Building an Nmap for Your Car

SEKAR KULANDAIVEL, TUSHAR GOYAL, ARNAV KUMAR AGRAWAL, AND VYAS SEKAR

Sekar Kulandaivel is a PhD candidate in electrical and computer engineering at Carnegie Mellon University. Sekar's interests lie in automotive network and systems security with a focus on practical solutions. skulanda@andrew.cmu.edu

Tushar Goyal currently works at Microsoft Search. His primary focus is in systems, computer architecture, and security. He is a graduate of Carnegie Mellon University. tgoyal1@alumni.cmu.edu

Arnav Kumar Agrawal currently works at Microsoft on the Xbox Live Graph Team. His focus is primarily on large-scale distributed systems and security. He is a graduate of Carnegie Mellon University. akagrawa@alumni.cmu.edu

Vyas Sekar is the Angel Jordan Early Career Chair Associate Professor in the ECE Department at Carnegie Mellon University, with a courtesy appointment in the Computer Science Department. His research is in the area of networking, security, and systems. His work has received best paper awards at ACM SIGCOMM, ACM CoNext, and ACM Multimedia, and the NSA Science of Security prize. vsekar@andrew.cmu.edu

The network inside a modern car is no longer static; modern in-vehicle networks grow more complex and can change over time. We have developed CANvas, a network mapping tool that identifies what devices in your car communicate on the network and how messages are exchanged between them. CANvas helped us identify an unknown device in a modified 2009 Toyota Prius and pinpoint potentially vulnerable devices in a 2017 Ford Focus.

In 2015, two security researchers, Charlie Miller and Chris Valasek, remotely hacked into a Jeep Cherokee to demonstrate the potential impact of a hack against a modern car [6]. They accomplished their goal by compromising one of the vehicle's computers, known as Electronic Control Units (ECUs), and then used this compromised ECU to communicate with other ECUs in the car over the vehicle's Controller Area Network (CAN) bus. To figure out what devices were a part of this network, they had to physically disconnect components, which is a painstaking process even for analyzing a single car. As vehicles continue to integrate more electronics and contain increasingly complicated networks, we need a tool that can produce a map of the car's network to help us keep our vehicles protected without having to take apart a car.

The obvious solution here is to ask the automaker for the network map of the cars that we own. However, this is highly proprietary information that automakers are not willing to give out even to mechanics and researchers. Even if we could obtain this network map from an automaker, we face a new challenge in today's world: these networks are no longer static. Once a car leaves the factory, the automaker loses control of what devices are connected to the network and how they interact with each other. We now look at a few scenarios where you can expect your car's network to change.

Imagine if you took your car to a potentially untrustworthy mechanic. Under the guise of a repair, this mechanic could add a new device to your car's network without your permission. The number of components in your car that communicate on the network is getting larger and larger; you can even buy headlights that talk on the CAN bus. Consider another scenario: imagine if you had a traction control ECU replaced by even a trusted mechanic, but the ECU was counterfeit and was not programmed to send the correct messages. In this case, you may only discover the ECU was counterfeit *after* you needed your traction control.

Perhaps you want to replace the radio in your modern car, so you decide to buy a replacement radio from Amazon or eBay. As observed in some modern vehicles, this radio could be connected to your vehicle's CAN bus, and a malicious seller could program that radio to transmit new or different information onto the network. We also see that automakers are now considering pay-as-you-go services where your car comes pre-installed with hardware, like a turbocharger or high-output batteries. These automakers envision customers paying to activate these features, which will introduce new communication between the computers in a car's network. In defending our vehicles against attacks, the ability to differentiate between expected traffic from these services versus malicious traffic from an attack could prove useful.

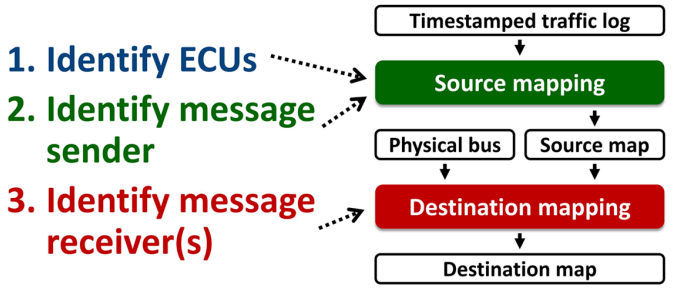

Figure 1: Design overview of the CANvas network mapper.

Now consider recent work that demonstrated a new type of attack that can shut down any ECU in your car by compromising just a single ECU in your vehicle [3]. A requirement for this attack is that the compromised ECU must receive messages from its victim. Since shutting down an ECU while the car is in motion could be potentially catastrophic, this type of attack has a real and significant impact. Imagine if that radio you bought from an online seller was programmed to launch the shutdown attack against your vehicle's engine ECU while it was in motion.

It is clear from these scenarios that the network inside a modern car can change. As automakers produce new models of a car each year and even provide over-the-air update capabilities, the frequency of producing updated network maps for a vehicle will quickly increase. To ensure that security researchers and vehicle owners can keep up with these networks and understand what goes on in our cars, we need to build a network mapping tool that is both practical and accessible.

Building a Practical Mapper

As a first stepping stone to future automotive security work, we developed CANvas, a fast and inexpensive automotive network mapping tool for a vehicle's CAN bus [5]. For less than 50 dollars of hardware and under 30 minutes, CANvas produces a network map that can identify the ECUs in a car as well as where messages originate from and which ECUs receive each message. We focus on building a practical tool that works on real modern vehicles.

Current approaches for mapping a vehicle's network require physically taking apart the car. Instead of requiring our users to go through this extensive process, we aim to build a mapper that satisfies a few practical goals. Since we require the vehicle to be on and running, our mapper should not take hours of time to complete. It should also be inexpensive and not require the use of an oscilloscope or logic analyzer. Anything we do to the network should leave no permanent damage, and the mapper should simply plug into the the On-Board Diagnostics (OBD-II) port of your car, which, starting in 1996, all vehicles manufactured or sold in the United States are required to have. Our users should not expect to cut into the network of their car or worry that their car will have permanently lit malfunction indicator lights.

We define three main outputs for our network mapper: (1) a list of all active ECUs in the vehicle, (2) the transmitting ECU for each unique CAN message, and (3) the set of receiving ECUs for each unique CAN message. As seen in Figure 1, we combine our first two outputs into a module called *source mapping* and our third output into a module called *destination mapping*. The source-mapping module takes in a timestamped traffic log as input and produces a source map that identifies the source ECU of each message. Then the destination mapping module uses the source map and access to the physical CAN bus of a running vehicle to produce a destination map that identifies which ECUs receive each CAN message.

A significant challenge in designing this tool is the broadcast nature of the CAN protocol, as network messages contain no information about their sender or recipients. To address this challenge, we repurpose insights that were previously proposed for an intrusion detection system for the CAN bus [4] and a shutdown attack against CAN-enabled ECUs [3]. Since the techniques used in these works had limitations when applied to the mapping problem, we develop two approaches for our source and destination mapping modules, respectively: a pairwise clock offset tracking algorithm that identifies transmitting ECUs and a forced ECU isolation technique that identifies receiving ECUs. In designing these techniques, we focus on the practical challenges that we face when mapping real vehicles.

In the rest of this article, we detail our adventures in mapping our two main test vehicles and our vision for how the CANvas mapper can be used for future research. This article is based on a conference paper that appeared at the 2019 USENIX Security Symposium, which presents our design for CANvas, our experiments, and other considerations for mapping in detail [5].

The Hidden ECU in a Toyota Prius

We managed to acquire a hand-me-down 2009 Toyota Prius from a different department at the university. This Prius was converted into an all-electric vehicle with a lithium-ion battery installed in its trunk and became our primary ground truth for the mapper. Unfortunately, the only method at the time for obtaining the ground truth was to physically dismantle the car and gain direct access to all of the ECUs in the Prius as seen in Figure 2. Going through this process for just a single car made us thankful for the prospect of a network mapping tool.

To figure out how to take apart the car and find each of the ECUs in the car, we did what the mechanics at the dealership would do: we went to the service website for Toyota. After paying a small subscription fee, we followed the removal instructions for each ECU and found that this Prius contained eight ECUs. With this direct access, we could splice into each ECU's connector and determine what messages it sent and received. This information

Figure 2: Physically tearing apart any vehicle is exhausting; imagine doing this every time we need an updated network map.

served as our ground truth for verifying our output from the CANvas network mapper.

For identifying the source of each network message, one of the key assumptions we make is the periodicity of messages on the CAN bus. From prior work [4], we knew that the majority of messages on the network should be periodic in nature. When we connected to the network, we found that all messages at first *seemed* to be periodic. However, when we measured the average period and the deviation of the period, we found several examples of messages that were "almost" periodic. Some messages stopped transmitting for seconds at a time, some seemed to miss their deadlines and re-transmit at the next period, and others seemed to have two different periods that resulted in overlapping messages.

Where prior work only investigated purely periodic messages, we found that these "almost" periodic messages are expected in real vehicles. We discussed this finding with sources from the automotive industry, and they confirmed that CAN message transmissions can be complex and that there are many circumstances that affect how an ECU transmits its messages. To address these special cases, we designed CANvas to detect these instances and employed a slightly modified approach for identifying the source of these messages as detailed in our paper [5]. At its core, these messages have some notion of periodicity, which allows us to use our method of mapping messages to their source ECUs.

With the source-mapping component of our network mapper completed, we plugged our tool into the Prius's diagnostic port and expected to run CANvas and find eight ECUs as discovered in our ground truth; much to our surprise, we found *nine* ECUs. After scratching our heads, we decided to unplug all eight of the expected ECUs and see if we still saw traffic on the network. We still saw three messages and detected a single transmitting ECU with no receivers. We confirmed this finding with other online sources that did not see these three additional messages, so we knew that something must have changed in the car's network.

At this point, we knew something must be different with this car, especially when other cars of the same model year are missing these three messages on their networks. Looking into this Prius's history, we found that a new ECU was installed as part of the all-electric modification done almost a decade ago. Without this network mapping tool, we would have never known about this hidden ECU. Thankfully, this ECU was not malicious and was simply included as a modification from the previous department. But when we consider that this could have been a malicious device, the importance of having a network mapper becomes clear.

The Vulnerable ECUs of a Ford Focus

Our finding on the Prius was quite surprising, but we also wanted to test our network mapper on a newer car. We managed to find a salvaged 2017 Ford Focus with minor flooding but all of its electronics completely intact. As seen in Figure 3, we physically dismantled this salvaged car to obtain the ground truth, and we tested our network mapper on it. For this vehicle, we used the Ford service website for instructions on ECU removal, and we identified nine total ECUs.

To identify the destinations of each network message, we borrowed an insight from prior work [3] on shutdown attacks against ECUs. Due to limitations in this work, we implemented a different shutdown technique that permits CANvas to analyze each ECU one by one. Unfortunately, we came across an interesting challenge; we found that some ECUs automatically recovered or did not even shut down. For the purposes of network mapping, we had to make sure that some ECUs remained in the shutdown state as detailed in our paper [5]. Upon closer inspection, we found that these ECUs do remain in a shutdown state for some time. With additional add-on techniques, we can suppress an ECU and keep it from coming back online, allowing us to perform our destination mapping.

When mapping a modern car, we expected that automakers would limit what messages can be received by each ECU since all ECUs do not need to communicate with each other. For example, we would not expect a radio ECU in the Focus to receive messages from ECUs related to the powertrain. Using our network map of the Focus, we found that all of its ECUs were capable of receiving all messages on the network. This means that any ECU compromised by some future remote attack could be used to launch attacks on other ECUs, including the safety-critical powertrain ECUs. One might think that this type of attack has not been significant or realistic enough for automakers to implement restrictions on what messages an ECU can receive.

Figure 3: We found that all ECUs in this Focus could potentially launch a recent shutdown attack.

However, we find that the industry acknowledges this potential and is taking measures to defend against this. For purposes of restricting the messages that reach an ECU, Next eXPerience (NXP) is working on a new CAN transceiver, the NXP TJA115x Secure CAN Transceiver family [2]. If automakers implement these types of hardware-level filters on what messages an ECU could receive, a tool like the CANvas mapper could help ensure that these filter settings are correct. Until then, we have found that even a modern car like the Focus has no filter on the messages its ECUs receive.

Next Steps from Network Mapping

With the ability to network map a car, we envision several extensions to the mapper and alternative tools that could benefit from CANvas. A vehicle's network map could benefit from richer functionality, such as identifying the function of an ECU (transmission ECU, engine ECU, etc.), identifying gateway ECUs that potentially bridge multiple CAN buses, and identifying message filters implemented in software. We also envision extending this work into other automotive protocols, including Automotive Ethernet and Local Interconnect Network (LIN), to provide a broader map of a car's complete network and not just its CAN bus.

In previous conversations with industry, we asked how this network mapping tool could prove useful even though industry has the original network map. Besides being used to detect unauthorized changes to the network, the network mapper could be used to verify the state of the network. Over time, parts of the ECUs and their network could fail, such as diodes and wiring. These failures could change the output of the network map and indicate issues with the network. Instead of being used directly for security, CANvas could also serve as a network validation tool for your mechanic to use when changing the configuration of your vehicle's CAN bus.

We have also made interesting discoveries when we look at how messages are transmitted. When running our network mapper on a vehicle, we see changes in the contents of a message based on what ECUs are shut down during our destination mapping. We could use this new information to potentially infer details that are only found in the automaker's proprietary network map. Since we can know which ECU sends each message, we could implement better techniques to reverse engineer the message contents by correlating changes between an ECU's source messages. Additionally, when designing an attack, a network map could tell us what messages need to be replicated if we ever wish to pursue a masquerade attack. CANvas serves as a building block for future security research, and we have made it publicly available with the hope that the community will help us expand its capabilities in the future [1].

References

[1] CANvas network mapper: https://github.com/sekarkulandaivel/canvas.

[2] NXP TJA115x Secure CAN Transceiver family: https://www.nxp.com/docs/en/fact-sheet/SECURCANTRLFUS.pdf.

[3] K. Cho and K. G. Shin, "Error Handling of In-Vehicle Networks Makes Them Vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*, ACM, 2016, pp. 1044–1055: https://dl.acm.org/citation.cfm?id=2978302.

[4] K. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security '16)*, 2016, pp. 911–927: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_cho.pdf.

[5] S. Kulandaivel, T. Goyal, A. Kumar, and V. Sekar, "CANvas: Fast and Inexpensive Automotive Network Mapping," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security '19)*, 2019, pp. 389–405: https://www.usenix.org/conference/usenixsecurity19/presentation/kulandaivel.

[6] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Black Hat USA, 2015: http://illmatics.com/carhacking.html.

12th USENIX Workshop on Cyber Security Experimentation and Test (CSET '19)

PETER A. H. PETERSON AND ROB G. JANSEN

Peter A. H. Peterson is an Assistant Professor of Computer Science at the University of Minnesota, Duluth, where he teaches and researches operating systems and security, with a focus on R&D to make security education more effective and accessible. He received his PhD from the University of California, Los Angeles, for work on "adaptive compression"—systems that make compression decisions dynamically to improve efficiency. pahp@d.umn.edu

Dr. Rob Jansen is a Computer Scientist and a Jerome and Isabella Karle Distinguished Scholar Fellow at the US Naval Research Laboratory with research expertise in the areas of computer security and privacy, distributed systems, and parallel and distributed simulation. His research results have been published in the top peer-reviewed computer security and privacy conferences and workshops, and his work demonstrates an ability not only to develop and apply theoretical concepts, but also to build, evaluate, deploy, and measure real-world systems. When not in the lab, Rob enjoys jogging around the National Mall and through the streets of DC. rob.g.jansen@nrl.navy.mil

On Monday, August 12, 2019, 55 attendees joined us for the 12th USENIX Workshop on Cyber Security Experimentation and Test (CSET '19) in Santa Clara, California. CSET, one of the USENIX Security Symposium's co-located workshops, welcomes work in the broad categories of "cybersecurity evaluation, experimentation, measurement, metrics, data, simulations, and testbeds"—that is, research about research tools, data, and methods. The purpose of this article is to share our experience chairing CSET '19 and to highlight this year's papers.

Changes to the CSET PC

We made some experimental changes to the call for papers (CFP) and program committee (PC) this year, and we wanted to share them in the hope that they might be useful for other organizers. One of our main goals was to increase the community reach of the PC and the submission count, while reducing the PC review burden. To do this, we doubled the size of the PC to 46, inviting both established CSET community members and new people, including both junior and senior researchers. We also explicitly invited broad interpretations of the topics list. Additionally, we solicited a variety of paper lengths and types: traditional research papers, position papers, experience papers, preliminary work, and extended work. These could be long papers (eight pages), short papers (four pages), or extended abstracts (two-page talk proposals).

We explicitly invited preliminary work papers because CSET is a workshop; we wanted to encourage the lively discussion of new ideas, even if they were not fully developed. "Extended work" papers were meant to be expansions of security experimentation results, approaches, or tools developed in the course of other research (e.g., papers published at USENIX Security or elsewhere). Our rationale for soliciting these papers was that all security research requires an experimental approach; this often includes the development of tools, data, or knowledge that could be useful to the community. Unfortunately, these details are often drastically reduced in published papers due to space constraints. This cut material is often squarely in CSET's bailiwick, and we hoped that papers like this would be relatively easy for authors to prepare, interesting for attendees to discuss, and of service to the research community.

We are also happy to report that women comprised 46% of the CSET '19 PC, up from the recent peak of 32% in 2015. Women are in high demand and may already be committed to a full slate of PCs; to find the 21 women who were able to join the PC this year, we invited approximately double that number. Our takeaway was that it is absolutely possible to improve gender representation on PCs, but until the underlying diversity in our field improves, doing so may take a little time and effort.

Overall, our changes seemed to work well; we received 61 submissions, more than doubling 2018's submission count of 27. Each reviewer had approximately four papers to review. (We had wanted to limit each PC member to three reviews, but the volume of submissions precluded that.) Ultimately, we accepted 19 papers (31%). For more information about our process and statistics this year, please see our slides on the workshop site.

Sessions and Presentations

The 19 accepted papers this year were arranged into five sessions. The first session was "Cyberphysical and Embedded Testbeds and Techniques," chaired by Eric Eide (University of Utah). First, Paul Pfister (Iowa State University) presented a cyber-physical system (CPS) extension to ISEAGE, an event simulator used for cyberdefense competitions that included a physical model of a city, complete with LEDs representing system status. Next, Woomyo Lee (The Affiliated Institute of ETRI) presented a system for automatic generation of CPS research data about a power plant featuring a GE turbine, an Emerson boiler, and a FESTO water treatment system. After this, Sam Crow (UC San Diego) told us about Triton, a configurable testbed for avionics security research. Triton is, in the words of Crow, "real hardware from a real airplane that thinks it's running on an actual airplane in flight." Finally, we heard from Zachary Estrada (Rose-Hulman Institute of Technology) about CAERUS, a framework that is able to identify, through automated testing, timing sensitivities of undocumented embedded systems that can interact negatively with add-on security components.

Elissa Redmiles (Microsoft Research/Princeton University) chaired our "Data and Metrics" session. Michael Brown (Georgia Institute of Technology) described how debloaters can improve security by reducing the number of ROP gadgets through eliminating unimportant code, but also how they can accidentally introduce new high-quality gadgets. Instead of focusing on gadget count as the key metric, Brown proposes metrics based on gadget quality. Next, Aniqua Baset (University of Utah) discussed SecPrivMeta, an interactive website (secprivmeta.net) that provides visualizations of topic modeling on 36 years of security and privacy publications. After this, Josiah Dykstra (US Department of Defense) described how the NSA uses the Innovation Corps (I-Corps) methodology to improve the sharing of Cyber Threat Intelligence (CTI). Last, Jim Alves-Foss (University of Idaho) gave an entertaining talk containing a variety of cautionary tales of problematic data analysis and experimentation to admonish the community to use care and best practices in research.

"Usability, Effects, and Impacts" was chaired by Heather Crawford (Florida Institute of Technology). Zane Ma (University of Illinois at Urbana-Champaign) gave the first talk, which was about the effect of TLS and browser presentation on the success of phishing attacks in an A/B test on 266 users. Next, Victor Le Pochat (KU Leuven) described the design and evaluation of Tranco, a "top sites" ranking that aggregates Alexa, Majestic, Quantcast, and Umbrella, to create a stable and robust list for use by researchers. Third, Xiaodong Yu (Virginia Tech) presented work investigating how seven cache configuration parameters affected timing-based side-channel attacks; their talk included suggestions for improving security while minimizing performance impact. Last, Ildiko Pete (University of Cambridge) presented preliminary results from the Cambridge Cybercrime Center's analysis of usability issues with the data sets they share.

David Balenson (SRI International) chaired "Problems and Approaches," which began with Qiao Kang's (Rice University) presentation of their work automating the detection of attacks against the data planes of programmable routers. Our next presenter, Fatima Anwar (UCLA, now University of Massachusetts at Amherst) described how the timing capabilities of trusted execution environments (TEEs) can be vulnerable to timing attacks in realistic scenarios, and provided requirements for securing time facilities in these environments. Next, Sri Shaila G (University of California, Riverside) presented results of a study using IDAPro to reverse engineer the binaries of real-world IoT malware samples as compiled with various options, finding that, while unstripped binaries are amenable to analysis, performance on stripped binaries is generally poor. Last, Jonathan Crussell (Sandia National Laboratories) talked about their analysis of 10,000 experiments comparing differences between virtual and physical testbeds for research.

The final session of the day was "Testbeds and Frameworks," chaired by Jelena Mirkovic (University of Southern California Information Sciences Institute). Aditya Ashok (Pacific Northwest National Laboratory) described PACiFiC, a sufficiently realistic campus microgrid testbed model to allow a phish-to-blackout attack simulation. Second, Russell Van Dam (Sandia National Laboratories) presented Proteus, an emulation framework that supports the analysis of a wide variety of peer-to-peer distributed ledger technologies against different types of automated scenarios. Finally, Ryan Goodfellow (Information Sciences Institute) described the DComp Testbed, an open-source testbed using EVPN routing, a set of independently useful tools, and featuring a high level of abstraction and isolation.

For more detail, please see the workshop program online at www.usenix.org/cset19/program.

We would like to offer our sincere thanks to the fantastic USENIX staff, CSET's program and steering committees, authors, session chairs, shepherds, presenters, and attendees. The 13th CSET will once again be co-located with USENIX Security 2020 in Boston, with papers due in spring 2020. If you're interested in research around security experimentation, please consider submitting to and/or attending CSET next year!