Poster: Securing Appified Automated Vehicles With AVGUARD

Yunhan Jack Jia, Ding Zhao, Qi Alfred Chen, Z. Morley Mao

***Abstract*—The advancement in Autonomous Vehicle (AV) has created an enormous market for the development of self-driving functionalities, and has raised the question of how it will transform the traditional vehicle development process. One adventurous proposal is to open the AV platform to 3rd party developers, so that AV functionalities can be developed in a crowd-sourcing way, which is supposed to provide tangible benefits to both automakers and end users. Some pioneering companies in the autonomy industry have made the move to open the platform so that developers are allowed to test their code on the road.**

However, such openness brings serious security and safety issues by allowing untrusted code to run on the vehicle. In this paper, we introduce the concept of *Appified* AV platform that opens the development framework to 3rd party developers. To further address the safety challenges, we propose an enhanced appified AV design schema called AVGUARD, that focuses mainly on mitigating the threats brought by untrusted code, leveraging theory in the vehicle evaluation field, and also program analysis techniques in the cybersecurity area. Our study provides guidelines and suggested practice for the future design of open AV platforms.

I. INTRODUCTION

Appified platforms, where software applications (apps) are developed in a crowd-sourcing manner by 3rd party developers and distributed through the app market, have achieved astonishing success in the IT field in the last decade, due to the benefits brought by its open nature. Recent years have seen the *appification* of many other software platforms such as smart home [7] and even network switches [3]. And the success that has been achieved through the crowd-sourcing app development on these platforms has raised concerns in both industry and academia about whether the autonomous vehicle (AV) – the next much-anticipated software platform, will become appified.

Supporting crowd-sourcing app development on AV is supposed to benefit both automakers and customers. From the end-user's perspective, it will bring vast variety of apps into the market that enriches user's choices and provides them with the flexibility to personalize their driving and in-vehicle experience by only installing/uninstalling apps. For the automakers, the appification will transform the development of some AV functionalities from outsourcing to crowd-sourcing, which not only reduces the cost, but also promotes the improvement of app quality. The AV app market will also create the ecosystem where multiple functionalities can be cooperated to provide more intelligence and convenience. Some organizations have already started rolling out the autonomous vehicle with open development support. For example, in the industry, vehicle middleware platforms that open massive vehicle functionalities including steering wheel and brake to the developers have been built (e.g., Ford OpenXC [5], PolySync [6]). While in the academia, University of Michigan (U-M) starts to offer open-access to their testing AV equipped with sensors including lidar, radar, and cameras, so that researchers can rapidly

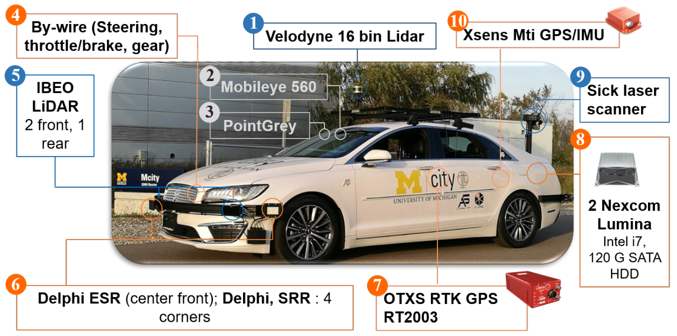
4 By-wire (Steering, throttle/brake, gear)
1 Velodyne 16 bin Lidar
10 Xsens Mti GPS/IMU
5 IBEO LiDAR 2 front, 1 rear
2 Mobileye 560
3 PointGrey
9 Sick laser scanner
8 2 Nexcom Lumina Intel i7, 120 G SATA HDD
6 Delphi ESR (center front); Delphi, SRR : 4 corners
7 OTXS RTK GPS RT2003

Fig. 1: The open-access automated vehicle of at the University of Michigan

test their self-driving or connected-vehicle technologies [8] (Fig. 1). Although the idea of "AV app store" hasn't been discussed by any of these initiators, to better realize the benefit of crowd-sourcing development, there are reasons to believe that appified AV platform will become reality in the near future.

However, every coin has two sides. To identify key challenges and issues for the appified AV, we compare aspects of traditional vehicle platform with appified AV platform. From the adoption of Controller Area Network (CAN) bus in the 1980s, the modern vehicles now have over 70 electronic control units (ECUs) for various subsystems [9], including critical control systems and also peripheral infotainment systems all communicate using the CAN bus. As the automakers usually outsource the development of those peripheral functionalities to reduce the development cost, security and safety problems are raised since these software developed by 3rd party that have access to CAN bus, can potentially be exploited to tamper the safety of the vehicle [1], [10]. Appified AV platform provides software abstraction for the physical CAN bus. Self-driving functionalities are developed as apps, and their interactions with the hardware actuators are proxied by a vehicle operating system that also act as the middleware. Since the appified platform opens full-fledged self-driving functionalities to the 3rd party developers, it could potentially introduce greater safety risk. For example, flaws in the proportional control algorithms of a cruise control app may put the vehicle in a situation where a collision becomes inevitable. Recent accidental records [4] of self-driving cars suggested that deficiencies in the AV software are inevitable due to the complexity of physical environment, so we believe that the vulnerable apps will remain a persistent threat to the AV industry. In addition, apps may also be developed for malicious purpose to tamper with the user's safety with embedded malicious logic [2].

Fortunately, the open nature of appified AV platform also provides us with the opportunities to build and deploy defense against above threats at the vehicle OS level. Thus sanity checks can be performed on the control messages from AV apps to mitigate potential risks raised by apps and guard the

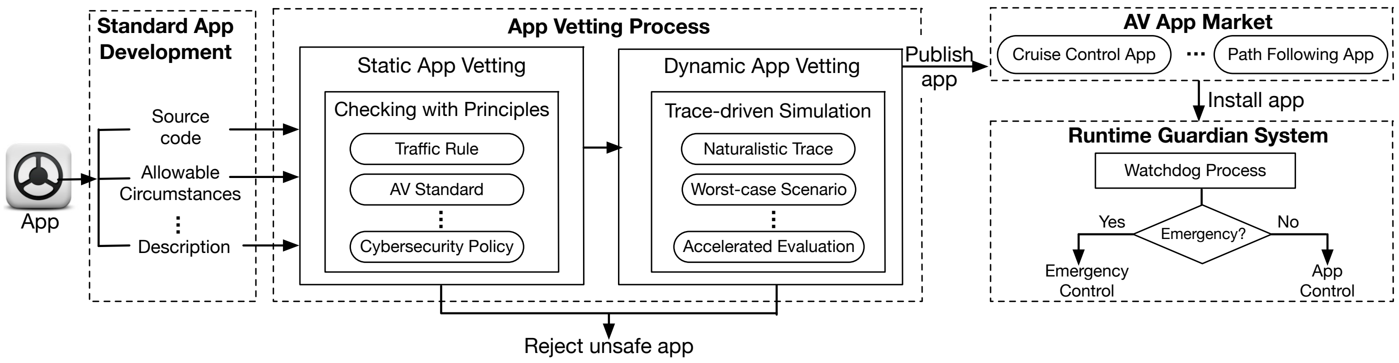

Fig. 2: System overview

safety of the vehicle. However, due to the extremely high safety requirement of vehicle platform compared with other appified platforms, dangerous apps that include fatal flaws or malicious logic are better to be detected even before they are installed on the vehicle. And the best practices that have already achieved success on appified platforms are building market-level app vetting to prevent problematic apps from entering the market. We draw lessons from the state-of-the-art security practices in the IT field, and leverage the open AV platform to propose our solution to the previously mentioned problem.

In this paper, we propose the first appified AV design scheme that focuses mainly on addressing the security and safety concerns raised by the crowd-sourcing app development. We identify key principles that need to be enforced in the context of various self-driving scenarios, and propose AVGUARD, which is a platform enhancement for open AV that incorporates both offline app vetting that performs early detection of unsafe apps, and also runtime safe guardian that provides baseline safety guarantee. We present the detailed design of the app vetting process, which adapts the program analysis techniques from cybersecurity fields to perform static app analysis, and leverages the recent advancement in the Accelerated Evaluation field to dynamically estimate the risks of apps using naturalistic trace. The vetting also enforces many principles proposed recently in the U.S. Department of Transportation's Federal Automated Vehicles Policy [11], such as requirements for the fall back approach and specification of allowable circumstances. We also advocate a runtime on-vehicle watchdog implemented as a privileged process on the vehicle OS that monitors the surrounding environment to avoid potential collisions caused by apps. Our study sheds light on the opportunity of realizing the benefits of crowd-sourcing development on self-driving vehicles without risking the safety of the platform, and provides guidelines for future research along this line.

II. THE AVGUARD APPROACH

In this section, we first describe our problem scope, and discuss the essential steps to mitigate them. We then introduce the major components of the AVGUARD approach, and present a roadmap to the detailed design of each component in remaining sections.

As mentioned earlier, the major safety issue of appified AV comes from the untrusted 3rd party app. In this paper, we mainly focus on two types of untrusted apps: *vulnerable* app and *malicious* app, which have been the persistent pain points for other appified platforms.

Shown in Figure 2, we advocate the AVGUARD approach, which consists of four components: (1) A standardized app development process that specifies the required information regarding of the properties of the self-driving app (e.g., source code, allowable circumstances) to be provided by the developer, in order to make the app functionality expressive and verifiable; (2) A static app vetting framework that checks app logic with a set of safety principles based on static program analysis techniques; (3) A dynamic risk evaluation system that test app in simulated environment against a set of benchmark and naturalistic driving traces to quantify the potential risk, and only apps that pass both static and dynamic vetting will be allowed to the market. The official app market should be the only authenticate source for users to download AV apps. It can be enforced by requiring digital signature for each app to be installed, and only app binaries that are signed by the official market are allowed to run on the consumer's vehicle. (4) A runtime guardian system that performs access control for AV apps running on the vehicle based on the environmental context, and ensures the baseline safety of the vehicle under various physical scenarios. We will discuss the detail about the implementation of AVGUARD in our poster

REFERENCES

[1] After Jeep Hack, Chrysler Recalls 1.4M Vehicles for Bug Fix. https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/.
[2] Car hacking: how big is the threat to self-driving cars? http://fortune.com/2014/10/07/car-hacking-how-big-is-the-data-threat-to-self-driving-cars/.
[3] HP SDN App Store. https://www.hpe.com/us/en/networking/applications.html.
[4] Inside the Self-Driving Tesla Fatal Accident. http://www.nytimes.com/interactive/2016/07/01/business/inside-tesla-accident.html.
[5] OpenXC Platform. http://openxcplatform.com/.
[6] PolySync. https://polysync.io/.
[7] Samsung SmartThings. https://www.smartthings.com/.
[8] U-M Open-access Self-driving Vehicle. http://ns.umich.edu/new/releases/24351-u-m-offers-open-access-automated-cars-to-advance-driverless-research.
[9] A. Albert. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded World*, 2004:235–252, 2004.
[10] S. Mazloom, M. Rezaeirad, A. Hunter, and D. McCoy. A security analysis of an in vehicle infotainment and app platform. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. USENIX Association.
[11] U. D. of Transportation. Federal automated vehicles policy. September 2016.