Cybersecurity Assessment of CAN Networks and Alternatives: A Comprehensive Survey on Recent Attacks and Resource Requirement

Pratiksha Ashok Kumar Pole ID 40230412
Deep Bhavesh Gajiwala ID 40231725
Rohan Yogeshkumar Modi ID 40255454
Snehpreet Kaur ID 40254443
Saket Suman ID 40225128
Meet Rakeshbhai Patel ID 40239187
Devina Shah ID 40238009
Simran Kaur ID 40241517
Karanjot Singh ID 40220861

Abstract—**Automotive control networks, anchored by the Controller Area Network (CAN) and its variants, constitute the digital backbone of modern vehicles, overseeing a spectrum of critical functions. As the automotive industry embraces connectivity and automation, the security of in-vehicle control networks emerges as a paramount concern. Potential cyber-attacks on these networks can have far-reaching consequences, from compromising privacy to jeopardizing vehicle safety. This survey analyses an in-depth examination of recent attacks on automotive networks, encompassing CAN, Local Interconnect Network (LIN), and FlexRay, to identify enduring vulnerabilities and quantify the computational and communication resources required for potential. The insights gained from this analysis will inform the development of advanced security measures, enhancing the integrity of in-vehicle control networks and fortifying the safety and reliability of modern automobiles.**

***Index Terms*—Automotive Control Networks, Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay, Cybersecurity, Attack Vectors, Vulnerabilities, Threat Landscape, Security Measures, In Vehicle Control Networks.**

I. Introduction

Overview of Automotive Networks

Automotive networks are the intricate web of communication protocols that connect electronic control units (ECUs) within a vehicle. The Controller Area Network (CAN) serves as the backbone, facilitating real-time data exchange for critical functions, including powertrain control, safety systems, infotainment, and more. Alongside CAN, alternative protocols like Local Interconnect Network (LIN) and FlexRay cater to specific requirements. The central challenge lies in securing these networks against cyber threats, as vulnerabilities could lead to unauthorized access, data manipulation, or system compromise. This project delves into recent attacks, assessing remaining vulnerabilities, and gauging the computational and communication resources required for potential breaches, ultimately reinforcing the security of automotive control networks. The significance of security in automotive networks cannot be overstated, as it stands at the crossroads of safety, privacy, and the future of transportation. In an era marked by rapid technological advancement, modern vehicles are no longer merely mechanical marvels; they are complex digital eco-systems on wheels. These vehicles rely on intricate networks of electronic control units (ECUs) and sensors to manage everything from engine performance and safety systems to infotainment and connectivity. While this digital transformation promises greater convenience and efficiency, it also ushers in a new era of cybersecurity challenges. The integrity of in- vehicle control networks is a linchpin in ensuring the safety of occupants, the privacy of data, and the reliability of the automotive industry. This discussion explores the multifaceted significance of security in automotive networks, shedding light on the critical factors that underpin the imperative need for robust and comprehensive cybersecurity measures.

The vulnerability assessment of the CAN protocol reveals shortcomings in ensuring confidentiality, integrity, and availability. The protocol's lack of cryptographic methods compromises data confidentiality, while issues in integrity checks and the priority-based messaging system impact the integrity and availability of communicated data. The discussion extends to the expansion of the automotive attack surface, detailing both physical access attacks (via the OBD port, selective DoS attacks, and indirect access methods) and remote access threats (exploiting wireless interfaces, OTA updates, and V2V/V2I communications). The classification of attacks on in-vehicle network systems identifies multiple entry points susceptible to security breaches, including various ports and communication interfaces.

In a detailed examination of notable attacks, the content explores real-world examples such as the lock picking attack on keyless entry systems, TPMS exploitation, road infrastructure attacks, and specific vulnerabilities in the CAN bus system. The Jeep hack via cellular network serves as a landmark case, illustrating the potential dangers of remote attacks on interconnected vehicle systems. Other instances include manipulation via the OBD-II port, selective DoS attacks, and vulnerabilities in the SAE J1939 standard. The content underscores the critical need for enhanced security measures in the automotive industry, considering the evolving attack surfaces, potential consequences, and the increasing complexity of in-vehicle communication networks.

II. OVERVIEW OF CAN, LIN, AND FLEXRAY NETWORKS

Securing automotive networks is paramount due to the potential implications of cyberattacks. Threats include unauthorized access, data manipulation, and system compromise. Attack vectors may exploit software vulnerabilities, hardware weaknesses, or weak network segmentation. Given the criticality of vehicle functions, any security breach can lead to dire consequences, ranging from loss of privacy to physical harm.

Controller Area Network (CAN) and its alternatives, namely Local Interconnect Network (LIN) and FlexRay, play a pivotal role in the automotive and industrial automation industries due to their distinct characteristics and applications.

1) **Controller Area Network (CAN):** It is a widely used communication protocol known for its robustness and efficiency. Its importance and usage in various domains are as follows:

Automotive Industry: CAN is the backbone of automotive networks, facilitating real-time communication between electronic control units (ECUs) that control various vehicle functions. It ensures the seamless operation of engine control, powertrain management, safety systems, and more.

Industrial Automation: It is employed in industrial automation, connecting PLCs, sensors, and actuators. Its determinism and reliability make it suitable for time-sensitive industrial processes.

The Controller Area Network (CAN) is an essential part of vehicle communication systems. Initially developed by Bosch in the 1980s for automotive applications, CAN has since become a standard in various industrial control environments. The primary purpose of CAN is to allow multiple microcontrollers and devices within a vehicle to communicate with each other without requiring a central computer. This feature makes it highly effective in managing complex operations where multiple subsystems need to interact seamlessly.

The network's physical layer usually consists of two wires forming a twisted pair, which helps in reducing electromagnetic interference. This robust design enables the CAN network to function reliably in the harsh electrical environments of vehicles. to prevent faulty nodes from disrupting the entire network. In terms of security, while CAN provides robust data transmission, it was not designed with security features to prevent malicious attacks. This lack of inherent security has become a concern in modern automotive systems, where the threat of cyberattacks is rising cost and compatibility difficulties all focus on this point.

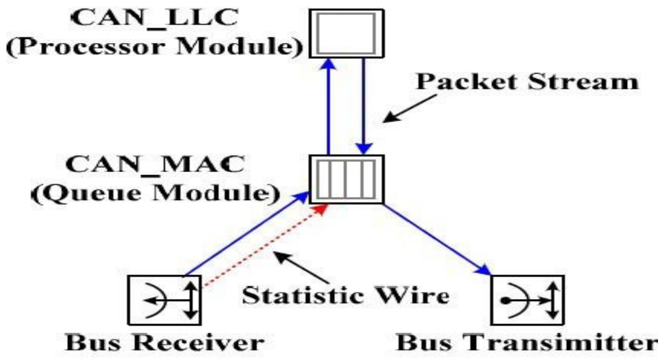


Fig. CAN NODE Architecture

2) **Local Interconnect Network (LIN):** LIN is an alternative to CAN with its own set of importance and applications:

Supplement to CAN: LIN is often used in conjunction with CAN in vehicles. It complements CAN by managing non-critical, low-speed functions like interior lighting, climate control, and infotainment systems. This ensures that CAN resources are reserved for critical tasks.

Energy Efficiency: LIN is designed with energy efficiency in mind, making it suitable for functions that must run continuously with minimal power consumption. The Local Interconnect Network (LIN) is a simpler, more cost-effective alternative CAN designed for less critical communication tasks within vehicles. Developed in the late 1990s, LIN is primarily used for managing simple actuators and sensors, such as mirror adjustments, seat positions, and rain sensors.

LIN operates as a single-master, multiple-slave network, where a central master unit controls communication with several slave nodes. This architecture simplifies the network design and reduces costs, making LIN an ideal choice for simpler and lower-speed applications. The communication speed in a LIN network is typically around 20 kbit/s, which is sufficient for the non-time-critical tasks it manages. A notable feature of LIN is its single-wire design, contrasting with the two-wire design of CAN.

Since LIN is used in less critical functions, the security risks are generally lower. However, as with CAN, the increasing connectivity of automotive systems raises the need for improved security measures in LIN networks as well.

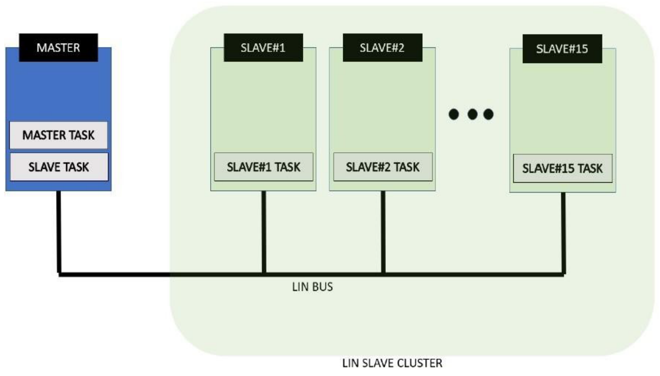


Fig. LIN architecture

3) **FlexRay:** FlexRay is a high-speed communication protocol that serves specific high-performance applications:

Advanced Driver Assistance Systems (ADAS): FlexRay is crucial for real-time, safety-critical functions in advanced driver assistance systems, like adaptive cruise control and lane-keeping assistance. It provides the necessary determinism and redundancy for these applications.

Redundancy and Fault-Tolerance: FlexRay's dual-channel design ensures redundancy and fault-tolerance, making it suitable for critical applications where system failure is not an option.

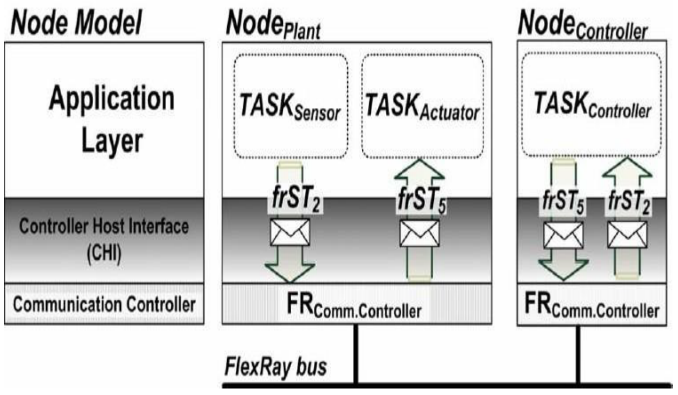


Fig. FlexRay Bus

FlexRay is the most advanced protocol among the three, designed to cater to the needs of complex and safety-critical systems in modern vehicles. Developed in the early 2000s by a consortium of automotive companies and suppliers, FlexRay addresses the limitations of CAN and LIN in handling high-speed and reliable communication required in advanced applications like x-by-wire systems.

III. SPECIFIC ATTACK VECTORS AND METHODOLOGIES

1. Vulnerability Assessment of the CAN Protocol

The Controller Area Network (CAN) protocol, a critical component in automotive communication systems, has been subjected to extensive vulnerability assessments due to its significant role in vehicle functionality and safety. A detailed analysis based on the principles of confidentiality, integrity, and availability reveals inherent weaknesses in the protocol, making it susceptible to various cyber-attacks.

• *Confidentiality Concerns*: Confidentiality, the principle of ensuring data accessibility only to authorized entities, is compromised in the CAN protocol due to its lack of inherent cryptographic methods. This omission allows intruders to access sensitive user data, leading to potential invasions of privacy and unauthorized data breaches.

• *Integrity Issues:* Integrity in data communication entails the assurance of data accuracy, completeness, and validity. The CAN bus utilizes a Cyclic Redundancy Check (CRC) for the verification of data integrity against transmission errors. However, this mechanism is insufficient in preventing data injections by malicious entities, leading to a breach of data integrity. The protocol's lack of comprehensive integrity checks results in its inability to sustain the integrity of communicated data.

• *Availability Vulnerabilities:* The principle of availability implies that authorized users should have consistent and reliable access to the system. In the CAN protocol, the priority-based messaging system can be exploited, wherein messages of the highest priority can dominate the network. This manipulation results in making the network inaccessible to lower-priority nodes, thereby violating the principle of availability.

2. Automotive Attack Surface Expansion

The evolution of automotive technology, especially the increase in embedded electronics, has led to a proportional rise in the attack surface for potential cyber threats. The integration of advanced electronic components and systems, such as sensors, actuators, control units, and communication systems, has transformed vehicles from closed to open systems. This transformation has introduced complexities in vehicle communication and expanded the potential for cyber-attacks, which can now be executed remotely without physical access to the vehicle.

• *Types of Attacks:* Automotive cyber-attacks can be categorized into physical access attacks and remote access attacks. Physical access attacks require direct interaction with the vehicle's network, typically through the On-Board Diagnostic (OBD) port or by installing a malicious node in the vehicle's network. Remote access attacks are executed via wireless communication interfaces, such as Bluetooth, Wi-Fi, and cellular networks, allowing attackers to exploit vulnerabilities from a distance.

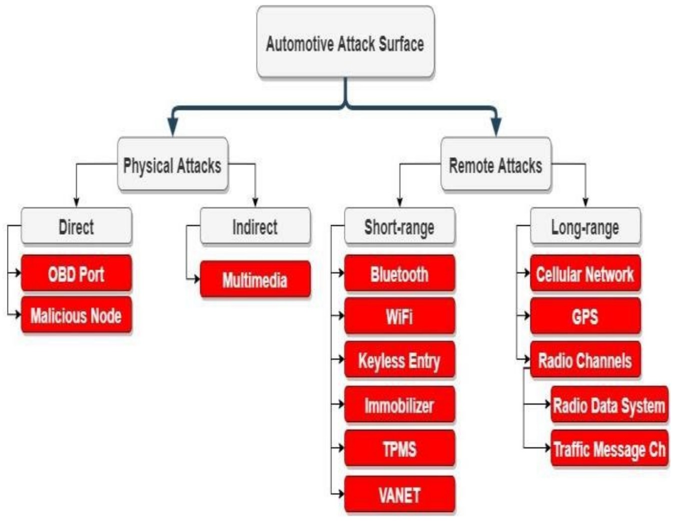


Fig. Automotive Attack Surface

3. Notable Physical Access Attacks

Physical access attacks involve direct interaction with the vehicle's network systems, often through accessible ports or by installing unauthorized devices within the vehicle's network.

• On-Board Diagnostics (OBD) Port Attacks: The OBD port, being a direct gateway to the vehicle's network, is a primary target for attackers. By exploiting the OBD port, attackers can manipulate various vehicle modules, including critical systems like brake and engine control. Such attacks have demonstrated the ability to release brakes, prevent brake activation, manipulate instrument clusters, change engine parameters, and even disable the engine while the vehicle is in motion.
• Selective Denial-of-Service (DoS) Attacks: These attacks disrupt the network without necessitating full message transmission. They can be executed by overwriting specific bits in the transmitted data, thereby generating transmission errors, and exploiting the vulnerability of the CAN standard. Research in this area has focused on exploiting these vulnerabilities, leading to government alerts and increased awareness of the susceptibility of vehicles to such attacks.
• Indirect Physical Access Attacks: These attacks do not require direct access to the vehicle's network. For instance, hacking the IT system of a car service can provide indirect access to the CAN. Another method includes attacking via multimedia devices like CDs, USBs, or MP3 players. While these attacks may not directly breach the CAN, they can affect the driver by flashing warnings on the screen or playing alarm signals.

4. Remote Access Attacks

Remote access attacks represent a significant threat in modern vehicles due to the integration of various wireless interfaces necessary for communication with systems like anti-theft devices, tire pressure monitoring systems (TPMS), Bluetooth, and telematics units.

• Exploiting Wireless Interfaces: These interfaces, typically connected to the CAN via a gateway ECU, have been demonstrated as vulnerable points for hacking. Successful compromises of these systems can lead to unauthorized control over the vehicle, including unlocking doors and manipulating vehicle functions remotely.
• OTA Software Update Vulnerabilities: Over-the-Air (OTA) updates, while convenient and cost-effective for software delivery, present another attack surface. Hackers can potentially intercept these updates to infiltrate the vehicle's communication network, leading to ransomware attacks or other forms of cyber sabotage.
• V2V and V2I Communications: The advent of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, which are integral to vehicular ad hoc networks (VANETs), introduces new vulnerabilities. These systems, designed for traffic optimization and collision avoidance, can be compromised by spoofed messages, resulting in disruptions to in-vehicle communication networks.

5. Classification of Attacks on In-Vehicle Network System

The in-vehicle network system is prone to various attacks, with the following being the most significant entry points for attackers:
• *OBD-II Port:* This port, used for monitoring vehicle diagnostics, is a critical vulnerability as attackers can easily collect diagnostic data, gaining access to the in-vehicle network and deploying malicious programs.
• *USB and Charging Ports:* These ports are susceptible to severe security threats, such as the installation of malicious codes and reprogramming of the controller processor, enabling hackers to control critical vehicle systems like braking and engine control.
• *TPMS, LiDAR, and Keyless Entry Ports:* These systems can be exploited for eavesdropping attacks, signal jamming, and interception of key signals, leaving the vehicle network open to unauthorized access and manipulation.
• *Bus Network Ports:* The lack of communication protection in CAN protocols allows attackers to send fake frames to each node, leading to unintended vehicle behavior.
• *Vehicular Communication Ports:* Enabled with technologies like Bluetooth, Wi-Fi, DSRC, and cellular networks, these ports are vulnerable to various attacks, including jamming and eavesdropping, potentially allowing attackers to gain full access to vehicles.

IV. ATTACKS ON IN-VEHICLE NETWORK SYSTEM

A. Detailed Discussion of Notable Attacks with Examples

1. Lock Picking Attack on Keyless Entry Systems

The lock picking attack exploits the keyless entry system's vulnerability, typically used for car doors and garage openers. This technique has been demonstrated on various car brands and garage door openers, with devices like **Rolljam** and **OpenSesame** specifically designed for these types of attacks. Despite their widespread availability and potential misuse, sales of these devices have continued unabated, prompting concerns and calls for enhanced security measures from manufacturers.
There are two main algorithms used to send opening signals to cars:

- **Single code:** The key fob always sends the same code to the car that accepts it and opens. This is an old implementation used by cars manufactured until ~2002. This legacy implementation lacks basic security since whoever intercepts the signal is able to use it to open the car (known as a replay attack). Surprisingly, still some modern cars are implementing such algorithms.

- **Rolling code:** The key fob uses an array of codes, each of which is only usable once. This is much safer implementation that protects against replay attacks and is mostly used in modern cars.

Types of Remote Keyless Entry:

Keyless entry systems can be categorised into three broad types, as seen below.

1) **One-way RKE:** It requires a manual button press to perform an action. The vehicle receives the signal and confirms that it is a valid code, then performs the required action. With a rolling code system, a cryptographically secure pseudorandom number generator (PRNG), installed in the vehicle and the key fob, is

used to periodically change the required code after a keypress, usually with a buffer to account for accidental out-of-range button presses. One-way RKEs are the simplest and most common form of keyless entry.

2) **Two-way RKE:** It require a 'response' from the key fob given a certain 'challenge' from the vehicle.

3) **Passive RKE:** It automatically unlock within a certain radius or upon the user touching a door handle; they are usually paired with a push-button ignition switch.

Tools Used:

In order to detect Radio frequencies from a computer, there's the need to use radio peripherals capable of converting radio signals from analog to digital.

- **HackRF:** It is a wide band software defined radio half-duplex transceiver created and manufactured by Great Scott Gadgets. It is able to send and receive signals. The idea is to have a device that is able to receive and transmit different radio protocols just by configuring its software. The HackRF One's popularity stems from several of its benefits, such as: Its ability to receive and transmit radio in the 1Mhz to 6GHz range. One of the best devices for RFHacking so far. It has Wide range of frequencies handled, versatile, easy to use and have quite a lot of documentation online.

- **Yard Stick One:** The YARD Stick One (Yet Another Radio Dongle) is a palm-sized, low-speed USB wireless transceiver (similar to a Software Defined Radio or SDR) from Great Scott Gadgets that can transmit or receive digital wireless signals at frequencies below 1GHz. It is mostly used for jamming.
- **RTL SDR:** It is a low-cost USB device that can be used as a computer-based radio for receiving live radio signals. Depending on the RTL-SDR it could receive frequencies from 500 kHz up to 1.75 GHz. It gets overheated very often until there's the need to unplug it and wait for it to cool down. It can be used to track signals.

Initial Reconnaissance:
The first step in reverse engineering the key fob was to determine its operating frequency. The key fob case was inspected, however there were no visible frequencies or radio IDs listed that offered clues to the operating frequency. Hence, the RTL-SDR was utilised. Attacker can tune to multiple frequencies and try to press the key fob repeatedly until a signal became visible on the fast Fourier transform (FFT) plot, which displays a live view of the RF spectrum. The horizontal axis represents frequency in megahertz and the vertical axis the amplitude in decibels relative to full scale (dBFS).

Identifying Modulation:
Attacker will find one or two peaks. From that attacker can reduce the frequency. However, the plot does not provide information regarding how the data is actually encoded.

Rolling Code Bypass Theory:

The key of such attack is Jamming, sending a strong signal that blocks the car receiver from detecting the message sent from the key fob. Since the frequency of a signal depends on different factors, some of which are casual - like the temperature - the receiver has a range of accepted frequencies, also called bandwidth. The Jamming signal must be sent within the car's receiving window (or bandwidth), at a slightly moved frequency from the one used from the key fob. The tricky part of the attack is that, while jamming, the attacker has to be able to detect and filter the signal sent by the owner of the car trying to open his vehicle. If the attacker is able to do so, the rest of the path is just a matter of implementation. Once the first code has been stored from the device used by the attacker (that usually is a computer with some radio dongles) and not received by the car because of the jamming, the owner of the car will think that the car did not receive the signal because of other reasons and will try to open it again. While clicking the button the second time, the device will store the second signal, stop jamming and send the first signal in a matter of milliseconds. The car will receive the 1st signal from the device and open and the owner will think that now everything worked properly, but the reality is that the attacker will still own the 2nd code able to open the car.

Implementation:

The key to the Rolljam attack and the most difficult part of its bypass is the ability to jam and record at the same time. The Yard Stick One is used to jam the signal and the HackRF to deal with the unlocking signals sent from the keyfobs. Attacker can use the python script to jam signal using Yard stick one. Once the script is run, the car won't be able to receive other signals and will keep being closed even if the owner clicks the opening button. Meanwhile when jamming, attacker need to use a tool able to both record and send signals from device like laptop. GNU Radio is the best tool to do so. Attacker will need two scripts. one to record and the other one to replay the signal recorded. The first script will work with the HackRF, recording a signal at a central frequency and with a given bandwidth. The bandwidth setting will allow the HackRF to cut out the jamming signal, cleaning it and recording only the part attacker need (the actual key fob opening signal). It will then send it to a live GUI that will show attacker the peaks in the frequency spectrum and save it to a file called "file_name". All of these must happen while the Python3 jamming script keeps running. Once the python script has been interrupted, the second script will grab the "file_name" file with the filtered signal in it, multiply it and send it to both a GUI and to the HackRF, which will transmit it to the car. The car will receive the signal and open.

2. TPMS Exploitation

The man-in-the-middle attack on a vehicle's tire pressure monitoring system using radio frequency equipment and programming tools. The attack involved intercepting the signal sent from the tire sensor to the vehicle's computer and manipulating it to generate a false signal, which could potentially lead to dangerous situations such as enabling the tires to inflate themselves to account for leaks, pressure loss, and temperature change.

Implementation:

- The hackers used radio frequency equipment and spectrum analysis software to intercept the TPMS signal.
- A software-defined radio called Hack RF to simulate a TPMS and an open-source toolkit called Good News Radio to interface with it.
- The captured signal was analysed using the program Audacity to determine how the data was encoded.
- Data in the TPMS system is encoded using on-off keying, which translates to binary zeros and ones.
- The binary signal contains a unique ID number and the tire pressure.
- By generating a false signal and changing the pressure to desired values, the TPMS system can be manipulated.
- Exploiting vulnerabilities in the TPMS system can have impacts beyond just creating an annoying notification light, as it can affect commercial tractor trailers equipped with TPMS systems.

Software/hardware requirements: **HackRF, Audacity**

3. CAN Bus Specific Attacks

CAN Sniffing: The objective of this section is to comprehensively understand the purpose and ethical considerations of CAN sniffing. For **hardware requirements**, a CAN-do board or equivalent, along with CAN interface hardware, is necessary. A computer equipped for data analysis is also crucial. On the **software side,** tools such as Wireshark with the CAN plugin, device drivers for the CAN interface hardware, and potentially custom analysis tools should be employed. Implementation involves connecting the CAN interface hardware to the target bus, configuring and initializing the CAN-do board, and utilizing Wireshark for capturing and analysing CAN bus traffic. Custom analysis tools can be developed for more specific insights.

CAN Fuzzing: This section aims to explore the impact of random CAN data frames on vehicle behaviour.

Hardware requirements encompass CAN interface hardware and a computer for the fuzzing implementation. Fuzzing tools like Socket CAN and necessary analysis utilities make up the

Software requirements. Implementation steps involve connecting the CAN interface hardware, configuring or developing fuzzing tools for random frame generation, executing the fuzzing tools, and observing changes in vehicle behaviour. The results should be thoroughly analysed and documented.

Frame Falsifying and Injection Attacks : The goal of this section is to investigate the effects of manipulating CAN message payloads.

Hardware requirements include CAN interface hardware, hardware for message manipulation, and a computer for execution.

Software requirements involve custom payload modification tools, CAN interface software, and analysis tools. Implementation steps include connecting the CAN interface hardware, developing tools for modifying CAN message payloads, executing payload modification attacks, and monitoring the bus behaviour. The observed effects on vehicle behaviour should be meticulously documented.

DoS Attacks : This section focuses on understanding the impact of Denial of Service (DoS) attacks on the CAN bus.

Hardware requirements encompass devices for sending high priority CAN frames, CAN interface hardware, and a computer for execution.

Software requirements include custom DoS tools, CAN interface software with DoS capabilities, and analysis tools. Implementation steps comprise connecting the CAN interface hardware, developing custom DoS tools for generating high-priority frames, executing DoS attacks, and monitoring their impact on the bus. The disruptions in sensor parts and overall vehicle functionality should be documented.

Comparing different types of attacks: In the realm of Controller Area Network (CAN) bus security, various attacks exploit vulnerabilities within the system. Sniffing Attacks leverage the absence of segmentation and data encryption, allowing attackers to gain unauthorized access to confidential data by eavesdropping on CAN bus messages, with the potential to unlock similar attacks on other vehicles. Fuzzing Attacks, targeting the absence of segmentation and authentication, aim to discover unknown vulnerabilities in the CAN system, potentially leading to zero-day attacks that could be replicated across multiple vehicles. Frame Falsifying attacks, exploiting the absence of segmentation and authentication, involve attackers manipulating CAN bus messages to disrupt normal operation or disseminate false information, although such attacks do not unlock similar exploits on other vehicles. Injection Attacks, exploiting the absence of segmentation, authentication, and data encryption, involve injecting malicious code into an Electronic Control Unit (ECU), enabling attackers to sniff the network and inject further attacks, potentially unlocking similar exploits on other vehicles. Lastly, DoS Attacks exploit the absence of segmentation, causing a denial of service by flooding the CAN bus with high priority messages, although they do not unlock similar attacks on other vehicles.

Feasibility and Conditions:

- **Sniffing Attacks:** Possible in situations where an attacker can gain physical or remote access to the CAN bus.
- **Fuzzing Attacks:** Feasible in scenarios where attackers can inject specially crafted messages to identify vulnerabilities.

• **Frame Falsifying Attacks:** Possible when an attacker gains access to the CAN bus and can manipulate messages.

• **Injection Attacks:** Successful when attackers can inject malicious code into an ECU, either physically or remotely.

• **DoS Attacks:** Can be carried out if an attacker can flood the CAN bus with high priority messages. The success of these attacks often depends on the level of security measures implemented in the vehicle's CAN bus architecture. Implementing security measures, such as encryption, authentication, and intrusion detection systems, can help mitigate the risks associated with these attacks.

4. ECU Impersonation

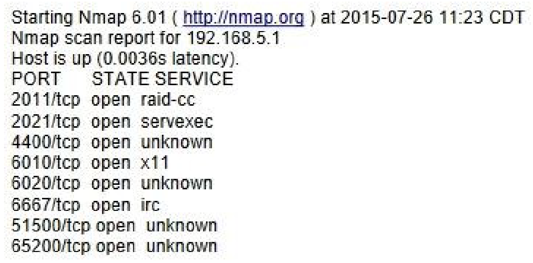
Software Requirements:
SPICE Simulation Tool:
● The authors used SPICE (Simulation Program with Integrated Circuit Emphasis) for simulating the attack and modelling the behaviour of the CAN transceivers.
● SPICE is a general-purpose analogue electronic circuit simulator.
Optimizer: An optimizer tool was used to converge on the attack parameters.
● Modelling Tools for CAN Transceivers: Tools for creating SPICE models of the CAN transceivers based on the schematics provided in the paper.

Hardware Requirements:
The attack involves manipulating the CAN bus, and the paper uses CAN transceivers. The specific model mentioned is the NXP TJA1050 transceiver. Other transceivers from different vendors are also mentioned as part of ongoing work. GPIO peripherals in microcontrollers used in ECUs. specific microcontrollers are used such as Texas Instruments TMS470M and TMS570. Voltage Measurement Tools (e.g., Oscilloscope): oscilloscope was used to capture voltage waveforms during the simulation Implementation Setup. The attack involves a physical-layer data manipulation technique targeting the Controller Area Network (CAN) bus, a common in-vehicle communication protocol. The attack aims to induce arbitrary bit-flips in transmitted messages by colluding multiple compromised Electronic Control Units (ECUs). The attackers exploit transient voltages in the CAN bus, heightened by the parasitic reactance of the bus and non-ideal properties of line drivers. The theoretical basis of the attack involves manipulating the state of transistors in the CAN transceiver, causing voltage discrepancies. To validate the theory, the authors conducted experiments using SPICE simulation tools, modelling the behaviour of CAN transceivers and optimizing attack parameters. The simulation results indicated that with more than eight compromised ECUs, the attackers could induce sufficient voltage drops to flip dominant bits to recessive ones. For the implementation setup, SPICE (Simulation Program with Integrated Circuit emphasis) was employed for simulating the attack. The CAN transceivers were modelled based on the schematic of the NXP TJA1050 transceiver. An optimizer tool was used to converge on attack parameters, and specialized modelling tools were employed to create SPICE models of CAN transceivers. The simulation involved the legitimate ECU transmitting at 100 kbps and a rogue ECU switching at a higher data rate of 400 kbps. The number of compromised ECUs, their data rate, and synchronization were critical parameters. The SPICE circuit models included practical p-channel and n-channel MOSFETs, representing the transistors in the CAN transceiver. The simulation setup aimed to observe the effects of the transmission line and parasitic capacitance on the induced transients in the CAN bus. Ongoing work involves hardware realization of the attack and further optimization of attack parameters. The researchers are exploring ways to reduce the number of compromised ECUs needed for a successful bit-flip attack. Additionally, they plan to study the impact of variable switching speed, delay among rogue ECUs, and varying transmission line lengths. The research is supported by the National Science Foundation (NSF) grants CNS-1801611 and CNS-1658225. The ultimate goal is to demonstrate that the proposed attack can manipulate data on the CAN bus, potentially leading to severe consequences in a vehicle's operation.

5. Jeep Hack via Cellular Network

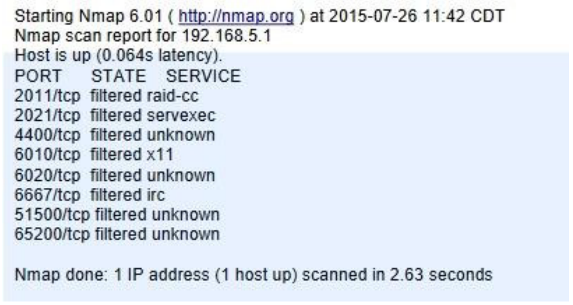
1. Identify the target vehicle's IP address. This can be done by scanning IP ranges where vehicles are known to reside until you find one with a corresponding VIN or GPS.

   Process:
   A fix was made by Chrysler for this issue and can be found in version 15.26.1. they did not extensively study this patch although the net result is that the vehicle now no longer accepts incoming TCP/IP packets. This is the result of a nmap scan before the patch (version 14.25.5)



```
Starting Nmap 6.01 ( http://nmap.org ) at 2015-07-26 11:23 CDT
Nmap scan report for 192.168.5.1
Host is up (0.0036s latency).
PORT      STATE SERVICE
2011/tcp  open  raid-cc
2021/tcp  open  servexec
4400/tcp  open  unknown
6010/tcp  open  x11
6020/tcp  open  unknown
6667/tcp  open  irc
51500/tcp open  unknown
65200/tcp open  unknown
```

   Additionally, the Sprint network was reconfigured to block (at least) port 6667 traffic even within the same cellular tower. Therefore, the only way to attack a vulnerable, unpatched, vehicle is to either do it over Wi-Fi, if available, or over a femtocell connection. Both require close range to the vehicle.



```
Starting Nmap 6.01 ( http://nmap.org ) at 2015-07-26 11:42 CDT
Nmap scan report for 192.168.5.1
Host is up (0.064s latency).
PORT      STATE    SERVICE
2011/tcp  filtered raid-cc
2021/tcp  filtered servexec
4400/tcp  filtered unknown
6010/tcp  filtered x11
6020/tcp  filtered unknown
6667/tcp  filtered irc
51500/tcp filtered unknown
65200/tcp filtered unknown

Nmap done: 1 IP address (1 host up) scanned in 2.63 seconds
```

This is the scan after the patch has been installed:

2. Exploit the OMAP chip of the head unit using the execute method of the appropriate D-Bus service. The easiest thing to do is to upload an SSH public key, configuration file, and then start the SSH service. At this point, you can SSH to the vehicle and run commands from the remote terminal.

   Resources: Used a Sprint Airave 2.0 device to access the Jeep's network via Telnet, and a laptop running Ubuntu Linux to run their custom software tools.

Fig. Sprint Airave 2.0

3. Once you have access to the vehicle's network, you can send CAN messages to the vehicle's electronic control units (ECUs) to control various functions, such as the brakes, steering, and engine. The researchers used a custom-built CAN framework to send messages to the Jeep's CAN bus.

   The CAN Badger framework consists of two main components: a hardware interface and a software interface. These hardware interface is a small device that connects to the OBD-II port of a vehicle and provides a way to communicate with the CAN bus. The software interface is a set of Python libraries that allow researchers to interact with the CAN bus using the hardware interface. The CAN Badger framework provides several features that make it useful for security researchers. For example, it allows researchers to send and receive messages on the CAN bus, monitor the bus for specific messages, and inject custom messages into the bus. It also provides a way to log all messages on the bus for later analysis. The framework is designed to be flexible and extensible, allowing researchers to customize it for their specific needs. It is also designed to be easy to use, with a simple command-line interface that allows researchers to quickly get up and running.

4. The researchers developed a library of tools that would aid in continuing automotive research and reduce the barrier of entry to new researchers into the field. These tools were released to the public and have been used by many researchers, including the National Highway Traffic Safety Administration.

Procedure: identifying the target vehicle's IP address and exploiting the OMAP chip of the head unit. The researchers assumed that a remote compromise was possible, due to the material released by the academic researchers in previous years. Therefore, they assumed that they could inject CAN messages onto the bus in a reliable fashion.

6. Replay Attack

The number of ECUs within a vehicle was about 70 ECUs. This number has doubled nowadays. The CAN protocol, along with other communication protocols, encounters mainly three security issues such as backward compatibility, replay attacks, and message authentication. In LCAP - Lightweight Controller Area Protocol has been discussed. It differs from CAN as the main purpose of the LCAP is to keep messages authenticity and integrity for all messages traveling over the CAN bus. However, it is highlighted that the LCAP has a vulnerability related to replay attacks, specifically in channel requests. The attackers can exploit this by periodically replaying channel request messages, leading to a Denial of Service (DoS) attack, blocking normal data messages. A malicious node can record messages on the bus, identify the channel request message through trial and error, and replay it periodically. Controlling messages take priority over data messages so periodically sending the control messages for a replay attack will block the network pipeline to keep sending controlling messages, and data messages don't get passed through. The attack is quite severe as it affects controls in an automotive system. A delay or failure of those could cause major issues in a vehicle. The detection of this attack is also quite difficult but can be prevented by adding a nonce to the messages. A nonce or a timestamp will prevent a replay attack.

The simulation of this attack is conducted using the Vector CANoe simulation, a well-known automotive network simulator. The proposed solution includes using nonce in messages to prevent replay attacks and improve system security. Another form of replay attack was implemented in. One is to replay the entire message and the other one is to replay only part of the frame excluding the identifier. In the first attack, the entire message is stored by a node and then transmitted again when the CAN bus is idle. In this attack all nodes will receive faulty data. The only way to detect this attack is when ID is compared and the original source node detects receiving a message with the same ID as theirs. They can create an error frame when a replay attack is detected. In the second method, partial frames are replayed and ID frames are replaced. This attack is harder to detect as the source uses its own ID. Since it's a replay attack the way to prevent this is also through adding timestamps. CAN does not offer security services such as encryption or data frame authentication. This means that eavesdropping and replay attacks in CAN are possible.

7. Attack through a Malicious App

Other than replay attacks, a common attack researched was through a diagnostic malicious App. When a malicious app is used for attack, it enables the attacker to perform a long-range wireless attack where the attacker or attacking equipment need not be in the close range of the vehicle to perform the attack. The user will install a self-diagnostic app to monitor status information after installing an OBD2 scan tool on the vehicle and then pairing it with his/her smartphone by Bluetooth. When the driver installs on his/her smartphone the malicious self-diagnostic app distributed by an attacker, the attacker can launch the actual attack. The attacker can obtain status information of the vehicle from the malicious self-diagnostic app and use it to inject malicious data into the in-vehicle network. As per this research vulnerabilities of in-vehicle CAN are weak access control, no encryption and no authentication. After the malicious app is installed on the victim's device, it transmits data frames of the in-vehicle CAN to the attacker's server using the smartphone's mobile communication network. This would force control of an ECU to the in-vehicle CAN via the malicious app. The target vehicle would have a physical malfunction caused by the abnormal control data that was transmitted from the attacker's server. Another prominent attack with malicious apps. Apps for vehicles can be easily forged/repackaged and redistributed. Using a vehicle diagnostic device, The attacker can get a CAN data frame that can drive a specific ECU mounted in the target vehicle. Furthermore, the attacker can download a vehicle application that is distributed/sold in the app market and repackage it in a desired form. The diagnostic app will provide many services to the victim while they are driving and can use this service to start an attack. An attacker identifies vulnerabilities in a vehicle's wireless communication module (ELM327), creates a malicious code based on the analysis, and inserts it into a legitimate vehicle app. The tampered app is then distributed through app markets. A victim unknowingly downloads and installs the malicious app, which is designed to exploit the ELM327 module in their vehicle. When the victim runs the app and starts driving, the malicious code executes harmful actions. As countermeasures to such attacks, app obfuscation and a whitelist-based firewall were proposed. Car manufacturers must install the security feature in the OBD-II interface, because we cannot depend on an external device to counter threats to the vehicle itself.

8. *Bus off Attack*

Apart from attacks such as DOS or Replay attacks, there are other attacks that CAN is vulnerable to. One of those attacks is a bus-off attack, which can cause a victim ECU to disconnect itself from the CAN bus and, subsequently, an attacker can masquerade as that ECU. A limitation of the bus-off attack is that it requires the attacker to achieve tight synchronization between the transmission of the victim and the attacker's injected message. The bus-off state is a state of error in a CAN controller in which the node is disconnected from the bus communications which means it can neither transmit nor acknowledge frames. A node that is in the bus-off state can only rejoin the network after observing 128 occurrences of the bus-free signal of 11 consecutive recessive bits. When a node recovers from bus-off, it resets its counter and starts from the initial error-active state. The attack's success depended on meeting a few conditions: matching message IDs with the victim and synchronization with the victim message. The attack is achieved by identifying a unique message that precedes the victim. However, if no unique preceded ID exists, fabricating and injecting unique preceded IDs to interfere with the victim's transmission was proposed. It can still be detected by an IDS. Alternatively, the schedule-based bus-off attack doesn't rely on unique preceded IDs for synchronization. It targets instances of the victim message facing blocking or interference, regardless of the preceding message's ID. The attacker gains knowledge about message sets on the CAN bus before launching the attack, allowing them to identify opportunities for successful synchronization.

9. Manipulation via OBD-II Port

1. Install ScanTool Software:
   - open a terminal in Kali Linux.
   - Type the following command to install the ScanTool software: `sudo apt-get install scantool`.
   - Press Enter to execute the command.
2. Run ScanTool Application:
   - After installation, run the ScanTool application by typing `scantool` in the terminal.
   - Press Enter to launch the application.
3. Connect OBDLink Cable:
   - Plug the OBD-II cable into your laptop or computer using the USB adapter end.
   - Connect the OBD-II connector to your car's OBD port located under the steering wheel on the driver's side near the floorboard.
   - Turn on the automobile's ignition.
4. For Virtual Machines (VM):
   - If using a virtual machine, connect the USB device to your VM in VirtualBox.
   - In VirtualBox, go to the virtual machine's menu and select Devices > USB > ScanTool OBDLink (or similar).
   - This step ensures the virtual machine can access the OBDLink hardware.
5. Using ScanTool Software:
   - The ScanTool software will now connect to the automobile's CAN network.
   - Perform tasks such as reading diagnostic codes, resetting them, and reading sensor data.
   - The GUI interface of the ScanTool software provides an easy-to-use platform for these tasks.
6. For Additional Capabilities:
   - Consider upgrading to a more capable CAN-to-USB converter like CANable, CANtact, or Korlan USB2CAN .
   - With these devices, you can read raw CAN packets and attempt advanced operations, such as replay attacks.
7. Advanced Operations
   - If attempting replay attacks, use tools like canplayer to capture, manipulate, and reinject CAN packets.
   - Exercise caution as the effects of packet injection can lead to unpredictable behavior.
   - Only perform such actions in a controlled and safe environment, ideally on a vehicle that can be taken to a professional mechanic if issues arise.

Safety Reminder:

- Always be aware that manipulating vehicle systems can have unpredictable consequences.
- Perform advanced operations only on a vehicle you can afford to take to a real mechanic if needed.

By following these steps, you can effectively connect the OBDLink SX to a Kali VM and use it for various car hacking and diagnostic tasks.

Software/hardware requirements: **ODB-II to USB cable**

Is the attack possible now and in what situation we can successfully perform the attack : yes

10. SAE J1939 Standard Attacks
Implementation:

The ECM is filled with a lot of high volume of requests for the component identifier. Our approach involved transmitting four distinct varieties of messages on the CAN network, each with varying intervals of transmission: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1 millisecond. The initial category of message was assigned a CAN ID of 0000000016. Consequently, this particular CAN frame, being of utmost priority, possesses the capability to inundate the entire bus if disseminated at elevated rates. The subsequent category of message, however, featured the lowest SAE J1939 priority (7), accompanied by a PGN (Parameter Group Number) of 0, a DA (destination address) of 0, and an SA (source address) of 0. Consequently, the corresponding CAN ID was identified as 1C0000016. The third category of message, while maintaining the same SAE J1939 priority (7), constituted a request (PGN EA0016) from the engine controller directed towards itself. Thus, it was assigned a CAN ID of 1CEA000016. It is important to note that we ensured the engine controller responded to this particular request. Furthermore, this request pertained to the component identifier that was present in all the ECMs utilized within the experiment.

We shall hereby refer to this request as the valid quest. Lastly, the fourth category of message consisted of a request for a PGN FFFF16, which was not present within any engine controller. Consequently, we shall henceforth refer to this request as the invalid request. Overall, we noticed that, in general, a request overload attack launched at 0.3 milliseconds or below, had an effect on the target ECM, even when launched with the lowest priority.

Software/hardware requirements :

Kenworth T270 research truck, CAN backbone operating at a baud rate of 250 kbps, custom-built Beagle Bone Black1 node controllers that were equipped with CAN controllers and provided access to the CAN network through the CAN-utils2 software installed on a 32-bit Linux distribution(Software)f, ECM Is the attack possible now and in what situation we can successfully perform the attack. Yes, the attack is possible in current scenario. The situation in which this attack can be explained is by just sending out multiple requests as specified by the defence system in a millisecond.

The seemingly obvious countermeasure against this assault is rather uncomplicated: refraining from undertaking a quantity of inquiries surpassing a specific threshold within a time span of a thousandth of a second. However, it must be acknowledged that this resolution necessitates a modification in the Electronic Control Module's software.

B. Computational and Communication Resources Needed by Attackers

The complexity of attacks on the Controller Area Network (CAN) and its variants in modern vehicles requires diverse computational and communication resources. These resources range from sophisticated software for data analysis to specialized hardware for network interfacing. This review consolidates insights from various academic sources to understand the resources attackers utilize in executing these cyberattacks.

Data Collection and Analysis

• CAN Bus Data Capture: Attackers capture data from a vehicle's CAN network, often via the On-Board Diagnostics (OBD) port, using tools like Vehicle Spy simulation test software. This process demands precision to avoid packet loss and maintain data integrity. Large datasets are created for deep analysis, requiring significant storage and processing capabilities.

• Anomaly Detection Algorithms: Post data collection, sophisticated algorithms analyze anomalies in the CAN ID's message stream, a process that requires substantial computational power. These algorithms discern between normal vehicle operation changes and deliberate tampering, making them crucial in attack execution.

Exploiting Connectivity Interfaces

• Hacking Connectivity Systems: Modern vehicles are equipped with a plethora of connectivity interfaces such as Bluetooth, Wi-Fi, and cellular networks. Exploiting these systems requires specialized software capable of intercepting and manipulating wireless communications. Such tools need to be adaptable to different technologies and robust enough to breach security protocols.

• Diagnostic Port Vulnerabilities: The OBD-2 port, used for diagnostics, is a frequent target for cyberattacks. Attackers use specialized hardware to interact with this port, leading to potentially severe safety compromises. The hardware employed here is often compact yet capable of powerful interfacing with the vehicle's internal systems.

• Targeting Vehicle Sensors and Actuators: Physical availability attacks, like signal jamming, are directed at crucial vehicle sensors and actuators. This method requires hardware that can effectively disrupt or manipulate sensor data, such as devices capable of eavesdropping on TPMS signals.

• Signal Jamming and Relay Attacks: Systems like LiDAR and keyless entry are susceptible to signal jamming and relay attacks. These attacks necessitate sophisticated hardware and software to capture and relay signals, enabling unauthorized access to vehicles.

C. Dependencies on Physical Access or Specific Conditionsfor Successful Attacks

1. Physical Access to the Vehicle: Many attacks on the CAN network require physical access to the vehicle. This can range from a mechanic to a family member inserting a malicious component into the car's internal network, usually through the OBD-II port. The impact of these attacks can be profound, allowing control over critical vehicle functions.

2. Direct Access through OBD Port and Malicious Nodes: Direct access attacks are often executed through the OBD port, which provides comprehensive access to the vehicle's network. Such attacks can control critical vehicle systems, including braking and engine functions. Malicious nodes, once connected, can disrupt the network by intercepting or sending deceptive messages.

3. Denial-of-Service (DoS) Attacks: DoS attacks, particularly on commercial vehicles, involve overloading the network with excessive request messages or manipulating network connections. These attacks require an understanding of specific network protocols and the ability to generate substantial traffic to overload the system.

4. Indirect Physical Access: Indirect physical access attacks involve inserting a physical object into the car, without direct network access. Checkoway et al. developed an indirect access attack model by hacking the car service's IT system and accessing the CAN via a computer. Another indirect attack involved using multimedia devices like CDs, USBs, or MP3 players. Although these attacks may not directly breach the CAN, they can disrupt the driver by displaying warnings and playing alarm signals, emphasizing the importance of securing both direct and indirect access points to the CAN bus network.

V. CONCLUSION

Vulnerability to Evasion Techniques: A significant issue with existing malware detection methods is their susceptibility to evasion. Modern malware uses various obfuscation techniques, like throttling execution across multiple Electronic Control Units (ECUs) or leveraging multi-core processors to spread activities and avoid detection. These methods can make malware difficult to analyze and, consequently, evade existing detection systems. This vulnerability is particularly concerning in the context of intelligent vehicles, where passenger safety is paramount.

Challenges in Detecting New Malware: The current approaches struggle to identify new, sophisticated malware forms, posing a risk to driver and passenger safety in intelligent vehicles. Most traditional detection methods also face challenges in staying updated over a vehicle's lifespan, making them impractical due to the excessive costs and logistical complexities involved in updating millions of vehicles regularly. In contrast, cloud-based approaches, which update configurations regularly in the cloud, are seen as more viable, especially with the emergence of high-speed 5G technology, offering a more adaptable and up-to-date defense against malware.

The survey highlights the vulnerabilities in the Controller Area Network (CAN) protocol, emphasizing the need for enhanced security measures to address confidentiality, integrity, and availability concerns. As the automotive industry continues to evolve, understanding these cybersecurity challenges is crucial for developing robust solutions and exploring alternative technologies that offer improved resilience against cyber threats in vehicle communication networks.

REFERENCES

[1] Murvay, P., & Groza, B. (2020). Efficient Physical layer key agreement for FlexRay networks. *IEEE Transactions on Vehicular Technology*, *69*(9), 9767–9780. https://doi.org/10.1109/tvt.2020.3002616

[2] Adly, S., Hamed, A. M., Hammad, S., & Maged, S. A. (2023). *Prevention of Controller Area Network (CAN) attacks on electric autonomous vehicles. Applied Sciences*, 13(16), 9374. https://doi.org/10.3390/app13169374

[3] Tanksale, V. (2020). Controller Area Network Security Requirements. *Controller Area Network Security Requirements*. https://doi.org/10.1109/csci51800.2020.00034

[4] https://labs.jumpsec.com/car-hacking-manual-bypass-of-modern-rolling-code-implementations/

[5] Lin, C., & Sangiovanni-Vincentelli, A. (2012). Cyber-Security for the Controller Area Network (CAN) Communication Protocol. *Cyber-Security for the Controller Area Network CAN Communication Protocol*. https://doi.org/10.1109/cybersecurity.2012.7

[6] Taylor, A., Leblanc, S., & Japkowicz, N. (2016). Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks. *Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks*. https://doi.org/10.1109/dsaa.2016.20

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

[8] Zhang, H., Pan, Y., Lu, Z., Wang, J., & Liu, Z. (2021). A Cyber Security Evaluation Framework for In-Vehicle Electrical Control Units. *IEEE Access*, *9*, 149690–149706. https://doi.org/10.1109/access.2021.3124565

[9] Scalas, M., & Giacinto, G. (2019). Automotive Cybersecurity: Foundations for Next-Generation Vehicles. *Automotive Cybersecurity Foundations for Next-Generation Vehicles.* https://doi.org/10.1109/ictcs.2019.8923077 [10] A. Sui and G. Muehl "Security for Autonomous Vehicle Networks", 13 Novembre,2020

[11] Bozdal, M., Samie, M., Aslam, S., & Jennions, I. K. (2020b). Evaluation of CAN bus security challenges. *Sensors*, *20*(8), 2364. https://doi.org/10.3390/s20082364

[12] Fakhfakh, F., Tounsi, M., & Mosbah, M. (2021). Cybersecurity attacks on CAN bus-based vehicles: a review and

open challenges. *Library Hi Tech*, *40*(5), 1179–1203. https://doi.org/10.1108/lht-01-2021-0013
[13] Brewer, J. N., & Dimitoglou, G. (2019). Evaluation of Attack Vectors and Risks in Automobiles and Road Infrastructure. *Evaluation of Attack Vectors and Risks in Automobiles and Road Infrastructure*. https://doi.org/10.1109/csci49370.2019.00021
[14] Takahashi, J., Aragane, Y., Miyazawa, T., Fuji, H., Yamashita, H., Hayakawa, K., Ukai, S., & Hayakawa, H. (2017). Automotive attacks and countermeasures on LIN-Bus. *Journal of Information Processing*, *25*(0), 220–228. https://doi.org/10.2197/ipsjjip.25.220
[15] Sommer, F., Dürrwang, J., & Kriesten, R. (2019). Survey and classification of automotive security attacks. *Information*, *10*(4), 148. https://doi.org/10.3390/info10040148
[16] Zhang, H., Pan, Y., Lu, Z., Wang, J., & Liu, Z. (2021b). A Cyber Security Evaluation Framework for In-Vehicle Electrical Control Units. *IEEE Access*, *9*, 149690–149706. https://doi.org/10.1109/access.2021.3124565
[17] Bozdal, M., Samie, M., & Jennions, I. K. (2018). A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions. *A Survey on CAN Bus Protocol- Attacks, Challenges, and Potential Solutions*. https://doi.org/10.1109/iccecome.2018.8658720
[18] Aliwa, E., Rana, O., Perera, C., & Burnap, P. (2021). Cyberattacks and countermeasures for In-Vehicle networks. *ACMComputing Surveys*, *54*(1), 1–37. https://doi.org/10.1145/3431233
[19] Elkhail, A. A., Refat, R. U. D., Habre, R., Hafeez, A., Bacha, A., & Malik, H. (2021). Vehicle Security: A survey of security issues and vulnerabilities, malware attacks and defenses. *IEEE Access*, *9*, 162401–162437. https://doi.org/10.1109/access.2021.3130495
[20] Hubaux, J., Capkun, S., & Luo, J. (2004b). The security and privacy of smart vehicles. *IEEE Security & Privacy*, *2*(3), 49–55. https://doi.org/10.1109/msp.2004.26
[21] Rathore, R. S., Hewage, C., Kaiwartya, O., & Lloret, J. (2022d). In-Vehicle Communication Cyber Security: challenges and solutions. *Sensors*, *22*(17), 6679. https://doi.org/10.3390/s22176679
[22] Hubaux, J., Capkun, S., & Luo, J. (2004c). The security and privacy of smart vehicles. *IEEE Security & Privacy*, *2*(3), 49–55. https://doi.org/10.1109/msp.2004.26
[23] 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
[24] Choi, W., Joo, K., Jo, H. J., Park, M. C., & Lee, D. H. (2018). VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Transactions on Information Forensics and Security*, *13*(8), 2114–2129. https://doi.org/10.1109/tifs.2018.2812149
[25] Bozdal, M., Samie, M., Aslam, S., & Jennions, I. K. (2020). Evaluation of CAN bus security challenges. *Sensors*, *20*(8), 2364. https://doi.org/10.3390/s20082364
[26] Xu, R., Joshi, J., & Li, C. (2019b). CryptoNN: Training Neural Networks over Encrypted Data. *CryptoNN Training Neural Networks Over Encrypted Data"*. https://doi.org/10.1109/icdcs.2019.00121
[28] "ProceedingsofIEEE19"
[29] Wang, Q., Yan, Z., Lu, X., Wang, Z., Qin, Z., & Ren, K. (2016b). Real-time and Spatio-temporal Crowd-sourced Social Network Data Publishing with Differential Privacy. *IEEE Transactions on Dependable and Secure Computing*, 1. https://doi.org/10.1109/tdsc.2016.2599873
[30] Amirtahmasebi, K., & Jalalinia, S. R. (2010). *Vehicular Networks – security, vulnerabilities, and countermeasures*. https://odr.chalmers.se/bitstream/20.500.12380/123778/1/1237 78.pdf
[31] Engoulou, R. G., Bellaïche, M., Pierre, S., & Quintero, A. (2014). VANET security surveys. *Computer Communications*, *44*, 1–13. https://doi.org/10.1016/j.comcom.2014.02.020
[32] Zhang, H., Shi, Y., Wang, J., & Chen, H. (2018). A new Delay-Compensation Scheme for networked control systems in controller area networks. *IEEE Transactions on Industrial Electronics*, *65*(9), 7239–7247. https://doi.org/10.1109/tie.2018.2795574
[33] Park, I., & Sunwoo, M. (2011). FlexRay Network Parameter Optimization Method for Automotive Applications. *IEEE Transactions on Industrial Electronics*, *58*(4), 1449–1459. https://doi.org/10.1109/tie.2010.2049713
[34] Kim, J. H., Seo, S., Hai, N. T., Cheon, B. M., Lee, Y. S., & Jeon, J. W. (2015). Gateway framework for In-Vehicle networks based on CAN, FlexRay, and Ethernet. *IEEE Transactions on Vehicular Technology*, *64*(10), 4472–4486. https://doi.org/10.1109/tvt.2014.2371470
[35] Kelkar, S., & Kamal, R. (2014). Implementation of data reduction technique in Adaptive Fault Diagnosis Algorithm for Controller Area Network. *Implementation of Data Reduction Technique in Adaptive Fault Diagnosis Algorithm for Controller Area Network*. https://doi.org/10.1109/cscita.2014.6839252
[36] Oberti, F., Sánchez, E., Savino, A., Parisi, F., Brero, M., & Di Carlo, S. (2022). LIN-MM: Multiplexed Message Authentication Code for Local Interconnect Network message authentication in road vehicles. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2206.02602
[37] Kim, B., & Park, K. (2009). Probabilistic delay model of dynamic message frame in flexray protocol. *IEEE Transactions on Consumer Electronics*, *55*(1), 77–82. https://doi.org/10.1109/tce.2009.4814417
[38] Sami, M., Ibarra, M., Esparza, A. C., Al-Jufout, S., Aliasgari, M., & Mozumdar, M. (2020). Rapid, Multi-vehicle and Feed-forward Neural Network based Intrusion Detection System for Controller Area Network Bus. *Rapid Multi-vehicle and Feed-forward Neural Network Based Intrusion Detection System for Controller Area Network Bus*. https://doi.org/10.1109/igessc50231.2020.9285088
[39] Yu, Z., Liu, Y., Xie, G., Li, R., Liu, S., & Yang, L. T. (2023). TCE-IDS: Time interval Conditional entropy- based Intrusion Detection System for automotive controller area networks. *IEEE Transactions on Industrial Informatics*, *19*(2), 1185–1195. https://doi.org/10.1109/tii.2022.3202539
[40] Sunny, J., Sankaran, S., & Saraswat, V. (2020). A Hybrid Approach for Fast Anomaly Detection in Controller Area Networks. *A Hybrid Approach for Fast Anomaly Detection in Controller Area Networks*. https://doi.org/10.1109/ants50601.2020.9342791
[41] Wu, Y., Fu, L., Xu, Y., Ma, F., & Lu, Y. (2018). Controller Area Network Modeling and Its Application in Cyber-Physical

Power System Co-Simulation. *Controller Area Network Modeling and Its Application in Cyber-Physical Power System Co-Simulation*. https://doi.org/10.23919/chicc.2018.8483930
[42] Kang, S., Seong, J., & Lee, M. (2018). Controller area network with flexible data rate transmitter design with low electromagnetic emission. *IEEE Transactions on Vehicular Technology*, *67*(8), 7290–7298. https://doi.org/10.1109/tvt.2018.2832659
[43] Othman, H., Aji, Y., Fakhreddin, F., & Al-Ali, A. R. (2006). Controller Area Networks: Evolution and Applications. *Controller Area Networks Evolution and Applications*. https://doi.org/10.1109/ictta.2006.1684909
[44] Ahn, B., Park, B., Ki, Y., Jeong, G., Ahn, H., & Kim, D. (2006). Development of a Controller Area Network Interface Unit and Its Application to a Fuel Cell Hybrid Electric Vehicle. *Development of a Controller Area Network Interface Unit and Its Application to a Fuel Cell Hybrid Electric Vehicle*. https://doi.org/10.1109/sice.2006.315774
[45] Novák, J. (2009). Flexible approach to the Controller Area Networks test and evaluation. *Flexible Approach to the Controller Area Networks Test and Evaluation*. https://doi.org/10.1109/idaacs.2009.5343029
[46] Prasad, B., Gao, R., Jing-Jou, T., & Jhen, H. C. (2022). LIN Bus Based Touchpad System for Smart Vehicle Cabin. *LIN Bus Based Touchpad System for Smart Vehicle Cabin*. https://doi.org/10.1109/icasi55125.2022.9774481
[47] Fax, J., & Murray, R. M. (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, *49*(9), 1465–1476. https://doi.org/10.1109/tac.2004.834433
[48] 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*, 1–14. https://doi.org/10.1109/tits.2014.2351612
[49] Lee, Y., Woo, S., Lee, J., Song, Y., Moon, H., & Lee, D. H. (2019). Enhanced Android App-Repackaging attack on In-Vehicle Network. *Wireless Communications and Mobile Computing*, *2019*, 1–13. https://doi.org/10.1155/2019/5650245
[50] Hounsinou, S., Stidd, M., Ezeobi, U., Olufowobi, H., Nasri, M., & Bloom, G. (2021). Vulnerability of Controller Area Network to Schedule-Based Attacks. *Vulnerability of Controller Area Network to Schedule-Based Attacks*. https://doi.org/10.1109/rtss52674.2021.00051
[51] Thirumavalavasethurayar, P., & Ravi, T. (2021). Implementation of Replay Attack in Controller Area Network Bus using Universal Verification Methodology. *Implementation of Replay Attack in Controller Area Network Bus Using Universal Verification Methodology*. https://doi.org/10.1109/icais50930.2021.9395871
[52] Noureldeen, P., Azer, M. A., Refaat, A., & Alam, M. F. (2017). Replay attack on lightweight CAN authentication protocol. *Replay Attack on Lightweight CAN Authentication Protocol*. https://doi.org/10.1109/icces.2017.8275376

GOOGLE DRIVE
https://drive.google.com/drive/folders/1nMuopkeCDkhJPtdQKh9G_Bnz07LM5w3b?usp=sharing

GIT HUB
https://github.com/Deep6776/INSE6120-Fall2023-Project-Group10

Roles and Responsibiliti

Team Member	Roles and Responsibility
Pratiksha (*40230412)*	-Overview of Automotive Networks -Brief discussion on the significance of security in automotive networks and Introduction to CAN -Importance and usage of Controller Area Network (CAN) and its alternatives (LIN, FlexRay)
Simaran (*40241517*) Snehpreet (*40254443*)	- Overview of CAN, LIN, and FlexRay networks - Comparison of functionalities and use cases - Security concerns and challenges in each network type
Deep (40231725), Devina (*40238009*), Saket (*40225128*)	-Specific attack vectors and methodologies (e.g., DoS attacks, spoofing, message manipulation) -Detailed discussion of notable attacks with examples -Summary of recent academic literature on attacks - Feasibility of attacks on these networks - Computational and communication resources needed by attackers - Dependencies on physical access or specific conditions for successful attacks
Rohan (*40255454*), Meet (*40239187*) Karanjot (*40220861*)	-Existing security measures and their effectiveness -Latest advancements in securing automotive networks - Recommendations and potential future directions for enhanced security
Pratiksha (*40230412*), Deep (40231725)	-Conclusion -Key takeaways from the surveyed literature