Titley, Joshua

Professor Martin

CS 4960

10-17-15

An Overview of
"In - Vehicle Networks: Attacks, Vulnerabilities, and Proposed Solutions"
by Carsten et al,
given by Joshua Titley

The goal of this paper is to explore some of vulnerabilities that exist in modern automobiles with respect to their on board computer systems and a few possible solutions to such vulnerabilities. As such the paper will begin with necessary background information regarding the sub-fields of computing involved. The next subject will involve a few of the potential vulnerabilities of these sub-fields when they are combined. Finally an examination of some potential partial solutions that may offer incite into practical and proper solutions. This paper will primarily concern itself with the CAN (Controller Area Network) protocol, although other protocols for automobiles exist.

Embedded systems comprise the vast majority of computers in existence today with more than 90% of all new processors destined for embedded systems. (Massa, 2007, p. xiii). Modern automobiles are no exception, they are saturated with embedded systems, as many as 50-70 such devices known as ECU's (Engine Control Units or Electronic Control Unit) can be present in a vehicle (Carsten et al 2015). According to Schulze et al, “An automotive system encompasses the hardware, ie., sensors, actuators, and electrical control units (ECU), and several bus systems used for their connection and communication amongst each other in a modern car.”. Another important factor with respect to the automotive computing system is that the ECU's are a very complex heterogeneous system without standardized data structures or concurrency controls (Schulze 2015). This has the effect that the data management is not as efficient as it could be and there is no guarantee that the data being operated on is the most recent data, or that another ECU isn't modifying the same data elsewhere. Embedded computer systems in general are designed for a single or very narrow range purpose and as such possess very limited resources. To give a sense of

how limited the resources can be Schulze et al claims that a typical micro controller in an ECU has as little as 40-50 KB of memory and operates at speeds of less than 10 MHZ. The limited quantity of resources is a chief hurtle for embedded systems designers, code must be efficient, and robust, security, error checking, and concurrency, if they exist are generally limited.

Another critical component of CAN are real time systems, typically associated with embedded systems but not required, they add another level of constraints to the system. The chief constraint with the addition of a real time system is that it now operates in a real environment and is subject to deadlines, and the physical environment in which it operates. This has the effect that all calculations must be performed with a period of time not to exceed some real limit regardless of any and all external forces that may hinder operation. Automobiles for example are devices whose mass and velocity can easily cause property damage, injury, and loss of life, the braking system for example absolutely must work, and work quickly, even in conditions that are less than ideal such as rainy or icy conditions.

Atomicity, a way of ensuring that an operation or set of operations is executed without interruption is vital to real time systems. In a general purpose computer the operating system is constantly handling interrupts and juggling processes in and out of the CPU, requiring that some operations be atomic is not generally a concern, however in real time systems it is just the opposite, interrupts can not stop the processing of a command to apply the brakes for example. Atomicity in the auto industry is typically handled by using C language or a C language variant for programming, MISRA-C (Motor Industry Software Reliability Association C) is one such variant commonly used in automobile manufacturing. C language is versatile in that a programmer can work in C language at the assembly level or at the object oriented level easily in the same project, there also exist certain libraries that ensure that sections of code will be compiled in such a way as to preserve their atomicity. The other notable benefit of C language over other popular object oriented languages such as Java is that that there is no need for translators such as the JVM, the compiled C language

can be run nativity on the hardware which helps to save on resource requirements which by the nature of embedded systems is already stretched.

Finally all of these embedded systems operating in real time need to be able to communicate with one another, to do this we must introduce networking into the system. Networking generally faces several major concerns, collisions, errors, and security. Since this is a real time system collisions and errors are very much a concern while security according to Schulze et al has been neglected for a very long time, only recently has it become a research focus. The CAN system uses a broadcasting method for transmitting data across the network, combined with the real time aspect packets can be sent at any time, collisions are bound to occur in such a system. It is therefore imperative that a priority system be in place to handle the collisions and ensure that dead lock does not occur much for the same reasons atomicity must be maintained.

In order to handle collisions CAN implements an identifier on each packet which specifies the priority level of the packet, when a collision does occur the packet with highest priority is sent. Arbitration of packets is handled via an identifier on each packet, CAN uses two modes for packet identification, an 11 bit identifier for standard messages, and 29 bit identifier for extended messaging, the lower the number the higher the priority typically. Since CAN uses a broadcast system to deliver packets, every device receives a copy of the packet and determines if the packet is intended for them or not and acts accordingly to each packet.

Security and error checking need to be done quickly and cheaply in an embedded real time system, CAN implements five error checks, two at the bit level and three in the message, in addition there is also a resend request, If a system should send too many faulty messages the offending system can be shutdown (Carsten 2015). Security is very expensive in terms of processing power which is limited in embedded systems, as mentioned earlier since a vehicle is assumed to be an isolated system without public facing ports and resources are limited security is often simplistic.

As automobiles begin communicating with the internet and remote devices it becomes clear that they have wide open networks with no real way to protect themselves from malicious intent. Access to the bus can come several ways; physical access through the OBDII (On Board Diagnostics) port, a federally mandated interface in all cars, through wireless communications with an ECU or component installed on the network, or through compromised after market or OEM parts (Carsten 2015).

According to their website, Progressive Auto insurance offers a device they refer to as Snapshot, effectively it plugs into the OBD II port and gathers data from the ECU's about the drivers habits. While the ethics of this are outside of the scope of this paper, I believe it is obvious that this technology can be used for malicious purposes as the snap shot device now creates a wireless link from the cellular network directly to the cars unprotected bus. The same information being gathered by the intended recipient namely Progressive Insurance company can also now be gathered by data thieves.

Another common attack to networks is DOS (denial of service) attacks, there are several that can afflict the CAN, DDOS (Distributed denial of service), and spoofing. As it is the CAN uses broadcasting without verifying the sender, flooding the bus with spoofed high priority packets can easily render the system inoperable, or worse, one could alter the physical operations of the vehicle while in motion. In addition the error checking system can easily be manipulated into shutting down communications with a system, the more critical the system the more dangerous the attack. Police use the ability to send messages along the bus to fight car theft, in the televised series “Bait Car” the undercover officers install a device in unmarked cars and wait for thieves to come along and unwittingly steal the car. The police then tail the suspects and using remote access to the bus are able to disable the vehicles engine, lock doors, and disable power windows to prevent escape. As with an technology if the good guys have it, the bad guys can get it too, imagine how easy it would be to break into a car with remote access to send spoofed packets onto the bus to unlock the

doors, or start the engine. Anyone running a script could potentially cause the vehicle to perform functions from irritating such as blasting the radio or heater, to more dangerous things such as apply the breaks unevenly or turn the steering wheel at speed.

The addition of ECU's have allowed diagnostics to be far more efficient and accurate, it also allows for the vehicle to modify operational components to achieve better efficiency, such as adjusting the crankshaft, timing, or mixture of fuel and air ratios. These same systems however could potentially be used to achieve inefficiencies with malicious intent. Altering the timing of a vehicle to decrease life expectancy of parts, or reduce fuel or emissions efficiency to attack a user or manufacturer.

None of these are new problems in computing, but as a combination of computing systems, automobiles do create some unique challenges that prevent the use of existent tried and true solutions. The attack vectors that exist in each of the three sub fields have all been solved independently, but how for example to you add proper networking security to an embedded real time system whose constraints are limited resources, and limited time? There are a few papers out currently with some proposed solutions to some of these issues, but as the author points out they are partial solutions at best, and need to be better adapted before they could be realized fully.

A data management system (DMS) as proposed by Schulze et all might provide some solutions for CAN. It is theorized that a DMS could provide uniform data structures, concurrency controls, and access management. According to Schulze et al, the current arraignment of ECU's is ad hoc, each using it's own internal data structures which give rise to inconsistencies and concurrency issues, which serves to greatly increase the complexity, and decrease, flexibility, extensibility, and maintainability. First by creating uniform data structures data could be more easily verified globally through the DMS, in addition concurrency control could be handled there as well, ensuring that any data is the most current form and correct. Access management which has been overlooked for too long is now critical, whether an attack is used to listen or to spoof they

operate by gaining access to the unprotected bus. A DMS could control and monitor read write access thereby thwarting unwanted access.

Schulze et al proposes three modes of DMS that could in theory provide these solutions, they are as follows; central, distributed, hybrid DMS. A central DMS would incorporate one device centrally located in one ECU, in which all data would pass though for verification and concurrency purposes, this of course would create a bottle neck and single point of failure, the authors agree that this is not their ideal solution. The distributed version would place the software on multiple ECU's throughout the system, increasing the computing power of the DMS, and limiting the bottleneck effect and eliminating the single point of failure. The distributed system would however greatly increase the amount of traffic on the already strained network, the authors instead propose a hybrid solution. The hybrid DMS would have one ECU on each subbus host a DMS, this would give the benefits of a distributed DMS without increasing the traffic to much, and avoid the single point of failure. With the hybrid version each of the three DMS could communicate with one another to offer global concurrency controls and verify data, yet should one fail the other two could still manage the system.

This proposal offers some partial solutions for the networking aspect, it offers security from external attacks, and concurrency solutions, while limiting additional network traffic. The proposed solution however in my opinion places an extra strain on both the real time and embedded aspects of the system, the more traffic the more collisions which are not good for real time processes, the more computations the less free resources for the embedded systems. According to Carsten et al, the proposal is more a proposal of why a DMS would not work rather than why it might, and that furthermore it offers only a conceptual model and no actual implementation. Without a real implemented model it is difficult to predict or test this model against the current systems already in place.

Ling et al propose an algorithmic solution to thwart DOS attacks and error flag exploitation.

Recall that CAN operates in the following ways; message priorities, broadcasting, error detection leads to ECU shutdown. Due to these features an attacker can shutdown ECU's one after another, spoof messages with high priority and either block access tot eh bus or send messages with malicious and harmful intent, or snoop on the vehicle. The algorithmic approach is designed to detect such intrusions and can help determine the legitimacy of the sending device.

The Detection algorithm itself operates by assigning all known devices a starting identifier, when unknown devices send too many messages the system sends off an alarm, preventing a large influx of messages from flooding the bus unchecked. In addition the algorithm also listens for known devices and if too many messages are sent an alarm will be raised that too many messages are coming from that system. This in effect allows a detection of spoofed messages, and DOS attacks. According to Carsten et al, the algorithm itself does function properly and identifies suspicious behavior, and adheres well to the limited resources and real time constraints involved in an actual CAN system. The authors of the Algorithmic approach however fail to elaborate on exactly what is to be done once an intruder is identified, or how the initial system state should be setup to keep track of all initial ID's. Further work on how to respond to these alarms would be need to be conducted, as it is now the algorithm can detect intrusions but is powerless to effect any stop.

Attestation-based security as proposed by Oguma et al attempts to verify the legitimacy of the software in a vehicle upon start up each time based on an original manufacturers hash table key system, in order to prevent tampering. The central component of this security proposal is a master ECU that inherits from the manufacturers servers a serial number and key, Which it will use at vehicle start up to verify all devices on the network are genuine. Once all the devices are proven genuine they will include in their packets part of their key and a counter to insure the packet is from a genuine ECU and to prevent packet flooding. According to Carseten et al the solution is very ambitious and encryption heavy which might prove too difficult for embedded systems with few

resources. I have two concerns as well about this proposed solution, one being the system Oguma et al used to test was a desktop computer with many more resources than any group of ECU's. My other major concern for a solution such as this is that as Oguma et al state “Vehicles have a relatively long life, usually more than 10 years, compared with IT products.” (Oguma et al 2008). Automotive manufacturers are in the business of selling cars to customers and one factor in doing so is the resell value of a brand, part of the manufacturers suggested retail price is based on resell value. Secondary consumers buy used vehicles in hopes of being able to maintain vehicles after the warranties have long expired and often take cars to non-dealer mechanics or do repairs themselves. A vehicle whose on board computers cannot be easily replaced by non-dealer mechanics will surely suffer in demand effecting over all first time sales. Oguma et al does not provide any detail on what should happen assuming the ECU's do become corrupt, ECU's that can make up as much as 23% of the initial cost of producing the vehicle. If the vehicle was not allowed to start if it contained malware, or if the ECU's could only be replaced via a dealership, that would put a lot of responsibility on the dealer to have these parts in stock for 20+ years at a time, and financial burden on used car owners who might be required to choose between vehicle disposal or high repair costs.

Phung et al propose a solution that uses of a device called a transformation module that would stand between the network and the wireless gateway. In essence the goal would be to not burden CAN with any extra work and essentially shield it from malicious intrusions through this transformation module. However according to Carsten et al it is an interesting but vague solution and the paper focuses more on add-on software security than ECU security. The paper offers a Java based solution to the problem, and only gives a modest proposal as to how to implement the transformation module in C or C like languages, and no where does it mention how long such a transformation operation would take (Carsten et al 2015). When one considers we are working with embedded systems the majority of which are implemented in C or ADA (Rogers, 2011), Java solutions are rarely applicable.

The problems inherent with in vehicle networks are not new problems, each sub field of computer science has solved these issues independently, it is only when we combine these sub fields with the automotive industry that new hard to solve problems arise. Each of these proposed solutions typically tackled most but not all problems while creating new ones, or leaving others unsolved. The correct solution to protect ourselves and vehicles from malicious intent will be one that respects the limited resources of embedded systems, meets the constraints of real time systems with elegant fast code, and provides the networking with security, error checking, and concurrency. As Ling et al wrote "Safety is the degree to which accidental harm is prevented, reduced, and properly reacted to; and security is the degree to which malicious harm is prevented, reduced, and reacted to". Automotive marketing were concerned with features and sales, as Schulze et all said "The IT security in automobiles was neglected for a long time, but becomes more and more a focus of research due to the potential devastating consequences when it is violated." Perhaps the best solution of all lies in proper design from the ground up and waiting to sell new features until they are deemed safe.

WORKS CITED

Barr, M., Massa, A. J., & Barr, M. (2006). *Programming embedded systems: With C and GNU development tools*. Sebastopol, CA: O'Reilly.

Carsten, Paul, Todd R. Andel, Mark Yampolskiy, and Jeffrey T. Mcdonald. "In-Vehicle Networks: Attacks, Vulnerabilities, and Proposed Solutions." *Proceedings of the 10th Annual Cyber and Information Security Research Conference on - CISR '15*. Print.

Ling, Congli, and Dongqin Feng. "An Algorithm for Detection of Malicious Messages on CAN Buses." 2012 National Conference on Information Technology and Computer Science. Atlantis Press, 2012.

Oguma, Hisashi, et al. "New attestation based security architecture for in-vehicle communication."Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE. IEEE, 2008.

Phung, Phu H., and Dennis Kengo Nilsson. "A model for safe and secure execution of downloaded vehicle applications."(2010): 06-06.

Rogers,J.S. (2011). Language choice for safety critical applications. *ACM Sigada Ada Letters*. doi:10.1145/2070336.2070363

Schulze, Sandro, et al. "On the Need of Data Management in Automotive Systems." BTW. Vol. 144. 2009.