Software Engineering Institute
Carnegie Mellon University

On Board Diagnostics: Risks and Vulnerabilities of the Connected Vehicle

Dan Klinedinst
Christopher King

March 2016

CERT® Coordination Center

http://www.sei.cmu.edu

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM-0003466

Table of Contents

Executive Summary **v**

1 Background of Automotive Cybersecurity **1**

2 Architecture of OBD-II and CAN **2**
2.1 OBD-II Architecture 2
2.2 CAN Bus Architecture 3

3 Threat Model **6**

4 Vulnerabilities in Aftermarket OBD-II Devices **8**

5 Common Architectural Issues **10**
5.1 The Primary Processor 10
5.2 The External Network Interface 10
5.3 Undocumented Features 10
5.4 Insecure Firmware Updates 10

6 Recommendations **11**

7 Conclusion **12**

Bibliography **13**

Appendix A: Testing Methodology **14**

Appendix B: Test Plan **15**

Appendix C: Test Results **17**

Appendix D: Impact Scoring **20**

List of Figures

Figure 1: OBD-II Connector (Image courtesy Wikipedia/Michiel1972.) 2

Figure 2: ECUs Connected to a Single CAN Bus 4

Figure 3: OBD-II Port Data Capture with Decoded PIDs 5

Figure 4: Generic OBD-II Device Threat Model Diagram 6

List of Tables

Table 1:	OBD-II Pinout Diagram (SAE Standard, 2002)	3
Table 2:	Vulnerability Impact on the Device and the Vehicle	6
Table 3:	Vulnerability Impact on Vehicle with Complete Device Compromise by Proximity	7
Table 4:	List of Devices Tested	17
Table 5:	Vulnerability Impacts from SAE (Ward, et al., 2013)	20

Executive Summary

The modern vehicle is often referred to as a "computer on wheels." With over 100M lines of code, the complexity is greater than your desktop operating system (85M lines of code) and slightly less complex than the genome of a mouse (120M base pairs of DNA). As these vehicles become more advanced, semi-autonomous, and connected, it opens the automobile to the same risks as any other computer—only with physical effects.

Unlike the traditional PC, vehicles are heavily regulated; laws promote consumer safety, increase gas efficiency, and provide freedom to tinker (Miller, 2013). The 1990 Clean Air Act requires all vehicles built after 1994 to include on-board computer systems to monitor vehicle emissions and to provide repair shops and car owners with the same tools that dealerships use to test and repair vehicles. One of the effects of this law is the mandate for the on-board diagnostic (OBD-II) port and a related subsequent standard. This port allows anyone connecting to it access to the vehicle to monitor the performance and function.

With the smartphone revolution, sensors and microprocessors have become increasingly cheaper and miniaturized. Enterprising startups have used these components to develop aftermarket OBD-II plug-in devices that contain (via wireless internet) cellular connectivity, and even GPS. These devices are used by insurance companies, consumers, and fleet managers to track and monitor their vehicles.

While these devices may provide consumers with useful features, they also expose the vehicle to potential risks that were never considered when it was originally designed. The OBD-II port was expected to be a physical-access port only, not one that is open to the Internet.

The Department of Homeland Security's US-CERT tasked the CERT Coordination Center (CERT/CC) at Carnegie Mellon University's Software Engineering Institute (SEI) to study these devices to better understand the cybersecurity impact to consumers and the public. The CERT/CC analyzed a representative sample of these devices for vulnerabilities and found widespread failure to apply basic security principles. If these devices are compromised, the potential impact includes loss of privacy, vehicle performance degradation or failure, and potential injury. The CERT/CC hopes this research will better inform consumers, enterprise fleet managers, insurance companies, and policy makers about the potential risks of these devices. The OBD-II port was created to provide consumers with choice and control over their purchase. At the same time, this freedom must be balanced with thoughtful conversations on how to limit adversaries' access to vehicle internals.

1 Background of Automotive Cybersecurity

The modern automobile consists of over 100M lines of code (Charette, 2009), operating dozens of electronic control units (ECUs) that manipulate everything from brakes, wipers, braking, and even steering. These ECUs interact in simple networks called controller area networks (CANs) that allow them to communicate at high speeds. With the advent of partial vehicle autonomy, automotive computers perform more and more driving functions for the driver. In addition, some manufacturers are adding Internet connectivity to vehicles to provide additional services to the consumer.

This increasing connectivity has led to more scrutiny of in-vehicle networks used by automotive manufacturers, and several high-profile vulnerabilities in recent years (Andy Greenberg, 2015) (Koscher, 2010). Unlike conventional information technology, the security of in-vehicle networks has safety ramifications in addition to traditional security risks (i.e., confidentiality and integrity).

One of the primary concerns with these networks is whether they can be accessed without physical access to the car. While vehicles have traditionally been isolated from other digital networks, modern cars often have multiple access points to the Internet or other digital devices. With the advent of the smartphone revolution and increased miniaturization, startups and existing vehicle aftermarket manufacturers have developed devices that attach to the on-board diagnostic (OBD-II) port that is present in all modern cars (OTAQ, 1996). This port has traditionally been used by mechanics to download diagnostic data and run tests, but there is a market emerging to allow car owners to access the same data via their mobile device or even over the Internet. These OBD-II ports provide raw access to the CAN bus, potentially allowing direct manipulation of CAN traffic in the vehicle. In prior research by Miller and Koscher, they show that connecting directly to the vehicle in this way could result in control over safety-critical functions because CAN, by design, offers no protection from manipulation (Miller, 2013), (Koscher, 2010). With the proliferation of these devices to support applications such as mile-based insurance, fleet management, and consumer vehicle diagnostics, the Department of Homeland Security's US-CERT tasked Carnegie Mellon's CERT Coordination Center to perform an initial security analysis of these devices to determine their common vulnerabilities, security controls, and risks.

2 Architecture of OBD-II and CAN

2.1 OBD-II Architecture

The OBD-II port (see Figure 1) is usually located in the foot well of the driver's side of a car and has been mandated on new cars in the United States since 1996 (OTAQ, 1996). The OBD-II standard itself defines both the electrical interface and messages, primarily Diagnostic Trouble Codes (SAE Standard, 2002). OBD-II was created to be used as standard way for vehicles of any manufacturer to interoperate with independent repair shops and dealers to test for emissions standards. Automotive manufacturers realized, however, that this port could be enhanced to provide a richer set of data to repair shops.

OBD-II is a 16-pin connector port, although only 9 pins are mandated by the standard itself (see Table 1). Most vehicles include additional connectivity, such as the General Motors LAN or the Chrysler CCD (Chrysler Collision Detection) Bus. These additional connections provide backend connectivity to other buses within the vehicle itself. They are used at the dealership, for example, to update software on the ECUs, perform additional testing, or troubleshoot problems specific to that vehicle make and model.

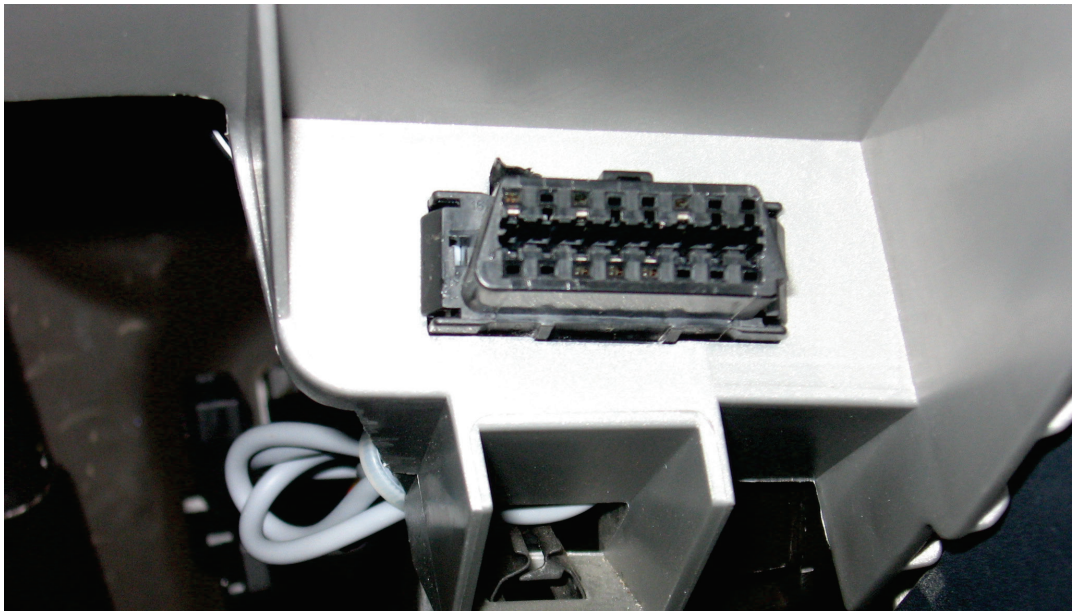

Figure 1: OBD-II Connector (Image courtesy Wikipedia/Michiel1972.)

Automotive manufacturers have kept the specifics of each of their respective standards under tight control, with high fees and subscriptions necessary to access information about a particular vehicle's underlying bus architecture. Despite this control, aftermarket OBD-II manufacturers have been able to reverse engineer the communication architectures to enhance their own offerings. At first, only standard OBD-II connectors were able to read diagnostic information specific to a particular vehicle. With the advent of inexpensive cellular, Wi-Fi, and Bluetooth components, aftermarket OBD-II

manufacturers have expanded their products to include entirely new classes of service, such as pay-by-mile insurance, vehicle-use tracking, and commercial fleet management.

Table 1: OBD-II Pinout Diagram (SAE Standard, 2002)

Pin	Signal	Description
2	J1850 Bus+	-----
4	CGND	GND
5	SGND	GND
6	CAN High	J-2284
7	ISO 9141-2 K-LINE	Tx/Rx
10	J1850 Bus-	-----
14	CAN Low	J-2284
15	ISO 9141-2 L-LINE	Tx/Rx
16	+12v	Battery power

2.2 CAN Bus Architecture

Inside the vehicle, the OBD-II port is connected to one or more communication buses. CAN is a bus protocol that is always supported, although there may be others (General Motors also uses GM-LAN for example). CAN itself is a high-speed, promiscuous protocol, broadcasting all network traffic to all nodes on a given bus. In modern vehicles, the CAN bus is connected to the OBD-II port and can often receive and relay all traffic to the external side of the interface.

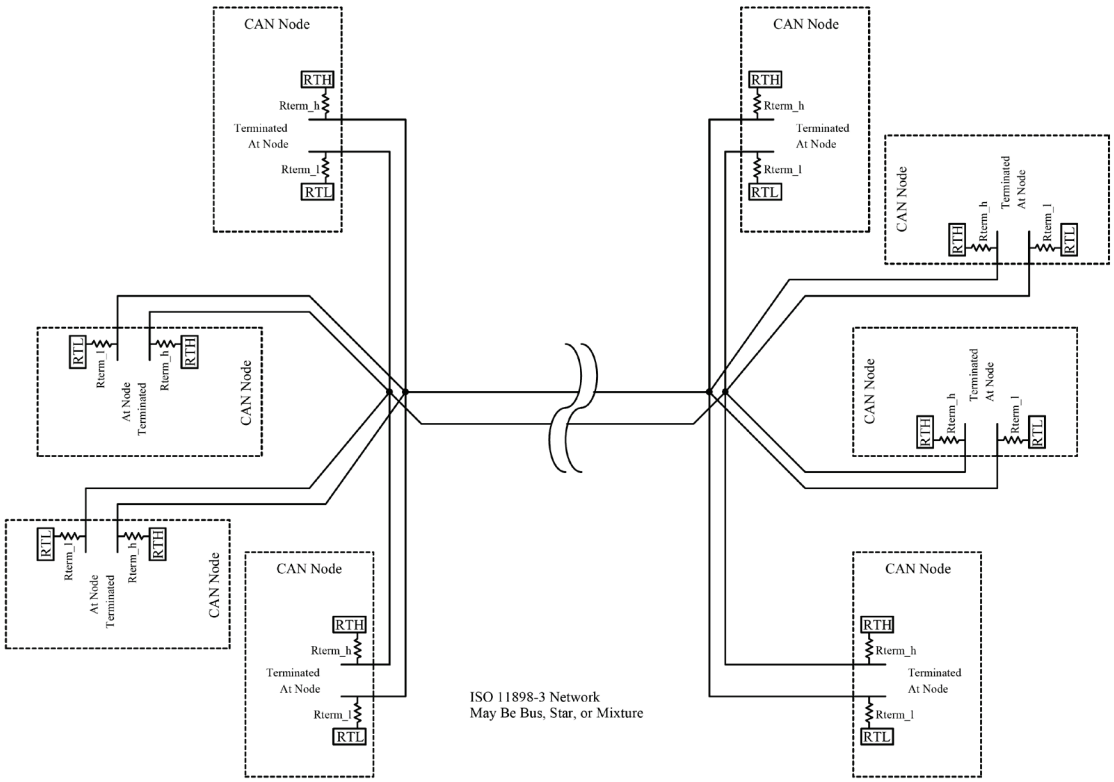

Figure 2: ECUs Connected to a Single CAN Bus
A device plugged into the OBD-II port acts like another ECU. (Image from the International Standards Organization [ISO, 2015].)

CAN is used in the vehicle for much more than transmitting Diagnostic Trouble Codes, however. Modern cars have hundreds of sensors and ECUs that control everything from vehicle operations (steering, braking, accelerating) to peripheral functions such as door locks or infotainment systems. All of the information these components exchange travels over the CAN bus or a similar (often proprietary) bus and is normally accessible via the OBD-II port. For example, a sensor measures the speed of the wheels and sends that data to the speedometer.

Unlike the TCP-based packet-switched networks present in most organizations' IT infrastructures, the CAN protocol sends all data to all nodes on a given bus. Each CAN frame includes a destination address; nodes are expected to ignore traffic not addressed to them. However, a CAN frame does not include a source address, so the destination node has no way to know if the source is legitimate. In our speedometer example, the speedometer has no way to differentiate between speed data sent from the wheels and speed data sent from another device, including the OBD-II port. How the device deals with conflicting data varies, but common responses are to use the last value received or use to the value that is received most frequently.

```
010D: Vehicle Speed: [7E8 03 41 0D 00 AA AA AA AA ]
0110: Air Flow Rate (MAF): [7E8 04 41 10 03 C7 AA AA AA ]
0144: Command equivalence ratio: [7E8 04 41 44 80 00 AA AA AA ]
0111: Throttle Position: [7E8 03 41 11 5C AA AA AA AA ]
Speed = 00 km/hr
010D: Vehicle Speed: [7E8 03 41 0D 00 AA AA AA AA ]
0110: Air Flow Rate (MAF): [7E8 04 41 10 03 F2 AA AA AA ]
0144: Command equivalence ratio: [7E8 04 41 44 80 00 AA AA AA ]
0111: Throttle Position: [7E8 03 41 11 5C AA AA AA AA ]
Speed = 01 km/hr
010D: Vehicle Speed: [7E8 03 41 0D 01 AA AA AA AA ]
0110: Air Flow Rate (MAF): [7E8 04 41 10 03 DB AA AA AA ]
0144: Command equivalence ratio: [7E8 04 41 44 80 00 AA AA AA ]
0111: Throttle Position: [7E8 03 41 11 5C AA AA AA AA ]
Speed = 02 km/hr
010D: Vehicle Speed: [7E8 03 41 0D 02 AA AA AA AA ]
0110: Air Flow Rate (MAF): [7E8 04 41 10 03 F5 AA AA AA ]
0144: Command equivalence ratio: [7E8 04 41 44 80 00 AA AA AA ]
0111: Throttle Position: [7E8 03 41 11 5C AA AA AA AA ]
Speed = 03 km/hr
```

Figure 3: OBD-II Port Data Capture with Decoded PIDs

The CAN protocol suffers from several security issues that are not addressed in most modern vehicles. We hypothesize two reasons for these issues. The first is that most of the networks and ECUs were designed when access to the bus required physical access to the vehicle. As such, security was not a primary concern. The second is that speed and timing are deemed more important to the safety and performance of the vehicle than data security. It would be undesirable, for example, to have to wait an extra millisecond to verify the sender of a byte of data when that millisecond is needed to engage an airbag.

A vehicle with security vulnerabilities introduces safety concerns because an intelligent adversary could actively exploit a vulnerability. Compare this concern to traditional safety considerations that operate under the principle that a system or node must be able to operate under certain tolerances and respond to stimuli in a particular way. Malicious hackers explicitly attempt to violate logical constructs and controls to manipulate the system to their own needs.

There are safety concerns about the current automotive bus architecture as a result of this security architecture. An OBD-II aftermarket device can potentially receive arbitrary CAN traffic from outside the vehicle via its wireless radio interface and pass it unfiltered to the internal CAN bus through the OBD-II port. The susceptibility to this type of attack was the focus of our initial research.

3 Threat Model

Threat modeling is a technique used to enumerate potential areas of risk in an application or system. Here we model the potential areas of vulnerability and particular types of threats that may take advantage of those vulnerabilities.

With varying types of vehicle bus architecture and varying types of OBD-II devices, we use a simplified diagram (Figure 4*)* to present potential connections in the vehicle. Each ECU in Figure 4 represents the one or many connected ECUs on the same bus as the OBD-II port. The ability to control an ECU results in attacker control of that vehicle's function.

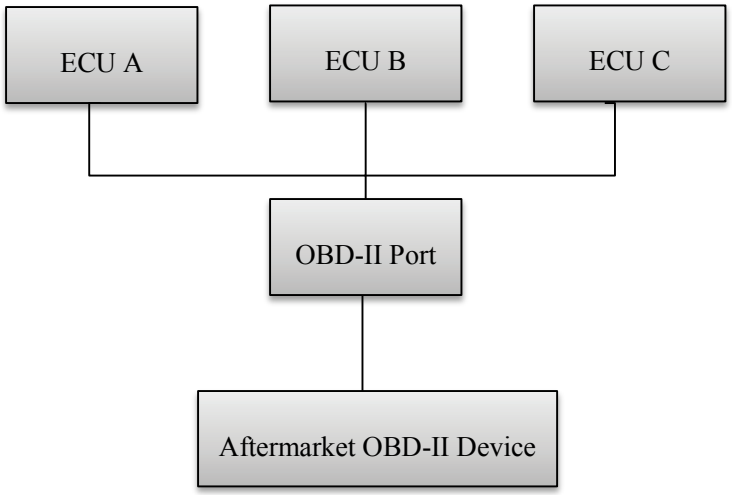

Figure 4: Generic OBD-II Device Threat Model Diagram

We begin by analyzing the impacts of various attacks assuming the OBD-II device can be compromised and an attacker can execute arbitrary code. Although each attack is the same, the impact depends on the capabilities of the device (e.g., how far away the attacker needs to be). Once the attack proximity and vulnerability are defined, the vulnerability is classified using Microsoft's STRIDE technique (Microsoft, 2005). We also use the Society of Automotive Engineers (SAE) safety, privacy, financial, and operational impact to define how a vulnerability may affect a vehicle (Ward, et al., 2013). (Both STRIDE and the SAE techniques are described in Appendix D.)

Table 2: Vulnerability Impact on the Device and the Vehicle

Vulnerability	**ECU Affected**	**Comments**	**Vulnerability Impact (STRIDE)**						**Impact (Ward, et al., 2013)**			
Hardcoded credentials	None							X	S0	S0	S0	S0
Arbitrary command injection	OBD-connected buses					X			S0	S3	S0	S0

Vulnerability	ECU Affected	Comments	Vulnerability Impact (STRIDE)						Impact (Ward, et al., 2013)			
Arbitrary CAN injection	OBD-connected buses	Full device compromise (See Table 3 for complete impact.)	X	X	X	X	X	X				

Table 3: Vulnerability Impact on Vehicle with Complete Device Compromise by Proximity

Vulnerability	ECU Affected	Proximity	Vulnerability Impact (STRIDE)						Impact (Ward, et al., 2013)			
			S	**T**	**R**	**I**	**D**	**E**	S	P	F	O
Compromise of OBD-II device	OBD-connected buses	Physical	X	X	X	X	X	X	S1	S1	S2	S2
Compromise of OBD-II device	OBD-connected buses	Short range (Bluetooth)	X	X	X	X	X	X	S2	S2	S3	S3
Compromise of OBD-II device	OBD-connected buses	Long range (Wi-Fi)	X	X	X	X	X	X	S2	S2	S3	S3
Compromise of OBD-II device	OBD-connected buses	Anywhere (cellular)	X	X	X	X	X	X	S4	S4	S4	S4

4 Vulnerabilities in Aftermarket OBD-II Devices

The risks raised by network security problems in vehicles are somewhat different from those of traditional computing platforms such as PCs or mobile devices. While data integrity and confidentiality are still concerns, security problems in vehicles can lead to physical effects. The risk of injury or even death to the occupants of the vehicle or to people nearby is a level of risk that isn't faced by traditional computing platforms.

While the complexity of the modern-day automobile is significant and automotive manufacturers maintain tight control over internal vehicle design and operations, security researchers are becoming more versed in the operation of the vehicle. It is only a matter of time before information becomes more widespread and further vulnerabilities are discovered and possibly weaponized by malicious attackers.

Potential safety-critical risks include[1]

1. Driver distractions (volume, wipers, etc.)
2. Engine shutoff or degradation
3. Steering changes (in drive-by-wire vehicles)

There are other, less safety-critical risks, some of which are fairly unique to vehicles:

1. Theft of the car or its contents
2. Enabling physical crimes against the occupants
3. Insurance or lease fraud
4. Eavesdropping on the occupants
5. Theft of information (e.g., phone list)
6. Vector for attacking mobile devices in the car
7. Theft of personally identifiable information (PII)
8. Tracking the vehicle's location

In enterprise IT environments, the majority of attackers are assumed to be remote, attacking the systems over the Internet. A specific automobile would be difficult to identify on the Internet, if it is directly accessible at all. Attackers are also likely to use computer security vulnerabilities as enablers of other, more physical crimes. Therefore, the threat actors are likely to be local to a targeted vehicle, generally within Wi-Fi or Bluetooth range. This doesn't rule out remote attacks, as a compromised

[1] Note: These risks are difficult to discuss in a general fashion because vehicles can differ markedly between makes and models. In Miller's work, his system was able to cause engine shutoff, volume modification, and manipulation of the wipers (Miller, 2015). We extrapolate that similar effects might be caused in other attacks.

mobile device with Internet connectivity could be connected to the car via an OBD-II device, USB, Bluetooth, or Wi-Fi.

A secondary risk of using these devices is that compromise of the manufacturer or operator's back-end server may allow an attacker to access any device connecting to its network. When a consumer decides to plug one of these devices into their car, they are unintentionally moving the security boundary from the vehicle itself to the device manufacturer's network, associated services, and any other connected device.

Vulnerabilities discovered through this research include

- Insecure firmware updates and downloads
- Hardcoded or non-existent Bluetooth PINs
- Weak WPA2 passwords
- Hardcoded credentials
- An Internet-enabled administrative interface

In our research, we were able to successfully conduct the following attacks:

- Arbitrarily modify firmware
- Maliciously update remote firmware
- Lock/unlock doors
- Turn on/off vehicle
- Affect vehicle GPS tracking as well as speed, heading, and altitude
- Read the car's internal data: temperatures, fuel levels, diagnostic trouble codes, etc.
- Inject arbitrary CAN packets

5 Common Architectural Issues

Aftermarket OBD-II devices have the potential to introduce serious safety and security risks to an automobile. The design of the OBD-II port means that such a device has unlimited access to some or all of a car's internal networks. These OBD-II devices also have some sort of external interface that is accessible from outside of the car—typically Wi-Fi, Bluetooth, or cellular. We broke this fundamental architectural risk into several more specific concerns.

5.1 The Primary Processor

For the most part, these devices have a simple processor at their core that merely transfers commands from some sort of external network protocol to CAN frames and vice versa. They generally leave any logic to the upstream app, whether that's on a nearby mobile device or remotely on a cloud service. They do not include any sort of security, such as authentication or checking whether any given command should be accepted by the vehicle.

5.2 The External Network Interface

Because the processor and the OBD-II port itself generally do not perform any filtering, the external network interface is the only real security on these devices. The security on this interface varies widely, from WPA2 with strong, unique passwords, to Bluetooth with easy-to-guess, widely shared PINs. This aspect of OBD-II devices is likely to be the primary differentiator in security between different products.

5.3 Undocumented Features

Some of these devices had undocumented features, such as Bluetooth, that were not listed in the specifications. The purpose these features is probably for testing, but their presence presents an additional risk to the user.

5.4 Insecure Firmware Updates

The devices tested contained the ability to be remotely updated. While this is a desired feature from a security perspective, the update mechanism needs to be properly designed to prevent an attacker from sending a malicious update to the device. Only one of the devices tested used end-to-end encryption and cryptographically signed their updates.

6 Recommendations

The result of our testing and analysis is the following recommendations.

Separate CAN communication from the network stack.

In some of the devices tested, CAN frames were passed directly from the vehicle through the device to the external network. Directly passing data to the external network is very dangerous and can allow an attacker to submit arbitrary CAN commands to ECUs on the bus. This functionality should be separated from the network stack and allow an application to send a request only from a list of pre-chosen OBD-II commands.

Sign and encrypt firmware updates.

Firmware updates should always be cryptographically signed and encrypted to prevent firmware modification by an attacker. OBD-II and telematics device vendors should consider using The Update Framework (https://theupdateframework.github.io/) or another open, well-supported standard for doing software updates.

Be secure by default.

In the course of investigating these devices, there is a supply-chain relationship between device resellers (who are really selling the use of their web-based service) and the actual device OEM. In our conversations with several vendors, we found that they sell their products as blank, development devices with very weak or non-existent security. They expect the reseller to then choose a secure configuration for the device before selling the device to the customer. Unfortunately, it appears that some resellers are unaware that this configuration is necessary and simply ship these devices without a secure configuration. We recommend that the OEMs provide the device to the resellers already within the most secure configuration. The reseller can then modify the configuration as needed, without needing specialized expertise to modify the device configuration.

Obey the principle of least privilege.

Devices should provide only enough connectivity to enable it to operate for its desired function. For example, if the device does not need access to manufacturer networks (i.e., General Motors LAN), then the physical connector itself should be removed. Additional features introduce additional risk.

7 Conclusion

As part of the CERT standard policy to disclose vulnerabilities,[2] we reached out to the affected vendors and disclosed the vulnerabilities directly to them. If we were unable to reach the vendor (and to inform the public), we posted them in our public vulnerability database.[3]

All software has vulnerabilities, but OBD-II devices (and any other cyber-physical device) are an important focus because of the potential physical impact on the driver. It is important to point out, however, that our findings were not all bad. After studying this sampling of devices, we found some of them did the *right* thing by default. The Automatic Link used signed, encrypted firmware updates and the Wi-Fi Link MX used a hardware button to enable the configuration web server.

Some questions consumers, fleet managers, and buyers need to consider before buying these devices are

- How is the device updated?
- Does the device use strong encryption when updating?
- Does the device send CAN traffic to the Internet?
- Does the vendor have a vulnerability disclosure policy?

As vehicle telematics get more complex and devices such as these become ubiquitous or built-in, manufacturers need to build security in and learn the lessons traditional information technology and mobile computing already learned. We can't afford to be complacent when the impact is on human lives and property.

[2] http://www.cert.org/vulnerability-analysis/vul-disclosure.cfm

[3] http://www.kb.cert.org/vuls

Bibliography

Andy Greenberg Hackers Remotely Kill a Jeep on the Highway - with Me in It [Online] // Wired. - 07 21, 2015. - 11 30, 2015. - www.wired.com/2015/07/hackers-remotely-kill-jeep-highway.

Charette R. N. This Car Runs on Code [Journal] // IEEE Spectrum. - 2009. - 3 : Vol. 46. - p. 3.

ISO ISO 11898-3 - Road Vehicles -- Controller Area Network (CAN) [Report]. - [s.l.] : International Standards Organization, 2015.

Koscher K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., ... & Savage, S Experimental Security Analysis of a Modern Automobile [Journal] // IEEE Security and Privacy. - 05 2010. - pp. 447-462.

Microsoft The STRIDE Threat Model [Online] // Microsoft. - 2005. - 11 30, 2015. - https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx.

Miller C. Valasek, C. Adventures in Automotive Networks and Control Units [Conference] // DEF CON 21. - Las Vegas : [s.n.], 2013. - pp. 260-264.

Miller C. Valasek, C. Remote Exploitation of an Unaltered Passenger Vehicle [Conference] // Black Hat USA. - Las Vegas : [s.n.], 2015.

OTAQ On Board Diagnostics (OBD) Basic Information [Online] // United States Environmental Protection Agency. - EPA Office of Transportation and Air Quality, 1996. - 11 30, 2015. - http://www3.epa.gov/obd/basic.htm.

SAE Standard SAE J1979 -- Vehicle Diagnostic Test Modes [Report] : Standard. - [s.l.] : Society of Automotive Engineers, 2002. - p. 128.

Ward David, Ibarra Ireri and Ruddle Alastair Threat Analysis and Risk Assessment in Automotive Cyber Security [Journal] // SAE Int. J. Passeng. Cars – Electron. Electr. Syst. - 2013. - 2 : Vol. 6.

Appendix A: Testing Methodology

Device Selection Criteria

The initial testing was a general overview of the space rather than comprehensive tests of many devices. The goal was to educate consumers, manufacturers, and fleet managers on potential security issues and their impact. Therefore, we chose several consumer devices at different price points with different types of connectivity. We also selected one common commercial device used for fleet management and one device that tracks a car without using the OBD-II port at all. Finally, we examined several development devices; these are left unnamed as the vulnerabilities were fixed prior to deployment.

Research Questions

The goal of this work was to answer the following questions:

- Are there vulnerabilities present in OBD-II aftermarket devices?
- Are there commonalities in the design of these aftermarket devices that makes them secure or insecure?
- How much access do these devices have to the CAN bus?
- If an OBD-II device is compromised, how much control can it have over the vehicle's functions?
- What, if any, safeguards should be considered to maintain safe and assured operation of a vehicle?

Testing Infrastructure

- 12V Batteries connected via leads to the OBD-II device
- 2-node CAN network from Microchip
- Raspberry Pi model B+ with wireless dongle running a custom Raspbian OS (working title: "CERT IOTA")
- Mac Laptop with VMWare Fusion
- Kali Linux and CERT TAPIOCA VMs
- Ubertooth Bluetooth Sniffer
- 2013 Chevy Volt
- OBD-II Splitter

Appendix B: Test Plan

Examine External Connections

External device communication can contain vulnerabilities in the underlying wireless implementation.

- Types of connections offered by these aftermarket OBD-II devices were
  - 802.11x wireless
  - Bluetooth and Bluetooth LE
  - Cellular (GSM/CDMA[4])

The steps taken were

- Using the Ubertooth, we examined the pairing process and key exchange.
- Using airmon-ng on Kali, we examined the wifi security.

Architectural Issues

These devices operate in one of two ways to maintain connectivity:

1. Device-to-phone-to-Internet connection via an app (Automatic)
2. Device-to-Internet connection directly via cellular (Delphi Connect)

Both methods of communication can introduce vulnerabilities into the system as a whole. We looked for Secure Sockets Layer (SSL) implementation errors, default or static passwords, etc.

Test Cases

1. Initial boot up
   - Device is given power
2. Pre-configuration phase
   - Device asks for default credentials
3. Post-configuration phase
   - User configures the device
4. Update process
   - Either via automatic update or manual update
5. Modify behavior via App
   - Send commands in the same format that the app uses
6. Modify behavior via CAN

[4] GSM is an acronym for Global System for Mobile; CDMA is an acronym for Code Division Multiple Access.

- Send arbitrary CAN commands to the device

7. Modify behavior via network
   - Modify (man-in-the-middle) the network upstream from the app or device

The steps taken were

- Full packet capture of the device-to-phone connection
- Full packet capture of the app's connections to the server

Content Issues

We suspect that some of these devices encapsulate CAN packets directly for further processing at the server. If so, can they be sniffed and used to gain information about the vehicle?

The steps taken were

- We examined full-packet capture for the encapsulated CAN traffic.

Examine Internal Connections

One goal of this test is to understand what traffic is being passed to and from the CAN bus via the OBD-II port. Prior research showed that it may be difficult to send adversarial CAN traffic to the CAN bus without using an intact vehicle. Without the full, interoperable ECUs on the bus they are programmed to communicate on, it is difficult to test the effect of inserting CAN packets independent of the vehicle.

Passive Communications

While the Linux kernel supports CAN natively, testing shows that it is unable to communicate directly with a vehicle CAN bus due to differences in speed (socketcan on Linux is an emulation of CAN; it isn't the same as a natively running CAN bus).

Instead, an OBD-II splitter was used on the vehicle, with one side plugged into the vehicle dash and the other side split between the dongle-under-test and a laptop with a ELM327 adapter to capture the responses. CAN is a broadcast protocol, so we expected the data to be reflected back to the laptop.

- We tested CAN connectivity from the vehicle to the laptop/pi and from the OBD-II device to the CAN bus.
- We captured CAN commands as they were sent and manipulated by the tester.

Appendix C: Test Results

Below is a list of the devices we tested, their capabilities, and the testing results.

Table 4: List of Devices Tested

Device Under Test	Bluetooth	Wi-Fi	Cellular	Results
Audiovox Car Connection	No	No	Yes	Unable to test fully; cellular connection only
Automatic Link	Yes	No	No	No vulnerabilities found
Cell Control	Yes	No	No	N/A (No OBD-II)
Delphi (Verizon) Connect	No	Yes	Yes	Weak WPA2 passcode; open ports on Wi-Fi; WPA2 shared password with admin interface
Wi-Fi Link MX	No	Yes		No vulnerabilities found; Bluetooth version available but not tested
[Development Devices]	No (optional)	No	Yes	Remote firmware update via SMS command; no firmware encryption (CRC check only)
Lemur Blue Driver	Yes	No	No	No security at all; Passes arbitrary CAN to the vehicle
Generic	Yes	Yes	No	Poor (passcode of 1234) or no security at all; Either Bluetooth or Wi-Fi

Below are additional notes related to our testing of these devices.

Audiovox Car Connection

Website: http://www.mycar-connection.com/

The only connectivity on this device was cellular, so we did not monitor the traffic of the device itself, only the mobile app. Using the CERT Tapioca tool, we found that the app did not validate SSL certificates; however, firmware updates to the device were done over the cellular connection, so it is not known whether those updates used SSL. (We did not commence a man-in-the-middle attack of the cellular connection.)

Automatic Link

Website: https://www.automatic.com/

Device Power

Two leads with alligator clips were connected from PIN 16 and 5 to a 12V battery. The battery was a 12V lantern battery used to simulate the same 12V power that the vehicle provides to the device. There is a short “chirp” when the device is connected.

Bluetooth Testing

Each Automatic is printed with a 6-digit PIN on the back of the device for use when connecting to it. According to prior research, these pre-printed PINs are vulnerable to re-pairing attacks.

App Testing

The Automatic Link has a mobile app that pairs to the dongle, providing Internet connectivity to the device and displaying information to the user. The app requires the user to create an account before pairing an Automatic Link with a phone. The registration process asks for your name, a username, and password to create an account. Once created, it asks the user for the 6 digit PIN printed on the back of the device. After this is entered, the app asks the user to plug the dongle into the vehicle. These steps are the start of the pairing process. From trial and error, we noticed that the dongle shuts off the pairing attempt after a few minutes, which is re-enabled after cycling the power. The phone must have connectivity to the Internet for the pairing process to work. It is possible that there is a weakness in how the pre-printed Bluetooth PIN is generated, but this was not exploitable. No other vulnerabilities were found.

Development Devices

CERT/CC tested various devices considered to be "development" or "unconfigured". These devices had very little to no security, with open ports on the WAN (cellular) side and the use of SMS commands to update firmware. The SMS commands required no authentication and would attempt to download firmware from any chosen IP address, in the clear, without cryptographic verification. While these devices are considered "development," there have been reports of the reseller being unable (or unwilling) to properly secure the devices they purchased from the manufacturer, leading to default configurations being deployed to the end user. CERT/CC believes that these devices should be sold secure by default, with the reseller being explicitly forced to open ports and enable services.

Lemur Blue Driver

Website: http://www.lemurmonitors.com/

This device is a Bluetooth-only device with an associated app. It does not require a password or PIN for Bluetooth pairing, and allows a simple serial connection over Bluetooth. After this connection, a user within Bluetooth range can send any AT, CAN, or OBD-II command to the vehicle's internal network. The Blue Lemur app does not use SSL to download new firmware.

Cell Control

Website: https://www.cellcontrol.com/

Note: This device does not connect to the OBD-II port and definitely cannot be used to control the car!

The Cell Control is a device that attaches to your windshield and connects to cell phones nearby via Bluetooth. The app also connects to the Internet. The primary purpose of the device is to monitor younger drivers and disable their cell phone (texting / calling) while driving. This device has a GPS so

it could theoretically be used to track a vehicle. However, we did not examine it in depth since it doesn't connect to the OBD-II port and has no access to the vehicle's internal network.

OBDLink MX Wi-Fi

Website: https://www.scantool.net/obdlink-mxbt/

The OBDLink MX is a Wi-Fi-only OBD-II device. It allows access only to the car's internal network and does not act as a Wi-Fi hotspot providing Internet access. It can pair with multiple mobile and desktop apps.

We found no vulnerabilities in the device as configured. However, we are concerned that users will either change the Wi-Fi password to something simpler or connect the device to another Wi-Fi Access Point, such as a mobile hotspot in the car's telematics unit. These possibilities raise two risks: (1) that the Wi-Fi password chosen will be too weak and (2) that the device will now be directly connected to the Internet.

(Verizon) Delphi Connect

Website: http://www.delphiconnect.com/

The Connect contains a cellular modem and serves two purposes. It connects to the OBD-II port and provides diagnostic information about the vehicle to Verizon web servers. These servers then send the information back to the user's mobile device. The second purpose is to provide a mobile hotspot within the vehicle so that mobile devices can use the Internet connection over the same cellular mode.

There were not any vulnerabilities that allowed us to access the vehicle's internal network. All access to the OBD-II port is through the cellular connection. The device does not appear to listen on any ports on the Internet, although we could ping it.

The Wi-Fi hotspot has a unique but weak WPA2 password that was crackable in ~2 days. Cracking the password gave us access to the Internet connection as well as the traffic of any device connected to it. The Wi-Fi interface also listens on ports 53 and 80. There is an admin interface on port 80 using the same password as the Wi-Fi WPA2 password. The admin interface allowed us further access to the connected devices but no access to the OBD-II port. We were unable to access the underlying operating system or load malicious software.

Generic OBD-II Devices

There are many low-cost (~$10) OBD-II devices available on sites such as Amazon.com, EBay, and Ali Baba. These use many different brand names, but are all very similar. They consist of a CAN controller (usually ELM-327 or compatible) connected via serial to either a Bluetooth or Wifi chip. We physically tested one sample device and reviewed the documentation for a number of them. They all used simple, hardcoded passwords or PINS, such as “1234,” “0000,” or “12345678.” They cannot be updated. We classed all of these together in this report.

After Bluetooth or Wi-Fi pairing, these can be easily accessed as a serial port. After connecting, a user can send any valid AT, CAN, or OBD-II commands.

Appendix D: Impact Scoring

Table 5: Vulnerability Impacts from SAE (Ward, et al., 2013)

Class	Safety-Related Severity
S0	No Injuries
S1	Light or moderate injuries
S2	Severe and life-threatening injuries (survival probable) *Light or moderate injuries for multiple people*
S3	Life threatening (survival uncertain) or fatal injuries *Severe injuries for multiple people*
S4	Life threatening or fatal injuries for multiple people

Class	Financial-Related Severity
S0	No financial loss
S1	Low-level loss (~$10)
S2	Moderate loss (~$100) *Low losses for multiple people*
S3	Heavy loss (~$1,000) *Moderate losses for multiple people*
S4	Heavy losses for multiple people

Class	Privacy-Related Severity
S0	No unauthorized access to data
S1	Anonymous data only
S2	Identification (personally identifiable information) of person or technology *Anonymous data for multiple people*
S3	Tracking of individual or technology *Identification of multiple people or technologies*
S4	Tracking of multiple people or technologies

Class	Operational-Related Severity
S0	No impact on operational performance
S1	Impact not discernible to user
S2	User aware of performance degradation *Indiscernible impacts for multiple users*
S3	Significant impact on performance *Noticeable impact for multiple users*
S4	Significant impact for multiple users

The STRIDE Threat Model (Microsoft, 2005)

- **Spoofing identity**. An example of identity spoofing is illegally accessing and then using another user's authentication information, such as username and password.
- **Tampering with data**. Data tampering involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet.
- **Repudiation**. Repudiation threats are associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations.
- **Nonrepudiation** refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package.

- **Information disclosure**. Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to or the ability of an intruder to read data in transit between two computers.
- **Denial of service**. Denial of service (DoS) attacks deny service to valid users—for example, by making a web server temporarily unavailable or unusable. You must protect against certain types of DoS threats simply to improve system availability and reliability.
- **Elevation of privilege**. In this type of threat, an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and becomes part of the trusted system itself, a dangerous situation indeed.