Commercial Vehicle Electronic Logging Device Security: Unmasking the Risk of Truck-to-Truck Cyber Worms

Jake Jepson
Colorado State University
jepson2k@rams.colostate.edu

Rik Chatterjee
Colorado State University
rik.chatterjee@colostate.edu

Jeremy Daily
Colorado State University
jeremy.daily@colostate.edu

Abstract—**In compliance with U.S. regulations, modern commercial trucks are required by law to be equipped with Electronic Logging Devices (ELDs), which have become potential cybersecurity threat vectors. Our research uncovers three critical vulnerabilities in commonly used ELDs.**

First, we demonstrate that these devices can be wirelessly controlled to send arbitrary Controller Area Network (CAN) messages, enabling unauthorized control over vehicle systems. The second vulnerability demonstrates malicious firmware can be uploaded to these ELDs, allowing attackers to manipulate data and vehicle operations arbitrarily. The final vulnerability, and perhaps the most concerning, is the potential for a self-propagating truck-to-truck worm, which takes advantage of the inherent networked nature of these devices. Such an attack could lead to widespread disruptions in commercial fleets, with severe safety and operational implications. For the purpose of demonstration, bench level testing systems were utilized. Additional testing was conducted on a 2014 Kenworth T270 Class 6 research truck with a connected vulnerable ELD.

These findings highlight an urgent need to improve the security posture in ELD systems. Following some existing best practices and adhering to known requirements can greatly improve the security of these systems. The process of discovering the vulnerabilities and exploiting them is explained in detail. Product designers, programmers, engineers, and consumers should use this information to raise awareness of these vulnerabilities and encourage the development of safer devices that connect to vehicular networks.

I. Introduction

Commercial vehicles, specifically medium and heavy-duty trucks, serve as the backbone of supply chains and critical infrastructure globally. These trucks are not just vehicles, but pivotal cogs in the vast machinery that drives the world's economies. Their role extends far beyond simple transportation; they are integral in the distribution of a wide range of goods, from consumer products to industrial materials. This distribution network, supported by these commercial vehicles, is essential for maintaining the continuity and efficiency of global economic systems.

According to the US Bureau of Transportation Statistics, the United States alone has over 14 million medium and heavy-duty trucks registered, underscoring their prevalence and importance in national infrastructure [1]. Moreover, the American Trucking Association's report highlighted these trucks moved approximately 72.6% of the nation's freight by weight in recent years, showcasing their critical role in the country's freight transportation system [2]. This statistic further emphasizes the reliance of economies on these vehicles, not only for domestic transport but also for international trade and commerce. The seamless operation of these commercial vehicles is vital for the smooth functioning of supply chains, directly impacting everything from local businesses to international markets.

A. Background on Electronic Logging Devices (ELDs)

Many heavy vehicles are required to be equipped with Electronic Logging Devices (ELDs), since they are mandated by the Federal Motor Carrier Safety Administration (FMCSA) under the ELD Final Rule [3]. This so-called ELD Mandate is a component of the Moving Ahead for Progress in the 21st Century Act (MAP-21) and it went into effect December 18, 2017. These devices are essential for recording driving hours and ensuring compliance with Hours of Service (HOS) regulations, which are designed to prevent accidents due to driver fatigue. ELDs automate the process of capturing data on engine operation, vehicle movement, and miles driven, serving as a modernized alternative to traditional paper logbooks. The legal framework for ELDs requires self-certification and registration with the FMCSA. The regulation also includes provisions to protect drivers from harassment based on ELD data.

ELDs acquire data through communication with the vehicle's engine control module (ECM) via the vehicle newtork. Either the diagnostic connector or RP1226 connector are used as those interfaces expose the Controller Area Network (CAN) bus and the J1939 protocol. This setup enables ELDs to record mandated information such as engine hours, vehicle motion, and distance traveled, which are necessary for adherence to HOS regulations. Interestingly, the J1939 standard suggests the engine hours parameter is available **only** ”on request” over a J1939 network. This required parameter means the ELD

Symposium on Vehicles Security and Privacy (VehicleSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-7-6
https://dx.doi.org/10.14722/vehiclesec.2024.23047
www.ndss-symposium.org

must have some ability to write to the network and request the engine hours data.

The diagnostic connector, such as a the 9-pin connector defined in SAE J1939-13 or the J1962 connector, allows the ELD to interface with the vehicle's communication network. Additional connections defined by RP1226 provide dedicated access to third-party devices on the vehicle networks [4]. The CAN bus, known for its robustness, low latency and efficiency, facilitates real-time communication between modules [5], while the J1939 protocol, specific to heavy-duty vehicles, standardizes the physical construction of the CAN bus and the format of application layer messages [6].

SAE J1939 messages are organized into Parameter Groups (PG), each identified by a unique Parameter Group Number (PGN). These messages carry operational parameters and are encapsulated in the J1939 Protocol Data Unit (PDU), which also contains information like the source and destination addresses, message priority, and data payload.

However, neither CAN nor J1939 are renowned for their security features. While a comprehensive analysis of CAN and J1939 vulnerabilities is beyond the scope of this paper, they provide relevant context. Notable research includes Miller et al.'s identification of a network-level vulnerability leading to network overload, and Burakova et al.'s explanation of application-layer vulnerabilities capable of controlling truck engines and disabling critical functions [7], [8]. Murvay et al. and Campo et al. highlighted network management layer weaknesses, such as ECU isolation and denial-of-service vulnerabilities [9], [10], while Mukherjee et al. and Chatterjee et al. demonstrated data-link layer vulnerabilities, including ECU overloads and denial of service via open connections [11], [12].

In cybersecurity, a worm is defined as self-replicating malware that autonomously propagates across a network, exploiting security vulnerabilities or software flaws. Although worms have traditionally targeted standard computer network systems, the increasing connectivity of systems, including ELDs and vehicles, has created new attack surfaces for embedded systems [13], [14], [15].

B. Contributions

This research encompasses several significant contributions to the domain of cybersecurity, particularly in the context of Electronic Logging Devices (ELDs) and vehicular networks. These contributions are itemized as follows:

- **Reverse-Engineering Analysis of an ELD** Our study entails a detailed reverse-engineering process of a common off-the-shelf ELD. We thoroughly investigate its firmware update mechanism and delineate a potential method to compromise its firmware. This compromise could result in detrimental effects on both the vehicle and its immediate environment. The study also examines the implications of the malicious code on the device's standard operational performance.
- **Design and Demonstration of a Novel Truck-to-Truck Worm** We introduce the design of an innovative worm, specifically conceptualized for truck networks, which is capable of autonomous propagation from one ELD to another. This self-replicating worm is practically demonstrated using ESP32 development boards. The demonstration serves to validate the feasibility and operational mechanics of the worm in a controlled environment.
- **Empirical Assessment of the Compromised ELD and Truck Worm** The research includes an evaluation of the effects induced by both the compromised ELD and the prototype truck worm. This assessment is conducted through a series of experiments designed to demonstrate the impact of the worm in real-world scenarios, providing empirical evidence of its potential repercussions.
- **Discussion of Countermeasures and Defensive Strategies** Finally, the paper discusses a range of potential countermeasures and defense strategies aimed at mitigating the risks posed by the identified vulnerabilities. These strategies are proposed to enhance the security posture of ELDs against similar attacks in the future, thereby contributing to the broader field of vehicular cybersecurity.

C. Organization

Following the introduction, which sets the stage by discussing the significance of trucks in the U.S. supply chain and providing a background on Electronic Logging Devices (ELDs), the paper is structured into cohesive sections. The second section, "Understanding the System and Threats," delves into the intricate relationship between heavy trucks and ELDs, and introduces a detailed threat model for the Truck to Truck Worm. This sets the foundation for the subsequent section on "ELD Firmware Modification," where the focus shifts to the technical aspects. Here, the architecture of ELDs, the vulnerabilities uncovered, and the reverse engineering methods are thoroughly explored. This section also describes the creation and testing of malicious firmware, emphasizing its real-world implications.

Moving forward, the paper presents the "Design of a Truck to Truck Worm," elaborating on the development process and evaluation of such a worm, especially in the context of ESP32 development boards. Addressing the challenges posed, the next section, "Mitigations Against Worm Attacks," proposes various strategies to counter the identified vulnerabilities and potential threats. Concluding the paper, the final section synthesizes the key findings and posits future research directions, underscoring the necessity for continuous enhancement of ELD security within the trucking industry and its associated infrastructures. This structured approach ensures a comprehensive understanding of the vulnerabilities in ELDs and formulates strategies for strengthening these pivotal components in the logistics and transportation sector.

II. Understanding the System and Threats

A. System Definition

The system of interest is composed of two distinct systems that come together for use: a heavy truck and an electronic logging device. This distinction is important because it illustrates the interplay and interdependence within complex systems, which may exhibit emergent behaviors. In systems engineering, the integration of the heavy truck (the primary operational system) and the electronic logging device (ELD, a key data management subsystem) create a composed architecture. Each system by itself may not have cybersecurity concerns; the truck without an ELD does not have a wireless connection, and the ELD by itself cannot command a truck. The heavy truck acts as a dynamic operational platform, encompassing various mechanical and electronic components, while the ELD serves as a mandated interface for data logging, regulatory compliance, and potentially, vehicle control. This systemic integration, therefore, not only enhances operational capabilities but also introduces potential vulnerabilities and points of failure that can be exploited. The combination of a truck and ELD creates a system with an exploitable wireless connection that affects the operation of the truck.

B. Threat Models

Typical cyber-physical control oriented threat scenarios for ELDs or On-Board Diagnostic scanners (OBD-II scanners) involve either close proximity wireless access (e.g. via Wi-Fi or Bluetooth) or long-range access via cellular networks. In both scenarios, attackers exploit security flaws to launch attacks on vehicles, using the attached device as a proxy to the vehicle's internal CAN bus [16], [17]. This paper introduces a novel approach for a diagnostic plug-based attacks, capable of spreading to other devices encountered along the driver's route.

From an attacker's perspective, a widespread worm-like attack necessitates the compromise of numerous ELDs from multiple companies to inflict substantial damage. In addition, since the late 2010s, vehicle manufacturers have integrated vehicle network gateways as a countermeasure to such exploits. These gateways, analogous to firewalls, obstruct malicious traffic while permitting legitimate communications. Furthermore, an ELD's disconnection from the vehicle by the trucker can promptly impede the attack.

Among the approximately 880 ELDs registered [18], our analysis reveals a much narrower diversity than initially apparent, with only a few tens of distinct ELD models actually in use. This limited variety is primarily due to a high degree of rebranding, where many ELDs are essentially clones of each other with minimal variations, sharing similar form factors, execution environments and codebases. Consequently, the differences between models, often limited to firmware variations, do not significantly hinder the portability of malicious firmware across devices from the same manufacturer. In addition, recent studies have identified attacks capable of bypassing vehicle gateways by exploiting flaws in protocols permitted through these gateways [12]. Additionally, the legal requirement for drivers to use ELDs reduces the likelihood of device removal, except during diagnosis by a technician. Typically, a technician disconnects the ELD to access the diagnostic port to scan for codes; however, we posit that this may obscure the diagnosis of issues originating from the ELD.

As an illustrative proof of concept, we introduce a malicious variant of the firmware for a widely used consumer-grade ELD, demonstrating its potential to initiate an attack against the vehicle. We abstain from constructing and publishing a comprehensive proof of concept for the worm-like capabilities. Instead, we shift our focus to the developer board of the same ESP32 platform. Utilizing these boards, we will exhibit and assess a proof of concept for a malicious firmware worm that propagates from one device to another.

The attacker initially requires the compromise of at least one device, achievable through a drive-by attack or by positioning at locations frequented by trucks, such as truck stops, rest stops, hubs, distribution centers, or ports. The attacker wirelessly connects to the device and uploads the malicious firmware. Once re-flashed, the device commences scanning for similar devices through the unutilized interface (either WiFi or Bluetooth Low Energy). Upon detecting another device, it attempts to connect using default credentials if WiFi is inactive [1], or without credentials for Bluetooth. Success leads to an over-the-air (OTA) firmware upgrade, transferring the malicious firmware to the subsequent device. There is a brief interruption in the other device's service before it resumes its standard functions and concurrently seeks additional devices for propagation. After spreading to a predetermined number of devices (X), set by the attacker, the device waits for a specific set of conditions (Y), akin to a logic bomb, to execute an attack (Z).

While our emphasis is not on devising the most destructive attack sequence, we offer a potential scenario for illustrative purposes: The device initially waits until it has propagated to multiple devices or a certain amount of time has elapsed. This stage ensures broader dissemination of the worm before executing an attack, which could otherwise impair the ELD and limit its infectious potential. Subsequently, the device waits until the vehicle reaches a vulnerable state (as indicated by CAN bus data), such as the first signs of deceleration after achieving highway speeds (65+ mph). At this time, the malicious ELD may flood the CAN bus with Torque/Speed Control 1 (TSC1) messages, commanding the engine to spin to maximum capacity, potentially causing the vehicle to unexpectedly collide with an object it was attempting to slow down and avoid.

III. ELD Firmware Modification

In this section we describe how we modified the firmware of a popular consumer off-the-shelf ELD to execute an attack on the vehicle. While we focus on a specific ELD for the purpose of this paper, our research has indicated that these types of problems are not limited to this specific model or this specific manufacturer. As such, we do not believe it adds

[1]The ELD discussed in this paper left all interfaces on by default. Therefore ”active” and ”inactive” refers to whether the interface on the compromised ELD is in use by a connected device.

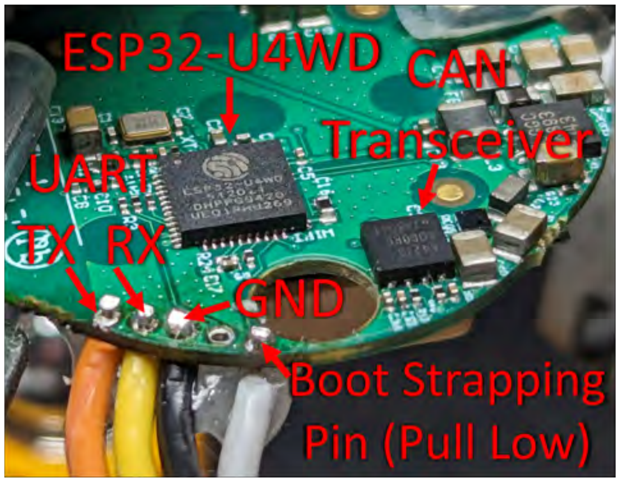


Fig. 1: ELD PCB Up Close

any value to this paper to disclose the manufacturer's name or device model.

A. ELD Architecture

The hardware and software described here are specific to the device we analyzed. However, we observed that multiple ELDs from the same manufacturer utilized almost identical architectures. Furthermore, we noted that there were large overlapping similarities between many of the diagnostic port mounted ELDs.

1) Physical Device: The ELD we analyzed is a small, hand-held device that plugins into the 9-pin diagnostic port on Heavy Vehicles. The device gathers data from the vehicle through the CAN channels exposed on the diagnostic port and is powered by the diagnostic port. The device has no other ports on it as all other communication is performed over wireless interfaces. The ELD presents 3 wireless interfaces namely: WiFi, Bluetooth, and GPS. GPS is used for gathering location data while WiFi and Bluetooth are used to communicate with an application on the user's phone or tablet. Depending upon the reseller, either WiFi or Bluetooth Low Energy (BLE) is used, but we did not find an instance where both where necessary. An ESP32 chip is in charge of controlling the device. While the exact chip model varied between devices, they all were dual threaded with integrated storage, and supported WiFi, BLE, GPS, and CAN. Other notable features include a small PCB antenna and an ESP programming port.

2) Execution Environment: The ESP32 chips executes a 32-bit Xtensa instruction set architecture (ISA) that blends 16-bit and 24-bit instructions in a RISC-based framework [19]. The device utilizes the Espressif Integrated Development Framework (ESP-IDF) which makes use of a Symmetric Multiprocessing implementation of FreeRTOS, a lightweight and fast Real Time Operating System [20]. Unless the General Purpose Input/Output (GPIO) pin 0, which is exposed with the ESP programming port described earlier, is pulled down, the board will boot the program code stored in flash. Otherwise, it will boot into the serial bootloader. The bootloader connects to a host computer via a serial connection and allows for reading/writing to the flash memory, setting EFuses, and more [20].

3) Software Architecture: Since the ESP32 IDF utilizes FreeRTOS, most of the device's functionality is segmented into tasks or threads. At the core of the ELD's architecture is the main thread, initiated after basic hardware setup and system checks. This thread begins by initializing critical components like flash memory, GPIO pins, and the TCP/IP adapter. Following this initialization phase, the software architecture diverges into several distinct threads, each dedicated to a specific function. Notable threads include, operating the WiFi interface for network connectivity, data transmission and device control, a similar thread for Bluetooth, a dedicated thread for WiFi debugging, an upgrade thread which hosts a web server for easy firmware updates, and another for data collection and logging related to vehicle interfacing.

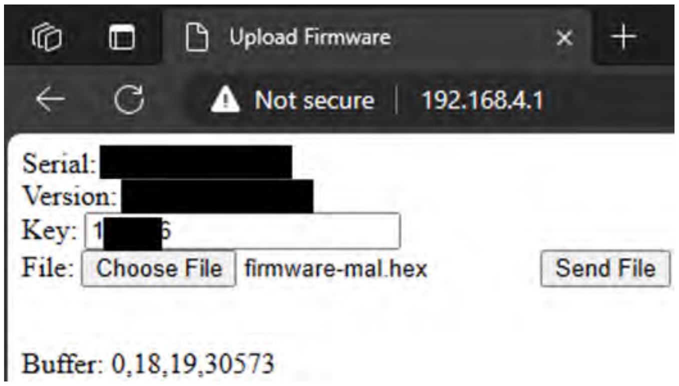
One of the most crucial aspects of this architecture, as it relates to this paper, is the over-the-air (OTA) update mechanism. This feature allows the ELD to update its firmware wirelessly, using WiFi or Bluetooth. To support this functionality, the device's partition tables are configured to include two OTA application partition slots and an OTA Data Partition. During an update, the system checks the checksum and SHA256 hash of the new firmware image for integrity violations before writing it to an inactive OTA slot. Once verified, the OTA Data partition is updated to boot from the new firmware.

B. Related Vulnerabilities Discovered

In our evaluation of ELD units procured from various resellers, we discovered that they are distributed with factory default firmware settings that present considerable security risks. Our reverse engineering endeavors exposed an API that permits extensive device feature configuration, including over-the-air (OTA) updates. Contrary to expectations, initial connections with the re-seller's mobile applications do not activate a setup or reconfiguration routine towards more secure settings.

- **Default Network Settings** By default, WiFi and Bluetooth functionalities are activated. The WiFi Service Set Identifier (SSID) is predictable, following the format <Vendor Name> ELD: <MAC_ADDR>, and is secured with a default password of minimal strength, 'de*******77'. [2]
- **Web Server and OTA Update Endpoint** The device hosts a web server, accessible via the IP address 192.168.4.1. A critical endpoint on this server is /upload.php, which facilitates OTA firmware updates. This page is safeguarded by a basic password (1****6), rendering it susceptible to unauthorized access.

[2]At the request of the manufacturer we have obscured the passwords and change the commands. Although their form and function remains the same.



Serial: [redacted]
Version: [redacted]
Key: 1[redacted]6
File: Choose File firmware-mal.hex Send File

Buffer: 0,18,19,30573

Fig. 2: Firmware Upload Page

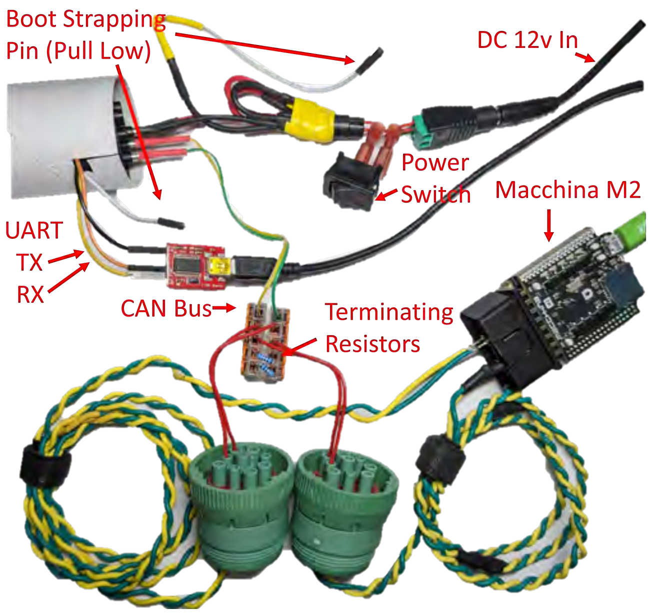


Fig. 3: Table top lab setup

- **Exposed Debug Thread and APIs** The device features a wirelessly accessible socket debug routine on port 22. Additionally, an API, which enables extensive control over the device, including the transmission and reception of arbitrary CAN messages as well as OTA updates with no protection, is exposed via Bluetooth and WiFi on port 23 via the Telnet protocol. The API endpoints and parameters pertinent to the security vulnerabilities discussed in this paper are outlined as follows, though this is not an exhaustive list:
  - **CONFIGURE/ACTIVATE/DEACTIVATE Commands:** These commands are instrumental in configuring the device, encompassing tasks such as disabling unused interfaces and modifying default passwords. While these parameters were present in the firmware, their application within the reverse-engineered Android applications was minimal or non-existent.
  - **INITIATEUPDATE/COMPLETEUPDATE Commands:** These commands are utilized for conducting OTA updates on the ELD. It is critical to note that, in contrast to the web-based update mechanism which requires a password, these commands do not require any form of authentication for OTA update initiation.
  - **DATASTREAM Command:** This command activates the streaming of raw CAN messages to a connected device. Enabling this setting is a prerequisite for utilizing the ELD to dispatch arbitrary CAN messages onto the connected CAN bus.
  - **Arbitrary CAN Message Transmission:** Unlike the other commands that employ ASCII-based words, the commands for sending arbitrary CAN messages consist of specific non-printable hexadecimal characters. These commands, along with the DATASTREAM command, were absent from all the Android applications from the resellers which we examined through reverse engineering, suggesting a reliance on the obscurity of these features as a form of security measure, though such a strategy was not explicitly confirmed. The following list delineates the initial character of each command, the corresponding CAN bus, and the format of the CAN ID (standard 11-bit or extended 29-bit IDs):
    - 0xF4: can1, extended.
    - 0xF5: can1, standard.
    - 0xF6: can2, extended.
    - 0xF7: can2, standard.
    - 0xF8: can3, extended.
    - 0xF9: can3, standard.
- **Firmware Security Flaws** The device lacks enforcement of firmware signing for authenticity, relying only on rudimentary integrity checks via checksums and hashes, which are inadequate for assuring firmware security. Digitally signed firmware is a better alternative.

1) Criteria for Successful Exploitation:

- **Proximity for Wireless Access** Attackers must be within the wireless range to leverage WiFi and Bluetooth vulnerabilities for actions like sending arbitrary CAN messages or uploading malicious firmware.
- **Exploiting Default Settings**
  - **Bluetooth Access** With a predictable Bluetooth identifier and a "Just Works" pairing mode, attackers can connect to the device if the driver is not currently connected [21]. Post-connection, the API allows them to either upload malicious firmware or manipulate CAN messages.
  - **WiFi Access** The device's WiFi, identifiable by a predictable SSID and protected by a weak password, grants network access. Once connected, attackers can either utilize the /upload.php endpoint on the internal web server for firmware uploads or exploit the API on port 23 via Telnet for sending/receiving CAN messages or firmware manipulation.

C. Analysis Techniques

1) Firmware Extraction and Acquisition: The initial phase of our research involved gathering information from public sources such as reference manuals, official documentation, and various online forums, which provided foundational knowledge about the ELD's operation and potential security issues [20], [22], [18]. Using this background information, we proceeded with the firmware extraction from the ELD using ESPTool.py from Espressif [23]. The extraction process was facilitated by a serial to USB converter, which connected to the programming port on the ELD. Additionally, newer firmware versions were obtained from the update servers, the addresses of which were identified through reverse engineering of the mobile applications associated with the ELD using JADX, a popular Dex to Java decompiler [24]. The discovery of credentials and webpage endpoints was achieved by running the GNU `strings` command on the firmware, providing further insight into embedded information in the binary files [25].

2) Firmware Analysis Methodology: Our analysis of the ELD's firmware involved a suite of tools, each contributing to understanding of its structure and potential vulnerabilities. Ghidra, a software reverse engineering tool, played a central role in this analysis [26]. The integration of ESP32-specific plugins into Ghidra was crucial as these plugins enabled the intake of firmware, creation of memory maps, and disassembly of the assembly code, specifically tailored to the ESP32 architecture [27], [28], [29]. Complementing Ghidra's capabilities, Binwalk was used to examine and extract the contents of the firmware images [30].

3) Network Analysis: The network footprint of the ELD, including open network ports and services, was mapped out using Nmap [31]. The analysis revealed various open ports and the services associated with them, providing insights into the network-based aspects of the ELD's security architecture.

4) Observation of Default Credential Usage: A significant finding in our research was the ease of access to the ELD's default WiFi credentials, which were readily available through a basic internet search. This observation highlights a concerning security practice where default settings are often left unchanged, increasing the risk of unauthorized access and exploitation. The widespread availability of such sensitive information emphasizes the need for more robust and conscientious security measures in the deployment and management of ELD systems.

D. Development of Malicious ELD Firmware

In the context of this research, the development of the malicious firmware for Electronic Logging Devices (ELDs) was approached with a focus on demonstrating the feasibility of manipulating ELD firmware to perform an attack on a truck, rather than creating highly destructive software. To this end, the constructed firmware was relatively simple and designed for safe testing.

Listing 1: Malicious Firmware Inserted in Debug Thread

```
void debug_thread(void) {
  while (((STATUS & 2) == 0) &&
         ((STATUS & 4) == 0)) {
    vTaskDelay(1000);
  }
  while(true) {
    can_data = 0xCB000000FF00FFFF;
    send_can(
      0xC000003, &can_data, 8);
    vTaskDelay(1);
  }
}
```

The code in Listing 1 is explained in the following subsections.

1) Construction of the TSC1 Message: The J1939 Torque/Speed Command 1 message (TSC1) message is a legitimate message to command the engine output over the J1939 network. This message, however, can be abused. The following details explain how the message was composed.

CAN ID: The CAN ID used is a 29-bit identifier, following the J1939 standard. The PGN for TSC1 is 0, and the default priority is 3, which was retained for this experiment. The source address selected was 3 (0x03), corresponding to the spoofed Transmission Controller. These elements combined to form the ID: 0x0C000003.

J1939 Suspect Parameter Number (SPN) Specifications:

- SPN 695 controlled the Engine Override Control Mode, with values ranging from 'Override Disabled' to 'Speed/-Torque Limit Control'.
- SPN 696 defined the Engine Speed Control Conditions, optimized for various driveline engagement scenarios.
- SPN 879 set the Override Control Mode Priority, ranging from 'Highest Priority' for urgent safety actions to 'Low Priority' for driver comfort.
- SPN 898 and SPN 518 specified the Engine Speed/Speed Limit and Engine Torque/Torque Limit, respectively, offering control over engine speed and torque based on external input.
- The other SPN are not important for this scenario and are set to FF's as default values.

Data Packet Formation: The final data packet for the TSC1 message was composed as 0xCB000000FF00FFFF.

2) Integration into the Firmware: The malicious code was integrated into a debug thread that was left in the production devices. This placement allowed the malicious code to function as its own FreeRTOS thread/task. Initially, upon creation, the thread/task would begin checking a global STATUS variable. While the true meaning of the STATUS variables states remains elusive, a set of states appeared to indicate when the CAN interfaces completed their setup routines, including automatic baud rate detection.

The core function, termed `send_can`, was responsible for dispatching the constructed CAN messages. It accepted parameters including the ID, data buffer, and data length. After invoking `send_can` with the formulated TSC1 message data, a delay of 1 millisecond was introduced to ensure adequate propagation time for the message onto the CAN bus.

To bypass integrity checks after modifying the firmware, we recalculated the checksum, and updated the SHA256 hash to make the firmware appear legitimate.

3) Safety Measures in Experimentation: To ensure safety, the experimental use of the TSC1 message was carried out on a private and empty airfield runway. The parameters chosen for the TSC1 message were set to slow down the truck instead of speeding it up, presenting a safer testing scenario. This controlled environment allowed for a detailed evaluation of the firmware's capabilities without posing a risk to public safety or property.

E. Evaluation of the Malicious Firmware and its Deployment

The evaluation of the malicious firmware and its deployment was conducted through two primary types of tests: bench-top tests and a real-world drive-by attack simulation.

1) Bench Top Testing: The bench-top test setup, as depicted in 3, provided a controlled environment for initial evaluations. In this setup, the ELD, with its ESP programming port connected to a USB-to-serial adapter, facilitated monitoring via the serial console. The ELD was also connected to a CAN bus shared with a Macchina M2, replaying logged data from a truck, simulating a real vehicle CAN bus in a bench-top setting. The firmware was uploaded to the ELD, through the previously described interfaces and APIs and averaged around 12 seconds for flashing, with additional operations bringing the total average duration to approximately 30 seconds. These tests aimed to verify the firmware's functionality, particularly focusing on the data logged on the CAN bus. The evaluations confirmed that both the malicious firmware and the API access point, which enabled sending and receiving arbitrary CAN messages, performed as expected.

2) Real-World Drive-By Attack Simulation: The drive-by attack simulation was conducted on an empty mile-long airfield. This test aimed to demonstrate the potential of the ELD to spread in scenarios where trucks congregate, such as rest stops, or even while in transit. For this experiment, a 2014 Kenworth T270 Class 6 research truck, equipped with the ELD, was driven at approximately 20 miles per hour down the runway. The attacker, a passenger in a Tesla Model Y with a laptop and an Alfa extended range wireless adapter, executed the attack. This setup was chosen to expedite the attack due to the airfield's limited length, although subsequent experiments (discussed in the next section) suggest that an extended range adapter might not be necessary if more time is available to conduct the attack.

The attack vehicle initially followed the truck, then, to simulate a passing scenario, moved alongside it. The attacker connected to the truck's WiFi and used the ELD's web interface to re-flash the device while both vehicles were in motion. Notably, the signal strength inversely affected the flashing duration; however, the extended range adapter allowed the re-flashing process to be completed in approximately 14 seconds. Around 20 seconds later, allowing time for the ELD

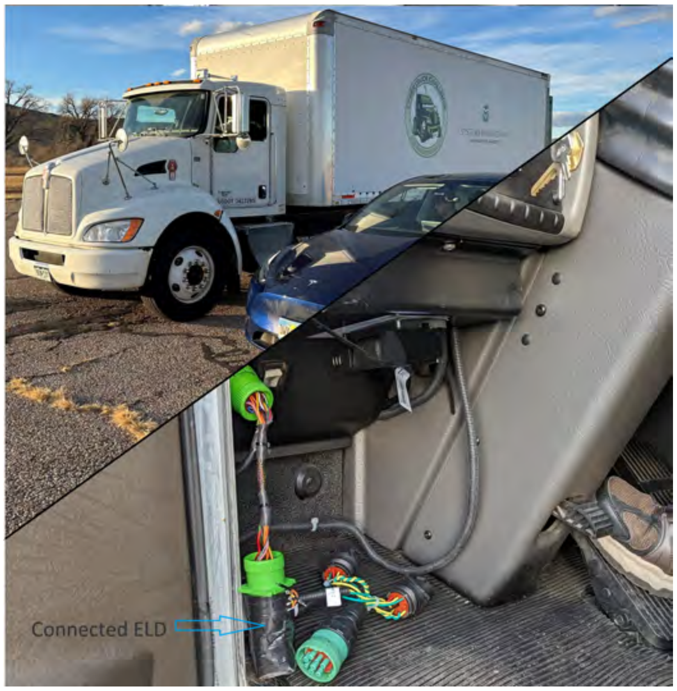
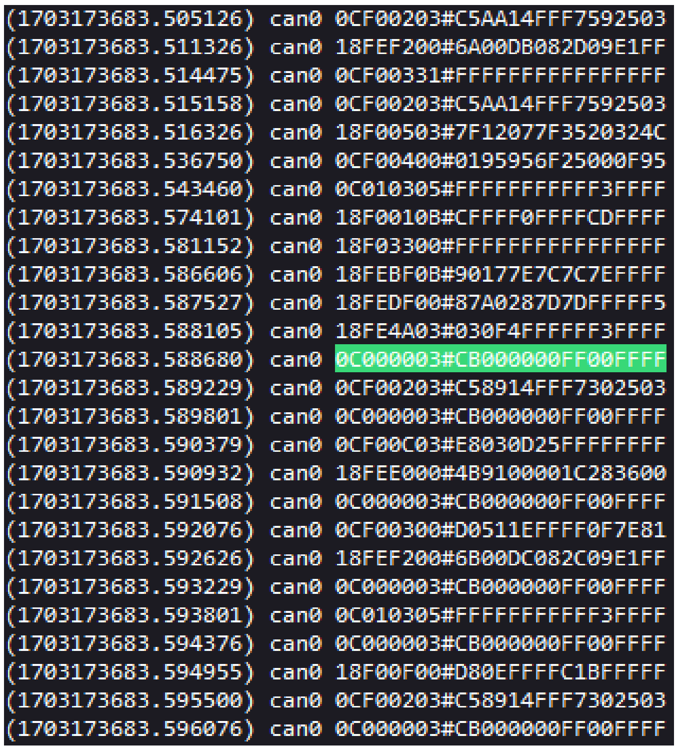


Fig. 4: Research Truck (top) with ELD plugged into Diagnostic Port and hanging to the right of the door edge (bottom).



```
(1703173683.505126) can0 0CF00203#C5AA14FFF7592503
(1703173683.511326) can0 18FEF200#6A00DB082D09E1FF
(1703173683.514475) can0 0CF00331#FFFFFFFFFFFFFFFF
(1703173683.515158) can0 0CF00203#C5AA14FFF7592503
(1703173683.516326) can0 18F00503#7F12077F3520324C
(1703173683.536750) can0 0CF00400#0195956F25000F95
(1703173683.543460) can0 0C010305#FFFFFFFFFFF3FFFF
(1703173683.574101) can0 18F0010B#CFFFF0FFFFCDFFFF
(1703173683.581152) can0 18F03300#FFFFFFFFFFFFFFFF
(1703173683.586606) can0 18FEBF0B#90177E7C7C7EFFFF
(1703173683.587527) can0 18FEDF00#87A0287D7DFFFFF5
(1703173683.588105) can0 18FE4A03#030F4FFFFFF3FFFF
(1703173683.588680) can0 0C000003#CB000000FF00FFFF
(1703173683.589229) can0 0CF00203#C58914FFF7302503
(1703173683.589801) can0 0C000003#CB000000FF00FFFF
(1703173683.590379) can0 0CF00C03#E8030D25FFFFFFFF
(1703173683.590932) can0 18FEE000#4B9100001C283600
(1703173683.591508) can0 0C000003#CB000000FF00FFFF
(1703173683.592076) can0 0CF00300#D0511EFFFF0F7E81
(1703173683.592626) can0 18FEF200#6B00DC082C09E1FF
(1703173683.593229) can0 0C000003#CB000000FF00FFFF
(1703173683.593801) can0 0C010305#FFFFFFFFFFF3FFFF
(1703173683.594376) can0 0C000003#CB000000FF00FFFF
(1703173683.594955) can0 18F00F00#D80EFFFFC1BFFFFF
(1703173683.595500) can0 0CF00203#C58914FFF7302503
(1703173683.596076) can0 0C000003#CB000000FF00FFFF
```

Fig. 5: Start of the attack (highlighted in green)

to initialize its CAN interfaces, the truck began to slow down as the ELD flooded the CAN bus with malicious TSC1 messages. This test successfully demonstrated the feasibility of a drive-by attack, highlighting the realistic possibility of such an attack occurring under actual driving conditions.

IV. Design of a Truck to Truck Worm

A. Overview

The Truck to Truck Worm (TtTW), developed for ESP32 development boards, executes an infection and propagation strategy tailored to Electronic Logging Devices (ELDs). The concept demonstrated with development boards is shown in Fig. 6.

1) Initial Setup and Network Configuration: The worm begins by initializing the essential system components for its operation. This setup involves configuring network interfaces in both access point (AP) and station (STA) modes, preparing the file systems, and setting up CAN bus communication protocols.

2) Scanning and Identification of Targets: During the scanning phase, the worm utilizes its WiFi capabilities to search for nearby ELDs that are potential targets. It specifically looks for devices with SSIDs starting with ”VULNERABLE ELD:”. Although this may sound contrived the SSID of the ELD we examined was predictable and could be used to identify the vulnerable devices.

3) Infection Process: Once a vulnerable ELD is identified, the Truck to Truck Worm proceeds with the infection process. The worm exploits a vulnerability by connecting to these ELDs using hardcoded default credentials. Upon establishing a connection, the worm transfers its malicious payload to the target ELD. This payload is designed to embed the worm within the system, overwrite existing firmware, and prepare the device for its role in further propagation.

4) Post-Infection State and Propagation Readiness: After successfully infecting an ELD, the worm alters the WiFi AP of the infected device to ”INFECTED ELD.” This change serves an important purpose: it signals to other instances of the worm that the device is already infected, preventing unnecessary re-infection attempts. Moreover, this post-infection state sets the stage for the device to become an active participant in spreading the worm. The infected ELD, now operating under the control of the worm, continues the cycle of scanning, identifying, and infecting other vulnerable ELDs, perpetuating the spread of the Truck to Truck Worm.

B. Truck to Truck Worm Evaluation

In our study, we conducted two primary tests to evaluate the Truck to Truck Worm developed on ESP32 development boards, focusing on its range and effectiveness. [3]

[3]The firmware for the conceptual Truck to Truck Worm Design can be found here: https://github.com/SystemsCyber/Truck-Worm

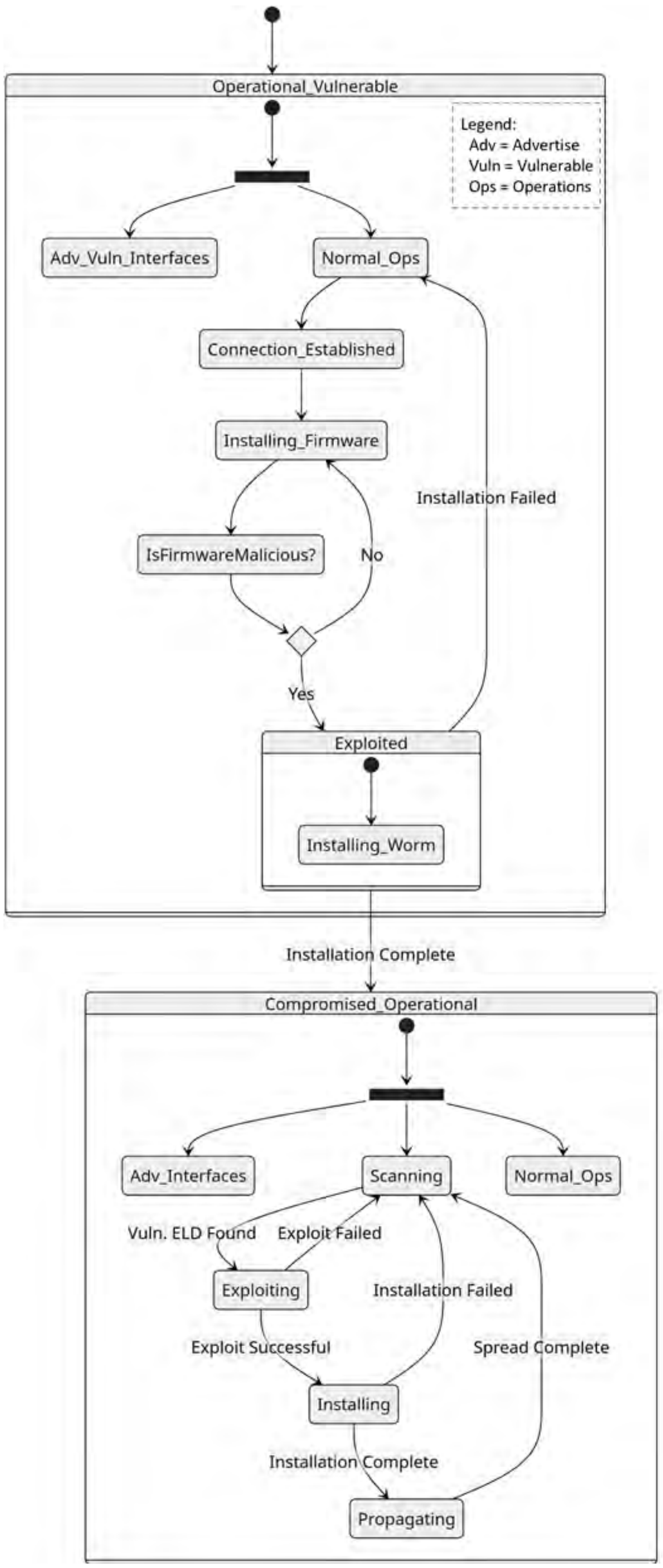


Fig. 6: ELD State Machine Diagram

1) Testing Worm Spread in Stationary Conditions: The first test aimed to assess the worm's spreading distance and effectiveness in a fully parked, stationary setting. For this, we utilized two ESP32 development boards, which closely emulate the power and antenna size of actual ELD devices. This parity in hardware specifications is critical for obtaining realistic results, especially in contrast to the earlier drive-by test where an extended range WiFi antenna, with its superior power and larger size, was used.

The procedure began with positioning one ESP32 development board adjacent to the diagnostic port of our research truck. Simultaneously, we activated the programs on both ESP32 devices: one functioning as a WiFi network host and the other programmed to scan and connect to this network. The key aspect of this test was to methodically move from one parking space to another, placing the second ESP32 device near the metal door of each truck to gauge the maximum effective connection distance.

Our findings revealed that, even in a full parking lot, a connection could be established up to approximately 12 parking spots away, equivalent to about 120 feet. This distance demonstrated a considerable range for the Truck to Truck Worm to spread in environments where trucks commonly congregate. Furthermore, the test indicated that the proximity of trucks is a critical factor for the worm's spreading efficiency, with closer distances leading to quicker and more successful propagation. Under optimal conditions the Truck to Truck Worm was capable of spreading between ESP32 devices in under 30 seconds.

V. Mitigations Against Worm Attacks

To address the vulnerabilities identified in our research and effectively prevent Truck-to-Truck Worm attacks in electronic logging devices (ELDs), a multifaceted approach is required. This approach encompasses the enhancement of default security settings, implementation of robust firmware integrity and authenticity checks, and the elimination of unnecessary and high-risk features. The specific recommendations for enhancing the security of ELD systems, as outlined in this section, are tailored to balance stringent cybersecurity needs with the practical requirements of affordability, reliability, and user-friendliness inherent in ELDs. This balance is crucial in the context of the trucking industry, where ELDs are a widespread and mandatory technology. These mitigations are inspired by the National Motor Freight Traffic Association, Inc. (NMFTA) with their Cybersecurity Requirements for Telematics Systems [32]. The detailed strategies are as follows:

A. Enhancing Default Security Settings

1) Disabling Unused Interfaces and Services: To minimize the attack surface, it is crucial to disable any interfaces and services that are not in active use. Our study revealed that while some resellers utilized Bluetooth, others employed WiFi, but none concurrently used both interfaces or the web server. Therefore, ELDs should be configured to disable unused wireless interfaces and the internal web server by default.

High-Entropy Default Passwords Implementing high-entropy passwords for initial device access significantly enhances security. This could be achieved through two approaches:

- *Complex Randomized Passwords* Generate long, completely random passwords during the first provisioning of the device. These passwords should be unique to each device and securely labeled on it, akin to practices observed in modern router configurations.
- *Semi-Randomized Passwords* Alternatively, a standard password prefix could be used, with the last four digits randomized and labeled on each device. This method also ensures a degree of randomness while maintaining user convenience.

2) Firmware Integrity and Authenticity Checks: **Secure Firmware Signing Mechanism:** It is imperative to utilize a secure firmware signing mechanism. This involves cryptographic signing of firmware updates to ensure they are not tampered with and originate from a verified source. This mechanism ensures that only authentic and untampered firmware is installed on the ELDs, thereby preventing the installation of malicious firmware.

3) Restriction on Arbitrary CAN Message Functionality: **Eliminating Unnecessary API Features:** Our findings suggest that the ability to send and receive arbitrary CAN messages via an API in a production ELD presents an unwarranted risk without a valid use case. Thus, it is recommended to eliminate this feature from ELDs. Restricting this functionality will significantly reduce the risk of unauthorized access and control over the vehicle's CAN network, thereby mitigating potential security threats.

B. Implementing Telematic Device Firewalls/Gateways for Enhanced Security

In addition to the device-centric preventative measures previously discussed, trucking companies should consider adopting proactive security practices for their fleets, particularly in the case of older vehicles that might not have built-in security gateways. A significant step in this direction involves the use of Telematic Device Firewalls or Gateways, which serve as an intermediary layer of security between the Electronic Logging Devices (ELDs) and the vehicle's diagnostic port.

Functionality and Benefits: Telematic Device Gateways are designed to filter the data between an ELD and the vehicle's diagnostic port, essentially acting as an bolt-on gateway or firewall that can help to prevent unauthorized access and malicious messages from reaching critical vehicle control systems through the diagnostic port. This is particularly important for older vehicles, which may lack some cybersecurity features found in newer models.

In summary, the implementation of these preventative measures is essential for enhancing the security of ELD systems against Truck to Truck Worm attacks. By focusing on more secure default settings, rigorous firmware integrity checks, and the elimination of unnecessary and risky functionalities, manufacturers and users of ELDs can significantly reduce the likelihood and impact of such cybersecurity threats.

VI. Conclusion and Future Work

This research has illuminated significant vulnerabilities in electronic logging devices (ELDs), a mandated technology in the trucking industry, underscoring the critical need for enhanced cybersecurity measures. Through comprehensive testing, both in controlled settings and in real-world environments, we have demonstrated the practical risks and potential impacts of a Truck to Truck Worm facilitated by these devices.

The findings from our study highlight the importance of security in technologies that are not only integral to operational efficiency but also legally mandated. The vulnerability of such systems poses a broader risk to the entire supply chain, making it imperative that security measures evolve in tandem with technological advancements.

Our recommendations for bolstering ELD security, such as optimizing default security settings, ensuring firmware integrity, and limiting unnecessary API features, are designed to be practical and effective, considering the constraints of cost, reliability, and user-friendliness. These steps are crucial for mitigating the risks identified and setting a foundation for more secure operations.

Looking ahead, it is clear that continuous innovation and vigilance in cybersecurity are essential, especially for technologies mandated by regulatory bodies. Furthermore, regulating bodies need to be aware of the increased security risks associated with mandated technologies that interface with deployed control networks. Future research should focus on developing and implementing advanced, adaptable security measures that can protect against evolving threats while ensuring seamless operational integration. This balance is vital for safeguarding the trucking industry and, by extension, the critical supply chains it supports.

Ethical Considerations

This research was conducted with a strong commitment to ethical standards and responsible practices. Prior to public disclosure of the findings, a coordinated disclosure process was initiated with the affected manufacturer and the Cybersecurity and Infrastructure Security Agency (CISA). This process adhered to a 90-day timeline, ensuring that the manufacturer had time to address and mitigate the identified vulnerabilities.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Naval Information Warfare Center Pacific (NIWC Pacific) under Contract No. N66001-20-C-4021. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or NIWC Pacific.

References

[1] Bureau of Transportation Statistics, United States Department of Transportation. , "National Transportation Statistics (NTS)," 2019. [Online]. Available: https://tinyurl.com/rosapntlbtsNTS

[2] "Economics and Industry Data." [Online]. Available: https://www.trucking.org/economics-and-industry-data

[3] Federal Motor Carrier Safety Administration, "Electronic logging devices and hours of service supporting documents," https://www.govinfo.gov/content/pkg/FR-2015-12-16/pdf/2015-31336.pdf, 12 2015, accessed: November 30, 2023.

[4] Technology Maintenance Council, "RP1226: Vehicle Accessory Connector Guidelines," in *2020-2021 Recommended Practices Manual*. American Trucking Association's Technology Maintenance Council, 2020.

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

[6] Society of Automotive Engineers, "SAE J1939 Standards Collection." [Online]. Available: https://www.sae.org/standardsdev/groundvehicle/j1939a.htm

[7] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," in *Blackhat USA*. Las Vegas, NV, USA: Blackhat Press, 2015.

[8] Y. Burakova, B. Hass, L. Millar, and A. Weimerskirch, "Truck Hacking: An Experimental Analysis of the SAE J1939 Standard," in *Proceedings of the 10th USENIX Conference on Offensive Technologies*. Austin, TX, USA: USENIX Association, 2016, pp. 211–220.

[9] P. Murvay and B. Groza, "Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4325–4339, 2018.

[10] M. T. Campo, S. Mukherjee, and J. Daily, "Real-Time Network Defense of SAE J1939 Address Claim Attacks," *SAE International Journal of Commercial Vehicles*, vol. 14, no. 3, pp. 02–14–03–0026, Aug. 2021. [Online]. Available: https://www.sae.org/content/02-14-03-0026/

[11] 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*, 2016, pp. 23–42.

[12] R. Chatterjee, S. Mukherjee, and J. Daily, "Exploiting transport protocol vulnerabilities in SAE J1939 networks," in *Proceedings of the Inaugural International Symposium on Vehicle Security & Privacy*. San Diego, CA, USA: Internet Society, 2023. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2023/02/vehiclesec2023-23053-paper.pdf

[13] C. Miller and C. Valasek, "A Survey of Remote Automotive Attack Surfaces," in *Black hat USA*. Las Vegas, NV, USA: Blackhat Press, 2014, p. 94.

[14] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *USENIX Security Symposium*, vol. 4. San Francisco, CA, USA: USENIX Association, 2011, pp. 447–462.

[15] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," in *IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE, 2010, pp. 447–462.

[16] D. Klinedinst and C. King, "On Board Diagnostics: Risks and Vulnerabilities of the Connected Vehicle."

[17] C. Theun, "Heavy truck and electronic logging devices: What could go wrong," Presented at DEF CON 25 Car Hacking Village, 2017, video recording available online. [Online]. Available: https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20villages/DEF%20CON%2025%20Car%20Hacking%20Village%20-%20Corey%20Theun%20-%20Heavy%20Truck%20and%20Electronic%20Logging%20Devices-%20What%20Could%20Go%20Wrong.mp4

[18] "ELD List." [Online]. Available: https://eld.fmcsa.dot.gov/List

[19] "Xtensa Instruction Set Architecture (ISA) Reference Manual."

[20] Espressif, "ESP32 IDF Github." [Online]. Available: https://github.com/espressif/esp-idf/

[21] Kai Ren, "Bluetooth Pairing Part 2 Key Generation Methods," Jun. 2016. [Online]. Available: https://www.bluetooth.com/blog/bluetooth-pairing-part-2-key-generation-methods/

[22] Espressif, “ESP32 Techical Reference Manual.” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
[23] Espressif, “ESPTool.” [Online]. Available: https://github.com/espressif/esptool
[24] Skylot, “JADX.” [Online]. Available: https://github.com/skylot/jadx
[25] GNU Operating System, “GNU Binutils.” [Online]. Available: https://www.gnu.org/software/binutils/
[26] National Security Agency, “Ghidra.” [Online]. Available: https://ghidra-sre.org/
[27] Dynacylabs, Austin Tyler Conn, Ebiroll, and Yath, “Ghidra-Xtensa.” [Online]. Available: https://github.com/dynacylabs/ghidra-xtensa
[28] Dynacylabs, Austin Tyler Conn, Ebiroll, and Tslater2006, “Ghidra ESP32 Flash Loader.” [Online]. Available: https://github.com/dynacylabs/ghidra-esp32-flash-loader
[29] Leveldown Security, “SVD Loader.” [Online]. Available: https://github.com/leveldown-security/SVD-Loader-Ghidra
[30] ReFirmLabs, “Binwalk.” [Online]. Available: https://github.com/ReFirmLabs/binwalk
[31] Gordon Lyon, “Nmap.” [Online]. Available: https://nmap.org/
[32] National Motor Freight Traffic Association, Inc., “Cybersecurity Requirements for Telematics Systems,” Mar. 2022. [Online]. Available: https://nmfta.org/wp-content/media/2022/11/NMFTA-Cybersecurity-Requirements-for-Telematics-Systems-v1.5.pdf