VPTP: Vehicle Penetration Testing Platform

No Author Given

No Institute Given

Abstract. Modern vehicles expose numerous electronic attack surfaces but lack consistently deployed cybersecurity protections. The *Vehicle Penetration Testing Platform* (VPTP) was created to give researchers and defenders an inexpensive, handheld environment for exercising those attack surfaces in a scientifically repeatable way. Built around community-driven Raspberry Pi Hardware with support for various external interfaces, such as Logic Analyzers, it includes an off-the-shelf penetration-testing suites under a task-oriented user interface. This paper details the design choices, engineering trade-offs and evaluation results obtained from validation testing on a 2018 Freightliner Cascadia test bench and a 2014 Kenworth T270. The results confirm that the platform can reproduce previously published attacks (e.g. J1939 DoS, and Spoofing), while automatically logging the results and generating failure mode and effects analysis (FMEA) artifacts. We release all hardware prints, software, and documentation under an open-source license.

Keywords: Automotive cybersecurity, vehicular networks, penetration testing, CAN bus, SAE J1939, SAE J1708, in-vehicle network security, open-source hardware, Raspberry Pi, protocol fuzzing, FMEA automation, automotive Ethernet, LIN bus, heavy-duty vehicle security

1 Introduction

Connected and automated vehicles employ dozens of electronic control units (ECUs), multiple in-vehicle networks, and increasing wireless connectivity. While functional safety is mature, cyber-physical security still trails [8]. Attack demonstrations on passenger and heavy-duty vehicles, most famously by Miller and Valasek [10], show that remote adversaries can alter vehicle state. Researchers, regulators, and original-equipment manufacturers (OEMs) therefore need repeatable ways to probe vehicular networks during design, production, and field-support phases.

Despite increased attention to vehicle security in recent years, the tools and techniques for cybersecurity assessment remain inefficient and up to the manufacturer for implementation. Most penetration testing platforms rely on proprietary hardware and software, lack portability, or only support a narrow set of protocols. Standards such as ISO/SAE 21434 [3] and UN ECE R155[4] outline high-level requirements but fall short on prescribing detailed methods for testing and validation.

This results in a lack of consistency across the industry, with security assessments often relying on ad hoc processes, limited automation, and minimal integration with broader systems engineering practices. This disconnect is problematic because vulnerabilities that affect vehicle safety can arise from purely cyber origins, and vice versa [15].

This paper asks whether a low-cost, portable, and fully open-source device can provide comprehensive penetration testing capabilities across a wide range of vehicular environments. To explore this question, we introduce the Vehicle Penetration Testing Platform (VPTP).

2 Background and Related Work

2.1 Gaps in Existing Methodologies

A recent survey of automotive security testing highlights the lack of standardized tools and repeatable methodologies needed to address the complex threat models present in modern vehicles (Fig. 1) [11]. Official standards such as ISO/SAE 21434 and UN ECE R155 specify what must be secured, but stop short of prescribing how to test those requirements [15]. Although researchers have proposed model-based and fully automated test frameworks [12], publicly available implementations remain rare. Worsening this gap, many existing tools are protected as proprietary intellectual property, limiting access for public use, academic research, and community-driven development. In short, OEMs have clear standards to meet—but lack the open, standardized tooling needed to verify compliance.

2.2 Foundations for VPTP

This work builds on prior research with truck hacking platforms [14], expanding the focus to portability, user-driven extensibility, and integrated results reporting. The predecessor platform runs a custom Yocto-built operating system with tools for evaluating the Controller Area Network (CAN), Local Interconnect Network (LIN), and SAE J2497. While effective for data gathering on vehicle networks, it is not a standalone device—it requires a host computer for configuration and operation. The Vehicle Penetration Testing Platform (VPTP) addresses these limitations by combining the same core software suite with true portability, incorporating an onboard battery and touch screen for independent use, and packaging everything in an open-source toolkit. As Clarke and Dorwin note [9], open-source culture fosters faster peer review, lowers acquisition costs, and increases transparency. The VPTP also integrates with an early version of the Cyber-Physical Penetration Testing Framework (CPPTF) [13], which unifies its web and command-line interfaces and enables portability to other platforms such as Windows and macOS.

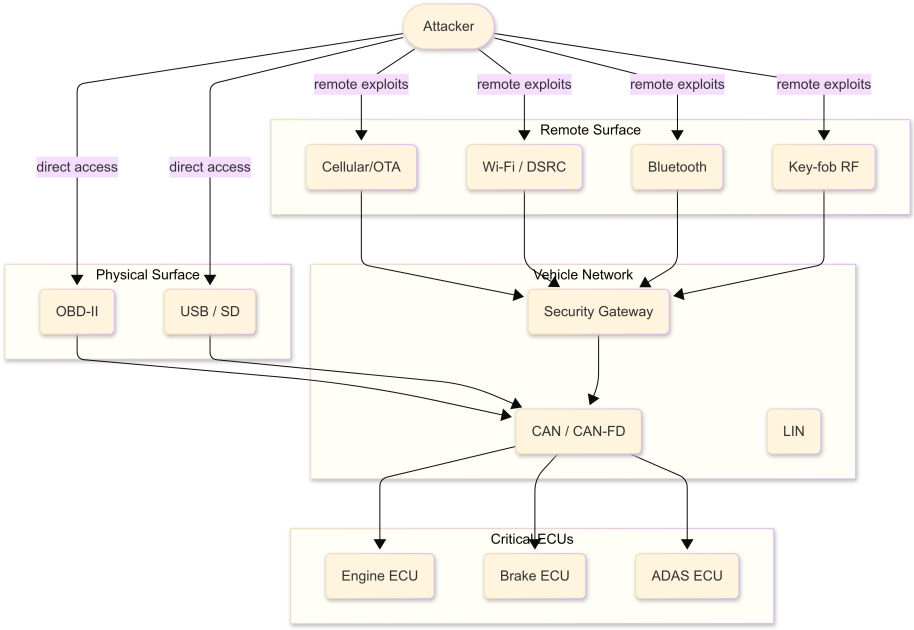

Fig. 1. Threat Model for the Medium-Heavy Duty Industry

3 System Requirements

3.1 Hardware Requirements

HW-001 **Compute Core:** The system shall utilize a compute module capable of hardware-accelerated cryptographic operations (e.g., AES, SHA, RSA, ECC). The Raspberry Pi 5 single-board computer (SBC) has been selected for its quad-core 64-bit processor, GPU acceleration, and integrated cryptographic extensions.

HW-002 **Network Interfaces:** The system shall support, at minimum:

- HW-002a Four (4) independent CAN channels compliant with ISO 11898-2/FD [5].
- HW-002b One (1) Power Line Communication (PLC) or SAE J1708/J1587 channel for heavy-duty vehicle/trailer networks [2].
- HW-002c One (1) Local Interconnect Network (LIN) channel compliant with ISO 17987 [6].
- HW-002d One (1) automotive Ethernet interface compliant with IEEE 802.3bw (100BASE-T1) [1].

HW-003 **Input/Output:** The system shall provide a minimum of three (3) USB 3.0 Type-A ports, one (1) Gigabit Ethernet port, one (1) HDMI output, accessible General-Purpose Input/Output (GPIO) pins, and a USB-C power input supporting battery-backed power delivery.

HW-004 **Form Factor:** The system shall be housed in a ruggedized ABS plastic enclosure with external dimensions of 152.4 mm × 228.6 mm, and shall integrate a 7-inch capacitive multi-touch display with gesture support.

HW-005 **Environmental Ruggedization:** The system shall be mechanically secured with a mounted PCB, environmentally sealed using caulk or gasket sealant, and capable of continuous operation in ambient temperatures from −20 °C to +60 °C.

3.2 Software Requirements

SW-001 **Penetration Testing Toolkit:** The system shall include pre-installed network and wireless penetration testing tools, including but not limited to `nmap`, `Wireshark`, `Metasploit`, and `Aircrack-NG`.

SW-002 **Bus Access Utilities:** The system shall provide SocketCAN-compatible drivers and utilities (`can-utils`) for raw CAN bus interfacing.

SW-003 **Scripting Environment:** The system shall include Python 3.12 with relevant libraries (`python-can`, `scapy`) and custom VPTP scripts for protocol analysis and exploitation.

SW-004 **Data Streaming and Visualization:** The system shall run an MQTT broker configured to forward bus data to Grafana dashboards for real-time visualization.

SW-005 **Secure Remote Access:** The system shall provide containerized SSH access to the full VPTP environment, with authentication and session logging enabled.

SW-006 **Web-Based Control Interface:** The system shall host a browser-accessible landing page enabling execution of preconfigured attack scripts, monitoring of active sessions, and access to stored data visualization outputs.

4 Research Goals

Our design is driven by three high-level questions:

1. Portability. Can a handheld device execute complex attack chains without tethered laptops, lab power, or additional network adapters?
2. Reproducibility. Can tests be parameterized, and replayed across vehicle models?
3. Usability. Can non-expert maintainers trigger tests and obtain meaningful, standard-based reports such as FMEA tables or UN ECE R155 evidence?

5 Implementation

The VPTP was implemented as a proof-of-concept system to meet the outlined research goals.

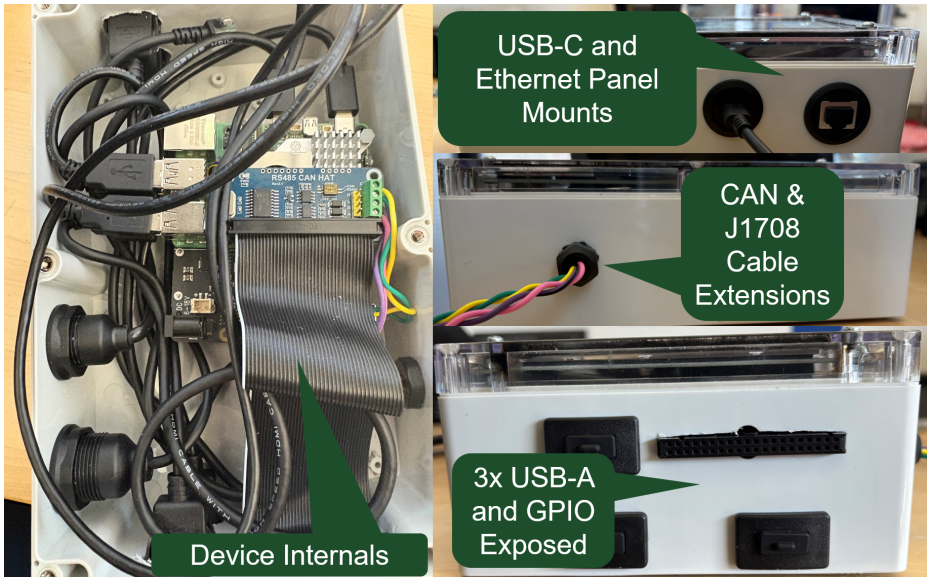

Fig. 2. Internal layout of the VPTP prototype showing transceiver daughter-cards and cabling.

5.1 Hardware Architecture

Fig. 2 shows the major subsystems of the VPTP. A microcontroller-based HAT manages real-time CAN bus interactions and interfaces directly with the Raspberry Pi. Panel-mounted external connectors simplify in-vehicle wiring while ensuring durability in typical operating environments. Inside the enclosure, a Raspberry Pi 5 base PCB integrates four 18650 lithium-ion batteries, providing portable, battery-backed operation. Equipped with automotive Ethernet for use as a network tap, GPIO pins for low-level protocol analysis (e.g., SPI), and multiple USB-A ports for peripherals such as SDRs or logic analyzers, the hardware architecture is designed to support the full range of relevant networking stacks.

5.2 Software Framework

Fig. 3 summarizes the layered software stack. At the core is a test-runner daemon that orchestrates containerized Metasploit modules, Python fuzzers, and SDR tools. Each test advertises its metadata (affected protocol, CVE, expected pre-conditions) via a JSON manifest consumed by the touchscreen User Interface (UI). The UI integrates with the software framework (VPTF) such that users can submit queries to run the back end logic (e.g., metasploit, scapy, other custom submodules), and simultaneously view the data in a representable way with Grafana. This works with Grafana's Business Forms plugin, and a python-based (flask) backend.

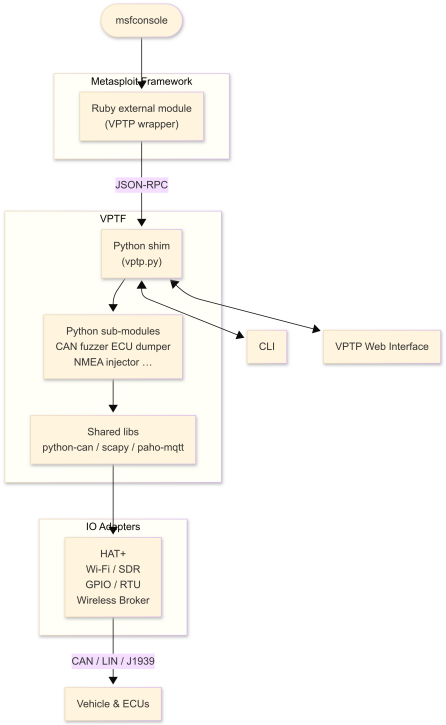

Fig. 3. Software stack and data flow inside VPTP.

5.3 Automated FMEA Generation

The operator inputs are formatted to generate a Failure Mode and Effects Analysis (FMEA) report documenting potential attack vectors and mitigations (Fig. 4). The RPN, calculated as Severity × Occurrence × Detection, each rated from 1 to 10 — highlights critical failures, allowing engineers and security personnel to prioritize corrective actions.

$$\text{RPN} = \text{Severity} \times \text{Occurrence} \times \text{Detection} \tag{1}$$

Each factor in the equation is rated on a scale from 1 to 10. Severity describes the seriousness of the consequences of a specific failure mode, occurrence outlines how likely the failure is to happen, and detection is how likely the failure can be detected before it reaches a customer. A higher RPN indicates a critical failure mode that should be addressed urgently. This common reporting mechanism works well with an iterative design approach, allowing engineers and security personnel to bridge security findings with functional safety processes throughout a design lifecycle. Based upon the presence and simplicity of following the FMEA generation script, we believe that the VPTP allows operators to run automated tests to generate meaningful reports from observed behavior, satisfying our third research question.

Failure Mode and Effects Analysis (FMEA) Entry

Generated: 2025-05-06 14:04:42

Field	Value
Process Step/Input	Primary Test: Address Claim
Potential Failure Mode	Address claim kicks legitimate ECU off of the network
Potential Failure Effects	ECU is unable to control its dedicated features, and is also unable to transmit periodic status messages
Severity (1-10)	8
Potential Causes	Malicious node/attacker on network, or a misconfigured ECU
Occurrence (1-10)	1
Current Controls	Network gateway filtering messages with defined ruleset. Network segmentation.
Detection (1-10)	3
Action Recommended	Re-claim address if all (0-255) addresses have been claimed
Responsible	Embedded Security Engineer
Actions Taken	Firmware patch created 04-28-2025
RPN	24

Fig. 4. Failure Mode and Effects Analysis Output from the VPTP.

6 Testing and Evaluation

The prototype was connected to a fully-instrumented 2018 Freightliner Cascadia testbench filled with common ECUs relevant to heavy-duty vehicles, such as an engine ECU, brake ECU, and a cab ECU. SocketCAN captured baseline traffic for 30 minutes under ignition-on and idle conditions. Each attack module was then executed once, with automated metrics recorded.

6.1 Address-Claim Spoofing

The first test attack was the address claim, which is part of the SAE J1939 protocol and involves a node asserting its identity on the CAN bus by broadcasting a message to claim a specific Source Address (SA). In a properly functioning J1939 network, each ECU must have a unique SA to ensure proper communication and routing of messages. The Address Claim message (PGN 60928 or 0xEE00) contains a 64-bit NAME field that encodes information about the device's function, manufacturer, and instance. An address claim attack exploits the arbitration and address management behavior of the protocol. Due to J1939's lack of cryptographic authentication in address assignment, a malicious ECU can transmit crafted address claim messages to spoof the identity of another legitimate ECU or to force a conflict that causes the real ECU to relinquish its address. This can lead to denial-of-service conditions or enable more sophisticated spoofing and man-in-the-middle attacks as seen in Fig. 5.

Using the J1939 address-claim Metasploit module, we injected 200 claims per second for 15 seconds. The instrument cluster lost gauge readings after 3.2 s

on average, confirming reproducibility of the Burakova *et al.* attack [7]. These tests were confirmed to work on both test setups, therefore satisfying the second research question. In addition to visual results gathered from the 2014 Kenworth and 2018 Cascadia, the Pytest framework is used to provide corroborative evidence that each CAN address claim attack message is sent properly, seen in Fig. 6. The results indicated a 100% success rate in our automated process of testing the address claim, meaning all of the intended messages were transmitted onto the bus correctly as seen in Fig. 7 with the passing indication.

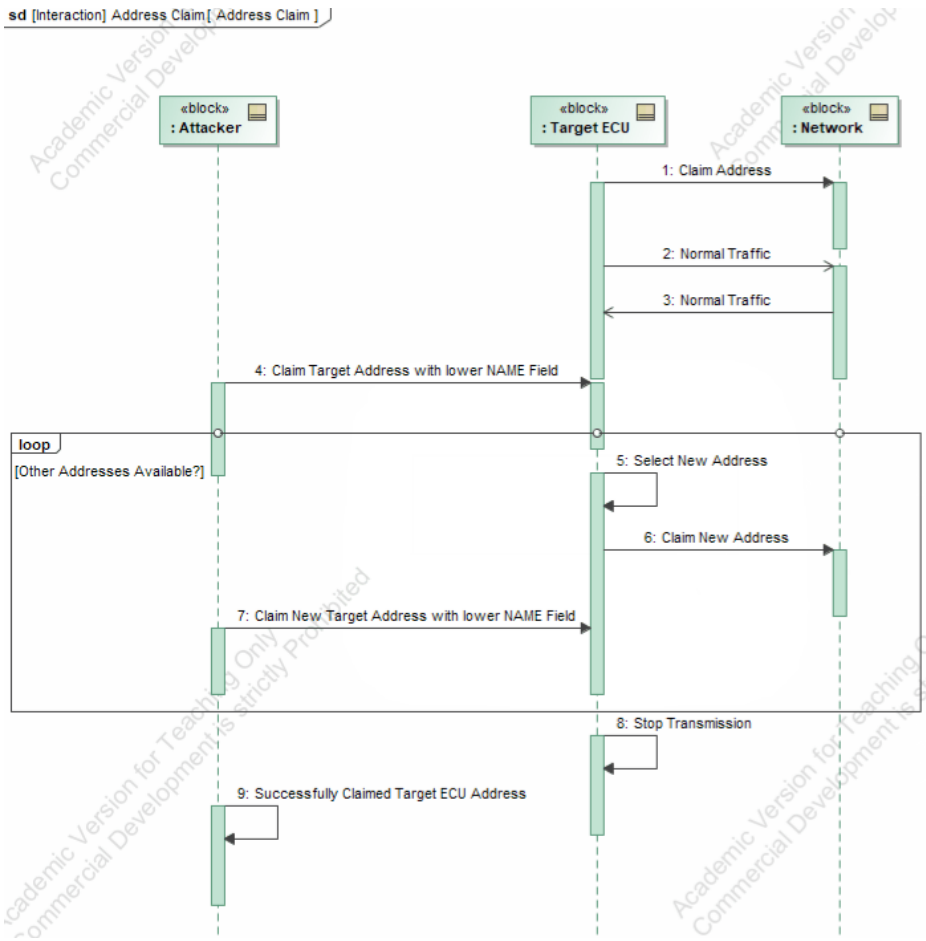

Fig. 5. Sequence Diagram of an Address Claim Attack Flow.

```python
import can
import pytest

START_ID = 0x18EEFF00
END_ID = 0x18EEFFFF
EXPECTED_COUNT = 256  # Inclusive of both start and end
DATA_BYTES = bytes([0x00] * 8)

@pytest.fixture
def test_address_claim_sequence(can_interface):
    """Test if the address claim script sends 256 correct frames from 18EEFF00 to 18EEFFFF."""
    seen_ids = set()
    timeout_seconds = 45
    listener = can.BufferedReader()
    notifier = can.Notifier(can_interface, [listener])

    try:
        import time
        start = time.time()
        while len(seen_ids) < EXPECTED_COUNT and (time.time() - start) < timeout_seconds:
            msg = listener.get_message(timeout=0.1)
            if msg and START_ID <= msg.arbitration_id <= END_ID:
                if msg.data == DATA_BYTES:
                    seen_ids.add(msg.arbitration_id)
    finally:
        notifier.stop()

    missing_ids = [hex(i) for i in range(START_ID, END_ID + 1) if i not in seen_ids]
    assert len(seen_ids) == EXPECTED_COUNT, f"Missing IDs: {missing_ids}"
```

Fig. 6. Automated Address Claim Functionality Verification Script.

```
beersc@vptp:~/pytest $ pytest test_addrclaim.py
============================= test session starts ==============================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /home/beersc/pytest
collected 1 item

test_addrclaim.py .                                                      [100%]

============================== warnings summary ================================
../../../usr/lib/python3/dist-packages/can/interfaces/__init__.py:38
  /usr/lib/python3/dist-packages/can/interfaces/__init__.py:38: DeprecationW
arning: SelectableGroups dict interface is deprecated. Use select.
    entries = entry_points().get("can.interface", ())

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================== 1 passed, 1 warning in 5.11s ========================
```

Fig. 7. Address Claim Testing Results.

6.2 Fuzzer Evaluation

A bit-flip fuzzer generating random CAN IDs and data at a baud rate of 500k bits/s corrupted DM1 diagnostic traffic. Additionally, a test using a previously-recorded log of CAN traffic on the network was used to replay the same messages onto the network. The effect of this replay style of attack involved the visual indications of diagnostic lights illuminated, various dials moving back and forth, and an audible beep heard during the attack. No permanent fault codes remained

after power-cycle, yet the test highlighted insufficient input validation. An image of this demonstrated in Fig. 8.

Fig. 8. Random Fuzzing Results on a 2018 Freightliner Cascadia Cab.

7 Discussion

7.1 Security Weaknesses Observed

- Malicious Message Effects: ECUs accepted the address claim and replayed CAN frames, causing emergent behavior such as a denial of service or enabling faults onto the instrument cluster.
- Lack of message authentication: Classical CAN frames are unsigned, enabling spoofing and denial of service.

The scope of this paper is limited to evaluating and verifying the VPTP system itself, without proposing mitigations for the demonstrated attacks. The evaluation demonstrates that automating and streamlining a security-based testing process for cyber-physical systems is both feasible and practical using the VPTP.

7.2 Comparison with Prior Platforms

Unlike proprietary bench testers, the VPTP is fully scriptable and facilitates customization, more cost-efficient ($250 in BOM costs), and supports heavy-vehicle standards often ignored by car-centric tools. Its automated report pipeline uniquely links cybersecurity evidence to safety artifacts.

8 Conclusion and Future Work

This paper introduced the Vehicle Penetration Testing Platform (VPTP), an open-source, portable, and extensible tool for assessing the cybersecurity posture of modern vehicular systems. By consolidating key attacks including CAN injection, J1939 spoofing, multi-packet abuse, and automated FMEA reporting—into a single, handheld device, the VPTP addresses critical gaps in existing vehicle security testing methodologies. Our evaluation demonstrates that the VPTP can reliably reproduce known attack scenarios from prior academic literature while providing structured, repeatable output for both functional and safety-oriented analysis. Automated generation of FMEA artifacts helps bridge the domains of cybersecurity and functional safety, aligning with regulatory trends like UN ECE R155. Compared to proprietary or lab-bound platforms, the VPTP enables flexible field-deployable assessments at a fraction of the cost. Its modular architecture and support for scripting, containerization, and open protocols make it suitable for OEM validation, academic research, and training scenarios alike.

Planned enhancements include:

1. Expanding protocol support to include LIN, FlexRay, and Automotive Ethernet.
2. Implementing GPS spoofing and Wi-Fi/Bluetooth fuzzing with SDR modules.
3. Adding passive sniffing and payload signature detection to aid intrusion detection system (IDS) development.
4. Incorporating CAN-FD authentication techniques to study mitigation efficacy.
5. Extending the FMEA engine to support additional threat modeling frameworks.
6. Full fledged, custom PCB with HAT+ protocol support for the Raspberry Pi 5 for streamlining the look and operation.
7. Incorporating baseline templates for maritime, aerospace, industrial, and other automotive industries.

The results validate that a low-cost, community-driven tool can meaningfully contribute to the ongoing improvement of vehicular cybersecurity and enable further collaboration between systems engineers, safety personnel, and cybersecurity researchers.

Acknowledgments

The authors thank the Systems Engineering Department at the University for laboratory access.

References

1. Ieee standard for ethernet amendment 1: Physical layer specifications and management parameters for 100 mb/s operation over a single balanced twisted pair cable (100base-t1). Standard IEEE Std 802.3bw-2015, IEEE Standards Association (Oct 2015). https://doi.org/10.1109/IEEESTD.2015.7321977, https://standards.ieee.org/standard/802_3bw-2015.html, approved 6 October 2015
2. Serial data communications between microcomputer systems in heavy-duty vehicle applications. Recommended Practice J1708_201609, SAE International (Sep 2016). https://doi.org/10.4271/J1708$_2$01609, https://saemobilus.sae.org/standards/j1708_201609-serial-data-communications-microcomputer-systems-heavy-duty-vehicle-applications, stabilized September 2016; Issued 1986
3. ISO/SAE 21434:2021 road vehicles – cybersecurity engineering. Standard ISO/SAE 21434:2021, International Organization for Standardization (2021), https://www.iso.org/standard/70918.html, accessed 2025-03-15
4. UN regulation no. 155 – cyber security and cyber security management system. Regulation UN R155, United Nations Economic Commission for Europe (2021), https://unece.org/transport/vehicle-regulations/wp29/un-regulations, accessed 2025-03-15
5. Road vehicles — controller area network (CAN) — part 2: High-speed physical medium attachment (PMA) sublayer. Standard ISO 11898-2:2024, International Organization for Standardization (2024), https://www.iso.org/standard/85120.html, edition 3
6. Road vehicles — local interconnect network (LIN) — part 3: Protocol specification. Standard ISO 17987-3:2025, International Organization for Standardization (2025), https://www.iso.org/standard/85127.html, edition 2
7. Burakova, Y., Hass, B., Millar, L., Weimerskirch, A.: Truck hacking: An experimental analysis of the SAE J1939 standard. In: USENIX Workshop on Offensive Technologies (WOOT). pp. 211–220 (2016)
8. Checkoway, S., et al.: Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of USENIX Security. pp. 447–462 (2011)
9. Clarke, R., Dorwin, D.: Advantages of open source over proprietary software. Tech. rep., University of Washington (2005)
10. Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. Black Hat USA white paper and slides (2015)
11. Roberts, A., et al.: A global survey of automotive cybersecurity validation and verification. SAE International Journal of Connected and Automated Vehicles (2023)
12. Sommer, F., Kriesten, R., Kargl, F.: Survey of model-based security testing approaches in the automotive domain. IEEE Access **11**, 55474–55514 (2023)
13. Spencer Beer: Cyber physical penetration testing framework. GitHub repository (2025), https://github.com/CPPTF
14. SystemsCyber: Ultimate truck hacking platform (uthp). GitHub repository (2023), https://github.com/SystemsCyber/UTHP
15. Wolf, M.: Combining safety and security threat modelling to improve automotive pen testing. Tech. rep., Fraunhofer AISEC (2019)