Off is Not Off: On the Security of Parked Vehicles

Kyong-Tak Cho
University of Michigan, Ann Arbor
ktcho@umich.edu

Kang Shin
University of Michigan, Ann Arbor
kgshin@umich.edu

Yu Seung Kim
Ford Motor Company
ykim41@ford.com

Byung-Ho Cha
Ford Motor Company
bcha@ford.com

***Abstract*—While various ways of attacking and thus controlling the vehicle have been demonstrated, all these attacks were shown to be feasible and effective only while the vehicle is running, i.e., ignition is on. In this paper, we invalidate the conventional belief that remote vehicle attacks are feasible and hence their defenses are required only when the vehicle's ignition is on. We first analyze how operation (e.g., normal, sleep, listen) modes of electronic control units (ECUs) are defined in various in-vehicle network standards and how they are implemented in real vehicles. From this analysis, we discover that an adversary can exploit the wake-up function of in-vehicle networks—which was originally designed for enhanced user experience/convenience (e.g., remote diagnosis, remote temperature control)—as an *attack vector*. Ironically, the battery-saving feature in in-vehicle networks makes it easier for an attacker to wake up ECUs and, therefore, mount *Battery-Drain* (BD) or *Denial-of-Body-control* (DoB), and *Unattended Control* (UC). In particular, we show that the adversary mounting the BD attack can completely drain the vehicle battery within an hour in the worst case, the attack mounting the DoB attack can disable the communications between the vehicle and its key-fob by shutting down the associated ECU, thus immobilizing the vehicle, and the adversary can physically access a vehicle by controlling the door and/or trunk locks via the UC attack.**

I. Introduction

Software-defined ECUs and wireless connectivity of modern vehicles have enabled new vehicle applications/functions, such as remote Over-The-Air (OTA) update and diagnostics, Advanced Driver Assistant System (ADAS), and Passive Keyless Entry and Start (PKES). Meanwhile, they have introduced more attack surfaces and thus vulnerabilities that an adversary can exploit remotely for controlling vehicle functions [12–14, 28]. Researchers have demonstrated how vulnerabilities in vehicles can be exploited remotely to compromise an ECU and access the in-vehicle network [13, 27]. Via remotely compromised ECUs, researchers have shown to be able to control vehicle maneuvers or even shut down a vehicle via packet injection in the in-vehicle network [10, 14, 22, 25].

While various ways of attacking vehicles using vulnerabilities have been demonstrated, all of them are shown to be feasible and effective, *only when the vehicle power is on and thus ECUs are turned on*. That is, attacks are commonly believed feasible and hence their defenses are necessary only when the vehicle is running. This has led to the lack of understanding of, and attention to what an adversary can do while the vehicle's ignition is off. In this paper, we show such a common belief does not hold by demonstrating three attacks — *Battery-Drain* (BD), *Denial-of-Body-control* (DoB), and *Unattended Control* (UC) — through which an adversary can disable, abuse, and/or access parked vehicles with the ignition off. Ironically, the main reason for this feasibility is the "wake-up functions" — which are intended to enhance the driver's convenience — let the adversary wake up ECUs (of a parked vehicle) and then control them. That is, the wake-up functions that were originally designed for a good cause become an *attack vector*. Wake-up functions are standardized, implemented, and provided in various in-vehicle networks so that manufacturers can provide remote standby functions, such as remote diagnostics, door control, and anti-theft. Although only a few ECUs of a parked vehicle can be awakened by the wake-up function and then controlled, these attacks are still feasible mainly because they are achieved by controlling ECUs that are asleep, but not completely turned off. The rationale behind our findings is as follows.

- A common irony of the three demonstrated attacks is that the wake-up function of ECUs — originally for enhanced user convenience — is exploited as an attack vector.
- The ease of mounting the attacks stems from the fact that the wake-up signal was defined to be very simple. A simple (agreed-on) wake-up message facilitates extended *battery operation time*, which is defined as the time duration the battery can provide enough power for the driver to start the vehicle. From a security perspective, this battery-saving feature makes it easier for the attacker to wake up ECUs and then drain the vehicle battery.
- It is nontrivial to use message authentication code (MAC), message encryption, or some state-of-the-art defenses against a compromised ECU to send a wake-up signal due to the way the signal was designed and thus defined in the first place; in Controller Area Network (CAN), the wake-up signal is simply a single 0-bit.
- The number of ECUs that can/must be awakened, tends to increase as more enhanced standby features are added to newer vehicle models. This will allow the attacker to immobilize the newer models far more quickly and easily than the older ones.

Through extensive experimental evaluations of 11 test vehicles — i.e., 2008–2017 model-year (compact to mid-size) sedans, coupe, crossover, Plug-in Hybrid Electric Vehicle (PHEV), Sport Utility Vehicles (SUVs), truck, and a Battery Electric Vehicle (BEV) — with Commercial Off-The-Shelf (COTS)/customized OBD-II dongles and connected vehicle apps, we show that all but one 2008 model-year of the test vehicles are equipped with the wake-up functions, rendering BD, DoB, and UC attacks feasible. Moreover, we demonstrate

the BD attack on 4 different representative vehicle types — compact & mid-size SUVs, sedans, and pickup truck built by different OEMs — and show that the adversary can speed up the average battery consumption, draining the vehicle battery within an hour. We also demonstrate the DoB attack on one of our test vehicles and show that the attacker can shut down an ECU indefinitely and thus completely disable the key-fob function, thus preventing the victim from entering and starting his vehicle. Our test vehicles are also used to evaluate the feasibility of an attacker controlling the door and trunk locks via the UC attack and thus accessing the victim's vehicle.

II. Background

A. Operation Modes of ECUs

Even when the ignition is off, in order to provide various functions while minimizing their power consumption, not all ECUs are completely turned off; some ECUs simply go to *sleep*. The need of ECUs asleep to be awakened is increasing for enhanced user/driver experience/convenience. For example, vehicle OEMs are introducing new useful functions, such as PKES, overnight remote diagnostics, remote temperature & door control, and anti-theft while the vehicle is parked and turned off. We refer to such functions that are executable/executed while the ignition is off as *standby functions*. Safety-related controls such as powertrain/engine controls are *not* standby features since the ECUs controlling them are usually configured to be completely turned off when parked with the ignition off. To meet such an increasing need, those ECUs asleep are configured to be awakened via a local or bus wake-up. A local wake-up is triggered when a switch attached to the ECU (e.g., a receiver for the remote key) is turned on. This drives a logic state change on its WAKE pin and thus re-activates the whole ECU. Another mechanism in which an ECU wakes up is whenever it sees a specific (i.e., wake-up) message/signal on the bus. Upon detection of a wake-up signal by the ECU's transceiver, which remains ON even while the ignition is off, it turns on the power supply of the ECU, wakes up the whole ECU, and thus enters *normal* operational mode. We will later elaborate on the wake-up signals along with how the proposed attacks work. If no additional wake-up signal is received within a certain (preset) time period, the ECU goes back to sleep.

B. Fault Confinement and Bus-off Recovery

Error handling is built in the CAN protocol and is important for its fault-tolerance. It aims to detect errors in CAN frames and enables ECUs to take appropriate actions, such as discarding a frame, retransmitting a frame, and raising error flags. If an ECU experiences or incurs continuous errors while transmitting or receiving a message, the CAN protocol specifies that its Transmit Error Counter (TEC) or Receive Error Counters (REC) should be increased [19, 21]. If its TEC exceeds a pre-defined threshold of 255 due to persistent errors, the ECU is forced to enter a state called *bus-off* and shut down. Exploiting such a standardized CAN feature, Cho and Shin [14] proposed a new attack called the *bus-off attack*, which enforces other healthy/uncompromised ECUs to shut down. See [14] for more details of the bus-off attack.

The bus-off mechanism enables recovery of nodes. There are two recovery modes of bus-off status: (1) *automatic* bus-off recovery and (2) *manual* bus-off recovery upon a user request.

III. Threat Model

Like previously-known attacks [13, 14, 22, 27], we consider the adversary capable of remotely (but *not physically*) compromising an in-vehicle ECU via numerous attack surfaces, and can thus gain its control; physically compromising an ECU requires physical access and is thus out of this paper's scope.

An adversary under consideration can

- AV_1. compromise a COTS OBD-II dongle/device in advance, and gain remote control of the vehicle once the driver plugs it into his car [16];
- AV_2. compromise an in-vehicle ECU (e.g., Telematics Control Unit (TCU)) remotely [13, 22, 25]; or
- AV_3. compromise a connected vehicle app (e.g., GM OnStar RemoteLink, Nissan Leaf App, FordPass, BMW Connected, Mercedes me) so as to access the in-vehicle network [9].

The practicability of such adversaries (i.e., gaining control of such attack vectors) has already been proved and demonstrated in [9, 13, 16, 22, 25].

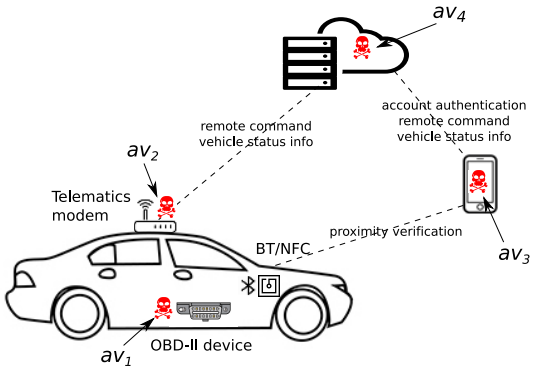

Fig. 1: Illustration of various attack vectors in connected vehicles. Modern vehicles have external interfaces such as OBD-II port, Telematics, and Bluetooth, enabling communication with cloud servers or smartphones for enhanced user experience.

Note that as depicted in Figure 1, the attacker can also attempt to compromise the cloud service that links the connected vehicle app to the vehicle (AV_4). However, due to its lack of practicality and feasibility, we do not consider this as an attack vector.

Since such an adversary would have access to the vehicle's CAN bus and would be able to inject a message, we call such an adversary a *CAN attacker*. Through a compromised in-vehicle ECU (AV_1–AV_2), the CAN adversary can inject any message with forged ID, DLC (Data Length Code), and data.[1] Through a compromised app (AV_3), the attacker can at least issue commands to inject predefined messages (e.g.,

[1]Once plugged in, as COTS OBD-II dongles will also be just another ECU on the in-vehicle network, we consider them as an in-vehicle ECU.

turn on heater, or request diagnostic results) or can even inject arbitrary messages in the worst case. As these messages are all initiated and controlled by the attacker, we call such injected messages *attack messages*. Also, since CAN is a broadcast bus, the adversary can sniff messages on CAN bus. We also assume that the compromised in-vehicle ECU can be continuously powered on — even while the ignition is off — or (at least) has a separate power source/supply during its operation. This can easily be achieved by an adversary as we will elaborate in Section IV. Note that for the compromised app attack vector, such an assumption is neither applicable nor required.

We consider a threat that was never explored before: the adversary's objective is to *immobilize* the victim's vehicle and/or *steal* valuable items inside by controlling a parked vehicle with its ignition *off*, and make its forensic reconstruction as difficult as possible. The adversary achieves this by first waking up in-vehicle ECUs (through the attack vectors) and thereby controlling them. Here, difficult forensic reconstruction (e.g., resembling typical mechanical/electronic failures) is the main motivation to mount cyber (not physical) attacks since it will likely result in replacing wrong parts (wasting money unnecessarily), extending service outage, or blaming wrong suppliers/OEMs. Although the adversary could immobilize a vehicle by simply flooding the in-vehicle network when the driver tries to start the car, thus preventing any other ECUs from receiving/processing commands, we do not consider such an adversary since its symptom is very different from typical mechanical/electronic failures, thus exposing itself for easy detection and removal.

As the attacks under consideration focus on parked vehicles with the ignition off, the adversary must "play" within the given boundaries of its CAN bus. In parked vehicles, only those ECUs that contribute/relate to *standby* features are asleep, while others are completely turned off. This imposes a certain restriction: the CAN attacker only controls ECUs that remain asleep (i.e., not completely off), once the ignition goes off. That is, the attacker's capability is limited to controlling *only* those ECUs equipped with standby features, e.g., door and lighting ECUs. So, we consider the CAN attacker to be incapable of, for example, driving/crashing a car since the powertrain/engine ECUs are *completely turned off* when its ignition is turned off, due to its irrelevance to standby features.

IV. Waking Up ECUs

In order to attack and control a parked car, the attacker first needs to wake up ECUs so that they can be controlled. In this section, we provide details of how the attacker can achieve this and also evaluate its feasibility using different connectivity of the considered attack vectors (AV_1—AV_3).

A. Wake-Up Function

When the ignition is off, some ECUs are configured to run in sleep mode and continuously monitor if there is any incoming bus wake-up signal. The adversary needs to determine the type of *bus wake-up* signal to wake up ECUs.

Standardized wake-up. The remote wake-up behavior of a CAN ECU was first introduced and specified in the ISO 11898-5 standard which defines the bus wake-up behavior as *"One or multiple consecutive dominant (0-bit) bus levels for at least t_{Filter}, each of them separated by a recessive (1-bit) bus level, trigger a bus wake-up."* While t_{Filter} is defined to be within [$500ns$, 5µs], its actual value depends on the transceiver being used.

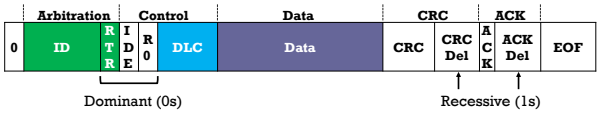

Fig. 2: Format of a CAN data frame.

In a CAN bus with bit rates up to 200kBit/s, i.e., bit width longer than 5µs, *any* 0-bit within a CAN frame triggers a wake-up. For bit rates up to 500kBit/s, the wake-up condition is always met for any normal CAN data message since its 1) RTR, 2) IDE, and 3) r0/FDF bit are all defined to be dominant (0s) as shown in Figure 2. That is, in a 500kBit/s bus, since those three bits—each with width 2µs—are sent consecutively, the resulting duration of dominant bus level becomes at least 6µs, thus always satisfying the wake-up trigger condition.

Note that the ISO standard specifies such dominant bus levels to be separated by a recessive bus level. This is easy to satisfy since CAN always 1) stuffs a recessive 1-bit after 5 consecutive 0 bits, i.e., bit-stuffing [14, 21]; 2) has certain fields fixed with a 1-bit, e.g., CRC delimiter, ACK delimiter as shown in Figure 2; and 3) the user can determine what value(s) to fill in such fields as ID, DLC, and DATA. The same also applies to the extended CAN format with 29-bit ID. However, in this paper, we only consider the basic/standard CAN data format (with 11-bit ID) since the extended format is seldom used (due to its bandwidth waste) in contemporary vehicles.

The reason for OEMs' agreement on a standardized and simple wake-up signal was to not only guarantee a 100ms link acquisition time [20] but also to allow for a low-power design (e.g., RC-circuit) of wake-up detection, i.e., an energy-efficient sleep mode [20]. That is, the simple design/definition of a wake-up signal was to prolong the battery operation time.

Although a similar design is also found in other in-vehicle networks (e.g., FlexRay and Local Interconnect Network (LIN)), we focus on CAN, the *de facto* standard in-vehicle network, for a more in-depth treatment of the attacks.

Power source. One remaining requirement for a CAN adversary to achieve this is that his attack vector has to remain powered on in order to trigger the wake up signal; especially when the attack vector is an in-vehicle ECU. The attacker achieves this fairly easily thanks to two interesting facts. First, the in-vehicle ECUs which a CAN adversary would (or can) compromise or those that he gets to control via some other compromised entities (e.g., connected vehicle app) are most likely to be always powered on, or (at least) have a separate power source/supply during their operation. A typical example ECU that an attacker would target (to compromise) is the TCU due to its wide variety of wireless connectivity. The practicability of the TCU being compromised has already been demonstrated in [13, 22, 25]. Interestingly, the TCU—which

is regarded as one of the most vulnerable ECUs [13–16]—is usually completely (or at least periodically) powered on so as to respond to external events (e.g., requests for remote diagnosis, remote door/temperature control, anti-theft) even after removing the ignition key [3]. Moreover, a TCU is usually equipped with an alternative power supply so that it can operate even when the vehicle power system is off [24]. Similarly, an OBD-II device/dongle, which is also a good target for an adversary to compromise (as demonstrated in [13, 16]), can also be separately powered on (by the attacker) if it is equipped with an external power source, which is often observed in many aftermarket telematics implementations.

Second, although the operational mode of a (compromised) ECU is preset to run in sleep mode when the ignition is off, it does not restrict the adversary to change such a setting. Two most common CAN controllers—Microchip MCP2515 and NXP SJA1000—both allow modification of their operation mode (e.g., normal, sleep, listen-only) through software commands [1, 2]. For ECUs with the Microchip MCP2515 CAN controller, the Serial Peripheral Interface (SPI) remains active even when the MCP2515 is in sleep mode, thus allowing access to all registers. Thus, through the SPI, it is also possible for the user/adversary to read/write the CAN controller registers, including the operational mode register [2]. Such user-level features for configuring the CAN controller allow attackers to easily switch from sleep to normal mode via software commands.

As a result, a CAN adversary can inject wake-up messages to the CAN bus with the ignition off. The transceivers of ECUs asleep observe a wake-up signal on the bus, switch on the ECUs' power supply (usually via an interrupt), and boot up the microcontroller. Hence, the ECUs will return to normal operational mode. Since CAN is a broadcast bus, even a single injected message wakes up all ECUs asleep.

B. Wake-Up Evaluation

To verify whether ECUs in real vehicles can indeed be awakened via simple wake-up messages while the ignition is off, we setup three different remote control settings:

- RC_1. COTS dongle plugged into the OBD-II port — controlled via Wi-Fi or cellular
- RC_2. Raspberry-Pi + PiCAN module to the OBD-II port — controlled via Wi-Fi
- RC_3. Connected vehicle app specific for the test vehicle(s) — controlled via cellular

Note that the wake-up evaluation aims to verify if other ECUs can be awakened under the considered remote control settings (RC_1—RC_3), which are assumed to be already compromised. The detailed process and feasibility of compromising each attack vector were heavily studied and shown in other literature and therefore are beyond the scope of this study. Under the settings for RC_1—RC_2 which correspond to AV_1, we injected wake-up messages/signals to the CAN bus. The wake-up message had its ID, DLC, and DATA fields—that a user/adversary can control at the application layer—all filled with 1s. This was to verify that messages with the minimum number of 0s can also function as a valid wake-up message. In RC_3 where we evaluated using the connected vehicle app, we simply executed any command that was available on the app which therefore injected a pre-defined CAN message while the ignition was off. This setting corresponds to not only AV_3 but also AV_2 in which the compromised TCU is consequently awakened and can therefore be exploited.

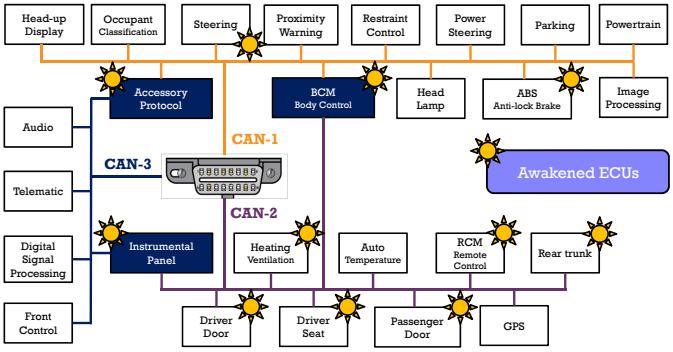

Fig. 3: Vehicle architecture of one of our test vehicles.

In-depth analysis on a test vehicle. Figure 3 shows the in-vehicle network architecture of one of our test vehicles — a 2017 year model — and the ECUs that responded to the (wake-up) messages. The results were equivalent in all three evaluation settings RC_1–RC_3. Note that this network architecture of the test vehicle is not unique for the OEM of our test vehicle but is valid for the vehicles built by other OEMs except for only slight variations in the network architecture [26]. We verified which ECUs were awakened by logging the CAN traffic that contains the message IDs observed on the bus, and by mapping those IDs to the corresponding transmitter ECUs using the test vehicle OEM's CAN Data Base Container (DBC). The CAN DBC describes the properties of the CAN network, the ECUs connected to the bus, and the CAN messages and signals.

When the wake-up message/signal was injected, one can see from Figure 3 that not all ECUs were awakened. This would be most probably due to different ECUs being attached to different battery terminals (with different terminal control policies) in order to minimize battery/power consumption and, at the same time, provide various standby functions. In CAN-1 which connects ECUs performing safety-critical functions, only 4 of 13 of them were awakened. On the other hand, in the CAN-2 bus where ECUs responsible for vehicle body control were connected, almost all ECUs but two were awakened. Considering the fact that contemporary/newer vehicles provide various standby "body control" functions such as keyless entry, hands (foot) free trunk opening, and anti-theft, more ECUs being awakened in CAN-2 than CAN-1 would be the norm.

Verifying wake-up functionality in various vehicles. Using Raspberry Pi + PiCAN device (RC_2), we also verified how different vehicles (OEMs/years/models) react to the injection of a wake-up message.[2] To corroborate the existence of wake-up function in different cars, when evaluating the wake-up

[2]As the results were equivalent in all settings RC_1–RC_3, the following evaluations were done via just RC_2.

function, we chose 11 different test vehicles with various types like (compact and mid-size) sedans, coupe, crossover, PHEV, SUVs, truck, and a BEV with model-years 2008–2017.

Figure 4 shows the number of distinct messages observed on the CAN bus 1) when the ignition was on, and 2) when we woke up ECUs on the bus by injecting a wake-up message while the ignition was off. Since the feasibility of wake-up stems from how the in-vehicle network standard is specified and implemented, instead of OEMs' design decisions, we have chosen not to identify/reveal the particular make and model used in our evaluation. Note, however, that the 11 test vehicles (shown in Figure 4) we examined are from different OEMs and also represent different models. For some vehicles, since their OBD-II pinout was configured to not provide full access to all of their buses, we only show those that were awakened in the accessible bus(es).

When waking up ECUs in some old cars (most with low-level trims), far less distinct message IDs and lower percentages of them (compared to the case with the ignition on) were observed on the bus than other newer cars. Since the number of ECUs is proportional (albeit not linearly) to that of distinct message IDs, we can infer that there were less awakened ECUs transmitting them in older cars; not all messages can be sent by a single ECU due to the high overhead. This would most probably be due to the fact that the older cars (2008 and 2013 model-year) do not require/provide any (or not many) standby functions (e.g., PKES).

On the other hand, when a wake-up message was injected on the buses of nine 2015–2017 model-year test vehicles, we observed that 49.12–94.95% (75.44% on average) of the distinct message IDs sent while the ignition was on, were *also* sent when ECUs were awakened while the ignition was off. Such a high number/portion of ECUs being awakened by a wake-up message and thus sending more message IDs on the bus is because they had numerous standby functions installed for enhanced driver's experience and convenience (e.g., PKES/Remote Keyless Entry (RKE), hands-free trunk opening, anti-theft) — a trend that is expected to grow. These results corroborate the fact that vehicle ECUs are indeed equipped with the wake-up functionality (adhering to the standard) and can thus be exploited by the CAN adversary as an attack vector.

V. Detailed Attack Analysis

Figure 5 presents an attack flow diagram for BD, DoB, and UC attacks. As one can see from the flow diagram, the attacker exploits one of the three considered attack vectors (AV_1–AV_3) to wake up ECUs on the bus(es). For DoB attack (Section V-B), the attacker attempts to shut down awakened ECUs by exploiting their error handling and recovery mechanisms, i.e., mounting a large-scale bus-off attack. In case of BD attack (Section V-A) and UC attack (Section V-C), however, the attacker attempts to exploit the awakened ECUs to control certain functionalities of the vehicle (rather than shutting them down). As a result, the attacker can drain the vehicle battery and unlock certain doors or trunk to access the vehicle, respectively.

Since BD and UC attacks require *control* of some ECUs, the adversary may have to figure out which message ID(s) to use. That is, the attacker may have to reverse-engineer messages since IDs vary with vehicle manufacturers and models. This, in general, becomes a (high) technical barrier for the adversary, especially when mounting state-of-the-art attacks on different vehicles. However, for the purpose of BD attack and UC attack, the message reverse-engineering can be done easily without fuzzing. Specifically, by reverse-engineering messages based on the *driver context*, it becomes much easier for the adversary to figure out which messages to use in mounting BD attack and UC attack on different vehicles. Driver-context based reverse engineering is achieved based on the *routined and expected* driver's behavior(s) of when s/he starts the car.

A. Battery-Drain (BD)

Once the CAN adversary wake up the ECUs asleep, they switch to, and run in normal operational mode. Note, however, that an awakened ECU goes back to sleep after a certain period of time. Hence, by waking up ECUs as much and as frequently as possible, the adversary can *continuously* force those ECUs to run in normal mode, although they should remain asleep. If ECUs are configured to stay in normal mode (after waking up) for a duration of T_{wakeup}, the frequency of wake-up messages from the attacker has to be at least $\frac{1}{T_{wakeup}}$.

Some ECUs are connected to physical actuators, which consume high electric energy, thus becoming likely targets of a BD attacker. Examples of such ECUs—which are mostly related to vehicle body control functions—are the ones controlling interior/exterior lights and Heating, Ventilation, and Air Conditioning (HVAC). To further enhance the driver's convenience, OEMs are increasingly letting car owners control their cars (e.g., unlock/lock, turn on heater) by using their smartphone apps with an eventual goal to totally replace key-fobs with smartphone software [5–7]. This indicates that an adversary exploiting a connected vehicle app already has the capability of indirectly/directly mounting BD attack.

Evaluation. We use the following controls that increase battery consumption in our attack evaluation.

C1. *Turning on lights by changing the power mode:* An adversary first reverse-engineers the control message (via driver context) which determines the vehicle's power mode (e.g., off, accessory mode, run, and start), wakes up ECUs on the bus, and attempts to change the power mode using the reverse-engineered message/ID. When we controlled the power mode of a parked vehicle, various indicators on the dashboard were (temporarily) illuminated. Likewise, the infotainment system was also booted up.

C2. *Turning on lights by unlocking/locking doors:* A CAN adversary can also attempt to repeatedly unlock/lock the vehicle's doors (while parked). When the driver (or the adversary) unlocks the car, *welcome lights* of the vehicle illuminate for enhanced visibility. Note that the numbers and types of the welcome lights that illuminate may vary with the vehicle manufacturer/year/model, and also, depending on whether or not the lighting control system is set to "automatic," the default setting for most

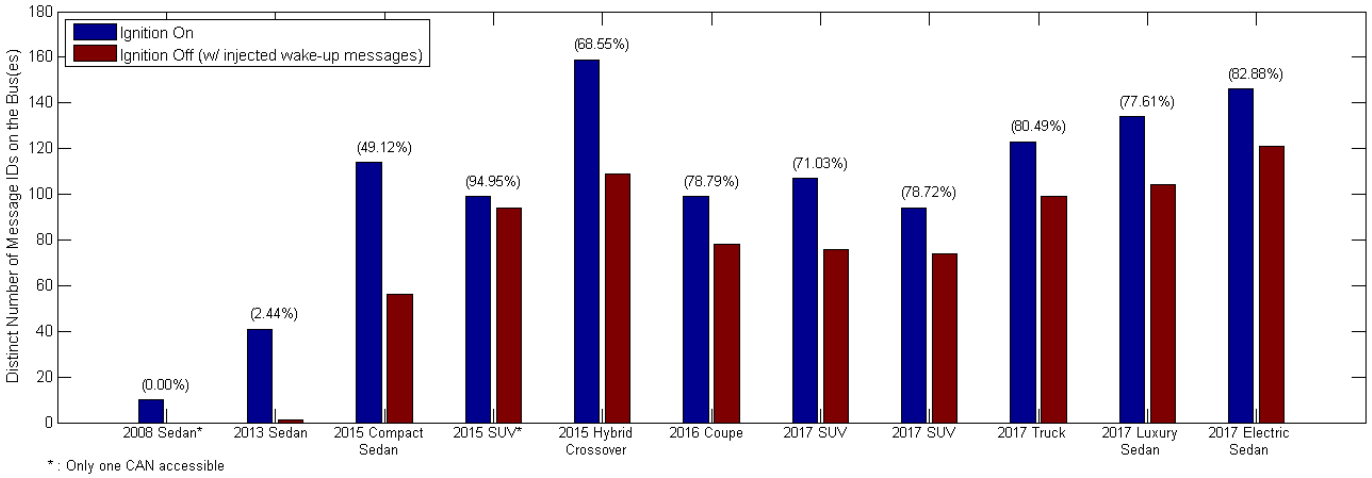

Fig. 4: Verifying wake-up messages in 11 different vehicles. The number of distinct message IDs were observed when the ignition was off and the ECUs were awakened via a wake-up message is compared with that when the ignition was on.

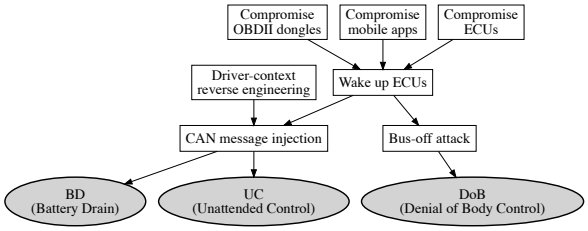

Fig. 5: Depicted is how each attack is mounted after waking up ECUs in a parked vehicle.

drivers [13]. An adversary can continuously illuminate all the welcome lights and thus significantly increase the average battery consumption. Similarly to the attack case C1 where the power mode was controlled, the door control module is another ECU which provides a standby function and hence must not be completely off, but must be asleep instead.

C3. *Turning lights on by opening the trunk:* The adversary may attempt to open the trunk of a car. When the trunk is opened, for enhanced visibility, vehicles are configured to illuminate its interior map, dome, and trunk lights. In contrast to C1 and C2, the attacker is only required to inject a single trunk-control message into the bus if the lights remain on while the trunk is open (as some vehicles do). Even if the lights automatically go off after some time, the attacker can re-inject the trunk-control message to re-illuminate them. Note, however, that opening the trunk could be visibly intrusive, and thus it needs to be mounted at special situation (e.g., overnight parking).

C4. *Turning on HVAC:* Compared to the previous controls, HVAC is visibly non-intrusive from outside, i.e., more stealthy, and at the same time drains the battery faster. Similarly to C1–C3, HVAC is another function that can be controlled (and turned on) via message injections after waking up the ECUs. In fact, C4 is one the most likely features to be equipped on connected vehicle apps for remote control; not to mention that C1-C3 are as well. Many vehicles are now equipped HVAC functions such as heating/cooling seat, air conditioning, defroster, and they usually consume a large amount of electric power.

After verifying that the ECUs in our test vehicles can be awakened via basically any wake-up message (even with all 1s in ID, DLC, and DATA fields), we measured the amount of battery drain for the different controls mentioned above: 1) simple wake-up, 2) power mode control, 3) repetitive door unlock & lock, 4) opening the trunk, and 5) HVAC operation. Out of the 11 test vehicles, we picked 4 vehicles which each represent the most popular vehicle types. We evaluated the proposed attacks on the following test vehicles from different OEMs: Vehicle A – 2019 model compact-size SUV, Vehicle B – 2017 model mid-size SUV, Vehicle C – 2017 model pick-up truck, Vehicle D – 2017 model BEV sedan. We were able to reverse-engineer the control messages for the above functionalities via driver-context-based reverse-engineering.

To measure the drained/discharged current from the car battery, we placed the current clamp meter while the vehicle was parked with its ignition off. Then, we injected (iterative) sequence(s) of control messages to the vehicle through the OBD-II port. Table I summarizes the average amount of current measured to be drawn from the vehicle battery when each attack was mounted additionally.[3] When the ignition was off and no ECU was awakened, less than 0.1A was consumed in all test vehicles. As we controlled more functionalities, the average battery consumption increased significantly. In most vehicles, the HVAC functionality (with max A/C) drew a vast amount of current from the battery. One interesting finding during our experiments was that a negative current was observed in the BEV (Vehicle D) when changing the power mode and running the HVAC operations. It turned out that these functions in the BEV activated the *main battery* pack

[3]The currents were averaged over 5-min measurements. We averaged them across time, not trials; otherwise, each attack might have been mounted when the car's battery state is different, possibly making the measurement/comparison inaccurate.

Attacks	Discharged Current [A]				Max. Battery Operation Time [Hours]			
	Vehicle A	Vehicle B	Vehicle C	Vehicle D	Vehicle A	Vehicle B	Vehicle C	Vehicle D
Wake-up	2.5 (1.0x)	4.5 (1.0x)	5.0 (1.0x)	5.6 (1.0x)	18.0	10.0	9.0	8.04
Change Power Mode	12.7 (5.1x)	13.8 (3.1x)	20.0 (4.0x)	-9.5	3.54	3.26	2.25	—
Lock&Unlock Door	9.2 (3.7x)	9.4 (2.1x)	8.7 (1.7x)	5.4 (1.0x)	4.89	4.79	5.17	8.33
Open Trunk	14.5 (5.8x)	20.0 (4.4x)	—	—	3.10	2.25	—	—
HVAC (max A/C)	26.3 (10.5x)	33.1 (7.4x)	40.0 (8.0x)	-4.7	1.71	1.36	1.13	—

TABLE I: Shown are the measured discharged current and maximum battery operation time under different BD controls. The multiplication factor (Nx) shows the relative amplitude of discharged current. We estimate the (theoretical) maximum battery operation time while assuming a 45Ah standard car battery for all test vehicles for easy comparison.

(for motors) to charge the *start battery* unit. This indicates that the proposed attack can, in fact, even drain the main battery in BEVs. In reality, an adversary will be able to combine multiple controls to drain the vehicle battery quickly. For example, when we launched the power mode change, heating seat, and max A/C functions together on Vehicle A, the total current draw was measured as 50.4A, which is approximately 25.0x higher than when we simply woke up ECUs.

For an easy comparison of battery current consumption under different BD attack controls, we summarize the estimate battery operation time in Table I while assuming a 45Ah standard car battery for all test vehicles. Note that the estimation might be different from the reality with different battery capacity and battery saving technique across the test vehicles. With the HVAC operation — which seemed to drain the most amount of current — the car battery is expected to be completely depleted within 2 hours for all tested vehicles. Note that such a figure is when *only* the HVAC was being controlled by the adversary during the BD attack. So, when the adversary controls not only HVAC but also other functionalities, s/he can drain the victim's vehicle battery much faster. Taking the measurements from Vehicle A as an example, the time to completely drain a fully-charged battery is estimated to be approximately 53 minutes when the attacker launches all 5 attacks listed in Table I.

B. Denial-of-Body-control (DoB)

In addition to BD attack, the CAN adversary can mount a Denial-of-Body-control (DoB) attack. While the ignition is off, the CAN adversary wakes up ECUs from the compromised modules so as to make them responsive to his injected messages. Then, the adversary switches its bit rate (e.g., from 500 kBits/s to 250 kBits/s). According to the CAN error-handling mechanism, this makes *all* awakened ECUs on the bus continuously experience and incur errors, and eventually enter the bus-off state, i.e., shut-down. This way, the adversary not only mounts the bus-off attack on a target ECU (as demonstrated in [14]) but also on all ECUs on the bus. Instead of changing the bit rates, changing internal resistances or capacitances can be an alternative way of achieving this. As a result, per bus-off recovery specification, depending on the ECUs' software configurations, some ECUs would recover from the bus-off state, whereas some others will *not*.

The DoB attack is mounted similarly to the bus-off attack, except it further exploits the following fact specified in the ISO 11898-1 standard [4]: "*a node can start the recovery from the bus-off state only upon a user's request*," where the user's request depends on the ECU software configuration. DoB attack thus exploits this definition of bus-off recovery as follows. Depending on the car manufacturer and year/model, ECUs such as Body Control Module (BCM) or Remote Control Module (RCM) — that authenticates each message to and from the remote key-fob — can be configured/defined not to recover from the bus-off state mainly for either safety (since bus-off is a serious problem [14]) or anti-theft purposes. Hence, if the CAN adversary were to mount the DoB attack on such a vehicle, then s/he can indefinitely shut down BCM or RCM, thus cutting off the communication channel between the (driver's) remote key-fob and the vehicle. Contemporary/newer vehicles are usually equipped with PKES, which allows users to unlock and start their cars while keeping their key-fobs in their pocket [17] and is installed in BCM or RCM. For the vehicle to be unlocked/started, PKES must verify that the legitimate key-fob is in the vehicle's vicinity. Therefore, by shutting down BCM/RCM (and thus PKES), the vehicle will *not* be allowed to receive and authenticate any remote key signals (sent by the key-fob).

Once the attacker succeeds in mounting DoB attack, there is no need for the attacker to mount the attack again because some ECUs that have entered bus-off will never boot up again anyway. This allows the attacker to succeed in mounting the attack in a very short period of time without leaving any trace, thus making its forensic difficult.

Evaluation. Through experiments on one of our test vehicles, we also verified the consequences of the proposed DoB attack. After launching the DoB attack on one of our parked test vehicles, only in a few seconds, we confirmed from the CAN traffic that all ECUs on the bus were continuously incurring and/or experiencing errors, causing all the ECUs to enter the bus-off state. After mounting the DoB attack, we observed most, but *not all* of the ECUs, recovered from the bus-off state as configured. We observed that the number of distinct message IDs sent on the bus was actually reduced by 6 after the DoB attack. By mapping those missing IDs to the actual transmitter ECU using the DBC file, we found that RCM did *not* recover from the bus-off, i.e., remained shut down, most probably due to its distinct recovery policy configuration. For new functions such as anti-theft or engine immobilizers to be materialized, RCM might have to be set in that way, which ironically *benefits* the attacker in mounting and succeeding in the DoB attack. Since the RCM was indefinitely off, the key-fob was not authenticated and thus could not

establish a connection to the vehicle. In our testing, the vehicle could not detect that the key was in its vicinity although the key was in fact placed right in front of the dashboard.

As discussed earlier, this consequence comes from the fact that OEMs (or their ECUs) may have different bus-off recovery configurations. In our test vehicle, the setting of an RCM to not recover from the bus-off "favors" the attacker in mounting a critical DoB attack. We found the only way to restore the vehicle back to its original state after a DoB attack was to disconnect the battery, wait for a few minutes, and re-connect the battery. Such a process resets the *states* stored in each ECU and thus lets them run in their original/intended states. However, imagining a victim confronting the symptoms of DoB attack, i.e., the key-fob neither working nor being detected, he might first try to change the key-fob battery. Obviously, since that won't work, he would consider the car battery completely dead and therefore, would probably have the car towed to the service station for a battery replacement, wasting money and time unnecessarily.

We tested the DoB attack on only one test vehicle due to the restriction imposed by the OEM that provided the vehicle and collaborated with us. For safety, warranty, and legal reasons, we did not launch the DoB attack on our personal cars or other test vehicles, including those that we had tested for the BD attack. DoB attack, however, can be generalized since its effectiveness depends on the targeted ECU's software configuration (as specified in the ISO 11898-1) and some ECUs are required to be set so that they do not recover from bus-off.

C. Unattended Control (UC)

An interesting consequence of waking up ECUs is not only the increased battery consumption but also the pathway it provides for the attacker to *control* ECUs. After waking up, since ECUs previously asleep now run in normal operational mode—the same as when the ignition is *on*—the CAN adversary becomes capable of controlling them. We refer to "controlling an ECU" as executing the ECU's function(s) via message injections. The severity of malicious controls when the vehicle's ignition is on and moving would be very different from that when the vehicle is parked with its ignition off. Meanwhile ensuring that the brakes do not unwillingly engage/disengage in a moving vehicle is safety-critical, it is not when the ignition is off. However, harmful consequences are still caused by an attacker targeting an unattended vehicle.

Unattended control of a vehicle has been shown to be feasible by thieves and researchers using a device to mimic a key-fob to unlock/lock doors or by mounting attacks on the remote keyless entry systems [8, 11, 18]. Interestingly, attackers can, in fact, achieve the same goal — launch the proposed UC attack — by injecting messages from the compromised modules via the wake-up functions and driver-context reverse engineering. As a result, the adversary/thief can steal the victim's valuable items inside the car by unlocking the door or opening the trunk without any key-fob, thus causing financial losses to the victim.

With the similar evaluation set-up in BD attack, we were able to unlock the door and opened the trunk on 4 test vehicles via message injections. We omit the detailed results.

VI. Related Work

Exploiting a remotely compromised ECUs, researchers have shown how various vehicle maneuvers can be (maliciously) controlled by injecting packets into the in-vehicle network [22, 25]. Similarly, in 2015, researchers were able to compromise and remotely kill a Jeep Cherokee running on a highway [27], which triggered a recall of 1.4 million vehicles. In 2016 and 2017, researchers were able to hack Tesla cars and control their maneuvers [10]. Researchers have also demonstrated that an adversary can shut down a specific ECU or even the entire in-vehicle network merely via packet injections [14]. Also proposed is new hardware that can generate/fabricate magnetic fields, spoof the wheel speed sensor of a running vehicle, thus activating the Anti-lock Braking System (ABS) [29].

Although such attacks were effective, they were mounted and thus considered malicious only when ECUs were turned on while the vehicle was running. To the best of our knowledge, there has been no study on what the adversary can achieve while the ignition and thus the ECUs are turned off.

VII. Discussion

Countermeasures. As of the current CAN standard, since the wake-up itself can be achieved with any CAN message having a 010 bit-sequence, adding MAC or message encryption cannot prevent the adversary from waking up ECUs; a message with MAC/encryption will still have such a sequence.

One may think of continuously running an IDS even when the ignition is off in order to capture any abnormal wake-up messages, e.g., wake-up messages should not be seen very frequently. However, since the operation of an IDS would increase the current drawn from the battery, such an approach may defeat the very purpose of reducing battery consumption. Like other ECUs asleep, the IDS ECU can also be configured to sleep most of time and wake up only when it sees a wake-up message. As a countermeasure against both types of attack, the wake-up pattern of an IDS can then be modeled and used to detect any abnormal wake-up requests without continuously running it. Similarly, the IDS can be configured to wake up periodically, check the battery SoC—if there was any significant drain recently—and react accordingly. Moreover, especially for the DoB attack, how to recover from the bus-off state has to be re-examined in order to withstand the DoB attack, as we had demonstrated.

The instant behavior of UC attack will make the (wake-up pattern) model-based IDS ineffective, which can be addressed by secure hardening against each attack vector in Figure 1.

Enhanced wake-up functionality. Partial deactivation of subnets within a given network has been discussed and planned by car manufacturers, mainly to reduce energy consumption and CO_2 emissions [23]. In such a setting, only the pre-defined wake-up messages that pass the wake-up masks/filters of selective ECUs can wake them up during operation. However, since that message is "pre-defined" and can easily be learned

from the CAN traffic and its sudden change in the number of message IDs, the wake-up message itself can still be learned and used by an adversary.

Limitations. Although we succeeded in mounting the three attacks on most test vehicles, not many ECUs were awakened when a wake-up message was injected in *older* vehicles, because they had less standby functions than *newer* models, and thus had less ECUs asleep when the ignition was off. For DoB attack, since its success will totally depend on how the OEMs configured their "bus-off recovery" for different ECUs, it might not be as feasible as BD and UC attacks. The BD and UC attacks will still be feasible unless the standard wake-up procedure is changed or standby functions are not installed.

VIII. Conclusion

In this paper, we have analyzed architectural vulnerabilities and discovered *Battery-Drain* (BD), *Denial-of-Body-control* (DoB), and *Unattended Control* (UC). They are counter-intuitive in that attacks are commonly believed to be possible and effective only with the ignition on. Specifically, an attacker is shown to be able to remotely wake up ECUs and mount attacks through the compromised dongle, TCU, or connected vehicle app, even while the ignition is off. Through extensive experiments on 11 real vehicles using 3 different remote connectivities, such attacks are shown to be easy to mount and very critical. Ironically, the adversary exploits, as attack vectors, the in-vehicle network features originally designed for either energy-efficiency or enhanced user experience (e.g., standby functions). There may still remain different types of unknown and unintuitive vehicle vulnerabilities. It is therefore important to understand what consequences existing standardized functionalities can lead to. It calls for concerted efforts from both academia and industry on this possibility and countermeasures thereof to build more secure vehicles.

Acknowledgements

This work was supported in part by the NSF Grant CNS-1646130 and the ONR Grant N00014-18-1-2141.

References

[1] "Philips/NXP SJA1000 Stand-alone CAN controller datasheet. [Online] Available: http://web.archive.org/web/20170926054355 / https://www.nxp.com/," 1997.

[2] "Microchip MCP2515 Datasheet.[Online] Available: http://web.archive.org/web/20171029071459 / http://www.microchip.com/," 2003.

[3] "Application-driven power management keys in-car telematics [online] available: https://web.archive.org/web/20171105225654 / https://www.eetimes.com/," 2004.

[4] "ISO 11898-1. Road Vehicles interchange of digital information controller area network (CAN) for high-speed communication [Online] Available: http://web.archive.org/web/20170609150140 / https://www.iso.org/," 2015.

[5] "Volvo wants your phone to be the only car key you ever need [online] available: https://web.archive.org/web/20170703130519 / https://www.theverge.com/," 2016.

[6] "Lexus 2018 technology [online] available: http://www.herbchamberslexusofhingham.com," 2017.

[7] "Tesla model 3 has no key: so don't forget your phone [online] available: https://web.archive.org/web/20171115054847 / https://www.cnet.com/," 2017.

[8] "Thefts spike as thieves harness technology to get inside locked cars [online] available: https://newyork.cbslocal.com/2017/01/24/stolen-cars-key-fob -technology," 2017.

[9] "A reverse engineered interface for the bmw i3 electric car [online] available: https://github.com/edent/bmw-i-remote," 2018.

[10] "Tesla Responds to Chinese Hack With a Major Security Upgrade. [Online] Available: http://web.archive.org/web/20171104002232 / https://www.wired.com/," *WIRED*, Sep. 2016.

[11] R. Benadjila, M. Renard, J. Lopes-Esteves, and C. Kasmi, "One car, two frames: Attacks on hitag-2 remote keyless entry systems revisited," in *Proceedings of the 11th USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, 2017.

[12] R. R. Brooks, S. Sander, J. Deng, and J. Taiber, "Automobile security concerns," *IEEE Vehicular Technology Magazine*, vol. 4, no. 2, pp. 52–64, June 2009.

[13] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028073

[14] K.-T. 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*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1044–1055.

[15] ——, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 911–927.

[16] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Washington, D.C.: USENIX Association, 2015.

[17] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," in *In Proceedings Of The 18th Annual Network and Distributed System Security Symposium (NDSS)*, 2011.

[18] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlid, "Lock it and still lose it —on the (in)security of automotive remote keyless entry systems," in *Proceedings of the 25th USENIX Security Symposium*. USENIX Association, 2016.

[19] B. Gaujal and N. Navet, "Fault confinement mechanisms on can: Analysis and improvements," *IEEE Trans. on Vehicular Technology*, vol. 54, no. 3, pp. 1103–1113, 2005.

[20] T. Hogenmuller and H. Zinner, "Tutorial for Wake Up Schemes and Requirements for Automotive Communication Networks [Online] Available: http://grouper.ieee.org/groups/802/3/RTPGE/," Jul. 2012.

[21] *CAN Specification Version 2.0, Robert Bosch GmbH [Online] Available: http://web.archive.org/web/20170926054355 / https://www.nxp.com/*, International Standards Organisation (ISO) Std., 1991.

[22] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 447–462.

[23] T. Liebetrau, U. Kelling, T. Otter, and M. Hell, "Energy Saving in Automotive E/E Architectures [Online] Available: https://www.infineon.com/," Dec. 2012.

[24] J. Mikulski, *Transport Systems Telematics: 10th Conference, TST 2010, Katowice - Ustron, Poland, October 20-23, 2010. Selected Papers*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2010.

[25] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Defcon 21*, 2013.

[26] ——, "A survey of remote automotive attack surfaces," *Black Hat USA*, 2014.

[27] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.

[28] D. Nilsson and U.Larson, "A Roadmap for Securing Vehicles against Cyber Attacks," in *NITRD National Workshop on High-Confidence Automotive Cyber-Physical Systems*, Apr. 2008.

[29] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 55–72.