Global Information Assurance Certification Paper

Developments in Car Hacking

GIAC (GSEC) Gold Certification

Author: Roderick Currie, roderick.h.currie@gmail.com
Advisor: Manuel Santander
Accepted: December 5th, 2015

Abstract

The modern automobile is a complex network of information systems. As the car becomes increasingly computerized, so too does its attack surface increase. With security researchers recently demonstrating a series of vulnerabilities in automotive systems, it is becoming clear that auto manufacturers have not placed enough emphasis on developing secure vehicular information systems. Indeed, the field of automotive systems security is in its infancy. Automakers are struggling to come to grips with the need to secure vehicles' on-board systems. As a result, the modern passenger vehicle presents an attractive target, with a broad attack surface and a multitude of potential vulnerabilities to exploit. This paper analyzes recent research into automotive security vulnerabilities and reflects on the need for more secure vehicles.

1. Introduction

In the developed world, there is arguably no appliance more prevalent in people's lives than the automobile. Certainly, in the United States, there is scarcely a household without access to a car or some other form of vehicular transportation. Statistics from the United States Department of Transportation (2015) show that in recent years, over 250 million highway vehicles have been registered per year in the U.S. And, this number continues to grow steadily.

Growing at an even faster rate is the prevalence of computers in modern automobiles. Computers began to find their way into passenger vehicles in the early 1980s with the advent of the Engine Control Unit (ECU). By managing basic engine functions, early ECUs brought about improvements in performance and fuel efficiency, while also lowering vehicle emissions (Eyal, 2007). A diagram of a basic ECU implementation is shown below in *Figure 1*.

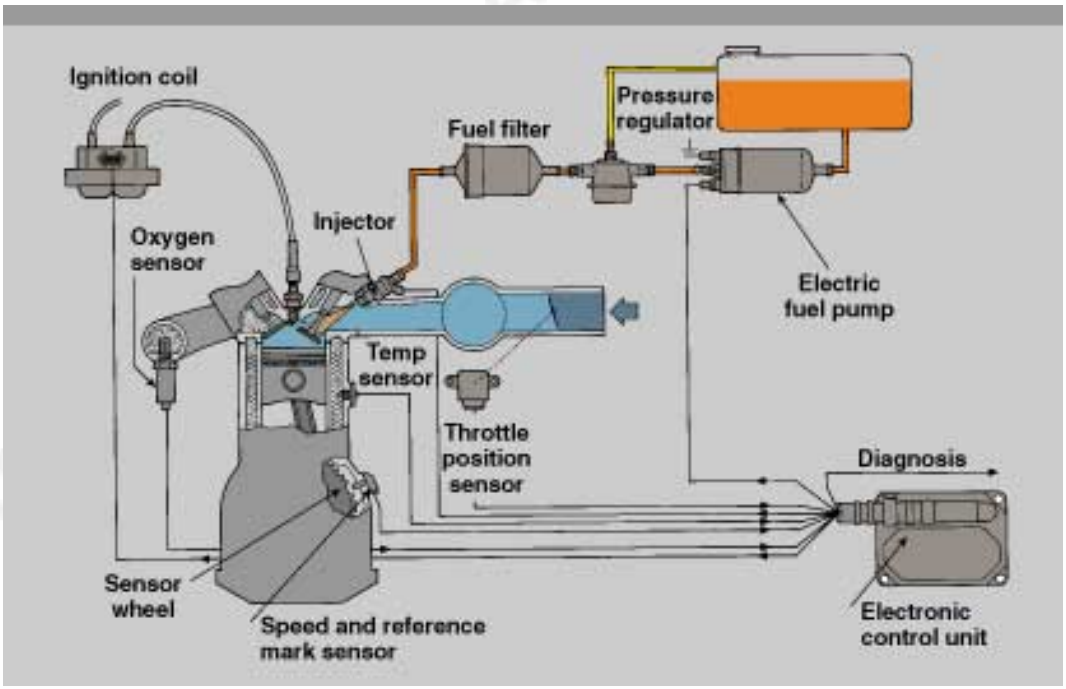

***Figure 1*: Engine Management System (Eyal, 2007)**

An ECU works by interfacing with different engine components including the fuel pump, fuel injectors, throttle body, spark plugs, and various sensors (Eyal, 2007).

Roderick Currie, roderick.h.currie@gmail.com

The ECU receives inputs from the sensors and makes adjustments to a series of actuators controlling the function of the engine's physical components accordingly. This allows for ignition timing and fuel/air mixture to be dynamically adjusted in real-time, which can save fuel and optimize performance. Prior to the use of ECUs for engine management, these functions had to be controlled mechanically (Eyal, 2007).

The Engine Control Unit has been found in almost every vehicle manufactured since the mid-80s (Micheli & Ernst, 2002). Unlike in the 1980s, however, the modern automobile now relies on computerization for almost everything from engine management to steering, braking, climate control, navigation, infotainment, and much more. A group of researchers recently found that the average modern high-end car now contains over 100 million lines of code – more than an F-35 fighter jet or a Boeing 787 passenger airliner (Doughty-White & Quick, 2015). This heavy reliance on computers, therefore, means that the modern automobile has a very broad attack surface and the potential for a plethora of vulnerabilities.

Today, in 2015, automotive systems security is a hot topic. Several recent high-profile car hacking demonstrations have brought media attention to the fact that the modern automobile is a largely insecure platform; an insecure platform that the public has come to rely on heavily, and one which plays an integral role in most people's lives. The implications of automotive system security vulnerabilities range from the mundane – such as an attacker being able to activate a vehicle's horn or flash its lights – to the downright terrifying – an attacker remotely gaining control of a vehicle and running it off the road at speed with occupants inside.

Therefore, now more than ever, automobile manufacturers are coming to realize that automotive systems security is a big deal. But, as of yet, there has not been a unified approach to the challenge of securing vehicular systems. Manufacturers are floundering when it comes to locking down their vehicles' systems, with some taking the approach of "security by obscurity" (Bourne, 2015), and yet others using litigation as a means to silence security researchers and keep vulnerabilities under wraps. In one recent case, Volkswagen engaged a team of European security researchers in a 2-year long legal battle to prevent the group from presenting its research paper on a vulnerability they had

found in Volkswagen's remote keyless entry system (Cimpanu, 2015). Unfortunately, this type of stance by the big automakers does nothing to improve vehicle security. It is clear that a more proactive approach is needed to address vulnerabilities and make the modern automobile a more secure platform overall.

2. The Interconnected Car

To understand the magnitude of the security problems facing the modern automobile, it is first necessary to address just how interconnected a modern vehicle is. The cutaway diagram in *Figure 2* provides a visualization of the many different vehicular components which are managed by a car's on-board computer systems.

***Figure 2*: Diagram of electronic components in a car, courtesy of the Mercedes-Benz Museum (as cited in Computer History Museum, 2011)**

In the diagram above, components associated with physical control of the vehicle are shown in blue, components associated with safety are shown in pink, and components associated with entertainment and convenience are shown in yellow. Unlike the early Engine Control Units of the 1980s, the computer systems in a modern automobile reach far beyond the engine. The interconnected components highlighted in *Figure 2* include the vehicle's engine management system, brake controller, airbags, seatbelt pre-tensioners, door locks, gauge cluster, sound system, CD changer, seat controls,

communications system, telematics unit, and more. Running throughout the vehicle is a network of wires on which sensor data and vehicle control commands transit back and forth. Also visible in *Figure 2* are several long rectangular boxes that represent controllers. These controllers are responsible for issuing commands to the different vehicular components based on the inputs they receive, either in the form of sensor data or commands from the vehicle operator.

Because of the interconnectivity between these different systems, an attacker could potentially enter through a vulnerability in the infotainment system before pivoting to exploit the vehicle's safety or control systems. Every component highlighted here has the potential to be exploited by an attacker. Likewise, every highlighted component represents a possible entry point or pivot point for an attacker. A weakness in any one of these components could grant an attacker access to the rest of the vehicle's systems.

3. The CAN Bus Architecture

At the heart of any modern vehicle's interconnected systems is the Controller Area Network bus, or CAN bus. The CAN bus is a single, centralized network bus on which all of a vehicle's data traffic is broadcast. The CAN bus carries everything from operator commands such as "roll down the windows" or "apply the brakes", to readouts from sensors reporting engine temperature or tire pressure. The advent of the CAN bus brought about improvements in efficiency and a reduction in complexity while also reducing wiring costs.

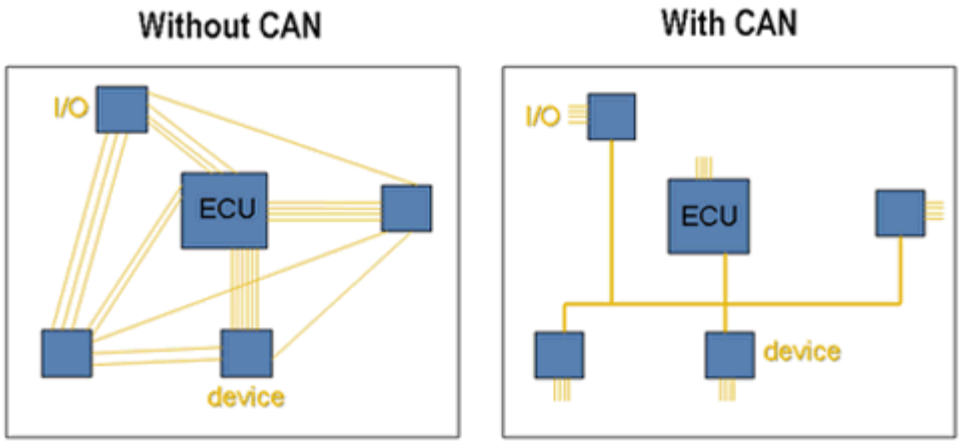

***Figure 3*: Can Networks Significantly Reduce Wiring (National Instruments, 2014)**

Prior to the development of CAN bus technology, any two vehicular components needing to communicate with each other would have required a dedicated point-to-point connection between them (National Instruments, 2014). The diagram shown above in *Figure 3* demonstrates how a CAN network can significantly reduce the amount of wiring needed in a vehicle by eliminating the old point-to-point topology in favor of a more efficient, centralized approach. Whereas the pre-CAN architecture diagram places the ECU at the center of the logical network, the CAN diagram highlights the network bus itself as the focal point, removing point-to-point connections between devices and lessening the involvement of the ECU (National Instruments, 2014).

With the majority of a vehicle's computerized components being centrally connected via CAN, the bus receives a high amount of traffic. Eric Paton, a technical specialist at Ford, describes the design of the CAN bus as being "similar to that of a freeway system [where] data move like vehicles from high-traffic highways to local roads via on and off ramps" (as cited in Wojdyla, 2012). The diagram shown in *Figure 4* builds on Eric Paton's analogy of the CAN bus as a freeway system, with each component or device connecting to the central bus via an "on-ramp".

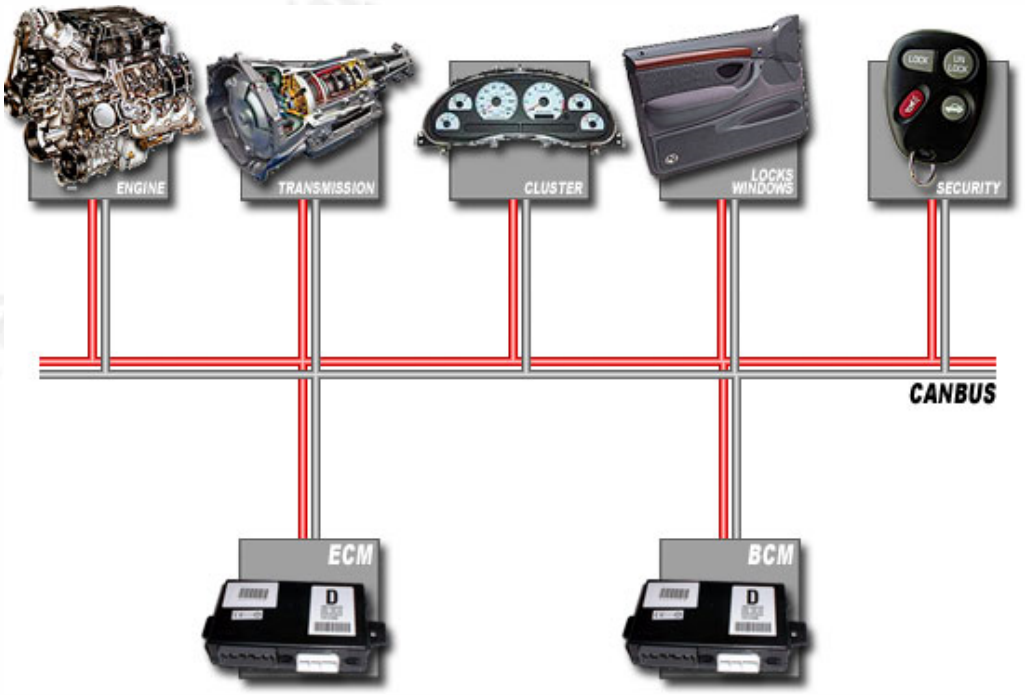

***Figure 4*: CAN Bus Network (Fortin Electronic Systems, 2006)**

What sets the CAN bus apart from other common network bus topologies is that data constantly flows on the CAN bus whether it is actually requested or not. An example of this can be illustrated using the vehicle's tachometer, which displays the number of revolutions per minute (RPM) being performed by the engine. For the tachometer to display the engine RPM, it does not first need to send a query to the engine; instead, the engine's ECU constantly broadcasts the engine RPM out over the CAN bus to any listening controllers. All the tachometer's controller needs to do is monitor CAN bus traffic for RPM messages, and when one is detected, the tachometer's display is updated with the new information. By repeating this action many times per second, the driver sees a tachometer readout that appears to change dynamically in real time.

Because of varying implementations of CAN protocols, the message size and broadcast frequency can vary. In the tachometer example, the ECU might encode the RPM data into a 1-byte message to be broadcast on the CAN bus every 100 milliseconds (Tindell, Burns & Wellings, 1995). In addition to this 1-byte of data, an 11-bit identifier is attached to the message (Tindell et al., 1995). It is this identifier that a host controller uses to determine whether or not a message should be picked up and processed. In this case, the tachometer's controller will be on the lookout for any CAN messages that contain an identifier known to be associated with RPM data. Only messages with that specific identifier will be received, processed, and the data displayed on the tachometer.

The ECU is just one of many different components constantly broadcasting data on the CAN bus, and RPM data is just one small piece of CAN data. Some other examples of data continually being broadcast on the CAN bus include cabin temperature, tire pressure, steering input, vehicle speed, and many more. In fact, at any given time, the CAN bus can be transporting up to 2,000 unique signals from different components of the car (Wojdyla, 2012). With CAN message sizes ranging from 1 byte to 8 bytes (Tindell, Burns & Wellings, 1995), and with thousands of CAN messages transiting the bus at the same time, the CAN bus is a very noisy and bandwidth-intensive environment.

At its core, CAN is a "high-integrity serial bus system for networking intelligent devices" (National Instruments, 2014) which has become a globally accepted standard for in-vehicle networking. CAN has been accepted as the international standard, ISO 11898

(National Instruments, 2014). CAN is lightweight and robust, allowing for additional components to be added easily to the CAN network without the need to modify existing components. The CAN protocol also allows for message prioritization, which is accomplished using the message ID field, and for error checking, which utilizes a cyclic redundancy check (CRC) field (National Instruments, 2014). It is these qualities and capabilities that have led CAN to be embraced as the modern standard for in-vehicle networking.

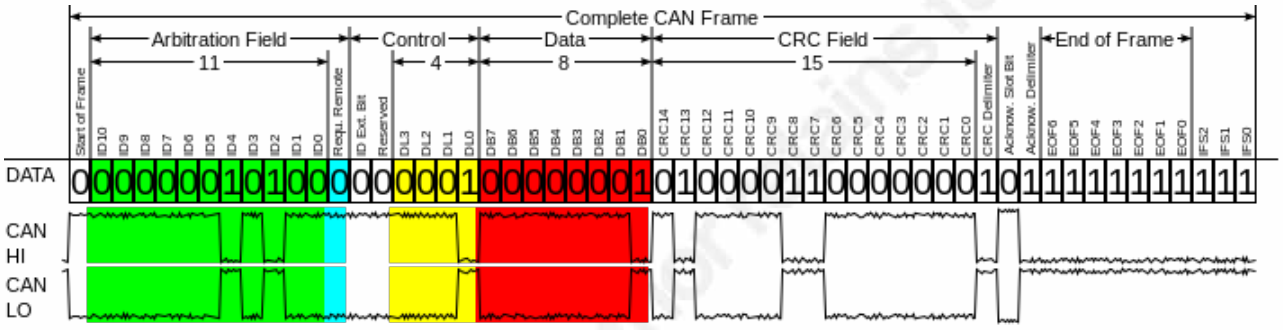

***Figure 5*: Complete CAN Frame (Wikipedia, 2014)**

The diagram shown above in *Figure 5* breaks down the structure of a standard CAN data frame. The first bit represents the start of a new data frame and is known as the SOF bit. The 11 bits that follow are used for the identifier. The next bit, shown in blue, is the remote transmission request (RTR) bit. Although CAN data is typically broadcast out onto the CAN bus without being solicited, it is also possible for a CAN controller to request data from another controller by utilizing the RTR bit. Following the RTR bit is an extended ID field bit and a reserved bit. Next comes a 4-bit data length field, shown in yellow, which is used to signify the length of the data which follows. The data portion, shown in red, comes next and can be anywhere from 0 to 64 bits (8 bytes) in length. Following the data field is a CRC field used for message integrity. The two bits that follow the CRC field are message acknowledgment (ACK) bits. Finally, the end of frame (EOF) bits are used to signify the end of the CAN data transmission (National Instruments, 2014).

Roderick Currie, roderick.h.currie@gmail.com

3.1. A Brief History of the CAN Bus

Development of the CAN bus protocol was begun in 1983 by German company Robert Bosch GmbH (Sewell Direct, 2015). After three years of development, CAN bus technology hit the public market in 1986, first showing up in the BMW 850 coupe (Sewell Direct, 2015). Up until the release of the 1986 BMW 850, a vehicle's various electrical components would have been connected through dedicated point-to-point wiring harnesses. This meant that as the number of electrical components increased, so too did the amount of wiring needed in the vehicle. The development of the CAN bus eliminated the need for most of that bulky wiring, resulting in a weight reduction of over 100 pounds for the 1986 model year 850 coupe (Sewell Direct, 2015).

Despite the obvious benefit of weight reduction, there was another driving factor in the development of CAN bus technology. Due to the more stringent vehicle emissions requirements of the late 1970s and early 1980s, the National Highway Traffic Safety Administration (NHTSA) and the California Air Resources Board (CARB) sought a way to easily and effectively monitor a vehicle's emission control systems (Wojdyla, 2012). These federal and state mandates led to the creation of the On-Board Diagnostics protocol, or OBD. The second generation of this protocol, which is still in use today, is known as OBD-II (Wojdyla, 2012). Since the development of OBD in the 1980s, automakers have been legally required to equip all new vehicles with the OBD protocol for diagnostic and emissions testing purposes. Because OBD relies on gathering readings from a wide range of different vehicle sensors, it became necessary to develop a centralized network bus through which these sensors and their respective controllers could all communicate; and so, the Controller Area Network (CAN) was born.

3.2. Inherently Insecure

The CAN bus is a 30-year old architecture that was developed for various valid reasons, but security certainly was not one of them. Automakers at the time could not possibly envision the risk of cars being hacked decades into the future, nor could the governing bodies that mandated the CAN and OBD standards. The CAN architecture was designed to be lightweight and robust, and those qualities it accomplishes very well. However, CAN contains numerous vulnerabilities that are inherent in its design.

3.2.1. Lack of Segmentation and Boundary Defense

Network segmentation is a fundamental part of secure system design. If a network is not segmented, a trivial vulnerability in a non-sensitive system component can be exploited to grant access to the rest of the network, including its most critical and sensitive parts. According to the Center for Internet Security's (CIS) Critical Controls, segmentation of the network is essential "because once inside a network, many intruders attempt to target the most sensitive machines" (Center for Internet Security, 2015, p. 42). The Critical Controls also advise that "protecting each segment with a proxy and a firewall will greatly reduce an intruder's access to the other parts of the network" (Center for Internet Security, 2015, p. 42).

Unfortunately, the CAN bus architecture fails to implement this important control. While the CAN bus utilizes many different Electronic Control Units (ECUs) to oversee different vehicle functions, all sections of the bus are connected back to a common ECU at some point (Baltieri, 2013). This means that the window switches have a potential path of communication to the brake controller, the entertainment system has a channel to communicate through to the vehicle's airbags, and so on. Had CAN bus technology been designed with security in mind, there would have been no legitimate reason that system components dealing with entertainment and convenience should have the ability to communicate directly with critical vehicle control systems. This is a direct consequence of a system being designed with convenience and efficiency as the primary drivers, with little (if any) thought given to the security implications.

3.2.2. Lack of Device Authentication

Another way in which the CAN bus is inherently vulnerable to attackers is the lack of device authentication on the network. The Controller Area Network – as the name implies – is a network of different controllers. Each controller serves a different function. Some controllers are used to broadcast data onto the bus; an example would be the engine control unit constantly sending a CAN message onto the network containing the current engine speed (RPM) (OpenXC, 2015). This data, once broadcast onto the bus, becomes available to all other vehicular components on the CAN bus whether they require that information or not. Other controllers on the CAN bus constantly listen for specific

messages; an example of this would be the controller for the gauge cluster, which constantly listens for RPM messages on the bus so it can update the vehicle's tachometer accordingly.

This architecture, under normal circumstances, works very well. However, the system does nothing to prevent unauthorized devices from joining the CAN bus and broadcasting messages out to any listening controllers. This inherent vulnerability is an attacker's dream. By gaining access to the CAN bus, an attacker can send spoofed messages over the bus. Because CAN utilizes its own native protocol, a would-be vehicle hacker must take the time to learn the CAN protocol before being able to launch an attack. A successful attacker must understand and be able to manipulate a CAN data frame's ID and data fields. An attacker's malicious data frames will then be picked up and processed by listening controllers. This gives the attacker the very real potential to cause not only mischief but possible harm to the vehicle and its occupants. To pull off such an attack effectively, an attacker need only understand the format of legitimate CAN messages.

CAN reconnaissance can be done by listening passively on the bus to record the different messages for different vehicle functions, and is trivial in its level of difficulty. Once an attacker understands the legitimate message format for the given vehicle, he can craft his own CAN messages to manipulate the vehicle. There are many third-party solutions available today which enable even a novice user to sniff traffic on the CAN bus. One such product is CANdo from netronics (CANdo, 2015). The CANdo suite of products includes a CAN to USB interface cable, a streamlined CANdo application to allow for ease of CAN data monitoring, a CANdo SDK which allows for programming of CAN messages, and a CANdo OBD-II cable which enables communication with the CAN bus through a vehicle's OBD (Onboard Diagnostic) port (CANdo, 2015).

A screenshot of netronics' CANdo application is shown below in *Figure 6*. The application provides a graphical user interface (GUI) which could easily be used by an attacker to decipher CAN messages with little prior CAN knowledge or experience. The application separates out the identifier, data, and message type fields, even providing color coding for different types of messages. For an attacker, learning a specific vehicle's

language is simply a matter of trial and error. For example, during the learning phase, an attacker could sniff CAN bus traffic using the CANdo application to determine which message activates a vehicle's headlights. By physically turning the headlights on and off multiple times while scanning the CAN bus for duplicate messages, it may be possible to determine which specific message corresponds to the instruction for "lights on" or "lights off." Once an attacker has figured out the specific message, the CANdo software can be used to replay that same message over the CAN bus to see if it does, in fact, activate the vehicle's lights.

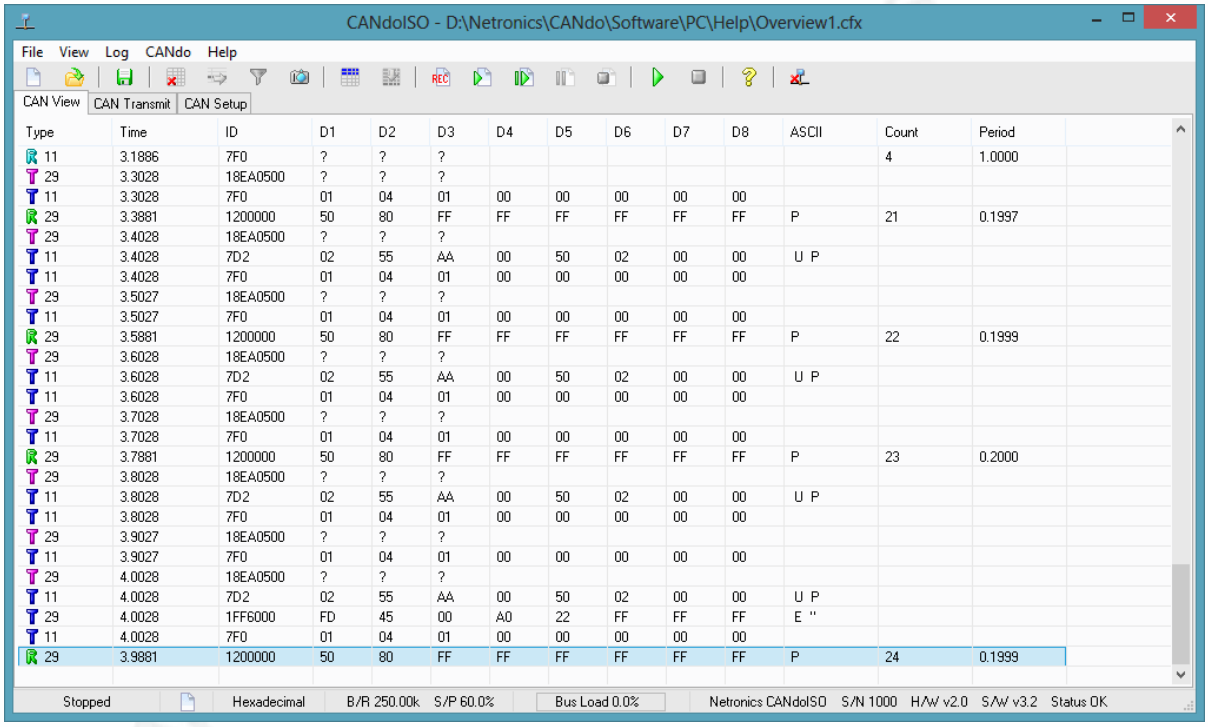

***Figure 6*: CANdo Application (CANdo, 2015)**

A legitimate instruction to turn a vehicle's headlights on or off should only ever come from the vehicle's own physical headlight switch. But because CAN was not designed with security in mind, it is possible for a rogue device on the CAN bus, such as an attacker's laptop, to impersonate the headlight switch – or any other CAN component. Such a vulnerability exists because the controllers on the CAN bus perform no authentication or validation of the devices sending messages. The lightweight CAN protocol only addresses message priority and message integrity but does not authenticate the sending device. The Critical Controls recommend that network level authentication

should be used to limit which devices can operate on the network (Center for Internet Security, 2015, p. 7). Further, the Critical Controls state that client certificates are a viable method for authenticating systems and preventing unauthorized devices from connecting to the network (Center for Internet Security, 2015, p. 7). An implementation of Network Access Control (NAC) and a client certificate infrastructure would require modification of the existing CAN bus architecture, but it would eliminate many existing CAN vulnerabilities.

3.2.3. Unencrypted Traffic

A further critical flaw in the design of the CAN bus is a complete lack of encryption. CAN is an unencrypted bus by design (Yoshida, 2015). Encryption was never taken into consideration when CAN was being developed, as any type of encryption implementation at the time would have only served to bloat CAN messages and clog the bus. Keep in mind that CAN was designed specifically to be lightweight and robust. And, again, car hacking was something that CAN's designers could not even envision being possible at the time of its inception in the mid-1980s.

Fast-forward to today, when car hacking is very much a reality, and the total lack of encryption on the CAN bus is now one of the system's core vulnerabilities. The implications of unencrypted CAN messages are twofold. First, a major weakness of unencrypted CAN traffic is that it can be sniffed. With the right hardware – which is readily available at a low price – an attacker can connect to the bus and passively sniff data in transit. Technology company netronics sells their CANdo suite of hardware and software for a few hundred dollars (CANdo, 2015). The necessary interface cable alone costs as little as $30 (CANdo, 2015). Utilizing a laptop computer, a CANdo interface cable, and the CANdo GUI-driven software application, an attacker may be able to determine which messages control which functions of the vehicle, essentially learning the vehicle's "language." This discovery phase is the first step in being able to craft spoofed CAN messages. The second step is the actual modification of CAN messages or the injection of entirely new ones. The lack of encryption, again, allows for this. Without some form of encryption, there is no way to guarantee message integrity or authenticity (Yoshida, 2015). And so, the car will carry on processing manipulated CAN messages as

if they were legitimate. Once an attacker is inside the network, one of the best ways to prevent him from reading or manipulating the data is with data encryption.

3.3. Security by Obscurity

The automobile industry has long relied on the principle of "security by obscurity", which is defined as "the reliance on the secrecy of the implementation of a system or components of a system to keep it secure" (OWASP, 2009). Unfortunately, this model almost always fails when it is the only security control a system has (OWASP, 2009). Until recently, automakers had assumed that no one would waste time trying to decipher proprietary CAN messages. Researchers have begun to show, however, that CAN messages can indeed be deciphered and modified for sinister means. In the words of security researcher Charlie Miller, "if the only thing keeping you from crashing your car is that no one is talking about this then you're really not safe anyway" (as cited in Greenberg, 2013). Researchers argue that the best way to pressure automakers into producing a more secure product is to demonstrate to the public just how potentially dangerous these vehicular security vulnerabilities truly are.

4. A Timeline of Recent Automotive Hacks

When the CAN bus was being developed in the mid-1980s, its designers certainly did not envision that one day the bus would be targeted by attackers seeking to take over or otherwise manipulate the function of an automobile. In fact, as recently as ten years ago, the concept of car hacking received little media attention and was not a worry to most vehicle consumers. It is only in the last decade, and particularly in the last several years, that car hacking has become a very real concern. In a 2015 study by Kelley Blue Book in which members of the car-buying public were polled, it was found that 78% of study participants believed vehicle hacking "will be a frequent problem in the next three years or less" (as cited in PR Newswire, 2015). This perception among the general public is due in large part to several recent high-profile vehicle hacks. The timeline below summarizes some of the more notable automotive hacks that have occurred recently.

4.1. 2010 – Vehicles Disabled Remotely Via Web Application

One of the first widely reported accounts of vehicle hacking came in 2010 when a disgruntled former employee of an Austin, Texas car dealership sought revenge against his former employer (Poulsen, 2010). In reality, this attack did not involve any hacking of the actual vehicles themselves. Nonetheless, the attacker was able to physically disable the vehicles of innocent owners without their knowledge or consent. The former dealership employee used stolen credentials to log into a web application that allowed remote access to functions of customers' vehicles including the engine immobilizer and the horn (Poulsen, 2010). This web application's intended purpose was to let dealership personnel immobilize the vehicles of customer who failed to make their loan payments on time. In fact, it ended up being used to cause mayhem as car owners found themselves locked out of their vehicles with the horns constantly honking (Poulsen, 2010).

The web application used by the dealership, in this case, was WebTeckPlus from Pay Technologies, LLC (Payteck, 2003). The WebTeckPlus application provides a web portal for dealership employees to interface with PayTeck electronic controllers installed in customers' vehicles. The PayTeck hardware consists of an electronic keypad and controller that is installed inside the customer's car. The controller is wired into the vehicle's engine immobilizer and horn. Each time a customer makes a payment on time, they are given a new code to enter into the electronic keypad. If the correct code is entered, the vehicle will continue to function as normal. If payment is late and a code is not entered into the keypad on time, the controller will activate the engine immobilizer, rendering the vehicle useless. The WebTeckPlus application also allows a dealership employee to log in and remotely disable a particular customer's vehicle at will, if necessary. The PayTeck hardware and WebTeckPlus software allows dealerships to save time and money by avoiding having to repossess vehicles.

This hack can best be summarized as an unauthorized intrusion of a web-based application, which is certainly nothing new. The perpetrator faced computer intrusion charges (Poulsen, 2010). However, this particular incident highlights the link between a vehicle's critical control systems and the modern, connected world, and shows how that link can potentially be exploited by someone with malicious intent.

4.2. 2010 & 2011 – CAESS Experimental Analysis

In 2010, a group of researchers from the Center for Automotive Embedded Systems Security (CAESS) – a joint venture between the University of Washington and the University of California, San Diego – released a research paper entitled "Experimental Security Analysis of a Modern Automobile" (Koscher et al., 2010). The team conducted a range of lab experiments and road tests and found that it was possible to manipulate a vehicle's functions by injecting messages on the CAN bus (Koscher et al., 2010). The researchers successfully demonstrated that a would-be attacker could disable the brakes, selectively brake individual wheels on demand, stop the engine, falsify information on the vehicle's speedometer, and more (Koscher et al., 2010).

Although the CAESS team highlighted serious security flaws in a modern automobile's systems, their research was largely met with criticism. At the time, automakers and the media alike claimed that it was neither realistic nor plausible for an attacker to have wired access to a vehicle's CAN bus to be able to carry out this type of attack in the real world (Miller & Valasek, 2015, p. 5).

The following year, in 2011, the CAESS team fired back with a new research paper entitled "Comprehensive Experimental Analyses of Automotive Attack Surfaces" (Checkoway et al., 2011). This paper was a response to the media skepticism surrounding the team's previous findings. The team acknowledged that the previous threat model of an attacker having physical access to a vehicle's internal network had "justifiably been viewed as unrealistic" (Checkoway et al., 2011). This time, the researchers sought to analyze the external attack surface of a modern vehicle and determine whether an attack could be carried out remotely.

In analyzing the attack surface of a modern car, the CAESS team created the illustration shown below in *Figure 7*. The infographic shows the different I/O channels on a modern automobile, with each one representing a potential entry point for an attacker. The "lightning bolt" symbols represent possible sources of remote wireless access and control. As automakers continue to increase the connectivity of their vehicles, the attack surface only broadens. The vehicle's cellular, Bluetooth, and Wi-Fi systems make particularly attractive entry points for a would-be attacker.

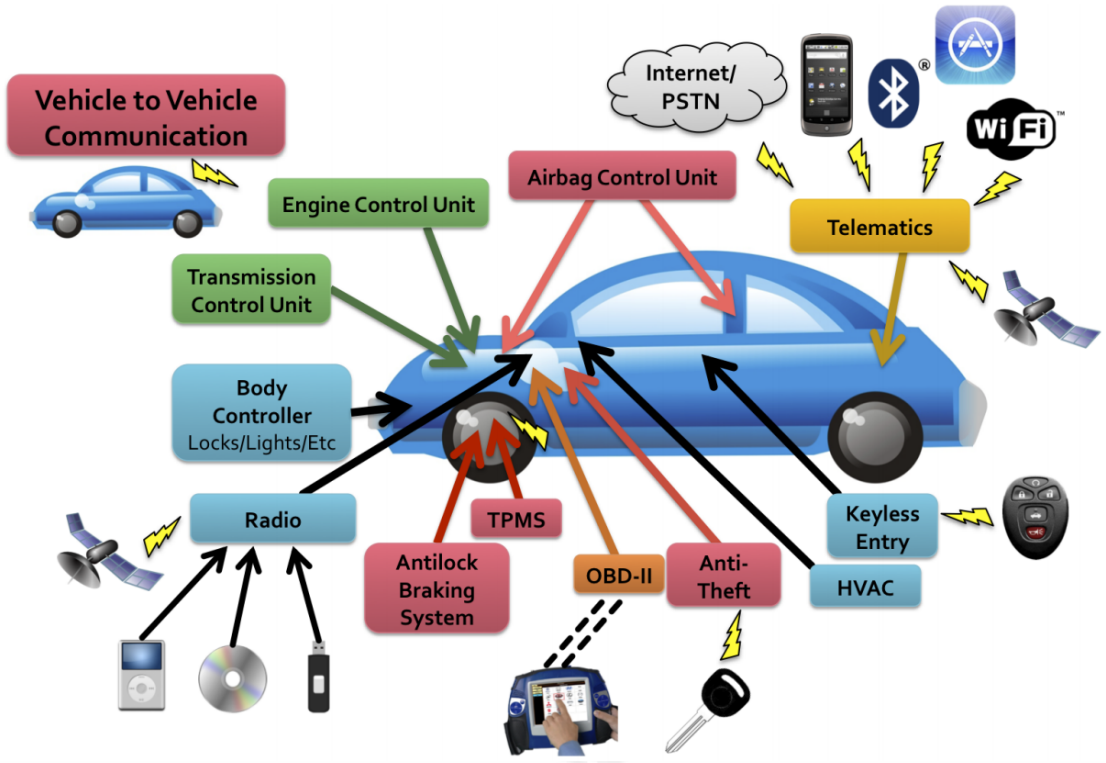

***Figure 7*: Digital I/O Channels on a Modern Car (Checkoway et al., 2011)**

Ultimately, the CAESS team found that it was possible to remotely exploit their test vehicle via a range of different vectors including the radio's MP3 parser, the vehicle's Bluetooth system, and the cellular connection used for the vehicle's telematics system (Checkoway et al., 2011). From there, CAN messages could be injected on the bus as had been demonstrated in the group's previous findings.

This research was hailed by some as groundbreaking because "it showed that vehicles were vulnerable to attacks from across the country, not just locally" (Miller & Valasek, 2015, p. 5). Despite having answered to their critics, the CAESS team's findings failed to garner much media attention or response from the auto industry. This was due in part to the fact that the researchers did not share *how* their exploits could be replicated, nor did they reveal the specific vehicle they tested on (Miller & Valasek, 2015, p. 5). While it is understandable that the research team would choose not to release the details of their exploits so as not to aid the "bad guys", this also made the findings a lot easier for automakers and the general public to shrug off.

4.3. 2013 – Miller & Valasek Physical Hack

A more recent high-profile case of vehicle hacking came from researchers Charlie Miller and Chris Valasek. Working with an $80,000 grant from the Defense Advanced Research Projects Agency (DARPA), Miller and Valasek were tasked with finding security vulnerabilities in automobiles and published their findings in 2013 (Greenberg, 2013). The duo conducted a series of real-time demonstrations for journalists and security professionals, before going on to present their findings at the Defcon conference in Las Vegas, Nevada that same year. Specifically, Miller and Valasek targeted the systems of a 2010 Ford Escape and a 2010 Toyota Prius (Greenberg, 2013). They were essentially able to reverse engineer the vehicles' CAN bus communications to demonstrate "everything from annoyances like uncontrollably blasting the horn to serious hazards like slamming on the Prius' brakes at high speeds" (Greenberg, 2013). The infographic in *Figure 8* lists many of the vehicular functions that Miller and Valasek were able to manipulate on their 2010 Toyota Prius test vehicle. Some of these capabilities, such as being able to jerk the steering wheel or slam on the brakes, propelled car hacking from a nuisance to a serious safety concern for automakers.

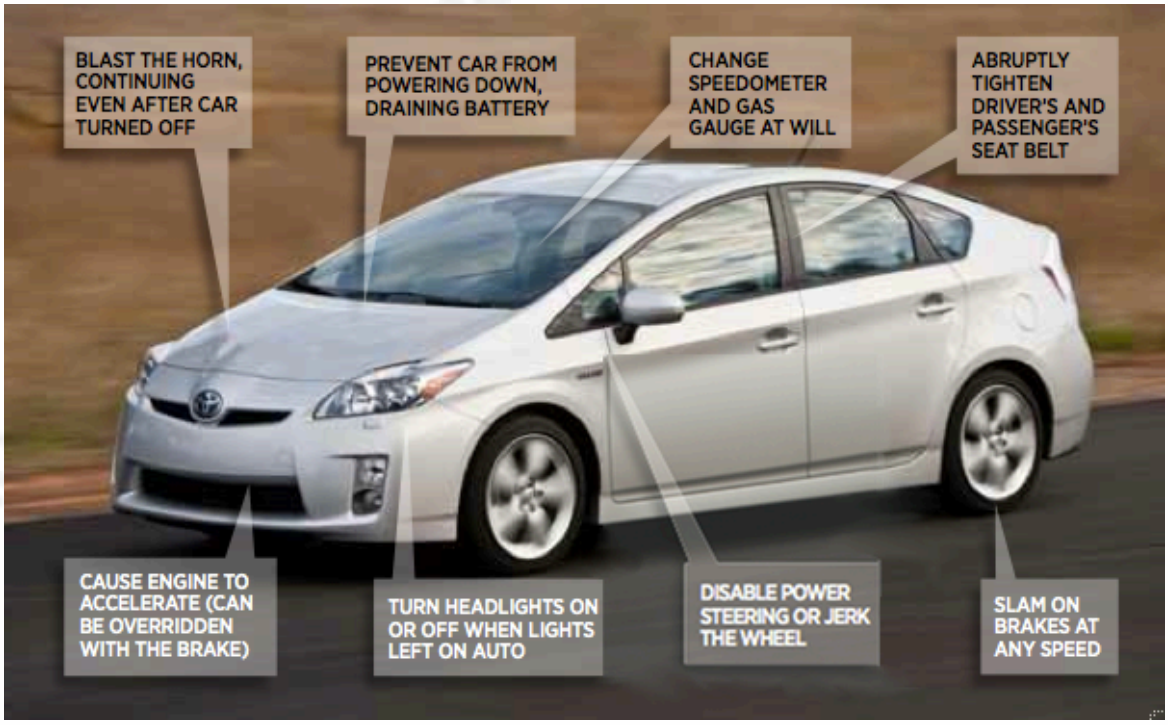

***Figure 8*: Anatomy of an Automotive Hack (Greenberg, 2013)**

Miller and Valasek's method involved using a laptop PC running Windows XP hooked into the vehicle's OBD-II port via a series of cables (Miller & Valasek, 2014, p. 23). The OBD-II port is traditionally used by mechanics and repair shops to retrieve fault codes and diagnose problems with a vehicle, but it also represents an attractive point of entry for a vehicle security researcher or an attacker performing reconnaissance. Miller and Valasek used a proprietary "ECOM" cable from EControls, Inc. which was hooked to their laptop via USB. They then fashioned a custom ECOM-to-OBD-II connector to allow them to interface with the car's OBD-II port (Miller & Valasek, 2014, p. 22). At this point, all Miller and Valasek had to do was listen and observe the CAN messages transiting the CAN bus to begin building a picture of which message corresponded to which vehicular functions. The next step was to use the connected laptop to replay captured CAN packets, recording the vehicle's response each time. Finally, the duo crafted modified CAN packets and were able to manipulate the behavior of the vehicle (Miller & Valasek, 2014, p. 26).

It is worth emphasizing again that in Miller and Valasek's 2013 car hacking demonstration, the researchers had physical access to the vehicle's CAN bus. The rationale behind this was that, according to the researchers, it had already been shown by prior scholarly research (Checkoway et al., 2011) that various interfaces such as Bluetooth or a vehicle's telematics unit could be hacked to allow for remote code execution (Miller & Valasek, 2014, p. 4). Considering the challenge of gaining remote access to be trivial, the researchers sought to find out what could be accomplished *after* access had been gained (Miller & Valasek, 2014, p. 4).

Following the release of Miller and Valasek's findings in 2013, the general public and big automakers seemingly failed to recognize the triviality of the prerequisite of gaining remote access to a vehicle's systems. Miller and Valasek faced skepticism for having demonstrated security flaws that required an attacker to be physically located inside the vehicle with a laptop hooked up to the car's data port, and – as in the case of the Prius demonstration – with the dashboard completely disassembled for ease of access (Greenberg, 2013). Indeed, in response to Miller and Valasek's work, Toyota's safety manager, John Hanson, argued that "[Toyota's] focus, and that of the entire auto industry, is to prevent hacking from a remote wireless device outside of the vehicle" (cited in

Greenberg, 2013), indicating that Toyota was largely unimpressed by the hacking demonstration. Hanson went on to state, "we believe our systems are robust and secure" (cited in Greenberg, 2013).

4.4. 2015 – Miller & Valasek Remote Hack

Charlie Miller and Chris Valasek made headlines again in 2015, this time for successfully demonstrating that an unaltered passenger vehicle – a 2014 Jeep Cherokee in this case – could be remotely exploited without the need for any physical access (Miller & Valasek, 2015, p.6). Unlike the duo's 2013 hack of a Toyota Prius and Ford Escape, this new research mimicked a real-world attack scenario in that it demonstrated both the ability to gain remote access and the ability to remotely execute code. And unlike the 2013 hack, which was largely met with incredulity by automakers, the 2015 hack prompted Fiat Chrysler Automobiles (FCA) to recall some 1.4 million vehicles for a critical security update and forced Sprint Corporation to enhance the security of its cellular carrier network (Miller & Valasek, 2015, p.87).

Miller and Valasek's Jeep hack took advantage of the vehicle's onboard connectivity features, in addition to the familiar lack of security controls on the CAN bus. Access was obtained through a vulnerability in Uconnect, a system that governs the vehicle's infotainment, navigation, built-in apps, and cellular communications (Greenberg, 2015a). What made the Uconnect system so attractive to the pair of researchers was that in addition to being a hotbed of connectivity, Uconnect also contains a microcontroller in its head unit which can communicate with other modules on the vehicle's CAN bus (Miller & Valasek, 2015, p.20). The hack also took advantage of a weakness in Sprint's cellular network, to which the vehicle's on-board telematics system was connected. The telematics system is used for real-time traffic data, in-car Wi-Fi, and other remote connectivity functions (Miller & Valasek, 2015, p.32).

Through port scanning, the pair found Uconnect's D-Bus port (6667) to be open. D-Bus, also known as Diagnostic Bus, is a messaging system used to communicate between processes (Miller & Valasek, 2015, p.28). Under normal conditions, the D-Bus service should not be subject to user input or manipulation, as it is an intended for internal systems messages only. Miller and Valasek then found that – prior to Sprint's fix

– any 3G device on the Sprint network could communicate with the open D-Bus port on any Uconnect-enabled vehicle (Miller & Valasek, 2015, p.46). For their attack, Miller and Valasek used a laptop computer tethered to a 3G cellular phone on the Sprint network. The laptop was then able to communicate directly with vehicles running the vulnerable Uconnect system (Miller & Valasek, 2015, p.46). Knowing the IP address of a specific vehicle allowed for a targeted attack; however, the pair also found that an Internet port scan of port 6667 across IP ranges 21.0.0.0/8 and 25.0.0.0/8 would yield responses from vulnerable Uconnect systems in vehicles nationwide (Miller & Valasek, 2015, p.46). The researchers' scans of Internet-facing vulnerable devices turned up a wide range of vehicles across the country from the Dodge, Ram, Jeep, and Chrysler brands, spanning multiple model years (Miller & Valasek, 2015, p.47).

With access to the vehicle's Uconnect system obtained, Miller and Valasek then pivoted to the CAN-connected microcontroller in the Uconnect head unit. They were able to flash the controller with a new firmware version – one that they had reverse-engineered to include their malicious code (Miller & Valasek, 2015, p.50). With their modified firmware residing on the CAN bus, they were then able to send commands to many different vehicle components and control systems. During a press demonstration, Miller and Valasek showed that they were able to remotely set the air conditioning to its maximum cold setting, turn the radio on at full volume, and cover the windshield with wiper fluid making it difficult for the driver to see. More worryingly, they could also disable the transmission, control the throttle, and disable the brakes (Greenberg, 2015a).

This latest automotive hacking demonstration propelled vehicle security into the general public's consciousness in a way that had not been seen previously. Shortly after news of the Jeep Cherokee hack hit the media, a Kelley Blue Book study of the car-buying public found that 72% of respondents were "aware of the recent Jeep Cherokee hacking incident" (as cited in PR Newswire, 2015). And, perhaps more tellingly, 41% of respondents said they would "consider this recent vehicle hacking incident when buying/leasing their next car" (as cited in PR Newswire, 2015).

For the first time, an automotive hack had the very real potential to cost a large automaker a significant amount of money. Fiat Chrysler Automobiles, facing a reputation

hit and possible loss of future customers, made the wise – but nonetheless costly – decision to patch any vehicles that were vulnerable to Miller and Valasek's exploit. By some estimates, the amount which this critical security update cost FCA in labor hours alone was in excess of $10 million (Cobb, 2015). Miller and Valasek have long stated that their shared goal has been to provide their research to the auto industry and security community "so that we can learn to build more secure vehicles in the future, so that drivers can trust they are safe from a cyber attack" (Miller & Valasek, 2015, p.88). Certainly, hitting an automaker's bottom line is an effective way to accomplish this greater good.

5. Securing the Automobile

The extensive media attention given to Charlie Miller and Chris Valasek's Jeep Cherokee hack has sent automakers scrambling to better secure their vehicles. The old industry status quo of "security by obscurity" is no longer acceptable. Researchers have published detailed accounts of how to gain access to the CAN bus and manipulate CAN messages. As such, the security of vehicle systems is no longer a topic that can simply be ignored by manufacturers. While it is to be expected that each automaker will approach the problem of vehicle systems security in its own way, outlined below are some fundamental security best practices that should be followed going forward.

5.1. Encryption

One of the most fundamental flaws of the CAN protocol is a lack of message confidentiality. As in the "security by obscurity" model, automakers rely solely on a proprietary message format that is unknown to the public as a means of security. But as has been shown, it is possible to decipher these proprietary CAN messages to uncover their function, and to then modify or replay CAN messages for malicious purposes. The most effective way to prevent CAN reconnaissance is to apply some form of encryption to the CAN protocol.

A significant limitation facing CAN encryption is the CAN protocol's maximum data field size of 8 bytes. According to Gene Carter of security firm Security Innovation, "CAN is an old technology with limited data streams, [therefore] it isn't possible to use

encryption of any meaningful size" (as cited in Yoshida, 2015). It is widely accepted that a strong encryption algorithm requires a 128-bit or 256-bit block size. But that hasn't stopped several different researchers and security companies from proposing solutions for CAN encryption.

One promising encryption solution for CAN messages is SecureCAN from Trillium, a small Japanese start-up (Yoshida, 2015). SecureCAN is designed for payloads of 8 bytes or less, such as those found on the CAN bus, and supports variable block size and key length (Yoshida, 2015). The Trillium cipher found in SecureCAN utilizes three different algorithms: a message first undergoes substitution; the resulting ciphertext then passes through a transposition algorithm; finally, time-multiplexing is applied before the ciphertext is transmitted (Yoshida, 2015). Trillium claims the full process of encryption, transmission, and decryption can be performed in less than one millisecond, which falls within the time threshold required for real-time automotive CAN bus applications (Yoshida, 2015). SecureCAN also employs a novel key management system known as "Dynamic Key-Lock Pairing" (Yoshida, 2015). With this solution, a new shared master key is generated each time a car's ignition is turned on. Additionally, SecureCAN can change the ciphertext at random intervals, potentially multiple times per second, using frequency channel hopping (Yoshida, 2015). The SecureCAN encryption solution makes it nearly impossible for an attacker to intercept or modify CAN messages. A visualization of the SecureCAN algorithm is provided below in *Figure 9*.

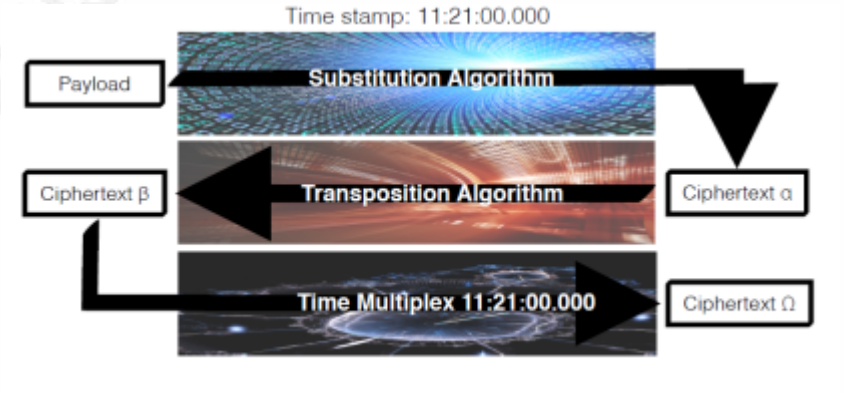

***Figure 9*: Trillium Cipher Algorithm (as cited in Yoshida, 2015)**

5.2. Device Authorization

Another key element in preventing an attacker from being able to transmit malicious messages on the CAN bus is to require authentication or authorization of devices. Previous attack demonstrations have utilized devices – generally laptop computers – which should have no legitimate reason for transmitting data on the CAN bus. To prevent unauthorized computers or rogue CAN controllers from broadcasting CAN messages, the receiving CAN controller needs to be able to validate that the message comes from an authentic source.

Conceptually, CAN device authorization can be accomplished by preprogramming CAN controllers with a whitelist of CAN identifiers for known good devices. For example, the vehicle's steering controller should know to only trust commands coming from the controller associated with the vehicle's steering wheel – and not from some external source. Of course, this opens the door to CAN identifier spoofing, whereby an attacker spoofs messages to make them appear as if they are coming from a legitimate source. And so, for device authorization to work effectively, the CAN identifier field *must* be encrypted. Encryption of the CAN data field prevents an attacker from being able to decipher the function of a message. Now, with the addition of encryption of the CAN identifier field, an attacker is also unable to spoof an authorized CAN device.

Published in 2011, a patent held by Patrick K. Richards entitled "Secure Communications Between and Verification of Authorized CAN Devices" offers a solution for CAN device authorization (Richards, 2011). This solution makes use of "a unique encryption code stored in each of the authorized CAN bus devices, [so that] unauthorized CAN bus nodes will not be able to communicate with the authorized nodes" (Richards, 2011). Unlike Trillium's SecureCAN encryption solution that encrypts the data field, Richards' solution requires encryption of the identifier field. This is problematic because any modification of a data frame's CAN identifier field will result in the recipient CAN controller ignoring the message, as it no longer recognizes the source. Therefore, encryption of the CAN identifier requires the use of a hardware-based encryption solution placed between the sending and receiving CAN controllers. Richards'

solution calls for the use of a pair of KEELOQ peripheral devices to act as encryption and decryption devices between the CAN endpoints (Richards, 2011). KEELOQ is a proprietary hardware-based block cipher that is owned by Microchip Technology Incorporated (Richards, 2011). There are numerous potential downsides to this solution, however, as it would add additional processing time to CAN transmissions, additional expense for automakers, and additional weight to the vehicle. Indeed, the implementation of any security solution will always come with some trade-offs.

5.3. Defense in Depth

There is no single solution to the security vulnerabilities facing automotive systems. What is needed is a broad approach providing multiple layers of security, also known as "defense in depth" (McGuiness, 2001). The concept of defense in depth utilizes "a series of defensive mechanisms such that if one mechanism fails, another will already be in place to thwart an attack" (McGuiness, 2001).

In the Information Security field, it is a commonly accepted reality that perimeter defenses will fail at some point. Boundary defense alone is not enough to secure an information system. It must be assumed that an attacker is capable of bypassing perimeter defenses and will eventually find himself within the internal network. When that happens, there must be a sufficient arsenal of other security controls in place to keep him from maneuvering within the system or otherwise being successful in performing an attack.

A holistic approach to securing a vehicle's systems should include – at the very least – better network segmentation, locking down of external interfaces, controller authentication, and data encryption.

The diagram shown below in *Figure 10* offers a conceptual model for applying a defense-in-depth approach to securing CAN communications. The diagram depicts the flow of data through multiple layers of security, as a CAN controller prepares a CAN packet for transmission onto the CAN bus.

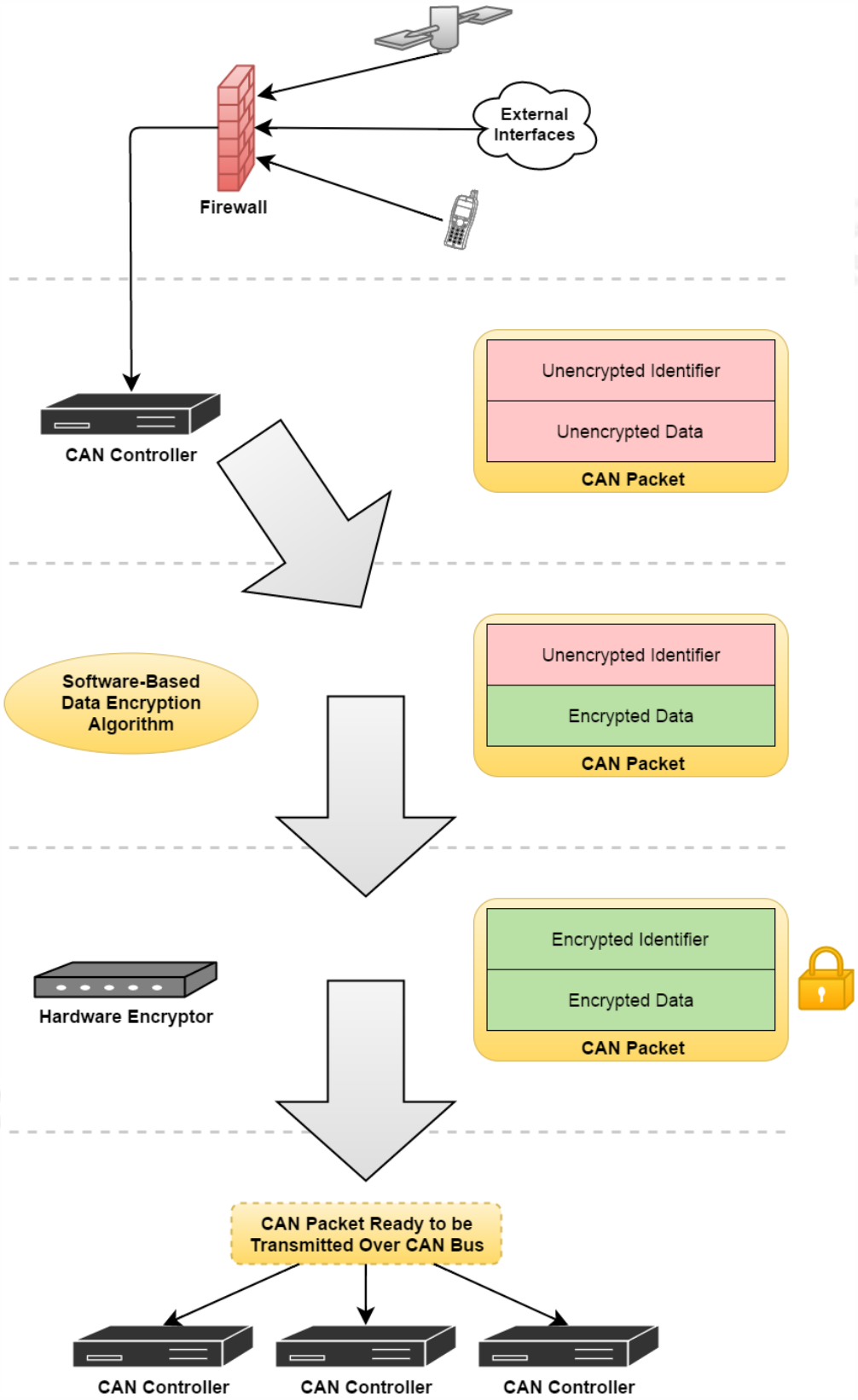

***Figure 10*: Defense-in-Depth Approach to Secure CAN Communication (Original)**

The preceding model features several layers of security so that CAN data would still be protected if an attacker were somehow able to subvert one of the security controls. At the top of the diagram, it can be seen that the CAN controller for a particular vehicle component is protected from external interfaces (such as the Internet, cellular network, or data ports) by a firewall. The firewall should be configured to allow only legitimate, known good traffic through to the vehicle's internal network.

Beyond the firewall, the diagram shows how a CAN packet created by the CAN controller is initially in an unencrypted, insecure form. The next layer of security to be applied is a software-based encryption algorithm, such as the SecureCAN algorithm from Trillium (Yoshida, 2015). This software-based encryption would be performed within the CAN controller itself, and serves to encrypt the data portion of the packet. The packet is then passed through a hardware encryption solution, such as KEELOQ (Richards, 2011). The hardware encryptor encrypts the CAN identifier, thereby protecting against device spoofing. Once both forms of encryption have been applied, the secure CAN packet is ready to be broadcast on the CAN bus. Any receiving controllers for which the packet is destined would then be required to perform hardware and software decryption before processing the packet. Through this defense-in-depth approach, the CAN bus is protected against even the most determined attacker.

5.4. Security by Design

A major obstacle to the development of secure automobiles is the archaic CAN bus technology that lies at the core of almost every modern car. According to Kathleen Fisher, a professor at Tufts University, "the CAN bus is hopelessly insecure [because] it was developed decades before cars were connected to the Internet and lacks features to block malware programs or reject commands from unauthorized intruders" (as cited in Bray, 2015). Because of the significant limitations of CAN, automakers will be forced to implement "Band-Aid" fixes for CAN until a fundamental overhaul of vehicle networking architecture occurs.

Ideally, security should be designed into vehicle systems from the ground up. Security should never be an afterthought, nor should security features be applied reactively. As automakers explore the prospect of replacing CAN with a more

fundamentally secure infrastructure, Ethernet has shown promise as one possible solution (Yoshida, 2013). However, the automobile industry is known to move slowly and resist change, so just how soon the Ethernet backbone will become commonplace in new vehicles remains a point of debate (Yoshida, 2013).

Because of the years of development time and millions of dollars of associated costs involved, car companies are not likely to take the plunge on a whole new automotive system architecture unless their competitors do the same (Fisher as cited in Bray, 2015). Meanwhile, automakers may have their hands forced by legislation that has recently been introduced to set data security and privacy standards for all new vehicles sold in the United States (Bray, 2015). Until such a measure becomes law, new vehicles will remain inherently vulnerable.

6. Conclusion

The automobile touches the lives of many. And, with the last few years bringing so many significant developments in the area of automotive systems security, automakers are finally being held accountable by the public. People are becoming aware of the serious real-world implications of security vulnerabilities in vehicles, and this may be the catalyst needed to bring about change. The problem is deep-rooted and complex; the solution is neither straightforward nor cheap. But the time has come for automakers to stop simply patching security flaws, and start building secure systems from the ground up. In the words of the SANS Institute's Ed Skoudis, "every new area has to be done badly first and then we have to clean it up" (as cited in Yadron, 2015).

References

Baltieri, F. (2013). *Hacking into a Vehicle CAN bus.* Retrieved November 13, 2015, from http://fabiobaltieri.com/2013/07/23/hacking-into-a-vehicle-can-bus-toyothack-and-socketcan/

Bourne, J. (2015). *Report warns of more malicious security fears in connected cars*. Retrieved November 26, 2015, from http://www.connectedcar-news.com/news/2015/jun/01/report-warns-more-malicious-security-fears-connected-cars/

Bray, H. (2015). *After car hack, Internet of Things looks riskier*. Retrieved November 16, 2015, from http://www.betaboston.com/news/2015/08/03/after-car-hack-internet-of-things-looks-riskier/

CANdo. (2015). *CANdo CAN Bus Analyser*. Retrieved November 28, 2015, from http://www.cananalyser.co.uk/index.html

Center for Internet Security. (2015). *The CIS Critical Security Controls for Effective Cyber Defense, Version 6.0*. Retrieved November 13, 2015, from http://www.cisecurity.org/critical-controls.cfm

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S. et al. (2011). *Comprehensive Experimental Analyses of Automotive Attack Surfaces.* Retrieved November 15, 2015, from http://www.autosec.org/pubs/cars-usenixsec2011.pdf

Cimpanu, C. (2015). *Volkswagen Sued Researchers for 2 Years to Prevent Them from Publishing a Security Flaw.* Retrieved November 11, 2015, from http://news.softpedia.com/news/volkswagen-sued-researchers-for-2-years-to-prevent-them-from-publishing-a-security-flaw-489347.shtml

Cobb, S. (2015). *Cybersecurity and manufacturers: what the costly Chrysler Jeep hack reveals.* Retrieved November 15, 2015, from http://www.welivesecurity.com/2015/07/29/cybersecurity-manufacturing-chrysler-jeep-hack/

Computer History Museum. (2011). *Diagram of electronic components in a car.* Retrieved November 12, 2015, from http://www.computerhistory.org/revolution/real-time-computing/6/134/548

Doughty-White, P., & Quick, M. (2015). *Codebases: Millions of Lines of Code.* Retrieved November 11, 2015, from http://www.informationisbeautiful.net/

visualizations/million-lines-of-code/

Eyal, N. (2007). *Vehicle Lab – Engine Control Unit*. Retrieved November 26, 2015, from http://www.vehicle-lab.net/ecu.html

Fortin Electronic Systems. (2006). *What is CAN Bus?* Retrieved November 12, 2015, from http://canbuskit.com/what.php

Greenberg, A. (2013). *Hackers Reveal Nasty New Car Attacks--With Me Behind The Wheel.* Retrieved November 14, 2015, from http://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-new-car-attacks-with-me-behind-the-wheel-video/

Greenberg, A. (2015a). *Hackers Remotely Kill a Jeep on the Highway—With Me in It.* Retrieved November 15, 2015, from http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

Greenberg, A. (2015b). *5 Lessons from the Summer of Epic Car Hacks*. Retrieved November 16, 2015, from http://www.wired.com/2015/10/five-car-hacking-lessons-we-learned-this-summer/

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S. et al. (2010). *Experimental Security Analysis of a Modern Automobile.* Retrieved November 15, 2015, from http://www.autosec.org/pubs/cars-oakland2010.pdf

Lambert, F. (2015). *Tesla hired Chris Evans from Google's Project Zero to lead the company's security team*. Retrieved November 16, 2015, from http://electrek.co/2015/08/06/tesla-hired-chris-evans-from-googles-project-zero-to-lead-the-companys-security-team/

McGuiness, T. (2001). *Defense In Depth.* Retrieved November 13, 2015, from https://www.sans.org/reading-room/whitepapers/basics/defense-in-depth-525

Micheli, G., & Ernst, R. (2002). *Readings in Hardware/Software Co-Design* (p. 314). San Francisco: Morgan Kaufmann.

Miller, C., & Valasek, C. (2014). *Adventures in Automotive Networks and Control Units.* Retrieved November 11, 2015, from http://www.ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf

Miller, C., & Valasek, C. (2015). *Remote Exploitation of an Unaltered Passenger Vehicle.* Retrieved November 15, 2015, from http://illmatics.com/Remote%20Car

%20Hacking.pdf

National Instruments. (2014). *Controller Area Network (CAN) Overview*. Retrieved November 27, 2015, from http://www.ni.com/white-paper/2732/en/

OpenXC. (2015). *Vehicle Interface Concepts.* Retrieved November 13, 2015, from http://openxcplatform.com/vehicle-interface/concepts.html

OWASP. (2009). *Avoid Security by Obscurity*. Retrieved November 16, 2015, from https://www.owasp.org/index.php/Avoid_security_by_obscurity

Payteck. (2003). *How PayTeck Works*. Retrieved November 29, 2015, from http://www.payteck.cc/aboutpayteck.html

Poulsen, K. (2010). *Hacker Disables More than 100 Cars Remotely.* Retrieved November 14, 2015, from http://www.wired.com/2010/03/hacker-bricks-cars/

PR Newswire. (2015). *Nearly 80 Percent Of Consumers Think Vehicle Hacking Will Be Frequent Problem In Near Future, According To New Kelley Blue Book Survey.* Retrieved November 14, 2015, from http://www.prnewswire.com/news-releases/nearly-80-percent-of-consumers-think-vehicle-hacking-will-be-frequent-problem-in-near-future-according-to-new-kelley-blue-book-survey-300121740.html

Richards, P. (2011). *Secure Communications Between and Verification of Authorized CAN Devices*. Retrieved November 30, 2015, from http://www.google.com/patents/US20110093639

Sewell Direct. (2015). *A Brief Explanation of CAN Bus.* Retrieved November 12, 2015, from https://sewelldirect.com/learning-center/canbus-technology

Tindell, K., Burns, A., & Wellings, A. (1995). *Calculating Controller Area Network (CAN) Message Response Times*. Retrieved November 25, 2015, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8665&rep=rep1&type=pdf

United States Department of Transportation. (2015). *Table 1-11: Number of U.S. Aircraft, Vehicles, Vessels, and Other Conveyances.* Retrieved November 11, 2015, from http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_statistics/html/table_01_11.html

Wikipedia. (2014). *File:CAN-Bus-frame in base format without stuffbits.svg*. Retrieved November 27, 2015, from https://commons.wikimedia.org/wiki/File:CAN-Bus-frame_in_base_format_without_stuffbits.svg

Wojdyla, B. (2012). *How it Works: The Computer Inside Your Car.* Retrieved November 12, 2015, from http://www.popularmechanics.com/cars/how-to/a7386/how-it-works-the-computer-inside-your-car/

Yadron, D. (2015). *Hacking Cars to Take Focus at Black Hat Conference*. Retrieved November 16, 2015, from http://www.wsj.com/articles/hacking-cars-to-take-focus-at-black-hat-conference-1438723360

Yoshida, J. (2015). *CAN Bus Can Be Encrypted, Says Trillium.* Retrieved November 12, 2015, from http://www.eetimes.com/document.asp?doc_id=1328081

Yoshida, J. (2013). *Ethernet Backbone in Car: Hype or Reality?* Retrieved November 12, 2015, from http://www.eetimes.com/document.asp?doc_id=1319157