Exploring Controller Area Networks

IAN FOSTER AND KARL KOSCHER

Ian Foster recently completed his master's degree at the University of California, San Diego, where he worked on analyzing the security of aftermarket telematics dongles. In doing so he found that security of some of these devices was often an afterthought, if existent at all. idfoster@cs.ucsd.edu

Karl Koscher is a postdoctoral researcher at the University of California, San Diego, where he specializes in embedded systems security. He earned his PhD in 2014 from the University of Washington, working with Tadayoshi Kohno. As part of his dissertation work, he was one of the lead researchers to perform the first published, comprehensive, experimentally backed security analysis of a modern automobile. supersat@cs.ucsd.edu

The highly publicized attack by Miller and Valasek during the summer of 2015 once again drew attention to weaknesses in automobile security. All modern automobiles rely on a broadcast network called CAN, and interfaces into that network are actually required by law. In this article, we explain how the CAN bus works and how it can be exploited.

Background

The Controller Area Network (CAN) is a serial bus standard designed for reliable, real-time message delivery between distributed control systems. Originally intended for vehicle applications, the CAN bus standard has found its way into many types of control systems, such as those used in elevators, medical devices, and robots. As detailed below, the standard is commonly implemented as a shared, differentially signaled bus, and enables priority-based arbitration. Multiple bitrates are supported, up to one megabit per second.

In automotive contexts, CAN buses are now commonly used to connect the various computers (known in the industry as electronic control units, or ECUs) of a car together. These ECUs now control most aspects of modern automobiles, including the engine, transmission, brakes, airbags, lights, and locks. Additional systems, such as "infotainment" (e.g., radio/navigation systems) and telematics systems (e.g., OnStar), are often connected to these ECUs. Vehicles will often have multiple CAN buses connecting various subsets of ECUs together.

The Controller Area Network Standard

Bosch, a German manufacturer of automotive control systems, began work on the Controller Area Network standard in 1983. Intel and Mercedes-Benz became involved with the project shortly thereafter, and in 1986 a paper introducing the "Automotive Serial Controller Area Network" standard was presented at the annual International Congress of the Society of Automotive Engineers (SAE) [1]. Version 2 was released in 1991 and forms the basis of all modern CAN implementations. CAN was subsequently adopted as an ISO standard (11898) in 1993 [2].

The CAN standard is optimized for low latency, high throughput, and reliable transmission. Low latency is achieved through short frame sizes (with a maximum payload length of eight bytes) and a priority-based, carrier sense multiple access (CSMA) arbitration scheme. While the maximum bitrate of 1 Mbps may seem low by today's standards, it meets the needs of most control systems. A new, backwards-compatible extension called CAN FD supports higher data rates. Reliability is ensured through multiple mechanisms. Differential signaling is commonly used at the physical layer, which provides immunity to common-mode noise (i.e., interference that couples onto one line will couple on to the other as well, canceling out its effect), as well as potential redundancy if one of the lines should fail. A 15-bit CRC (cyclic redundancy check) field at the end of each frame provides a high amount of certainty that frames are received correctly and are uncorrupted. An ACK slot at the end of the frame allows the sender to ensure that the frame was received correctly by at least one node, and many CAN controllers will automatically retransmit unacknowledged CAN frames. Figure 1 shows the CAN frame format.

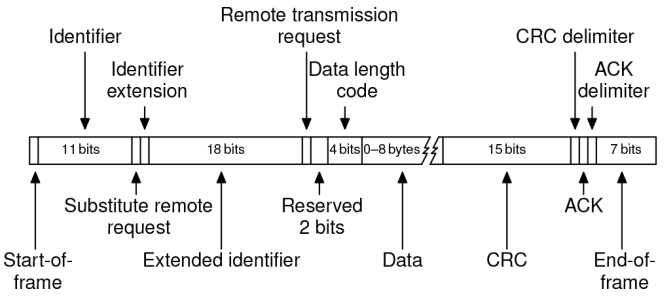

Figure 1: The CAN frame format

The CAN standard allows some flexibility in the physical layer (and in fact is not specified in Bosch's original standards), but relies on there being a "dominant" logical zero state and a "recessive" logical one state. Most CAN applications implement the physical layer described in ISO 11898-2, which specifies two lines, CAN_H and CAN_L, which are connected to each other at both ends of the bus with a 120Ω terminating resistor. In the recessive state, CAN_H and CAN_L are at approximately the same voltage (nominally 2.5v to ground). In the dominant state, CAN_H is pulled to 5v, while CAN_L is pulled to ground.

CAN frames primarily consist of an 11-bit or 29-bit arbitration ID, a four-bit length field, up to eight bytes of payload, a 15-bit CRC, and an acknowledgment bit. The arbitration ID typically is treated as a message type, but is sometimes (such as with OBD-II diagnostics, described below) used as a controller source or destination identifier. When transmitting the arbitration ID, the CAN transceiver monitors the bus. If it sends a recessive bit but detects a dominant bus condition, it aborts the message transmission. Note that the frame from the node that asserted the dominant bus condition has not been corrupted and thus can continue to be sent. Since a dominant state indicates a logical zero, and data is transmitted most-significant-bit first, lower arbitration IDs take precedence over higher arbitration IDs.

At this point, the astute reader may notice some security issues with the CAN protocol as described. In particular, CAN buses are broadcast networks, typically don't provide a way to identify the sender or recipient of a message, and are subject to trivial denial-of-service attacks. Each node on a CAN bus can observe all traffic. In fact, aspects of the CAN protocol, such as arbitration, *require* this. Furthermore, each node can send arbitrary CAN frames, without other nodes being able to verify the sender. Source or destination IDs, if used, can be trivially spoofed. Constantly asserting a dominant bus state will cause all other nodes to back off indefinitely, although well-designed CAN transceivers will detect this and enter a receive-only mode, making this type of denial-of-service attack difficult to pull off in software alone.

Given that the CAN standard provides no protection against malicious behavior, an attacker with access to a CAN bus is often able to take control of many critical aspects of the attached control systems. In the case of modern automobiles, there are many potential entry points into a vehicle's CAN bus(es), and these buses expose almost complete control over every aspect of the car's operation.

CAN Buses in Vehicles

Since CAN was invented with automotive applications in mind, we should step back and explain why vehicle ECUs may want to communicate with each other. Early engine control systems were introduced to meet stringent new emissions limits. In particular, by monitoring multiple sensors, the air/fuel ratio could be tightly controlled to minimize emissions. Since then, ECUs have evolved and proliferated to support ever-increasing fuel efficiency, emissions, safety, and reliability standards, and there can be further synergies with cross-ECU communications. For example, as Bosch explained in their original CAN bus paper, a transmission control unit can request the engine control unit to reduce torque, which reduces wear on the clutch and provides smoother shifting [1]. Similarly, the airbag controller can ask the engine controller to shut off the fuel pump if the airbags deploy, minimizing the chance of a fuel leak after an accident. Faced with a growing amount of inter-ECU communication, moving to a shared communications bus reduced the number of expensive (and heavy) point-to-point links.

Today, most aspects of a vehicle's operation go over one or more CAN buses. For example, in one vehicle we looked at [3], the anti-lock braking/stability control system reports the vehicle's speed to other modules, such as the speedometer, as well as to the engine controller (as input to its cruise-control algorithms). The radio also receives these speed messages to dynamically adjust its volume. In fact, the familiar click-clacks of the turn signal relays are now simulated by the radio, which receives the turn signal status from the body controller. The telematics system routes its audio through the radio, and can command the HVAC system to turn down the fans when a call is received.

These are just a few examples of inter-ECU communication in today's modern vehicles. In fact, the amount of information being transferred has grown to a point where vehicles often have *multiple* CAN buses, with gateway nodes that route selected messages between these buses. The architecture of how ECUs are connected together varies a great deal by manufacturer, but in 2014 Miller and Valasek published a survey of CAN bus architectures across a wide range of OEMs [4].

On-Board Diagnostics

In 1996, the OBD-II (On-Board Diagnostics) connector became federally mandated by the US Government. The OBD-II connector provides a way to verify the status of emissions control systems and facilities emissions testing. For example, emissions control systems can indicate over OBD-II port whether any sensor faults have been detected, the overall confidence in sensor performance, whether any unapproved firmware modifications have been made (which may affect emissions), as well as current sensor readings, which can be validated against external testing equipment. At the time, while the physical connector was standardized, there were several OBD-II communication protocols used by different manufacturers. The widespread adoption of CAN for powertrain ECU communications led it to be the natural choice for a single OBD standard. Since 2008, all vehicles sold in the US are required to provide OBD functionality over CAN. Practically speaking, this means that one or more major CAN buses are typically exposed to the OBD-II port.

The legally-required implementation of OBD over CAN ("legislated OBD") is defined by ISO 15031 and ISO 15765 and provides a relatively limited set of services, such as reading certain powertrain parameters such as engine speed, retrieving and clearing trouble codes, retrieving historical parameters recorded when a trouble code was raised, and requesting sensor test data. Under these standards, diagnostic messages are directed to ECUs at fixed CAN IDs, with their responses coming back with other fixed CAN IDs. ISO 15765-2 defines a simple transport layer, known as ISO-TP, which can be used to assemble larger diagnostic messages across multiple CAN frames and ensures in-order delivery [5].

In addition to legislated OBD, many vehicles also support the newer Unified Diagnostic Services (UDS) standard, defined by ISO 14229-3, which builds on legislated OBD. UDS provides several additional services, such as the ability to read and write arbitrary memory locations in ECUs, reflash ECU firmware, and override ECU I/O. For sensitive operations, such as reflashing safety, theft, or emissions-critical ECUs, or performing potentially unsafe I/O overrides, an OEM-defined unlocking process must usually be performed with the UDS SecurityAccess service, which defines a challenge/response-type mechanism for authentication. However, these unlocking schemes are often not robust—some OEMs use small keys that can be brute-forced, while others use simple algorithms such as XORing the challenge with a known secret [3, 6].

OBD-II Example

In the following example we show how to request the vehicle's VIN from the engine control module using OBD over CAN. OBD query and response packets are sent over the CAN bus using ISO-TP standard [5]. In this example, all nibbles and bytes shown are part of the CAN frame's eight-byte data section except for the ID field.

ID	Type	LEN	SID	PID
#0x7E0	0	2	09	02

Figure 2: OBD-II VIN query frame—only the ID field and eight-byte data field of the CAN frame are shown.

The ID or priority of the example CAN frame shown in Figure 2 is 0x7E0. CAN identifiers for legislated OBD are defined by ISO 16765-4 [7], which specifies that ECUs can be physically addressed with IDs 0x7E0–0x7E7, with corresponding replies sent to 0x7E8–0x7EF. The frame type and length are both nibbles. The type is 0x0 to indicate a "single frame," which means that the entire OBD request can fit within a single CAN frame. The LEN field specifies how many more bytes follow in the request. SID is the service identifier, which in this case is 0x09, or the "Request Vehicle Data" service. The "Request Vehicle Data" service takes a parameter ID (PID). For this service, a PID of 0x02 corresponds to the VIN.

ID	Type	LEN	SID	PID	NO	Data
#0x7E8	1	014	49	02	01	VIN[00-02]

Figure 3: OBD-II initial VIN response frame

The response to the query frame is shown in Figure 3. The ID and PID code fields should be the same as the query frame. As with the request, if the response can fit within a single frame, the type is 0. However, in this case, the response is split across many frames, so a frame type of 1 is used to indicate the "start frame" of a multi-frame packet. The LEN field of a start frame indicates the total number of bytes in the response. In an OBD response, the SID field is equal to 0x40 plus the SID from the query. For service 0x09, NO is the number of data items (in this case 1 for the VIN). Data contains the first three bytes of the requested data.

ID	Status	BS	ST
#0x7E0	30	00	00

Figure 4: ISO 15765-2 OBD-II flow control frame sent to main ECU

In order to get the remaining 17 bytes, a flow control frame needs to be sent to the ECU informing it of the parameters for sending consecutive frames. Figure 4 shows a flow control frame that will instruct the ECU to send all of the remaining packets immediately. The ID is the same as the OBD query. A status of 0x30 requests the rest of the data to be sent now and a status of 0x31 requests the ECU to wait. BS is the block size, defining the number of frames to send before waiting for next flow control frame (0 means no further flow control frames are needed). ST is the separation time delay between frames in milliseconds.

ID	Type	Index	Data
#0x7E8	2	1	VIN[03-09]
#0x7E8	2	2	VIN[10-16]
#0x7E8	2	3	VIN[17-19]

Figure 5: Remaining OBD-II VIN response frames

Figure 5 shows the remaining data frames sent by the ECU after receiving the flow control frame. The ID is the same as the initial response frame. Type and Index are both nibbles. The type is 0x2 to indicate consecutive frames, and the index is a frame counter starting at 1. Data contains up to seven bytes of the response data per consecutive frame. In this example, VIN is represented as a 20-byte string that is divided up into an initial frame and three consecutive frames.

This example uses a well-known query to request the VIN from the main ECU. However, for every ECU on the bus there are many other packets that are not well-documented. The more interesting CAN frames that can affect things like the engine, brakes, locks, etc. are proprietary and are generally not shared by the manufacturer.

Most of what is publicly known about the non-standard CAN frames has been reverse-engineered. Each vehicle may have different CAN messages, and sometimes even different generations of the same vehicle will use different frames. For example, the CAN frame to unlock the trunk on one vehicle may activate the windshield wipers of another vehicle.

Exploiting Vehicular Controller Area Networks

We now turn our attention towards how these automotive CAN buses can be abused. An attacker may be able to get access to these CAN buses in a variety of ways. Since these buses are often exposed over the ODB-II port, aftermarket devices that plug into this port (such as dongles that track your driving for insurance purposes) are potential entry points. At WOOT '15 we demonstrated several attacks against a popular OBD-II dongle platform that gives an attacker complete access to at least one CAN bus [8]. These dongles connect cars to the cellular network and can be exploited via SMS or their built-in Web server. Prior work by researchers at UC San Diego and the University of Washington, as well as Miller and Valasek, have also demonstrated *multiple remotely exploitable vulnerabilities in unmodified vehicles,* which can also be used to gain complete access to CAN buses [9, 10]. With vehicles becoming increasingly connected to the outside world, the number of potentially vulnerable entry points to these vehicles' CAN buses is rapidly growing.

With access to the CAN buses, an attacker can either use standard inter-ECU messages to influence vehicle behavior or may be able to exploit diagnostic services. For example, Miller and Valasek demonstrated partial control of the steering wheel by spoofing parking-assist and lane-keep-assist messages. These messages are relatively easy to discover—since the CAN bus is a broadcast network, we can simply monitor the messages sent while eliciting a behavior we want to reproduce. These messages can be captured using a CAN frame logger connected to the ODB-II port, and we can verify that we've found the correct message by replaying it and seeing if it produces the desired effect.

Given the relatively fragile nature of CAN, an attacker can override messages as well. For example, the UW/UCSD researchers were able to falsify speedometer readings—and in fact, invert them such that the displayed speed was 100 MPH *minus* the actual speed—simply by flooding the bus with spoofed messages [3]. A slightly more sophisticated attack could detect speedometer messages sent by other ECUs and assert a dominant bus state during the CRC, causing all other receivers to reject the message as invalid, although this requires fairly precise timing.

Some "functionality" is not exposed by standard inter-ECU messages. For example, there is no message that will let another ECU completely disable the brakes, and especially not for an extended period of time. In these instances, diagnostic services can often be abused to achieve the desired effect.

One powerful diagnostic service is the ability to override device I/O. While the exact payload of these message varies by OEM and ECU, the UW/UCSD team found it extremely easy to enumerate virtually every possible behavior by just sending random payloads. Combined with elevated privileges obtained by exploiting weak SecurityAccess schemes, an attacker can potentially perform dangerous operations, such as taking direct control of the brakes while the vehicle is moving at high speed.

Another useful diagnostic service is ReadMemoryByAddress, which can enable an attacker to read arbitrary pieces of an ECU's address space. This service can often be used to dump an ECU's firmware for reverse-engineering or to leak sensitive values such as authentication keys. While suppliers are cautioned to prevent leaking sensitive data over this service, many do not heed this warning. Others may not implement ReadMemoryByAddress restrictions correctly. For example, an ECU may prevent you from reading out sensitive values from flash, but does not prevent you from reading the same values out when they are copied to RAM.

Finally, the RequestDownload/TransferData services can be used to reflash ECUs, which allows an attacker to implement arbitrary behavior. These services should normally be restricted, but in many cases they aren't, and in other cases the SecurityAccess mechanism protecting access can often be defeated.

Summary

Modern automobiles have dozens of control units that communicate with each other via CAN buses. CAN buses are a shared broadcast medium, and while they are designed for reliability, they aren't designed to withstand malicious attacks. Many critical aspects of a vehicle's operation can be controlled with access to these buses, either by spoofing ordinary inter-ECU messages or by abusing diagnostic services. These CAN buses are becoming increasingly vulnerable to attack. Aftermarket devices plugged into the ODB-II port are in a position of privileged access and may be vulnerable to wireless attacks. Furthermore, vehicles themselves are now incorporating wireless connectivity (e.g., Bluetooth, WiFi, and cellular) in their infotainment and telematics systems, further broadening the potential attack surface. However, with recent media attention on these types of vulnerabilities, we are hopeful that automakers and aftermarket device manufacturers will devote more resources to securing their products.

References

[1] U. Kiencke, S. Dais, and M. Litschel, "Automotive Serial Controller Area Network," SAE Technical Paper 860391, 1986: doi:10.4271/860391.

[2] International Organization for Standardization, "Road Vehicles— Interchange of Digital Information—Controller Area Network (CAN) for High-Speed Communication," ISO/DIS Standard 11898:1993.

[3] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, "Experimental Security Analysis of a Modern Automobile," in the *Proceedings of the 31st IEEE Symposium on Security and Privacy*, May 16–19, 2010, Oakland, CA.

[4] C. Miller and C. Valasek, "A Survey of Remote Automotive Attack Surfaces," Black Hat USA 2014, August 2014, Las Vegas, NV.

[5] International Organization for Standardization, "Road Vehicles—Diagnostics on Controller Area Networks (CAN)—Part 2: Network Layer Services," ISO Standard 15765-2:2004.

[6] C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units," DEF CON 21, July 2013, Las Vegas, NV.

[7] International Organization for Standardization, "Road Vehicles—Diagnostics on Controller Area Networks (CAN)—Requirements for Emissions-Related Systems," ISO Standard 15765-4:2005(E).

[8] I. Foster, A. Prudhomme, K. Koscher, S. Savage, "Fast and Vulnerable: A Story of Telematic Failures," in the *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT '15)*, August 2015, Washington, DC.

[9] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in the *Proceedings of the 20th USENIX Security Symposium*, August 2011, San Francisco, CA.

[10] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Black Hat USA 2015, August 2015, Las Vegas, NV.

ENIGMA

It's time for the security community to take a step back and get a fresh perspective on threat assessment and attacks. This is why the USENIX Association is excited to announce the launch of Enigma, a new security conference geared towards those working in both industry and research.

Enigma will deliver three days of talks from leading practitioners and researchers, all of whom are uniquely qualified to discuss security as it relates to the Internet of Things, black markets, election issues, threats, scalability, and much more.

Featured speakers include:

Bryan Payne, Netflix:
"PKI at Scale Using Short-Lived Certificates"

Adrienne Porter Felt, Google:
"Why Is Usable Security Hard, and What Should We Do about It?"

Damon McCoy,
New York University:
"Bullet-Proof Credit Card Processing"

The full program and registration are now available.

enigma.usenix.org

JANUARY 25–27, 2016

SAN FRANCISCO, CALIFORNIA, USA

usenix ASSOCIATION