# Open-source Systems and Cybersecurity in Autonomous Vehicles

By Jake Goldenfein (Cornell Tech) and Mainack Mondal (IIT Kharagpur)

# Open-source Systems and Cybersecurity in Autonomous Vehicles

**By Jake Goldenfein (Cornell Tech) and Mainack Mondal (IIT Kharagpur)**

Cyber-attacks are ubiquitous, but in 2019 we've already experienced data breaches from automotive companies, ransomware attacks on city governments, and targeted smart phone exploits through Facebook owned WhatsApp. This combination of vehicle manufacturer, government, and tech giant is meaningful because these are also the entities involved in researching, building and operating autonomous transport systems. Those attacks resulted in leaking of personal data, paralyzing government systems, and more tragically with the WhatsApp attack, giving control of devices used by activists and journalists to states interested in detaining or silencing them. While some attacks may be inconvenient or costly, where secure communications are essential cyber-attacks are also incredibly dangerous. Nowhere is this clearer than in the world of autonomous vehicles, where a compromised vehicle, or fleet of vehicles, can be life-threatening.

One way to think about security vulnerabilities in autonomous vehicles is through the lens of open-source and closed-source (or proprietary) systems. Using open-source components introduces vulnerabilities into vehicle systems because malicious actors have access to the software code and can therefore manufacture exploits. At the same time, open-source approaches also facilitate the discovery of vulnerabilities by allowing a larger number of security researchers to analyze a system. Open-source thus brings trade-offs into security decisions. Those security trade-offs change according to the application and context of the system in question, including the consequences of an attack. An archetypal case of open vs closed software environments is smart phones, where Google Android and Apple iOS implement dramatically different security philosophies. While phones and cars are clearly different products, we suggest smart phones offer a useful analogy for thinking about vehicle cybersecurity because the role and relationships of tech companies and equipment manufacturers may ultimately follow a similar trajectory. In this Critical Reflections post, we discuss how these commercial and technical relationships and practices are likely to influence the uptake or rejection of open/closed-source components, and consequently how regulatory auditing can be a useful tool for striking a balance between *proactive* and *reactive* approaches to handle cybersecurity issues in autonomous vehicles.

Vehicle manufacturers have a tendency to avoid open-source software. Recent research suggests only 23% of automotive applications use open-source components, and almost all vehicle control system software (i.e. the computational systems that move a vehicle from point A to point B) is closed. But as the computational ecology of vehicles becomes more complex, there is a growing role for open-source. Much

of the software used in vehicles has become standardized and uniform. While vehicle software *applications* are useful for brand differentiation, operating systems and middleware are relatively invisible to consumers and unlikely to impact their buying decisions. Using open-source software also lowers production costs for vehicle manufacturers and decreases time to market. The automotive open-source movement thus looks to be growing, with a group of automotive companies called the Open Automotive Alliance pushing the role of Android in vehicles for the sake of open approaches to device integration, communications, and entertainment. An automotive Linux has also been developed to assist with open-source vehicle applications.

With existing vehicles, open-source components are mainly associated with entertainment (or infotainment) systems. As smart phones and tablet devices integrate into vehicle information environments, Android and Apple already exert significant influence. If tech companies become dominant providers of autonomous vehicle *control systems* (i.e. driving) however, questions of open or proprietary software must also be considered in that context. The commercial and technological future of autonomous vehicles is still contested, but the technical and economic power of tech companies working on autonomous vehicles may result in a platform ecosystem not unlike what we see with smart phones. That is, vehicle control systems or vehicle *platforms*, provided by tech companies, installed and integrated into OEM built vehicles. For instance, it would not be a surprise to find Waymo or Uber becoming the dominant provider of autonomous driving control software, with vehicle hardware still provided by a variety of automotive manufactures or marques. Volvo running its own driving software may in the end be like Nokia running the Symbian operating system – ultimately it makes more commercial sense to use Android. On the other hand, we may see vehicles with fully integrated hardware and software using entirely closed and proprietary software, more like Apple iPhones.

These different commercial and technical possibilities raise questions as to whether open or closed systems afford more desirable cybersecurity outcomes for autonomous vehicles. In particular, what advantages and disadvantages, pertaining to what stakeholders, are associated with proprietary versus open-source approaches? Put another way, considering there are always bugs in the code, how does the use of open-source or proprietary software affect the ability to interrogate and test vehicles for vulnerabilities? These questions are part of the broader issue of how autonomous transport companies should satisfy their responsibility for ensuring that vehicles are safe and secure.

As noted above, open systems are typically more vulnerable than closed systems. With more parties having access to security information about a particular system, there are more parties with access to the information necessary to discover a vulnerability and create a security breach. Open systems, however, also result in the detection of vulnerabilities more quickly. Open systems are subject to ongoing examination by security researchers across various institutions, for various purposes. In a closed security

context, while the vulnerabilities of the system may be less visible to malicious actors, detection of a security breach takes longer as there are fewer eyes on the code, with only a dedicated security team analyzing the system. It could thus be argued that open systems are, in general, more vulnerable, and consequently the security approach is more *reactive* (quickly patching security vulnerabilities as they are discovered), whereas closed systems are more secure, but the security approach is more *proactive* (trying to enumerate all attack surfaces and patch them internally even before the software is sent to end users).

The consequences of a cyber-attack on a vehicle could be dire. Discovering vulnerabilities and patching them, hopefully before the software ships, is imperative. The analysis above would then suggest that proactive rather than reactive approaches should then be preferred. Proactive approaches operate on the basis of 'security through obscurity' – by restricting relevant security information to a limited number of entities. But this approach is only superior in circumstances where a limited group has access to a system. As the scale of a system grows, the security calculus also changes. With more actors in the vehicle manufacture, operation, and control environment, such as equipment manufacturers, software companies, corporate vehicle control platforms, and connected controlling infrastructures, it becomes more difficult to limit the number of parties with access to security information. This makes it more difficult for an in-house security team to monitor all vulnerabilities. If tech platforms only provide the driving software, then facilitating interoperability with hardware manufacturers makes security by obscurity more difficult to maintain.

Another factor influencing the security calculus is the consequence of a breach. As the number of system users increases, the consequences of a cyber-attack might change. With a small number of affected users, for instance as with the WhatsApp attack, detection may only be prompted by suspicion of the specific target. When the number of targets is small, it may be that an open or closed approach offers no advantage to the amount of time taken to detect a vulnerability or sophisticated attack. Shipping proprietary software containing very few vulnerabilities that are unlikely to be detected before being exploited may thus be more acceptable if only a small number of users can be affected. When the target is system-level or, in vehicle terms, 'fleet-wide', this might change. The commercial and technical ecosystem for autonomous vehicles may eventually resemble that of smart phones, but cyber-attacks on smart phones are life-threatening for only a small number of users, which may not be the case for vehicles. Does the responsibility of manufacturers and operators then become one of constant vigilance, with as many analysts as possible looking at the code?

Some OEMs using third-party software may choose to monitor for vulnerabilities themselves, as companies like Volkswagen do. Alternatively, they may prefer to use open-source systems that enable and encourage the participation of security researchers more broadly. If animated by different business

models or normative imperatives, integrated hardware and software manufacturers may also prefer open-source approaches. Tesla, for instance, has made all their vehicle control security software open-source for the sake of enhancing security. They believe their software is already secure because of proactively patched security vulnerabilities found in-house, but can be made more secure by encouraging security researchers to explore it for vulnerabilities (that is, by *also* enabling a reactive approach to detect, for instance, zero-day vulnerabilities). Other types of combined systems that mirror, for instance, smart phones with open-source Android operating systems but also running proprietary Google Play Services packages for higher level functions are also imaginable.

However, considering the gravity of vehicle security breaches, the potential size of the attack surface (something else that mirrors smart phones), and the desirability of discovering vulnerabilities before they are exploited, it seems like the more eyes searching for vulnerabilities the better. That may indicate greater desirability of open-source components as analysis of closed systems, while not impossible, is far more difficult. A good example is the Volkswagen emissions scandal. This was a different concern in many ways, not the least being that the 'exploit' was an internally developed system designed to deceive rather than an external system designed to attack. Nonetheless, it shows how closed system vulnerabilities cannot be uncovered through general analysis, instead requiring measurements of inputs and outputs through a black box. In the Volkswagen case, the deception was only discovered through dedicated experiments provoked by discrepancies in vehicle emissions data, that required driving very long distances and measuring the levels of chemicals in vehicle emissions. Only after the emissions data was confirmed did Volkswagen come forward about the deceptive software code. It may be speculation, but it seems sensible to suggest that if the Volkswagen system was open, the problematic emissions algorithm would have been quickly discovered.

Fortunately, the security risks associated with autonomous vehicles and the high level of proprietary software in vehicle control systems have prompted a great deal of interest in auditing and testing tools. Several companies are now developing systems for auditing the security of electric control units (ECUs) and controller area networks (CANs) in vehicles, even in closed systems. The Toyota PASTA system for example, is an open-source vehicle security testing platform for security researchers that Toyota believes will help prepare for next-generation attacks on connected vehicles. But what is the best approach to building vehicle systems in the first place? How should relevant stakeholders fulfil their responsibility for making vulnerability detection possible? Does a system provider have a greater responsibility to ensure that the system is as secure as possible, or to ensure that a breach has the best chance of being detected prior to a vulnerability being exploited? If these two goals cannot be achieved simultaneously, which is to be preferred?

It may also be that open-source vs proprietary software is a red-herring for security. Open vs closed source software was not really an issue in the Chrysler internet-connected entertainment system being hacked in 2015, where security researchers managed to discover a vulnerability in a closed system. However, as companies further develop systems built for interoperability, use more software developed by third-parties, and possibly follow the tech company software platform and OEM hardware trajectory, the question of open and closed ecologies looks likely to remain important.

Carmakers do not need to choose open or proprietary entirely. There are decisions to be made up and down the automotive software stack. Some open-source and some proprietary software implementations may accommodate security needs as well as commercial imperatives. But proper testing and auditing may require opening up closed systems as much as possible. As Bruce Schneier noted in relation to the Volkswagen emissions scandal, addressing these problems requires both transparency and oversight. That is, access to the code and institutional structures capable of analyzing it. One way to achieve that is through auditing systems that give at least a limited number of authorized entities (i.e. a regulator) access to those closed systems, along with the means to test them extensively prior to public deployment. But considering the consequences of an attack on vehicles, even this might not be vigilant enough, and mechanisms for ongoing analysis by the largest number of researchers may also be necessary.


**Jake Goldenfein**

**Cornell Tech**

**jg2323@cornell.edu**

**Mainack Mondal**

**IIT Kharagpur**

**mainack@cse.iitkgp.ac.in**