TrendLabs

A Vulnerability in Modern Automotive Standards and How We Exploited It

Technical Brief

TrendLabs Security Intelligence Blog

Federico Maggi

Forward-Looking Threat Research (FTR) Team

August 2017

This research is a joint effort between Politecnico di Milano, Linklayer Labs, and Trend Micro's FTR. In this report, we describe a vulnerability in modern cars' networks that allows a completely stealthy denial-of-service attack which is undetectable by current security mechanisms and works for every automotive vendor. This attack differs drastically from other previously reported car hacks because it does not exploit easily patchable software vulnerabilities. Rather, the element exploited is a design flaw, which is thus fundamentally hard to solve, in the standard that defines how in-vehicle networks work.

This attack was presented at the 2017 international conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) in Bonn (Jul 6–7). Prior to that, we coordinated with the ICS-CERT, which promptly disseminated an alert (ICS-ALERT-17-209-01).

Introduction

When security experts hear about a new cybersecurity issue in the automotive domain, they tend to dismiss it quickly because car hacks often require either some form of local access or a remotely exploitable vulnerability (which can be patched to solve the problem). The former is even considered less likely. In fact, remotely exploitable vulnerabilities found in in-vehicle systems (like those used by Charlie Miller and Chris Valasek in their well-known "Jeep hack") are much more interesting for an attacker, and much more severe for automotive vendors, compared to attacks that require local access. And, at the end of the day, once a remotely exploitable flaw is found and patched, the security issue is deemed resolved.

Current trends in transportation challenge such beliefs as it is now much more common to have anyone gain local access to someone's car. Consider car sharing, ride sharing, autonomous vehicles—these are becoming commonplace scenarios where many users can access the same vehicle. This change in paradigm calls for an appropriate change in the threat model, which must now encompass a local attacker.

Our discovery involves an actual working local attack which is not only completely undetectable and vendor neutral but also cannot be easily resolved by a simple remote patch and/or update. In addition, in our DIMVA paper, we discuss how our attack can be performed remotely. With this, both industries (security and automotive alike) must prepare to embrace a paradigm change and address this burgeoning security issue.

How Modern Cars Work from a CAN Standpoint

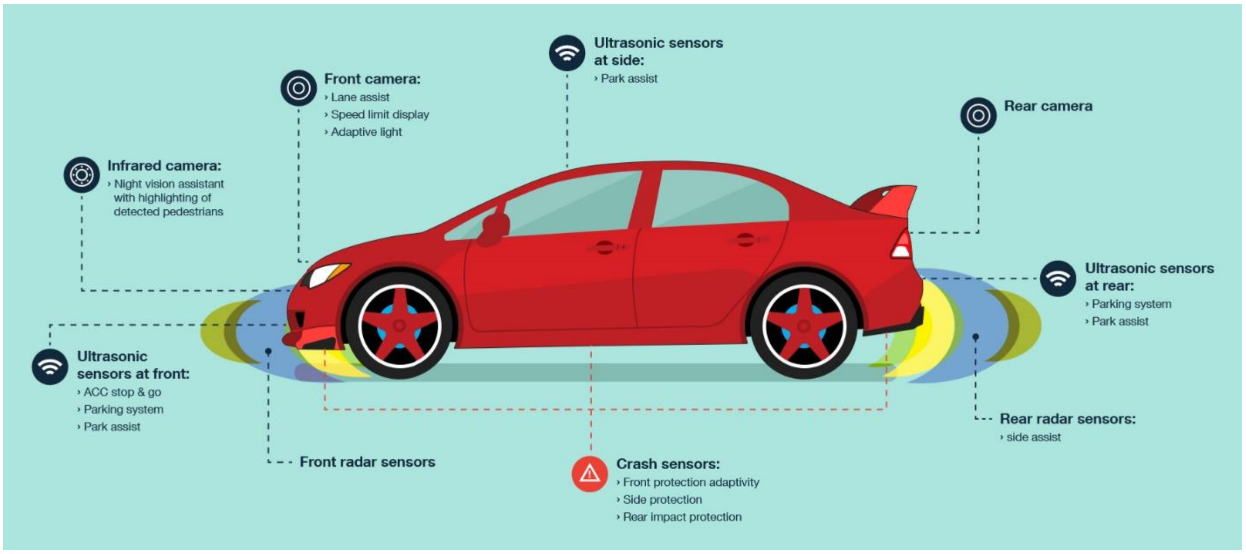
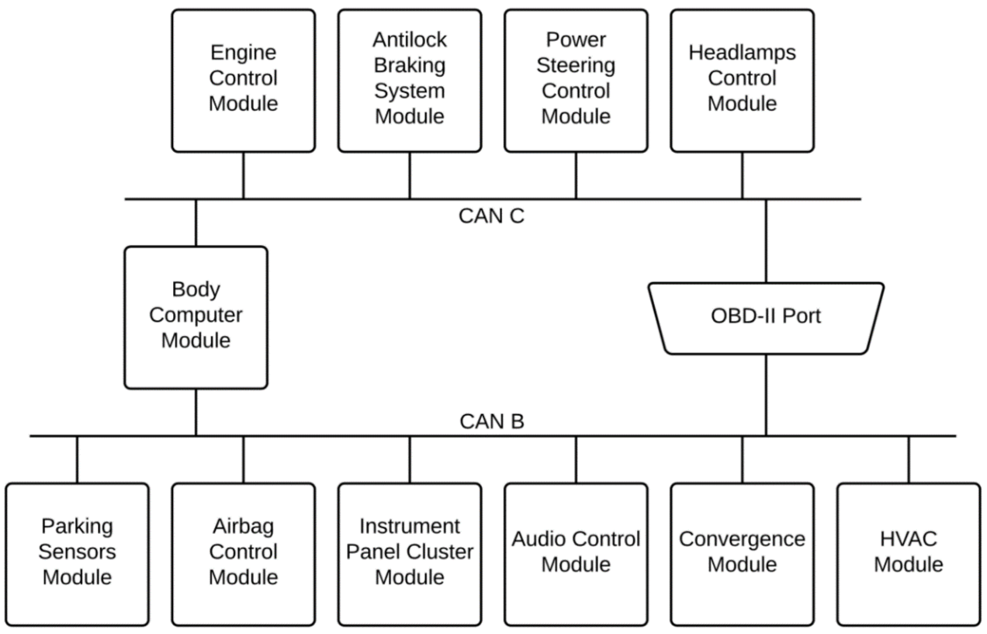


Figure 1: The technologies inside the modern car

In-vehicle equipment (e.g., parking sensors, airbag, active safety system, anti-braking system) and systems (e.g., infotainment) communicate with each other through what is called a Controller Area Network (CAN).

After initial development by Bosch in 1983, the CAN protocol was officially released in 1986 and was first featured in production vehicles in 1989. In 1993, the International Organization for Standardization (ISO) accepted CAN as a standard and published ISO 11898 for road vehicles. Since then, CAN has been used as a standard for practically every light-duty vehicle and was being pushed even further as the “only acceptable protocol used for standardized on-board to off-board communications for vehicles.” (US Federal Register, Vol. 70, No. 243, 86.005–17, Dec 20th, 2005). Among several standards, CAN is preferred because of its simplicity, extremely low cost, and minimal wiring requirements.

The CAN standard specifies how the various devices are interconnected. For example, it defines how your infotainment or safety system receives messages from your (car) body control module, or airbag, to know whether it must call home because you are in an accident. Similarly, your active safety system receives updates from front- or rear-facing cameras and radars, in order to know how brakes must be engaged.



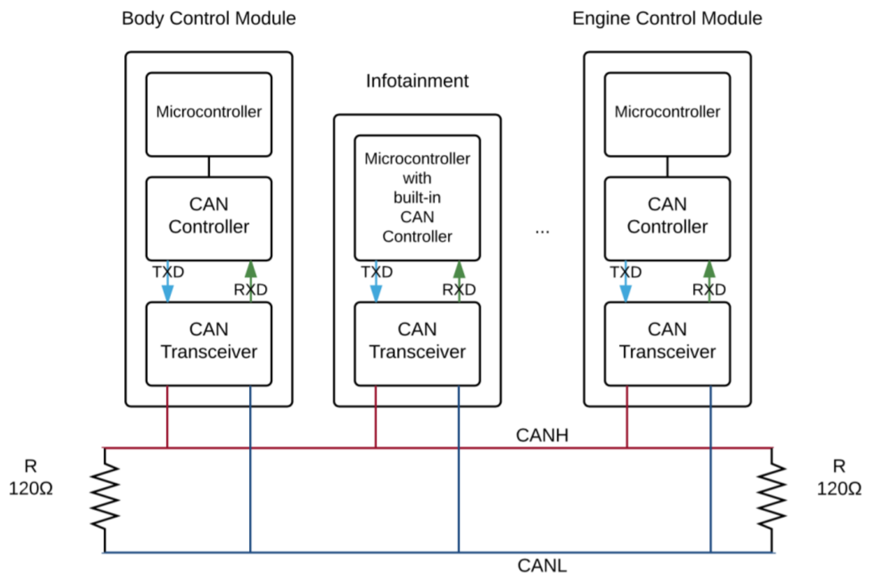


Figure 2: Typical CAN-based in-vehicle network (top) and how it's electrically connected to the CAN bus (bottom). The Microcontroller implements the "application" logic, the CAN Controller implements the CAN logic, and the CAN Transceiver is responsible for translating logical "zeros" and "ones" into electrical signals.

According to the CAN standard, there are two wires—called CAN High (CANH) and CAN Low (CANL)—onto which all devices are attached. This is also called the CAN “bus” because it appears like a "shared cable" running across all the various subsystems.

When one of these subsystems wants to “talk,” it writes a message (also called "frame")—properly encoded as a series of ones and zeros—onto the CAN bus.

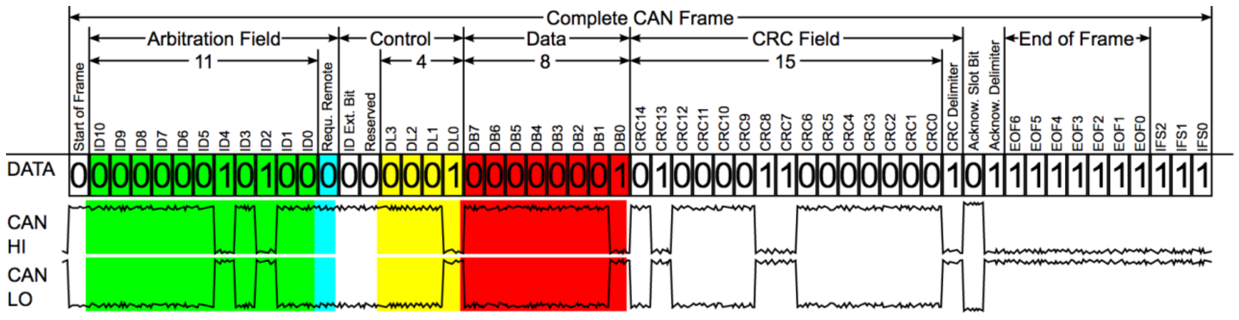


Figure 3: Structure of a CAN frame (source: Wikipedia - https://en.wikipedia.org/wiki/CAN_bus)

Any message circulating on the CAN bus can be read by any other device connected to the bus. If the message is relevant to the device (e.g., RPM reading from the engine-control module), it can take action (e.g., the infotainment turns the volume up, so music can be heard better).

Because bus errors are frequent, and because the bus is contended, any device writing a frame onto the CAN bus is also responsible for checking the actual value on the wires. If the value read at a certain time corresponds to the original expected value, everything proceeds. Otherwise, if there is a mismatch, the device must immediately write an error message onto the CAN bus in order to recall the previous frame and to notify the other devices listening that they should ignore it. Errors occur because of "natural" causes or because multiple devices are attempting to concurrently write on the CAN bus.

When a device starts sending too many error messages, the device itself may be malfunctioning. Because of the reliability and speed requirements, the CAN standards mandate that such a device must enter a so-called Bus-Off state. At this state, the device is isolated from the rest of the network and, as such, not allowed to read or write any data. This fault-containment mechanism is useful in preventing a malfunctioning (or rogue) device to detrimentally affect the car’s performance.

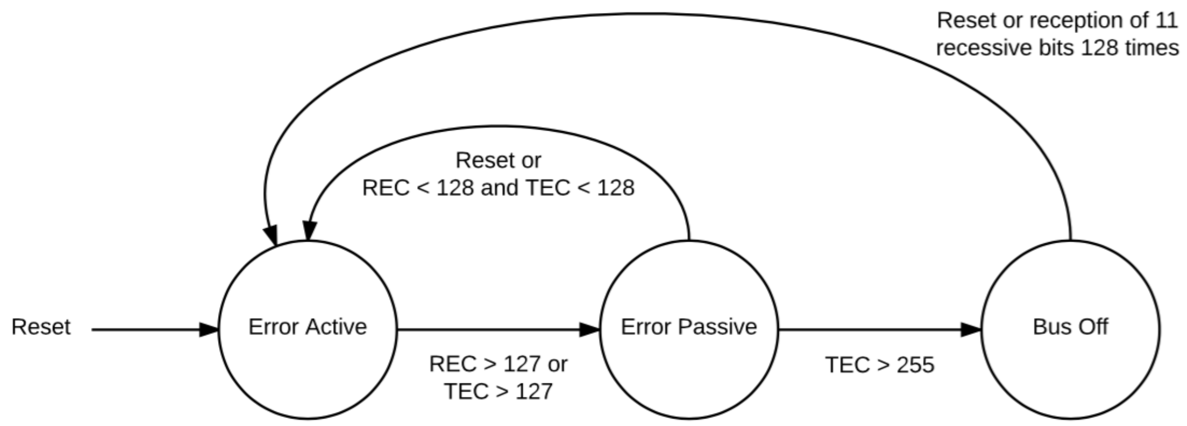


Figure 4: States of a CAN node (TEC = Transmit Error Count, REC = Receive Error Count). The "Error Active" state is the normal state; past producing 127 errors, the node enters the "Error Passive" state, which means that its communication capabilities are limited (for example, it can only read and produce errors at a limited rate). When in "Bus Off" state, no communication is allowed.

Errors in the System: The Design Vulnerability Within the CAN Standard

Given how in-vehicle networks work, what are the security issues? There are several.

• First, any device connected to the CAN bus is allowed to read and write without any regulation. There is no authentication or access control mechanism.

• Secondly, all data coming from the CAN bus is trusted because the security model assumes that an attacker will never gain unauthorized access to the CAN bus.

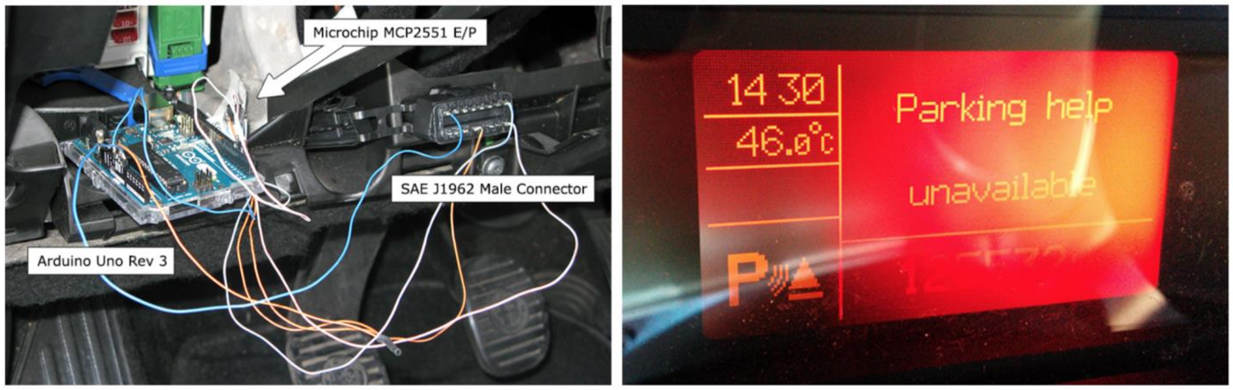
• Lastly, there is no way to distinguish a genuine error message from a crafted one. In other words, it's impossible to know whether a device (e.g., infotainment) is truly faulty or if it's been compromised and is now going "off bus" because of an attacker's command.

All together, these design issues have more serious consequences. Any device on the CAN bus can craft messages such that any other one would be cut off from any communication. This is like a selective denial-of-service (DoS) attack.

Other researchers have already demonstrated how the vulnerability in the CAN standard design can be exploited to selectively DoS-attack specific devices. Possible attacks against the CAN protocol have been investigated since 2010, especially frame-injection attacks.

All of the attacks demonstrated so far can be detected. This is because they need to generate anomalous CAN frames, which stand out from typical traffic. In fact, there are even vendors providing IDS and IPS technology for in-vehicle networks based on the principle of detecting "anomalous" CAN traffic. In other words, all of the known attacks are not stealthy, and thus can be spotted and blocked.

However, in our attack, we demonstrate that an adversary can cause damage to in-vehicle equipment in a very stealthy way, without the need to write any message on the CAN bus, and with a very modest investment of US$30 (for the attack device that we connect to the target car's CAN bus).



(a) The attacking device attached to the Giulietta's OBD-II port.

(b) The parking sensors malfunction reported on the driver's LCD.

Figure 5: Sample attack device and effect

Our attack device works this way: instead of injecting CAN frames like what previously demonstrated techniques did, we reuse existing frames that are already circulating on the CAN bus, and modify a single bit of them. More precisely, we "flip" one bit (from 1 to 0) in a specific position such that it would induce an error, thereby forcing the device that has transmitted the original frame to write an error message as required by CAN standards. The tricky part is to be able to read the right message at the right time and be fast enough to modify that single bit.

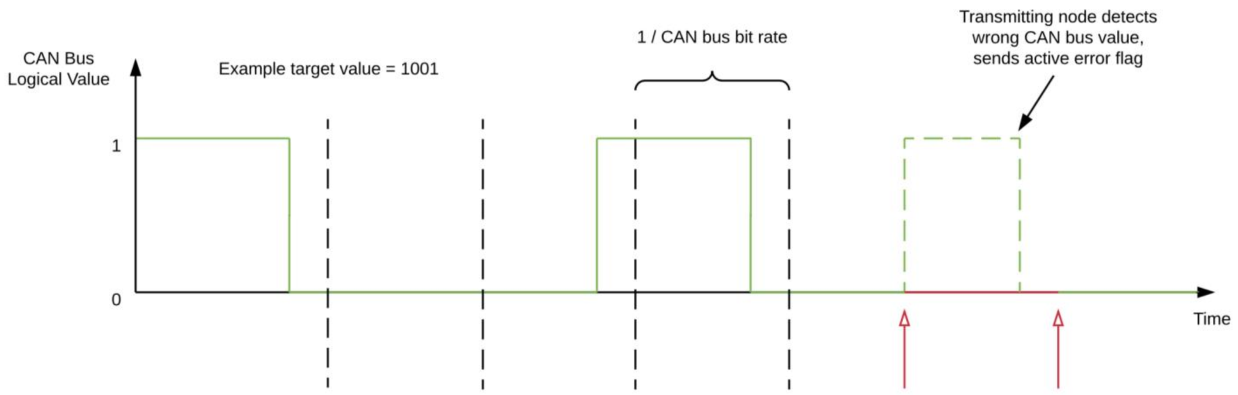


Figure 6: Attack device attack chain

We repeat this process 32 times in a row, thus triggering the CAN fault-containment mechanism. The device goes into the Bus Off state, effectively cutting it off from any communication.

The preconditions for our attack are the same as that of Miller & Valasek's Jeep hack. In the presence of a remotely exploitable vulnerability (e.g., in the infotainment system), which essentially brings the attacker to the local network, the Jeep attack would inject frames on the CAN bus to take control of the car (e.g., accelerate or brake suddenly). Fortunately, these frames would be easy to spot. For instance, it's not expected that the infotainment system will try to engage the brakes or the cruise control. Without remotely exploitable vulnerabilities, neither our attack nor the Jeep hack would be possible.

However, our attack brings up an important point. Even under the assumption that a state-of-the-art IDS/IPS is monitoring the CAN bus, our attack is undetectable. Moreover, we show that, given the criticality of a car system, an attacker doesn't necessarily need to take active control of the car's equipment to create life-endangering situations. Most of the time, all the attacker needs is to disable a critical subsystem (e.g., airbag, park-assisting sensors).

Effectively detecting and blocking our attack would require changes in the standard, major architectural changes in the network topology, and the redesign of in-vehicle networks. This attack is well beyond a simple software vulnerability, which can be solved by recalling vehicles for an upgrade or via over-the-air (OTA) upgrade.

Affected Sectors: Automotive is Just the Beginning

The automotive sector is the main sector affected by the results of our research. However, it's not the only one. The CAN standard is widely adopted in other transportation sectors such as in trains (e.g., linking door units, brake controllers, passenger-counting units), in maritime (e.g., controlled-by-wire ships), avionics (e.g., flight-state sensors, navigation systems), and aerospace (e.g., fuel systems, pumps), as well as in other sectors like in industry (e.g., packaging, semiconductor manufacturing), hospitals (e.g., lights, beds, X-Ray and other diagnostic machines), and building automation (e.g., elevators, doors).

While the cost model and the technical requirements in these sectors are different from those in the automotive industry, the results of our research should be taken into account by these communities.

Threat Scenarios

What kind of threats can arise if an attacker decides to mount this attack against a vehicle in the real world, even in the presence of state-of-the-art IDS/IPS solutions? In this section we discuss the various scenarios wherein our attack may be utilized, as well as the vectors from which it can be executed.

Scenario 1. Active Safety System DoS

Active safety systems are beneficial, but they could create a dangerous scenario if the driver becomes too complacent in their use. An adversary using our attack could induce specific "faults" in the CAN frames generated by the active safety systems and suddenly cause the said systems to shut down. This may lead to vehicles failing to stop autonomously as expected by drivers, a failure which may cause fatal accidents.

Scenario 2. Throttle DoS

CAN has been used to carry "throttle-by-wire" functionality. For instance, the 2010 Toyota Prius internal combustion engine throttle actuator is controlled by CAN frames sent from the power-management control unit to the engine-control module. An adversary may use our attack against such frames, preventing the driver from controlling throttle position and thus from moving the vehicle. Despite not causing a direct hazardous condition, a financially motivated attacker could leverage our stealthy DoS to mount a ransomware-like attack and later show the classic message on the infotainment display. The car owner's first reaction would typically be to bring the car to a repair shop, which would still cause financial loss.

Scenario 3. Door Lock DoS

Most modern premium cars' door locks are controlled by CAN frames, which are typically accessible via other devices (e.g., infotainment) as well as via the OBD-II diagnostic port. Isolating the frames responsible for locking the doors is much simpler and faster than reverse-engineering active safety equipment messages. A single press of the lock-unlock button on the driver's door corresponds exactly to one fixed set of frames issued to the door module actuators. Therefore, in a matter of minutes, an adversary can isolate the frames responsible for doors locking, prepare the exploit in order to DoS-attack the locking frames, and then leave the exploit active, preventing car doors from being locked again after being unlocked.

Threat Vectors

Remote

Previous researches described as "remote car attacks" actually revolved around a chain of remotely exploitable vulnerabilities. For example, the Jeep hack exploited vulnerabilities in the Harman Kardon Uconnect system which allowed the researchers to remotely re-flash the embedded Renesas V850ES/FJ3 microcontroller—responsible for the Uconnect CAN communications—with an ad-hoc firmware. Once this happens, the attacker essentially becomes as powerful as a local attacker. Such vulnerabilities are frequently discovered, reported, and patched.

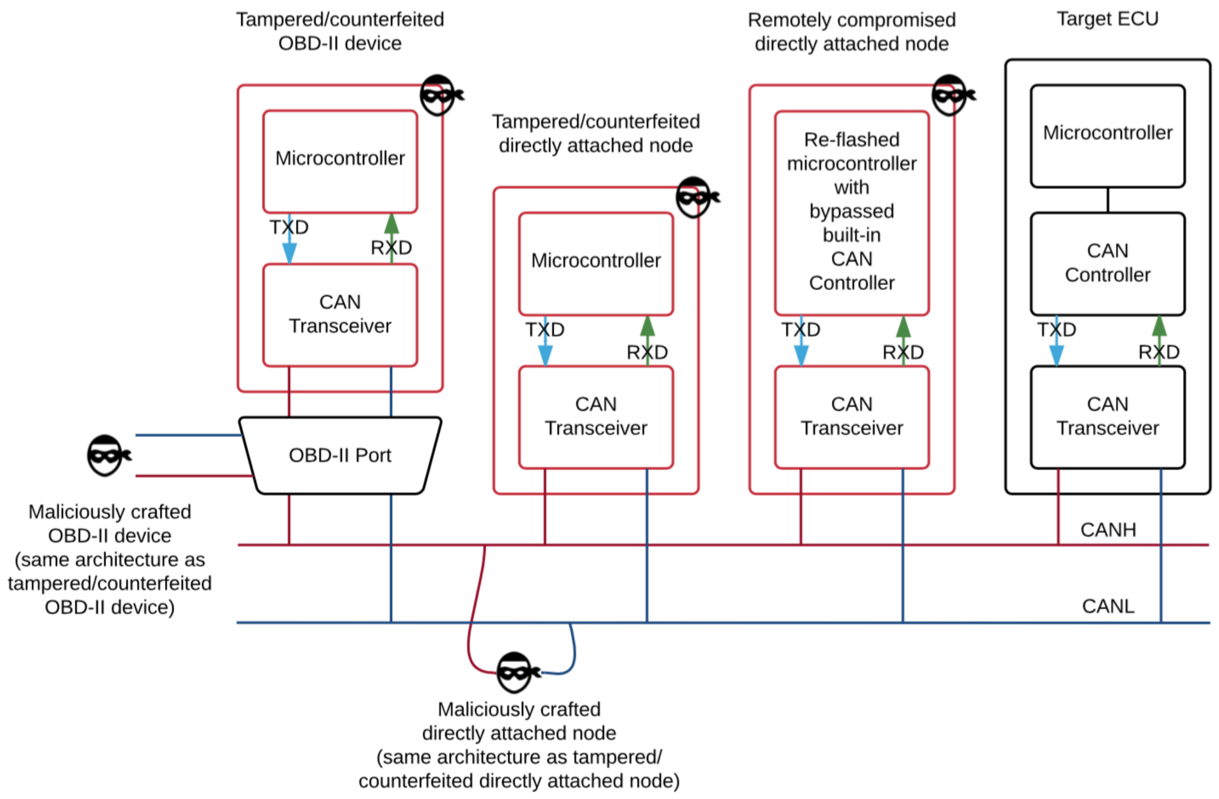

Tampered/counterfeited OBD-II device
Microcontroller
TXD
RXD
CAN Transceiver
OBD-II Port
Maliciously crafted OBD-II device (same architecture as tampered/counterfeited OBD-II device)
Tampered/counterfeited directly attached node
Microcontroller
TXD
RXD
CAN Transceiver
Remotely compromised directly attached node
Re-flashed microcontroller with bypassed built-in CAN Controller
TXD
RXD
CAN Transceiver
Target ECU
Microcontroller
CAN Controller
TXD
RXD
CAN Transceiver
CANH
CANL
Maliciously crafted directly attached node (same architecture as tampered/ counterfeited directly attached node)

Figure 7: CAN bus attack vector map

Therefore, all of the known attacks against cars require some form of local access, either direct or with the assistance of a remote exploit. Our attack is no different as it can be performed by re-flashing an ECU firmware or by local access to the CAN bus. The unique aspect of our attack is its “undetectability” and the fact that there is no easy fix, like there was for previous attacks.

Local

As with previous attacks, the easiest way by which our DoS attack can be mounted is via a crafted device attached to the OBD-II port. In most vehicles, the OBD-II port serves as a direct interface into all car internal buses, provides 12V direct current output for powering connected devices, and is conveniently located. Therefore, in a matter of seconds, an adversary with local access is able to install a working attack device inside a car. Real-world scenarios in which this may happen are numerous and include, for instance, valet parking, car sharing, car renting, car lending, or self-driving car settings.

The car owner may also unknowingly be using a rogue (trojanized/counterfeited) aftermarket OBD-II device after opting for a low-cost replacement part, looking for do-it-yourself car diagnostics, or simply enriching car infotainment functionality.

Note that local-access attacks are by no means limited to the OBD-II diagnostic port. An adversary may or must opt to attach and hide the attacking device anywhere along the CAN bus. For example, think of a malicious repair shop or the installation of rogue replacement parts that require CAN bus connections for their operation, like aftermarket infotainment units, parking sensors modules, or anti-theft systems.

Mitigation

The challenge is to detect a forthcoming attack before the DoS has been executed, as the attacking device will not participate in any way with the CAN activity but will remain completely silent to all other CAN devices. The following mechanisms could be implemented without any major change or rewiring of the in-vehicle network, at the same cost of recalling a vehicle for a firmware upgrade (like what happened after the Jeep hack was disclosed).

• **Power drain detection:** A system or program that notifies the user of possibly-malicious devices added to the CAN bus network through the detection of strange or added drain to the bus current, as any added device would add to the power load.

• **Error determination:** A system or program that can detect possibly-malicious devices through the nature of the error frames being sent. Malicious errors may present themselves as identical to each other. Note that such a detection approach may erroneously flag a truly faulty device as malicious.

Existing CAN bus IDS/IPS technologies are essentially based on the anomaly detection of malformed frames because, in the majority of attacks, an injection of frames is needed. Our attack, instead, is based on the transmission of bits concurrently with the transmission of a legitimate frame.

From the receiving devices' point of view, there would simply be a frame transmission interrupted by an error. A frame-analysis-based IDS could only notice the effect of our attack, which is a device going Bus Off. However, this would be ineffective in mitigating the attack itself since the detection would only take place after the fact.

We hope that the next generation of vehicles will take into account the following solutions to prevent our attack from happening.

• **Network Segmentation or Topology Alteration:** Create various CAN sub-buses, or change the network topology from a bus to a star-like shape, to prevent free circulation of can frames to all devices.

• **Regulated OBD-II Diagnostic Port Access:** Require a special hardware key in order to open the case where the port is physically located, or implement a software-level authentication in order to allow traffic from and to the port. This would require a change in the regulations.

• **Encryption:** Encrypt CAN frame ID fields to prevent attackers from identifying CAN frames to target, thus resulting in a noisier and much more detectable attack pattern.

Conclusion

Gaining access to someone else's vehicle has become a common situation, with many legitimate use cases. Along with the classic scenarios and vectors (e.g., car renting or lending, valet parking, repair shop), newer scenarios and vectors are emerging (e.g., car sharing, autonomous vehicles, pervasive ride-sharing service, availability of aftermarket infotainment or "plug 'n play" devices for car enthusiasts).

In addition, there is a sheer number of vulnerabilities found embedded in in-vehicle systems. While the existence of vulnerabilities has been considered exceptional so far, in the near future, this will be normal.

All of these conditions increase the probability of local attackers targeting a car. It is time that standardization bodies, decision makers, and car manufacturers take this shift into account and revise the design of the cyber-physical systems that govern future automobiles.

Resources

A complete list of references for this research is available in our paper:

• A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks, Andrea Palanca (Politecnico di Milano (Italy)); Eric Evenchick (Linklayer Labs); Federico Maggi (FTR, Trend Micro, Inc.); Stefano Zanero (Politecnico di Milano (Italy))

TREND MICRO™

Securing Your Journey to the Cloud

Created by:

TrendLabs

Global Technical Support & R&D Center of TREND MICRO