Network (CAN) in the 1980s, developed by Bosch, which became a standard due to its robustness and efficiency in handling communication in the electrically noisy automotive environment. A powerful automotive bus standard called CAN evolved to enable devices and microcontrollers to talk to one another in applications without the need for a host computer. This protocol was designed to address the growing complexity of vehicle electronics, enabling various electronic control units (ECUs) within a vehicle to communicate with each other over a single or dual-wire network. This innovation drastically reduced the amount of wiring needed, leading to improvements in vehicle reliability, maintainability, and fuel efficiency due to decreased weight. The CAN protocol is known for its high resilience to interference, high-speed capability (up to 1 Mbps), and efficient handling of high-density network traffic, making it particularly well-suited for the demanding environments found in automotive applications. Over the years, CAN has become STANDARD in the automotive industry and other areas such as industrial automation, medical equipment, and aviation, reflecting its versatility and reliability as a communication network.

C. Local Interconnect Network (LIN):

LIN is a low-speed, low-cost serial communication protocol that is mostly utilized for non-critical car applications like window and door lock control. It is intended for parts that don't need the real-time functionality or high-speed data transfer of more sophisticated systems like FlexRay or CAN. LIN operates on a single master with multiple slaves' configuration, supporting up to 16 nodes. It uses a single wire for communication, offering a data transmission rate ranging from 1 to 20 kbps. [17]

D. FlexRay

FlexRay provides a high data rate and is used for more demanding applications that require greater bandwidth and more reliable data transmission. This includes systems related to safety and driving dynamics, such as adaptive cruise control and active suspension systems. FlexRay supports data rates up to 10 Mbps and offers advanced features like error detection and correction, fault tolerance, and redundancy. It's designed to fulfill the needs of high-speed and safety-critical applications, providing a deterministic and fault-tolerant communication medium.[17]

E. Data Privacy in Modern Personal Vehicles

With a growing number of sensors, cameras, safety, and communication systems installed, automakers are incorporating Internet of Things technologies into their new models. Due to this connectivity, cars collect and exchange huge amounts of data. This prompts several crucial queries: Who has access to this information? Who is the owner of that? Can customers protect their data in any way? How might this information be used to assign blame in the case of an accident? Could it be used by car owners to prove to the manufacturer that something went wrong? Can emergency services or law enforcement access it? Despite the importance of these concerns in the context of developing automotive

automation and telematics, definitive answers are still absent. [1]

F. Private Information Discovery

A lack of data encryption emerged when Tesla enthusiasts GreenTheOnly and Theo found a ton of personal information, including contact information, navigation history, and accident recordings, from a salvaged Tesla Model 3. This incident highlights the critical need for laws protecting the privacy of vehicle data and giving owners control over their information in the event of an accident, loss, or sale. Although Tesla suggests that user data be erased upon car resale, there are cases where user data is preserved, indicating that manufacturers and users should share responsibility for data deletion. Privacy concerns are heightened by the changing world of automobile telematics, where insurance firms use driving data to modify charges. A study also showed how driving behaviors could be used to identify drivers, highlighting the need for strong privacy measures, and revealing a wide range of privacy issues. [1]

Modern cars, equipped with advanced connectivity and computing capabilities, face significant cybersecurity risks, as evidenced by researchers remotely hijacking a Jeep Cherokee's functions in 2015. The potential for cyberattacks on connected vehicles could lead to scenarios ranging from widespread immobilization to uncontrolled movements during rush hour, posing serious safety and privacy threats. In response, Tesla has initiated a bug bounty program, offering substantial rewards for identifying vulnerabilities, highlighted by a successful attack on its infotainment system at the Pwn2Own event. This underscores the critical need for automotive manufacturers to enhance the security of vehicle systems to protect against emerging cyber threats. [1]

There are four main types of vulnerabilities for possible entry points within in-vehicle systems which are attacks initiated by sensors, attacks originating from the infotainment system, attacks that begin through telematics systems, and attacks that start from direct interactions with the vehicle's interfaces., these draw attention to the two main attack vectors—wireless and physical access—that hackers use to compromise internal car networks. These entry points are used by external inputs to compromise Electronic Control Units (ECUs). Among other methods, hackers may use software bugs or remotely obtain the car's key over the internet to control ECUs. Many security issues pertaining to the in-vehicle network have led to continued study into developing stronger security frameworks. Vulnerabilities in wireless networks can be used to target the bus system of the car. Though many features are already available in smart cars due to the quick development of automotive technologies, no security system can completely prevent all possible attacks. Hackers are improving their techniques to take advantage of smart cars in the same way that technology is advancing.[2]

III. ATTACKS ON IN-VEHICLE CAN BUS

A. DOS attacks with different methods.

1) CAN Signal Extinction-based DOS Attack on In-Vehicle Network:

a) Attack Mechanism:

A novel attack technique that is undetectable by current IDS called CEDA (CAN signal extinction-based DoS attack) employs a voltage drop to destroy the CAN signal. This attack method is a kind of DoS attack, and the goal is to remove the target system instead of making it unavailable. The attack exploits a weakness in the Controller Area Network (CAN). CAN has two logical states: a recessive state and a dominant state. If both CANH and CANL are of the same voltage around 2.5v it is called a recessive state. The dominant state is CANH, which has a higher voltage around 3.5V compared to CANL around 1.5V. By subtracting the voltage potential of both CANH and CANL, the differential voltage is determined. If the differential voltage is less than 0.5 V, the bus will be considered in the recessive state, and if the differential voltage is greater than 0.9 V, it is considered in the dominant state. However, if the differential voltage is between 0.5 V and 0.9 V, it is neither a dominant state nor a recessive state. In this case, the bus state is not defined according to the CAN standard; this area is the gray zone. ECUs ignore messages with an undefined voltage level (gray zone) [5].

When the attacker conducts this attack on a specific ECU by ID, the ECU continuously generates an error frame, and when the error frame reaches the threshold, the ECU becomes "bus-off." This DoS attack performs only on a CAN-based in-vehicle network. According to the paper, increasing resistance on the CAN bus can lower the differential voltage; both are inversely proportional. The attacker calculates the exact resistance to push the voltage of a specific message into the gray zone using its ID [5].

b) Attack Model:

The goal of this attack is to make targeted ECU unusable and remove it from the network instead of making it unavailable. Attackers must set up a monitoring device and keep tabs on messages in the CAN-based in-car network and use this to determine the CAN ID of the targeted device or function and raise the resistance to stop the target ECU from sending messages to other ECUs.

2) DoS Attack Detection and Mitigation for Autonomous vehicles using Raspberry Pi [18]

a) Attack Implementation:

The experimental hardware arrangement consists of four main parts: the attacker node, the countermeasure node, a prototype vehicle subsystem, and the CAN Bus, a straightforward two-wire system ended with 120Ω resistances. Critical automotive systems including the Airbag Deployment System, the Anti-lock Braking System (ABS), and the Adaptive Cruise Control System (ACC) are capable of Denial of Service (DoS) attacks. The purpose of this study is to introduce a denial-of-service attack against a prototype reverse parking assist system, as shown in Figure 3, which was built utilizing two Arduino Uno R3 development boards in communication with two MCP2515 SPI-to-CAN modules. [18]

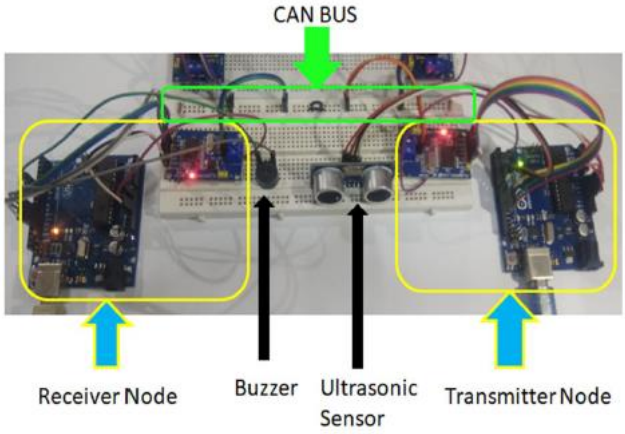


Figure 3: Prototype of Reverse Parking System [18].

Any pre-existing or externally introduced node that is forced to carry out the DoS attack by flooding the CAN bus with pointless, high-frequency garbage data bits to gain arbitration over other nodes is known as the attacker node, or Electronic Control Unit (ECU). As a result, primary nodes are prevented from sending their data and come across an unavailable bus while doing so. Using Raspberry Pi 3B+ development boards, MCP2515 controllers are used to interface with the bus for the attacker ECU and countermeasure node prototypes, as shown in Figure 4. [18]

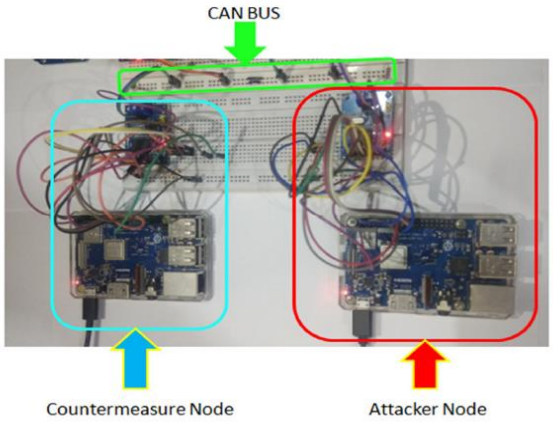


Figure 4: Attacker and Countermeasure Node Prototype [18].

By taking advantage of CAN ID's frame structure, the DoS attack is injected into the CAN bus. First, the validity of the CAN Frame is examined, and if it is found to be invalid, the Increment Frame (IF) Counter is raised. When there are constant invalid frame introductions that push the IF Counter beyond the threshold, the node malfunctions, signifying that a DoS attack has been successfully launched. Of the four nodes shown in the figures, one is the attacker node that broadcasts incorrect frames nonstop, while the other is the node that is being attacked by a denial-of-service attack. [18]

b) Attack Explanation:

The buzzer node (receiver) and the ultrasonic sensor node (transmitter) can communicate with each other when the CAN bus is initially configured and functioning normally. A series of "1" data bits are sent over the CAN bus by the ultrasonic sensor when it detects there are no obstacles. The SPI protocol transforms these "1" bits to ASCII, which is then converted into the hexadecimal representation "31" on the Raspbian terminal. In response, the buzzer node just stays inactive and flashes the message "no obstacle detected."

Similarly, a series of "0" data bits are sent over the CAN bus by the ultrasonic sensor in response to the detection of an obstacle. On the receiver node terminal, this data is converted to "30". The buzzer node responds by turning on and notifying people that an "obstacle had been detected."

However, to obstruct authorized communication, an attacker node that launches a Denial of Service (DoS) attack on the CAN bus floods the bus with useless high-frequency garbage data bits. To effectively overload the transmitter node in this case, the attacker node sends "2 3" data bytes with a higher priority message ID. The transmitter node is unable to send its messages quickly due to the flood of junk data. The overload caused by the DoS attack prevents valid communication from occurring on the CAN bus. Python code in the receiver recognizes an increase of invalid data and reacts by displaying the message "invalid data," which denotes that the attack has affected the system's ability to function.

c) Computation/Communication Resources needed for Attack:

To carry out the Denial of Service (DoS) assault on the Controller Area Network (CAN) bus, the attacker needs an MCP2515 controller and a Raspberry Pi 3B+ development board. In addition, any pre-existing or externally inserted node that wins the arbitration against the other nodes and is compelled to carry out the DoS attack by transferring useless high-frequency garbage data bits over the CAN bus may be the attacker node. [18]

d) Attack Mitigation:

An efficient countermeasure is put in place to reduce Denial of Service (DoS) assaults on the CAN bus, making use of the built-in BUS-OFF feature of the CAN protocol. specifically built countermeasure node must continuously monitor the bus for the existence of invalid messages to implement this countermeasure. When the countermeasure node notices an unusual flood of false frames that might indicate a denial-of-service effort, it launches a BUS-OFF attack directed at the attacker node. By forcing the attacker node into an error-passive state, the BUS-OFF attack successfully stops the malicious activity (as shown in Figure 5). After that, the attacker node is switched to the BUS-Off state, which cuts it off from the network and stops it from sending any further illegal data. The integrity and functionality of the CAN bus are maintained by isolating the attacker node, permitting appropriate communication to resume uninterrupted. This mitigation strategy ensures that the vehicular subsystems relying on the CAN bus remain protected from disruptive attacks, safeguarding the overall functionality and security of the automotive system. [18]

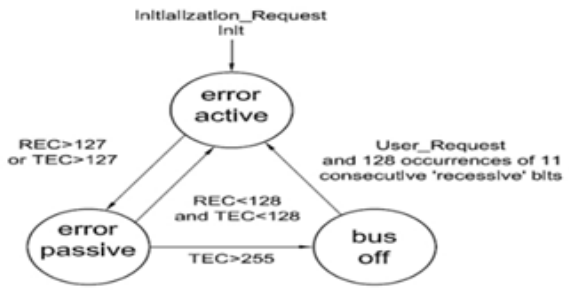


Figure 5: CAN Error State Diagram [19].

3) DOS attack using False Request to Send (RTS):

a) Overview of the FRTS attack:

J1939-21 standard uses CAN protocol to communicate in some commercial vehicles and in this protocol if they are flooded by RTS messages the protocol ignores the old messages and considers only the latest ones.

b) Implementation:

When a requester receives an RTS message (request-to-send) from the original source, the request is allocated to the buffer it might be of 2-3 bytes, later the requester sends CTS message (clear-to-send) which requests certain packets in sequence.

Now the attacker can exploit it by sending self-crafted RTS messages with smaller bytes which look like the original RTS message. But the buffer will allocate these RTS messages, and every message will be registered later the buffer size will overflow, making the electronic control unit (ECU) crash.

4) DOS attack using Connection exhaustion method:

a) Overview:

Assuming the attacker somehow gets access to the car's CAN bus, and it has J1939-21 standard implemented in it.

b) Implementation:

In J1939-21 standard each device has an 8-byte address which allows an ECU to have 225 connections at once. But there are usually less than 255 so that an attacker can scan for active devices and make fake connection, so theypretend to be the same and block the other devices from connecting to it. So other ECUs cannot connect to it and jam's the network.

c) Mitigation:

Authenticating the ECUs before connection can prevent this type of attack, or by implementing some monitoring agent to check the ECUs sometimes can prevent this attack.

5) A selective Link layer DOS attack:

a) Overview:

Unlike traditional attacks, selective denial-of-service doesn't need to transmit complete frames. As a result, this would be undetectable via frame analysis. In modern vehicles, all CAN Buses are vulnerable due to CAN protocol weakness. The attack is low-cost and accessible.

b) Devices and Tools used:

Arduino Uno that served as a microcontroller, MCP2551 CAN Transceiver, ODBLink SX USB-to-ODBII Cable, CANtact 1.0.

c) Implementation:

The attacker physically connects the well-directed error flags like the ODB-II dongle to the car's ODB-II port, forcing other nodes to enter a bus-off state. The attack can be done by attaching and hiding the attacking device anywhere in the car's internal network. Additionally, the attack can be executed without any physical interaction with the target vehicle [8].

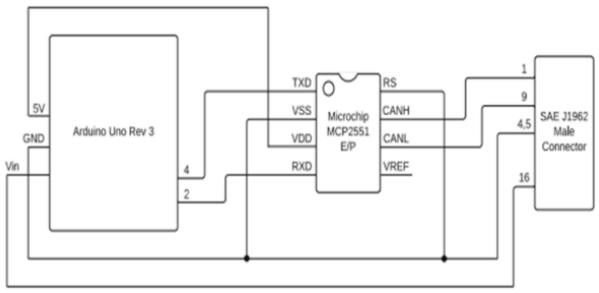


Figure 6: Schematic of crafted Attacking Device [8].

d) Mitigations:

To mitigate these security risks, a proposed solution involves a novel detection mechanism based on the measurement of current flow on the CAN bus. This mechanism aims to identify the addition of rogue devices to the vehicle's internal network by detecting changes in the bus load caused by the presence of unauthorized nodes. While this solution can address physical tampering, it may not protect against remotely compromised nodes. However, it provides a practical approach to enhance the security of CAN bus networks in vehicles [8].

6) Hardware Trojan Enabled Denial of Service Attack on CAN Bus:

The new electrical vehicles have some new functionalities such as smart features, better connectivity, and autonomous behavior. For these functionalities naturally, more ECUs are connected using CAN. For this reason, the security of the vehicle is become a major concern [7].

Hardware Trojan is a novel threat that enables attackers to use backdoor access to implement their attacks. It is an

event-triggered activation attack. This can neither be detected nor prevented.

a) Attack:

Security not only depends on the software but also on the hardware. But nowadays, due to economic interest, it is compromised and hardware security threats such as Hardware Trojan are emerging [7].

Hardware Trojan is nothing, but a malicious modification of hardware implemented at the hardware level, and this is not active until a rare event. This is why it is hard to detect. It is easy to implement, and the result is catastrophic [7].

Trojanised CAN Controller is attached to a node, and it masquerades as another node. Since the CAN has broadcast and no authentication. So, it is easy to take control of the system. There is no main computer in CAN for communication, So CAN relies on itself to communicate. It has a special rule called "BIT STUFFING RULE" which means after five consecutive bits of the same logic level, the next bit should be the compliment. HT messes it by sending the same bit (Logic zero) six times consecutively. Logic zero is dominant so it overwrites Logic one. It will cause an error frame. Usually, other nodes will discard this message, and then the error counter increases. Once, the error counters more than 255, the node will go to BUS-OFF state [7].

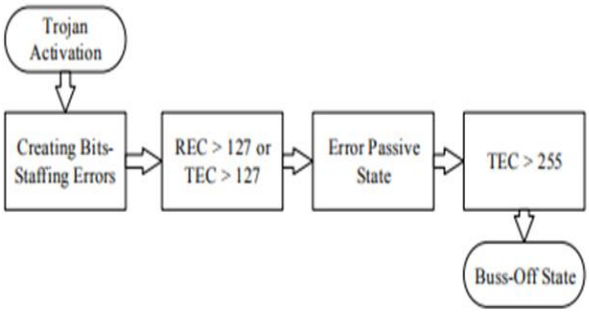


Figure 7: The process of disabling the Node.

B. Reply attacks with different methods.

1) Replay Attack:

Replay attacks are a type of Man-in-the-Middle (MITM) attack, where previously captured messages are retransmitted or delayed in their transmission. This can be particularly hazardous in vehicle environments, where such attacks can influence the functionality of safety-critical systems like braking. The insidious nature of replay attacks lies in their ability to be executed even when messages are encrypted, bypassing the traditional security measures that ensure message confidentiality. An example provided illustrates the potential danger an adversary captures a braking system's message and retransmits it at a critical moment to prevent the actual braking signal from being correctly processed, potentially leading to a collision [10].

a) Experimental Scenario Involving Distance Measurement:

This system comprises a distance sensor and a control unit, highlighting the process through which an adversary can exploit the CAN network to execute a replay attack. The core components involved in this setup are as follows Distance Sensor (M16 – Solid State LiDAR) is used for detecting, locating, and estimating the distance between the vehicle and potential obstacles on the road. Control Unit with Microcontroller (LPC-1768) acts as the brain of the collision avoidance system, processing data from the distance sensor and making decisions based on programmed safety thresholds. Adversary System (Raspberry Pi with Kali Linux and MCP 2515 CAN Transceiver) represents the attacker's toolkit, capable of recording and replaying CAN messages.

b) Specifics of the Replay Attack Scenario:

Distance Recording is when the adversary records the CAN messages transmitted by the distance sensor under normal driving conditions. For example, the sensor might transmit a message 100 meters from an object when it is far away. Critical Distance for Collision Avoidance is the control unit is programmed to initiate braking when the detected distance falls below 30 meters, a safety threshold to prevent collisions. Attack execution is when vehicle approaches an object and the actual distance falls below the critical 30 meters, the adversary replays the previously recorded message indicating a safe distance of 100 meters.

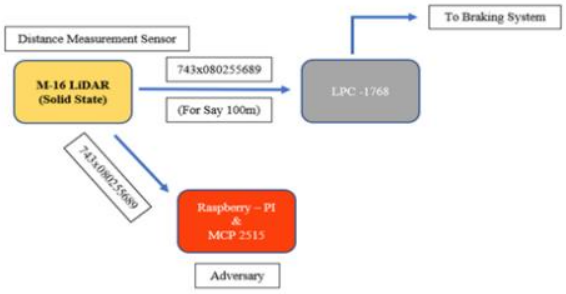


Figure 8: Record Transmit Message Scenario [10].

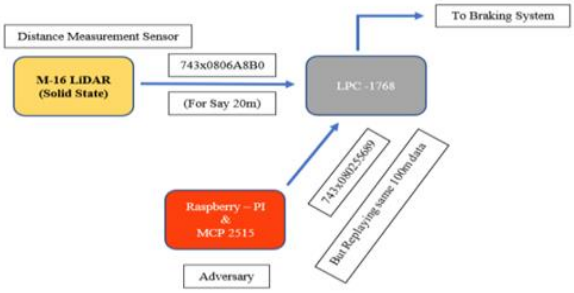


Figure 9: Replay / Playback Message Scenario [10].

c) Impact of the Replay Attack:

The control unit, deceived by the replayed message, perceives the object as being safely distant and thus fails to activate the braking system, leading to a potential collision. The inherent vulnerabilities of the CAN protocol, such as its lack of built-in encryption and authentication mechanisms, make it susceptible to various forms of cyberattacks, including replay attacks. Despite the development of countermeasures like authenticated encryption and the use of timestamps or counters to ensure message freshness, the practical implementation of these solutions varies across the automotive industry.

2) Performing reply attack by acting as ECU:

a) Overview:

Here the attack is performed on a stimulator, considering 3 nodes acting as ECUs and they are connected in a CAN bus. To achieve this in real life, the attacker should somehow get a connection to the CAN bus of the car and act as an ECU. The reply attack can be done in 2 ways, one with transmitting the same and other with manipulation [14].

b) Implementation 1: without manipulation of reply can message.

The nodes A, B and C communicate over CAN bus, where a node such as C generates and transfers data on CAN bus, nodes A and B receives the data. One of the units acts as an adversary, like node A which saves the message. Here node B uses the message, when the CAN bus is ideal the node A will transmit the saved message. Now both B and C nodes receive the spoofed message [14].

c) Implementation 2: with manipulation of reply to messages.

Before an attack is done by transmitting the same data frame sent by C. But the one has the same ID as C. Now we are changing the stored message and transmitting the manipulated message over the CAN bus. Initially node C will send the message and node A will save the message. Now the node A will manipulate the message by replacing the ID by its own or some other and transmit over the Can bus after some delay. Node C and B receive the data, node B has very few chances to detect, but there's a chance for node C to detect it by checking all the fields in the data frame [14].

d) Mitigation:

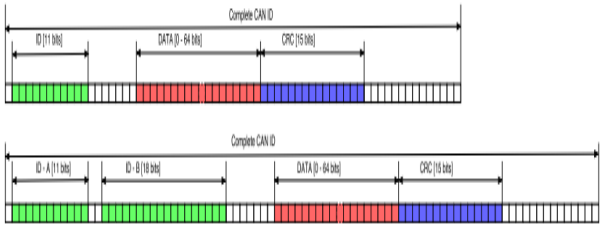
To mitigate this attack the node C, need to monitor the data and check the data frames it receives, it should check the contents in the frame for ID as it is a stored message which was originally transmitted by node C, now the node C can transmit the error message back and the reply attack can be avoided. Also, a check logic can be implemented to monitor the network and a time stamp can be implemented to avoid the reply to attacks.

C. Injection Attacks on CAN bus.

1) OVERVIEW:

Controller Area Network (CAN) systems involve unauthorized insertion of CAN messages into a self- driving or modern vehicle's network. The Message ID in a CAN bus system is crucial because it determines the recipient or the action to be performed by an ECU (Electronic Control Unit). Each message ID is associated with specific functionality or information within the vehicle, making them the primary target for attackers looking to manipulate vehicle operations. By crafting messages with specific IDs, attackers can directly target and manipulate the behavior of ECUs, leading to potential safety risks. Attackers do that by crafting messages that can either cope with legitimate operations to remain undetected or hijack vehicle functionalities.

Structure of CAN data frame:



There are two data frame formats: Base frame format 11-bit identifier and extended frame format 29-bit identifier. Each message has an ID, a payload, and a checksum. This data is organized according to blueprint known only to manufacturers. The CAN network keeps the car's components in sync, but it needs protection to prevent interference from outside sources.

a) Attacker's goal:

Controlling functionality of vehicles by compromising and altering vehicle functions such as brake system, steering, acceleration, lighting systems, etc. which could cause accidents. Safety features such as airbags, anti-lock braking system, tract control can be disabled by injecting malicious code compromising driver's safety. Access and transmission of confidential information to attackers such as location, personal media, driving routine, vehicle usage data compromises occupant's privacy. Overriding security protocols to unlock doors and engine ignition can cause vehicle theft. Manufacturer's reputation damage or to sabotage

Mechanism of Injection Attacks the process typically encompasses:

Access Point Identification: Attackers first identify access points to the vehicle's CAN network. This could be through physical ports like the OBD-II (On-Board Diagnostics II) port, wireless connections (Bluetooth, Wi-Fi), or even remotely via cellular connections.

Network Reconnaissance: After gaining access, the attacker monitors the CAN Bus traffic to understand the normal communication patterns, including the IDs and data formats of CAN messages exchanged between ECUs.

Crafting Malicious Messages: Based on the observed traffic, attackers craft malicious CAN messages. These could mimic legitimate messages (to trigger unintended actions) or be entirely novel commands designed to disrupt vehicle operation.

Message Injection: The crafted messages are then injected into the CAN network. Since the CAN protocol lacks authentication mechanisms by design, the network treats these messages as legitimate, executing the commands they contain. [16]

b) Implementation Considerations:

Centralized vs. Distributed Detection: The algorithm can be deployed in a centralized manner within gateway ECUs that monitor multiple CAN bus segments or distributed across ECUs in different segments of the bus for localized anomaly detection.

Optimization Strategies: To further reduce computational costs and improve detection speed, IDs frequently seen on the CAN bus can be prioritized in the algorithm's lookup processes.

c) Mitigation:

To effectively mitigate basic injection attacks involves establishing a baseline of normal CAN message sequences, creating a transition matrix reflective of regular vehicle operations, and Monitoring ongoing CAN traffic against the baseline to spot anomalies indicative of unauthorized injections.

2) Attack over CAN bus using vulnerabilities of ADAS:

a) OVERVIEW:

ECUs in the EVs are connected to the outside world via on-board navigation or entertainment. They play a vital role in ADAS, which significantly reduces the driver's stress load and makes the drive safe. At this stage, internal and external communication pass through the same hardware and the rules defined by software do not apply to the hardware. This opens a backdoor for the hackers [6].

b) Vehicle Investigation:

Using a CANalyzer tool make a physical connection to the CAN bus. Then read the frames that are transmitted by the CAN Bus. Then record the CAN trace (CAN log1) [6].

c) Creating CAN Database File:

This is done in two steps:

1) CAN bus Log1 is uploaded to service can2sky.com which decodes it and generate a CAN DBC1 file [6].
2) Some specific actions like acceleration and Cruise Control are performed on the vehicle and they record CAN bus Log2 [6].
   The file is analyzed to identify the byte of frames where there is a signal change related to acceleration and cruise control. Thus, a new custom CAN DBC2 file is created [6].
   The last step is to fusion these two files [6].

d) CAN Bus Load by Injection Attack [6]:

PRINCIPLE:

The Principle of This Attack Is To Send Hazard Frames Into The Can Bus With A Very Low Periodicity To Load The Bus. This Will Lead To Bus Saturation And A Total Stop Of The Vehicle Which Can Lead To A Very Dangerous Situation Especially When A High Driving Speed [6].

DEVICE USED:

ECU, Dashboard with vector panel, 7-inch touchscreen car radio, Harness grouping the 12V power supply, CAN bus with 120-ohm termination resistor [6].

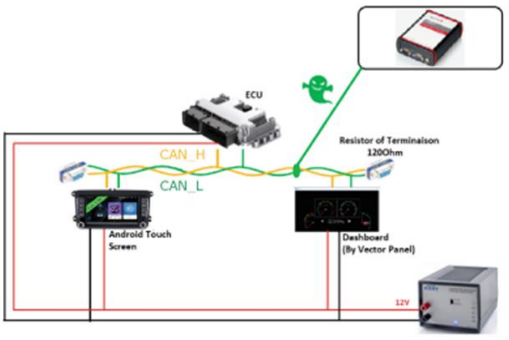


Figure 10: Block diagram Polishing the test model [6].

IMPLEMENTATION:

In this experiment, a simple frame of 1ms is sent on a 250 Kbit/s CAN bus, and at 51% of the total CAN load the bus can be overloaded and the ECU will be automatically restarted. Which is very dangerous [6].

e) Cruise Control Acceleration Attack [6]:

The cruise control system maintains the car at a desired speed. Once the driver activates the cruise control via a button, this system can take over the accelerator. So, now instead of the driver, the attacker can speed up and down the car. As a result, when the attacker accelerates or decelerates the car it can lead to a very dangerous situation [6].

3) Message Injection Attacks:

a) Overview of the Attack:

Message Injection involves unauthorized introduction of messages onto the CAN network. Since the CAN protocol lacks message authentication, any device connected to the

network can broadcast messages that appear legitimate to other ECUs (Electronic Control Units).

b) Execution of the attack:

An attacker can use a device connected to the OBD-II port or exploit a vulnerable ECU to inject malicious messages. Tools for this can range from simple microcontrollers equipped with CAN interfaces (e.g., Arduino with a CAN shield) to more sophisticated custom devices designed for stealth and remote control. Injected messages can trigger unintended actions from ECUs, such as engaging brakes, disabling safety systems, or manipulating dashboard displays.

c) Attack Implications on the Vehicle:

Injected messages can trigger unintended actions from ECUs, such as engaging brakes, disabling safety systems, or manipulating dashboard displays. The simplicity of message injection makes it a common attack vector for disrupting vehicle functionality or creating a precursor state for more complex attacks.

d) Resources Required:

Hardware: A basic microcontroller or computer with a CAN interface.

Software: Custom scripts or publicly available hacking tools designed for CAN networks.

Access: Physical or remote access to the vehicle's CAN network, often via OBD-II port.

e) Feasibility of the Attack at present times:

Due to the lack of authentication and encryption methods within the CAN protocol Message injection attacks remain possible.

4) Message Modification Attacks:

a) Overview of the Message Modification Attack:

Message Modification involves altering legitimate CAN messages in transit between ECUs without necessarily injecting new traffic. This requires intercepting communications, a task made possible by a Man-in-the-Middle (MITM) approach. Modification attacks can subtly alter vehicle behavior in dangerous ways, such as falsifying sensor readings or command signals, leading to unsafe driving conditions.

b) Execution:

A MITM device, placed in line with the CAN bus, reads legitimate messages, modifies their content, and forwards them to the intended recipient. This technique can bypass some of the inherent safety checks in CAN, such as Cyclic Redundancy Checks (CRC), by maintaining the original message structure but altering the data payload.

c) Implications:

Modification attacks can subtly alter vehicle behavior in dangerous ways, such as falsifying sensor readings or command signals, leading to unsafe driving conditions. The complexity and subtlety of these attacks make detection and mitigation challenging.

d) Resources Required:

Hardware: A sophisticated MITM device capable of real-time message interception and modification, usually involving two CAN interfaces and processing capabilities.

Software: Advanced software to analyze, modify, and forward CAN messages according to the attacker's objectives.

Access: Physical access to the vehicle's CAN network to install the MITM device.

D. Spoofing attacks on CAN bus methods.

1) Spoofing Attack Using Bus-off Attacks against a Specific ECU of the CAN Bus:

a) Spoofing attack on CAN:

In this attack, if a message is sent by the authorized ECU, purposefully set as an error, then the state is pushed to the bus-off state, and the send or receive by the authorized transmission ECU is prevented [3].

b) Bus-off Attack against an ECU:

1.BUS-OFF ATTACK USING THE BIT ERROR:

By injecting bit errors into the targeted ECU, the TEC (transmit error counter) of that ECU increases. A pointless message is inserted right before the target ECU starts transmitting. As a result, the targeted ECU transmission is stored and blocked. Simultaneously, the attacker's ECU's buffer stores messages that satisfy:

1. It must have the same CAN ID as the targeted ECU.
2. Dominant bits are sent by the attacker in response to recessive bits sent by the target's ECU.

As a result, as shown in Figure. 11, the attacker and target ECU communicate simultaneously. These ECUs' TEC increases by 8 when the transmission bits differ. The target ECU enters the bus-off state, and TEC > 255 is set by repeating this procedure.
[3].

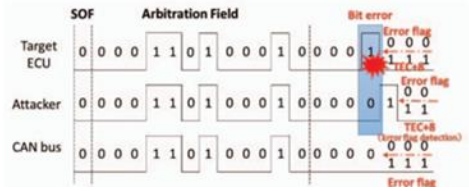


Figure 11: Bus-off attack using the bit error [3].

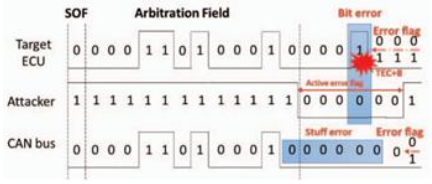


Figure 12: Bus-off attack using the Stuff error [3].

2.BUS-OFF ATTACK USING A STUFF ERROR:

The attacker verifies the data frame that is flowing on the CAN bus. If the data frame is sent from the targeted ECU, then the attacker sends an error frame into that data frame [3]. As shown in Figure 12. The target ECU detects a bit of error and increases TEC by 8. By repeating this process, the target ECU enters the bus-off state and TEC > 255.

3. BUS-OFF ATTACK IN ONE FRAME:

According to the CAN specification, '0' is the dominant state and '1' is the recursive state. If the dominant state carries on like this increase to 14 bits after error detection, the TEC increases by 8 [3]. For each time of dominant state increase in TEC of targeted ECU and this goes to bus off state (error bit 1 bit + 14 bits + 8 bits × 30), TEC > 255.

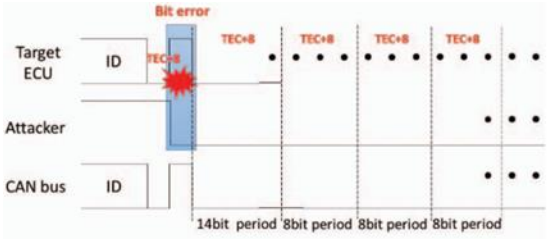


Figure 13: Bus-off attack in one frame [3].

Theory of Attack: When the targeted ECU is identified by CAN ID, it is transmitted to the bus-off state using the above bus-off attacks. Now the attacker will send the spoofed messages in the legitimate messages cycle. As a result, the receiving ECU cannot differentiate between spoofed messages and legitimate messages since it is receiving only spoofing messages [3].

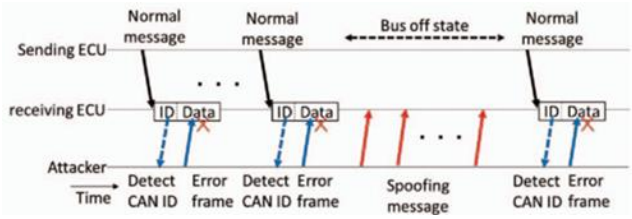


Figure 14: Spoofing attack using the bus-off attack [3].

E. Diagnostic Attack.

A diagnostic session is established on an ECU by sending a diagnostic message through Adversaries. Firmware updates or information queries on the ECU are the tasks performed by the attacker once the session is initiated. This vulnerability makes use of the ECU's built-in diagnostic mode to make maintenance operations easier. An attacker gains access to the internal functionalities of the ECU through this exploitation, through which they will gain unauthorized access to or manipulation of the vehicle's systems.

1) Attack Mechanism:

In a diagnostic attack, 10 packets are injected with CAN IDs greater than 0X700 in the files at random positions [23]. This approach probably intends to flood the ECU with many diagnostic messages, which could disrupt it or allow attackers to hide their malicious activity under the deluge of messages. It's important to consider that the specific vulnerabilities in the targeted vehicle's systems, the attacker's capabilities, and the countermeasures put in place by the vehicle manufacturer could all affect how effective and detailed this attack method is.

An attacker uses the diagnostic mode of an ECU to open a session by gaining OBD port access to the CAN bus network of a car. Internal function access is made possible by this, allowing the lock to undergo firmware manipulation and safety system compromise. They can tamper with vital systems like engine control by injecting false diagnostic messages. For detection and mitigation, secure coding techniques and strong intrusion detection are essential. This emphasizes how strict cybersecurity controls are necessary in cars to prevent access and possible safety risks.

F. Impersonation attack with Controller Area Network (CAN)

Impersonation attacks are a significant security thread in automotive systems caused by lack of message authentication in CAN protocol. These masquerading attacks as legitimate nodes in Electronic Control Units (ECU) risking safety, privacy and functions of automobiles by vehicle operation manipulation.

1) Understanding Impersonation Attacks

The CAN protocol in-vehicle communication systems, enabling ECUs to exchange messages for various vehicle functions, including safety-critical systems like braking and steering. Its design, prioritizing efficiency, and simplicity, omits message authentication, encryption, and integrity verification, allowing attackers to inject, modify, or replay messages without detection.

2) Mechanism of Impersonation Attacks

Physical access through ports like OBD-II or remotely exploiting vulnerabilities in connected services like infotainment systems is gained. The attacker can monitor and analyze CAN traffic to identify message patterns and functionalities controlled by ECU. And then crafting and sending forged messages, exploiting the lack of authentication to command unauthorized actions or overtake legitimate communication.

3) Implication

Operational Interference: Unauthorized commands can disrupt vehicle functionality, like altering dashboard displays to grave dangers like disabling brakes.

Safety Risks: Direct control over critical functions could result in catastrophic scenarios, endangering lives.

Privacy Breaches: Accessing telematics data could compromise user privacy, revealing location history and personal data.

Countermeasures: Secure CAN protocols

IV. COUNTER MEASURES

A. CAN bus attacks detection using Machine learning.

1) Overview:

Training a model based on previous data and experimenting on new cars to detect malicious content in CAN messages.

2) Implementation:

The benchmark dataset is collected from the cars KIA soul, Hyundai YF sonata and Chevrolet spark [12]. The data set consists of CAN messages and three different attacks such as Flooding, Fuzzy and Malfunction.

Each CAN message is assigned to one of the four labels as shown in the Figure 15 [12].

Target car model	Type of attack			
	Atk-free	Flooding	Fuzzy	Malfunc.
KIA	473212	120000	460000	400530
HYUNDAI	431135	96000	403299	419689
CHEVROLET	136934	85000	41000	51000

Figure 15: CAN Message data [12].

To build feature vector CAN ID, time stamp, and contents of the messages are used.

The Hybrid model is evaluated against pre-trained and retrained models with lower and upper performance bounds.

The hybrid model is an anomaly detector with auto encoder which is combined with pretrained model and implemented using Pytorch.

To evaluate the Hybrid model 2 out of 3 cars data is used as source data sets and the 3rd car acted as target, the process is repeated and changed the target car in each iteration.

The pretrained model is made up of supervised machine learning techniques with a mixture of attack and attack free data from the sources. The retrained model is made of an attack-free database of target car. The hybrid is created with an auto-encoder and trained using attack-free data from target car and merged with pretrained model [12].

After the training the models are tested against the target data which has everything included.

The hybrid model showed good results compared to the other two, however detecting malfunctions was challenging due to the usage of normal data which made the model too cautious.

Target car	Pretrained	Hybrid (proposed)	Retrained
KIA	0.828	0.988	0.989
HYUNDAI	0.893	0.976	0.994
CHEVROLET	0.608	0.813	0.935

Figure 16: Results of detection (Accuracy) by Pretrained, Retrained and Hybrid Models [12].

B. Intrusion Detection For Different Message Injection Attacks.

Vehicle security has become more crucial as conventional cars have evolved into computerized systems with network connectivity. To guarantee the security of both drivers and passengers, it is essential to protect cars from attacks. Many research efforts have concentrated on identifying and preventing attacks directed towards automobiles.

To detect anomalies some methods for in-vehicle intrusion detection includes usage of entropy-based anomaly detection techniques, defining attack detection based on protocol and ECU behavior, utilizing a structured approach with multiple sensors, and analyzing message rates. However, due to the limited computing power in vehicles, early methods that required large message datasets for analysis may have resulted in delayed detection times.

A lightweight intrusion detection method has been proposed to address this problem by streamlining detection algorithms for faster responses and lower computing power consumption in vehicles.

1) Light Weight IDS:

a) Threat model:

1. To identify known attack signatures and unusual events in vehicle networks, a hybrid intrusion detection system has been suggested. Its main objective is to detect message injection attacks by examining anomalies in traffic that are related to the frequency of messages. Attackers can still manipulate electronic devices like ECUs by injecting messages into

CAN even though it does not contain source or target information in its messages.

2. In Normal status, ECU-generated messages have consistent frequencies or intervals. Nevertheless, these frequencies or intervals unexpectedly alter during an injection attack. The message rate on the network significantly rises because of ECUs sending messages cyclically while attackers inject messages.
3. The detection method depends on message rate monitoring, but attacks take time to manifest before they are discovered. To address this, a more rapid response time while retaining high accuracy is achieved by streamlining the detection process.
4. There are two types of CAN injection attacks: standard message injections that mimic ECU messages and CAN diagnostic message injections. For experimental purposes, these attacks are divided into three categories: Type 1 involves the injection of a single CAN ID message, Type 2 includes the injection of random or pre-ordered messages of multiple CAN IDs, and Type 3 involves massive message injection, like a Denial of Service (DoS) attack. While each type has a distinct function, they are all executed similarly.

Three types of attacks utilizing message injection are analyzed [21]:

Type 1: Constantly injecting a particular message to interfere with how the vehicle operates. The process of detection includes finding messages with shorter time intervals.

Type 2: Using pre-ordered or random messages containing multiple CAN IDs to cause system malfunction. Finding clusters of messages with shorter time intervals is one aspect of detection.

Type 3: Overwhelming communication through a massive message injection. Monitoring the message transmission rate and flagging anomalies, like messages with intervals shorter than 0.2 milliseconds, are key components of detection.

2) Intrusion Detection:

To identify message injection attacks, the IDS system focuses on examining the time intervals of messages sent over the CAN bus.

This is a summary of the detection process:

1. Whenever a new message shows up on the CAN bus, first verifying the CAN ID, the IDS calculates the time interval of the latest message from its arrival time.
2. The IDS recognizes a new message as injected if its time interval is less than half of the average, which is shorter than usual.
3. If consecutive messages have a time interval of less than 0.2 milliseconds, raises the DoS attack score by 1 per message.
4. If the score exceeds a predetermined threshold, the event is classified by IDS as a denial-of-service attack.
5. On normal status, messages are sent every 0.5 milliseconds on average, and they are sent every 0.14 milliseconds on average. In order to lower the false positive ratio in DoS attack detection, a threshold is used because some normal messages have time intervals shorter than 0.2 milliseconds.

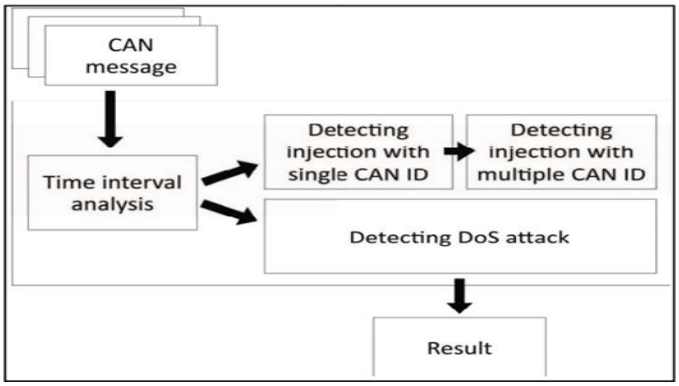


Figure 17: Proposed IDS [21].

After analysis of time interval of each detection module. The one is detecting or malfunction. Another one is detecting DoS attack to disturb CAN communication.

C. Secure CAN standard

1) CANsec is composed of hardware and software written mainly in Python.

1. CAN with Flexible Data-Rate (CAN-FD): CAN-FD extends the payload size and increases the possibility for more sophisticated security, including encryption and authentication and more in extended space in payload [34].
2. Controller Area Network Security (CANsec): The concept of CANsec introduces security layers on top of CAN, proposing message authentication and encryption using Hash-based Messages Authentication (HMAC) or symmetric key cryptography to validate the integrity and origin of messages.
3. Local Interconnect Network (LIN) and FlexRay: While not direct evolutions of CAN, these networks offer alternative communication protocols with potential for enhanced security features. They support more complex communication patterns and, in the case of FlexRay, provide built-in support for

encryption and error detection, which could mitigate some impersonation threats when used for critical communications.

CANsec is described as a sophisticated tool designed for the security evaluation of Controller Area Network (CAN) bus systems in In-Vehicle Networks (IVNs). The tool aims to address and analyze various vulnerabilities and attack vectors that can be exploited within the CAN bus system, which is a critical component of modern automotive technologies. CANsec offers a comprehensive suite of functionalities to assess and improve the security posture of CAN bus systems.

2) Core Functionalities:

1. Traffic Tracking, Transmitting, Logging, and Monitoring: CANsec is equipped to track real-time CAN traffic, transmit specific CAN messages, log traffic data for further analysis, and monitor the system for any anomalies or changes in vehicular status.

2. Evaluation Vectors: The tool supports 11 evaluation vectors targeting various assets and potential vulnerabilities within IVNs. These vectors are designed to comprehensively assess the security of the CAN bus by simulating different types of attacks.

3. Flexible Configuration: Users can select and configure evaluation vectors according to specific testing needs. This adaptability allows for tailored security assessments in diverse evaluation scenarios.

4. Data Processing Module: This module processes CAN traffic data to be displayed, enabling users to analyze and interpret the information more effectively.

5. Status Monitor: It keeps track of the vehicle’s status changes during evaluations, particularly when executing fuzzy evaluations to test the system's resilience against uncertain or unpredictable inputs.

6. Logging Module: Critical events, such as changes in vehicular status or software crashes, are logged for detailed incident analysis and forensic purposes.

7. Communication Layer: This layer facilitates the transmission and reception of CAN application data, enabling communication with the target vehicle for testing and evaluation purposes.

8. Hardware Support: Primarily, CANsec is designed to work with cost-effective CAN USB adapters, making it accessible for a wide range of security researchers and professionals.

3) Detailed Operation of CANsec:

CAN Architecture Scan: The first step involves scanning the vehicle's CAN architecture to understand its structure and identify all active CAN IDs.

ECU Drop-Off: This vector aims to test the resilience of Electronic Control Units (ECUs) by attempting to disrupt their communication.

Packet Analysis and Reverse Engineering: CANsec can reverse-engineer CAN traffic based on frame frequency and bit features to identify critical CAN IDs associated with specific vehicular actions or to decipher the communication matrix.

Diagnostic Services Evaluation: It evaluates the security of diagnostic functionalities by scanning for available diagnostic services and attempting to reverse-engineer or replay diagnostic commands.

Fuzzing: CANsec conducts fuzzing tests to uncover unknown vulnerabilities by sending mutated packets and observing the vehicle's response.

4) Advantages:

Comprehensive Evaluation: Unlike basic CAN traffic generation tools, CANsec offers an elaborate set of evaluation vectors for a thorough security assessment.

Reverse Traffic Functionality A unique feature that distinguishes CANsec from other tools is its ability to reverse-engineer CAN traffic, providing valuable insights into the communication matrix and enabling precise manipulation of CAN traffic.

In essence, CANsec stands out as a versatile and powerful tool for the security analysis of CAN bus systems in vehicles, providing a holistic approach to identifying vulnerabilities, testing ECUs' resilience, and enhancing the overall security framework of in-vehicle networks.

5) Implementation Considerations

Adding security layers to CAN network affect real-time performance, a critical consideration in automotive environments. Ensuring that security enhancements are compatible with existing infrastructure is crucial. This could mean incremental updates to ECUs or employing gateways that provide security functions without altering the entire network architecture [34]. Widespread adoption of secure protocols requires industry-wide standardization. Efforts by consortia like AUTOSAR, which defines standardized software architecture for automotive systems, are vital in promoting and implementing secure communication protocols.

V. CASE STUDIES

These case studies highlight how crucial it is to handle cybersecurity threats in automobile systems, especially as these vehicles grow more digitally connected and dependent. To guard against potential attacks and guarantee the security and privacy of vehicle users, cybersecurity measures must be integrated into the design and development phases as well as be updated and patched regularly.

A. Case Study 1: The Jeep Cherokee Hack

Researchers Charlie Miller and Chris Valasek shocked the automobile industry in 2015 by successfully carrying out a full remote takeover of a Jeep Cherokee in a major cybersecurity experiment. This demonstrated the critical necessity for strict cybersecurity standards in modern vehicles. By taking advantage of flaws in the Uconnect infotainment system, which was linked to the internet through a cellular network, the two were able to remotely access the internal network of the car. After getting inside, they were able to control the transmission, steering, brakes, and dashboard, which led to them remotely disabling the motor while a journalist was driving on a busy highway.[24]

1) Case Study 2: Tesla's Remote Keyless System Breach

Researchers from KU Leuven University in Belgium tested the security of Tesla Model S cars during their study. The group was able to duplicate the key fob and obtain unauthorized entry to the vehicle after they found a significant weakness in the encryption system. The communication between the car and its key fob was intercepted by the researchers using a combination of readily available devices that came at a cost of less than $600. Without getting into the automobile, they could unlock it and turn on the engine by deciphering the encryption and copying the signal from the key fob. Due to flaws in wireless key fobs that were made public by this assault, Tesla updated their cryptographic key fob system to a more secure version and added new security features including the "PIN To DRIVE" element.[25]

2) Case Study 3: BMW's ConanectedDrive Vulnerability

A security vulnerability in BMW's ConnectedDrive system was found by German Automobile Club (ADAC) experts in 2015. This vulnerability enabled hackers to remotely unlock the doors of up to 2.2 million cars. The system's unencrypted communication with BMW servers was the source of the vulnerability. Potential attackers may have impersonated the BMW server and instructed cars to unlock thanks to this error. To mitigate the risk and avoid the need for a physical recall of the impacted vehicles, BMW quickly implemented end-to-end encryption for the communication between vehicles and the BMW servers after learning of this discovery.[26]

3) Case Study 4: Nissan LEAF's Mobile App Exploit

Security researcher Troy Hunt discovered a flaw in the Nissan LEAF companion smartphone app in 2016 that permitted unauthenticated remote access to the car's controls and telemetry data. All it would take for an attacker to remotely regulate the air conditioning, examine driving records, and check the battery state is to know or surmise the Vehicle Identification Number (VIN). Significant worries regarding the security and privacy of Internet of Things (IoT) devices in cars were brought up by this issue. To fix the security flaws, Nissan temporarily stopped the app. Later, it was updated with the correct authentication methods.[27]

VI. EMERGING TECHNOLOGIES AND THEIR IMPACT ON AUTOMOTIVE CYBERSECURITY

VII. CONCLUSION

The survey on recent attacks of in-vehicle Controller Area Network (CAN) bus navigate complexity in aspect of security in modern automobiles. Integrity in advance computing and connectivity features is increasing, attack surface for cyber threads is widened. The need for robust cybersecurity frameworks for protection of integrity and privacy of in-vehicle networks. Thus, conclusion synthesized from reviewing literature and proposed future work directions in automotive cybersecurity is as follows.

The automotive networks evolution from point-to-point wiring systems to organized CAN bus architectures has significantly improved vehicle functionality and efficiency. However, CAN network security vulnerabilities are introduced. Attack vectors such as Denial of Services (DOS). Replay attacks, Message Injection attack, Hardware Trojan threats, shows potential risk to compromise vehicle operations, endangering passenger safety and vehicle integrity. Lack of security features within CAN protocol, such as encryption and authentication, leaving vulnerabilities leading to cyberattacks.

Countermeasures with machine learning and intrusion detection systems (IDS) show promising mitigating the risks. Machine learning models, especially with hybrid approaches algorithms give straightforward indication of anomalous behavior. Lightweight intrusion detection system offers practical solutions for real-life threat detection in vehicles.

VIII. RELATED FUTURE WORK

The technological development and the increasing specialized cyber threats increase the need for a proactive and innovative approach in building in-vehicle network security. Future work in automotive network security should focus on following key areas:

Enhancement in security of CAN protocol by standardized development of encryption, authentication, and integrity to significantly mitigates the cyberattack risks. Cryptographic solutions research for computational resources

of in-vehicle systems is crucial. The advanced machine leaning model for detection of cyber threats warrants can be explored. Future researchers should goal for improving accuracy to efficiency of machine learning models to differentiate normal network behavior and sophisticated attacks in network. The Cross-vehicle model and its in-vehicle network requires developing one IDS which is adaptable to different architecture. Hardware threats such as hardware trojan should be addressed by securing the hardware component in in-vehicle networks. Research should focus on securing hardware design and tamper resistance technologies. Methodologies such as detection and isolation of components. Unauthorized control of data should be secured. Development of data privacy frameworks so the owner of vehicles has data control. Automotive manufacturers, developers, cybersecurity experts to establish standards and regulations to anticipate future vulnerabilities along with current threats. Spread awareness along vehicle owners about associate risk with connected vehicles and induce practices to Maintenace to priories cybersecurity for detection and prevention of attack at early stage. In conclusion, as the automotive industry continues to evolve, the approach to ensure security of in-vehicle networks should evolve.

IX. REFERENCES

[1] Prevost, S., & Kettani, H. (2019). On Data Privacy in Modern Personal Vehicles. In Proceedings of the BDIoT'19, October 2019. ACM. https://doi.org/10.1145/3372938.3372940

[2] Rathore, R. S., Hewage, C., Kaiwartya, O., & Lloret, J. (2022). In-Vehicle Communication Cyber Security: Challenges and Solutions. Sensors, 22(17), 6679. https://doi.org/10.3390/s22176679

[3] Iehira, K., Inoue, H., & Ishida, K. (2018, January). Spoofing attack using bus-off attacks against a specific ECU of the CAN bus. In 2018 15th IEEE annual consumer communications & networking conference (CNC) (pp. 1-4). IEEE.

[4] Kumar, S. V., Mary, G. A. A., Suresh, P., & Uthirasamy, R. (2021, February). Investigation on cyber-attacks against in-vehicle network. In 2021 7th International Conference on Electrical Energy Systems (ICEES) (pp. 305-311). IEEE.

[5] Lee, Y., & Woo, S. (2022). Can signal extinction-based dos attack on in-vehicle network. Security and Communication Networks, 2022.

[6] Zerouali, S., Messaoudi, A., Rabhi, A., Karrouchi, M., & Nasri, I. (2023, January). Implementation and Demonstration of Threat Over CAN Bus for Automotive Embedded System Equipped with ADAS. In International Conference on Digital Technologies and Applications (pp. 748-756). Cham: Springer Nature Switzerland.

[7] Bozdal, M., Randa, M., Samie, M., & Jennions, I. (2018). Hardware trojan enabled denial of service attack on can bus. Procedia Manufacturing, 16, 47-52.

[8] Palanca, A., Evenchick, E., Maggi, F., & Zanero, S. (2017). A stealth, selective, link-layer denial-of-service attack against automotive networks. In Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14 (pp. 185-206). Springer International Publishing.

[9] Woo, S., Jo, H. J., & Lee, D. H. (2014). A practical wireless attack on the connected car and security protocol for in-vehicle CAN. IEEE Transactions on intelligent transportation systems, 16(2), 993-1006.

[10] Chandrasekaran, S., Ramachandran, K. I., Adarsh, S., & Puranik, A. K. (2020, July). Avoidance of Replay attack in CAN protocol using Authenticated Encryption. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[11] Gazdag, A., Ferenczi, C., & Buttyán, L. (2020, November). Development of a man-in-the-middle attack device for the can bus. In Proceedings of the 1st Conference on Information Technology and Data Science (pp. 115-130).

[12] Thirumavalavasethurayar, P., & Ravi, T. (2021, March). Implementation of replay attack in controller area network bus using universal verification methodology. In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) (pp. 1142-1146). IEEE.

[13] Nakamura, S., Takeuchi, K., Kashima, H., Kishikawa, T., Ushio, T., Haga, T., & Sasaki, T. (2021, September). In-vehicle network attack detection across vehicle models: A supervised-unsupervised hybrid approach. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC) (pp. 1286-1291). IEEE.

[14] Mukherjee, S., Shirazi, H., Ray, I., Daily, J., & Gamble, R. (2016). Practical DoS attacks on embedded networks in commercial vehicles. In Information Systems Security: 12th International Conference, ICISS 2016, Jaipur, India, December 16-20, 2016, Proceedings 12 (pp. 23-42). Springer International Publishing.

[15] Zeinab El-Rewini, Karthikeyan Sadatsharan, Daisy Flora Selvaraj, Siby Jose Plathottam, Prakash Ranganathan Cybersecurity challenges in vehicular communications. School of Electrical Engineering and Computer Science (SEECS), University of North Dakota, Grand Forks, ND, USA b Energy Systems Division, Argonne National Laboratory, IL, USA

[16] Mirco Marchetti, Dario Stabili. Anomaly detection of CAN bus messages through analysis of ID sequences. June 2017. Conference: 2017 IEEE Intelligent Vehicles Symposium (IV)

[17] EMAD ALIWA, OMER RANA, CHARITH PERERA, and PETER BURNAP. Cyberattacks and Countermeasures for In-Vehicle Networks , Cardiff University, UK

[18] Salunkhe, P. D., Patil, U. N., Patel, N. J., & Sontakke, P. V. (2022). DoS attack detection and mitigation for autonomous vehicles using Raspberry Pi. Int. J. Res. Appl. Sci. Eng. Technol., 8(9), 657-666.

[19] Cho, K. T., & Shin, K. G. (2016, October). Error handling of in-vehicle networks makes them vulnerable. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 1044-1055).

[20] Narayan Khatri, Rakesh Shrestha and Seung Yeob Nam, Security Issues with In Vehicle Networks and Enhanced Countermeasures Based on Blockchain.

[21] Song, H. M., Kim, H. R., & Kim, H. K. (2016, January). Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In 2016 international conference on information networking (ICOIN) (pp. 63-68). IEEE.

[22] Liu, J., Zhang, S., Sun, W., & Shi, Y. (2017). In-vehicle network attacks and countermeasures: Challenges and future directions. IEEE Network, 31(5), 50-58.

[23] Dupont, G., Den Hartog, J., Etalle, S., & Lekidis, A. (2019, November). Evaluation framework for network intrusion detection systems for in-vehicle can. In 2019 IEEE

International Conference on Connected Vehicles and Expo (ICCVE) (pp. 1-6). IEEE.

[24] Miller, C., & Valasek, C. (2015). Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat USA, 2015.

[25] Wouters, K., Mitev, R., Meerts, L., Quax, P., & Preneel, B. (2018). A Practical Attack on the MIFARE Classic. Cryptology ePrint Archive, Report 2018/127.

[26] German Automobile Club (ADAC). (2015). Security vulnerabilities in BMW ConnectedDrive.

[27] Hunt, T. (2016). Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs.

[28] K. Apruzzese, M. Colajanni, L. Ferretti, M. Marchetti, "On the Effectiveness of Machine Learning for Cybersecurity," IEEE Xplore, 2018.

[29] J. Giraldo, E. Sarkar, "Adversarial Machine Learning in Cybersecurity and Recommendations for Future Research," Frontiers in Computer Science, 2020.

[30] Y. Liu, C. Corbett, K. Chiang, "Machine Learning in Cybersecurity: A Comprehensive Survey," Journal of Computer Security, 2021.

[31] Leiba, O., Yitzchak, Y., & Bitton, R. (2019). Securing Vehicle-to-Vehicle Communications with Blockchain Technology. IEEE Communications Magazine.

[32] Dorri, A., Kanhere, S.S., Jurdak, R., & Gauravaram, P. (2017). Blockchain for IoT Security and Privacy: The Case Study of a Smart Home. IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops).

[33] Singh, M., Kim, S. (2018). Blockchain-Based Secure Firmware Update for Embedded Devices in an Internet of Things Environment. The Journal of Supercomputing.

[34] Zhang, H., Xu, M., Zhang, X., & Liu, Z. (2020). CANSEC: a practical In-Vehicle Controller area network security evaluation tool. *Sensors*, *20*(17), 4900. https://doi.org/10.3390/s20174900

APPENDIX

Team Members Contributions:

Teammate	**Contribution**	**Comments (if any)**
Aniket Agarwal (40266485)	Introduction, 1 attack,1 Countermeasures using CANsec and final editing of paper	ALL TEAM MEMBERS HAVE EQUAL CONTRIBUTION
Kalyani Batle (40243967)	1 Attack, 1 Countermeasures on Anomaly detection. Conclusion with future work	
Teja Venkata Sai Manikanta Sirigidi (40265966)	2 Attacks (CAN signal based DOS attack, Spoofing attack using Bus-off attacks)	
Geeshma Sri Sai Maddipati (40277629)	3 Attacks (Hardware Trojan DOS, Injection attack, Link Layer DOS) with mitigation	
Charan Tej Cheedella (40261465)	1 Attack detection (DoS attack for automotive cars using Raspberry) with its mitigation, Editing	
Sai Mahitha Bheemineni (40271856)	2 Attacks (Replay attack, Message Injection and Modification attacks)	
Chandra Vamsi Sekhar Peketi (40277985)	Attacks –2 Counter measures – 3 (DOS attack, Replay attack and IDS wit ML)	
Karunyamruta Subash Anguluri (40266020)	2 Attacks (Message injection attack, Diagnostic attack) and 1 Countermeasure (Message injection attack detection using IDS)	
Manikanta Chinthala (40271167)	Abstract, Editing, Countermeasures	
Alaa Alsharif (40279078)	Background Information Section, Emerging Technologies and Their Impact on Automotive Cybersecurity	