THESIS

SECURITY SHORTCOMINGS OF EMBEDDED NETWORK PROTOCOLS IN COMMERCIAL VEHICLES

Submitted by

Rik Chatterjee

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2024

Master's Committee:

Advisor: Jeremy Daily

Indrakshi Ray
Indrajit Ray

ABSTRACT

SECURITY SHORTCOMINGS OF EMBEDDED NETWORK PROTOCOLS IN COMMERCIAL VEHICLES

Modern commercial vehicles depend on embedded systems that communicate via standardized protocols, forming the foundation of their internal networks. The Controller Area Network (CAN) protocol is commonly employed for communication, with protocols such as SAE J1939 and Unified Diagnostic Services (UDS) playing critical roles in medium and heavy-duty vehicles. This thesis investigates multiple attack vectors that exploit vulnerabilities in both the SAE J1939 and UDS protocols, potentially compromising electronic control units (ECUs) in commercial vehicle networks.

The study presents five case scenarios related to the SAE J1939 standard, including two that validate previously proposed attack hypotheses using extensive testing setups. Additionally, three new attack vectors are explored through bench tests and in-vehicle trials. Simultaneously, the research highlights three vulnerabilities within the UDS protocol, specifically addressing weaknesses in the ISO 14229 and ISO 15765 specifications.

Testing was conducted on real-world systems, including bench setups with ECUs connected to a CAN bus and in-vehicle evaluations using a 2014 Kenworth T270 and a 2018 Freightliner Cascadia Truck Front Cab configured as a test bench. The results demonstrate how these protocol-based attacks can target and compromise specific ECUs, revealing significant security gaps in current vehicular communication systems. Engineers and developers working with SAE J1939 and UDS stacks must consider these vulnerabilities to enhance the resilience of communication subsystems in future designs.

ACKNOWLEDGEMENTS

Pursuing graduate studies at a U.S. university represents a dream come true for many, including myself. Without support, this dream might have remained just that—an unattainable aspiration. I am incredibly thankful to my advisor, Dr. Jeremy Daily, for his unwavering support and invaluable guidance. His mentorship continues to enrich not only my academic journey but also my personal growth. I am also grateful to my colleagues Subhojeet Mukherjee, Carson Green, and Jake Jepson, whose constant support and inspiration have been instrumental in my work. I would also like to extend my thanks to the other members of my thesis committee for their commitment to overseeing my work.

My heartfelt thanks go to all my educators, from kindergarten through to graduate studies, each of whom has played a pivotal role in shaping my educational path.

I extend my appreciation to the Defense Advanced Research Projects Agency (DARPA), and the Naval Information Warfare Center Pacific (NIWC Pacific) under Contract No. N66001-20-C-4021 for funding my research. Without their support, this work would not have been possible.

I fondly remember my late grandfathers and my late paternal grandmother, and I cherish my maternal grandmother who is still with us. I seek their blessings from both heaven and earth. Additionally, I remember my late uncle, Samir Chatterjee, a pivotal figure in my upbringing, with profound love and gratitude.

Immense gratitude is also due to my parents, Sarit and Sukla Chatterjee. My father, a professor, and my mother, an engineer, have provided unwavering love, support, and guidance, without which I would not have achieved what I have today.

I am forever grateful to my sister, Devalina Chatterjee, affectionately known as Owlet, for the joy, laughter, and love she brings into my life, proving to be the sweetest sister one could ever hope for. Finally and most importantly, my deepest love and thanks go to my wife, Chandrima Ghatak, without whose constant support through thick and thin and consistent pestering, I would not have been where I am today.

DEDICATION

I dedicate this thesis to all the dogs and cats in the world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Commercial Vehicles

Medium and heavy-duty (MHD) vehicles form an essential pillar of the U.S. critical infrastructure, playing a vital role in transporting goods and supporting various services, including emergency response. They form the most used form of logistics in the country with an average of 15,000,000 registered MHD vehicles [1]. In the past two decades, environmental concerns related to pollution caused by MHD vehicles have brought about strict emission standards and pushed them electrification. The evolution of MHD vehicles towards higher levels of electronification has led to most mechanical operations being controlled through low-power embedded computers, known as electronic control units (ECUs) which regulate most operations of a modern MHD vehicle. These ECUs, interconnected within the vehicle through a bus topology network, handle mission-critical information crucial for the vehicle's functionality and safety. They read vehicle parameters from an array of sensors and other ECUs within the network, using them to compute control signals that control actuators.

In MHD vehicles, the primary communication specifications within these networks are guided by a set of standards that enable inter-ECU communication. A vast majority of these communications are based on the Society of Automotive Engineers (SAE) J1939 standards [2]. The specifications of the SAE J1939 are structured in layers, akin to the Open System Interconnect (OSI) [3] standards prevalent in traditional IT networking. The foundational physical layers of the SAE J1939 standards employ the Controller Area Network (CAN) specifications [4] to facilitate the in-vehicle information exchange, a system widely used across automotive networking. As with all vehicles, MHD vehicles require maintenance and diagnostics for continued operations. The International Standards Organization (ISO) 14229 [5, 6] and ISO 15765 [7] protocols, define the standards for diagnostic and maintenance communications for ECUs and diagnostic testers. In

MHD vehicles, the diagnostics standards utilize parts the SAE J1939 protocol specifications for facilitating maintenance operations.

The use of common standards allows vehicle manufacturers to assemble different equipment from various original equipment manufacturers without worrying about interoperability. Data obtained from MHD vehicles are often used for logistics optimization, asset management, recording logging hours of service, improving fuel economy, and regulatory compliance through wireless telematics devices or more recently plug and play electronic logging devices. [8–10]. This essentially connects a modern MHD vehicle to the internet. While this has its benefits, it also creates attack surfaces in a system whose communication protocols were meant to be reliable and fault-tolerant not necessarily secure.

1.2 Security Concerns

While the Controller Area Network (CAN) protocol is widely recognized for its resilience in facilitating real-time communication between ECUs in vehicles, its security vulnerabilities, particularly in medium and heavy-duty (MHD) vehicles, have become a growing concern. CAN has been extensively used in both passenger and commercial vehicles for decades, and multiple studies have demonstrated that it lacks fundamental security features such as encryption and authentication [11–15]. These deficiencies allow attackers to disrupt or manipulate vehicle operations by exploiting both remote and local access points. The SAE J1939 protocol, built on top of CAN, which serves as the backbone for in-vehicle communication in MHD vehicles, is similarly susceptible to cyberattacks [16–23].

Research has shown that attackers can exploit vulnerabilities across different layers of the SAE J1939 protocol. At the application layer, attacks such as message spoofing have been demonstrated, allowing continuous control over critical functions such as engine power and braking. For example, Burakova et al. [16] highlighted how a malicious actor could send ad-hoc messages with a specific PGN (Parameter Group Number) to disable engine braking or reduce engine torque, thus jeopardizing vehicle safety. At the network management layer, Murvay et al. [18] demonstrated

that address claim messages could be exploited to render an ECU inoperative by sending repeated messages with all-zeros in the data field, thereby leading to a denial of service (DoS) for that ECU.

SAE J1939 inherits the vulnerabilities of the CAN protocol, which has been rigorously studied in the context of passenger vehicles. Initial research demonstrated that attackers with physical access to the vehicle's onboard diagnostic (OBD-II) port could inject arbitrary CAN messages to control or disrupt the vehicle's operation [13]. This issue has been further compounded by the discovery that some ECUs are susceptible to remote exploits, making physical access unnecessary in certain attack scenarios [12, 24]. MHD vehicles, similar to passenger vehicles, are exposed to such vulnerabilities through their diagnostic ports, trailer wiring, and other accessible entry points [8]. The increased connectivity and telematics integration in MHD vehicles, aimed at improving transportation management, inadvertently open up new attack surfaces for potential intruders.

The openness of the SAE J1939 standards presents both an advantage and a disadvantage. While it enables interoperability between ECUs from different manufacturers, it also simplifies the process of attack discovery. Burakova et al. [16] and Murvay et al. [18] have demonstrated attacks at both the application and network management layers of SAE J1939. For instance, continuous control over the engine has been achieved by exploiting specific message formats and transmission methods. Such attacks underline the potential risks that these vulnerabilities pose to MHD vehicles, particularly because their role in critical infrastructure could make them attractive targets for cyber attackers.

Despite the lack of documented real-world cyberattacks on MHD vehicles, the increasing reliance on digital systems and the growing number of exposed entry points make them potential targets. Recent cyberattacks on truck manufacturers further illustrate the rising threats in this domain [13]. As MHD vehicles play a pivotal role in essential services and supply chains, any successful cyberattack could have wide-ranging societal and economic impacts. The open availability of standards such as SAE J1939 makes it easier for attackers to study and exploit vulnerabilities, further emphasizing the need for robust security measures.

Security gaps within the broader automotive ecosystem have been revealed through recent research. For example, Kosher et al. [24] identified vulnerabilities in the seed-key exchanges between ECUs in passenger cars, raising concerns about the susceptibility of authenticated communication sessions to brute-force attacks. The use of short seed-key pairs (8-16 bits) in challenge-response authentication can be easily exploited, allowing unauthorized access to critical vehicle functions. Building on this, Miller et al. [13] demonstrated the feasibility of breaking into ECU security systems by reverse-engineering the firmware of diagnostic software used in various vehicles. These findings, while focused on passenger cars, are highly relevant to MHD vehicles as they share many underlying communication protocols and architectures.

In addition to the security concerns around CAN and SAE J1939, the Unified Diagnostic Services (UDS) protocol, governed by ISO 14229 and ISO 15765, has its own set of vulnerabilities. UDS is critical for vehicle diagnostics and repair, allowing authorized devices to communicate with ECUs for fault diagnosis and software updates. However, the lack of strong authentication mechanisms in UDS makes it susceptible to unauthorized access.

Maag et al. [25] contributed to understanding the cybersecurity shortcomings in seed-key exchanges between ECUs and vehicle diagnostics adapters (VDAs) in MHD vehicles. Their work indicated a linear mapping in seed-key pairs, making it feasible to predict these pairs in 16-bit configurations. Kulandaivel et al. [26] uncovered multiple vulnerabilities in the UDS implementation in passenger cars. His research was primarily focused on gaining secured access to ECUs, revealing that seeds used in challenge-response pairs are not entirely random and can be influenced by ECU uptime. This insight into predictable seed generation further emphasizes the vulnerabilities in automotive security protocols.

Diagnostic gateways are often used to mediate communication between the diagnostic tool and the vehicle's internal ECUs. However, these gateways frequently allow all UDS messages to pass through, enabling attackers to send malicious diagnostic commands if they gain access to the network. These vulnerabilities highlight the need for more secure diagnostic communication mechanisms and stricter filtering at the gateway level.

As MHD vehicles increasingly rely on interconnected digital systems, it is vital that their security is prioritized. The SAE J1939 protocol's lack of intrinsic security features, coupled with the expanding attack surface due to telematics and wireless integration, makes these vehicles vulnerable to both direct and remote attacks. As demonstrated through extensive testing, attacks on critical systems like braking, engine control, and transmission can have catastrophic effects on vehicle operations. Therefore, proactive efforts must be undertaken to secure these communication networks and protect MHD vehicles from potential cyber threats.

1.3 Security Solutions

To address the numerous security concerns surrounding CAN and SAE J1939 networks, researchers and industry professionals have proposed a range of mitigation strategies. These strategies typically focus on intrusion detection systems (IDS), cryptographic solutions, and gateway devices. Each of these methods seeks to prevent unauthorized access or manipulation of in-vehicle communication networks.

1.3.1 Cryptographic Solutions

Cryptographic solutions have long been suggested as a method to secure in-vehicle communication by ensuring that messages are authenticated and encrypted before transmission. However, cryptography introduces several challenges in the context of CAN and SAE J1939 networks, particularly for MHD vehicles. The resource constraints inherent to these systems make it difficult to implement heavy cryptographic algorithms without affecting performance [13]. For instance, each CAN frame can carry only 64 bits of data, and adding cryptographic signatures would either require additional frames, increasing overhead or significant modifications to existing message structures.

Furthermore, cryptographic authentication does not protect against attacks from legitimate but compromised ECUs. If an attacker manages to exploit a vulnerable ECU, such as a body controller with wireless connectivity, cryptographic measures may not be able to prevent malicious

commands from being transmitted. In scenarios where a compromised ECU sends a valid command, the receiving ECU may still process the message, highlighting a fundamental limitation of cryptography in preventing such attacks.

Additionally, key management is a critical challenge in the context of MHD vehicles. Dynamic key distribution becomes especially complex in scenarios where trailers are attached to tractors or other vehicles. Securely managing keys between the ECUs of both units requires robust key-granting mechanisms, which add complexity and raise questions about the legitimacy of the ECU requesting the key [24]. Moreover, if an ECU fails or requires replacement during a long-haul operation, managing key revocation and redistribution poses another significant obstacle.

1.3.2 Intrusion Detection and Prevention Systems (IDPS)

Intrusion detection and prevention systems (IDPS) are increasingly being proposed as an alternative or complementary approach to cryptography [27–53]. IDPS solutions can detect abnormal behavior on the network and potentially thwart malicious activity, even if the attacker has compromised a legitimate ECU. Currently, IDPS research for in-vehicle networks has focused on three primary methodologies: anomaly-based, specification-based, and rule-based detection systems.

Anomaly-Based Detection

Anomaly-based systems work by learning the normal behavior of the vehicle network and flagging deviations as potential attacks. These systems can detect unknown or zero-day attacks by observing significant deviations from established patterns [13]. However, anomaly-based systems are not without limitations. Since they rely on data collected during past drive cycles to establish a baseline, they may fail to detect attacks that occur under different network configurations or use cases not accounted for during training [54]. Frequent retraining may be necessary to accommodate changes in vehicle behavior over time, especially in MHD vehicles where network configurations can vary significantly during a vehicle's lifecycle.

Some researchers have explored using physical characteristics, such as the transmitting voltage on the CAN bus, to fingerprint ECUs and detect deviations from normal behavior. However,

the voltage characteristics of ECUs can vary based on environmental factors like temperature and supply voltage, making this approach less reliable [24, 55, 56]. Additionally, periodic traffic patterns on the CAN bus may not account for ad hoc messages, further complicating the accuracy of anomaly detection in dynamic vehicle environments.

Specification-Based Detection

Specification-based systems use predefined manufacturer specifications to establish a reference model for normal network behavior. These systems flag deviations from this model as potential attacks, making them well-suited for detecting unknown threats that violate established operational rules. Studnia et al. [29] proposed modeling ECU behavior using logical expressions derived from manufacturer specifications, while Larson et al. [57] suggested using deterministic finite automata to model network behavior.

However, specification-based systems can still fail to detect attacks that adhere to protocol specifications but exploit vulnerabilities within them. For instance, the engine control attack demonstrated by Burakova et al. [16] involves sending valid messages that conform to the SAE J1939 specifications but are used maliciously to disable critical vehicle functions. These types of attacks illustrate a key limitation of specification-based systems in addressing more sophisticated threats.

Rule-Based Detection

Rule-based systems rely on predefined attack signatures, comparing network messages to a database of known malicious patterns. If a match is found, the system raises an alarm and can potentially block the malicious message from being processed by the target ECU. Rule-based intrusion detection and prevention systems are commonly used in commercial solutions, such as NXP's secure CAN transceivers, which filter messages based on a whitelist or blacklist of CAN IDs [58].

Although rule-based systems generally have low false-positive rates, they struggle to detect more complex attacks that do not have identifiable signatures. For example, a message instructing the vehicle to unlock its doors might be legitimate under normal circumstances but could pose a

safety risk if sent while the vehicle is in motion. Current rule-based systems do not have the contextual awareness to flag such situations, allowing potentially dangerous messages to pass through.

1.3.3 Summary of Existing Solutions

In summary, the existing security solutions for in-vehicle networks can be classified into cryptography-based, anomaly-based, specification-based, and rule-based systems. Each approach has its strengths and limitations, as outlined in Table 1.1.

Table 1.1: Summary of Existing In-Vehicle Security Solutions

Methodology	Pros	Cons	References
Cryptography-based	High security	High resource consumption, communication overhead	[13, 24]
Anomaly-based	Can detect unknown attacks	High false-positive rate, requires retraining	[24, 36]
Specification-based	Strong against protocol violations	Can miss attacks that adhere to specifications	[16, 59]
Rule-based	Low false positives	Limited detection capability for complex attacks	[54, 58]

While each solution offers a unique defense mechanism, none are entirely foolproof, and a combination of methods may be required to achieve comprehensive security in MHD vehicles. As the attack surface for in-vehicle networks continues to expand, particularly with the rise of telematics and wireless systems, it is crucial to adopt a defense-in-depth approach to protect these critical infrastructures.

1.4 Research Questions

This thesis explores several key research questions aimed at identifying and addressing vulnerabilities in the SAE J1939 and UDS protocols. The research questions guiding this work include:

- What are the specific vulnerabilities within the SAE J1939 and UDS protocols that could be exploited in medium and heavy-duty vehicles?

- How do these vulnerabilities manifest in real-world systems, particularly through attack vectors that compromise the ECUs' functionality?
- What are the implications of these vulnerabilities for vehicle safety and operation?
- How effective are current security solutions, such as gateway devices, in preventing attacks, and what enhancements can be made?

By answering these questions, the thesis aims to contribute new insights into vehicular security, with a focus on practical attack scenarios and their real-world consequences.

1.5 Contributions

This thesis makes several key contributions to the field of vehicular cybersecurity, particularly in the context of MHD vehicles:

- Identification of new attack vectors targeting the data-link layer of SAE J1939, extending beyond previous research focused on the application and network management layers.
- Validation of existing attacks proposed by Mukherjee et al. [17] through extensive real-world testing, including trials on a 2014 Kenworth T270 truck.
- Exploration of UDS vulnerabilities, specifically within ISO 14229 and ISO 15765, showing how diagnostic systems can be exploited.
- Analysis of diagnostic gateways, particularly in a Freightliner Cascadia cab setup, highlighting how current gateway configurations can inadvertently expose the vehicle to UDS-based attacks.

1.6 Document Organization

The remainder of this thesis is structured as follows:

- Chapter 2 provides background on the SAE J1939 and UDS protocols, offering the necessary context to understand their vulnerabilities. Additionally, it discusses the threat model used for this research
- Chapter 3 discusses vulnerabilities discovered in the SAE J1939 data link layer protocol along with a detailed description of the testing setups, experiments, results and observations
- Chapter 4 discusses vulnerabilities discovered in the UDS protocol along with a detailed description of the testing setups, experiments, results and observations
- Chapter 6 concludes with key insights from the research and potential directions for future work.

Chapter 2

Background

Medium and heavy-duty (MHD) vehicles serve as a critical backbone for the global economy, responsible for the transport of essential goods, food, and freight across vast distances. As of 2020, there were approximately 15 million registered trucks and buses operating in the United States alone [60]. Historically, these vehicles were significant contributors to pollution due to high emissions. However, modern advancements in technology have allowed for tighter control over engine operations and emissions, resulting in cleaner and more efficient vehicles. Central to these advancements are embedded electronic systems and sophisticated communication protocols that manage vehicle operations.

At the core of these systems are low-power, embedded Electronic Control Units (ECUs), which are responsible for regulating various vehicle functions. ECUs gather data from sensors and other ECUs within the vehicle network to control actuators, ensuring optimal performance. For instance, they monitor inputs such as the accelerator pedal position or engine speed, compute appropriate control signals, and adjust vehicle parameters like fuel injection timing accordingly. Communication between these ECUs takes place over network segments that follow a bus topology, facilitating seamless data exchange. A typical example is the display of engine speed on the vehicle's instrument cluster, which is relayed from the engine control unit via the powertrain network.

In MHD vehicles, communication between ECUs is standardized through the SAE J1939 protocol [2]. This set of standards is organized into layers following the Open Systems Interconnection (OSI) model [61]. The layered structure allows for interoperability between devices from different manufacturers, making it possible to integrate a range of ECUs into the same physical network without compatibility issues. This plug-and-play functionality simplifies the integration process and reduces complexity.

Furthermore, the data generated by these embedded systems is increasingly being leveraged for various purposes, including logistics optimization, asset management, regulatory compliance,

and fuel efficiency improvements. Many MHD vehicles are now equipped with wireless telematics devices, which connect them to external networks and, in some cases, directly to the Internet [8]. These capabilities enable remote vehicle administration, diagnostics, and assistance, providing significant benefits in fleet management and operational efficiency.

2.1 In-Vehicle Networks

The communication systems in medium and heavy-duty (MHD) vehicles rely on a set of important protocols. These include SAE J1939 over Controller Area Network (CAN), ISO 14229 (Unified Diagnostic Services - UDS), and ISO 15765 (Diagnostic Communication over CAN). Each of these protocols serves a special role, from handling everyday vehicle functions to managing diagnostics and safety. In this section, we will take a closer look at each of these protocols to understand how they enable communication in MHD vehicles. In-vehicle communication in medium and heavy-duty vehicles is mostly guided by the SAE J1939 standards over the physical Controller Area Network (CAN). SAE J1939 messages carry operational parameters like engine speed, vehicle speed, switch status, etc. These parameters are bundled into logical groups referred to as Parameter Groups (PG). Each PG is identified by a unique number called a Parameter Group Number (PGN), which is also embedded in the message. Information in the J1939 message is carried in a J1939 Protocol Data Unit (PDU). A J1939 PDU also bears a source address (SA) identifying the sender, a destination address (DA) identifying the receiver, the priority of the message, the PGN, and up to 1785 bytes of data. The priority, PGN, SA, and DA are embedded into the identifier (ID) part of a CAN frame for PDUs with 8 or fewer bytes. For PDUs that have more than 8 bytes, a transport protocol (TP) is used, and the PGN is located in the last 3 bytes of the TP Connection Management (CM) message data. In case of diagnostic messages, the SAE J1939 specifies a special PGN, 55808 (`0x0DA00`).

2.1.1 Controller Area Network (CAN)

The Controller Area Network (CAN) protocol is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without the need for a host computer. Developed by Bosch in 1986, CAN was initially created for the automotive industry to replace complex wiring systems with a simpler, more efficient communication network [4]. Since its inception, CAN has become a critical component in the design of in-vehicle networks, particularly in medium and heavy-duty (MHD) vehicles. Due to its versatility and efficiency, CAN has been widely adopted beyond the automotive industry, finding use in industrial automation, aerospace, and other embedded systems.

CAN Architecture

CAN is a multi-master, message-oriented protocol that operates on a bus topology. The key feature of CAN is that multiple devices, known as nodes, can be connected to the same bus, and any node can transmit data without needing a central coordinator. Communication on the CAN bus is prioritized using message identifiers, and messages with higher priority are given preference in case of simultaneous transmission attempts. This is achieved through a process called arbitration, where nodes with higher priority messages (i.e., messages with lower numerical identifiers) are granted access to the bus, while other nodes must wait until the bus is available again.

CAN frames are sent in a broadcast fashion, meaning that every node on the network can receive the data. However, it is up to each node to determine if the message is relevant to its operation based on the message identifier. This broadcasting mechanism is beneficial in reducing communication overhead but presents potential security challenges, as any node on the network can observe all data being transmitted.

CAN operates at the data-link layer and physical layer of the OSI model, defining how data is encoded, transmitted, and received over the network. The protocol supports speeds of up to 1 Mbps in standard implementations, which is sufficient for most automotive applications where real-time data exchange between ECUs is critical. However, higher-speed implementations such as CAN

FD (Flexible Data-rate) have been developed to accommodate the increasing data requirements of modern vehicles.

CAN Message Structure

CAN messages are typically referred to as frames, and each frame contains a specific set of fields that are used to manage the transmission and arbitration processes. The standard CAN frame consists of the following fields [4]:

- **Start of Frame (SOF)**: This is a single dominant bit that marks the beginning of the frame and serves as a synchronization point for the network.
- **Arbitration Field**: This field includes the 11-bit or 29-bit message identifier (depending on whether standard or extended CAN is being used) and the Remote Transmission Request (RTR) bit, which distinguishes between data frames and remote frames (used to request data from another node).
- **Control Field**: The control field contains the Data Length Code (DLC), which specifies the number of data bytes in the frame (up to 8 bytes in standard CAN).
- **Data Field**: This is the actual data payload, which can contain between 0 to 8 bytes (or more, in the case of CAN FD).
- **CRC Field**: The Cyclic Redundancy Check (CRC) field is used to ensure data integrity by detecting errors during transmission.
- **ACK Field**: This field consists of an acknowledgment slot and delimiter. Any node that correctly receives the frame without errors sends an acknowledgment by overwriting the recessive ACK bit with a dominant bit.
- **End of Frame (EOF)**: This is a series of seven recessive bits marking the end of the frame.
- **Interframe Space**: A mandatory gap between frames that allows nodes time to process received data before the next frame is transmitted.

Table 2.1: CAN Frame Fields

SOF	ID	RTR	IDE	r0, r1	DLC	Data	CRC	ACK	EOF	IFS
1	11/29	1	1	2	4	0-64	15	2	7	3

Legend:

- **SOF** - Start of Frame
- **ID** - Identifier (11 bits for standard frame, 29 bits for extended frame)
- **RTR** - Remote Transmission Request
- **IDE** - Identifier Extension
- **r0, r1** - Reserved bits
- **DLC** - Data Length Code
- **Data** - Data Field (0 to 64 bits or 0 to 8 bytes)
- **CRC** - Cyclic Redundancy Check
- **ACK** - Acknowledgment
- **EOF** - End of Frame
- **IFS** - Intermission Frame Space

The CAN protocol uses a non-return-to-zero (NRZ) bit encoding scheme, where a dominant bit (logical 0) represents a bus voltage, and a recessive bit (logical 1) represents the absence of voltage. Arbitration is resolved based on the dominance of bits: if a node transmits a dominant bit while another node transmits a recessive bit, the node sending the dominant bit gains control of the bus.

Advantages and Limitations of CAN

One of the major advantages of CAN is its efficiency and fault tolerance. The protocol includes built-in error detection mechanisms such as the CRC check, bit stuffing (inserting additional bits after a series of identical bits to prevent misinterpretation), and error frames that notify all nodes of detected errors. Nodes experiencing errors transition into one of several error states, with the most severe state, "bus off," preventing a faulty node from further transmissions to protect the network.

Additionally, CAN's arbitration method ensures that critical messages, such as those related to vehicle safety, are prioritized over less important data, like diagnostic information. This characteristic makes CAN highly suitable for real-time applications where timely data delivery is essential.

However, despite these strengths, CAN also has limitations, particularly in terms of security. The protocol was not originally designed with security in mind, and as a result, it lacks basic security features such as encryption and authentication. Every node on the network can read all messages, making it vulnerable to attacks like message spoofing, injection, and denial-of-service (DoS) attacks. With the increasing connectivity of modern vehicles through wireless interfaces and telematics systems, these vulnerabilities have become a significant concern for automotive cybersecurity [13].

Moreover, the standard CAN protocol supports a maximum data payload of 8 bytes per frame, which can be limiting in modern vehicles where ECUs are exchanging large amounts of data. CAN FD, an extension of the original CAN protocol, addresses this limitation by allowing larger data frames (up to 64 bytes) and faster transmission speeds. However, many legacy systems still rely on the traditional CAN protocol, leaving them susceptible to bandwidth and performance constraints.

CAN in Medium and Heavy-Duty Vehicles

In medium and heavy-duty (MHD) vehicles, CAN plays a pivotal role in enabling communication between ECUs responsible for managing various functions such as engine control, transmission, braking, and safety systems. The SAE J1939 protocol, which operates on top of the CAN protocol, is the de facto standard for communication in MHD vehicles. J1939 standardizes higher-

layer protocols for diagnostics and control, making it possible for different ECUs from various manufacturers to communicate seamlessly over the same CAN network.

MHD vehicles typically rely on multiple CAN networks, each dedicated to a specific set of ECUs. For example, the powertrain CAN network may be responsible for engine and transmission control, while the body CAN network handles auxiliary functions like lighting and HVAC. The modularity and reliability of CAN make it well-suited for these types of applications, but the security vulnerabilities inherent in the protocol present challenges for ensuring the safety and integrity of these critical systems.

Security Challenges

As MHD vehicles become increasingly connected to external networks through telematics systems and wireless interfaces, the potential for cyberattacks on CAN networks grows. Attackers can exploit vulnerabilities in CAN to disrupt vehicle operations, gain control over ECUs, or perform denial-of-service attacks. Given that CAN does not natively support authentication or encryption, any node with access to the network can potentially transmit malicious messages, making unauthorized message injection a significant threat.

In light of these challenges, modern cybersecurity research in the automotive industry is focused on developing intrusion detection systems (IDS) and other security measures to protect CAN networks from malicious activities. These solutions aim to monitor CAN traffic in real-time and detect anomalies that may indicate an attack, providing an additional layer of protection for vehicles relying on CAN.

In summary, while CAN is a powerful and widely-used protocol in automotive applications, its lack of inherent security features makes it vulnerable in modern, connected vehicles. Ensuring the security of CAN-based systems, particularly in MHD vehicles, requires ongoing efforts to develop and implement robust security solutions.

2.1.2 SAE J1939

In-vehicle communication in medium and heavy-duty vehicles is mostly guided by the SAE J1939 standards. SAE J1939 messages carry operational parameters like engine speed, vehicle speed, switch status, etc. These parameters are bundled into logical groups referred to as Parameter Groups (PG). Each PG is identified by a unique number called a Parameter Group Number (PGN), which is also embedded in the message. Information in the J1939 message is carried in a J1939 Protocol Data Unit (PDU). A J1939 PDU also bears a source address (SA) identifying the sender, a destination address (DA) identifying the receiver, the priority of the message, the PGN and up to 1785 bytes of data. The priority, PGN, SA, and DA are embedded into the identifier (ID) part of a CAN frame for PDUs with 8 or less bytes. For PDUs that have more than 8 bytes, a transport protocol (TP) is used, and the PGN is located in the last 3 bytes of the TP Connection Management (CM) message data. The TP message has a PGN of 60416 (`0x0EC00`). The CAN ID is used at the time of bus arbitration in case of transmission collisions. As such, the CAN frame identifier with the lowest numerical value wins the arbitration and occupies the bus.

Table 2.3: SAE J1939 Document Organization

Layer	**SAE J1939 Documents**
Physical	SAE J1939-11, SAE J1939-13, SAE J1939-14, SAE J1939-15, SAE J1939-16
Data-Link	SAE J1939-21
Network	SAE J1939-31
Application	SAE J1939-71, SAE J1939-73, SAE J1939-74, SAE J1939-75, SAE J1939 Digital Annex
Network Management	SAE J1939-81, SAE J1939-82, SAE J1939-84

CAN allows a maximum of 64 bits of data transfer in one frame. However, SAE J1939 messages are allowed to carry parameter data equivalent of 64 bits or greater. For message data that is greater than 64 bits, SAE J1939 specifies the transport protocol. Depending on the recipient of the information, SAE J1939 specifies two types of message transport protocols: destination-specific (i.e. from a specific source address to a specific destination address) and broadcast (i.e. from a specific source address to the entire network).

Table 2.5: J1939 Message Fields

Priority	**EDP**	**DP**	**PGN (PF, PS)**	**SA**	**DA**	**Data**
3 bits	1 bit	1 bit	18 bits	8 bits	8 bits	0-64 bits

Legend:

- **Priority**: Priority of the message (3 bits).
- **EDP**: Extended Data Page bit.
- **DP**: Data Page bit.
- **PGN (PF, PS)**: Parameter Group Number, including the PDU format (PF) and PDU specific (PS) (18 bits total).
- **SA**: Source Address (8 bits).
- **DA**: Destination Address (8 bits).
- **Data**: The payload of the message, ranging from 0 to 64 bits.

Table 2.7: Formats of Messages Used in Multi-packet Message Transfer

Label	**PGN**	**Parameter**	**Data Bytes**			
			1	**2**	**3**	**4**
Request To Send (RTS)	60416	16	Number of bytes to send	Number of packets to send	Maximum number of packets that can be sent in response to one CTS	PGN of the message being transferred
Clear To Send (CTS)	60416	17	Number of packets that can be sent	Next packet number to send	FF16	PGN of the message being transferred
End of Message Acknowledgment (EoMA)	60416	19	Number of bytes received	Number of packets received	FF16	PGN of the message being transferred
Abort	60416	255	Reason for abort	Role of Sender	FF16	PGN of the message being transferred
Data Transfer (DT)	60160	Sequence number	Data bytes to send	-	-	-

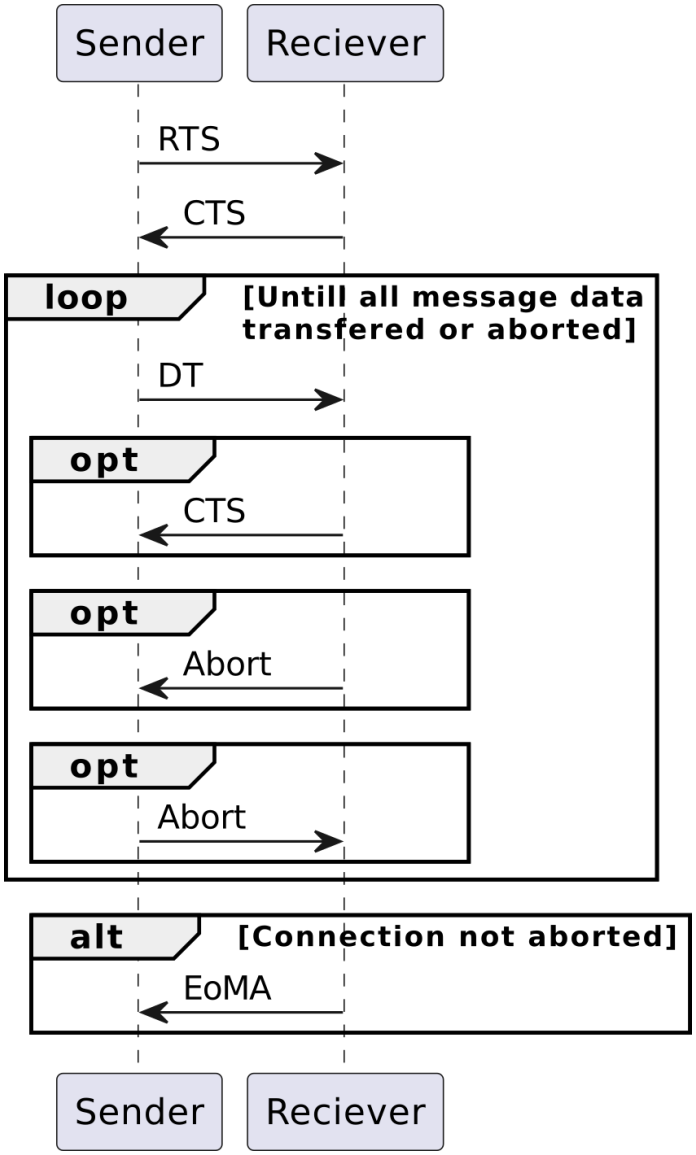

Figure 2.1: Logical Point-to-Point or Destination Specific Message using the Transport Protocol

As shown in Fig. 2.1, in a destination specific transfer, the sending party attempts to open a connection by sending a Request to Send (RTS) message. An RTS message carries information about the total number of packets and the number of bytes to be sent. In response to the RTS, the receiver sends a Clear to Send (CTS) message. CTS messages enforce flow control by limiting the number of packets that can be sent after they are transmitted. CTS messages also aid in missing packet retransmission by indicating the next packet number to be sent. Upon receiving the CTS, the sender can send message data using data transfer (DT) messages until all data bytes are sent.

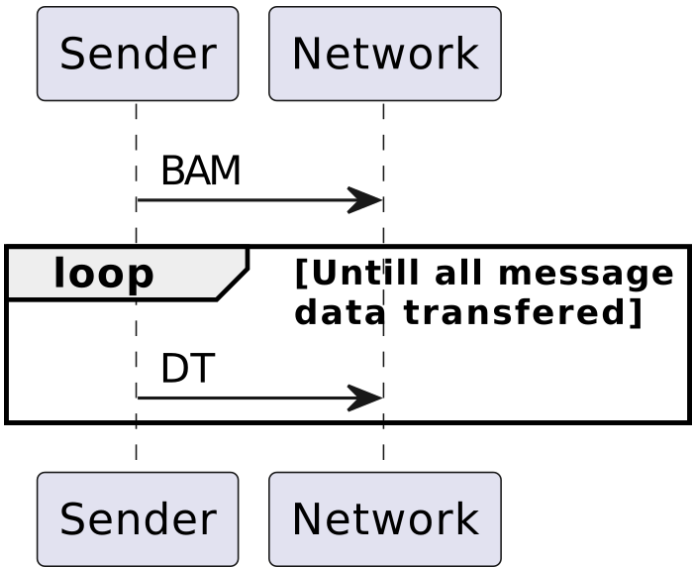

Figure 2.2: Transport Protocol for a Broadcast Announce Message (BAM)

The PGN of a DT message is 60160 (`0x0EB00`). The first byte of this message is reserved for a sequence number. Sequence numbers are used to reassemble the incoming data bytes. The remaining seven bytes of data in a DT message are used to transport the data bytes from the multipacket message being transferred. The flow of DT messages can be aborted by optionally sending an Abort message that includes information stating the reason for the abort. On successful completion of the transfer (i.e. if not aborted), an End of Message Acknowledgment (EoMA) is sent by the receiver to indicate closure of the connection.

As shown in Fig. 2.2, the sending party initiates a broadcast message transport by sending a broadcast announcement message (BAM). Similar to the RTS message, the BAM message carries information about the total number of packets and the number of bytes to be sent. Soon after, the message data is transported in data transfer or DT packets that are formatted the same way as in the destination-specific data transfer. No EoMA is required at the end of a broadcast announcement message.

Both destination-specific and broadcast message transport can be initiated by sending an SAE J1939 specified request message. This message has a PGN of 59904 (`0x0EA00`) and carries the

requested PGN in the last 3 bytes of the data field. Requests can be directed to a specific device or to the global address of 255 (`0xFF`).

2.1.3 UDS

Table 2.9: UDS Document Organization (ISO 14229 & ISO 15765)

Part	**ISO Documents**
Unified Diagnostic Services (UDS)	ISO 14229-1 (UDS Application Layer)
Session Layer Services	ISO 14229-2
Diagnostics Communication over CAN (DoCAN)	ISO 14229-3, ISO 15765 (ISO-TP)
Diagnostics Communication over IP (DoIP)	ISO 14229-5
Diagnostics Communication over LIN	ISO 14229-6

ISO 14229: Unified Diagnostics Servies

ISO 14229, commonly referred to as Unified Diagnostic Services (UDS), is a critical protocol in automotive diagnostics, facilitating communication between a vehicle's Electronic Control

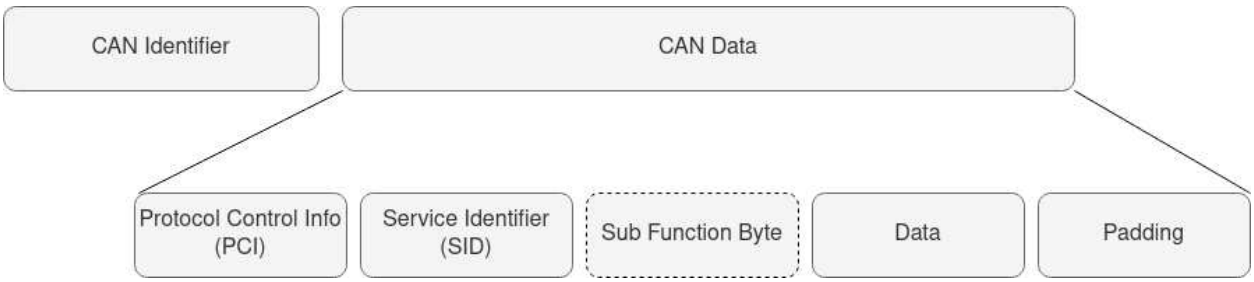

Figure 2.3: UDS Message Structure

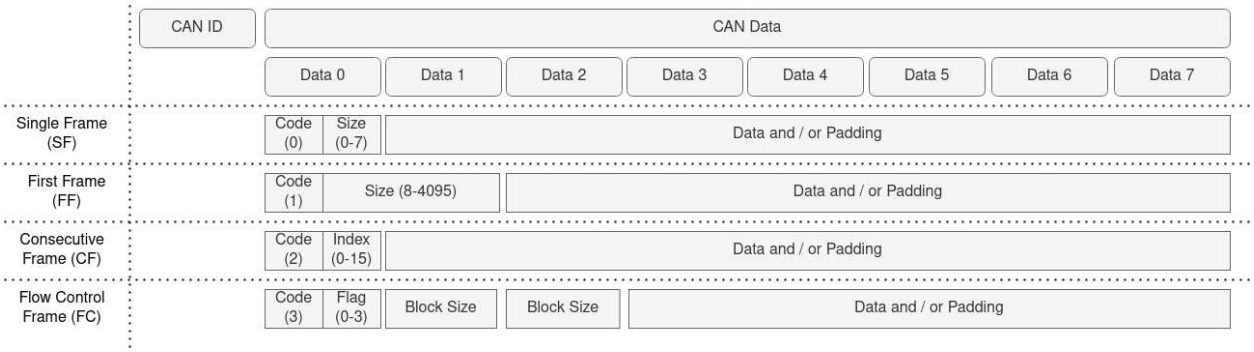

Figure 2.4: Different UDS Messages

Units (ECUs) and external diagnostic tools. Central to UDS are several key components that structure the diagnostic communication process as shown in Fig. 2.3. The Protocol Control Info (PCI) serves the role of identifying the type of UDS message, the size of the message, or other identifying parameters of the message. Accompanying the PCI is the Service Identifier (SID), which is pivotal in specifying the type of diagnostic service or function being requested or performed. Additionally, UDS messages often include a Sub-function field, providing further detail or instructions related to the diagnostic service. The Data segment of the message then carries the specific information or command pertinent to the service request. This structured approach enables a wide range of diagnostic operations, including reading or writing data, testing functions, and acquiring information about ECUs or the vehicle.

To illustrate the application of SIDs within UDS, Table 2.11 presents common service identifiers, along with typical request codes, positive response codes, and codes for negative responses.

Table 2.11: Common Service Identifiers in UDS

SID	Service	Positive Response
0x10	Diagnostic Session Control	0x50
0x11	ECU Reset	0x51
0x14	Clear Diagnostic Information	0x54
0x19	Read DTC Information	0x59
0x22	Read Data by Identifier	0x62
0x27	Security Access	0x67
0x28	Communication Control	0x68
0x2E	Write Data by Identifier	0x6E
0x31	Routine Control	0x71
0x3E	Tester Present	0x7E
Common Negative Response		0x7F

ISO 15765: Diagnostic Communication over CAN

ISO 15765 is an automotive standard for communication over the Controller Area Network (CAN), especially for diagnostics. As shown in Fig. 2.5, it handles the transmission of data packets larger than the limit of a single CAN frame, which is crucial for tasks like detailed diagnostics and ECU programming. Data in ISO 15765 is transmitted in different frame types as shown in Fig. 2.4, each with a specific format and purpose:

1. Single Frame (SF): Used for data up to 7 bytes. It starts with 0 in the first nibble, followed by the data length in the next nibble (half-byte). The remaining bytes carry the actual data.

2. First Frame (FF): Initiates a multi-frame transmission for data exceeding 7 bytes. It begins with 1 in the first nibble, followed by the length of the total data payload in the next 3 nibbles (12-bits). The rest of the frame contains the beginning portion of the data.

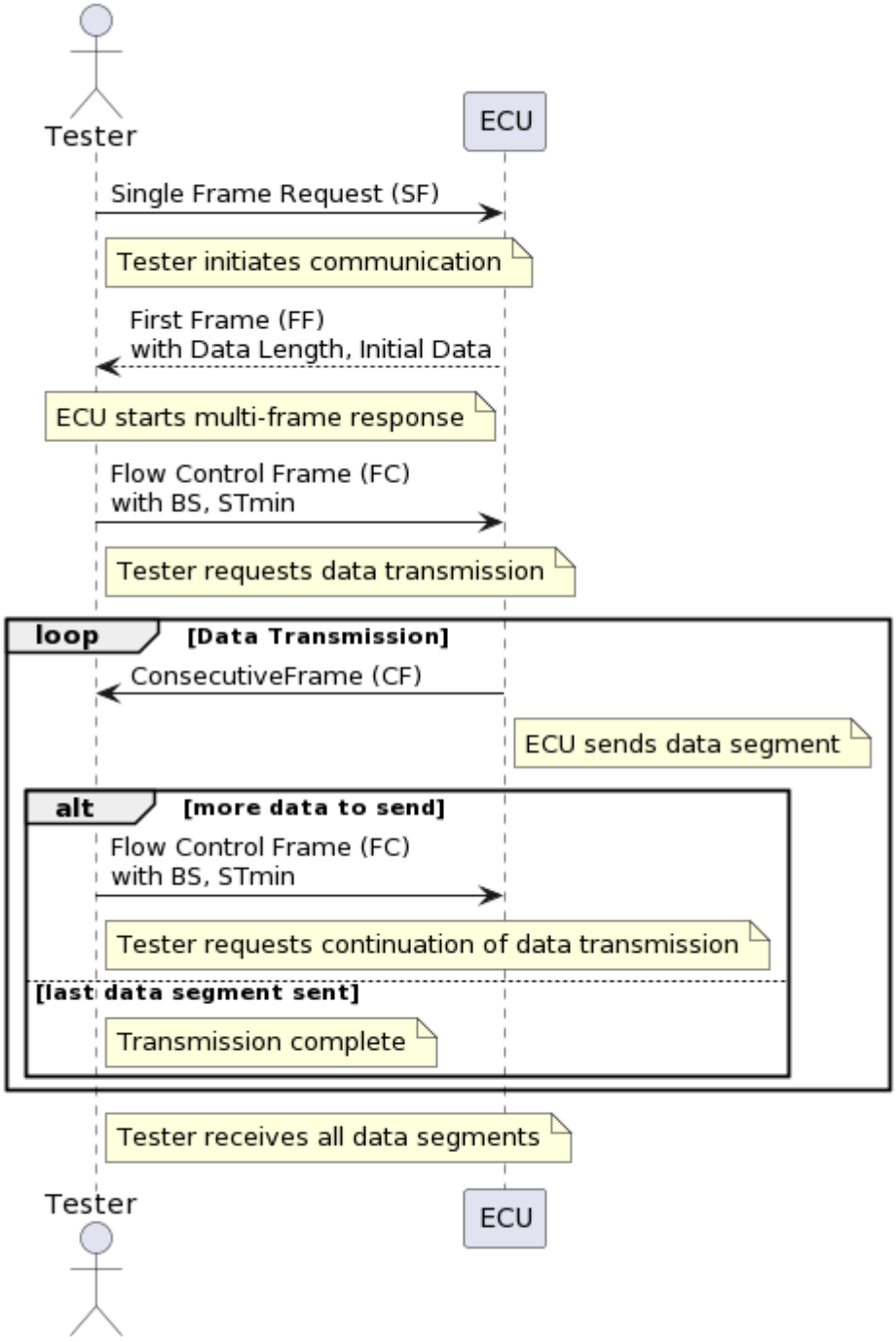

Figure 2.5: Logical Point-to-Point Multiframe Data Transfer using ISO-TP

3. Consecutive Frame (CF): Transports subsequent data chunks after the first frame. Each CF starts with 2 in the first nibble, followed by a frame number that increments with each consecutive frame in the second nibble. This number helps in keeping track of the sequence of the frames.

4. Flow Control Frame (FC): Manages the rate of data transmission in a multi-frame message. It begins with 3, and the next nibble indicates the flow status: 0 for Continue To Send (CTS), 1 for Wait, and 2 for Overflow/Abort. The CTS allows the transmission to proceed, Wait tells the sender to pause, and Overflow/Abort indicates that the receiver cannot handle more data. The frame also specifies how many frames can be sent at a time and the time gap between frames.

These frames collectively enable the transmission of large data packets over CAN. Single frames are for small data packets, while the combination of first, consecutive, and flow control frames manages larger data transfers efficiently and reliably.

The security aspects of automotive protocols, though critical, have historically been under-emphasized in research. Recent efforts, however, have begun to uncover multiple vulnerabilities within these protocols.

Kosher et al. [24] shed light on the vulnerabilities in seed-key exchanges among ECUs in passenger cars, revealing that commonly used 8 or 16-bit seed-key pairs are susceptible to brute-force attacks. This finding raises concerns about the ease with which authenticated security sessions can be compromised, potentially granting unauthorized access to critical ECU functionalities. Expanding on this, Miller et al. [13] demonstrated the feasibility of breaking security authentication to ECUs in passenger cars. Through reverse engineering the firmware of diagnostic software in a Ford Escape and a Toyota Prius, they discovered the algorithm used for key calculation from seeds. Their research illustrated several practical attacks, including manipulating brakes, lights, and engine functions, thereby highlighting security gaps.

2.2 Threat Model

This section outlines the threat model used in this research, detailing the attacker's capabilities and the strategies they might employ. The intrusion detection and prevention mechanisms proposed in this dissertation are designed to defend against the described attacks as well as those presented in later chapters.

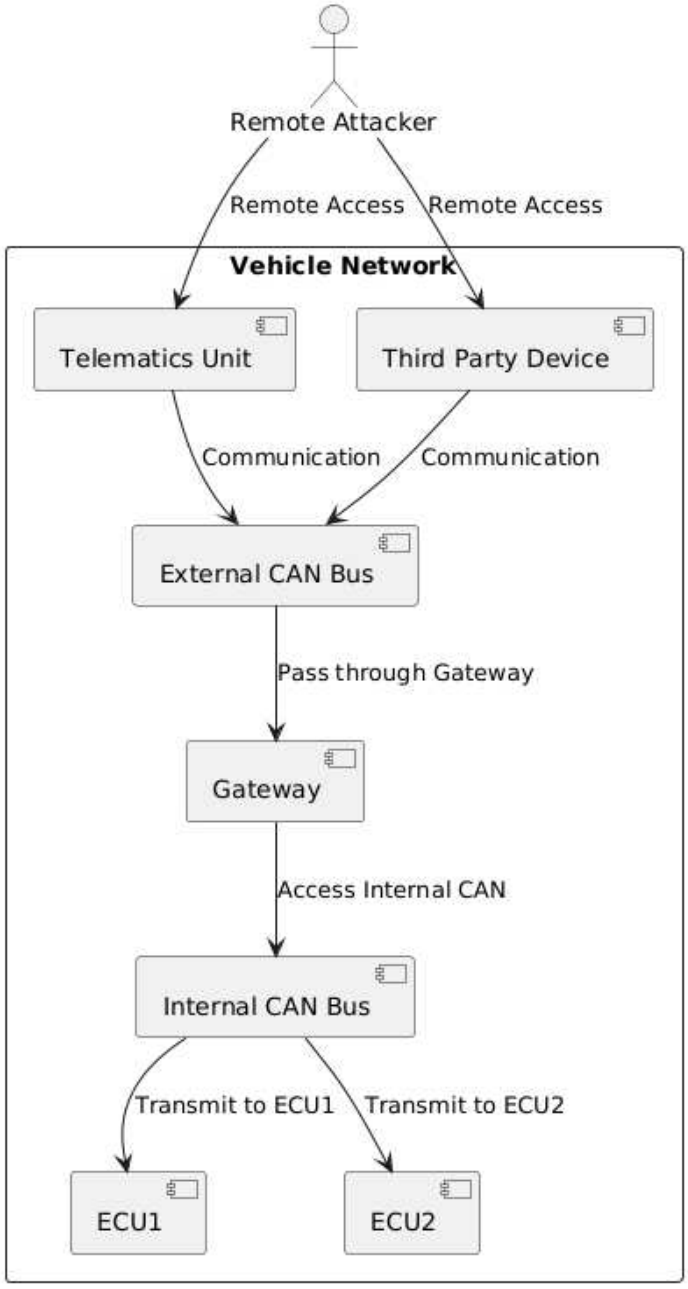

Figure 2.6: Threat Model: Remote Attacker accessing CAN Bus via Telematics or Third-Party Device.

2.2.1 Attacker Capabilities

In scenarios where the attacker gains physical access to the vehicle, they can be considered as powerful as an insider. Such an attacker could install custom hardware on the vehicle's network or tamper with existing security mechanisms. However, in this dissertation, we assume that the attacker does not possess physical access to the vehicle. Instead, the attacker gains access remotely, as depicted in Figure 2.6. Once inside the network, the attacker can transmit CAN frames using the onboard CAN controller or bypass it entirely by injecting NRZ pulses directly into the transceiver. Previous studies have demonstrated that this allows manipulation of bits in CAN frames, potentially leading to the shutdown of another ECU on the network and the ability to transmit in its place [62]. However, the success of such an attack depends on several factors, including physical access and existing vulnerabilities in the CAN controller, which might not always be present in real-world scenarios. Moreover, these attacks can often be detected [63]. As a result, this work assumes that the attacker cannot alter CAN bits directly.

An alternative strategy for the attacker is to reprogram an ECU on the network, gaining control over its inputs to the transceiver. However, such an attack typically requires elevated privileges, which are often difficult to obtain. Insiders, such as mechanics, drivers, or manufacturers, could theoretically carry out these actions, but this dissertation assumes that insiders are trusted individuals. Thus, we do not consider insider threats as part of the attacker's capabilities.

The attacker may target any ECU on the network they have direct access to, or alternatively, they may attempt to breach a different network by bypassing gateway firewalls. Furthermore, the attacker is assumed to have the ability to interpret and generate SAE J1939 messages, spoofing the source address of another ECU in the process.

2.2.2 Attack Strategies

Once the attacker can inject CAN frames into the network, a wide range of cyberattacks becomes possible. These attacks may either be specific to MHD vehicles or adapted from existing attacks on passenger vehicles. The following types of attacks are considered in this dissertation:

High Volume Denial-of-Service (HVDoS)

In an HVDoS attack, the attacker overwhelms the network by rapidly injecting large numbers of SAE J1939 messages, effectively consuming the available resources of the ECUs. This results in a denial of service as the ECUs can no longer process legitimate messages.

Published Attacks

One well-known attack is the network overload attack, first demonstrated by Miller et al. [13]. By injecting frames with CAN ID 0, the attacker can completely exhaust the network's available bandwidth, since CAN arbitration always favors the lowest ID. Experiments on the Kenworth T270 research truck confirmed that this type of attack can halt a running vehicle, a result also observed by Miller et al. in passenger vehicles like the Ford Escape and Toyota Prius.

Low Volume Denial-of-Service (LVDoS)

In contrast to HVDoS, LVDoS attacks aim to disable specific ECU services by injecting messages at a normal rate, making them harder to detect.

Address Claim Attack

As outlined in section 2.1.4, ECUs relinquish their network address when a contending address claim message with a lower NAME value is received. Murvay et al. [18] demonstrated that this behavior can be exploited to make an ECU inoperable by repeatedly sending address claim messages containing the target's address and all zeros in the data field. Our experiments on the Kenworth T270 truck confirmed that this attack stops communication to and from the targeted ECU. If the attacked ECU is critical, such as the engine controller, safety systems like anti-lock braking and traction control can be disabled, and the transmission can be interrupted.

Command and Control (CnC)

In a CnC attack, the attacker attempts to manipulate the cyber-physical operations of an ECU by sending command messages defined by the SAE J1939 standards [64]. Examples of CnC attacks include commands to unlock vehicle doors or activate windshield wipers. Such attacks have been

demonstrated on passenger vehicles [13] and are considered feasible in MHD vehicles due to the publicly available message format specifications in the SAE J1939 standards.

Published Attacks

Burakova et al. [16] demonstrated several CnC attacks, including the "engine control attack," where continuous control over the engine can be achieved by sending ad hoc messages with PGN 0000016. Another example is the "retarder jam attack," where engine braking is disabled by commanding 0% torque to the engine retarder via messages with PGN 0000016. A third attack, the "throttle jam attack," effectively cuts off driver input to the accelerator pedal by sending low torque requests to the engine controller.

Fuzzing

Before launching these types of attacks, an attacker may first use fuzzing techniques to discover vulnerabilities in the target ECU. Fuzzing involves sending randomized CAN IDs and data bytes at high rates, which can sometimes trigger safety-critical scenarios [41, 65]. While fuzzing has been reported in passenger vehicles, there is little published evidence of its use in MHD vehicles. Our experiments found that fuzzing caused erratic gauge movements in the Kenworth T270's instrument cluster, though no immediate impact on vehicle motion was observed.

Replay Attacks

Replay attacks, where previously captured CAN messages are resent to the network, can disrupt normal message periodicity. However, such attacks are easily detectable [24]. To maintain stealth, this dissertation assumes the attacker avoids using replay attacks.

Chapter 3

Vulnerabilities in SAE J1939 Protocol

3.1 Testing Setup

3.1.1 Bench-top Testbeds

The local testbed setup on the bench consisted of four different configurations (Testbed 1, Testbed 2, Testbed 3, and Testbed 4) as shown in Figs. 3.1, 3.2, 3.3, and 3.4. Each of the four testbed configurations hosted a different Engine Control Module (ECM): Testbed 1 had an older generation Cummins 870 ECM, while Testbed 2 had a newer Cummins 2350 ECM. Testbed 3 had an older Caterpillar ADEM 3 ECM while Testbed 4 had a newer generation Caterpillar ADEM 4 ECM. Each ECM had two different controller applications (CA) running on them, one was a CA for Engine #1, and the other was a CA for the engine retarder, which is commonly known as the jake brake. Each testbed setup also included a Bendix electronic brake controller (EBC). A CAN backbone connected the ECM to the EBC and a Linux laptop was used to capture and transmit CAN messages through a CAN to USB device accessed by SocketCAN and the can-utils software. The laptop was also used as the attacker's pivot point and, in some cases, as nodes participating in the attack(s). One or more power supplies were also included.

Our research truck is a PACCAR PX-7-powered 2014 Kenworth T270. There is a Cummins CM2350 engine ECU (same as Testbed 2), a Bendix EC-60 brake ECU, and an Allison RDS-2000 transmission ECU on the truck. These ECUs communicate with each other on a 250 kbps CAN bus. A picture of this truck along with its dashboard is provided in Fig.3.5 and additional details are available in Ref. [66].

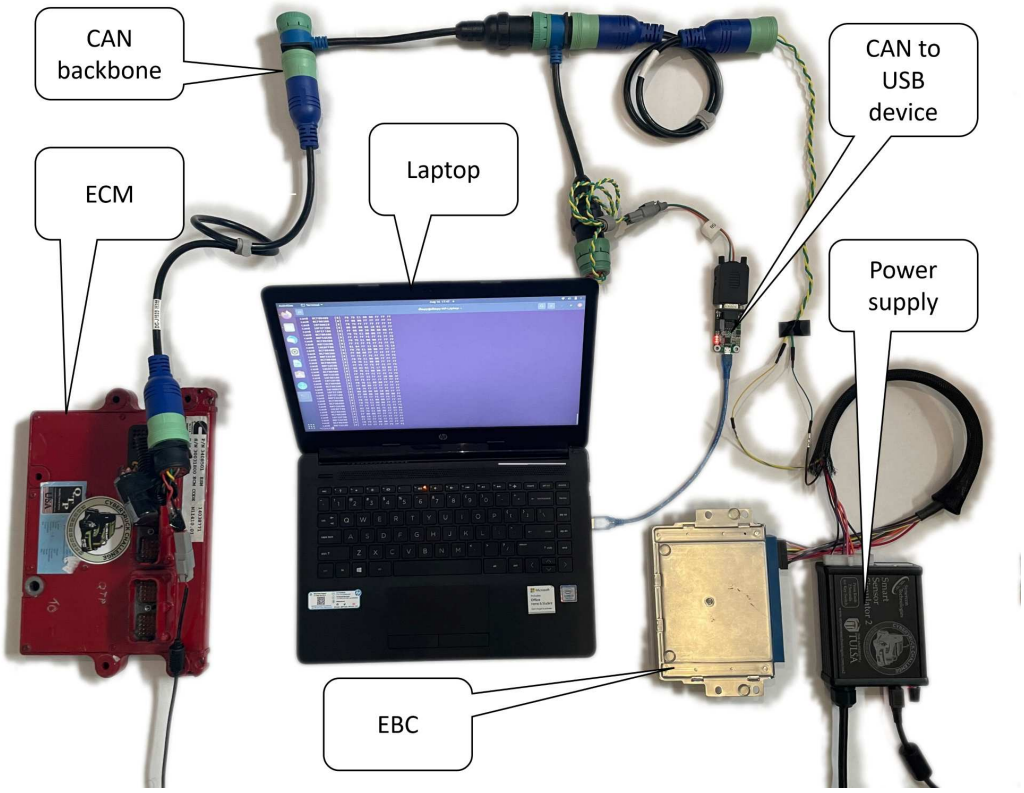

Figure 3.1: Testbed 1

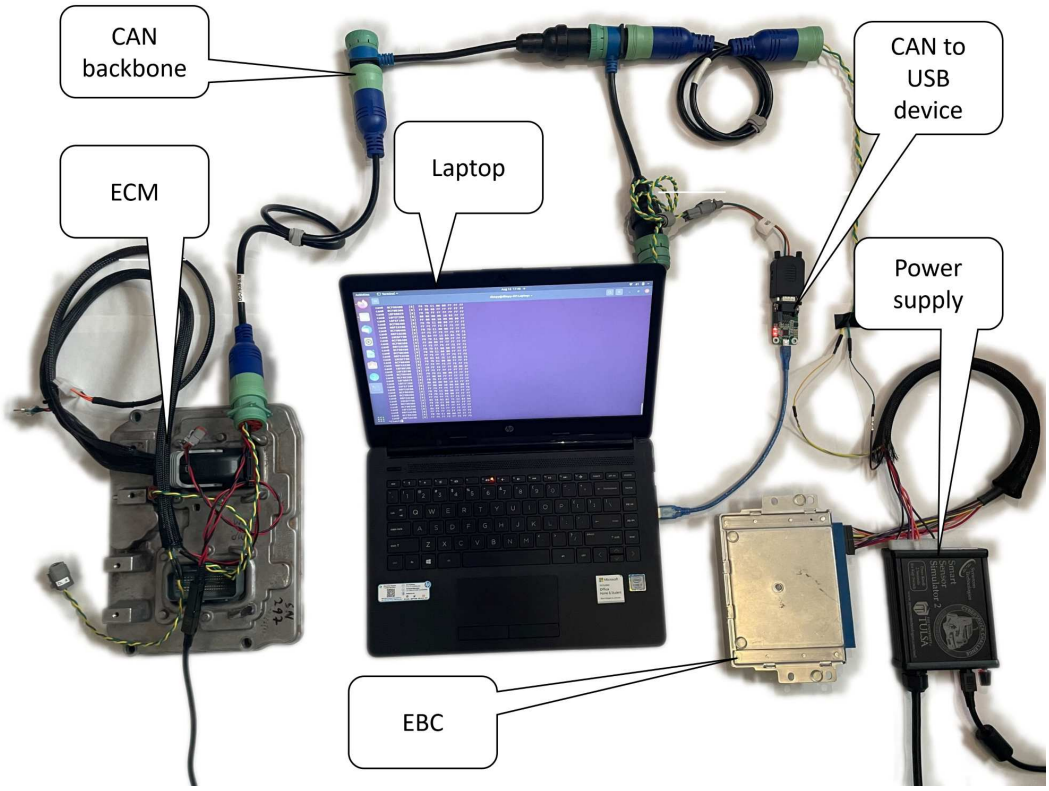

Figure 3.2: Testbed 2

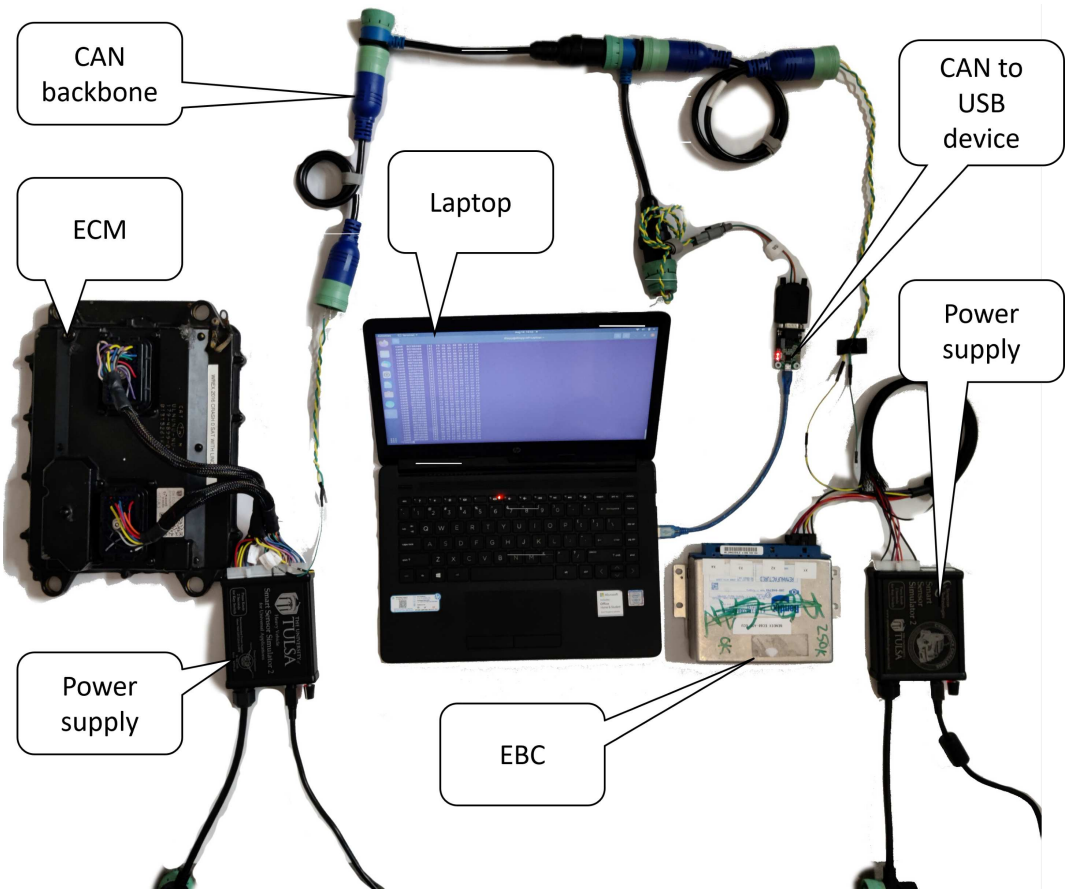

Figure 3.3: Testbed 3

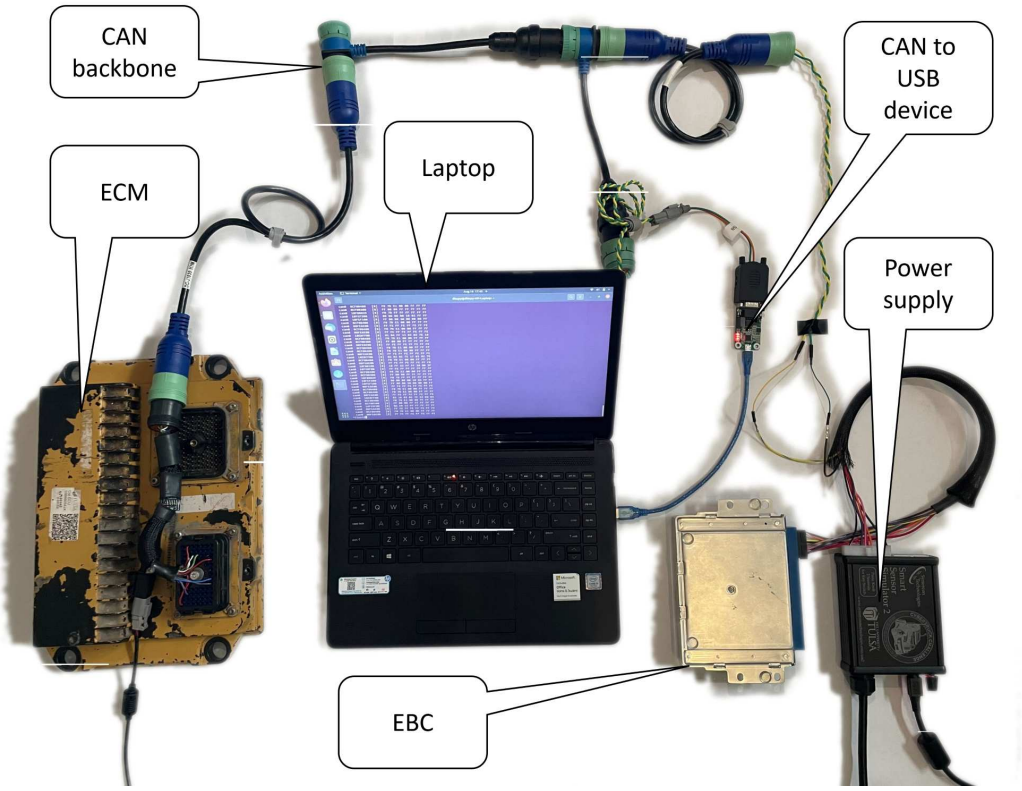

Figure 3.4: Testbed 4

3.1.2 Research Truck

3.2 Request Overload

The first of the five attacks was called the "Request Overload" as it involves overloading a target ECU with request messages for attacker-chosen PGNs.

3.2.1 Hypothesis

The SAE J1939-21 document specifies that all directed requests to an ECU must be processed. An attack can thus be constructed to send a high volume of J1939 request messages, PGN 59904 (`0xEA00`), to the target ECU with the expectation that, in an attempt to serve the sent requests, the ECU fails to perform regular, more critical tasks like the transmission of periodic messages. Fig. 3.6 shows a sequence diagram depicting the hypothesis of the attack.

Figure 3.5: 2014 Kenworth T270 Research Truck

3.2.2 Testing

Mukherjee et al. [17] tested the attack by sending a high volume of requests to a target engine control module (ECM). We did the same, albeit across multiple testbed configurations. Additionally, we also addressed a few shortcomings of the previous experimentation process. Firstly, it was not investigated if the drop in count was because of a request overload or messages losing arbitration to higher priority request messages. Secondly, it was not investigated if the rate of injection of the request messages had any relation with the success of the attack. Thirdly, not all PGNs are utilized in a controller application. Therefore, it was not investigated if requesting a Parameter Group Number (PGN) that is not used had any effect on the network traffic.

While conducting our experiments on the local testbed, we addressed the aforementioned shortcomings. We sent four different types of messages on the CAN network 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 milliseconds. The first type of message had a CAN ID of `0x00000000`. This is the lowest-valued CAN ID and is expected to flood

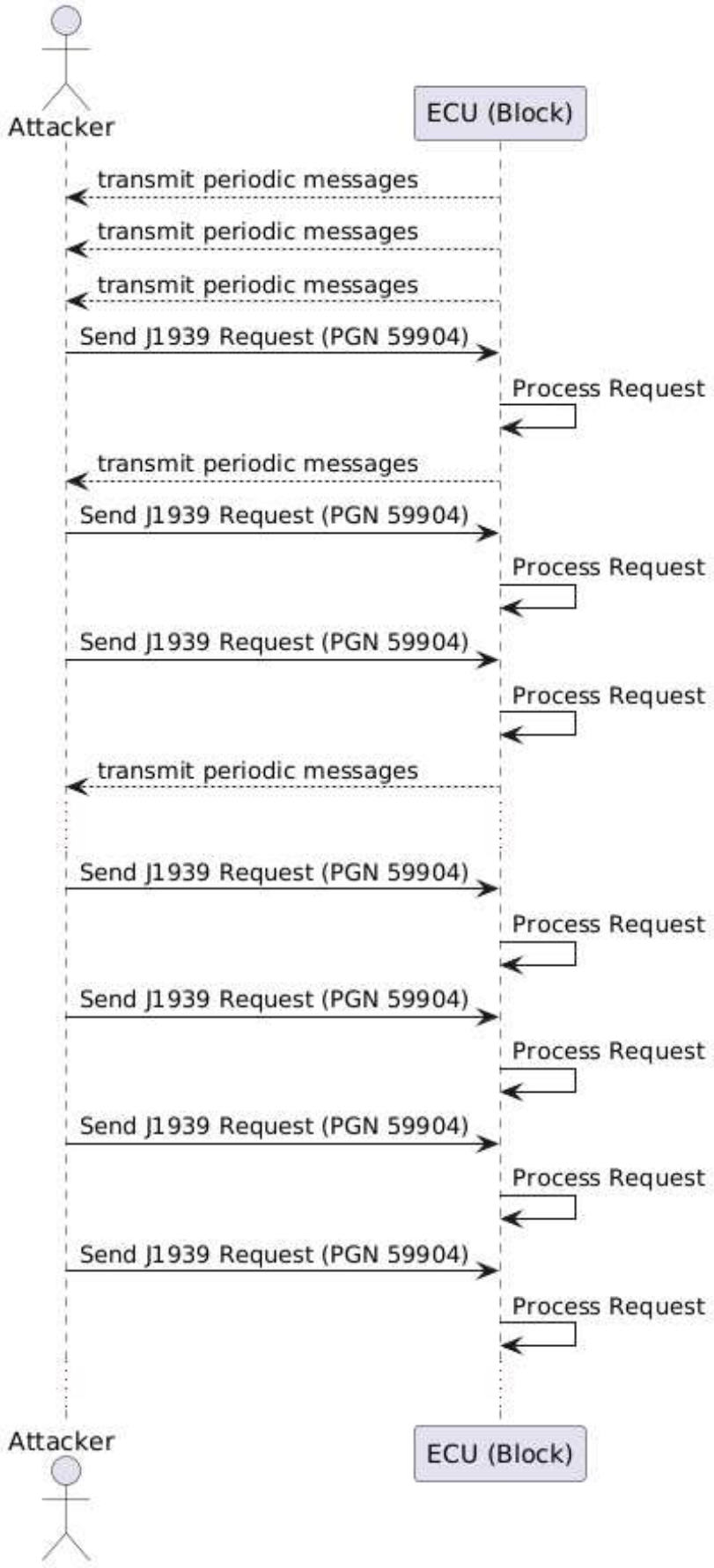

Figure 3.6: Request Overload Hypothesis

the entire bus by winning transmission arbitration. If the CAN bus is saturated with these high priority messages, no other message can gain access to the network. The second type of message had the lowest SAE J1939 priority (7), but 0 for the PGN, DA, and SA, which produced an extended CAN ID of `0x1C000000`. The third type of message had the same SAE J1939 priority (7) but was the request PGN 59904 from the engine controller to itself. The equivalent CAN ID was `0x1CEA0000`. The engine controller responded to this request. Also, this request was for the Component Identifier PG (PGN 65259), which was present with all the ECMs used in the experiment. This is a valid request and is often part of a diagnostics routine. Finally, we sent a request for PGN `0xFFFF`, which was not present in any engine controller, so this was an invalid request. The purpose of sending the invalid request was to address the final shortcoming in [17]. The first three types of messages were sent to address the first shortcoming. Messages with CAN ID `0x00000000` were sent to demonstrate network overload and the removal of all normal traffic from the bus. Messages with CAN ID `0x1C000000` were sent to demonstrate that lowering the SAE J1939 priority from 0 to 7 (causing a change in ID from `0x00000000` to `0x1C000000` had no effect on normal traffic. As such, our goal was to observe if messages with CAN ID `0x1CEA0000` (SAE J1939 priority = 7) had any effect on ECM traffic only. If so, we could conclude request overload had an effect on the traffic emanating from the target ECM and not the brake, thereby demonstrating two things: 1) request overload is indeed a targeted attack on broadcast CAN and 2) it works even at the lowest SAE J1939 priority (7).

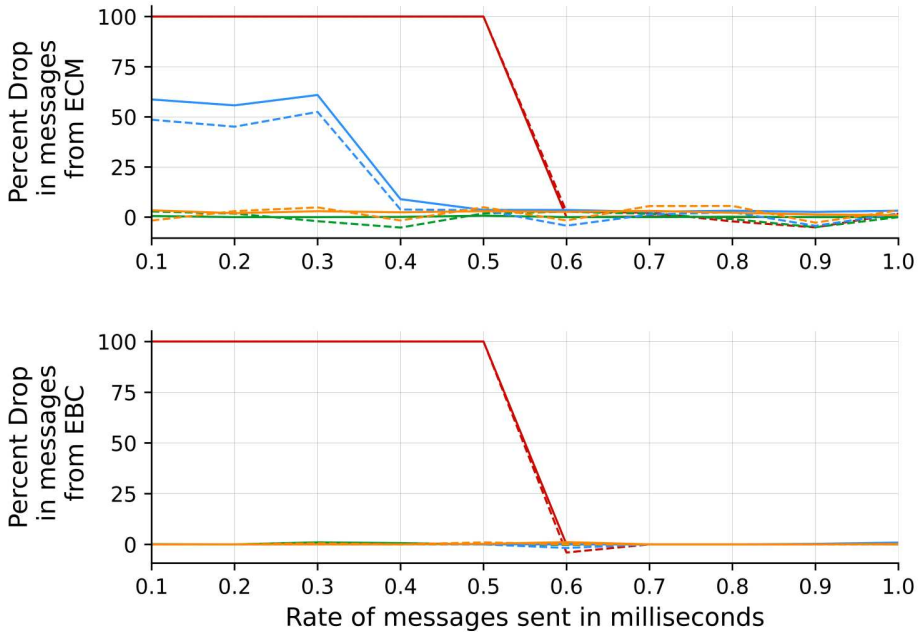

Figure 3.7: Request Overload Results on Local Testbed 1

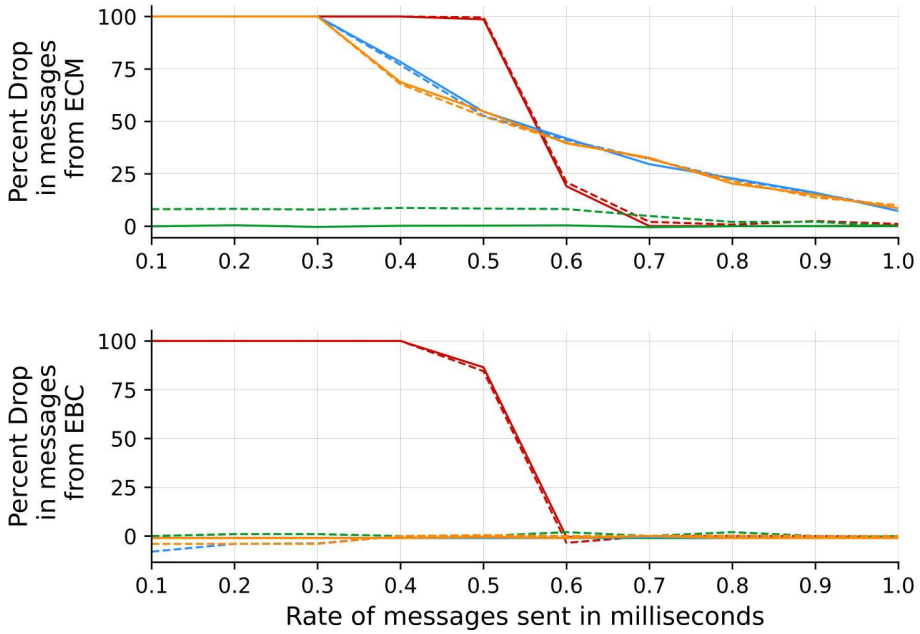

Figure 3.8: Request Overload Results on Local Testbed 2

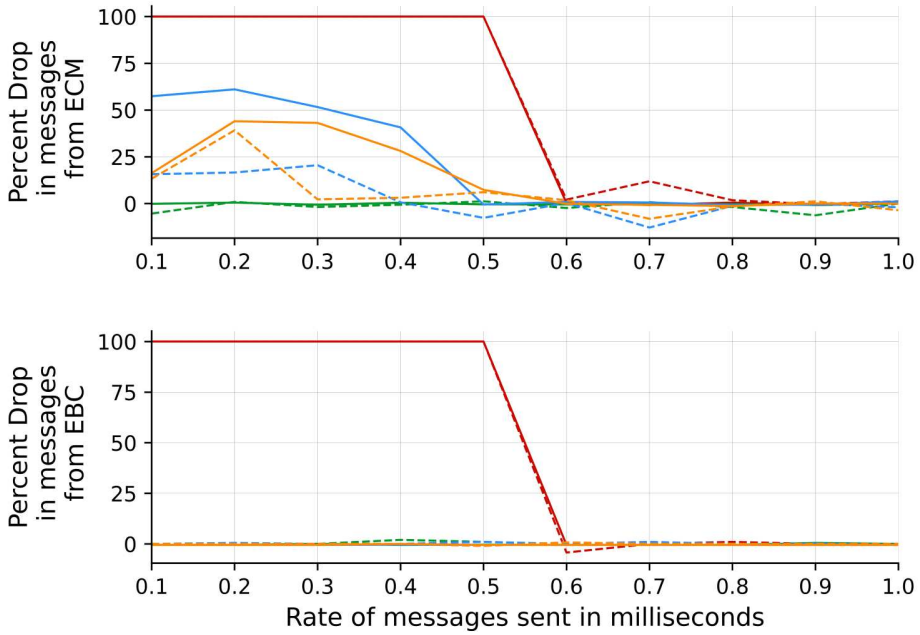

Figure 3.9: Request Overload Results on Local Testbed 3

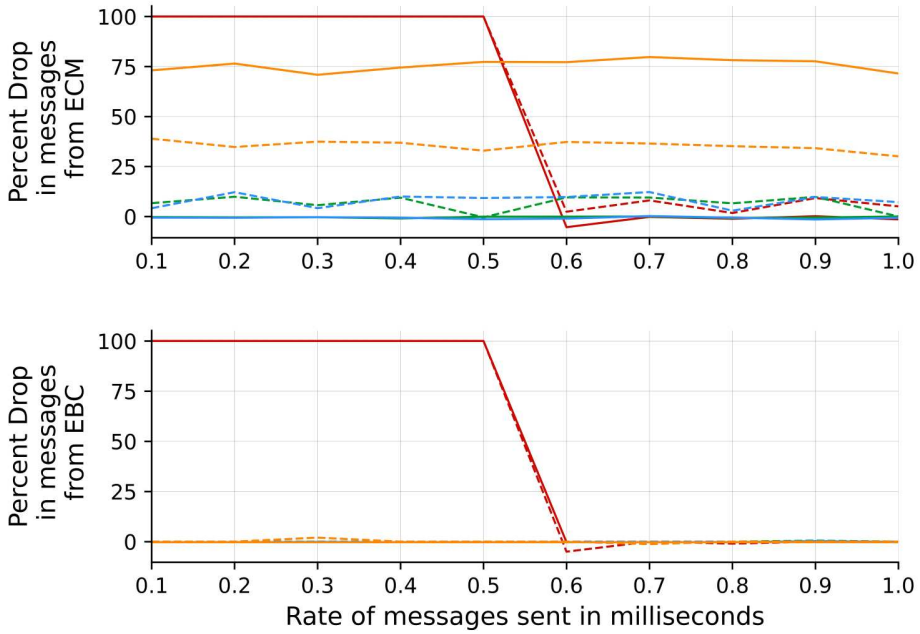

Figure 3.10: Request Overload Results on Local Testbed 4

Line color significance:
Red: On flooding with messages of ID 00000000_{16}
Blue: On overloading with valid request messages
Orange: On overload with invalid request messages
Green: On flooding with messages of ID $1C000000_{16}$
Line shape significance:
Solid: High priority ([0,3]) messages
Dashed: Low priority ([4,7]) messages

Figure 3.11: Request overload experimental results on different configurations of the local testbed

We conducted five experiments for each type of message transmitted at each interval mentioned in the previous paragraph. The average of the results from the five experiments is plotted. Note that, due to bus contention, CAN controller delay, frame collision, etc., the transmission interval on the CAN bus may not be equal to the exact intervals set for the experiments through the can-utils software.

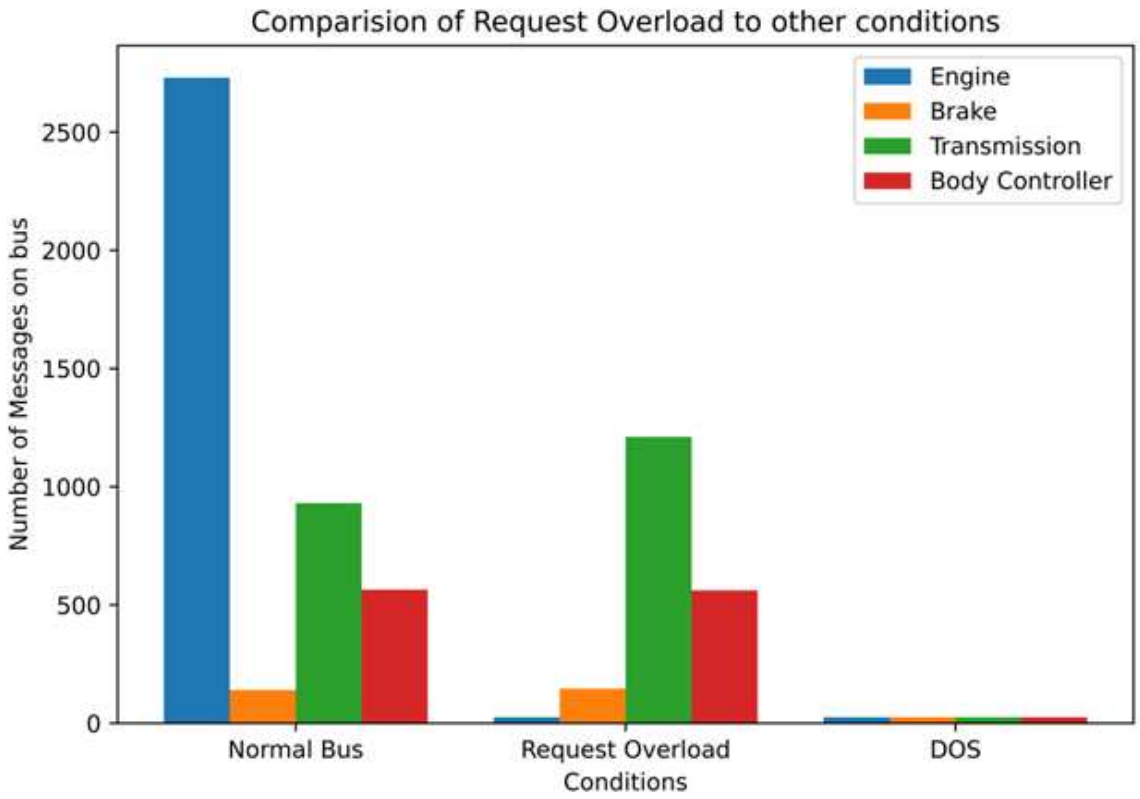

Figure 3.12: Comparison of the request overload attack on the Research Truck in other situations

Figure 3.13: Request overload attack on the Research Truck

3.2.3 Observations

As can be seen from Fig. 3.11, for all the test cases a drop in message count was observed from all sources when the network was flooded with messages with CAN ID `0x00000000`. Then again, in all the cases of flooding with messages of CAN ID `0x1C000000`, almost no normal traffic was removed. However, when the request overload attack was conducted, a certain percentage of normal messages transmitted by the ECM was removed. Albeit, the traffic volume from the brake controller remains constant during request overloads, thus indicating that it only affected the performance of the target. On Testbed 1, about 50 percent of both high and low-priority traffic were removed by valid request overloads up to 0.3 milliseconds, but invalid request overloads did not have any effect. On Testbed 2, all traffic transmitted by the ECM was removed by both valid and invalid request overloads up to 0.3 milliseconds. On Testbed 3, greater than 20 percent of both high and low-priority traffic were removed by valid and invalid request overloads up to 0.2 milliseconds. On Testbed 4, about 75 percent of high and 25 percent of low-priority traffic was removed by invalid request overloads up to a millisecond but valid requests had no effect. We noticed that it handled all valid requests promptly, initiating a multi-packet transfer in all cases. For invalid requests though, it transmitted a series of negative acknowledgments. 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.

We conducted this attack on the Kenworth T270 research truck. This truck had the same ECM as used in the local Testbed 2. As such, when a request overload attack was conducted at 0.3 milliseconds, all transmissions from the ECM stopped. The physical effect could be seen in Fig. 3.13 where the truck's dashboard displays erroneous information. Furthermore, the transmission did not shift gears and the engine speed remained high while moving forward. We also compared the effects of the request overload attack to normal bus conditions and a network flood attack(where the J1939 bus is flooded with high priority messages with CAN ID 00000000_{16}). As shown in Fig. 3.12, we can see that a network flood attack referred to as a denial of service (DOS) removes traffic from all ECUs at 0.3 milliseconds while the request overload attack on the engine controller removes all traffic only from the engine while other ECUs continue to broadcast data. This validates the request overload attack as a targeted denial of service attack.

3.2.4 Discussion

A potential solution to defend against this kind of attack is for ECUs to ignore request messages if they are received faster than a certain rate.

3.3 Connection Exhaustion

The second of the five attacks was named "Connection Exhaustion" as it exhausts the ability of the ECU to establish legitimate connections for multi-packet data transfer.

3.3.1 Hypothesis

According to the J1939-21 standards, there can only be one established connection for multi-packet transfer between a source ECU and a destination ECU at a time. It also states that after data has been transmitted a connection can be kept open for a maximum of 1250 milliseconds by not sending the end-of-message acknowledgment. In addition, a CTS message can be sent to request

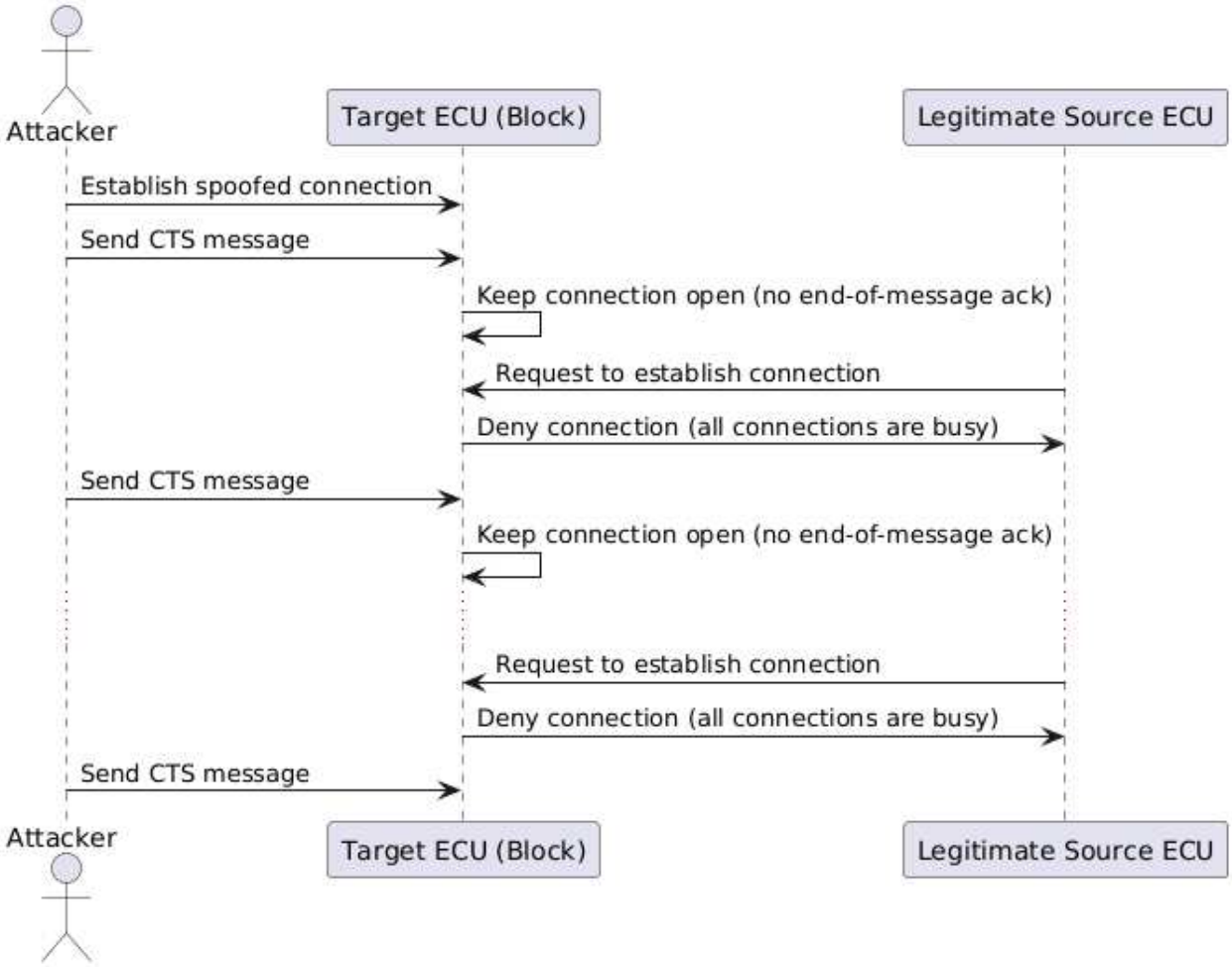

Figure 3.14: Connection Exhaustion Hypothesis

one or more packets that may have been sent already, but not received by the destination ECU. Using these three specifications, an attack can be crafted to deny legitimate connection attempts to an ECU by creating multiple spoofed connections and keeping them open periodically (typically for less than a second) sending a CTS message but not the end of message acknowledgment. Fig. 3.14 shows a sequence diagram depicting the hypothesis of the attack. It includes the attacker and the target ECU, along with a legitimate ECU.

3.3.2 Testing

The attack was carried out on the local testbed as well as on the research truck. A valid connection was established with the ECMs by sending requests and CTS packets from a spoofed source

address. After the data packets are transferred by the ECM, the connection is maintained by sending a periodic CTS message every second. After a while, an honest connection is attempted to be established from the same source address.

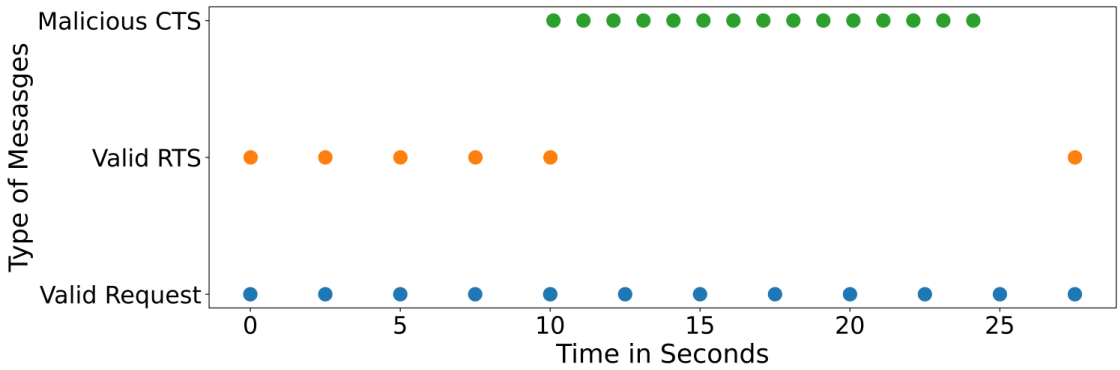

Figure 3.15: Connection Exhaustion Results on Local Testbed 1

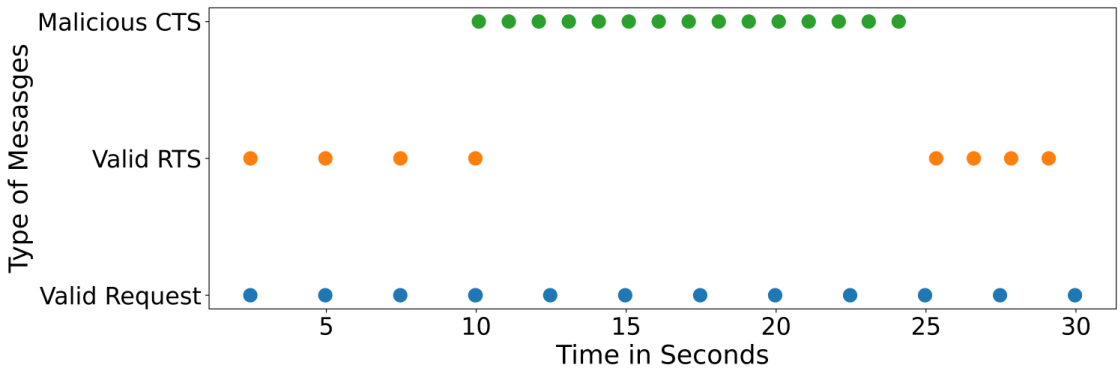

Figure 3.16: Connection Exhaustion Results on Local Testbed 2

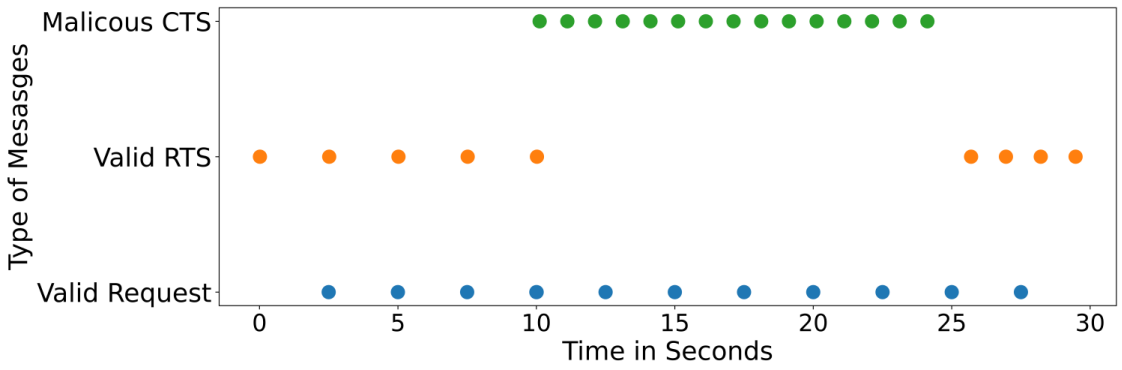

Figure 3.17: Connection Exhaustion Results on Local Testbed 3

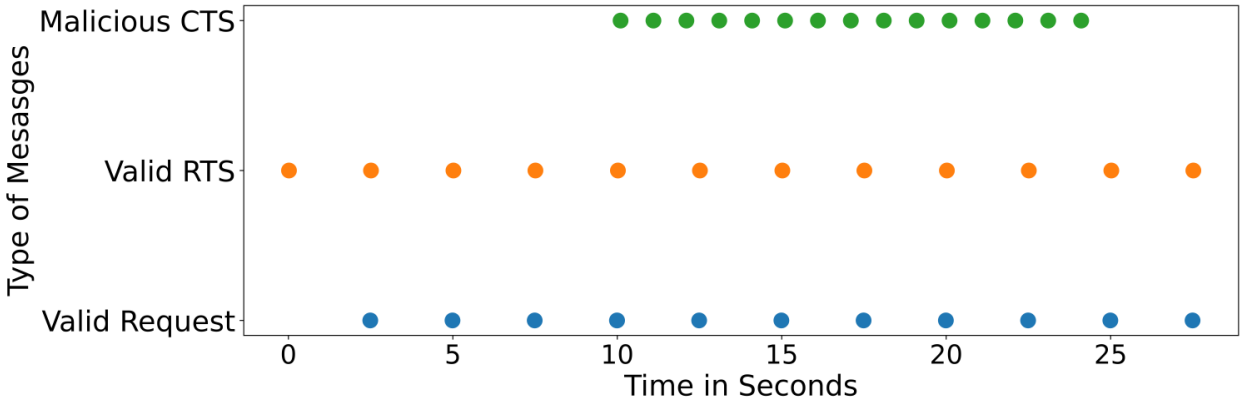

Figure 3.18: Connection Exhaustion Results on Local Testbed 4

3.3.3 Observations

Fig. **??** shows the "Connection Exhaustion" attack was demonstrated on three of four of our local testbench setups. The attack did not work on testbench 4. From Figs. 3.15, 3.16 and 3.17, we can observe a malicious CTS message was sent every second to keep the connection alive. A legitimate node on the network could not receive an RTS message for valid requests it sent. Thus, a legitimate connection could not be established as long the connection was maliciously occupied.

Even though the attack works on the Kenworth T270 truck, it did not have a physical impact. Albeit, a quick investigation of the SAE J1939 digital annex reveals that transport sessions are

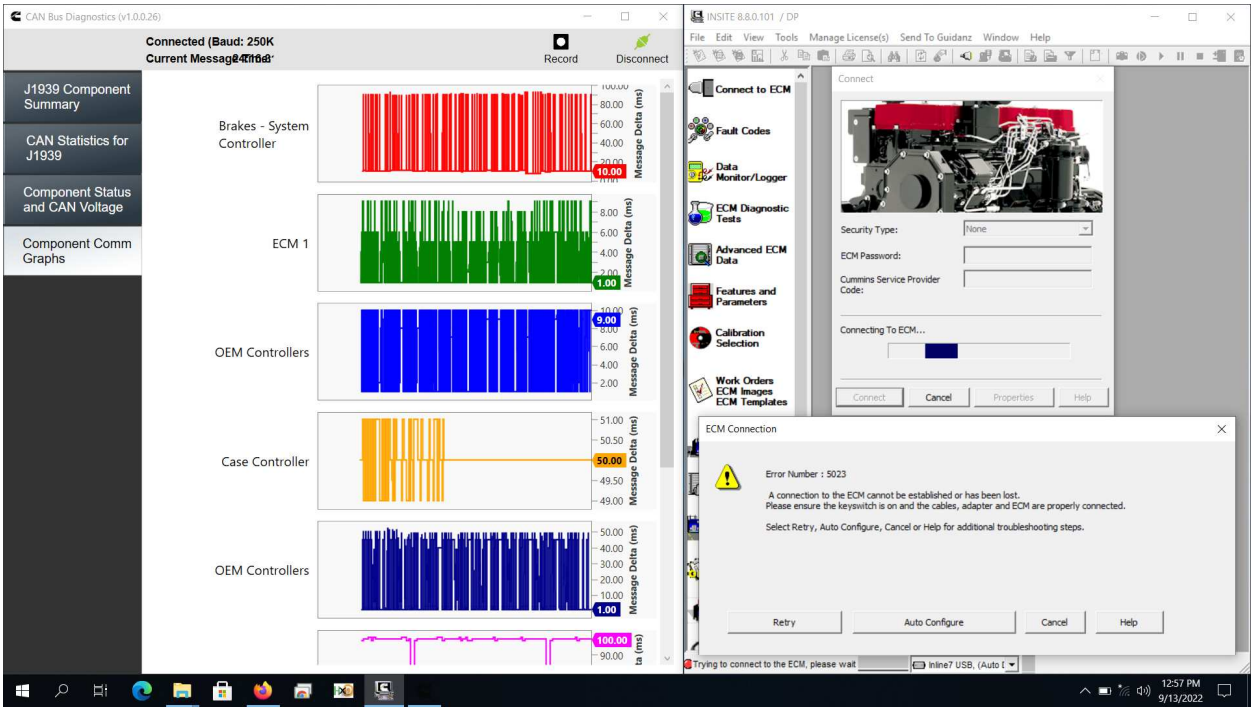

Figure 3.19: Connection Exhaustion Attack on Research Truck

critical for diagnostic and proprietary communication over SAE J1939. An example of hampering diagnostics is shown in Fig. 3.19 where the manufacturer diagnostics software tool failed to connect to the ECM. This can also be detrimental to the vehicle if data obtained from the session is used for control purposes. For example, PGN 65251 carries engine configuration information that may be required by more than one legitimate ECU. If this information is not received, those ECUs may malfunction.

3.3.4 Discussion

The apparent solution to this kind of attack is for ECUs to terminate open connections after a certain amount of time if connected ECUs are not exchanging data packets.

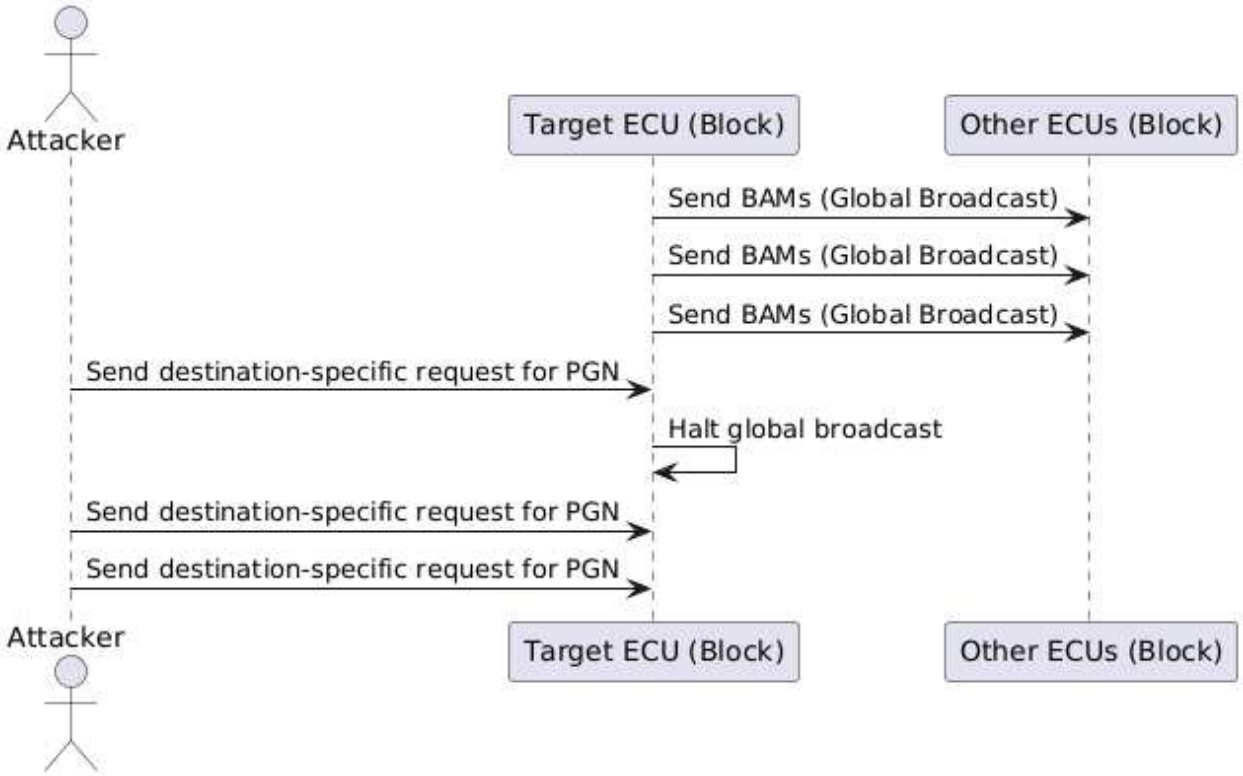

Figure 3.20: BAM Block Hypothesis

3.4 BAM Block

The third of the five attacks is coined the name "BAM Block" as it blocks broadcast announcement messages (BAMs) and subsequent data packets for multi-packet data transfer from a targeted ECU.

3.4.1 Hypothesis

ECUs periodically transmit BAMs which are multi-packet data frames to relay information globally to other ECUs on the network. The SAE J1939-21 standard suggests that an ECU must respond to destination-specific requests. Given this, an attack can be constructed whereby an attacker sends destination-specific requests for PGNs that an ECU broadcasts globally as BAMs with the expectation that this might force the ECU to respond to such a request and, in turn, the global broadcast communication halts denying information to all ECUs on the network. Fig. 3.20 shows a sequence diagram depicting the hypothesis of the attack.

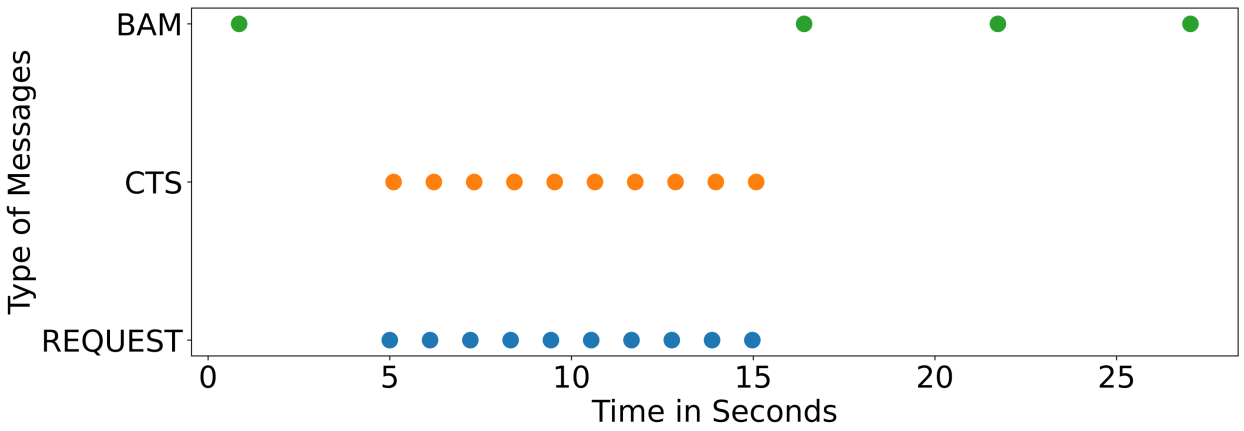

Figure 3.21: BAM Block Attack Demonstration

3.4.2 Testing

The attack was tested on the testbed setup. We wrote a bash script that sends a destination-specific request for a PGN the target ECM was broadcasting globally and a CTS message after successfully receiving an RTS message from the target. This process was iterated over a loop for the duration of the attack.

3.4.3 Observations

We noticed that this attack was only successful on Testbed 3 and did not have any impact on the other testbeds or the research truck. On execution of our script on Testbed 3, we observed that the Caterpillar ADEM3 ECU responded to the malicious destination-specific requests by redirecting data packets to the specific destination we requested the data. The ECU did not transmit any BAMs or multi-packet data globally as long as the malicious CTS were being sent by our attack script. This is shown in Fig. 3.21.

3.4.4 Discussion

The apparent mitigation against this attack is to keep transmitting BAMs for specific PGNs even if destination-specific requests for the same are received.

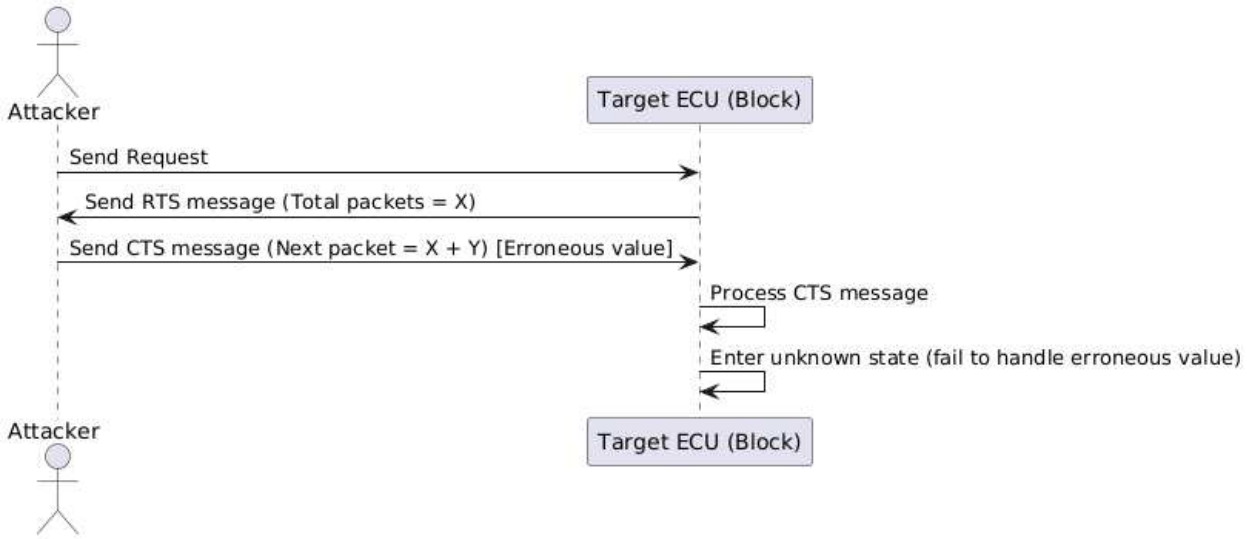

Figure 3.22: Malicious CTS Hypothesis

3.5 Malicious CTS

The fourth of the five attacks is coined the name "Malicious CTS" as it involved sending a malicious CTS packet to hinder J1939 transport protocol communication from a targeted ECU.

3.5.1 Hypothesis

The SAE J1939-21 document specifies that an RTS message communicates information about the total number of data packets that an ECU can send over multi-packet data transfer for a requested PGN. Additionally, it specifies that a CTS message should contain information indicating the packet number of the next data packet to be sent. If the value in the CTS message exceeds the total number of data packets in the RTS message for a multi-packet data transfer, it is possible that an ECU may not to be programmed to handle this erroneous information. Given this, an attack can be constructed to send a malicious CTS message with an erroneous value of the next packet to be sent that exceeds the total number of packets that can be sent indicated by the RTS message. This may cause the targeted ECU to enter an unknown state and thus hinder normal operations. Fig. 3.22 shows a sequence diagram depicting the hypothesis of the attack. =

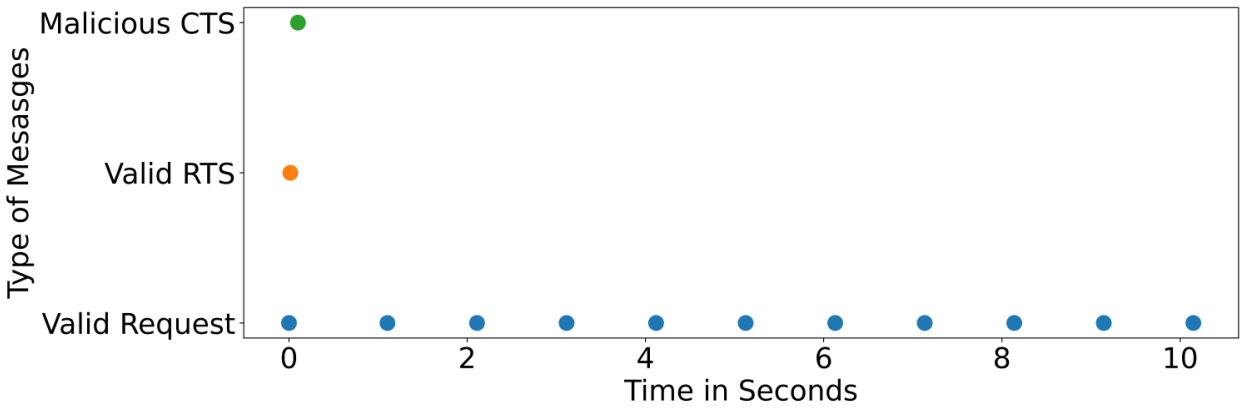

Figure 3.23: Malicious CTS Attack Demonstration

3.5.2 Testing

The attack was tested on the testbed setup. We wrote a shell script that sends a destination-specific request for a valid PGN to the target ECU and waits for an RTS. On receiving the RTS message a malicious CTS message is constructed with the value of the next packet to be sent in the CTS message exceeding the total number of packets indicated by the RTS message. This crafted CTS is then sent to the targeted ECU.

3.5.3 Observations

We noticed that this attack was only successful on Testbed 3 and did not have any impact on the other Testbeds or the research truck. On execution of our script on Testbed 3, we observed that the Caterpillar ADEM3 ECM stops responding to any further request messages as shown in Fig. 3.23. Additionally, all multi-packet communication from the ECM ceases until the ECM is rebooted.

3.5.4 Discussion

The apparent mitigation against this attack is to ensure invalid CTS messages as used in the attack are ignored.

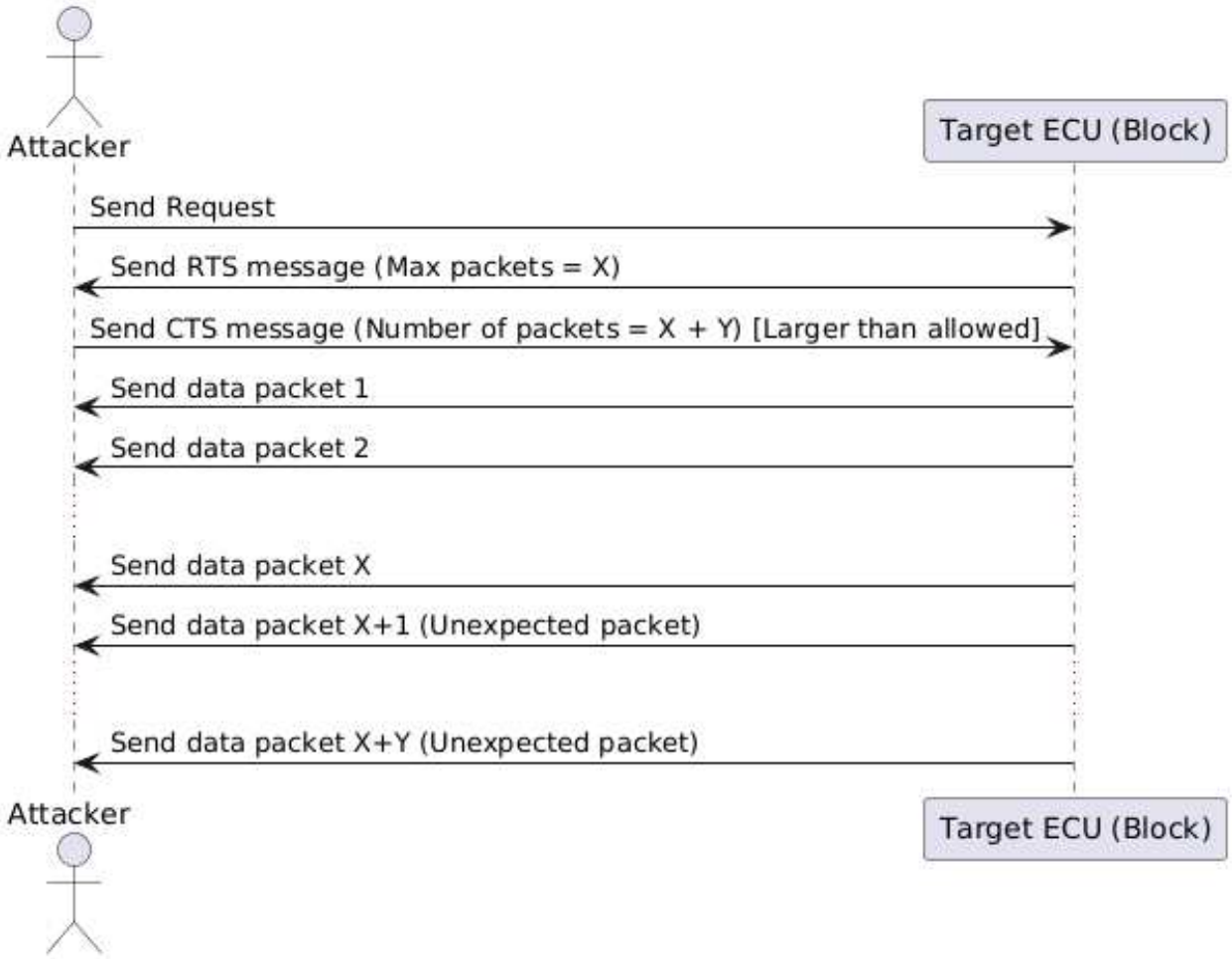

Figure 3.24: Memory Leak Hypothesis

3.6 Memory Leak

The final of the five attacks is coined as the "Memory Leak" attack as it leads to receiving data from an ECU that was not intended to be sent.

3.6.1 Hypothesis

The SAE J1939-21 document specifies that a CTS message should contain information about the number of packets that can be sent over multi-packet data transfer. In theory, this value should not exceed the maximum number of packets indicated by the RTS message. As such, an attack can be constructed by sending a crafted CTS message with the value of the number of packets that can be sent larger value indicated by the RTS message with the expectation this may cause the targeted

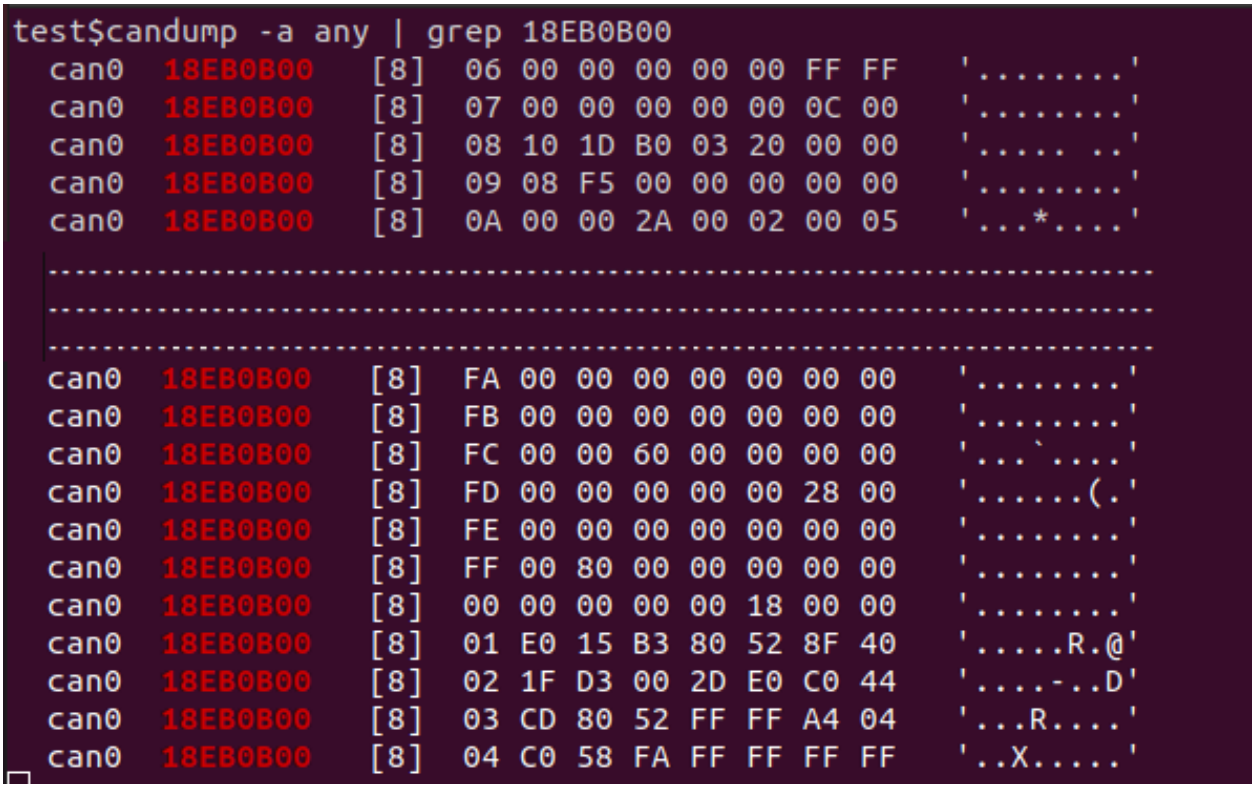

Figure 3.25: Memory Leak Attack Demonstration

ECU to leak data packets it never intended to send. Fig. 3.24 shows a sequence diagram depicting the hypothesis of the attack.

3.6.2 Testing

Similar to the Malicious CTS attack, we wrote a script that sends a destination-specific request to the target ECU and waits for a valid RTS. On receiving an RTS from the ECU, a CTS message is crafted with the number of packets to be sent set to a value much larger than indicated by the previous RTS message. This crafted CTS message was then sent to the targeted ECU.

3.6.3 Observations

We noticed that this attack was only successful on Testbed 3. As shown in Fig. 3.25, we look a dump of the J1939 traffic using the 'candump' feature of can-utils. The ASCII representation of the data was also obtained using the '-a' delimiter. The traffic was also filtered to show only multi-

packet data leaked from the ECU. This ECU was supposed to send on 6 packets for this specific transfer. Nevertheless, it leaked 255 data packets as requested in the CTS message. The attack did not succeed on the Kenworth T270 research truck carrying a different ECU. The contents of the leaked data has not been investigated as yet; it is reserved for future work.

3.6.4 Discussion

The apparent mitigation against this attack is to ensure invalid CTS messages, as used in the attack, are ignored. This may be implemented as part of the ECU firmware or as a centralized network security solution.

3.7 Summary

This chapter presents five different scenarios where ECUs on SAE J1939 networks are subjected to different types of attacks. First, two of the five scenarios demonstrate validations of attacks discovered in prior literature. The validation incorporates a more comprehensive testing setup. The latter three scenarios demonstrate new attack cases. Each of these attacks exploits specifications from the SAE J1939 protocol standards.

At its core, this chapter helps in enhancing the existing threatscape of vehicle security for medium and heavy-duty vehicles. Even so, a large part of the networking specifications still remains unexplored for security loopholes. In the future, we aim to investigate these opportunities, the focus still being medium and heavy vehicles. Additionally, we want to explore defense mechanisms to prevent these attacks using a centralized network-based solution.

Chapter 4

Vulnerabilities in Diagnostic Protocols

4.1 Testing Setup

4.1.1 Bench-top Testbeds

The local bench testbeds were set up in four distinct configurations as shown in Figs.4.1, 4.2, 4.3, 4.4 each involving different combinations of components. Each testbed had a target ECU for testing our vulnerabilities and another control ECU.

1. **Testbed 1:** This setup included a Bendix EC-80 Electronic Brake Controller (EBC) paired with a Detroit Diesel CPC 3 (Common Powertrain Controller), operating on a 250kbps CAN bus. The CPC 3 had a Controller Application (CA) with an address of 0 (`0x00`), while the brakes were assigned the address 11 (`0x0B`). The Bendix EC-80 EBC was the target ECU for this testbed.

2. **Testbed 2:** This configuration incorporated a Wabco Smarttrac system coupled with a CPC 3 EVO, operating on a faster 500kbps CAN bus. The controller addressing scheme remained consistent with the other testbeds, with the CPC 3 EVO having an address of 0 (`0x00`) and the EBC having an address of 11 (`0x0B`). The Wabco Smarttrac EBC was the target ECU for this testbed.

3. **Testbed 3:** This configuration was similar to **Testbed 1** and included a Bendix EC-80 Electronic Brake Controller (EBC) paired with a Detroit Diesel CPC 3 (Common Powertrain Controller), operating on a 250kbps CAN bus. The CPC 3 had a Controller Application (CA) with an address of 0 (`0x00`), while the brakes were assigned the address 11 (`0x0B`). The CPC 3 was the target ECU for this testbed.

4. **Testbed 3:** This configuration also featured a Bendix EC-80 EBC, but with a Detroit Diesel CPC 4, operating on a similar 250kbps bus. The addressing scheme was akin to Testbed

1, with the CPC having a CA with address 0 (`0x00`) and the EBC having an address of 11 (`0x0B`). The CPC 4 was the target ECU for this testbed.

Each testbed was equipped with a Linux laptop running SocketCAN and the 'can-utils' software for capturing and transmitting CAN messages. The laptops also served as points for initiating attacks. Additionally, each testbed included one or more power supplies. For diagnostic communication, a separate laptop equipped with Noregon DLA as the RP1210 compliant vehicle diagnostics adapter (VDA), interfacing with the ECUs in the testbed.

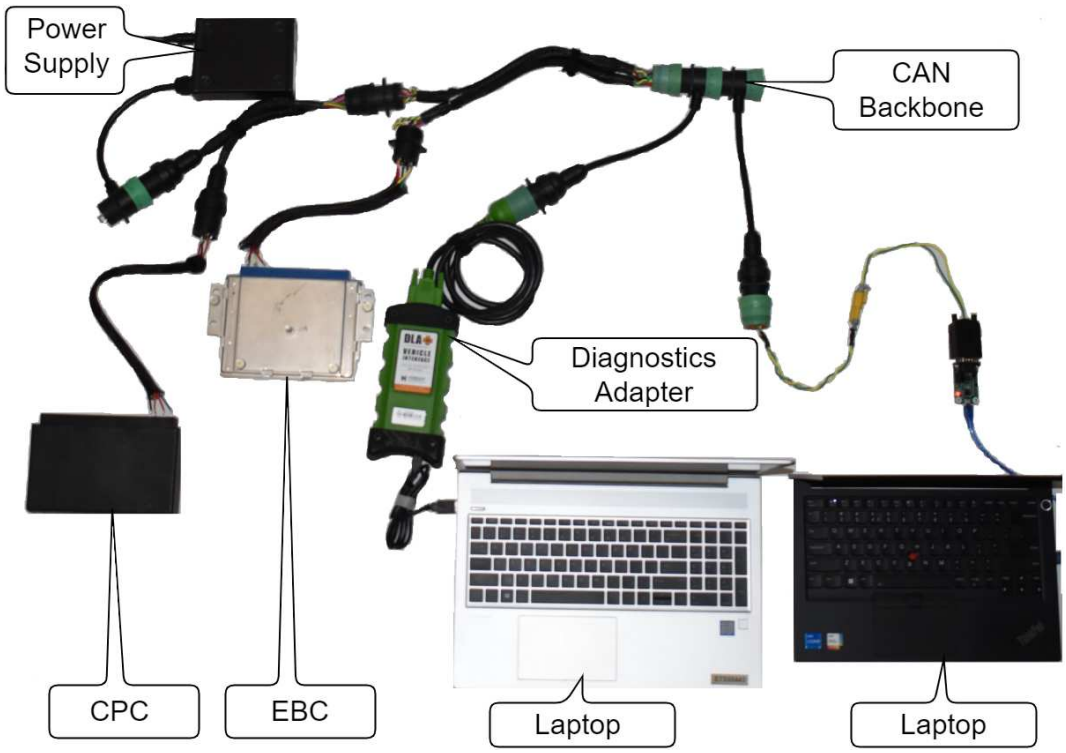

Figure 4.1: Testbed 1

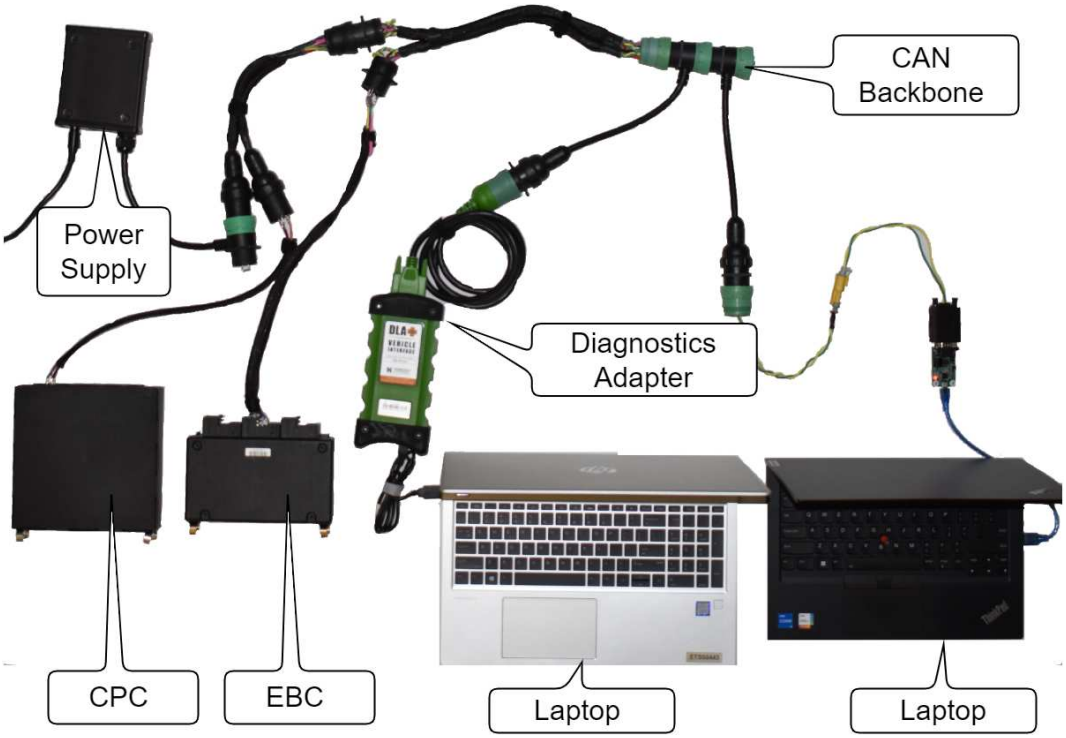

Figure 4.2: Testbed 2

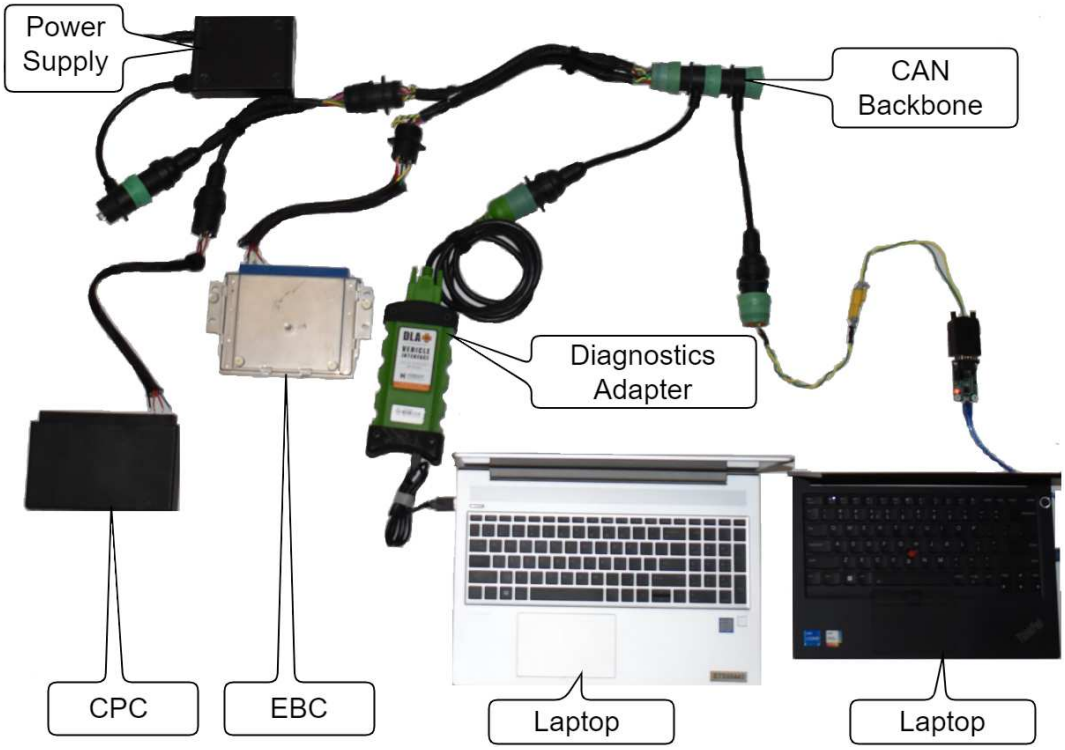

Figure 4.3: Testbed 3

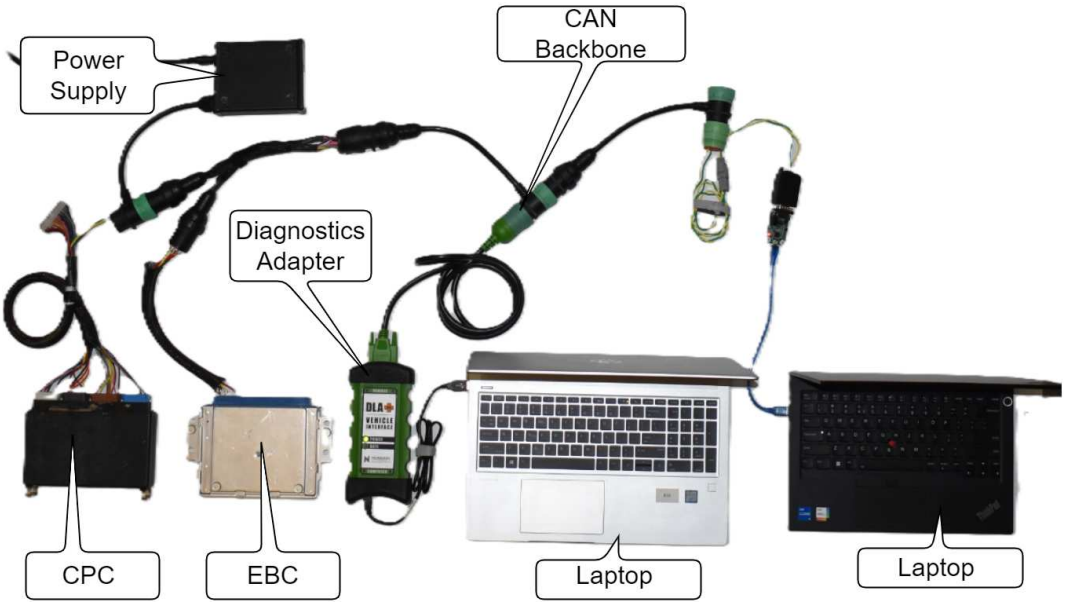

Figure 4.4: Testbed 4

Figure 4.5: Freightliner Cascadia Cab Testbed

4.1.2 Freightliner Cascadia Testbench

The Freightliner Cascadia testbed, shown in Fig. 4.5, specifically focusing on the front cab of a 2018 model, was utilized for additional demonstration of our findings. The configuration prominently featured two key components: a Wabco Smarttrac EBC and a Detroit Diesel CPC 3 EVO (Common Powertrain Controller) along with a Body Controller and a Cab Controller. However, their communication with external diagnostic tools and other networks was mediated by the Bosch gateway unit as shown in Fig. 4.6.

n this section, we describe the attack experiments conducted as part of our research. Each experiment is structured to include a research hypothesis, followed by detailed steps for hypothesis testing and an analysis of the results obtained. Finally, for each experiment, we discuss potential mitigation techniques that could be employed to counteract the identified vulnerabilities. It is important to note that all our tests were conducted from a black-box perspective, meaning that we did not have access to the source code or any runtime debug information of the systems being tested. This approach simulates the perspective of an external attacker with no insider knowledge, thereby ensuring the relevance of our findings to real-world scenarios.

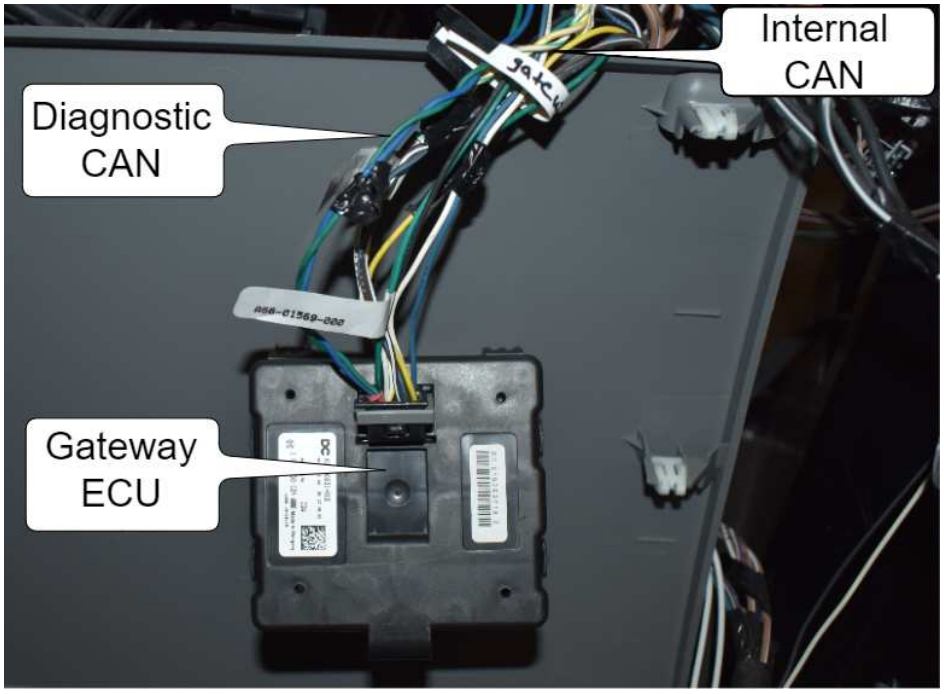

Figure 4.6: Gateway Unit in Cascadia Testbed

4.2 Read Data by ID Overload

Our first attack is the Read Data by ID Overload attack, in which an attacker sends a large number of Read Data by ID requests to a specific ECU, overwhelming it and causing a denial of service condition.

4.2.1 Hypothesis

The ISO 14229-1 document specifies that upon receiving a Read Data by Identifier request, the ECU shall access the data elements of the records specified by the data identifier and transmit their value. We hypothesize that sending a high volume of Read Data by Identifier requests may overwhelm the ECU and prevent it from carrying out more critical tasks such as the transmission of periodic messages. Fig. 4.7 shows a sequence diagram depicting the hypothesis of the attack. It includes the attacker and the target ECU.

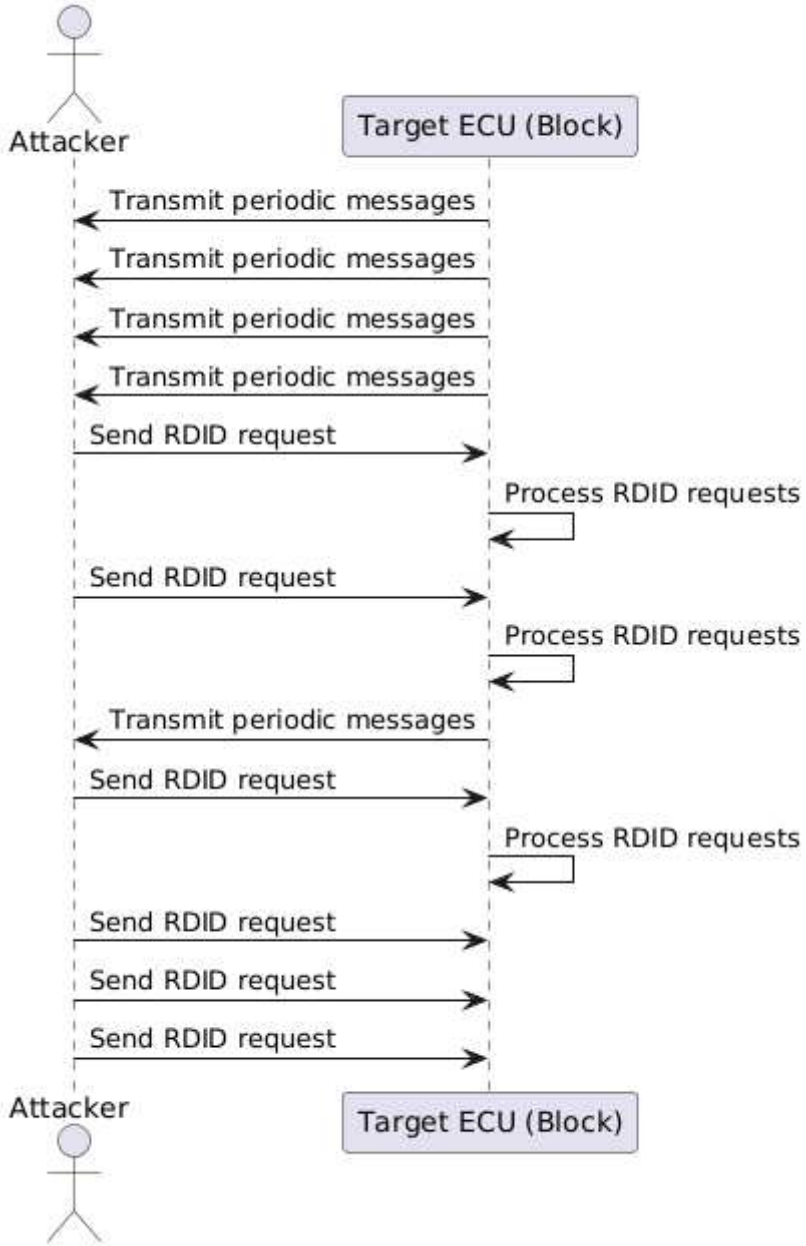

Figure 4.7: Read Data by ID Attack Hypothesis

4.2.2 Testing

The attack was tested out on the local testbeds and also on the Freightliner Cascadia cab testbed. Chatterjee et al. [19] in their demonstration of targeted denial of service attacks on MHD networks, have already shown that messages with high J1939 priority `0x00` (0) can flood the CAN network by winning transmission arbitration. Thus targeted denial of service can only be achieved by

sending low J1939 priority messages. In our experiments, we sent Read Data by Identifier (ID) requests with the lowest J1939 Priority `0x1C` (7) to the targeted ECU at varying intervals and observed any drop in periodic messages. By using a low priority, results are not conflated with arbitration mechanisms of CAN. The attack messages were sent at intervals of 0.1, 0.2, 0.3, 0.4, 0.5 and 0.6 milliseconds. Keep in mind, the messages may not end up on the network at these intervals due to CAN arbitration.

4.2.3 Observations

As can observed from Fig. 4.8, there was a drop in normal periodic messages from the targeted ECU on each testbed when the Read Data by ID attack messages were sent at 0.3 ms intervals. However, the normal traffic from the other ECU on the testbed remained constant. These results and the impact of Read Data by ID messages at different intervals are further detailed in Table 4.1. On the local testbed 1, where the Bendix EBC was the target ECU, normal traffic started dropping when the attack messages were sent at 0.4 ms intervals, dropping to 0 at intervals lower than 0.4 ms. On the local testbed 2, where the Wabco EBC was the target ECU, normal traffic started dropping when the attack messages were sent at 0.3 ms intervals. On the local testbed 3, where the CPC3 was the target ECU, normal traffic started dropping when the attack messages were sent at 0.5 ms intervals, dropping to 0 at intervals lower than 0.5 ms. Finally, on the local testbed 4, where the CPC4 was the target ECU, normal traffic started dropping to 0 when attack messages were sent at intervals of 0.5 ms or lower. Our observations on the Freightliner Cascadia cab testbed further validated our findings. We recorded the number of messages on the network in 1 second during normal conditions, during diagnostics sessions, and during the Read Data by ID Overload attack on the EBC. As can be observed from Fig. 4.9, the number of messages on the network from the EBC increased during a diagnostic session than during normal conditions as would be expected. However, during the attack, the number of messages from the EBC dropped, while the messages from other ECUs on the internal network remained constant. This validates the Read Data by ID Overload as a targeted denial of service attack.

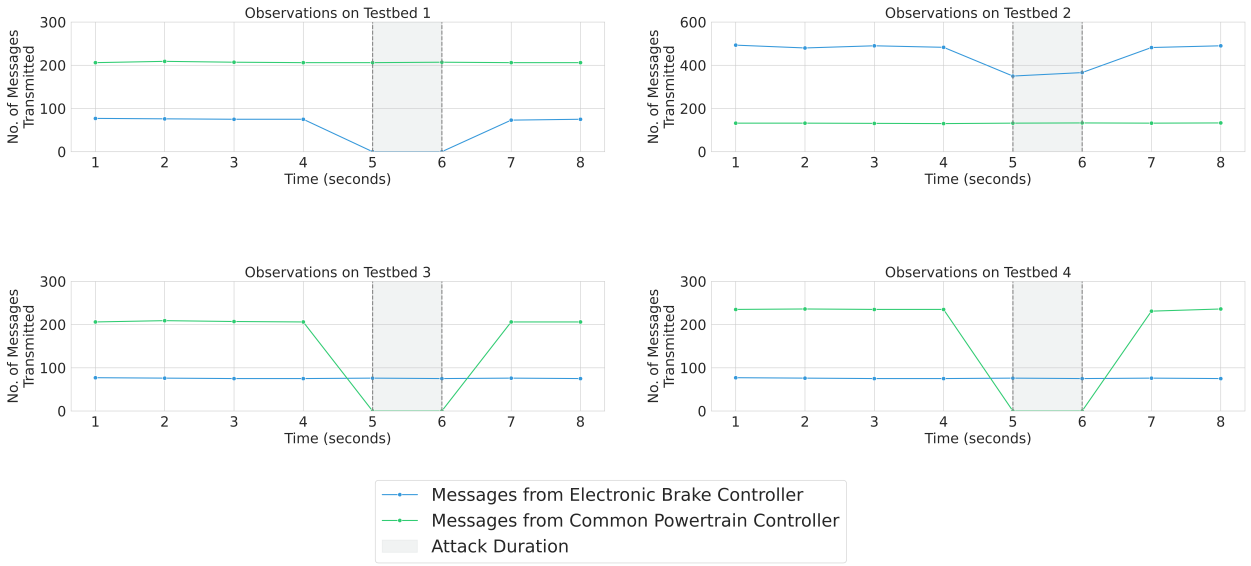

Figure 4.8: Read Data by ID Overload Attack at 0.3 ms Interval Attack Messages

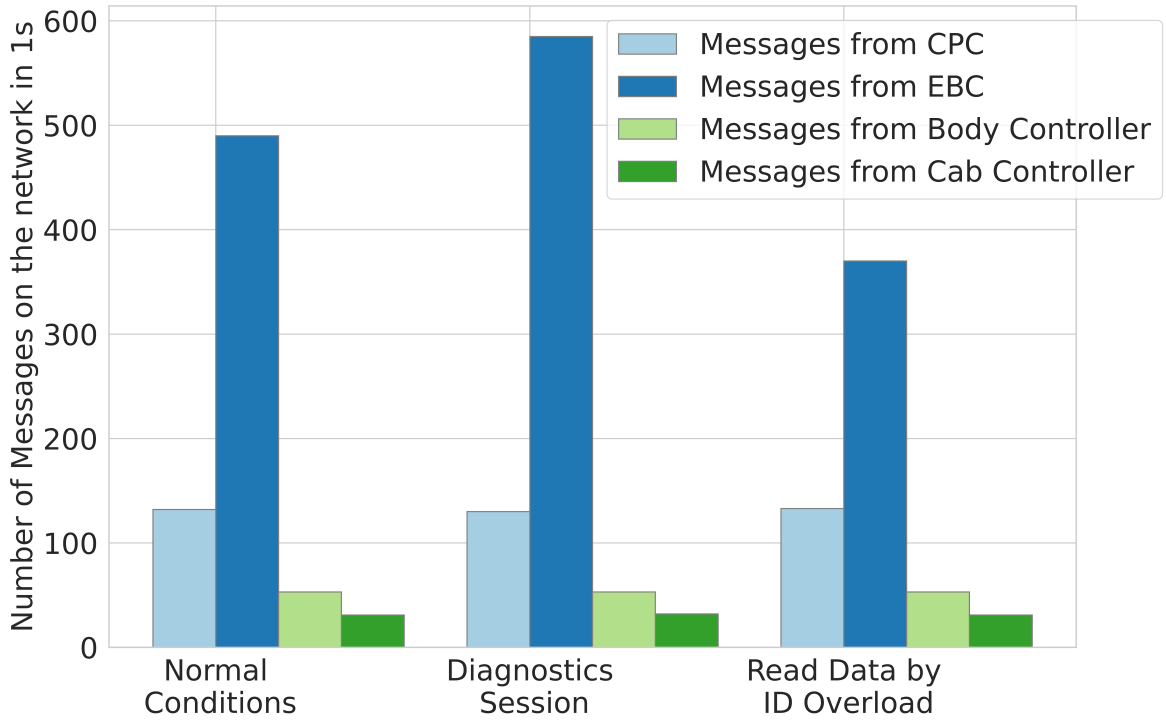

Figure 4.9: Read Data by ID on Cascadia Testbed

Table 4.1: Read Data by ID Effect on Normal Traffic

Attack Parameters			Average Message Count per ECU			
Testbed	**Target ECU**	**Message Interval (ms)**	**CPC**		**EBC**	
			Count	**% Decrease**	**Count**	**% Decrease**
Testbed 1	EBC	0.1	206	0%	0	100%
		0.2	206	0%	0	100%
		0.3	206	0%	0	100%
		0.4	206	0%	57	24%
		0.5	206	0%	75	0%
		0.6	206	0%	75	0%
Testbed 2	EBC	0.1	132	0%	275	44%
		0.2	132	0%	350	29%
		0.3	132	0%	350	29%
		0.4	132	0%	492	0%
		0.5	132	0%	492	0%
		0.6	132	0%	492	0%
Testbed 3	CPC	0.1	0	100%	75	0%
		0.2	0	100%	75	0%
		0.3	0	100%	75	0%
		0.4	0	100%	75	0%
		0.5	110	47%	75	0%
		0.6	207	0%	75	0%
Testbed 4	CPC	0.1	0	100%	75	0%
		0.2	0	100%	75	0%
		0.3	0	100%	75	0%
		0.4	0	100%	75	0%
		0.5	0	100%	75	0%
		0.6	235	0%	75	0%

4.2.4 Discussion

A potential solution to defend against this kind of attack is for ECUs to ignore Read Data by ID request messages if they are received faster than a certain rate. Additionally, if these requests are processed by an interrupt service routine, the normal processes may be subverted, thus showing a decrease in the message traffic.

4.3 Session Denial

In our second experiment, we explore the *Session Denial Vulnerability*, hypothesizing that Diagnostic Session Control messages, coupled with Tester Present signals, could lead an ECU to neglect other valid session requests. This scenario could potentially block diagnostic tools and software from successfully connecting to the ECU.

4.3.1 Hypothesis

The ISO 14229-1 standard specifies that there shall always be exactly one diagnostic session active in an ECU. We hypothesize that by establishing a session with an ECU by sending Diagnostics Session Control messages followed by Tester Present signals to keep the session alive, the ECU may ignore other valid Diagnostic Session Control requests. This could prevent diagnostic tools and software from establishing a connection to an ECU. Fig. 4.10 shows a sequence diagram depicting the hypothesis of the attack. It includes the attacker and the target ECU, along with a legitimate diagnostic tester.

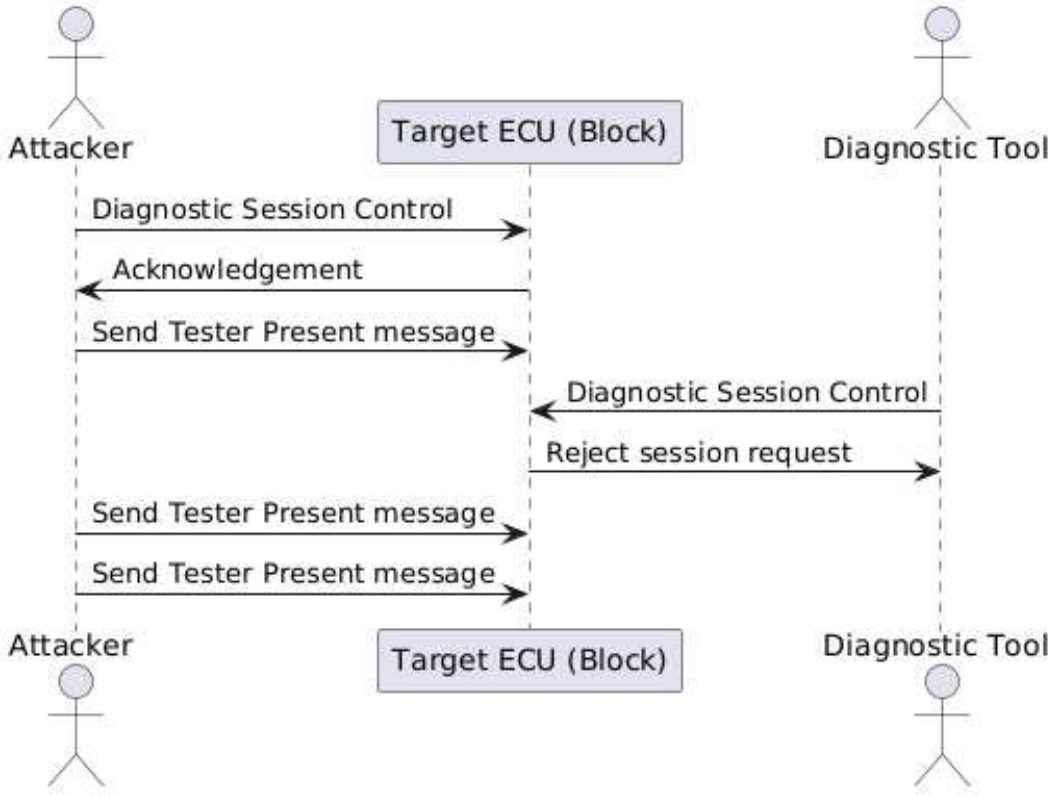

Figure 4.10: Session Denial Attack Hypothesis

4.3.2 Testing

The attack was carried out on both the local testbed and the Freightliner Cascadia cab testbed. A valid session was established with the target ECU using a Diagnostic Session Control Message from a spoofed source address, following which, the session was kept alive by sending Tester Present messages. During this time, attempts were made to establish a valid session using a diagnostic tool and the manufacturer's diagnostic software.

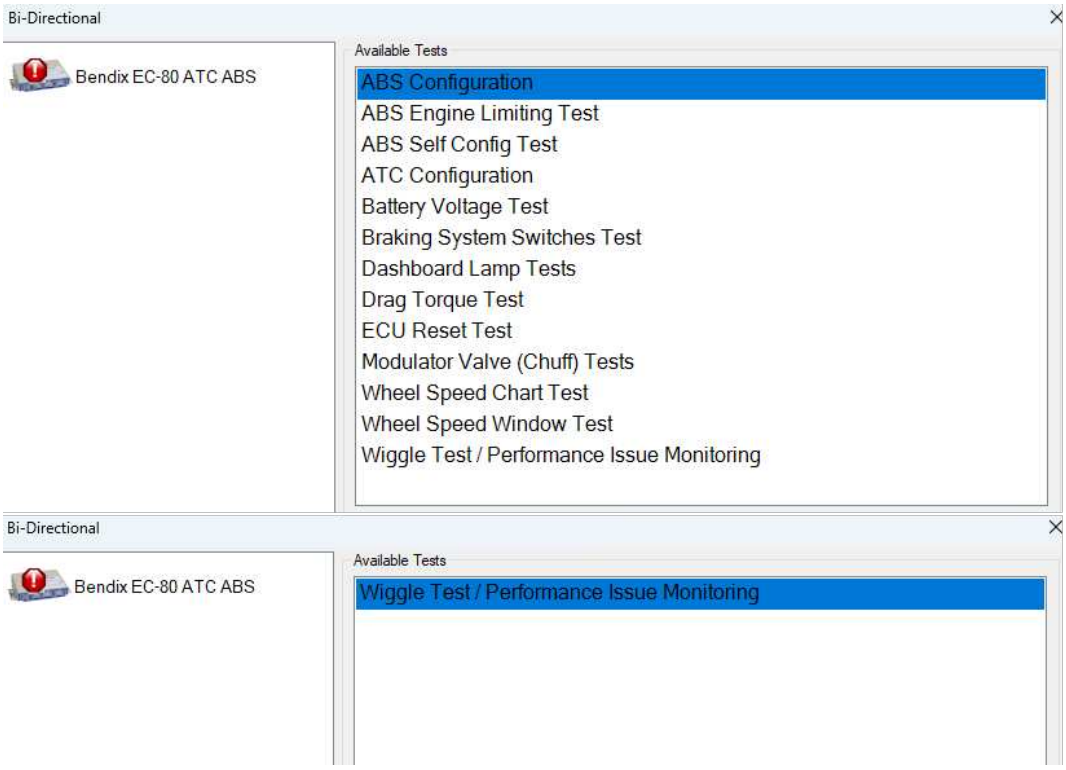

Figure 4.11: Session Denial Results on Local Testbed 1

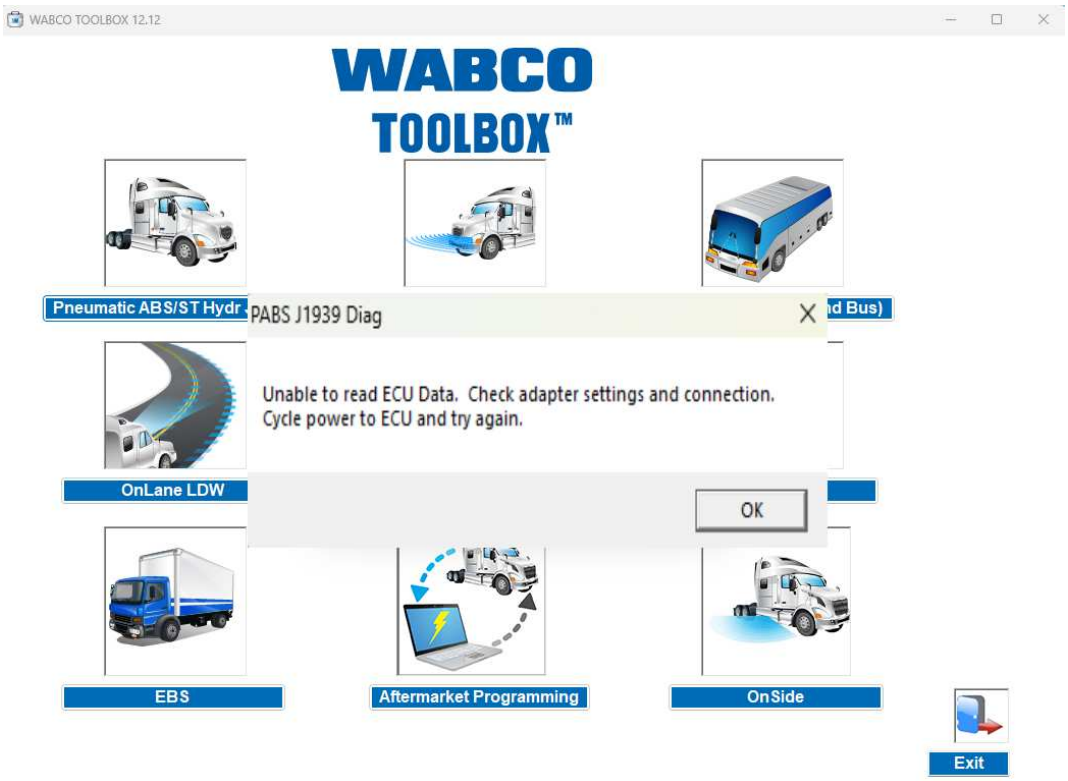

Figure 4.12: Session Denial Results on Local Testbed 2

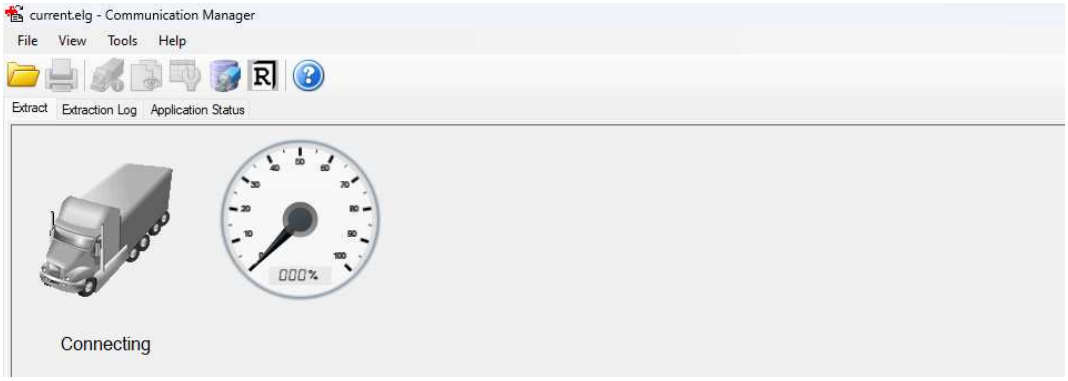

Figure 4.13: Session Denial Results on Local Testbed 3

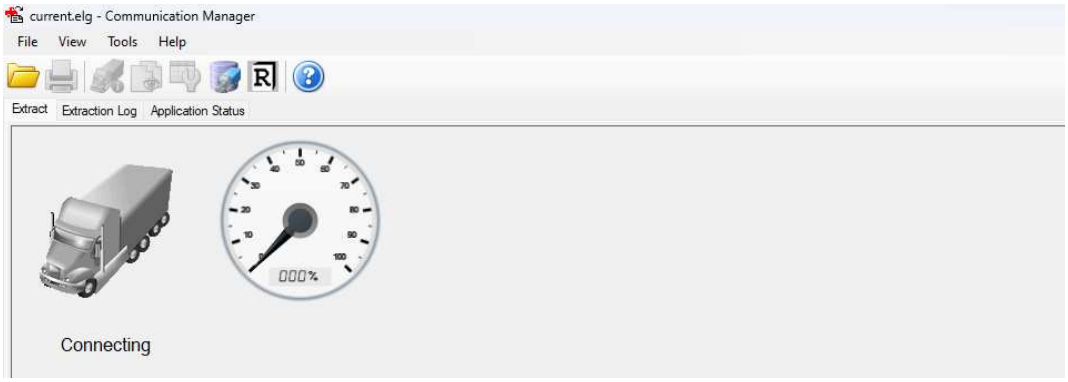

Figure 4.14: Session Denial Results on Local Testbed 4

4.3.3 Observations

As can observed, when the attacker was in an active session with the target ECU, the diagnostic software could not establish a successful UDS session with the target ECU. Fig. 4.11 shows the diagnostic tests available during normal conditions and during the attack on local testbed 1. The Bendix A-COM diagnostic software was unable to provide all available tests during the attack. Similar results were seen on local testbed 2 as seen in Fig. 4.12, where the Wabco Toolbox software was also unable to establish a diagnostic connection with the target ECU. Testbeds 3 and 4 showed the same results as seen in Fig. 4.13 and Fig. 4.14. The DDEC Reports diagnostic software could not connect to the CPCs. The experiment was also tested on the Freightliner Cascadia testbed and yielded similar results.

4.3.4 Discussion

Addressing the *Session Denial Vulnerability* identified in our experiments requires a layered security approach. One effective mitigation strategy is to implement a session request queue system that could allow the ECU to manage multiple session requests more efficiently, rather than being locked into a single session. To further enhance security, the introduction of source address validation for session requests can prevent unauthorized entities from establishing a session. This could involve authenticating diagnostic tools and software before allowing session initiation.

4.4 Diagnostic JAM

Our final experiment focuses on the *Diagnostics Jam Vulnerability*, where we test how sending a rapid mix of 'Wait' and 'Clear to Send' messages to an ECU can disrupt its normal operations and potentially lead to a service disruption.

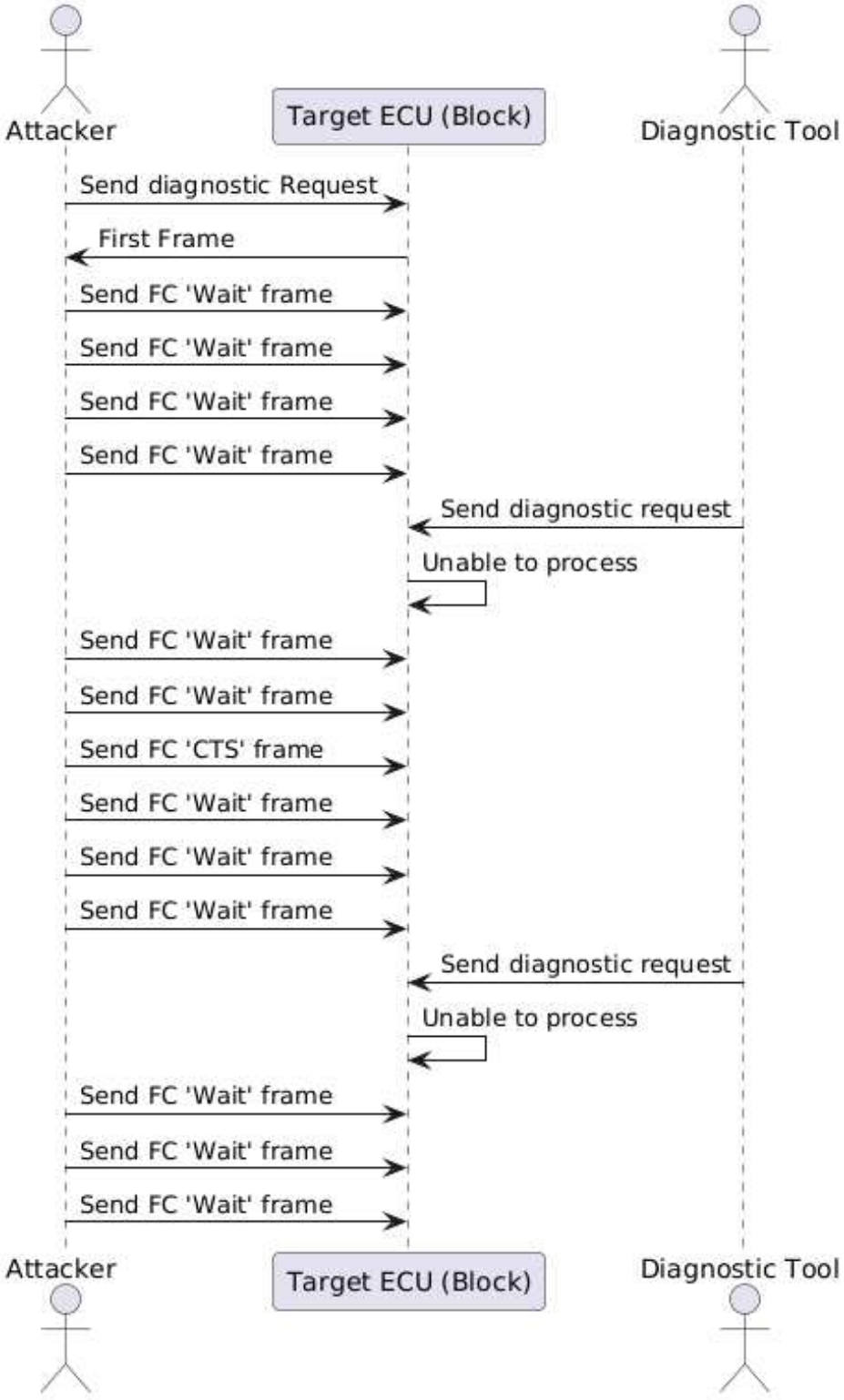

Figure 4.15: Diagnostics Jam Attack Hypothesis

4.4.1 Hypothesis

The ISO 15765-2 standard, commonly referred to as ISO-TP, facilitates communication and multi-packet data transfer over the transport protocol between an external diagnostic tester and a vehicle's Electronic Control Unit (ECU). As per the protocol specifications, Flow Control (FC) frames are used to manage the transmission of multi-frame messages, where 'Wait' frames indicate a pause in data transmission and 'Clear to Send' (CTS) frames signal the continuation of transmission. The protocol further specifies that an ECU shall keep track of the number of 'Wait' frames received in succession and terminate data transfer after it receives a certain number of 'Wait' FC frames in continuous succession specified by the manufacturer. However, this count is reset on receiving a CTS frame. Our hypothesis posits that a specific pattern of FC frames — repeatedly alternating between 'Wait' and 'CTS' within the maximum number of allowed 'Wait' frames — could exploit the ISO-TP's flow control mechanism. This may lead to a state where the ECU becomes overwhelmed and temporarily unable to process or respond to other diagnostic requests, effectively leading to a 'Diagnostics Jam' condition. Fig. 4.15 shows a sequence diagram depicting the hypothesis of the attack. It includes the attacker and the target ECU, along with a legitimate diagnostic tester.

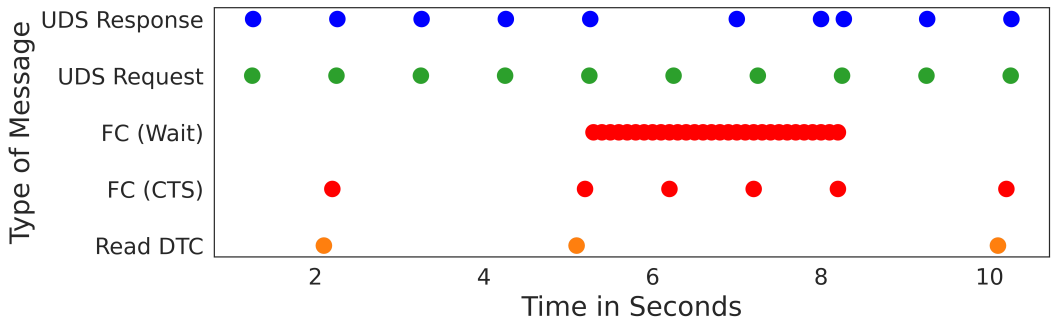

Figure 4.16: Diagnostics Jam Attack Results on Local Testbed 1

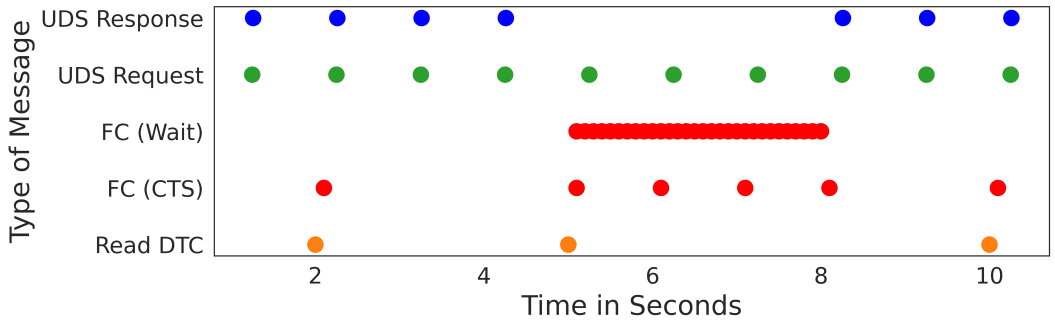

Figure 4.17: Diagnostics Jam Attack Results on Local Testbed 2

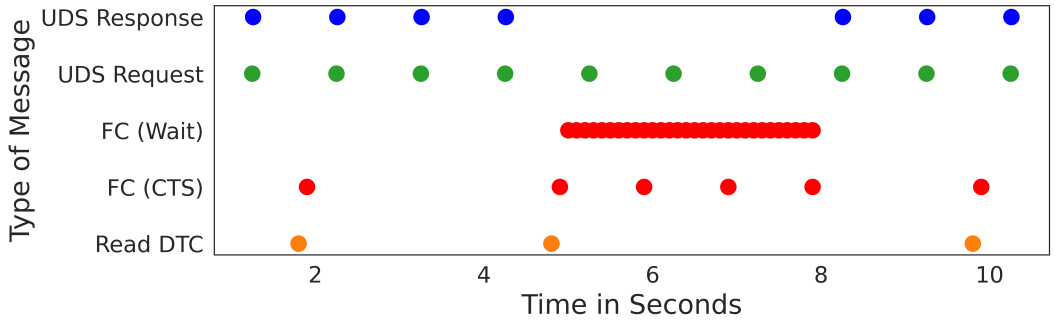

Figure 4.18: Diagnostics Jam Results on Local Testbed 3

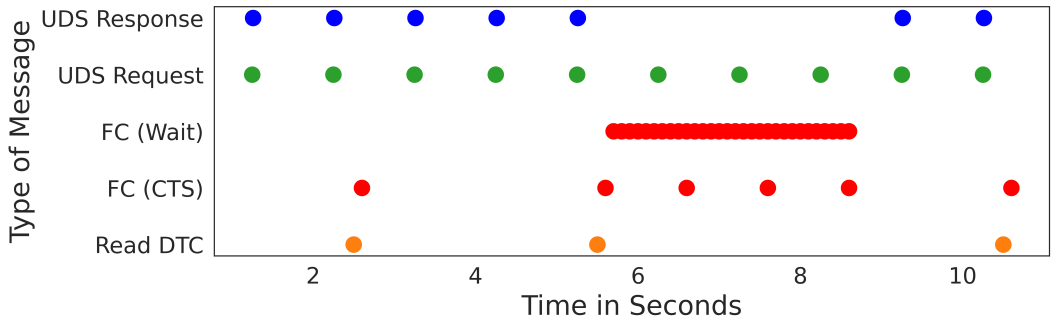

Figure 4.19: Diagnostics Jam Attack Results on Local Testbed 4

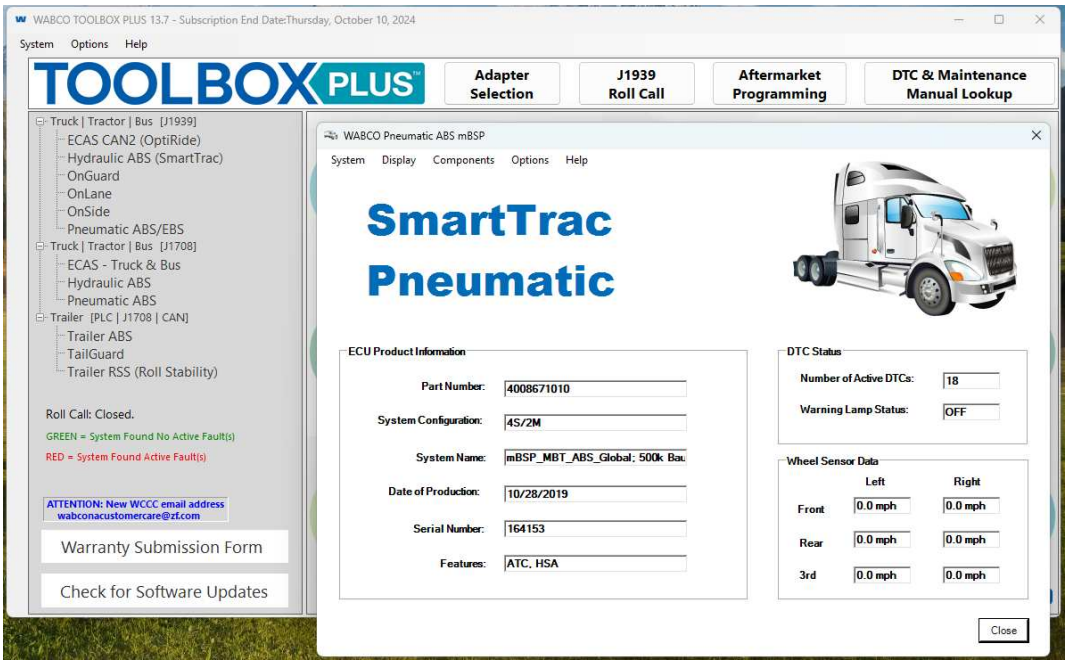

Figure 4.20: Diagnostics Software during Normal Conditions

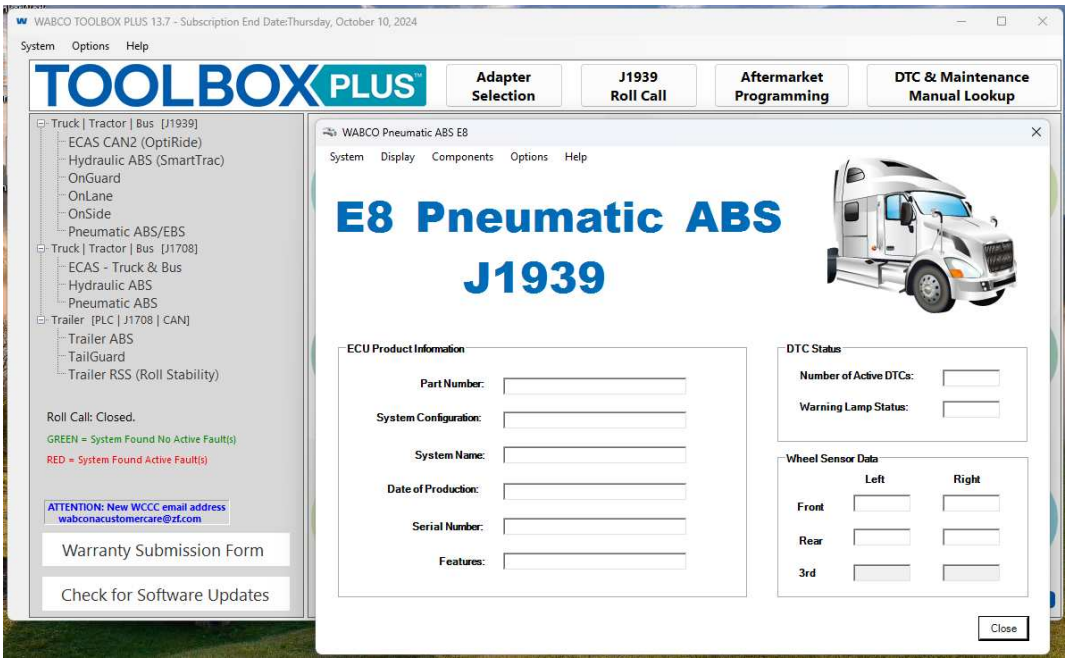

Figure 4.21: Diagnostics Software during Diagnostics Jam Attack

4.4.2 Testing

The attack was tested on both the local testbeds and the Freightliner Cascadia cab testbed. The testing process was set up as follows: A standard diagnostic tester consistently sent UDS requests in a routine manner. In parallel, our script was configured to request data for a Diagnostic Trouble Code (DTC) over the multipacket ISO-TP in a loop.

The script waited for the First Frame (FF) from the ECU. Once the FF was received, we sent a Flow Control (FC) message with a 'Clear to Send' (CTS) for data transfer. During the middle of this process, we strategically introduced a mix of 'CTS' and 'Wait' FC frames. Upon receiving a First Frame, we sent out a Flow Control (FC) message with a CTS for 1 packet followed by 10 FC frames with 'Wait' at intervals of 100 ms to prevent network flooding. This mixed sequence was maintained for a designated period before reverting back to the regular pattern of sending CTS frames only. The objective was to observe the interaction and potential impact of this mixed FC frame sequence on the ongoing UDS requests being handled by the ECU.

4.4.3 Observations

As observed from Figs. 4.16, 4.17, 4.18, 4.19 during our testing on local testbeds, the target Electronic Control Unit (ECU) displayed a standard behavior pattern under normal conditions, responding promptly to Unified Diagnostic Services (UDS) request messages during attempts to read Diagnostic Trouble Codes (DTC) over multi-packet data transfer. However, when the communication involved a mix of 'Clear to Send' (CTS) and 'Wait' frames, as per our hypothesized attack pattern, the response behavior of the ECU was absent.

During the attack, the target ECU reached a state where it became visibly overwhelmed. This was characterized by a significant delay in responding to UDS requests on local testbed 1, and a complete failure to respond on the other local testbeds. In essence, the ECU entered a 'Diagnostics Jam' condition, as hypothesized. The system's inability to process new diagnostic requests effectively rendered the diagnostic functionalities inoperative for the duration of the attack. Similar observations were made on the Freightliner Cascadia cab testbed. As can be seen in Fig. 4.20,

during normal conditions the diagnostic software was able to gather relevant information from the EBC, however during the attack as shown in Fig. 4.21, it failed to gather the same information.

4.4.4 Discussion

To mitigate the vulnerabilities identified in the ISO 15765-2 standard, particularly against the 'Diagnostics Jam' attack, a multifaceted approach is recommended. This should include adjusting protocol timeouts to counter abnormal flow control conditions, implementing anomaly detection algorithms to identify and react to suspicious patterns of Flow Control (FC) frames, and enhancing the ECU's processing capabilities to better handle high volumes of FC frames.

4.5 Summary

Our investigation into the Freightliner Cascadia's network systems included a detailed analysis of the gateway Electronic Control Unit (ECU). As part of our experimental setup, we transmitted a variety of diagnostic messages and attack messages to observe the gateway ECU's response. Specifically, we sent messages from the diagnostic CAN network and tracked their passage to the internal ECU network, which is crucial for understanding the results shown in Fig. 4.22.

The outcomes from these tests reveal a significant disparity in the gateway ECU's treatment of message types. It consistently permitted almost every diagnostic message, including those related to 'Read Data by ID', 'Diagnostic Session Control', and 'ISO-TP'. However, it effectively obstructed all standard J1939 Torque/Speed Command 1 (TSC1) attack messages which were shown by Burakova et al. [16] in their attack scenario.

This evidence aligns with the observations made in the preceding sections on 'Read Data by ID Vulnerability', 'Session Denial Vulnerability', and 'Diagnostics Jam Vulnerability'. Each of these vulnerabilities was successfully executed on the Freightliner Cascadia's system, impacting its ECUs.

This chapter presents three different scenarios where protocol vulnerabilities in unified diagnostics services (UDS) standards can be exploited to expose ECUs in Medium and Heavy Duty

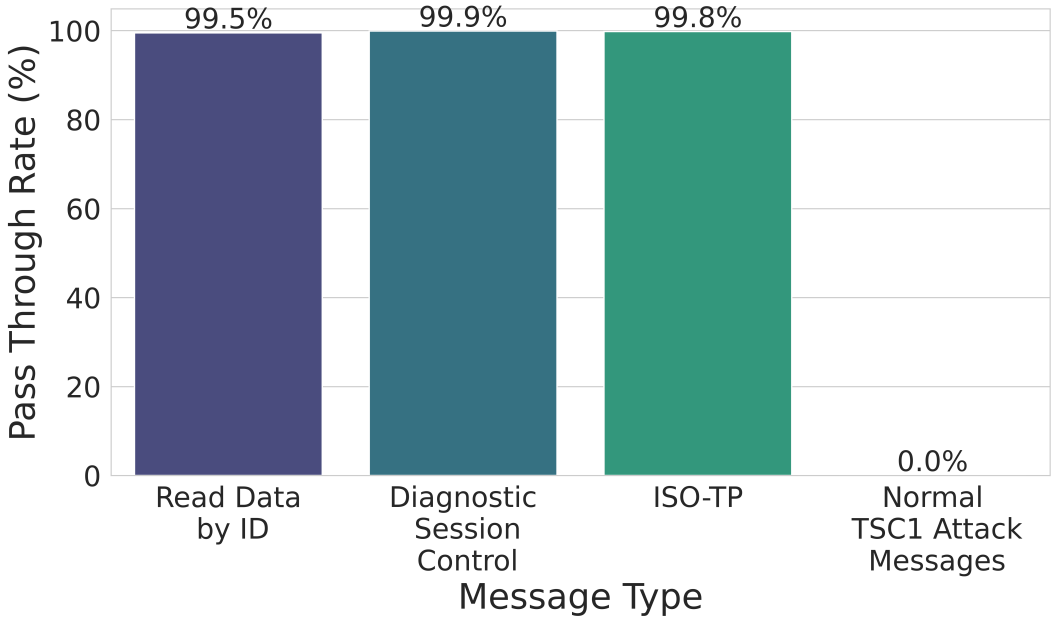

Figure 4.22: Behavior of the Gateway ECU in Filtering Different Message Types

Vehicles to different types of attacks. First, two of the three scenarios demonstrate novel vulnerabilities in the ISO 14229 standard. The final scenario demonstrates a new attack case exploiting the ISO 15765 (Diagnostic Communication over CAN) specifications.

At its core, this chapter helps in enhancing the existing threatscape of vehicle security for medium and heavy-duty vehicles. Security and functional testing should include these scenarios to catch potential logic issues in deployed components. A large part of the networking specifications remain unexplored for security loopholes and opportunities remain to investigate. Additionally, practical defense mechanisms to prevent these attacks are of keen interest.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This work presents a detailed exploration of eight distinct scenarios in which Electronic Control Units (ECUs) on SAE J1939 networks and Unified Diagnostic Services (UDS) protocols are susceptible to various cyber-attacks. These scenarios serve as critical examinations into the robustness of the ISO 14229 and ISO 15765 standards, alongside the broader SAE J1939 specifications, highlighting both existing and novel vulnerabilities within the system architectures of medium and heavy-duty vehicles.

The first two scenarios validated previously identified attacks through a more elaborate testing setup, underscoring the persistent relevance of known vulnerabilities in real-world vehicular networks. These validations are crucial as they not only reaffirm the need for continuous monitoring of known threats but also help in refining the defensive measures that protect against such exploits.

Subsequent scenarios introduced in this discourse unveil three new attack cases that exploit the intricacies of the SAE J1939 protocol, alongside three additional scenarios targeting the UDS standards. Each of these new scenarios was meticulously chosen to demonstrate the potential for unauthorized access and control, which could severely undermine the safety and functionality of vehicle operations. By exploring these vulnerabilities, this analysis sheds light on less charted aspects of protocol specifications, providing a fresh perspective on potential security loopholes.

Moreover, the findings from these investigations contribute significantly to the existing threatscape of vehicle security, particularly within the domains of medium and heavy-duty vehicles. The nuanced understanding of these vulnerabilities allows for a more informed approach to security, one that anticipates potential avenues of attack and preemptively addresses them. This proactive stance is vital in an era where vehicular technology is rapidly evolving and becoming increasingly interconnected.

Furthermore, the exploration of these scenarios underscores the urgent need for comprehensive security and functional testing of vehicle components. Such testing should not only seek to identify and mitigate known logical flaws but also remain vigilant against emerging threats that exploit new and existing protocol weaknesses.

In addressing these vulnerabilities, the research highlights the necessity for robust defense mechanisms. While current strategies largely focus on isolated solutions, there is a compelling argument for the development of centralized, network-based defensive architectures. These systems would offer a holistic safeguarding mechanism, enhancing the security posture of vehicle networks by integrating advanced detection and response capabilities.

In conclusion, this research significantly extends the understanding of vehicular network vulnerabilities, providing crucial insights that can inform the development of more secure systems for medium and heavy-duty vehicles. The continued exploration of these vulnerabilities not only enriches the academic and practical perspectives on vehicle cybersecurity but also plays a pivotal role in shaping the future of secure vehicular communications. As the landscape of vehicular technology progresses, the integration of robust, network-wide defensive strategies will be paramount in ensuring the safety and integrity of these critical systems.

5.2 Future Work

Future research based on the insights gained from this study is vital for advancing vehicle network security. One crucial area is the enhancement of protocols themselves. By employing static and dynamic analysis, researchers can delve deeper into the causes of vulnerabilities—static analysis allows for a thorough examination of codebase and protocol design flaws without execution, while dynamic analysis, including live code profiling and memory tracing during attack simulations, provides real-time insights into how attacks impact system operations. This dual approach can also extend to instruction-level tracing during simulated attacks to gain a deeper understanding of how unauthorized commands propagate within the system. Such insights could drive significant protocol redesigns that inherently minimize these risks.

Moreover, the development of advanced security solutions is essential. Improving network configuration and enhancing gateway security can serve as robust first lines of defense, managing traffic between different vehicular subnetworks more intelligently. Additionally, the creation of context-aware intrusion detection systems (IDPS) that not only detect anomalies but also understand the context of operations could revolutionize vehicle security. These systems could leverage techniques such as ECU clock fingerprinting to detect spoofed messages more effectively. As vehicle networks evolve towards CAN-FD and Automotive Ethernet, implementing stronger authentication measures, such as message authentication codes (MACs) and certificate-based authentication, will become increasingly critical. These measures will ensure a more secure communication environment as the complexity and connectivity of vehicle networks continue to increase.

The proposed future work aims to tackle these emerging challenges with a comprehensive approach, blending advanced analytical techniques with innovative security solutions to fortify the defenses of medium and heavy-duty vehicle networks against a growing range of cyber threats.

Bibliography

[1] Bureau of Transportation Statistics. Number of u.s. aircraft, vehicles, vessels, and other conveyances. https://www.bts.gov/, n.d. U.S. Department of Transportation.

[2] Society of Automotive Engineers. SAE J1939 Standards Collection.

[3] Information technology — open systems interconnection — basic reference model: The basic model, 1994.

[4] Robert Bosch GmbH. CAN Specification. Standard 2.0, Robert Bosch GmbH, 1991.

[5] International Organization for Standardization. Road vehicles — Unified Diagnostic Services (UDS) — Part 1: Application Layer. Standard ISO 14229-1, 2020.

[6] Road vehicles — unified diagnostic services (uds) — part 2: Session layer services, 2021.

[7] International Organization for Standardization . Road vehicles — Diagnostics Communication over Controller Area Networks (DoCAN) — Part 2: Transport and network layer services. Standard ISO 15765-2, 2016.

[8] Stephen Stachowski, Russ Bielawski, and André Weimerskirch. Cybersecurity Research Considerations for Heavy Vehicles. Technical report, University of Michigan, Ann Arbor, Transportation Research Institute, 2019.

[9] Marko Wolf and Robert Lambert. Hacking trucks – cybersecurity risks and effective cybersecurity protection for heavy-duty vehicles. In *Automotive - Safety & Security 2017 - Sicherheit und Zuverlässigkeit für automobile Informationstechnik*, pages 45–60, Stuttgart, Germany, 2017. Gesellschaft für Informatik, Bonn.

[10] Federal Motor Carrier Safety Administration. Electronic logging devices. https://www.fmcsa.dot.gov/hours-service/elds/electronic-logging-devices, n.d. Accessed: 2024-09-11.

[11] Marko Wolf, André Weimerskirch, and Christof Paar. Security in Automotive Bus Systems. In *Proceedings of the Workshop on Embedded Security in Cars*, pages 1–13, Bochum, Germany, 2004. Springer-Verlag.

[12] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security Symposium*, volume 4, pages 447–462, San Francisco, CA, USA, 2011. USENIX Association.

[13] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. In *Blackhat USA*, Las Vegas, NV, USA, 2015. Blackhat Press.

[14] Charlie Miller and Chris Valasek. A Survey of Remote Automotive Attack Surfaces. In *Black hat USA*, page 94, Las Vegas, NV, USA, 2014. Blackhat Press.

[15] Cesar Bernardini, Muhammad Rizwan Asghar, and Bruno Crispo. Security and privacy in vehicular communications: Challenges and opportunities. *Vehicular Communications*, 10:13–28, 2017.

[16] Yelizaveta Burakova, Bill Hass, Leif Millar, and Andre Weimerskirch. Truck Hacking: An Experimental Analysis of the SAE J1939 Standard. In *Proceedings of the 10th USENIX Conference on Offensive Technologies*, pages 211–220, Austin, TX, USA, 2016. USENIX Association.

[17] S. Mukherjee, H. Shirazi, I. Ray, J. Daily and R. Gamble. Practical DoS Attacks on Embedded Networks in Commercial Vehicles. In *Proceedings of 12th International Conference on Information Systems Security*, pages 23–42, 2016.

[18] P. Murvay and B. Groza. Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol. *IEEE Transactions on Vehicular Technology*, 67(5):4325–4339, 2018.

[19] Rik Chatterjee, Subhojeet Mukherjee, and Jeremy Daily. Exploiting transport protocol vulnerabilities in SAE J1939 networks. In *Proceedings of the Inaugural International Symposium on Vehicle Security & Privacy*, San Diego, CA, USA, 2023. Internet Society.

[20] Rik Chatterjee, Subhojeet Mukherjee, and Jeremy Daily. Transport layer vulnerabilities in the sae j1939 protocol-request overload. *Scholar Articles*, n.d. Available online.

[21] Rik Chatterjee, Ben Karel, Ricardo Baratto, Michael Gordon, and Jeremy Daily. Assured micropatching of race conditions in legacy real-time embedded systems. *Scholar Articles*, n.d. Available online.

[22] Rik Chatterjee, Carson Green, and Jeremy Daily. Exploiting diagnostic protocol vulnerabilities on embedded networks in commercial vehicles. In *Symposium on Vehicles Security and Privacy (VehicleSec)*, San Diego, CA, USA, 2024. VehicleSec Symposium.

[23] Jake Jepson, Rik Chatterjee, and Jeremy Daily. Commercial vehicle electronic logging device security: Unmasking the risk of truck-to-truck cyber worms. In *Symposium on Vehicles Security and Privacy (VehicleSec)*, San Diego, CA, USA, 2024. VehicleSec Symposium.

[24] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. In *IEEE Symposium on Security and Privacy*, pages 447–462, Oakland, CA, USA, 2010. IEEE.

[25] John Maag, Christopher Reding, and Kelly Howell. Seed-key security exchange, 2017. Presented at the 2017 Heavy Vehicle Cyber Security Workshop sponsored by the National Motor Freight Traffic Association, Inc.

[26] Sekar Kulandaivel. *Revisiting Remote Attack Kill-Chains on Modern In-Vehicle Networks*. Phd thesis, Carnegie Mellon University, 2021. Available at: https://kilthub.cmu.edu/ndownloader/files/34106900.

[27] Karen Scarfone and Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94, Naitonal Institute of Science and Technology, 2007.

[28] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom. Survey of Automotive Controller Area Network Intrusion Detection Systems. *IEEE Design Test*, 36(6):48–55, December 2019.

[29] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi. Survey on security threats and protection mechanisms in embedded automotive networks. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12, Budapest, Hungary, 2013. IEEE.

[30] S. Abbott-McCune and L. A. Shay. Intrusion prevention system of automotive network CAN bus. In *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 1–8, Orlando, FL, USA, 2016. IEEE.

[31] Dario Stabili, Mirco Marchetti, and Michele Colajanni. Detecting attacks to internal vehicle networks through Hamming distance. In *2017 AEIT International Annual Conference*, pages 1–6, Cagliari, Italy, 2017. IEEE.

[32] H. Giannopoulos, A. M. Wyglinski, and J. Chapman. Securing Vehicular Controller Area Networks: An Approach to Active Bus-Level Countermeasures. *IEEE Vehicular Technology Magazine*, 12(4):60–68, 2017.

[33] Kyong-Tak Cho and Kang G Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 911–927, Austin, TX, USA, 2016. USENIX Association.

[34] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks. *IEEE Transactions on Vehicular Technology*, 67(6):4757–4770, 2018.

[35] M. Gmiden, M. H. Gmiden, and H. Trabelsi. Cryptographic and Intrusion Detection System for automotive CAN bus: Survey and contributions. In *Proceedings of the 2019 16th International Multi-Conference on Systems, Signals Devices (SSD)*, pages 158–163, 2019.

[36] Ivan Studnia, Eric Alata, Vincent Nicomette, Mohamed Kaâniche, and Youssef Laarouchi. A language-based intrusion detection approach for automotive embedded networks. In *Proceedings of the IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, page 11, Zhangjiajie, Chin, 2015. IEEE.

[37] A. Taylor and N. Japkowicz and S. Leblanc. Frequency-Based anomaly detection for the automotive CAN bus. In *Proc. of WCICSS*, pages 45–49, 2015.

[38] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 130–139, Montreal, QC, Canada, 2016. IEEE.

[39] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. 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)*, pages 63–68, 2016.

[40] Michael R. Moore, Robert A. Bridges, Frank L. Combs, Michael S. Starr, and Stacy J. Prowell. Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection. In *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, pages 1–4, Oak Ridge Tennessee USA, 2017. ACM.

[41] André Weimerskirch, Steffen Becker, and Bill Hass. Commercial Vehicle vs. Automotive Cybersecurity – Commonalities & Differences. In *Cybersecurity for Commercial Vehicles*, pages 19 – 64. SAE International, August 2018.

[42] Yoshihiro Ujiie, Takeshi Kishikawa, Tomoyuki Haga, Hideki Matsushima, Tohru Wakabayashi, Masato Tanabe, Yoshihiko Kitamura, and Jun Anzai. A Method for Disabling Malicious CAN Messages by Using a CMI-ECU. In *SAE 2016 World Congress and Exhibition*, Detriot, Michigan, USA, 2016.

[43] Bogdan Groza and Pal-Stefan Murvay. Efficient Intrusion Detection With Bloom Filtering in Controller Area Networks. *IEEE Transactions on Information Forensics and Security*, 14(4):1037–1051, 2019.

[44] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. CaCAN - Centralized Authentication System in CAN (Controller Area Network). In *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014).*, page 10, Munich, Germany, 2014. ESCAR.

[45] Kyong-Tak Cho and Kang G. Shin. Viden: Attacker Identification on In-Vehicle Networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1109–1123, Dallas Texas USA, 2017. ACM.

[46] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. OBD_securealert: An Anomaly Detection System for Vehicles. In *Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6, St Louis, MO, USA, 2016. IEEE.

[47] Aymen Boudguiga, Witold Klaudel, Antoine Boulanger, and Pascal Chiron. A simple intrusion detection method for controller area network. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7, Kuala Lumpur, Malaysia, 2016. IEEE.

[48] Tsvika Dagan and Avishai Wool. Parrot, a software-only anti-spoofing defense system for the CAN bus. In *Embedded Security in Cars, EUROPE*, page 10, Munich, Germany,, 2016. ESCAR EUROPE.

[49] Mohammad Raashid Ansari, W. Thomas Miller, Chenghua She, and Qiaoyan Yu. A low-cost masquerade and replay attack detection method for CAN in automobiles. In *2017 IEEE*

International Symposium on Circuits and Systems (ISCAS), pages 1–4, Baltimore, MD, USA, 2017. IEEE.

[50] T. Matsumoto, M. Hata, M. Tanabe, K. Yoshioka, and K. Oishi. A Method of Preventing Unauthorized Data Transmission in Controller Area Network. In *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, pages 1–5, Yokohama, Japan, 2012. IEEE.

[51] Teri Lenard and Roland Bolboaca. A Statefull Firewall and Intrusion Detection System Enforced with Secure Logging for Controller Area Network. In *European Interdisciplinary Cybersecurity Conference*, pages 39–45, Virtual Event Romania, 2021. ACM.

[52] George Loukas, Eirini Karapistoli, Emmanouil Panaousis, Panagiotis Sarigiannidis, Anatolij Bezemskij, and Tuan Vuong. A taxonomy and survey of cyber-physical intrusion detection approaches for vehicles. *Ad Hoc Networks*, 84:124–147, 2019.

[53] Salvador Garcia, Julian Luengo, José Antonio Sáez, Victoria Lopez, and Francisco Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE transactions on Knowledge and Data Engineering*, 25(4):734–750, 2012.

[54] Matthew Timothy Campo, Subhojeet Mukherjee, and Jeremy Daily. Real-Time Network Defense of SAE J1939 Address Claim Attacks. *SAE International Journal of Commercial Vehicles*, 14(3):02–14–03–0026, August 2021.

[55] Chandrima Ghatak, Saira Jabeen, Hossein Shirazi, and Indrakshi Ray. Improving the resiliency of embedded networks in heavy vehicles-towards fault tolerance. In *Proceedings of the Ninth Annual Industrial Control System Security (ICSS) Workshop. Annual Computer Security Applications Conference (ACSAC)*. ACSAC, 2023.

[56] Chandrima Ghatak. Toward robust embedded networks in heavy vehicles-machine learning strategies for fault tolerance, 2024. Available online.

[57] U. E. Larson, D. K. Nilsson, and E. Jonsson. An approach to specification-based attack detection for in-vehicle networks. In *2008 IEEE Intelligent Vehicles Symposium*, pages 220–225, Eindhoven, Netherlands, 2008. IEEE.

[58] NXP. Secure TJA115x CAN Transceivers | NXP Semiconductors, 2021.

[59] Ulf E. Larson and Dennis K. Nilsson. Securing vehicles against cyber attacks. In *Proceedings of the 4th annual workshop on Cyber security and informaiton intelligence research developing strategies to meet the cyber security and information intelligence challenges ahead - CSIIRW '08*, page 1, Oak Ridge, Tennessee, 2008. ACM Press.

[60] Bureau of Transportation Statistics. Number of U.S. Aircraft, Vehicles, Vessels, and Other Conveyances.

[61] International Organization for Standardization. ISO - 43.040.15 - Car informatics. On board computer systems.

[62] Sibylle Fröschle and Alexander Stühring. Analyzing the Capabilities of the CAN Attacker. In *Computer Security – ESORICS 2017*, volume 10492, pages 464–482, Oslo, Norway, 2017. Springer International Publishing.

[63] Marko Wolf and Robert Lambert. Hacking Trucks – Cybersecurity Risks and Effective Cybersecurity Protection for Heavy-Duty Vehicles. In *Automotive - Safety & Security 2017 - Sicherheit und Zuverlässigkeit für automobile Informationstechnik*, pages 45–60, Stuttgart, Germany, 2017. Gesellschaft für Informatik, Bonn.

[64] SAE International. Vehicle Application Layer. Standard J1939-71, SAE International, 2015.

[65] Miki E. Verma, Michael D. Iannacone, Robert A. Bridges, Samuel C. Hollifield, Pablo Moriano, Bill Kay, and Frank L. Combs. Addressing the Lack of Comparability & Testing in CAN Intrusion Detection Research: A Comprehensive Guide to CAN IDS Data & Introduction of the ROAD Dataset. *arXiv:2012.14600 [cs]*, 2022.

[66] Jeremy Daily. The CyberTruck Research Vehicle.