Companion Apps or Backdoors? On the Security of Automotive Companion Apps

Prashanthi Mallojula[1], Fengjun Li[1], Xiaojiang Du[2], and Bo Luo[1]

[1] University of Kansas, Lawrence, KS, USA
[2] Stevens Institute of Technology, Hoboken, NJ, USA
{prashanthi.mallojula, fli, bluo}@ku.edu, xdu16@stevens.edu

Abstract. Automotive companion apps are mobile apps designed to remotely connect with cars to provide features such as diagnostics, logging, navigation, and safety alerts. Specifically, onboard diagnostics (OBD) based mobile applications directly communicate with the in-vehicle network through the OBD device. This can lead to several security issues, for instance, onboard information of vehicles can be tracked or altered through a malicious or vulnerable app. We conduct a comprehensive measurement study including static, runtime, and network traffic analysis of OBD companion apps. Our analysis has been applied to 125 Android mobile applications available on the Google Play Store. We identify a set of vulnerabilities and further validate these vulnerabilities with real-world vehicles. We show that 70% of the apps have vulnerabilities that can lead to private information leakage, property theft, and direct risk while driving. For instance, 18 apps could connect to open OBD dongles without requiring any authentication, accept arbitrary CAN commands as inputs from the (potentially malicious) user, and deliver the commands to the CAN bus without any validation. We discuss the possible countermeasures and also make responsible disclosures to app developers.

Keywords: Automotive companion apps · Security · Privacy.

1 Introduction

On-board diagnostics (OBD) is a standardized system in vehicles that enables self-diagnosis and reporting of crucial engine and system data. The legislation mandates the use of OBD-II for the collection of emission information in all gasoline vehicles that are manufactured after 1996 [42]. In addition, all the major vehicle manufacturers utilized OBD-II to assist in vehicle diagnostics, since it is capable of accessing internal vehicular information such as sensor values, error codes, ECU self-diagnostics, etc. In the past decade, the cost of OBD-II access devices has dropped dramatically. Besides the manufacturers and repair/maintenance service providers, OBD-II was further utilized by the industry for other advanced functions. For example, insurance companies attach small monitoring devices through OBD-II to collect users' driving habits, so that they can provide discounts for safe driving behaviors. The consumer market has also adopted

this trend, leading to the development and adoption of various low-cost OBD-II access dongles and companion mobile apps to provide real-time monitoring and diagnostic functionalities to end users. While these devices and apps serve practical purposes, they raise security concerns.

A limited number of research efforts have been dedicated to the security analysis of OBD-II-based consumer devices. Wen *et. al.* was the first to investigate the security issues with wireless OBD-II dongles [46]. With an extensive investigation of all the OBD-II dongles on the retail market, the paper revealed a wide range of vulnerabilities concerning connection and communication with vehicles. [46] also introduced a custom tool, DongleScope, to systematically detect security vulnerabilities in the dongles. Meanwhile, the OBD-II dongles connect wirelessly with smartphones, in which the companion apps are installed to provide interfaces for users to interact with the vehicle through the dongles. The apps that lack proper security controls will become additional attack surfaces. However, an in-depth security investigation of OBD-II companion apps is still missing in the literature. A previous effort adopted off-the-shelf tools to detect general Android source code issues such as external storage, warnings, and file access, related to auto infotainment and OBD-II apps [36]. Their methodology or findings were not specific to OBD-II companion apps or vehicles.

We argue that a systematic and holistic study of the security of OBD-II companion apps is still missing in the literature, and, in this paper, we make the first attempt to address the gap by investigating the attack surfaces that are not exclusively studied in the literature. We focus on the security and privacy issues in various aspects of OBD-II companion apps. In a broader context, we reveal vulnerabilities within the app that could be exploited by adversaries to (1) gain unauthorized access to the dongles and to the vehicles, (2) compromise the normal operation of the vehicles; (3) steal sensitive personal or operational data from the vehicles, and (4) cause driver distraction and risk while driving.

To achieve this, we first identify all vehicle companion applications that work with OBD-II dongles from the Android Play Store. We start with collecting information from each application to assess their compatibility with various vehicle models and OBD-II dongles. Next, We implement static code analysis and network traffic analysis to gain insights into the authentication and secure communication protocols implemented in these apps. Simultaneously, we reverse engineer the CAN bus communication with the vehicle to extract vehicle control commands. Finally, we draw conclusions regarding vulnerabilities for each app through dynamic testing of app execution, utilizing the security information extracted. In this paper, we have identified a series of security concerns that are directly related to potential security threats. For instance, apps without authentication and/or user-vehicle binding can be vulnerable to unauthorized access to the OBD-II dongle and further to the connected vehicle. Moreover, companion apps with vehicle control features, such as the ability to lock/unlock the vehicle, and terminals that allow manual CAN bus commands, could be exploited to compromise the vehicle and inject high-risk commands when accessed by anyone within the covered wireless range. In addition, the companion apps often store

sensitive user and vehicle-related data and provide access to vehicle diagnostic logs. With inappropriate data protection protocols, this can result in the leakage of sensitive information. Finally, apps with no access control over features allowed while driving can lead to driver distraction and risk while driving.

The main contributions of the paper are summarized as follows:

• We are the first to present a comprehensive study on the security and privacy issues with vehicle/OBD-II companion apps.

• We uncover distinct vulnerabilities in companion apps that are rooted in inappropriate security practices. These vulnerabilities may lead to vehicle diagnostic leaks, eavesdropping, and, more critically, the disruption or compromise of a vehicle by attackers.

• To demonstrate the practical risk associated with these vulnerabilities, we tested them on different real-world vehicles. Our extensive analysis of 125 apps gathered from the Google Play Store reveals that 70% are vulnerable to unauthorized access, potentially leading to the leakage of sensitive vehicle information, while 40% of the apps can leak driving information logs, trip locations, etc from vehicles. Worst of all, 18 apps could be easily exploited as practical hacking tools to compromise vehicle control.

Ethical Considerations. Our goal is to expose vulnerable security practices in OBD II companion apps and demonstrate the potential impact of these vulnerabilities. All the app analyses were conducted in a lab and did not target or interfere with any vehicle. The final experiments on real vehicles were conducted in a safe environment (empty parking lot, stationary vehicle, did not attempt to move the vehicle). We exclusively tested on vehicles that we have authorized access to and ensured that no harm or interference occurred. Given that this is a mobile app-based analysis that can be applied to any vehicle model, we adhere to the authority of the vehicle owner. We further made responsible disclosures to app developers on all the identified vulnerabilities.

The rest of the paper is organized as follows: We introduce the system and threat models in Section 2. We present our methodology in Section 3, followed by the vulnerability identification results in Section 4. We discuss the potential controls in Section 5, followed by a brief literature review in Section 6. Finally, we conclude the paper in Section 7.

2 The System Model and the Threat Model

2.1 The System Model

The system model of our study is shown in Figure 1.

(a) On-Board Diagnostics II Port. OBD-II is an industrial standard data link connector for vehicles. While OBD-II was initially used by professionals for diagnosis and service, it has been recently adopted in the consumer market to provide real-time vehicle operation information to end users. The OBD port is standardized as a 16-pin connector. Each of these pins is dedicated to a specific purpose, e.g., power supply, CAN data communication (pins 6 and 14), and

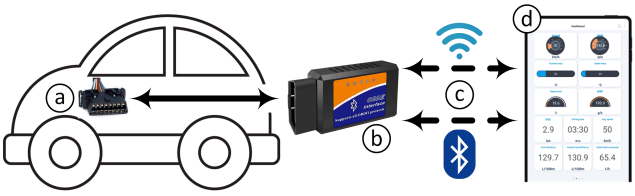

Fig. 1: The system model: (a) OBD-II port; (b) wireless OBD-II Dongle; (c) WiFi or Bluetooth communication; (d) companion app.

ground connections. The majority of modern vehicles support a range of communication protocols, ISO 9141-2, ISO 14230-4 (referred to as Keyword Protocol), ISO 15765-4 (CAN), and SAE J1850 [9].

(b) Wireless ODB-II Dongle. A wireless OBD-II dongle connects to the OBD-II port of a consumer vehicle and interacts with the ECUs through the Control Area Network (CAN) bus. OBD dongles are responsible for converting OBD commands into CAN standards. Most of the OBD-II dongles are capable of two-way communications with the CAN bus. The majority of OBD dongles are ELM327-based [5] and support a standardized "AT" command format. The security issues of the OBD-II dongles have been extensively studied in [46].

(c) Wireless Communication. The wireless OBD-II dongles are capable of communicating with external devices through WiFi (IEEE 802.11) or Bluetooth. They connect with smartphones with automotive companion apps so that they serve as bridges between the CAN bus and the apps. A Bluetooth dongle could communicate with only one smartphone at any time, while a WiFi dongle may communicate with multiple devices simultaneously.

(d) Automotive Companion Apps. As the primary objective of our paper, numerous OBD-II or automotive companion apps are available in app markets, such as the Apple App Store and Google Play. Each companion app is compatible with one or more OBD dongles. They support a wide range of operations and commands to access and control the connected vehicles. For instance, they are designed to provide diagnostic trouble codes, real-time vehicle information (speed, engine RPM), and parking assistance. These apps may communicate with the vehicle's ECU when the vehicle is parked, powered off, or even driving at high speed. Some apps also support vehicle control operations such as remote lock/unlock, control AC, and more. Their functions are valid for legitimate purposes. However, if not implemented with appropriate access control, these apps become attack surfaces that place vehicles and drivers at risk.

2.2 The Threat Model

The main attack surface to be investigated is the companion app and its periphery. An attacker can be any malicious external individual. In particular, we consider two types of attackers: (1) attackers who misuse the companion apps:

they are akin to the script kiddies, who are not skilled enough to perform vulnerability scanning or hacking into the companion apps. Instead, they utilize the apps as hacking tools, i.e., they use the published functions of the apps to perform malicious actions against the target vehicles. And (2) attackers who target the companion apps: they are the skilled attackers who are capable to eavesdrop on the communication or examine the app code to identify vulnerabilities. In both cases, the attacker's primary objectives are: (1) To gain access to the vehicle through a connected OBD II dongle; (2) to eavesdrop on app-to-vehicle communication; (3) to steal information of the vehicle or the user; and (4) to interfere with vehicle operations and potentially take control of the vehicle.

We assume that a Bluetooth or WiFi dongle is plugged into the OBD-II port of the victim's vehicle. This is likely to happen since such OBD-II dongles have entered the consumer market at very low prices ($15 to $50 each). The dongles and the companion apps have been advertised as on-board monitors to provide real-time vehicle information such as speed and torque. They also track the users' driving habits and provide personalized safe driving tips. Such functionalities and advertisements imply that the dongles are supposed to stay plugged in during the daily operations of the vehicles. Meanwhile, the attacker must be physically within the range of the vehicle's WiFi/Bluetooth signals to initiate an attack. The Bluetooth dongles advertise connection ranges of 20 to 50 meters outdoors, while WiFi dongles cover larger ranges. It is very practical for an attacker to get into such distance with a parked or even moving vehicle without being noticed. It is also possible for an attacker to wander around and scan for Bluetooth/WiFi signals from available dongles. This attacker model is the same as other Bluetooth-based attacks such as [46,15,12,14].

The attack begins with the attacker's detection of the vehicle based on the broadcast network information from the plugged OBD-II dongle. Note that the dongles connected to OBD ports remain powered even when the vehicles are off. While the dongles may have limited access to the CAN bus when the vehicles are off, they are accessible to the companion app (to be articulated in Section 3.4). Any compatible companion app can be employed at this step. Even when multiple vehicles with connected dongles coexist, an attacker may simultaneously connect with each of them. The attack could be targeted (against an identified vehicle) or untargeted (against a random vehicle). The targeted attack is made possible by certain companion apps that can detect vehicle details. Furthermore, even if an app requires specific knowledge of vehicle make and model to access its diagnostics, these details can be easily inferred by visually observing the vehicle. Overall, with the simple installation of a companion app, anyone within the WiFi/Bluetooth range can potentially launch an attack on the victim's vehicle.

3 Vehicle Companion App Analysis

In this section, we provide a clear and concise explanation of our analysis structure that includes application information, source code, and network traffic analysis. We then detail our approach to vulnerability definition. Overall, our analysis

of auto companion apps focuses on security concerns specific to vehicular applications. While some existing techniques have been adopted as the basic tools, e.g., network packet capture or static code analysis, all the analysis techniques are customized for vehicle companion apps.

3.1 App Function Analysis

To gain a better understanding of companion app execution and usage, our initial step involves gathering app information and establishing the initial setup procedures for each app. First, we collect information from Play Store API such as app descriptions, manufacturer details, the number of downloads, etc. Based on the collected information, we have summarized a series of features such as compatibility with vehicle models, supported OBD devices, and the available modes of connection for each app. We then installed each app on a smartphone and performed initial setup procedures. We tested the apps to gain insights into the connection and access of each app, with a special focus on available user interface (UI) options. We also collected the privacy policy of each app to understand if these security practices have user consent.

The app features examined in our security analysis are listed as follows: connection mode (WiFi, Bluetooth, Both), authentication (open access, password), user account (required, optional, no), OBD device (ELM327, manufacturer-specific), vehicle compatibility (any, specific make/model), vehicle detection (yes, No), access when vehicle off (yes, no), report log & share (yes, no), CAN commands terminal (yes, no), data transmission (encrypted, non-encrypted).

Each feature in this list contributes to one or more vulnerabilities if it is not properly implemented with specific security measures. We classify OBD companion apps into two categories: generic apps that connect to any OBD-II Dongle that is ELM327-Compatible [5], and proprietary apps that are only compatible with manufacturer-specific dongles. A few apps claim to support both ELM327-based and manufacturer-specific dongles, and we still classify them as generic apps. We note the OBD compatibility of the app as ELM327-based or manufacturer-specific. Similarly, vehicle compatibility denotes whether an app supports all vehicle models or a specific set of vehicle models. During the initial setup, we record if the app requires vehicle information before launch, and whether it initiates the diagnostic page when the vehicle is off. Additionally, we assess if the app has the capability to share stored diagnostic reports or trip logs. Some apps even provide an OBD terminal as part of the user interface, allowing users to manually send (arbitrary) CAN bus diagnostic and crucial vehicle control commands. In these cases, unauthorized access could result in full control over the vehicle, leading to potentially severe consequences. Finally, we record if the app has encrypted data transmission based on network packet analysis.

3.2 Static Source Code Analysis

We perform source code analysis to verify and evaluate security protocols related to external communication. Our main objectives include identifying and address-

ing any insecure network communication practices. In particular, we focused on two main issues in the source code: (1) Insecure external communication, and (2) Unvalidated CAN command input.

- *Insecure External Communication :* Companion apps often require communication with external entities for various features, including user account verification (e.g., Google, Facebook, etc.), diagnostic analytics, and the storage of reports in external storage. This practice involves the transfer of sensitive information, such as user data, vehicle diagnostics, and financial information. Applications that employ insecure network transfer are at risk of leaking sensitive data. For instance, we have observed instances in several companion apps, where endpoint URLs are hard-coded or with insecure calls.

We decompile the source code of each app using the JADX decompilation tool [41]. After extracting the Java source code, we identify URL patterns within each app and filter out insecure links from the extracted URLs. We have developed a customized Python script responsible for extracting methods for each URL.

- *Unvalidated CAN command input:* Certain apps allow users to enter raw CAN commands and send them to the CAN bus (through the connected OBD-II dongle) for execution. For such apps, we examine the source code to identify any validation or post-processing of user input, including CAN command format validation, any potential access control, and any sanitization of malicious or unsafe commands.

3.3 Network Traffic Analysis

We subject each app to dynamic execution to observe network traffic. Our goal is to uncover potential vulnerabilities related to insecure data transmission. During this process, we execute each user interface (UI) within the app that corresponds to vehicle diagnostics and sharing of vehicle diagnostic/trip logs. Concurrently capture network packets. If any insecure data transmission is detected, we verify if vehicular data is leaked such as diagnostic commands or user information, etc.

We have noticed that some applications transmit sensitive information without encryption. When vehicle diagnostic commands are transferred without encryption, it leaves an opportunity for attackers to extract this data for malicious purposes. This vulnerability was further confirmed through network traffic analysis, which revealed instances of vehicle information leaks.

Apps require user consent for any information transfer and are informed on security practices relating to app communication. Hence, we also verify the consistency between the app's security and corresponding privacy policy. Any mismatch will be considered a privacy violation.

3.4 Vulnerability Identification

With extensive analysis of companion app behavior, code base, and execution, we have identified a range of security vulnerabilities. These vulnerability scenarios include every aspect of the companion app's interaction with the vehicle.

[V1] Unauthorized app connection: Companion apps that allow connection without secured authentication can be vulnerable as any unauthorized individual can connect to a vehicle that has a dongle plugged in. Such access can grant attackers continuous visibility and access to vehicle information and further can leak vehicle-sensitive information. It can be argued that in the presence of an open connection on the dongle, limitations may arise concerning the extent to which apps can safeguard user accounts. However, in practice, even within an open connection device, the responsibility falls upon the application to employ authentication and authorization. Hence, we argue that apps without strict security measures to authenticate can lead to potentially vulnerable connections.

[V2] Inadequate authentication: One of the common implementations in these apps is user account setup and vehicle identification based on make and model. We argue that this is considered an inadequate authentication, as any attacker can set up an account and enter vehicle details by observing the vehicle. We also point out that VIN is considered inadequate authentication, as it is visible through the windshield.

[V3] Vehicle information and diagnostic leaks: Real-time vehicle information and diagnostic data contain sensitive information regarding vehicle status, sensor readings, and more. Improper access control may expose real-time vehicle data to attackers. Most of the apps feature diagnostic logs and trip tracking, i.e. the user can access full diagnostic reports and driving histories. These reports can be stored and shared with any third party.

[V4] Unvalidated vehicle control: Certain applications provide direct OBD terminal access allowing users to manually inject *arbitrary* CAN bus commands or a user interface to transmit customized commands to the vehicle. In either scenario, the user inputs (as text strings) are never validated or sanitized in any capacity, so that an attacker can use the app as a hacking tool to remotely control the vehicle. Notably, such attacks do not require advanced equipment; a basic understanding of CAN bus commands is sufficient to compromise and manipulate the vehicle's functions, posing risks to both the vehicle and its occupants.

[V5] Live CAN traffic leak: Most of the companion apps are compatible with ELM327-based dongles and support ELM327, standard "AT" commands. There is a specific command "ATMA" (Monitor All), which allows a user with access to extract live CAN bus traffic. This can be reverse-engineered to extract CAN bus commands of particular vehicle models and further attack the vehicle.

[V6] Eavesdropping on Vehicular communication: In the communication between the companion app and the vehicle, we can identify two hops: one from the vehicle to the OBD dongle and the other from the OBD dongle to the companion app. If data transmission is not secured, the app becomes vulnerable and susceptible to eavesdropping on network communication.

[V7] Insecure External Communication: Apps may establish connections with external entities through network connections. To transmit sensitive information, it is important to ensure the confidentiality and integrity of these data transfers. Some of these apps use the insecure "HTTP" protocol when handling sensitive information, which raises concerns about data security.

Table 1: Apps with Top Downloads

App Name	Downloads	Rating
Car Scanner ELM OBD2	10,000,000+	4.7
InCarDoc - OBD2 ELM327 Scanner	5,000,000+	4.2
EOBD Facile: OBD 2 Car Scanner	1,000,000+	4.4
Torque Lite (OBD2 & Car)	10,000,000+	3.6
Carista OBD2	1,000,000+	4.2
Infocar - OBD2 ELM Diagnostic	1,000,000+	4.2
MotorData OBD ELM car scanner	5,000,000+	4.1
FIXD - Vehicle Health Monitor	1,000,000+	4.6
Carly OBD2 car scanner	1,000,000+	3.9
BlueDriver OBD2 Scan Tool	1,000,000+	4.6

[V8] Privacy Policy Violation: We verify the consistency between the defined privacy policy for each app and the associated information handling (collection and sharing) practice and report identified privacy violations.
[V9] Access when vehicle is off: Some apps can connect to the OBD dongle when the vehicle is off. The OBD port remains powered even when the vehicle is off, hence, the companion app can connect to the dongle with limited access. In this case, some apps would terminate or display a warning message and prevent the user from going further. However, some apps still allow access to the dongle's settings and tracking logs, even when the vehicle CAN communication is restricted in this case. Apps with direct terminal access may enable users to *modify* settings for OBD dongles when the vehicle is off, further complicating access control for the driver/owner.

4 Vulnerability Analysis and Results

4.1 App Data Collection

We have collected 125 Android-based OBD II companion apps from the Google Play Store. We have crawled all the available information from the Play Store API such as No. of downloads, privacy policies, and descriptions. 68% of them support ELM327-based OBD II dongles and the remaining apps are manufacturer-specific (only work with a specific dongle). Considering connection modes, 66.4% support WiFi, and 97% of apps have Bluetooth connectivity. These apps have collectively accumulated a substantial number of downloads, demonstrating their popularity and a consistent increase in daily downloads. The highest number of downloads is 10,000,000+ on app *Car Scanner*. Overall, we have tested 70.4% apps on real-time automobiles and the rest of them have vehicle and dongle compatibility limitations. Further apps, with top downloads, are given in Table 1 along with connection and vehicle compatibility. In this table, we report the vulnerability ratio for each type can observe that most of the top downloaded apps are highly vulnerable and can compromise vehicle security.

4.2 Experiment Setup

Our experiment requires the use of a smartphone to operate all the applications. In our testing process, we have assessed the functionality of all the apps using two distinct vehicle models: a 2012 Nissan Altima, and a 2016 Toyota Corolla. We have employed a set of OBD dongles that collectively are compatible with a majority of the applications, including two generic WiFi dongles: OBDII-WiFi-ELM327 and Forseal-WiFi, one generic Bluetooth dongle: OBDII-Bluetooth-ELM327, and one WiFi/Bluetooth dongle: Kobra-WiFi/Bluetooth. We also used three manufacturer-specific dongles: Infocar, Think-driver, and Fixed. Apps that are compatible with test vehicle models and available OBD dongles are evaluated with real vehicles. 50 apps cannot be tested with the vehicle due to OBD and vehicle compatibility issues, however, they are all validated with app function analysis and static code analysis.

4.3 CAN Control Messages and Vulnerability Evaluation

Following previous techniques [7,46] on CAN message extraction through manually triggering physical action on vehicles, we have reverse-engineered CAN bus messages that can alter vehicle function. With this, we can effectively test vulnerabilities corresponding to vehicle control features. We have obtained fundamental control commands to inject CAN bus such as Turning lights and headlights. In particular, the '`AT SH`' command serves the purpose of specifying the ECU identifier on the vehicle responsible for handling specific operations. After a successful configuration, when the command '`AT SH 60D`' is transmitted, the CAN bus identifies the target ECU governing the headlights. Following this command, a CAN message is sent, represented as `06 06 00 00 00 00`, which activates the headlights. Table 2 provides a list of derived control commands on our test automobiles. We did not test critical operations such as controlling vehicle speed while moving.

In each experiment, the app is first set up and executed in conjunction with a compatible OBD dongle to verify connection setup, access, and associated security measures implemented by the app. Each available UI of the app and any vehicle control operations are examined. Access control over diagnostic and vehicle control access is verified. Next, each app is executed, and in parallel, network packets are captured to verify if the app has encrypted information transfer. Vulnerability corresponding to this is assessed information such as vehicle diagnostics extracted from network packets. Next, each app's source code is analyzed for any potential information leaks. We have performed static source code analysis of each app and verified for potential traces of information leaks. As mentioned in 3, we find out that insecure communication apps and CAN bus commands are directly mentioned in apps.

4.4 Vulnerability Assessment

Our systematic experiments with OBD companion apps have detected security vulnerabilities within each app. We did not change any function/feature of any

Table 2: CAN bus control commands

Vehicle	CAN Control Message		Action
Nissan	60D	00 46 00 00.. 00	Turn on the left signal
	60D	06 06 00 00.. 00	Switch headlights On
Toyota	614	29 80 00 10.. 00	Turn on the left signal
	614	29 80 00 30.. 00	Switch headlights On

app. In this section, we explain the analysis results for each vulnerability. Table 5 explains vulnerabilities found in the top 50 apps and reveals the implications of these security gaps in each application. While the security of these applications depends on the authentication requirements of the connected dongles, our research emphasizes the critical role of security measures within companion apps. Notably, more than 70% of the apps exhibited vulnerabilities related to unauthorized access and diagnostic data leaks.

[V1] Unauthorized app connection: We have tested connections using both WiFi and Bluetooth protocols. 88 companion apps allow open connection access without any authentication. Among these, 64 apps support unencrypted and unauthenticated WiFi connection, while 86 of them support Bluetooth (62 apps support both). For WiFi dongles, the attacker could connect to the dongle even when the owner's smartphone/app is already connected, as multiple connections are allowed. For Bluetooth dongles, a pairing process is necessary, and only one device could be connected to the dongle at a time. Both features provide certain protection, however, the attacker could still remotely connect to the dongle without having any authentication (e.g., knowing a password printed on the dongle). The remaining 37 apps have predefined authentication processes to connect to the OBD devices and vehicles. For example, "Think Driver" and "Fixed" need an OBD device serial number before connection authentication.

[V2] Inadequate authentication. While connection establishment, we observe two main inadequate verification patterns: (1) User account setup, and (2) Vehicle information. Overall, 58 apps enable user account creation and subsequent verification via email or phone. Out of these, 22 apps support open connection i.e. allow unauthorized app connection. They also store the information of the connected vehicle as part of the user profile. However, we still view this as an insecure practice, as the attacker could easily create an account on her own smartphone and connect to the open dongle.

Before launching diagnostics, 53 apps that allow open connection enquire about vehicle information. They typically seek basic details such as the vehicle's make and model. For instance, apps like "OBD jScan" and "Alfa Romeo" only launch vehicle diagnostics after selecting the vehicle and providing corresponding details. We consider these verifications to be *inadequate* as they do not impose any restriction on actual OBD II connection with the app and, more importantly, do not validate that the user has (authorized) physical access to the vehicle. Table 3 shows % of apps for connection modes that are vulnerable.

Table 3: Connection Setup

Connection	App count	Open access	user account	Vehicle Info
WiFi	83	64	13	38
Bluetooth	122	86	21	52

Table 4: Vehicle information leakage and implications.

Information Leak	Implication	Information Leak	Implication
Fault code	Diagnostic	Sensor values	Diagnostic
Location	Privacy	Driving info	Privacy
Live Dashboard	Diagnostic	Driving Log	Privacy
Fuel calculation	Diagnostic	Emission Test	Diagnostic
ECU Code	Vehicle info	ECU Details	Vehicle info
Battery Info	Diagnostic	Graph and report	Privacy

[V3] Vehicle information and diagnostic leaks. All applications that grant unauthorized access are at risk of exposing onboard vehicle diagnostics, which may leak critical vehicle data, including fault codes, sensor readings, and ECU IDs. More severely, several apps also leak sensitive trip information such as location history and trip logs. Figure 2 (A) shows the "Incar Doc Car Scanner" app after an unauthenticated connection, where we can observe the details of the leaked data. Figure 2 (B) and (C) show the leak of real-time vehicle status and sensor readings from *Car Scanner*. Note that Figure 2 (C) is a very long screen that could be scrolled down to reveal the readings of all accessible sensors. Table 4 summarizes the most leaked attributes and the implication of each leak.

[V4] Unvalidated vehicle control. 18 out of 125 apps have direct "OBD terminal" features available without any access control or message level validation. With all 18 apps, we successfully launched the attack on our test automobiles to control the vehicle's headlights and turn lights from all of these companion apps. Almost half of the apps that allow unauthenticated and invalidated vehicle control are the most downloaded apps. These apps mention in their descriptions that the terminal can be used for advanced functions or manual commands to aware users. However, the seriousness of attacks is evident from performed experiments and shows the need to impose restrictions on accessing this feature, given the open connection provided by the app. All of these apps do have the potential to leak diagnostics, Manual CAN injection Traffic leak, etc.

[V5] Live CAN traffic leak. We observe that apps with terminal access to allow manual CAN bus commands support standard ELM327 commands. Hence, we tested the command "`ATMA`" to capture live CAN traffic, as shown in the screen of app *Car Scanner* (Figure 2 (D)). With live CAN traffic, one can reverse engineer and extract control commands of the vehicle, as shown in [1].

[V6] Eavesdropping on Vehicular communication. The next crucial component is network traffic between the app (smartphone) and the vehicle (OBD dongle). Our experiments reveal that, overall, 50 out of 125 apps have trans-

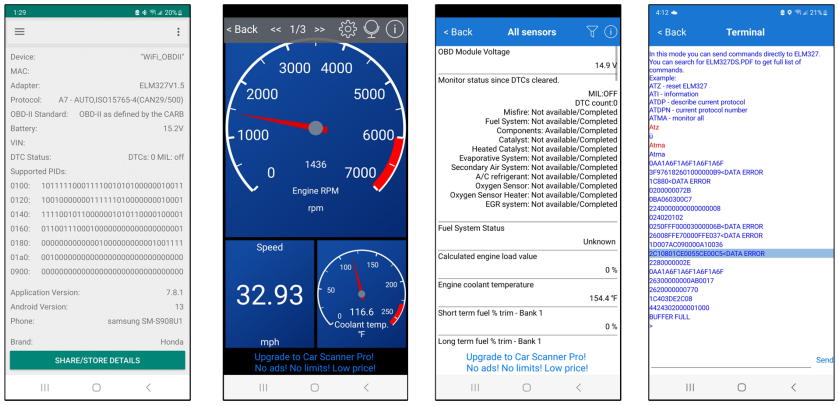

Fig. 2: (A) Vehicle information from InCarDoc; (B) Real-time vehicle status from CarSCanner; (C) Sensor readings from CarSCanner; (D) Real-time CAN traffic.

mitted information over unencrypted connections, posing eavesdropping vulnerability. With this, we have captured live CAN bus commands sent from the app to the dongle through network traffic. We were able to observe the vehicle diagnostic commands and vehicle status from the unencrypted communication.

[V7] Insecure External Communication. Our static analysis results reveal that 57 apps have utilized insecure URLs to send diagnostics reports and trip logs over unencrypted communication channels. We further validated each app through dynamic analysis and network traffic eavesdropping and found UIs responsible for third-party communication. We found that diagnostic reports and trip logs are stored mostly in device storage and shared with external entities. The trip logs contain the highly sensitive historical location information of the vehicle, which could be captured through network eavesdropping.

[V8] Privacy Policy Violation. We have mapped the consistency between the privacy policy and corresponding security practices. Apps with no user consent for information sharing are considered to violate privacy. We have verified the app API and 25 apps do not mention anything about information transfer but they indeed transmitted vehicle information and/or diagnostics reports.

[V9] Access when the vehicle is off. 47 out of 125 of the apps allow connection to the app when the vehicle is turned off. While CAN bus access is restricted in such cases, an attacker can still access previous diagnostics, if such information is continuously tracked. Moreover, apps with terminal access still allow access to connected dongles and configuration commands such as setting up CAN bus protocol. In this case, any malicious act can change the configuration related to a vehicle communication protocol, if the app has a specific UI that allows the change of settings related to vehicle access, these can be changed even when the vehicle is off. Apps without this vulnerability would terminate when the vehicle is turned off or display a warning message, which blocks all app functions.

4.5 Attack Summary

Our analysis results point critical importance of implementing robust security measures, especially in sensitive categories like vehicle communication. In this section, we evaluate attack objectives defined in section 2.2 using real-world vehicles, OBD-II dongles, and vulnerable companion apps. Each attack is executed by exploiting one or more vulnerabilities. Further, We explain the consequences associated with each attack in detail.

• *Gain access to the vehicle through a connected OBD II dongle.* The attacker's primary goal is to establish a connection and gain access to a vehicle. As shown in the experiments, this goal is easily achievable in the majority of the evaluated companion apps. Once access is obtained, the attacker may exploit the vehicle's systems, potentially leading to leaks of vehicle-related information, location data, and driving history. In cases of continuous connection, this poses a risk to the server and can further complicate the situation. This attack phase serves as a starting point to complicate further.

• *Eavesdrop on app-to-vehicle communication.* As we have validated, an external attacker could eavesdrop on app-to-vehicle communication due to a lack of link layer encryption and lack of end-to-end encryption. We have shown that vehicle information and diagnostics can be extracted from network packets captured during app-to-vehicle communication.

• *Steal information of the vehicle or the user.* As we have demonstrated, due to lack of authentication, encryption, and access control, vehicle information could be obtained through different attack surfaces, e.g., by directly connecting to a plugged dongle, or by eavesdropping on the app-to-vehicle communication of a legitimate user. This has the potential to leak location information, vehicle diagnostics, real-time vehicle operations, and even user information.

• *Interfere with vehicle operations and potentially take control of the vehicle.* As validated in our experiments, companion apps susceptible to unauthorized access can be leveraged to compromise vehicle control in two scenarios: 1. When an app features a user interface (UI) supporting vehicle control operations, such as door lock/unlock functions; 2. When an app includes an OBD II terminal for sending manual CAN bus commands. In both cases, the repercussions can be severe, as such attacks can potentially grant malicious actors the ability to compromise the vehicle's control and security. Also, this can cause Risky driving conditions as attackers can even connect to the vehicle while moving. Last, there is a possibility that an attacker can send incompatible commands to the vehicle's CAN bus, potentially causing further damage to the CAN bus.

5 Discussions

Each vulnerability identified in our study, concerning automotive companion applications, holds substantial implications for vehicle owners and their vehicles, ranging from property theft to vehicle information leaks and risks while driving, among others. Our analysis results point out the crucial importance of implementing robust security measures, especially in sensitive categories like vehicle

communication. Our comprehensive analysis executed each attack scenario and overall, results show that 70% of off-the-shelf OBD II companion apps are vulnerable. It is concerning to see that apps with poor security controls are allowed to be deployed in safety-critical domains.

Scope of our work. Our analysis includes all the app components that may serve as entry points for potential attacks. We have not provided an exhaustive examination of the source code, as this has been addressed in related research under the domain of mobile app source code analysis [34]. We focused on the unique elements that interact with OBD-II dongles and vehicles. Also, in the case of network communication, we did not decrypt the secured traffic or modify any protocol that any app uses.

Possible app security controls. We have observed that a few companion apps incorporate secure authentication methods by following measures such as user account verification, device identification, and restriction on full access to the app in the case of open-access devices. This reduces the likelihood of the app being malicious, even when the compatible dongle has open access. In summary, we suggest the following security controls for car companion apps: (1) Device authentication: a strong authentication measure should be employed when the companion app (the smartphone) attempts to connect to the OBD-II dongle, to validate that the user has physical ownership/access to the dongle/vehicle. For instance, the companion app may prompt the user for a serial number or a pin that is printed on the dongle. (2) User authentication: when the user profile is created and the vehicle information is associated with the profile, user authentication should be required when the user attempts to access the profile and the corresponding vehicle. (3) Encryption: all communications, including app-dongle communication and external communication, should properly utilize encryption. (4) Validate user input: while it could be a useful advanced function to allow users to directly enter CAN commands, such input should always be validated and sanitized. (5) Restricted functions for insecure dongles: for dongles that allow open access (unauthorized connection), the app should only provide very limited functionalities in terms of interacting with the dongle. In short, we emphasize the companion app's responsibility to guard against encouraging an attack and highlight specific security issues directly present in these apps.

Vehicle Hardware and CAN security. While the apps exhibit privacy and security concerns, it is equally important to address the security of OBD II hardware and the CAN protocol. Numerous studies proposed enhanced security measures for CAN bus communication [45] or OBD-II dongles [46]. However, these proposals have not yet seen widespread implementation in the market. CAN bus communication is a basic broadcast protocol that sends vehicle information without any encryption by design. External entities accessing the information are responsible for encrypting the information extracted from the CAN bus.

Responsible Disclosure We were not able to find developer details for most of the apps. For the 12 companion apps that provided developer contact information, we made a responsible disclosure in November 2023 regarding the

vulnerabilities found in our study. As of the submission of this paper, we have not received any response.

6 Related Work

A thrust of the existing work on vehicular security focuses on CAN bus security, e.g., [20,37,16,27,32,29,28]. The modern vehicles are integrated with network communication capacity enabling vehicles to communicate with external entities in real-time [23,51]. However, advanced integration such as companion apps needs to be thoroughly analyzed before they can be securely deployed. Here we briefly summarize the recent attacks against vehicles and discuss three major categories of defense: in-vehicle communication (CAN bus), vehicle communication with external entities, and mobile app security.

Recent Attacks on Vehicle. In the past decade, there has been a rise in vehicle attacks, particularly exploiting vulnerabilities in CAN bus and in-vehicle network systems. Attackers can compromise vehicles when they gain access to the vehicle's CAN bus [3,6,2]. A keyless car theft incident involving a Toyota RAV4 was reported as a CAN bus injection theft with CAN bus command [10]. Recent demonstrations of potential vehicle vulnerabilities highlighted security flaws in companion apps for electric vehicles like Nissan Leaf [4], and featured an illustrative showcase of an attack through Hyundai's Blue Link app [8]. These attacks highlight the importance of examining all the entities that can connect, access, and control the CAN bus, including the vehicle companion apps.

In-vehicle communication and CAN Bus Security. CAN bus communication has been studied in research, explaining possible vehicle attacks, in case of insecure CAN bus configuration and communication. CAN bus attacks such as "weeping attack" [18,30] can be executed through the OBD port. Several measures such as intrusion detection [31,38], cryptographic solutions [21], CAN firewall [27], and error handling have been proposed [39]. In this paper, we do not consider the security of the CAN bus itself, instead, we investigate how vehicle companion apps may be exploited to launch attacks against the CAN bus.

Auto Infotainment and OBD II Security. Several surveys explained security challenges in advanced vehicular communications [17,40,22]. [33] highlighted the importance of privacy-preserving vehicle information. "AT" commands sent to CAN bus are studied in [43]. Vehicle companion apps were analyzed to retrieve CAN commands [47,48]. [49,46,26] studies the attacks and defenses through OBD-II ports. In this paper, our scope is different from these works as we investigate how vehicle companion apps could be exploited as attacking tools.

Mobile App Security. Mobile apps have been widely studied for source code analysis, dynamic analysis, network traffic, etc [50,24,34]. Online tools such as MOBSF and Qark are available for static analysis from source code [11,35]. The backend cloud servers are also examined [52,13]. However, there have been limited efforts in security analysis for safety-critical domains. For instance, [25] works on Health IoT mobile apps to verify privacy information protection. Static analysis on the IoT apps revealed vulnerabilities such as poor access control [44].

Security analysis also revealed security concerns in industrial control systems (ICS) apps [19]. Our work further confirms that there has been minimal progress in enhancing security measures since the publication of [19].

7 Conclusion

In this paper, we conducted a comprehensive analysis of 125 OBD II companion apps, identifying 9 unique vulnerabilities through an examination of app functions, source code, and network traffic. We further evaluated each app to gain access to vehicle information, app information, user information, and vehicle control. Our experiments reveal that 70% of companion apps are insecure and impose vulnerabilities. Through the analysis and evaluation, we find that many existing vehicles and OBD-II companion apps continue the practice with minimal attention to security. Our work highlights the need to implement security protocols in safety-critical domains like vehicle communications.

Acknowledgments. This paper was supported in part by United States National Science Foundation (NSF) under grants IIS-2014552, DGE-1565570, DGE-1922649, CNS-2204785, and CNS-2205868, and the Ripple University Blockchain Research Initiative. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions

References

1. A complete guide to hacking your vehicle bus on the cheap and easy). https://theksmith.com/software/hack-vehicle-bus-cheap-easy-part-1/
2. A Remote Attack on the Bosch Drive log Connector Dongle. https://argus-sec.com/blog/cyber-security-blog/remote-attack-bosch-drivelog-connector-dongle/
3. Auto cyberattacks becoming more widespread, https://semiengineering.com/auto-cyberattacks-becoming-more-widespread
4. Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs. https://www.troyhunt.com/controlling-vehicle-features-of-nissan/
5. ELM 327 detailed info. https://www.sparkfun.com/datasheets/Widgets/ELM327_AT_Commands.pdf
6. Hacking cars remotely with just their vin, https://www.bitdefender.com/blog/hotforsecurity/hacking-cars-remotely-with-just-their-vin
7. How to hack a car — a quick crash-course, https://www.freecodecamp.org/news/hacking-cars-a-guide-tutorial-on-how-to-hack-a-car-5eafcfbbb7ec
8. Hyundai 'Blue Link' Vulnerability Allows Thieves To Start Cars Remotely (Update: Hyundai's Statement). https://www.tomshardware.com/news/hyundai-blue-link-vulnerability-thieves,34248.html
9. OBD2 Explained - A Simple Intro. https://www.csselectronics.com/pages/obd2-explained-simple-intro
10. There's a new form of keyless car theft that works in under 2 minutes (2023)
11. Abraham, A., et al.: Mobile Security Framework (MobSF). https://github.com/ajinabraham/Mobile-Security-Framework-MobSF, accessed January 2024

12. Ai, M., Xue, K., Luo, B., Chen, L., Yu, N., Sun, Q., Wu, F.: Blacktooth: breaking through the defense of bluetooth in silence. In: ACM CCS (2022)
13. Alrawi, O., Zuo, C., Duan, R., Kasturi, R.P., Lin, Z., Saltaformaggio, B.: The betrayal at cloud city: an empirical analysis of cloud-based mobile backends. In: USENIX Security Symposium). pp. 551–566 (2019)
14. Antonioli, D., Payer, M.: On the insecurity of vehicles against protocol-level bluetooth threats. In: IEEE Security and Privacy Workshops (2022)
15. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.B.: The {KNOB} is broken: Exploiting low entropy in the encryption key negotiation of bluetooth {BR/EDR}. In: USENIX security symposium (2019)
16. Avatefipour, O., Malik, H.: State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities. arXiv:1802.01725 (2018)
17. Bernardini, C., Asghar, M.R., Crispo, B.: Security and privacy in vehicular communications: Challenges and opportunities. Vehicular Communications **10** (2017)
18. Bloom, G.: Weepingcan: A stealthy can bus-off attack. In: Workshop on Automotive and Autonomous Vehicle Security (2021)
19. Bolshev, A., Yushkevich, I.: Scada and mobile security in the internet of things era. EMBEDI, IOActive, Whitepaper (2017)
20. Bozdal, M., Samie, M., Aslam, S., Jennions, I.: Evaluation of can bus security challenges. Sensors **20**(8), 2364 (2020)
21. Bruton, J.A.: Securing can bus communication: An analysis of cryptographic approaches. Nat. Univ. Ireland, Galway pp. 1–5 (2014)
22. De La Torre, G., Rad, P., Choo, K.K.R.: Driverless vehicle security: Challenges and future research opportunities. Future Generation Computer Systems **108**, 1092–1111 (2020)
23. Demba, A., Möller, D.P.: Vehicle-to-vehicle communication technology. In: IEEE international conference on electro/information technology (EIT) (2018)
24. Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M.S., Conti, M., Rajarajan, M.: Android security: a survey of issues, malware penetration, and defenses. IEEE communications surveys & tutorials **17**(2), 998–1022 (2014)
25. He, D., Naveed, M., Gunter, C.A., Nahrstedt, K.: Security concerns in android mhealth apps. In: AMIA annual symposium proceedings. vol. 2014, p. 645. American Medical Informatics Association (2014)
26. Humayed, A.: An overview of vehicle obd-ii port countermeasures. In: International Conference on Interactive Collaborative Robotics (2023)
27. Humayed, A., Li, F., Lin, J., Luo, B.: Cansentry: Securing can-based cyber-physical systems against denial and spoofing attacks. In: ESORICS (2020)
28. Humayed, A., Luo, B.: Poster: Cyber-physical security for smart cars: taxonomy of vulnerabilities, threats, and attacks. In: ACM/IEEE ICCPS (2015)
29. Humayed, A., Luo, B.: Using id-hopping to defend against targeted dos on can. In: International Workshop on Safe Control of Connected and Autonomous Vehicles (2017)
30. Iehira, K., Inoue, H., Ishida, K.: Spoofing attack using bus-off attacks against a specific ecu of the can bus. In: IEEE CCNC (2018)
31. Jedh, M., Othmane, L.B., Ahmed, N., Bhargava, B.: Detection of message injection attacks onto the can bus using similarities of successive messages-sequence graphs. IEEE Transactions on Information Forensics and Security **16**, 4133–4146 (2021)
32. Jo, H.J., Choi, W.: A survey of attacks on controller area networks and corresponding countermeasures. IEEE Transactions on Intelligent Transportation Systems **23**(7), 6123–6141 (2021)

33. Krishna, A.M., Tyagi, A.K., Prasad, S.: Preserving privacy in future vehicles of tomorrow. JCR **7**(19), 6675–6684 (2020)
34. Li, L., Bissyandé, T.F., Papadakis, M., Rasthofer, S., Bartel, A., Octeau, D., Klein, J., Traon, L.: Static analysis of android apps: A systematic literature review. Information and Software Technology **88**, 67–95 (2017)
35. LinkedIn: QARK. https://github.com/linkedin/qark, 2018
36. Mandal, A.K., Panarotto, F., Cortesi, A., Ferrara, P., Spoto, F.: Static analysis of android auto infotainment and on-board diagnostics ii apps. Software: Practice and Experience **49**(7), 1131–1161 (2019)
37. Nowdehi, N., Lautenbach, A., Olovsson, T.: In-vehicle can message authentication: An evaluation based on industrial criteria. In: IEEE VTC (2017)
38. Serag, K., Bhatia, R., Faqih, A., Ozmen, M.O., Kumar, V., Celik, Z.B., Xu, D.: {ZBCAN}: A {Zero-Byte}{CAN} defense system. In: USENIX Security (2023)
39. Serag, K., Bhatia, R., Kumar, V., Celik, Z.B., Xu, D.: Exposing new vulnerabilities of error handling mechanism in {CAN}. In: USENIX Security Symposium (2021)
40. Sharma, S., Kaushik, B.: A survey on internet of vehicles: Applications, security issues & solutions. Vehicular Communications **20**, 100182 (2019)
41. skylot: Jadx - Dex to Java decompiler, 2020.
42. skylot: On-board diagnostic ii (obd ii) systems fact sheet (2019), https://ww2.arb.ca.gov/resources/fact-sheets/board-diagnostic-ii-obd-ii-systems-fact-sheet
43. Tian, D.J., Hernandez, G., Choi, J.I., Frost, V., Raules, C., Traynor, P., Vijayakumar, H., Harrison, L., Rahmati, A., Grace, M., et al.: Attention spanned: Comprehensive vulnerability analysis of {AT} commands within the android ecosystem. In: USENIX Security (2018)
44. Tian, Y., Zhang, N., Lin, Y.H., Wang, X., Ur, B., Guo, X., Tague, P.: Smartauth: User-centered authorization for the internet of things. In: USENIX Security Symposium. vol. 5, pp. 8–2 (2017)
45. Van Herrewege, A., Singelee, D., Verbauwhede, I.: Canauth-a simple, backward compatible broadcast authentication protocol for can bus. In: ECRYPT workshop on Lightweight Cryptography. vol. 2011, p. 20. ECRYPT (2011)
46. Wen, H., Chen, Q.A., Lin, Z.: Plug-n-pwned: Comprehensive vulnerability analysis of obd-ii dongles as a new over-the-air attack surface in automotive iot. In: USENIX Security Symposium (2020)
47. Wen, H., Zhao, Q., Chen, Q.A., Lin, Z.: Automated cross-platform reverse engineering of can bus commands from mobile apps. In: NDSS (2020)
48. Yu, L., Liu, Y., Jing, P., Luo, X., Xue, L., Zhao, K., Zhou, Y., Wang, T., Gu, G., Nie, S., et al.: Towards automatically reverse engineering vehicle diagnostic protocols. In: USENIX Security Symposium (2022)
49. Zhang, Y., Ge, B., Li, X., Shi, B., Li, B.: Controlling a car through obd injection. In: IEEE International Conference on Cyber Security and Cloud Computing (2016)
50. Zhang, Y., Dai, J., Zhang, X., Huang, S., Yang, Z., Yang, M., Chen, H.: Detecting third-party libraries in android applications with high precision and recall. In: IEEE Intl. Conf. on Software Analysis, Evolution and Reengineering (2018)
51. Zhao, J., Chen, Y., Gong, Y.: Study of connectivity probability of vehicle-to-vehicle and vehicle-to-infrastructure communication systems. In: IEEE VTC (2016)
52. Zuo, C., Lin, Z., Zhang, Y.: Why does your data leak? uncovering the data leakage in cloud from mobile apps. In: IEEE Symposium on Security & Privacy (2019)

A Summary of Vulnerabilities in Top Apps

In Table 5, we summarize the vulnerabilities in the top most downloaded apps.

Table 5: Vehicle Companion Apps Vulnerability Summary. WF: WiFi, BT: Bluetooth

App Name and Version	**Conn.**	V1	V2	V3	V4	V5	V6	V7	V8	V9
CarScannerELMOBD2 v1.89	WF&BT	✓	✓	✓	✓	✓	✓			✓
Carista OBD2 v8.1	WF&BT	✓	✓	✓	✓					
Torque Lite (OBD2 & Car) v1.2	BT	✓	✓	✓						✓
inCarDoc - OBD2 ELM327 Scanner v7.6	WF&BT	✓	✓	✓	✓	✓	✓			✓
MotorData OBD ELM car scanner v1.25	WF&BT	✓	✓	✓						
BlueDriver OBD2 Scan Tool v7.13	BT							✓		
Carly — OBD2 car scanner v48.08	WF	✓	✓	✓						
DashCommand OBD ELM App v4.8	WF&BT	✓	✓	✓				✓		✓
EOBD Facile OBD2 Car Scanner v3.45	WF&BT	✓	✓	✓	✓	✓	✓			✓
FIXD OBD2 Scanner v7.33	BT									
Infocar - OBD2 ELM Diagnostic v2.24	WF&BT	✓	✓	✓						✓
Obd Arny - ELM327 car scanner v0.14	WF&BT	✓	✓	✓			✓			
Obd Harry Scan -ELM car scanner v0.99	WF&BT	✓	✓	✓			✓	✓		
Obd Mary – OBD2 car scanner v1.19	WF&BT	✓	✓	✓			✓	✓		✓
OBD2 Auto Scaner Olivia Drive v22.1	WF&BT	✓	✓	✓			✓	✓		✓
OBD Auto Doctor scanner v6.6	WF&BT	✓	✓	✓			✓			
OBDclick Car Scanner OBD2 ELM v0.9	WF&BT	✓	✓	✓			✓	✓		✓
OBDeleven VAG car diagnostics v0.60	BT	✓	✓	✓						
OBD Link v5.32	WF&BT									
Piston - OBD2 Car Scanner v3.1	WF&BT	✓	✓	✓			✓	✓		
Scan Master for ELM327 OBD2 v5.3	WF&BT	✓	✓	✓			✓			
Speedboat. GPS/OBD2 Speedometer v3.3	WF&BT	✓	✓	✓						
Torque Pro (OBD2 & Car) v1.8	BT	✓	✓	✓				✓		✓
Clear And Go - OBD2 Scanner v1.12	WF&BT	✓	✓	✓			✓			
Elm327 OBD Terminal v1.3	BT	✓	✓	✓	✓	✓	✓		✓	✓
GaragePro Car OBD2 Scanner v2.9	BT	✓	✓	✓						✓
GPS Speedometer OBD2 Dashboard v2.5	WF&BT	✓	✓	✓						
Hybrid Assistant v3.31	BT	✓	✓	✓						
OBD 2: Torque Car Scanner FixD v1.3	BT	✓	✓	✓						✓
RealDash v2.2	WF	✓	✓	✓			✓			
Repair Solutions2 v2.0	BT									
CarBit ELM327 OBD2 v3.5	WF&BT	✓	✓	✓	✓	✓	✓			✓
AlfaOBD Demo v2.3	WF&BT	✓	✓	✓				✓		
Autel MaxiAP200 v1.58	BT									
Auto Agent v1.29	BT									
Car Diagnostic Pro (OBD2) v7.17	WF&BT	✓	✓	✓			✓			
CarDiag: Car Diagnostic OBD2 v1.1	BT	✓	✓	✓				✓		✓
CarSys Scan (Best OBD2&ELM32) v1.8	BT	✓	✓	✓						
ChevroSys Scan Lite v1.1	WF&BT	✓	✓	✓						
DeepOBD v1.01	BT	✓	✓	✓				✓		
Dr. Prius / Dr. Hybrid v6.1	WF&BT	✓	✓	✓						
EconTool Nissan ELM327 v3.31	WF&BT	✓	✓	✓			✓		✓	
Elm327 WiFi Terminal OBD v1.1	WF&BT	✓	✓	✓	✓	✓	✓			✓
FordSys Scan Lite v1.1	WF&BT	✓	✓	✓						
GPS Speedometer: Car Dashboard v1.0	WF&BT	✓	✓	✓						
HobDrive OBD2 diag, trip v1.6	WF&BT	✓	✓	✓						
Hondash v2.8	BT	✓	✓	✓						
Ht200 v1.58	BT									
OBD Jscan v28.10	WF&BT	✓	✓	✓			✓		✓	
OBD2 Bluetooth Car Scanner v2.6	BT	✓	✓	✓						