Securing Self-Driving Cars (one company at a time)

Dr. Charlie Miller (charlie.miller@getcruise.com)

Chris Valasek (chris.valasek@getcruise.com)

August 2018

Contents

Introduction

This document outlines the experiences of the authors securing self-driving cars over the last few years and several jobs. There is not a lot of information available about the security of self-driving cars and that which is available is not from those with practical experience in the field. This paper sets out to lay down the fundamentals in this field in an effort to share information with both the public and other practitioners to advance the field overall and secure these amazing vehicles.

A "self-driving" car can mean many things to many people. Attack surfaces, threat models, and the security of these cars-of-the-future also appear to be quite subjective. This document means to directly address many of the ambiguities. While the subjects covered in this paper may not be completely comprehensive, it should give the reader insights into different levels of self-driving cars, current threat models, security architecture, and how we can work toward securing the driverless revolution.

Self-driving car definitions

Self-driving cars, also known as Autonomous Vehicles (AVs), come with a certain level of ambiguity. To clarify the distinctions of self-driving cars, SAE International developed the J3016 standard [1]. This standard categorizes the six levels of automation available in vehicles. Sometimes technologies will be difficult to classify as they may share elements from more than one category. In these cases, we use the examples provided by Car and Driver magazine [2].

Level 0: No automation

This describes vehicles that have no automation technologies to aid the driver. The driver is in full control of the vehicle at all times. Think of cars from twenty years ago. According to the definitions that follow, even a car with standard cruise control would fall into this category.

Level 1: Driver assistance

Driver assistance includes technologies that aid the driver with either steering or the speed of the car, but not both at the same time. The driver still has full responsibility for the vehicle and oversees monitoring the environment. Examples of these technologies include the Jeep Cherokee's Adaptive Cruise Control and Lane Keep assist functionality.

Level 2: Partial automation

At this level, the car can control the steering, acceleration, and braking in certain conditions. However, the driver still has responsibility to observe the environment and may be asked at any time to take control back over. These vehicles typically nag the driver to keep their hands on the wheel or their eyes on the road. Examples of this include the Tesla Autopilot and the Volvo Pilot Assist.

Level 3: Conditional automation

In the right conditions, the car can control all elements of driving including monitoring the environment. A human is still necessary as the car may prompt the driver and return control if it encounters a condition it cannot handle. The driver has no obligation for controlling the vehicle but must be prepared at any moment to take back control. An example of this technology is the Audi Traffic Jam Pilot which will be available on some 2019 Audi vehicles.

Level 4: High automation

Level 4 vehicles can operate without a human driver in areas restricted by road type or geography. So, within a pre-defined region or road, this vehicle can be operated without any responsibilities of a human. This may include a vehicle without steering wheel or other operator pedals. Examples of this include most of the "self-driving" cars currently being piloted by companies like Cruise, Uber, and Waymo.

The inside of a level-4 self-driving car from Cruise.
[https://www.theverge.com/2018/1/12/16880978/gm-autonomous-car-2019-detroit-auto-show-2018]

Level 5: Full automation

This is the holy grail for self-driving vehicles. The vehicle needs no human interaction and can operate on any road at any time. To give you some perspective, level 4 self-driving cars currently must gather high-resolution maps in a certain area before operating, while this operation would not be necessary for level 5. There are no current examples of vehicles with this level of autonomy, but it is the ultimate goal of vehicle manufacturers.

Use Cases

Beyond level of automation, there are a couple of different use cases for self-driving cars. One is production vehicles available for sale to customers. This is the traditional vehicle that we own and see on the road now, but with an advanced autonomy package provided. As discussed above, there are currently no vehicles for sale that have level 4 or greater automation available. This may be caused by the excessive cost or maintenance or other reasons. Regardless, most auto manufacturers are thought to be developing these types of vehicles, but none are currently available.

The other type of vehicle is a ride-sharing vehicle. These are vehicles that are owned by a company, but end users may ride in them. Examples are the vehicles being produced and tested by Uber, Waymo, and

Cruise. These vehicles come home every night to a garage for constant supervision, updating, and maintenance.

In this paper, we discuss the cybersecurity of this last class of vehicle, namely those that are **not owned by end users** and **have level 4 automation available**.

Current State of the Self-Driving Car

Before diving into the security of self-driving cars, it first makes sense to discuss the current state of them. It is important to go over the timelines and other details to understand the relative importance and speed necessary in securing them.

Waymo has been working on developing self-driving cars since 2009 [3]. It is amazing to think this technology is less than ten years old. Other companies, such as Uber and Cruise, started several years later [4][5].

In many places, regulations regarding self-driving cars do not exist. In places with regulations, until very recently, it was a requirement that in self-driving vehicles a human needed to be behind the wheel prepared to take over in case of an emergency. These humans are sometimes referred to as safety drivers. For example, since 2014 in California, it was legal to have a self-driving car as long as there was a safety driver behind the wheel.

This has begun to change. In March of 2018, Arizona made it legal for autonomous vehicles to have no driver [6]. This was the first law allowing level 4 and 5 autonomy. Since May of 2018, California allows a self-driving car without a driver with some restrictions in pilot programs [7]. These restrictions include that a remote operator must be monitoring the vehicle. Also, no money can be charged to the passengers for the ride and the vehicle requires a way to communicate with passengers or law enforcement officials in the event of an accident. Most interesting for this discussion, the California regulations also specify that the vehicles needed “industry best” protection against cyber-attacks.

So, in some places in the US, self-driving car projects have been given the green light by state governments. But how long will we expect self-driving cars to remain in the minority? When can we expect the majority of vehicles on the road to be have level-4 or above autonomy features? The answer is it will be some time.

To get an idea of the size of the current self-driving car fleets, you can look in some industry articles. Over the next few years, Waymo has agreed to buy 62,000 Chrysler Pacifica Minivans [8]. Meanwhile Uber agreed to buy 24,000 Volvo XC90 SUVs [9]. So, in the not-so-distant future, you may expect to find roughly 100 thousand self-driving cars on the road.

If you compare this to the 253 million cars in the US [10] or even the 3 million Uber drivers [11] you can see that it will be some time before most vehicles are self-driving. However, as the technology becomes tested and automobile manufacturers begin selling self-driving cars, it is not difficult to imagine self-driving cars dominating the roadways in a decade, especially in dense urban areas. Many car manufacturers plan to have level 4 or 5 vehicles available for purchase by 2021 or 2022 [12].

The Waymo self-driving car based on a Chrysler Pacifica
[https://jalopnik.com/googles-mass-purchase-of-chrysler-pacifica-hybrids-coul-1826537718]

Even within the existing self-driving fleets, a majority of vehicles are being tested in just a few locations. Most Waymo vehicles can be found in Mountain View, CA or Phoenix AZ [13]. Likewise, Uber's self-driving vehicles mostly live in Pittsburgh, PA [14]. Most Cruise vehicles drive around downtown San Francisco.

An Uber self-driving car based on a Volvo XC90
[https://timedotcom.files.wordpress.com/2016/08/ubervolvo-xc90.jpg]

You must remember that a typical self-driving car in these programs drives a whole lot more miles than a typical user owned vehicle. For example, reports indicate that Waymo vehicles have driven 4 million miles. Uber self-driving cars are reported to have driven 2 million while Cruise vehicles are reported to have been ridden around 500k [15]. For comparison, the average driver puts about 12-16 thousand miles on their car a year [16].

That means it would take the average driver 384 years to drive as much as the Waymo fleet has driven. By those numbers alone you can see how self-driving cars can be beneficial. Driving orders of magnitude more miles will result in experiencing many different scenarios that many drivers may never see in a lifetime. On top of that, human drivers typically avoid problematic areas: construction, high density of pedestrians, etc., while self-driving cars seek out those challenges to ensure that the technology can handle as many corner cases as possible [17]. Therefore, it is not only the amount of miles being accumulated, but also the complexity of miles driven. So while miles driven are not the only important metric, they do provide some insight into the amount of testing being done by self-driving car companies.

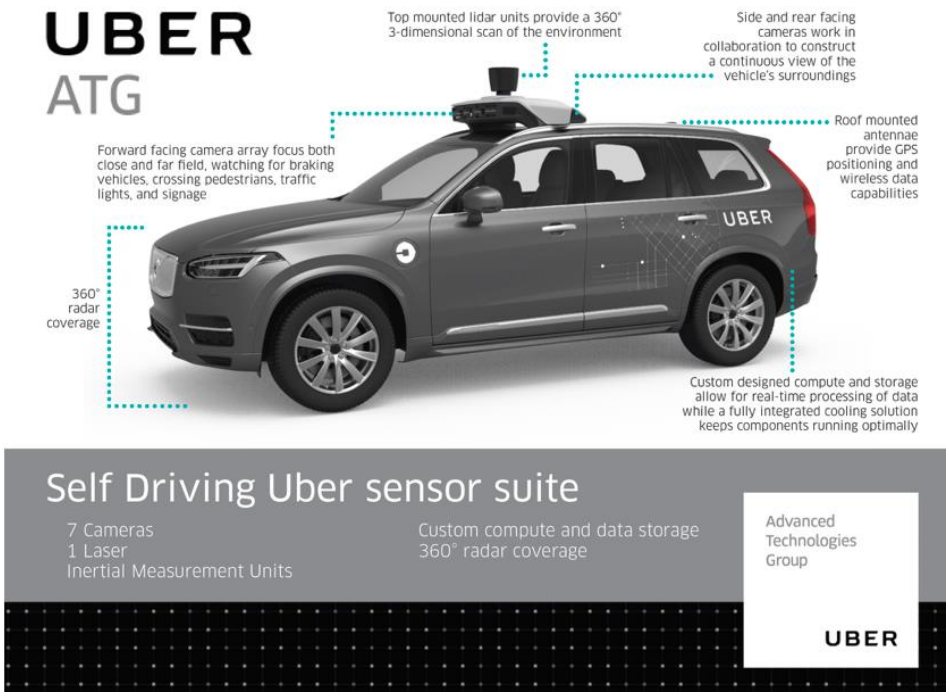
Self-Driving Cars 101

Before we discuss the security of self-driving cars, we need to understand how they work. Just a reminder to the read, our focus is on level 4 autonomous vehicles.

Hardware

Self-driving cars contain an abundance of hardware that does not exist on normal passenger vehicles. The most obvious difference is the additional sensors used by the self-driving stack (discussed below). That being said, some of these sensors can be found on high end automobiles, such as cameras and radar.

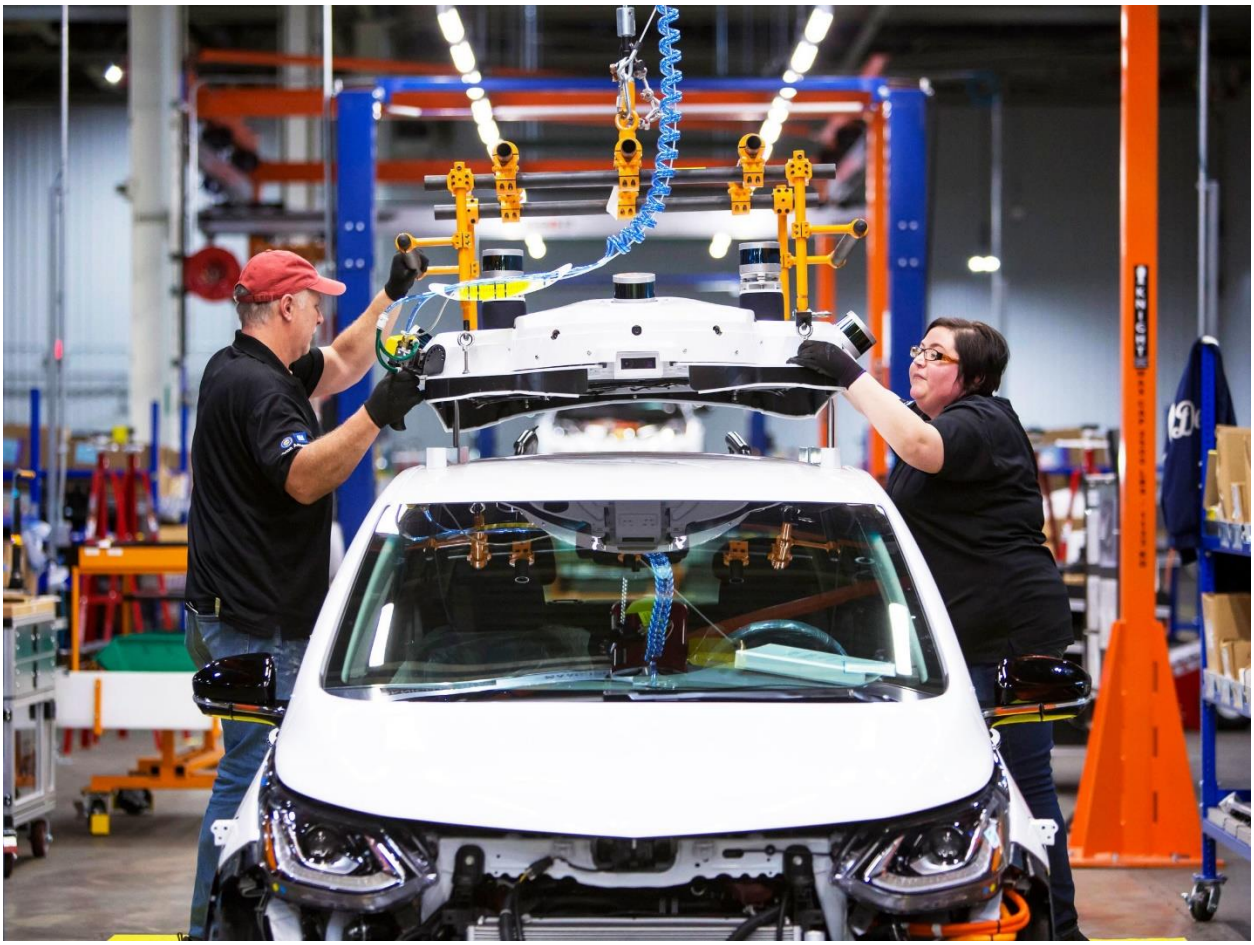
For example, the Jeep we studied in 2015 had radar for adaptive cruise control and a camera for lane keep assist features. However, on self-driving cars you may find many more cameras leveraging a much higher resolution. As shown below, Uber vehicles have seven cameras.



Seven cameras on Uber self-driving car [https://20kh6h3g46l33ivuea3rxuyu-wpengine.netdna-ssl.com/wp-content/uploads/2018/03/Screen-Shot-2018-03-19-at-7.25.12-PM.png]

The largest difference between production cars you can buy and level 4 autonomous cars is the Lidar. Lidar is like radar except it sends laser pulse pings and measures echoed readings from the reflection [18]. It sends out millions of laser beams per second, building up a detailed view of its surroundings. These devices are very sophisticated and can cost upwards of $75,000 per unit [19]. This cost leaves this technology out of reach for consumer automobiles, at least for now. Even more so, most self-driving car research groups are attempting to produce or purchase low-cost Lidar to significantly decrease the cost of manufacturing.

Older Uber Lidar sensor rack (Ford Fusion)
[http://media.pennlive.com/life/photo/self-driving-uber-ff4637c53b737463.jpg]



Workers at the Orion assembly plant adding the roof sensors for a Cruise self-driving car [https://www.wired.com/story/gm-softbank-investment-tesla-braking-news-roundup/]

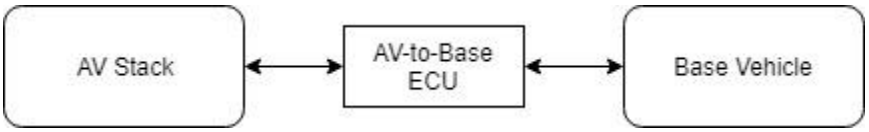
A waymo Lidar unit (during Lidar wiping testing)
[http://autoweek.com/article/autonomous-cars/waymo-wipers-coming-soon-autonomous-vehicle-near-you]

Another important difference is the computers that are present. It takes significant computational power to process the sensor input and make decisions about the world around the vehicle. This typically manifests itself in one or more large computers stored in the vehicle. Currently these compute modules require significant amounts of RAM, several multi-core CPUs, and multiple GPUs to process all the data being sent from the sensors (lidar, radar, cameras, etc). Additionally, the compute modules take up significant space, not to mention the hardware required to keep such a computer cool.

Audi compute stack
http://www.extremetech.com/wp-content/uploads/2013/01/Audi_10TTSC_PikesPeak_12_hrcmyk.jpg

After evaluating inputs and using complex algorithms, the compute stack will determine what it wants the car to do in the real world. For example, this may mean turning the steering wheel or applying the brakes. The computer needs to signal these components with its intention.

Currently, most self-driving vehicle computers (i.e. the AV stack) are decoupled from the underlying base vehicle. That is, there needs to be equipment between the portions of the vehicle that calculate the maneuvers of the AV and the underlying base vehicle, which contains hardware identical to currently available passenger vehicle.



Terrible diagram of AV stack and base vehicle communications

Be it that in most cases, self-driving cars are currently an add-on to a standard passenger vehicle, a component to communicate between the two functional units is required. This unit gives the AV stack access to the base vehicle (example: “turn the wheel 3 degrees clockwise”) and the unit also gives the base vehicle access to the AV (example: “my current steering angle is 94 degrees”). That being said, depending on the self-driving company and the vehicle in use, the separation between the AV stack and the underlying platform may be less apparent and the AV equipment may be more tightly integrated.

Beyond that, it is extremely likely the firmware on the steering and braking computers will need to be modified. Most automobiles will not allow you to inject CAN control messages and safely and reliably control the function of the vehicle. If you want to see the not-so-granular control normally offered to someone trying to control a car via a Nintendo controller, you make like this reference [20].

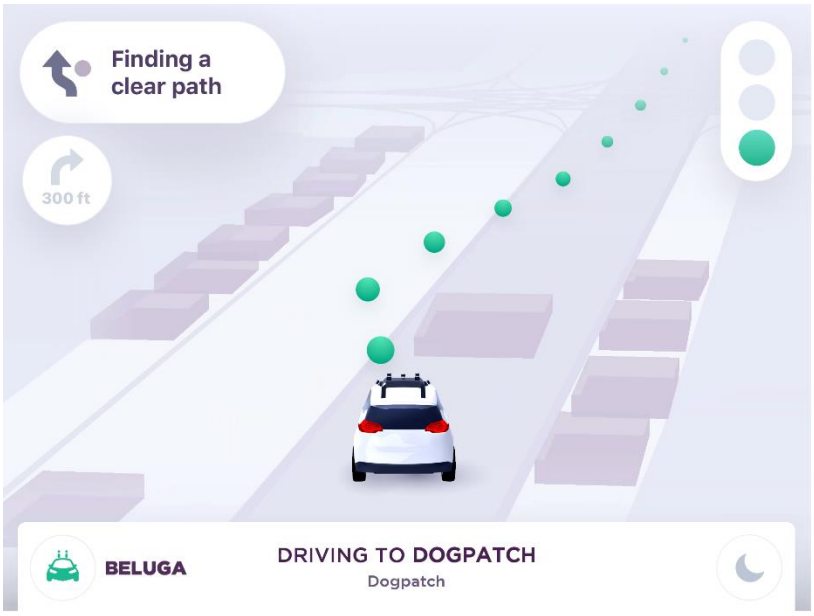
The base ECUs in question need to be modified to allow control messages to come from multiple sources or to accept entirely different types of CAN messages sent by the AV stack. So, while the ABS ECU is probably the same hardware as a car you have at your house, it is very likely the software running on it will be different than on a self-driving car made to run on the same platform.

Another difference is that self-driving cars need a way to communicate with their passengers. The major players all do this by providing a tablet type interface in the back seat of the vehicles.

User interface tablet in Uber self-driving car
[https://www.wired.com/2016/09/heres-like-ride-ubers-self-driving-car/]

These tablets allow the user to signal to the vehicle that a stop or destination change is required. Typically, the tablets will display information the vehicle is processing as well. It may include readings from its sensors or which paths in the road it is planning to take.

Waymo user interface tablet [https://techcrunch.com/2017/10/31/waymo-self-driving-ux/]



Cruise user interface
[https://techcrunch.com/2017/11/29/taking-a-ride-through-sf-in-cruises-self-driving-bolt-ev/]

Software

The way actual self-driving software works is mostly unimportant from a security perspective. In many ways we can treat it as a black box that takes in sensor input and outputs actuator commands. It doesn't matter to us whether it uses a machine learning approach, neural networks, or an enormous mass of procedural code. The only time it becomes necessary is when we think about attacks against the sensors. So, for this reason, we will briefly describe how it works.

The main difference between level 4 and level 5 autonomy is that level 4 vehicles are restricted to where they can operate. This is because they rely on detailed maps that they use to help them understand the world as they see it.

Imagine driving down a street you have driven down a hundred times. You may notice a parked car that is not usually there. This is the same principal used by these systems. If the system had to look at all inputs from, say, a Lidar it would be difficult to determine what everything in it is. But, if it knows exactly what to expect from a Lidar reading from this exact spot in the road, it will be able to notice anything that isn't usually there. This becomes a more computationally feasible task. Instead of identifying everything in its environment, it only needs to be able to identify things that aren't usually in its environment, whether that is a pedestrian, a parked car, a dog, or a driving vehicle.

Additionally, by using this set of internal maps of its environment, the vehicle can determine its location to a very precise point. By comparing its lidar sensor readings to a map of previous lidar sensor readings, it can know exactly where it is to a level that would not be possible using something like regular GPS. This process is known as localization. Likewise, these maps tell the vehicle what the speed limit is and where traffic lights and crosswalks are located. It doesn't need to figure this out on the fly by using cameras.

Knowing how self-driving cars actually work makes it clear that certain attacks often described simply won't succeed. For example, consider discussion about tricking self-driving cars by putting stickers on

stop signs to make them hard to process by cameras and character recognition systems [21]. But as outlined above, the vehicle already knows where the stop signs are and so does not need to figure this out dynamically. Likewise, if you plan on jamming GPS [22] or even cellular connections in an effort to defeat a self-driving car, you will likely fail. As described above, a self-driving car can happily navigate for some amount of time without these signals using its internal maps and localization.

Compute

As mentioned previously in this paper, currently a self-driving car needs quite a substantial amount of computing power. This compute stack (or AV compute stack as we'll call it) generally consists of a computer that has multiple multi-core CPUs, excessive amounts of RAM, and multiple GPUs. Think of a Bitcoin mining rig, not a laptop. The computation and redundancy required for self-driving means there are typically multiple compute modules in a single vehicle.

These compute nodes tend to run some Linux variant as it promotes a well-known platform for rapid development. Luckily, we generally know how to secure a Linux server.

Connectivity

In our experience, these self-driving cars will be utilized to pick up passengers and drop them off at different locations, all while ensuring that there are no problems in between. To provide this type of service, the AV will need one or more communication modules. These modules are the vehicle's access to the outside world. In addition to taking passengers on trips, these modules are used to send information back to home base. For example, you will want to know where the vehicle is, statistics about the AV compute stack, and even the current temperature of the vehicle. This telemetry information is gathered internally and sent off to the back office through the communications modules.

Back office

Both the software and connectivity sections mention the vehicle communicating with a central server and other traditional infrastructure components. The back office is a critical piece of operating autonomous vehicles.

The back office is responsible for communicating changes to the vehicle's map. These might include roads that are closed, construction, temporary changes in speed limits or temporary traffic lights. Additionally, when the vehicles return home each night, they will have a whole day's worth of lidar data, camera data, etc. that needs to be stored for potential future analysis. The data needs for these activities are enormous.

The vehicles also need to communicate with centralized management infrastructure. While the AVs can drive themselves, the owners still need to manage the fleet, supply software updates, and know the overall status of each vehicle.

End 101

As we've seen, a self-driving car has a lot of additional hardware and software within it. The hardware is typically expensive, cutting-edge components that need to communicate with each other. For example, lidar and camera data needs to get to the main computer and the computer needs to signal to the actuators.

Additionally, communications from the home server as well as sensors and camera feeds all need to reach the main computer. This means there needs to be a network for this communication to occur. Due to the nature of the components on this network, it is typically going to be an Ethernet network.

In general, in a self-driving car, you expect to see an Ethernet network or networks with a main computer, tablets, sensors, and communications modules as well as a module to allow these components to talk to the rest of the base vehicle. This will all be important to consider when we begin talking about securing these vehicles.

Automotive Cyber-Physical Attack Recap

In order to talk about the best ways to defend self-driving cars, it is beneficial to first look at attacks that have occurred in practice against standard connected cars. Understanding how entire attack chains work will give us insight into how to efficiently prevent these types of attacks in the future. The attacks discussed below begin by leveraging a vulnerability to execute code and proceed to move to physical control of an unaltered passenger vehicle. While there certainly are numerous other types of attacks (door key replay, jailbreaking head unit, etc.), we find attacks that are launched remotely and end with physical control of the vehicle to be the most important.

2011 Chevy Malibu

The first remote compromise of a vehicle that resulted in cyber-physical control was presented in 2011 against a Chevy Malibu by Checkoway et al [23]. They showed two different remote attack vectors.

In the first, they compromised the vehicle's radio using a vulnerability in the Bluetooth stack. This had a limitation in that either the attack had to come from an already sync'd phone or else the attack took an extended period of time to execute. After the radio was compromised, the attacker was isolated from the high-speed CAN network by a gateway device. However, they could reprogram this gateway device from the low speed CAN network, with which they had access. After this, they were able to send messages which could make the brakes lock up.

In another attack, they could remotely compromise the vehicle's telematics unit using its cellular connection. They could do this by dialing the unit's phone number and communicating with its modem. This required no user interaction by the driver. After compromising the telematics unit, they could directly send messages to control the brakes since the telematics unit was already on the high-speed CAN network.

2015 Jeep Cherokee

In 2015, the authors of this paper performed a similar attack against a Jeep Cherokee [24]. They could compromise the head unit of the Jeep due to a vulnerability that was accessible through the Internet. After getting code running on the head unit, they were able to reprogram the firmware of another processor on the head unit that had CAN network access. After this, they could send CAN messages which controlled the steering, brakes, and acceleration of the vehicle.

None of this required any user interaction and in fact the exploit was worm-able and could have quickly compromised all of the 1.4 million vulnerable vehicles.

Damn, that was baller.

2016 Tesla Model S

In 2016, members of Tencent's Keen Security Lab were able to remotely compromise a Tesla Model S [25]. They took advantage of the fact that the CID (i.e. information display) ran an old version of a web browser. If they could trick a user into visiting a malicious website with the browser, or if the car had previously joined a well-known Wi-Fi network (like the Tesla Guest Wi-Fi network at a dealership), they could exploit this vulnerability if they were in physical proximity.

After exploiting a vulnerability in the CID, they could reprogram the vehicle's gateway device which was connected to the CID over ethernet. This granted them the ability to send CAN messages which they used to engage the brakes of the vehicle.

Later Tesla added code signing to the gateway device to make reprogramming of it harder. These researchers later demonstrated a way to bypass this security mechanism and still reprogram it with unsigned code [26].

2018 BMW i3

Quite recently, Tencent's Keen Security Lab expanded their view beyond Tesla to a BMW i3 electric vehicle. In this case, if they were able to get a vehicle to connect to their rogue cellular base station, they could send a series of SMS messages to the vehicle that would compromise the telematics unit [27].

The telematics unit was isolated from the high-speed CAN network by a gateway device. They could not completely bypass this gateway device but were able to send diagnostic CAN messages through the gateway to the high-speed CAN network, theoretically allowing them to reprogram the ECUs on that bus.

As far as we know, they did not actually demonstrate cyber-physical control of the vehicle but since the research is relatively new, this may be possible.

Analysis

In general, the attacks described above share a set of common characteristics. The first step in the attack chain is the remote attack. Of these, some were more severe than others. They differed on the distance the attack could be performed, whether any user interaction was necessary, and if any special equipment was required. They also varied on what internal component the attacker ended up executing code on.

While having a remote attacker executing code on the vehicle is bad, the most dangerous part is when they could begin sending messages to the vehicle's critical ECUs. In almost all the cases, this was not possible without additional work. By design, the components that speak to the outside world were often somewhat isolated from those of the critical internal networks. This usually required the attacker to reprogram a device that was acting as a gateway. In the authors experience, this was not a trivial task and could be considered the most pain staking portion of 'car hacking'.

In the final step, the attacker was able to send CAN messages to control physical aspects of the vehicle's operation. For the most part once an attacker can send arbitrary CAN messages they can adversely affect the physical behavior of the vehicle. Research shows that the level of control varies per manufacturer and per researcher. However, even in the case where the attacker gets to the point that they can send CAN messages, it may still be possible to detect these new messages and stop the attack.

As a defender, we need to begin thinking about how to make the entire attack chains described above harder. If we can make any of the above operations extremely difficult, it may be enough to stop a remote cyber physical attack against a car, whether self-driving or not.

Advantages of Securing Self-Driving Cars

At first glance securing self-driving cars seems like an impossibly daunting task. For example, almost all self-driving vehicles are based on a readily available production vehicle. These production vehicles likely have security vulnerabilities and so one would think that the AV would share these. Additionally, self-driving cars have additional attack surface through their communications modules, as well as more computers and sensors.

Beyond this, attacks against self-driving cars seem to be higher impact than against normal cars. In a cyber-physical attack against a standard car, the driver has the possibility to maintain control of the vehicle. In contrast, an attack against a car without a steering wheel or brake pedal, the passenger has more reliance on the vehicle safety and security systems and will be less able to respond to an attack. At first glance, the problem of securing autonomous vehicles seems significantly harder than securing ordinary passenger vehicles. However, this is not necessarily the case.

One advantage is that in the current designs, the vehicles are owned by a service and are maintained by that service. This means the vehicles will be returning to a garage every day where they can be examined, modified, or updated. These modifications may include software or hardware changes. Any changes that need to be made, including security patches or cryptographic key rotations can be updated at this time. You can contrast this with production vehicles that, once they are purchased, may never be seen again by the manufacturer. Many times, the manufacturer is hoping to never see the car again since their objective is to make a perfect product.

Another advantage of the current self-driving vehicle is that it will be closely monitored. Automobile manufacturers don't typically track vehicles locations, speeds, or receive a constant stream of health updates and logs from their vehicles, but this is the case for self-driving cars. This can help spot problems before they are major issues. The other aspect of this level of control by a central authority is that if a problem is ever detected or a security violation occurs, it is possible to have all the vehicles return to their garage for inspection within a matter of minutes. Vehicles could even be immediately (and safely) shut down pending a critical security event.

Yet another advantage while securing self-driving cars is that in many ways they can have less attack surface. You don't necessarily need (or want) to pair your phone to a self-driving car using Bluetooth. All the remote attacks in the past involved the head unit/radio or telematics unit. These units can simply be removed from self-driving cars since they are largely unnecessary.

Instead, you'll have a custom communications module for these communications which can be more locked down. For example, some automobile head units have a web browser. This unnecessary attack surface will not be present in self-driving cars. Likewise, easy physical access points to a vehicle, such as an OBD-ii port, can be removed or locked in a non-standard location.

The final advantage is one of obscurity. While the entire security of a self-driving car should not rely solely on this, it does make it more difficult for an attacker. Attackers won't have significant access to a

self-driving car. They won't easily be able to extract firmware or test exploits against these custom components that aren't available for purchase.

For these reasons, at least in the ride sharing model, there are many inherent *advantages* to securing self-driving cars over traditional production vehicles.

Challenges of Securing Self-Driving Cars

While there are many advantages of securing self-driving cars, there are also many disadvantages. While these drawbacks have analogs in the software and hardware industry, we want to highlight several challenges we have faced in our years of securing AVs.

One of the most challenging aspects of securing self-driving vehicles is the pace at which software is developed and goals are set. The world is currently in a race to see who can deploy the biggest fleet of capable self-driving cars. While this pace does spur innovation, it can be hard to perfect security controls as a product is being tested and developed. Luckily for us, we have a background in software, and so security can be treated as an incremental process that requires constant revisions. Gone are the days, as the late great[*] Ron Popeil said: "Set it and forget it!". Security of self-driving vehicles depends on agile security engineers who are ready, willing, and capable to create security controls that can keep pace with development.

Another current disadvantage is having to secure a base vehicle *and* the self-driving portions. While the same problems exist in software (securing your code and all the libraries you use), the pace at which self-driving software is written and tested is much faster than the base vehicle development happens (which has advantages in its own right).

Lastly, as you can tell from this paper, Ethernet communications are key for the portion of the car that is responsible for data input, decision making, and movement. Unfortunately, device manufacturers making Ethernet vehicle components aren't as up-to-date as modern PCs. TCP stacks are non-standard, not to mention functional TLS implementations. More advanced ethernet security mechanisms like 802.1x are probably not possible on these low cost, low power devices.

As you can see, although there are many advantages to securing self-driving cars, like most security endeavors, there are also some challenges to overcome. Hopefully the last two sections provided you with some insight into the day to day pros and cons of securing our autonomous future.

Threat Model

It is important to understand the types and severity of different attacks that the AVs may face. By considering the impact, severity, and likelihood of an attack, we can formulate and prioritize defenses. Otherwise we could not be sure that we are designing defenses that can sufficiently withstand real attacks.

[*] *Note: Ron Popeil may or may not be dead. We could not be bothered to check. However, we do enjoy our rotisseries.

Long Distance (Remote Attacks)

The most impactful and detrimental attacks are those that can be performed from great distance. These typically would have the ability to affect multiple vehicles and perhaps an entire fleet at once.

There are a variety of possibilities for this type of attack. Let's begin by looking at those that directly affect the vehicle. These types of attacks have been executed by researchers and are generally the most concerning when it comes to securing AVs.

- Attack a listening service in the communications module
- Attack against a remote assistance style feature. For example Phantom Auto
- Attack again base devices on the base vehicle: telematics, infotainment, etc.

Another type of long distance remote attack includes attacks against the infrastructure that the vehicles rely on. This may include compromise of the fleet management service, or the service where the car gets its code or updates. Going even further up the supply chain, a compromise of a developer laptop or the source code repository may eventually lead to malicious code on the vehicle. As you can see, remote attacks provide the attacker with methods for exploitation without physical access to the vehicle or even being within eye shot.

Short Distance (Remote Attacks)

While the biggest threat is an attack that scales against an entire fleet at any range, a threat that also must be considered is remote attacks that are carried out over a short range. This type of attack doesn't scale well and, given the sensor and camera packages on most self-driving vehicles, the culprit is likely to be caught. Despite these inherent weaknesses, we still want to make sure that AVs are secure from attack, even at a close distance. These types of attacks are well understood and have been discussed in detail in several previous papers [23] [24]

- Attack against Wi-Fi communications module
- Attack against Bluetooth
- Attack against Tire Pressure Monitoring System (TPMS)

Another group of attacks that fall into this category are attacks against the vehicle's sensors. These might include general jamming attacks or more specific attacks to try to affect what a vehicle perceives in its environment. However, at the time of writing this paper, these sensor-based attacks appear to be difficult to perform outside of a controlled environment and do not scale [28].

Physical Access Attacks

There are a variety of attacks requiring physical access. For the time being, many self-driving car companies have a safety driver that can prevent most of these attacks, but soon, there will be a need to allow untrusted parties extended, unsupervised access to the vehicles. This level of access opens the possibility of this class of attacks.

The most serious of these attacks would be something that would allow direct injection of CAN messages onto the CAN bus. This might occur by implanting a device on the CAN network or plugging in an OBD-ii dongle [29]. A more sophisticated version of this attack includes reprogramming an existing ECU to do something nefarious.

Another point of attack would revolve around the Ethernet network in the vehicle. This might consist of plugging a laptop into the ethernet network and attacking other Ethernet devices like what was shown by Rodgers et al. on the Tesla. Or, an attacker may leave a rogue device on the Ethernet network that would affect the cars behavior at a later time. For example, this device may figure out a mechanism to inject CAN messages from the Ethernet network.

Of these examples, the hardest part for an attacker would be figuring out the internals of the self-driving car, which would not be readily available to the attacker. There will likely be some physical protections in place to make this difficult as well. An exception is the tablets in the back of the vehicle. These are designed to be used by the passengers and so attackers will have extended periods of time interacting with these devices as well as perhaps their wired connections (if any).

Attacking the Base Vehicle

It may be possible to attack pieces of the vehicle that are not specifically part the autonomous compute stack. Self-driving cars are in many ways even more vulnerable to these types of attacks. This is because they have special firmware on the ECUs to allow complete computer control, which might not be possible on a stock vehicle. While many portions of the attack surface (telematics, radios, etc.) may be removed from the base vehicle, some may remain. For example, it is likely the vehicle will still contain TPMS functionality.

Non-Safety-Critical Attacks

There are other attacks that must be considered besides those that affect the physical safety of the vehicles. While these have less impact than ones affecting safety, they are still important to acknowledge and plan as much as possible to defend against. These attacks involve different types of theft, fraud, and personal information.

Defending Self-Driving Cars

After considering the types of attacks described above, it is time to plan a way to defend the self-driving vehicle from attack. It is impossible to make the vehicles attack-proof. We have a limited budget and limited time to install defenses. Therefore, it is important to prioritize defensive efforts based on the analysis of the threat model.

For example, our number one concern is a cyber-physical attack that would threaten the safety of our customers. If a breach occurred that resulted in loss of privacy or personal information, it would be upsetting. However, such an attack fails to compare to the scenario where a car is compromised, and passengers are injured (or worse).

Likewise, we focus our defensive efforts on remote attacks, especially long-range attacks. These would have the largest impacts as they could affect many vehicles in a short timeframe. Again, while we try to secure the vehicle against attacks requiring physical access, we prioritize securing against remote attacks. This is because, while attacks requiring physical access are bad, they do not scale to the same level as remote attacks.

Beyond impact, securing a self-driving car is not significantly different than securing any other computer network. We use the same techniques as we would to secure a very small enterprise network or more precisely a small industrial control network. For the most part, we don't need to invent new ways to secure things. We only need to carefully apply industry best practices to this particular problem.

Trusted Execution

One of our first objectives is to ensure that when the AV boots up, we know that the code running has verifiably come from us. We do this by using technologies like secure boot to cryptographically verify the code through a trusted chain anchored by a key in a write-protected portion of the computer. So, the BIOS/firmware will verify the bootloader, which verifies the kernel, which verifies the software image.

Although code may become altered during runtime, we believe without booting up in a known, verified state, we cannot rely on any of the application code running at boot time. Furthermore, if there is a compromise, a reboot should result in the device being reset to code that is expected.

We would like this system setup on as many of the computers and ECUs as possible. Unfortunately, at the time of writing, it is not possible for all the computers and devices inside a self-driving car. For example, some third-party sensors will not support this level of security. We work with our vendors to try to get these products to implement this security in the future. In this respect, we are on the front line of pushing for advances in security for embedded systems and sensors.

Bootstrapping Cryptographic Keys

Many security mechanisms described in this paper will require cryptographic identities for the components in the vehicle. This begs the question of how you get this identifying information installed on the different vehicles. There are a few options.

One would be to have a secret value or private key burned into components at the factory. This is the simplest solution but has drawbacks around key rotation, management, and revocation. Additionally, this method could increase the cost of producing devices.

Another possibility is to have a user manually enter something at boot time, such as a username/password. This can be exchanged for an appropriate key from a network service. This has the disadvantage that a user must be involved every time the vehicle starts. While this solution may work while there are safety drivers in the vehicles, it might become infeasible once the vehicles become completely driverless.

The last option is to have the vehicle obtain a secret by providing identifying information about hardware present in the vehicle. For example, it could provide the VIN as well as identifiers from the motherboard and other components. This isn't foolproof but does have the property that if a single component is stolen from the vehicle, it will not have enough information to authenticate and receive a secret. Additionally, if this information is stolen from one car, it does not help to authenticate as another car.

The last method, not so shockingly, relies on corresponding infrastructure to support acquiring security artifacts, which goes to show, self-driving car security consists of many teams.

Key Storage

As much as possible, we want to make it extremely difficult for an attacker, even with system access, to extract the secrets used by the vehicle. For example, we could generate and store a private key in a TPM chip. That way the key never leaves the TPM hardware and is difficult to extract by an attacker. For hardware that doesn't have a TPM, we would utilize an HSM, ARM TrustZone, or secure enclave when available.

For situations where secrets are needed, and it isn't possible to store them in a secure manner, we create keys that are very short lived. For example, on each car boot, we obtain session keys that will only be valid for short periods of time, on the order of a few hours. That way if these keys are lost or stolen, they won't be useful to an attacker for very long.

Code and Data Signing

Earlier in this paper we talked about making sure only signed code is running at boot. This goes for any updates as well. Furthermore, we need to make sure that configurations and maps are also signed. Basically, any data that is downloaded or used by the vehicle needs to be cryptographically verified before use. Any data or code that is not signed cannot be trusted by the vehicle or individual components.

Attack Surface Reduction

Code has flaws and nothing we can do is going to change that. The best approach to eliminate vulnerabilities is to try to reduce the attack surface as much as possible. To start with, no inbound connections over the Internet should be possible. Only outbound connections will be allowed. This means that the vehicle must initiate all connections to the back office.

Removing any functionality that does not serve a specific purpose is of utmost importance. For example, if there is no need for Bluetooth connectivity then it should be removed as this was an avenue for a remote attack in the past. The only surefire way to ensure that we've addressed all potential vulnerabilities in software is for the functionality not to exist. The authors of this paper contend this may be one of the most important aspects of securing vehicles but is not always possible for passenger cars intended for sale that require many convenience features.

Encryption of data at rest

As a company, we'd rather not lose our intellectual property if a vehicle or component is stolen. Remember, these vehicles will be out and about with attackers, we mean passengers, riding in them. It isn't impossible to think about one of these going missing.

While the self-driving code needs to be available at run time, we can at least ensure when powered down that it is encrypted. We will have the base operating system start up, authenticate to a service to get an encryption key and then use it to decrypt the proprietary software. This way, if we know a vehicle or component is missing, we can deny its attempt to get the decryption key for the software. Combining this with the status checks performed by the cars (discussed below), if a car is stolen we can power it down and leave it in a state where the car cannot decrypt the self-driving software, acquire its ephemeral security artifacts, or otherwise operate autonomously.

Segregation

As much as possible, we use network segregation to isolate connected components from the components that control the automobile. For example, the communications module should not have a direct connection to the CAN bus and should not be the same as the main compute module. Likewise, the tablets should be isolated as much as possible from more trusted components of the vehicle (see below for more details).

If there is a separate gateway module between the Ethernet and CAN networks, this module should face additional security scrutiny. Any communication to this gateway module should be verified so that only authorized modules can communicate with it to send CAN messages.

Much like your corporate network, you need to leverage segregation of critical pieces of infrastructure to reduce the likelihood of a successful attack on key components.

Ethernet Switch

We described the prolific use of Ethernet connected components earlier in this paper. These components are not so shockingly connected via Ethernet switches. The ethernet switch or switches control the flow of the Ethernet traffic.

First, to try to prevent unauthorized devices being plugged into the network, we utilize 802.1x when possible. While this is an ideal solution, in practice this is not likely to be fully implemented. This is because many of the third-party sensors and components will likely not have the capability to support 802.1x. You could imagine that most Ethernet-based cameras aren't too concerned with device authentication at that layer.

Again, we work with our vendors to push our security plans but also must prepare for scenarios where this isn't possible. Given that, we attempt to isolate components from each other as much as possible and only allow traffic from given ports / IP addresses to specific ports / IP addresses. So, for example, we ensure that a sensor can only talk to the main compute node and not the communications module at a network level.

Beyond the network security provided by the switch, we will utilize application level protections to ensure that only authorized devices are connecting to sensitive devices on the network.

Message Signing

The previous section discussed how network-based device authentication may not be possible for all the devices connected to a self-driving vehicle's Ethernet network. At this point, our best approach is to take the fight to the application level.

For TCP streams we utilize mutual TLS to verify endpoints. For UDP messages, we sign each message and the receiver verifies the signature (along with a nonce or similar to prevent replays). Our school of thought is that if you can't have uniformity with regards to device authentication, then protect the data at an application level.

It may also interest the read to know that many times it is not possible to sign every message sent across the ethernet network. Ethernet based ECUs tend to communicate with a large quantity of UDP frames while leveraging processors that were not designed to perform a cryptographic operation on every message. Therefore, we prioritize messages that we deem 'critical'. For example, we want to ensure that we sign every message that could directly affect steering, braking, or acceleration.

Tablets

We discussed before how the tablets used in the AVs are directly accessible by the attackers, damnit, we mean passengers. We do our best to defend these devices, including putting them in kiosk mode and physically blocking access to the buttons on the device. However, for all intents and purposes, we will assume these devices may be compromised.

Using the network, we will closely isolate exactly what they can access. Preferably this will be one or two very simple interfaces either on the main compute stack or the communications module. All other traffic from them will be blocked at the network level.

Remote Assistance

Until self-driving cars become better at driving in intense real-world conditions, it will be necessary for remote human operators to be prepared to help the vehicles if they find themselves in unusual or unknown conditions. In fact, according to California state law, this is currently a requirement [27].

This is probably the most important security-relevant feature on the vehicle. By design, it allows remote operators to directly control physical aspects of the vehicle. Of all the features on the vehicle, this is the one that would be easiest for an attacker to cause immediate damage.

When we consider securing this feature, the first requirement is that all requests for remote assistance should be initiated from the vehicle. This will eliminate the possibility of an attacker just sending them to vehicles from the Internet. Also, this outbound connection should authenticate the endpoint, either via TLS/SSL or some application level protocol.

On the infrastructure side, there should be very few computers capable of performing remote assistance. These computers should be in a secure, isolated room. These computers should be locked down, boot from read-only media, and not have unnecessary Internet access. The security of the vehicles rely on the security of these computers and so we do not want the computers surfing eBaum's World or reading email. Likewise, computers on the enterprise network are not allowed to perform remote assistance, only the ones that are in secure room. This restriction will be carried out at the enterprise network level.

Finally, despite all of that, we need to imagine a situation where the system becomes compromised, either by an attacker or an insider. So, we add a restriction to limit the aid that remote assistance can provide. We do not give full control of the vehicle to the remote operator. Instead of direct access to steering or brakes, we only allow an interface that allows the operator to interact with the decision making of the vehicle. Or perhaps the operator has full control, but the car is limited to some very slow, safe speed. In all, we limit this feature such that a compromise of remote assistance will not lead to massive harm or damage.

Fleet Management

It is important to build security into the management of the vehicles. For example, the vehicles should report their security state in periodic update messages. They should also inquire about anything they need to do from the fleet management server.

Another example would be the possibility to make a change in the fleet management infrastructure such that a car will remove itself from service, pull over, or return to the garage. Additionally, if a vehicle has failed to contact the fleet management server for a specified time, say 5 minutes, it should assume there is a problem and remove itself from service and return to the garage for inspection. If this fails, it should pull over and power off.

Threat Detection

As hard as we try, there is no reason to think that we are going to be building a perfectly secure vehicle. There is always a possibility that an attacker will exploit some vulnerability in the car. Because of this, we spend resources to not only try to prevent attacks but also to detect and react to attacks.

Logs of all security critical operations should be kept. They should be aggregated and uploaded periodically to the fleet management server for analysis. More importantly, there needs to be real time detection and reaction on vehicle.

The Ethernet network should be analyzed for anomalies. While, in general, the problem of generic intrusion detection is difficult and often leads to false positives, in this case it works well. That is because this is a network devoid of human users like we are used to in an enterprise environment. All the traffic is periodically generated from machine to machine. If any anomaly is detected, it should report to the fleet management server. For more severe or more likely attack situations, the vehicle may bring itself out of service, safely pull over, or power down.

Like Ethernet, the CAN network traffic should be observed in real time to identify anomalies. All the attacks outlined in the historical section could have been detected (and prevented) with even the most trivial CAN network intrusion detection software. Again, when anomalies occur, the system should take appropriate actions.

External Communication

We will limit the outbound communications possible and restrict it to only the fleet management and key servers. While this isn't a perfect solution and can be bypassed in some cases, it may make it more difficult for an attacker to perform operations.

For example, there is no reason for the communications module to be able to access arbitrary address and/or IPs on the open internet. Currently these vehicles are owned by a specific entity. Therefore, external communications can literally be limited to several end points.

Component Reduction

All unnecessary components should be removed from the vehicle. For example, if the radio/head unit is not needed due to the tablets being employed in the back seat, remove it. If the telematics system is not necessary since a communications module is present, remove it.

This not only removes attack surface, but equally important, it removes publicly available attack surface. It is much more likely an attacker will be able to construct an exploit against a vehicle component they have purchased and have access to than a proprietary component only found inside the self-driving vehicle.

Again, the authors of this paper consider component removal and reduction one the most effective methods for securing the underlying base vehicle. Remember, just because a device is useful in a traditional passenger vehicle does not mean it is useful for a self-driving vehicle. This is especially true when the self-driving vehicle is used in a ride-share situation.

Sensors

While attacking sensors is not our biggest concern, it is nonetheless something we need to think about. The best way to deal with these types of attacks are to use multiple sensors and multiple sources of input. This way, we have multiple sources of data from which to draw conclusions.

In situations where sensors or data disagree, it should be possible to choose a source of truth based on the quality of the data being received. Some sensors are more reliable and higher quality than others. Other times you may have a sensor where an object just “appeared” or “disappeared” while another sensor seems more stable. Regardless, it should be possible to determine which sensor is to be believed and acted upon.

Although this has been the focus of several recently published research projects [30] [31] we tend to prioritize defensive efforts on remote attack surfaces. While these attacks do not currently seem to directly affect us, that does not mean that it will not change in the future. In other words, sensor attacks are something to keep an eye on but not something to be overly concerned with in a vehicle with redundant sensors and multiple sources of data.

Conclusion

In this paper we attempt to explain what self-driving cars are and how they work. To many, it may be news that there are a multitude of different components, both in the AV stack and underlying base vehicle, that permit a vehicle to operate autonomously. We also found out there are, much like any project, many different problems and solutions to securing our driverless revolution. Luckily the authors of this paper have spent a fair amount of time attacking base vehicles and securing self-driving cars to figure out what works and what does not. Hopefully these subjects provide insight into how self-driving cars work, how they are different from passenger vehicles, the potential areas of risk, and how our teams are attempting to protect our vehicles and customers.

Thanks

- Ya boy NickDe. He’s OK at photoshop
- Our peeps at Uber, Didi, and Cruise
- Raffi: Thanks for making us believe there was large sums of money to be made in AV security. Lesson learned.

References

[1] https://www.sae.org/standards/content/j3016_201401/

[2] https://www.caranddriver.com/features/path-to-autonomy-self-driving-car-levels-0-to-5-explained-feature

[3] https://en.wikipedia.org/wiki/Waymo

[4] https://www.techradar.com/news/uber-self-driving-cars

[5] https://en.wikipedia.org/wiki/Cruise_Automation

[6] https://www.engadget.com/2018/03/02/arizona-no-safety-drivers-autonomous-vehicles/

[7] https://www.mercurynews.com/2018/02/26/self-driving-cars-with-no-in-vehicle-backup-driver-get-ok-for-california-public-roads/

[8] https://www.mercurynews.com/2018/06/01/waymos-self-driving-car-service-to-include-62000-minivans/

[9] https://www.theverge.com/2017/11/20/16678578/uber-volvo-xc90-suv-driverless-cars

[10] http://www.latimes.com/business/autos/la-fi-hy-ihs-automotive-average-age-car-20140609-story.html

[11] https://www.theatlantic.com/technology/archive/2018/04/uber-driver-app-revamp/557117/

[12] https://venturebeat.com/2017/06/04/self-driving-car-timeline-for-11-top-automakers/

[13] https://waymo.com/ontheroad/

[14] https://www.uber.com/cities/pittsburgh/self-driving-ubers/

[15] https://medium.com/self-driving-cars/miles-driven-677bda21b0f7

[16] https://www.fhwa.dot.gov/ohim/onh00/bar8.htm

[17] https://medium.com/kylevogt/why-testing-self-driving-cars-in-sf-is-challenging-but-necessary-77dbe8345927

[18] https://en.wikipedia.org/wiki/Lidar

[19] http://www.latimes.com/business/la-fi-hy-ouster-lidar-20171211-htmlstory.html

[20] https://www.youtube.com/watch?v=L-uxKP94vZA

[21] http://fortune.com/2017/09/02/researchers-show-how-simple-stickers-could-trick-self-driving-cars/

[22] https://www.inverse.com/article/34124-autonomous-cars-spoof-todd-humphreys

[23] http://www.autosec.org/pubs/cars-usenixsec2011.pdf

[24] http://illmatics.com/Remote%20Car%20Hacking.pdf

[25] https://electrek.co/2017/07/28/tesla-hack-keen-lab/

[26] https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf

[27] https://keenlab.tencent.com/en/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf

[28] https://pdfs.semanticscholar.org/e06f/ef73f5bad0489bb033f490d41a046f61878a.pdf

[29] http://www.autosec.org/pubs/woot-foster.pdf

[30] https://driverless.wonderhowto.com/news/scariest-lidar-vulnerability-weve-seen-yet-0178611/

[31] https://www.bleepingcomputer.com/news/security/you-can-trick-self-driving-cars-by-defacing-street-signs/