IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 July 2025, accepted 14 July 2025, date of publication 16 July 2025, date of current version 24 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3589867

RESEARCH ARTICLE

pwnobd: Offensive Cybersecurity Toolkit for Vulnerability Analysis and Penetration Testing of OBD-II Devices

ROBERTO GESTEIRA-MIÑARRO[1], IGNACIO GUTIÉRREZ[1], RAFAEL PALACIOS[1,2], AND GREGORIO LÓPEZ[1]

[1]Instituto de Investigación Tecnológica, Universidad Pontificia Comillas, 28015 Madrid, Spain
[2]Massachusetts Institute of Technology, Cybersecurity at MIT Sloan, Cambridge, MA 02139, USA

Corresponding author: Roberto Gesteira-Miñarro (rgesteira@comillas.edu)

ABSTRACT The research field of vehicle cybersecurity has experienced a significant growth in interest due to the attack surface that the information systems comprising a vehicle provides and the ever-expanding body of regulations that provide special focus on cybersecurity on vehicular systems. Of particular interest is the attack surface exposed by OBD dongles, wireless devices that connect to the vehicle's diagnostic port, whose access to the vehicle's CAN buses could potentially be exploited by adversaries. However, acquiring a vehicle for use in the security assessment of these devices may not be possible for the researcher. In this article, we propose a software tool, `pwnobd`, that assists in developing proof-of-concept attacks seeking to take advantage of the found vulnerabilities, alongside an architecture for a research and demonstration platform that provides a testbed for vulnerability analysis and penetration testing for attacks towards these devices. A small battery of tests is then performed on several diagnostic devices using this platform, along with a focused study on one such device, proving the potential benefit of such platform for security researchers.

INDEX TERMS Cybersecurity, car hacking, on-board diagnostics, embedded device, Bluetooth.

ACRONYMS

ADB	Android Debug Bridge.
API	Application Programming Interface.
CAN	Controller Area Network.
CVE	Common Vulnerabilities and Exposure.
DNS	Domain Name System.
ECU	Electronic Control Unit.
GATT	Generic ATTribute Profile.
HCI	Host Controller Interface.
I²C	Inter-Integrated Circuit.
IoT	Internet of Things.
JTAG	Joint Test Action Group.
M2M	Machine-to-machine.
OBD	On-board Diagnostics.
SPI	Serial Peripheral Interface.
SWD	Single Wire Debug.
UART	Universal Asynchronous Receiver/Transmitter.
USB	Universal Serial Bus.

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

I. INTRODUCTION

The means of transport of common use within our society have increasingly incorporated functionality dependent on the capabilities provided by information systems in the last decade. On modern vehicles, infotainment systems, external cameras and even the basic mechanical control systems are part of a network of ECUs (Electronic Control Units). These are heterogeneous embedded systems with diverse capabilities that work together to orchestrate and supervise the physical processes that happen in a vehicle as a response to user and environmental inputs. New functionalities and paradigms, such as interoperability with mobile devices, the rise of electrical vehicles and assisted or automated driving, result in an exponential increase in the complexity of these embedded systems. These factors deeply underscore the necessity of cybersecurity applied to vehicles, now turned into small industrial control systems with a significant attack surface. Furthermore, as vehicle-to-everything communications, cooperative intelligent transportation systems and self-driving technologies continue to evolve, the need

FIGURE 1. Examples of commercial OBD dongles: (1) BlueDriver (model LSB2) from Lemur Vehicle Monitors, (2) Bluetooth OBDII Reader from Bafx Products, (3) generic ELM327-compatible device.

for robust cybersecurity measures becomes increasingly critical [1], [2].

From the perspective of an adversary, one of the attack surfaces posing the highest potential for impact is the direct access to the vehicle's CAN (controller area network) buses, through which its various ECUs communicate between each other. The OBD (on-board diagnostics) system, whose inclusion and easy access for the purpose of vehicle maintenance is regulated in the European Union under the EU Regulation 2018/858 [3], often allows to access a subset of the vehicle's CAN buses. While the implementation of an OBD port is mandatory under this regulation, research suggests an alarming lack of secure-design principles such as network segmentation [4].

As a result, access to OBD ports can be leveraged in order to exploit vulnerabilities found within these systems under a variety of attack scenarios and with a diverse spectrum of potential impacts. Access to the CAN bus of a vehicle through the OBD port may allow, for example, interact with the car's windows in order to steal valuables inside [5], unlock the vehicle by impersonating the ECU responsible for authenticating the electronic ignition key [6], and even disable the brakes while driving [7], [8], [9], [10]. However, access to the OBD port tends to be limited to the interior of the vehicle, itself constituting a security boundary, thus limiting the usefulness of such an attack vector in most scenarios.

Outside its use in vehicle maintenance done by service centers, such diagnostic ports have demonstrated other popular uses, many of them largely materialized in the so-called OBD dongles (Fig. 1): small devices that users can connect to the OBD port in order to gather information and telemetry from the vehicle. Generally, interaction and data acquisition is done with the help of a mobile application which communicates with the device either through Bluetooth or WiFi, allowing self-service access from the driver and/or owner of the vehicle, or through the use of a cellular modem (M2M, machine-to-machine) for use cases where centralized management is required. On the other hand, the direct access to the OBD port (and by extension to the CAN bus) that such devices rely on may potentially extend the attack surface towards the nearby proximity of the vehicle [5], [11]; if the device uses cell connectivity instead, this may even allow for remote exploitation of such devices over the Internet [12].

In vehicle cybersecurity research, one of the major challenges is the high cost associated with acquiring a physical vehicle to serve as a testbed for prototyping and demonstrating security attacks. As a result, simulation testbeds have become essential tools, offering cost-effective and accessible alternatives that allow researchers to evaluate and test the security of vehicular systems without the need for expensive real-world hardware. In addition, while it is possible to operationalize attacks in an *ad-hoc* manner, the resulting proof-of-concept programs may end constrained to a single vulnerable device model, further complicating scalable testing of the same vulnerability type against multiple devices. Thus, extensible software frameworks can help the researcher develop and launch device-agnostic attacks, progressing towards faster and more cost-effective operational implementations.

We propose `pwnobd` [13], a tool that comprises the aforementioned features. It was presented in Black Hat Europe 2024 (Arsenal) [14], one of the flagship hacking conferences worldwide, which highlights its research interest within academia and industry regarding automotive cybersecurity.

The rest of the article is organized as follows: Section II further describes the identified requirements which in turn drive the design proposal. Section III describes the design proposal itself, contemplating the testbed platform itself and the `pwnobd` tool. Section IV contextualizes the proposed design through a battery of tests performed on 3 different devices using the platform and a case study around CVE-2016-2354 [15] vulnerability, describing how the proposed platform and the capabilities it offers could be utilized to identify similar vulnerabilities; the resulting findings are also discussed. Section V compares and contrasts this research with previous academic literature, identifying the contributions of this proposal towards the wider body of academic literature and other work within the industry. Finally, section VI provides conclusions and outlines future research lines.

II. REQUIREMENT ANALYSIS

Effective penetration testing of an information system requires consideration of its behavior, i.e. how it collects, processes and interacts with its environment and the data it provides. An OBD dongle communicates primarily with the CAN bus of the vehicle through its OBD port and interacts with other systems and devices through interfaces such as Bluetooth, WiFi or cell connectivity (M2M) following the logic defined in its firmware. Under the scope defined on this article, the primary objective under consideration is to audit

devices that communicate with an Android-based device running the appropriate app, which in turn communicates with the device via Bluetooth.

In order to interact with the device, its firmware exposes a high-level API in the form of an application protocol overlaid on top of the corresponding wireless interface, through which the application can collect and perform the desired actions. Many devices implement the proprietary protocol offered by the ELM327 chip originally developed by ELM Electronics. As a result, it has become a *de-facto* standard, with a diverse ecosystem of devices and applications which support this protocol. Other devices implement proprietary protocols, along with specific mobile applications that interact with these devices.

In general, a significant number of these devices communicate with the ECUs of the vehicle via the CAN bus protocol (or, on older vehicles, through other protocols like ISO 9141-2 [16]) in order to collect telemetry and/or diagnostic messages from the vehicle. This replicates a functionality similar to that offered by specialized tools employed in automobile repair shops for diagnosing failures within the vehicle's subsystems.

Derived from this context, we identify various capabilities which the platform design should provide, focused on both the attack surface under study and the types of attacks that the researchers seek to perform. We consider the following attack surfaces under scope:

- The mobile application that interacts with the device in a vendor-supported manner.
- The underlying communication layers (Bluetooth, WiFi, etc.) through which the application seeks to communicate with the device.

Through the aforementioned attack surfaces, the platform should be able to perform the following attacks:

- Reverse-engineering.
- Low-level control, both of the CAN bus and of OBD dongles themselves.
- Man-in-the-middle interception of communications.
- Fuzzing of high-level protocols.
- Exploitation of firmware-level vulnerabilities.

Additionally, we identify the following non-functional requirements to consider in the platform's design:

1) **Accessibility**, providing ready-to-use capabilities and tools which provide a base allowing to perform attacks and collect results in less time.
2) **Efficacy**, assisting security researchers in their audit and research workloads through the automatization of simpler attacks, allowing increased effort to be employed on more complex attacks.
3) **Extensibility**, in order to support the continuous evolution of the platform as new technological innovations reach the commercial market.
4) **Cost**, allowing a diverse audience to benefit from the capabilities offered by the platform.

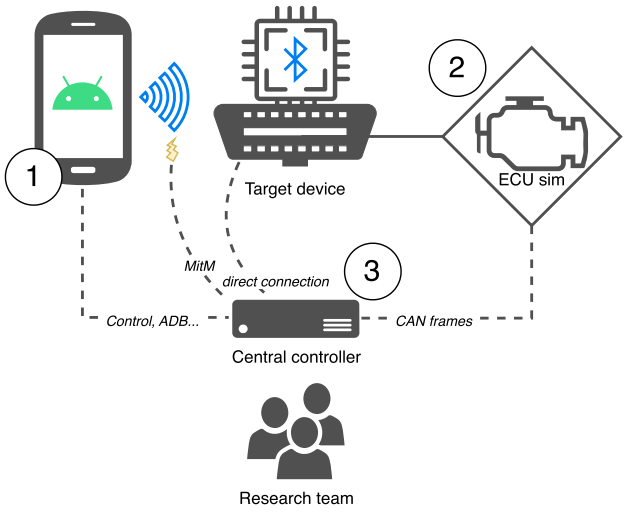

FIGURE 2. Design of the proposed platform with the primary elements.

III. PROPOSED DESIGN

As previously stated, the proposal introduced by this work has two components: a hardware testbed platform and the `pwnobd` companion software.

A. HARDWARE PLATFORM

In order to address the identified requirements, a design is proposed with these components (see Fig. 2 and Fig. 3):

1) An Android device on which the manufacturer-provided application for communicating with the target device is installed. This allows the researcher to test the target device as a part of the wider system it belongs to. The current implementation of this platform utilizes a Xiaomi Redmi 9 in order to fill this role.
2) An ECU simulator, electronic device with an OBD-II port to which the target device is connected. The ECU simulator presents itself to the device as one of the vehicle's electronic control units, providing partial emulation of the messages that a vehicle could exchange through its communication buses. The current implementation of this platform utilizes a simulator device fabricated and distributed by Lianghung Co [17] for this purpose. This device allows for the simulation of multiple protocols in addition to the CAN standard, with an additional network tap capability over serial communication for capturing exchanged data over the communication bus in real-time. It also provides easy access to the CAN bus itself through exposed through-hole pads within the circuit board, allowing for logic analysis and decoding using an oscilloscope.
3) A central controller to centralize the workload and tools employed in the analysis and experimentation with the devices under test and the rest of the platform. The

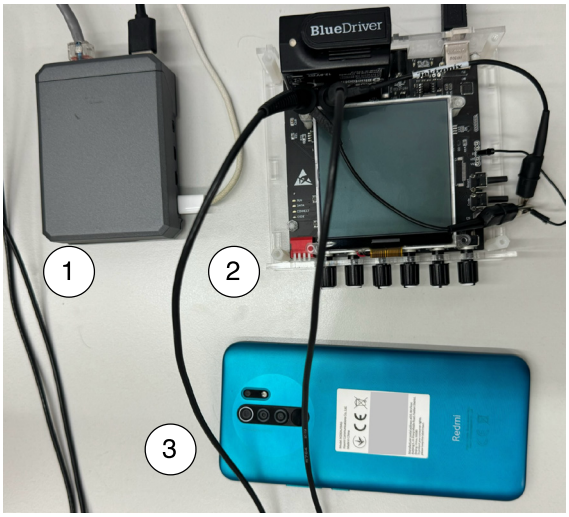

FIGURE 3. Implementation of the platform at the laboratory. (1) Central controller, a Raspberry Pi 4B with 8 GBs of RAM; (2) ECU Simulator, Lianghung Co [17], connected to a BlueDriver LSB2 OBD dongle and oscilloscope probes; (3) Android device, Xiaomi Redmi 9.

current version of this platform utilizes a Raspberry Pi 4B (8 GB of RAM), although it is possible to substitute this device with another of similar characteristics.

In regards to (1), the debugging and analysis capabilities offered by the Android operating system allow to study the interactions between the applications running on the device, intercept communications and firmware updates and even modify the application's behavior through patches. This allows to experiment with a diverse repertoire of attack scenarios, from man-in-the-middle attacks on both the communication with the target and communication with the Internet, to vulnerability exploitation performed on the OBD dongle or on the mobile application itself.

On the other hand, the use of an ECU simulator (2) simplifies the platform's design by averting the use of a real vehicle as a hard requirement, reducing the platform's total cost and allowing for riskier attacks that could compromise a real vehicle's availability and/or reliability.

The platform's various components are connected to the central controller (3), and the researchers interact with these devices through the tools available within it. For example, the central controller can connect to the Android device through USB or by using the WiFi access point that the central controller itself provides, and interact with it through the ADB (Android Debug Bridge) protocol. Scripts are provided which expedite the connection setup process for wireless debugging, automatically detecting and connecting to suitable devices available on the network via multicast DNS. Simultaneously, the central controller can also monitor the ECU simulator's network tap and collect CAN communications from it, which can aid in attack verification and debugging.

Along with these capabilities, tools are provided within the central controller to interact with the platform's components, develop and perform attacks and collect results. One such tool is `pwnobd`, which will be further described in subsection III-B. Another tool that was provisioned within the central controller is MobSF [18], an open-source software package that provides static and dynamic analysis capabilities for Android applications. Through the use of scripts and automations to manage the system's components, the platform seeks to standardize research procedures and improve efficiency, allowing the researcher to achieve results faster. In terms of performance, the proposed platform implementation is sufficient to meet our requirements.

In addition, the central controller exposes several hardware interfaces, including WiFi, Bluetooth, USB, SPI, I^2C and UART, among others. This allows integrating with additional hardware to arbitrarily expand and modify the platform's capabilities as the situation requires. For example, attacks on the Bluetooth communication layer could be performed either through the central controller's built-in Bluetooth adapter or through the use of external specialized devices. The central controller could also directly interact with the device under test through physical debugging interfaces such as SWD, JTAG or UART.

B. PWNOBD FRAMEWORK

The hardware platform described previously can already provide significant value as a safe, cost-effective environment for security research on OBD devices by itself. However, it alone cannot fulfill all the stated requirements; for example, the platform may not provide increased efficacy as opposed to an *ad-hoc* testbed.

To fill this gap, this article also introduces `pwnobd`, a Python-based framework from which researchers can develop attacks directed towards OBD devices in a device-agnostic manner. The developed attacks are then exposed as part of a command-line tool from which an operator can scan and connect to supported devices and launch and manage attacks at scale. This enables the automation of simple attacks and provides assistance in the development of more complex attacks.

In order to allow the researcher to develop device-agnostic attacks, `pwnobd`'s software architecture relies on an application of the bridge software design pattern [19], where the implementation of an attack does not need to rely on device-specific implementation details, but instead targets a framework-common set of interfaces, which can then be specified as required within the attack's definition (see Fig. 4)).

An example of a common interface is `SendCan`, which provides functions for sending arbitrary CAN frames to the CAN bus through the device. In order to add support for a new device to `pwnobd`, a device driver is implemented. Device drivers allow to set up and teardown persistent connections to the device under study, providing the necessary logic to maintain the connection and interact with the device, along with implementations for any given common interfaces that it may support. By specifying the device driver's concrete

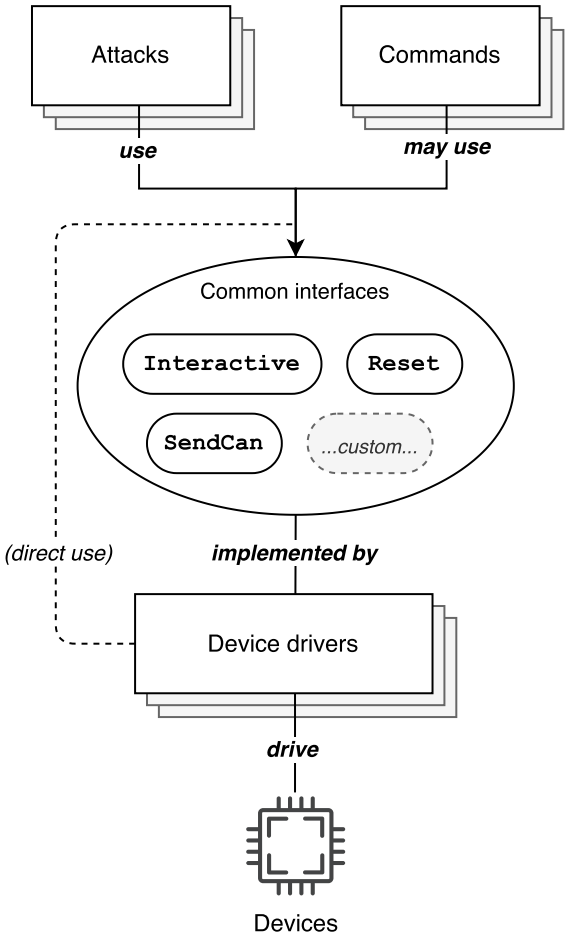

FIGURE 4. Architecture of the core abstractions provided by pwnobd.

class as a requirement, device-specific attacks (for example, fuzzing attacks against the device's high-level API) may be similarly implemented. Common interfaces may also provide functionality outside attack implementations; for example, implementing the `Interactive` interface allows an operator to open an interactive console from which they may send arbitrary commands to the device, and devices supporting the `Reset` interface can be returned to their initial state automatically after certain actions or manually through the use of a command.

An extensible system of scanners is also provided as part of the framework, which allows device driver implementations to provide additional logic to scan for and identify nearby devices that may be compatible with such driver.

These elements constitute the software framework through which attacks targeting OBD devices can be developed. The user experience principles for this interface take inspiration from industry-standard penetration testing and red-teaming tools such as Metasploit [20] and Sliver [21], seeking to provide the operator with effective tools which can be used to test the previously-described capabilities.

Installation and usage instructions for `pwnobd`, together with example code snippets, can be found in the associated GitHub repository [13].

IV. USE CASES

In order to evaluate the suitability of the platform as a whole, two classes of evaluation work were performed:

1) A suite of general tests performed against various devices using the platform.
2) A more detailed scenario describing a follow-through of the research done as part of CVE-2016-2354.

A. GENERAL PLATFORM EVALUATION

In order to evaluate the general capabilities provided by the platform, several tests were conceived and carried out against the three OBD devices pictured on Fig. 1. These include:

1) **Late connection test**, which seeks to identify whether the device accepts or reacts to connections done after an extended idle period of 10 minutes. This test seeks to model the device's susceptibility to be exploited in attack scenarios where the user of the OBD dongle forgets or does not wish to unplug the OBD dongle from the vehicle after its use, which would allow an attacker to approach the vehicle while unattended, connect to the plugged-in OBD dongle and perform subsequent actions on the objective vehicle.
2) **Simultaneous connection test**, looking to ascertain whether multiple clients can simultaneously establish a connection and interact with the device at the same time. This would allow an attacker to interact with the vehicle through the dongle even if a legitimate device was already connected, which could impact the reliability of the device. Two different clients are used for this test: the Android phone, posing as the legitimate client, and the central controller itself, posing as the attacker. If the connection is successful, an attempt to inject messages to be sent to the vehicle is done in order to rule out additional security measures.
3) **Arbitrary CAN frame injection**, i.e. determining whether the device is able to send arbitrary CAN commands. Most OBD dongles, by default, do not send pure CAN frames on request, but instead wrap the data into an ISO 15765-2 message [22] which is then sent (often as multiple CAN frames, if necessary) using a fixed CAN arbitration ID. For the purposes of this study, an injection is considered successful if a way is found to send CAN frames of arbitrary content without ISO 15765-2 formatting [22] and with an arbitrary CAN arbitration ID. In order to verify that injection is successful, the ECU simulator's sniffer is used first to determine successful injection of an arbitration ID, then the message content is verified on an oscilloscope. The ability to inject CAN frames this way can significantly increase the attacker's capabilities and potential impact once initial access to the OBD dongle is established.
4) **CAN fuzzing simulation**, in which a subset of CAN frames from the *Car Hacking: Attack & Defense Challenge 2020 Dataset* [23], [24] were replied to the simulator using the device. This test is used as a simple

TABLE 1. General testing results.

	(1)	(2)	(3)	(4)
BlueDriver LSB2	✗	✓	✗	✗
BAFX OBD Reader	✗	✓	✗	✗
Generic ELM327 device	✗	✓	✗	✗

exercise of the abstraction capabilities that pwnobd's framework provides: the concrete, per-device CAN injection logic is provided within the implementation of the SendCan common interface for the corresponding device drivers; whereas the dataset loading and general replaying logic were implemented separately as an attack, targeting devices that implemented the aforementioned common interface.

The results of these tests, as performed on the current implementation of the platform, are available on Table 1.

All devices tested were not vulnerable to allowing multiple simultaneous connections, which they should block during the Bluetooth connection establishment phase. All devices seem to accept connections no matter how much time the device has spent idle. While BlueDriver devices were previously considered not vulnerable since CVE-2016-2354 [15] was remediated [25], [26], we have identified that both firmware versions 2.59 and 2.60 (the latest as of testing) of the BlueDriver LSB2 device no longer provide this protection, as the device was shown to allow connections and command execution even after the defined idle period of 10 minutes.

Finally, all devices were found to be able to send arbitrary CAN frames. While the procedure to do so for ELM327-based devices is known and documented by ELM Electronics [27], the discovery of this capability within the newest firmware versions of BlueDriver LSB2 devices is particularly novel. The process of identifying this capability is described as part of the CVE-2016-2354 case study in subsection IV-B.

B. PLATFORM CASE STUDY: CVE-2016-2354

To contextualize the potential utility of this work, a research scenario is proposed using a real vulnerability found in a commercial OBD device. Through this scenario, we identify several applications of the proposed design where the platform and software can provide benefits for researchers.

On April 2016, research performed by Dan Klinedinst, by then one of the threat and vulnerability researchers employed by CERT/CC (Carnegie Mellon University), identified that the BlueDriver devices manufactured by Lemur Vehicle Monitors did not enforce authentication on the Bluetooth interface, from which the target vehicle's telemetry could be read using the manufacturer-provided mobile application [15]. An adversary within proximity of a vehicle that was left with a BlueDriver device plugged in could potentially connect to the device via Bluetooth and send arbitrary commands to the target vehicle's CAN bus using the OBD port through which the device interacts with the vehicle. The OBD device thus becomes a wireless entry point through which additional exploitation can be performed against the car's subsystems. It should be underscored that this exploitation would not require physical access to the interior of the vehicle, and therefore could potentially be performed without access to the car key.

The discovery of this vulnerability received press coverage [28], [29], and Lemur Vehicle Monitors quickly released updates for both its mobile application and its device firmware. The primary remediation measure, as stated by Lemur Vehicle Monitors support agents [25] and later verified in published research [26], caused the device to reject new connections after a period of 60 seconds, forcing users to physically disconnect and reconnect the device from the vehicle's OBD port in order to reset this timer. In addition, encryption was implemented on the communication protocol between the mobile application and the device, and the underlying protocol was changed to limit the device's own capabilities through a list of permitted actions that could be performed by the client.

As part of the research process of this vulnerability, the platform could provide benefits on various activities:

1) The researcher can install the manufacturer's proprietary mobile application on the platform's Android device. Using the platform's provided static and dynamic analysis platforms for Android applications, the researcher could identify the lack of authentication within the Bluetooth communication layer.
2) Likewise, by analyzing the application's logic and intercepting Bluetooth communications between the application and the device, the researcher could determine that the device can be directed to send arbitrary CAN frames through the device's OBD port, through which the vehicle's safety could ultimately be compromised.
3) By reverse-engineering the communications protocol, modifying the application or even identifying a way to alter the device's firmware itself through a malicious firmware update, it would be possible to send arbitrary commands to the device. This would allow, among other attacks, the development of tailor-made fuzzing attacks against the device itself, for which IoT-specific techniques exist in academic literature [30], [31]. The success of these attacks could be determined by examining the communications performed from inside the CAN bus, which are monitored and relayed to the central controller by the ECU simulator.

Additionally, the platform would allow to validate and evaluate the subsequent remediations performed by the manufacturer, verifying that the original vulnerabilities are mitigated and that additional vulnerabilities are not introduced. Through the offered capabilities, the platform can thus assist in various phases of the research process.

As part of the evaluation of the proposed platform, it was used to evaluate the stated remediations as applied in modern

firmware versions of the BlueDriver OBD dongle (along its companion Android application).

1) The device was set up on the platform's ECU simulator, the application was installed on the Android device and the system's behavior as intended was tested. Through this process, the device passively captures legitimate Bluetooth traffic through the use of Android's Bluetooth HCI (Host Controller Interface) snoop log debugging facilities (thereby averting the need to perform an over-the-air man-in-the-middle attack against the Bluetooth communication itself), which is later downloaded onto the central controller through ADB for further study. Initial traffic analysis revealed a stream of encrypted communications performed through Bluetooth GATT (Generic ATTribute Profile) characteristics.
2) The application was analyzed using the automated tools provided by MobSF. The resulting decompiled source code and metadata provided useful information about the application's security posture. This includes the use of a software packer in order to protect the application's intellectual property. No effort was put into breaking this protection, as the researchers were able to identify and reverse-engineer the cryptosystem responsible for encryption and decryption of in-transit traffic outside the protected code.
3) Once the cryptosystem was reimplemented, traffic analysis revealed a command-based high-level API. Captured traffic logs were sufficient to implement a basic `pwnobd` device driver for this device[1] which could initialize the device and allow the operator to interactively send and receive these commands.
4) Due to the significant amount of logic that the Android application had and the protection of relevant code and assets, further reverse-engineering of the application did not yield additional observations. Through traffic analysis, it was identified that all commands started with a common prefix, in a manner similar to the use of AT commands in the ELM327 protocol [32]. In order to find additional commands, the researchers implemented a brute-force attack against the command-based high-level API using `pwnobd` framework. The attack relies on an error-based side channel to identify valid commands. Several additional commands were unearthed as a result.
5) In order to identify the actions that the identified commands may result in, researchers utilized the interactive mode provided by the `pwnobd` software to quickly experiment with the changes caused to subsequently sent CAN frames; this included manually attempting to specify further arguments. Through this effort commands were quickly identified that allowed to arbitrarily modify the CAN arbitration ID and to disable ISO 15765-2 formatting.

Thus, it was identified that it is possible to send arbitrary CAN frames using a BlueDriver LSB2 device on firmware versions 2.59 and 2.60, thereby increasing the potential for ECU exploitation.

[1]In order to protect the intellectual property of Lemur Vehicle Monitors, the released source code for `pwnobd` does not include any code relating to the BlueDriver device.

V. DISCUSSION

This section compares the current work with other research works, identifying differences and contributions of the current proposal.

A. VEHICLE SIMULATION TESTBEDS

Some previous literature covers the development of simulation environments for vehicles that contemplate CAN-level ECU emulation, even going so far as to consider on-board diagnostics emulation. One such example is VEWE by Alvear et al. [33], a software-based end-to-end simulation which, unlike the work proposed in this article, 1) emulates vehicle behavior which is then fed to the simulated ECU and 2) emulates the OBD-II dongle itself, all in order to aid in developing and testing new devices that use OBD-II ports.

Works that focus on vehicle cybersecurity can also be found, such as the recent VISE by Kabir and Ray [34], a digital-twin-based platform which is able to emulate a large amount of ECUs, sensors and actuators for the purpose of simulating the complex effects of attacks within the CAN bus of vehicles. While our work requires the use of a hardware-based ECU in order to be able to physically plug in and interact with the devices under study, it is also able to use simpler off-the-shelf hardware that may only simulate an engine ECU. This is because our work is focused on assessing the security of OBD dongles themselves as a potential means to carry these CAN attacks wirelessly through an increased initial attack surface, and not CAN attacks themselves.

B. CAN-BASED ATTACKS

We have previously provided an overview of various potential real-world impacts on vehicle safety and security that can result from access to the CAN bus by an attacker [5], [6], [7], [8], [9], [10], [11]. While there are many attack scenarios that may be carried out within the CAN environment of a vehicle [4], [35], we focus mainly on attacks which can be carried out by sending arbitrary CAN frames through the OBD port. This includes spoofing and fuzzing of other ECUs exposed to the objective CAN bus, among others.

There exist several means to interact with the CAN bus from an offensive perspective which extend beyond the use of OBD ports. Which tool works best for a given situation will depend on how far the adversary is on the weaponization process (as described in Lockheed Martin's Cyber Kill Chain [36] cyber-attack model). Early on in the development lifecycle, significant flexibility is required in order to quickly iterate and achieve results faster. Research may be thus better performed through off-the-shelf OBD dongles driven

TABLE 2. Feature summary of CAN and OBD tools.

Tool	Features
`python-can` [37]	Python library that provides common abstractions to different hardware devices, and a suite of utilities for sending and receiving messages on a CAN bus
`CANalyse` [38]	Requires a hardware implant that sniffs, analyzes and uses the car information to command & control certain functions of a vehicle through a Telegram bot
ATG [35]	Vehicle CAN bus packet analyzer and attack packets generator that enables researchers to explore automotive cyber-physical system rapidly, to analyze different attack modes in ECUs or to generate a dataset for later analysis
`HWBridge` [39]	General-purpose Metasploit module to connect to hardware devices such as CAN sniffers, among others. It is a functional interface to use the SocketCAN capabilities of Linux

by *ad-hoc* code running on the researcher's computer. This is already possible through software libraries and tools that allow researchers to interactively or programmatically interact with CAN buses through various means, such as `python-can` [37]. As weaponization progresses, however, an effective application of the exploit may be done through the purpose-built hardware devices [6]. Table 2 summarizes selected open-source tools that can be used for automotive cybsersecurity research related to OBD and CAN.

The open-source body of software works also provides tools that focus on offensive exploitation of CAN buses. Some tools provide implementations of replay attacks ready for operational use, such as `CANalyse` [38]; others assists the researcher with traffic analysis and reverse-engineering such as ATG [35]. One of the most notable examples in this regard is Metasploit's own support for hardware devices, codenamed `HWBridge` [39], [40], which was originally contributed to the Metasploit Framework codebase, while not providing many advantages as of currently.

Our proposed tool `pwnobd` provides increased flexibility in the development of device drivers and attacks by providing more granular common interfaces, of which a combination can be independently implemented by a given device driver, and similarly required by an attack implementation. Our work also attempts to provide a more tailored and focused user experience that can adequately cover various operational stages as specifically needed for vehicle cybersecurity.

C. LIMITATIONS AND POTENTIAL IMPROVEMENTS

During evaluation of the platform as was implemented, we identified several problems regarding the hardware procurement choices made which limited the platform's potential capabilities. Future replications and other implementations of the platform described in this work may benefit from taking the following observations into consideration.

1) We did not use a rooted Android device, which limited data collection and reverse-engineering capabilities. For example, Bluetooth HCI snoop logs could not be retrieved in real-time; instead, a time-consuming bug report request would have to be triggered through ADB, which would in turn contain the requested logs. Additionally, lack of rooting limited the researchers' ability to perform dynamic analysis, which limited the scope of the research given the heavy protections used by some of the tested applications, such as code packing and obfuscation.
2) The ECU simulator always assumes incoming packets use ISO 15765-2 format, an assumption which may not necessarily be true. As a result, while testing arbitrary CAN frame injection and sniffing traffic using the ECU simulator, the data section of frames collected was often corrupted because of CAN frames that did not include valid ISO 15765-2 format. Common examples of this corruption include data truncation and over-padding. In order to determine the root cause of this problem and independently verify the results observed in subsection IV-A, the CAN bus itself was probed with an oscilloscope (see Fig. 3) and we verified that the CAN frames data section had the full data content as requested.

VI. CONCLUSION

This work has proposed, in concert with a set of functional and non-functional requirements, a design for a demonstration and testing platform for cyberattacks directed to OBD dongles, along with companion software `pwnobd`, seeking to provided cybersecurity researchers increased productivity in the process of auditing and analyzing vulnerabilities within those devices. Its utility was illustrated through the analysis and re-exploration of an industry-relevant case study, along with the evaluation of various devices within a basic test framework seeking to identify the target devices' susceptibility to become an entrypoint for further CAN-based vehicle exploitation. Furthermore, our proposed tool `pwnobd` [13] was presented in Black Hat Europe 2024 (Arsenal) [14], one of the flagship hacking conferences from all over the world.

As previously exposed, a common application of OBD dongles is to provide users with additional vehicle telemetry and diagnostics, in a way similar to the canonical use of such a diagnostic port. This may often be done in order to aid repair and maintenance efforts by both professional mechanics and users who may want to perform vehicle self-repair. These devices are also highly useful in various research fields involving vehicles, as they allow researchers to collect varied, precise and detailed real-time data from the vehicle itself. Even in consideration of the fact that these devices constitute a fairly effective and inexpensive solution for many applications, the potential for their malicious, safety-compromising misuse underscores the need for user education to develop and maintain a safety-oriented mindset towards their deployment and use, as with any other tool that may be used to work on a vehicle.

While there exist effective security mitigations, such as adding connection timeouts or CAN frame allow-lists, that vendors may follow in order to reduce the risk of vehicular exploitation being performed through their devices, users should consider the risk that connected OBD devices may pose on vehicle assets as part of their threat modeling and take precautions in order to limit the potential for exploitation, such as unplugging the device while it is not actively being used. Additionally, vehicle manufacturers should consider researching and deploying mitigations against the exploitation of ECUs vulnerabilities, particularly those that may impact the safety envelope, while balancing the potential use cases that OBD-II ports can provide. Some potential remediations for OBD may also not always be practical, such as limiting the content of messages that the device may be permitted to send through the CAN bus, as it may drive up the hardware requirements of such a device due to the need to store a database of allowed CAN frame contents to be sent per vehicle model within the device itself. Such mitigation must also be part of a secure firmware update mechanism as the only means to update this database, as otherwise it could be subverted by direct modification of either the database or the device firmware itself.

Future work may explore expanding the breadth of tests performed on a wider park of devices, potentially applying relevant methodologies such as IoTSF [41] and BSAM [42]. Additionally, further expansion of the platform may be performed, for which there's particular interest towards providing capabilities for Bluetooth connection interception and hijacking as an attack scenario. Additional software components and further integration between them within the platform could also be provisioned and experimented with, to increase the platform's capabilities. Finally, there's also potential for expanding the proposed architecture for analysis of a wider variety of devices within the wider scope of the Internet of Things.

REFERENCES

[1] A. Chowdhury, G. Karmakar, J. Kamruzzaman, A. Jolfaei, and R. Das, "Attacks on self-driving cars and their countermeasures: A survey," *IEEE Access*, vol. 8, pp. 207308–207342, 2020, doi: 10.1109/ACCESS.2020.3037705.

[2] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 546–556, Apr. 2015, doi: 10.1109/TITS.2014.2342271.

[3] Council of European Union. (May 2018). *Regulation (EU) 2018/858 of the European Parliament and of the Council of 30 May 2018 on the Approval and Market Surveillance of Motor Vehicles and Their Trailers, and of Systems, Components and Separate Technical Units Intended for Such Vehicles, Amending Regulations (EC), no. 715/2007 and (EC), no. 595/2009 and Repealing Directive 2007/46/EC (Text With EEA Relevance)*. [Online]. Available: http://data.europa.eu/eli/reg/2018/858/oj/eng

[4] 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 *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2010, pp. 447–462. [Online]. Available: http://ieeexplore.ieee.org/document/5504804/

[5] A. Luo and S. Hsieh. (Jul. 2023). *Remotely Hacking a Car Through an OBD-II Bluetooth Dongle*. Automot. Secur. Res. Group. [Online]. Available: https://www.youtube.com/watch?v=f19BNgVrWQ

[6] K. Tindell. (Apr. 2023). *CAN Injection: Keyless Car Theft*. [Online]. Available: https://kentindell.github.io/2023/04/03/can-injection/

[7] A. Greenberg. (Jul. 2015). *Hackers Remotely Kill a Jeep on the Highway-With Me in it*. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[8] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," presented at the Black Hat USA, 2015, pp. 1–93. [Online]. Available: https://www.youtube.com/watch?v=MAcHkASmXEc

[9] C. Valasek and C. Miller, "Adventures in automotive networks and control units," IOActive, Seattle, WA, USA, Tech. Rep., 2014. [Online]. Available: https://www.ioactive.com/wp-content/uploads/pdfs/IOActiveAdventuresinAutomotiveNetworksandControlUnits.pdf

[10] 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 *Proc. 20th USENIX Conf. Secur.*, San Francisco, CA, USA, Aug. 2011, p. 6.

[11] R. Gesteira-Miñarro, G. López, and R. Palacios, "Revisiting wireless cyberattacks on vehicles," *Sensors*, vol. 25, no. 8, p. 2605, Apr. 2025, doi: 10.3390/s25082605.

[12] R. Veredas and Y. Mehaboobe. (2023). *Attacking Vehicle Fleet Management Systems*. [Online]. Available: https://www.youtube.com/watch?v=FycVqUvicJw

[13] Nnubes256. *Pwnobd*. Accessed: Jul. 17, 2025. [Online]. Available: https://github.com/Nnubes256/pwnobd

[14] I. Gutiérrez and R. Gesteira-Miñarro. (2024). *Pwnobd: Offensive Cybersecurity Toolkit for Vulnerability Analysis and Penetration Testing of OBD-II Devices*. Black Hat Eur. 2024 (Arsenal). [Online]. Available: https://www.blackhat.com/eu-24/arsenal/schedule/index.htm#pwnobd-offensive-cybersecurity-toolkit-for-vulnerability-analysis-and-penetration-testing-of-obd-ii-devices-42009

[15] D. Klinedinst. (2016). *CERT/CC Vulnerability Note VU #615456*. [Online]. Available: https://www.kb.cert.org/vuls/id/615456

[16] *Road Vehicles—Diagnostic Systems—Part 2: CARB Requirements for Interchange of Digital Information*, Standard ISO 9141-2:1994, ISO Central Secretary, Geneva, Switzerland, 1994. [Online]. Available: https://www.iso.org/standard/16738.html

[17] Lianghung Co. *OBD-II ECU Simulator CAN BUS*. Accessed: Jul. 17, 2025. [Online]. Available: https://vehelec.com/products/obd-ii-ecu-simulator-iso15765-iso9141-2-iso14230-can-bus

[18] A. Abraham and MobSF collaborators. (Mar. 2024). *MobSF/Mobile-Security-Framework-MobSF*. [Online]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF

[19] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software* (Addison-Wesley Professional Computing Series). Reading, MA, USA: Addison-Wesley, 1995.

[20] Rapid7, Inc. and Metasploit Framework contributors. (2024). *Metasploit Framework*. [Online]. Available: https://github.com/rapid7/metasploit-framework

[21] Bishop Fox and Sliver contributors. (Jun. 2024). *Sliver*. [Online]. Available: https://github.com/BishopFox/sliver

[22] *Road Vehicles—Diagnostic Communication Over Controller Area Network (Do-CAN)*, Standard ISO 15765-2:2024, ISO Central Secretary, Geneva, Switzerland, 2024. [Online]. Available: https://www.iso.org/standard/84211.html

[23] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, 2021, "Car hacking: Attack & defense challenge 2020 dataset," doi: 10.21227/qvr7-n418.

[24] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, "Car hacking and defense competition on in-vehicle network," in *Proc. 3rd Int. Workshop Automot. Auto. Vehicle Secur.*, 2021, pp. 1–6.

[25] BlueDriverSupport. (Nov. 2018). *Friendly PSA: BlueDriver OBD2 Scanner Company (Lemur Vehicle Monitors) Seemingly Pulled From Market*. [Online]. Available: http://www.reddit.com/r/MechanicAdvice/comments/9t6e7m/friendlypsabluedriverobd2scannercompany/e8ujm1h/

[26] H. Wen, Q. A. Chen, and Z. Lin, "Plug-N-pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new over-the-air attack surface in automotive IoT," in *Proc. 29th USENIX Secur. Symp.*, Jan. 2020, pp. 949–965. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/wen

[27] (2018). *AN07—Sending Arbitrary CAN Messages*. [Online]. Available: https://www.elmelectronics.com/AppNotes/AppNote07.pdf

[28] M. Honorof. (Apr. 2016). *This Bluetooth Car Dongle Might Just Kill You*. [Online]. Available: https://www.tomsguide.com/us/lemur-bluedriver-security-flaw
[29] P. Roberts. (Apr. 2016). *CERT: Aftermarket Add-On Opens Cars to Life Threatening Hacks*. [Online]. Available: https://securityledger.com/2016/04/cert-warns-on-hacking-risk-to-bluedriver-plug-in/
[30] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang, "IoTFuzzer: Discovering memory corruptions in IoT through app-based fuzzing," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2018, pp. 1–15, doi: 10.14722/ndss.2018.23159.
[31] X. Feng, R. Sun, X. Zhu, M. Xue, S. Wen, D. Liu, S. Nepal, and Y. Xiang, "Snipuzz: Black-box fuzzing of IoT firmware via message snippet inference," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 337–350, doi: 10.1145/3460120.3484543.
[32] (2010). *ELM327—OBD to RS232 Interpreter*. [Online]. Available: https://www.elmelectronics.com/DSheets/ELM327DSH.pdf
[33] O. Alvear, C. T. Calafate, J.-C. Cano, and P. Manzoni, "Validation of a vehicle emulation platform supporting OBD-II communications," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2015, pp. 880–885, doi: 10.1109/CCNC.2015.7158092.
[34] M. R. Kabir and S. Ray, "ViSE: Digital twin exploration for automotive functional safety and cybersecurity," *J. Hardw. Syst. Secur.*, vol. 8, no. 2, pp. 133–144, May 2024, doi: 10.1007/s41635-024-00150-w.
[35] T. Huang, J. Zhou, and A. Bytes, "ATG: An attack traffic generation tool for security testing of in-vehicle CAN bus," in *Proc. 13th Int. Conf. Availability, Rel. Secur.*, Hamburg Germany, Aug. 2018, pp. 1–6, doi: 10.1145/3230833.3230843.
[36] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Lead. Issues Inf. Warfare Secur. Res.*, vol. 1, no. 1, p. 80, 2011.
[37] B. Thorne. (2024). *Python-Can*. [Online]. Available: https://github.com/hardbyte/python-can
[38] K. Lade. (Jun. 2021). *CANalyse*. [Online]. Available: https://github.com/KartheekLade/CANalyse
[39] (2024). *Hardware Bridge Session Connector*. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/client/hwbridge/connect/
[40] C. Smith. (2017). *Latest Metasploit Hardware Bridge Techniques*. [Online]. Available: https://www.youtube.com/watch?v=eqIIi7AZfOI
[41] (Nov. 2021). *IoTSF IoT Security Assurance Framework Release 3.0*. [Online]. Available: https://iotsecurityfoundation.org/wp-content/uploads/2021/11/IoTSF-IoT-Security-Assurance-Framework-Release-3.0-Nov-2021-1.pdf
[42] Tarlogic. (2024). *BSAM: Bluetooth Security Assessment Methodology*. [Online]. Available: https://www.tarlogic.com/bsam/

ROBERTO GESTEIRA-MIÑARRO received the B.S. degree in telecommunications engineering and the M.S. degree in telecommunications engineering and cybersecurity from the ICAI School of Engineering, Comillas Pontifical University, Madrid, Spain, in 2020 and 2022, respectively. He is currently pursuing the Ph.D. degree in vehicle cybersecurity with the IIT. He is also studying a B.S. degree in mathematics from the Universidad Nacional de Educación a Distancia (UNED), Madrid. He developed his master thesis at the Institute for Research in Technology (IIT) regarding vehicle cybersecurity.

Mr. Gesteira-Miñarro received the Extraordinary Award from ICAI in 2020 and the Best Final Degree Project Award from COITT/AEGITT in 2021. He achieved two Honorific Mentions, two Bronze Medals, and one Silver Medal at the International Mathematics Competition (IMC). He also enjoys learning hacking and exploitation techniques in CTF platforms.

IGNACIO GUTIÉRREZ received the B.S. degree in software engineering from the Universidad Francisco de Vitoria, in 2021, and the M.S. degree in cybersecurity from the ICAI School of Engineering, Comillas Pontifical University, in 2022.

He is currently a Research Student at the Institute for Research in Technology. His current research interest includes cybersecurity on various embedded platforms, such as vehicles. He also enjoys reverse engineering and CTF competitions. He has presented in conferences, such as JNIC 2024.

RAFAEL PALACIOS received the B.S., M.S., and Ph.D. degrees from the ICAI School of Engineering, Comillas Pontifical University, Madrid, Spain. He joined the ICAI School of Engineering as an Assistant Professor and the Institute for Research in Technology as a Researcher, in 1998, obtained Tenure, in 2004, and became a Full Professor, in 2020. He was the Head of the Programs in Telecommunications Engineering and Computer Science, from 2012 to 2024, and the Coordinator of the master's in cybersecurity, from 2019 to 2021. He is currently a Visiting Professor with Massachusetts Institute of Technology (MIT), Cambridge, MA, USA.

GREGORIO LÓPEZ received the Ph.D. degree in telecommunications engineering from the Universidad Carlos III de Madrid (UC3M), in 2014.

He is currently an Associate Professor with the ICAI School of Engineering, Comillas Pontifical University, where he is also the Coordinator of the M.S. in cybersecurity and a Senior Researcher with the Institute for Research in Technology. He has gained wide experience in close-to-market research gained through his participation in more than ten national and European research projects. As a result of his research activity, he holds a European patent and has published more than 50 papers in top-tier conferences and journals, receiving more than 1500 citations. His current research interests include cybersecurity in the IoT/OT, AI for cybersecurity and cybersecurity in AI, human and technological factors in cybersecurity, and children and adolescents online, having been the Coordinator of European H2020 Project RAYUELA (empoweRing and educAting YoUng pEople for the Internet by pLAying), which addresses this latter topic.