From Weeping to Wailing: A Transitive Stealthy Bus-off Attack

Paul Agbaje, Habeeb Olufowobi *Member, IEEE*, Sena Hounsinou, Gedare Bloom *Senior Member, IEEE*

***Abstract*—**
The integration of the Internet of Things (IoT) devices and solutions into passenger vehicles have transformed cars into a complex system with intelligence and a platform for extending information technology possibilities. These devices communicate through in-vehicle networks that use the controller area network (CAN) as a de facto standard for the safety-critical functionality of the vehicles. One creative exploit against CAN is the bus-off attack, which uses the fault tolerance capabilities of the CAN bus to coerce a victim electronic control unit (ECU) into the bus-off state from which it is not allowed to access the bus. As a result, the victim ECU is unable to send or receive messages. The WeepingCAN attack is a stealthy variation of the bus-off attack that reduces its observability and therefore the effectiveness of detection-based mitigation. In this paper, we introduce three software-based improvements that greatly increase both the efficiency and effectiveness of the WeepingCAN attack. First, we introduce a novel zero-phase approach for synchronizing the attack. Second, we discover an alternative approach to disable retransmissions, which is a key capability of WeepingCAN, that allows the attack to be conducted from more ECUs than before. Third, we identify a transitive attack strategy that enables an attacker to target many more ECUs than originally possible. We evaluate our improvements experimentally using a CAN benchmark and find that the zero-phase synchronization improves the attack success rate from 75% to over 90% and the transitive attack strategy enables all the ECUs in the benchmark to be attacked.

***Index Terms*—Controller Area Network, bus-off, WeepingCAN**

I. Introduction

Increasingly, embedded sensors have permeated modern vehicles, and their interconnectedness to the Internet has resulted in the Internet of Vehicles (IoV). Autonomous IoVs integrate the Internet of Things (IoT), software, and emerging network technologies to support vehicular applications for intelligent transportation systems (ITS) [1], [2]. While external vehicular networks allow vehicles to communicate with other IoV entities, such as the roadside units and intelligent devices using vehicle-to-everything technology, the controller area network (CAN) provides an inexpensive message-based protocol for electronic control units (ECUs) communication in the vehicle [3], [4]. However, the ever-increasing complexity and connectedness of passenger vehicles provide opportunities for adversaries to exploit the in-vehicle networks and ECUs that operate the electrical and electronic systems to control driving functions [5]–[13]. This vulnerability exposes CAN to attacks and raises the security risks of entities in ITS. The prior work investigates security of the CAN because it is the in-vehicle network used in the majority of vehicles for the safety-critical ECUs, i.e., those controlling the powertrain. Also, to mitigate attacks in CAN, two primary methods are widely explored: intrusion detection systems (IDSs) [14] and message authentication [15], [16].

The *bus-off attack* [5] is an interesting attack that exploits a security vulnerability introduced by the CAN mechanisms for fault tolerance to force an ECU to enter the bus-off state. In the bus-off state, an ECU is not able to participate on the CAN bus. The state is entered when an ECU's error rate exceeds a set threshold. This attack proceeds in two distinct phases: the first phase consists of a cascade of transmission errors ending with two successful transmissions, and the second phase exhibits a repeated pattern of transmission errors followed by successful retransmission. The cascade of errors in phase 1 is an unambiguous feature signaling that a bus-off attack is in progress [5], and countermeasures to bus-off rely on detecting the cascade as a prelude to prevention.

The *WeepingCAN attack* [12] is a variation of the CAN bus-off attack that removes these (and a few other) detectable features. This attack variant introduces three key changes to achieve the lower detectability: disabling retransmissions, recessive bit injections, and a skipping attack strategy to slow the attacker's error rate and allow it to succeed in more cases than a greedy strategy would. With recessive bit injection, instead of injecting a message with a dominant bit-error into the bus, the attacker inserts a recessive bit when the victim's message contains a dominant bit, causing a mismatch during transmission. This approach causes the attacker's CAN controller to raise an error-active flag instead of the victim's. Although the attack has been shown to work reasonably well, the attack can only target a limited set of *victim* ECUs based on the difference between the transmission rates of the attacker and the victim. This limitation restricts the range of potential targets and inhibits the attacker's capability. In addition, the proposed methods to disable message retransmission in the *WeepingCAN* attack are not available on many CAN controllers, limiting the generalizability of the attack.

To address these limitations, we propose novel improvements that broaden the attacker's capabilities and allow for a stealthy bus-off attack that can be better generalized to more CAN controllers. Firstly, our approach uses *zero-phase*

This work is supported in part by NSF CNS-2046705 and Colorado SB18-086. *Corresponding author: Paul Agbaje*.

Paul Agbaje and Habeeb Olufowobi are with the University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: pauloluwatowoju.agbaje@uta.edu; habeeb.olufowobi@uta.edu).

Sena Hounsinou is with Metropolitan State University, St Paul, MN 55106 USA (e-mail: sena.houeto@metrostate.edu).

Gedare Bloom is with University of Colorado Colorado Springs, Colorado Springs, CO 80918 USA (e-mail: gbloom@uccs.edu).

synchronization to allow attackers to coordinate attacks more effectively with the victim's message transmissions. Secondly, we introduce a new strategy to disable message retransmissions during the attack. Unlike previous approaches, our method applies to more CAN controllers and produces less detectable attacks. Thirdly, we propose a *transitive strategy* that allows attackers to target more victim ECUs than previously possible, enhancing the attacker's practicality and versatility in complex automotive networks. With these techniques, an attacker gains a significant advantage in executing more effective and stealthy bus-off attacks against in-vehicle ECUs.

In summary, we make the following contributions with this paper:

- a *zero-phase synchronization* approach that allows the attacker to better align and time its attack with the victim transmissions to improve the attack success rate;
- a new strategy to disable message retransmissions during the attack that is available on more CAN controllers than the prior strategies and also achieves a lower rate of retransmissions, thus it is less detectable;
- a *transitive strategy* to allow the attacker to target many more victim ECUs than possible in the prior work: WeepingCAN was able to target 3 ECUs in a benchmark [12], which we reproduce to show that the transitive approach allows the attacker to target the full set of 6 ECUs under different bus speeds and loads;
- analysis of attack features and possible countermeasures;
- and experimental evaluation of the improved attack.

In the next section we define the threat model and our notation. Section III reviews CAN, the bus-off and WeepingCAN attacks, and related work. We present our improved attack in Section IV and evaluate its effectiveness with experimental results in Section V. Limitations and countermeasures are discussed in Section VI, and Section VII concludes the paper.

II. Threat Model

Consistent with CAN cyberattacks in prior work [5]–[9], [11], [12], we assume an adversary (denoted $\mathcal{A}$ or attacker) that has gained complete control of the software executing on some *compromised ECU* with access to the CAN bus. The attacker can execute arbitrary code on this ECU and can send/receive arbitrary messages on the CAN bus. Acceptance message filtering may, in general, be bypassed by the attacker through reconfiguring the CAN hardware registers via software configuration commands [5]. Arbitrary code execution implies that the attacker can program registers associated with the CAN controller and override interrupt handlers.

The initial exploit that allows the attacker to compromise an ECU is out of the scope of this paper, but we note that ECUs and gateways are known to be vulnerable due to implementation flaws [6]–[8] and that numerous off-the-shelf OBD-II dongles are remotely exploitable [17]. From this initial foothold, we show that an attacker can silently DoS arbitrary ECUs.

The attacker's goal is to successfully cause a particular ECU (denoted $\mathcal{V}$ or victim) to enter the bus-off state. The main difference between our threat model and that of much of the prior work is that we add the constraint that the attacker remains *stealthy* to escape detection techniques that rely on observing visible features during the attack. The original CAN bus-off attack by Cho and Shin [5] identified these visible features, namely F1 and F2: F1 is due to a message encountering consecutive errors, and F2 happens when the victim observes a successful message transmission after a consecutive pattern of errors. Notably, the original attack does not satisfy this constraint.

We assume that the attacker has conducted an extensive offline analysis of the target vehicle. Specifically, the attacker has knowledge of the message parameters (ID, periodicity, senders, and receivers) used by the ECUs on the CAN bus [9], [18]–[22]. Based on the offline analysis, an attacker is able to determine the set of messages that each ECU transmits. At a minimum, WeepingCAN uses information about the messages sent by the victim and compromised ECUs. For example, Kulandaivel et al. [18] analyzed offline data by collecting several minutes of CAN traffic and using the data to map each unique source ID to its ECU. The analysis was performed on the 2009 Toyota Prius and 2017 Ford Focus, with a false positive rate of 0%, which includes determining the periodic characteristics of the message IDs. Also, Bloom [12] was able to identify a target message with a genuine preceded ID and determine its periodicity for the WeepingCAN attack on a 2016 Kia Optima. For our attack, we expect that, for a given vehicle make and model, an attacker would be able to determine offline the ECUs that satisfy this model.

Fault Model: We assume that the messages and ECUs involved in the attack (as victim or attacker) during the span of a bus-off attack do not incur any normal faults. This assumption simplifies the attack analysis without sacrificing its plausibility. In a real attack scenario, the occurrence of faults may either increase or decrease the success rate of attacks and countermeasures. We discuss the possible interactions of normal faults with attacks in Section VI.

III. Background and Related Work

We begin with an overview of CAN features before describing the WeepingCAN bus-off attack. Natale et al. [23] comprehensively covers CAN topics, and Hu [24] provides an excellent overview of CAN bus error handling and jitter. Bloom [12] explains in detail the original bus-off attack in the context of detectable features.

A. CAN Overview

CAN is a broadcast bus network that uses carrier-sense multiple access and collision detection (CSMA/CD) for communication. Each CAN controller and its connected transceiver, or PHY, attached to the physical bus is called a CAN node. To prioritize messages in the network, CAN relies on decentralized arbitration that considers the sender's identifier (ID). Priority is determined based on the ID, with smaller IDs having a higher priority when the IDs are considered as unsigned integers. CAN encodes a logic-0 bit as a *dominant* signal, while logic-1 is considered a *recessive* signal. Nodes in the network float a recessive signal when idle and transmit a dominant signal by

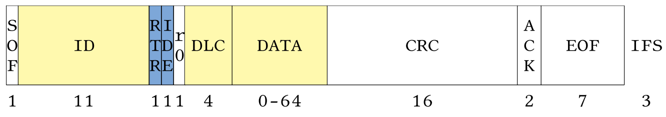


Fig. 1. CAN data and remote frame format with field lengths in bits [5]. The shaded fields are determined in software, and the light yellow shaded fields are used by the attacker. The other fields are managed by a (hardware) CAN controller.

driving the bus low. This mechanism allows CAN to prioritize dominant bits, allowing smaller integers (those with a longer prefix of 0s) to take priority over larger ones during the arbitration process.

1) Data Frame: Fig. 1 depicts the format of a basic CAN message frame, which is a data frame when the remote transmission request (RTR) bit is dominant and a remote frame when the RTR is recessive. If the ID extension (IDE) bit of a data frame is dominant, then it is a standard frame. However, extended frames have a slightly different format, using 29 bits for the ID and a recessive IDE bit. In CAN, two other frame formats are used to transmit error and overload frames. An error frame is transmitted when an error condition is detected on the bus, while an overload frame adds an extra delay between data and remote frames. However, we only consider data and error frames for this work and limit the following review to them.

In CAN message frames, the data field can be between 0 and 8 bytes inclusive. Senders encode the size of the data field within the data length code (DLC) field as an unsigned 4-bit integer. Although the standard does not prohibit DLC values between 9–15, CAN controllers generally ignore the excess beyond 8.

The cyclic redundancy check (CRC) delimiter and two other fields—acknowledgment (ACK) and end-of-frame (EOF)—are fixed-size. Every other field may vary dynamically due to *bit stuffing*, which is the addition of a single inverted *stuff bit* after sending 5 consecutive identical bits (possibly including a previous stuff bit). Thus, 6 identical consecutive bits is an error unless it occurs in the last 25 bits of the frame. Each frame is separated by at least 3 interframe space (IFS) bits, which are all recessive, and the next dominant bit indicates the start-of-frame (SOF) for the next message. For a successfully transmitted frame, the last bit of the ACK, called the ACK delimiter, is recessive, and the EOF is recessive. Thus, there is a minimum of 11 recessive bits between consecutive (data) frames on the bus. These 11 bits indicate the bus idle signal.

2) Fault Confinement: A key feature of CAN is its fault tolerance and error handling mechanisms, which play an essential role in the bus-off attack. We focus on two salient aspects: error states and error frames.

CAN includes three error states: error-active, error-passive, and bus-off. The state of a node is determined by a state machine, shown in Fig. 2, based on the values of its receive error counter (REC) and transmit error counter (TEC). For each error that a node observes while not transmitting, it increases its REC by 1. If an error frame occurs while a node is transmitting, it increases its TEC by 8. The REC or TEC is decremented by 1 for each successful receive or transmit, respectively.

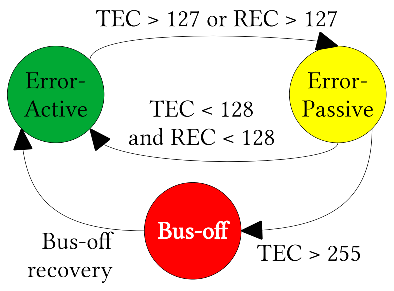


Fig. 2. CAN error state machine [5]. A node initially starts with TEC and REC of 0 and in the error-active state. If the TEC or REC exceeds 127 the node transitions to error-passive and will transmit error-passive flags when it observes an error. A node transitions back to error-active when both its TEC and REC are below 128, or it transitions to the bus-off state if TEC exceeds 255. The CAN bus standard stipulates that a node that enters bus-off can rejoin the network after 128 occurrences of the bus-free signal of 11 consecutive recessive bits. When a node recovers from bus-off it rejoins in the initial state with zeroed error counters.

An error frame indicates an error condition observed by one of the nodes on the bus. Each node that detects an error transmits either an error-active flag consisting of six dominant bits or an error-passive flag of six recessive bits, depending on whether the node is in the error-active or error-passive state, respectively. (Multiple nodes can transmit overlapping error flags; thus, the error flag may be a combination of between 6–12 recessive or dominant bits.) After arbitration, two error conditions—bit error and stuff error—are detected within a message frame. The sending node that won arbitration monitors the bus and reads back each bit on the bus, comparing with the bit it intended to transmit; a mismatch generates an error flag. Each node on the bus also monitors for more than 5 consecutive identical bits, which indicates an error with bit stuffing; multiple nodes may generate overlapping stuff error flags. Since our attack depends on generating an error flag due to a bit mismatch between what the attacker intends to transmit and what is actually on the bus, we consider only errors at the bit level and ignore errors at the frame level (e.g., bad CRC or negative ACK).

B. Bus-off Attack

The bus-off attack leverages the fault confinement mechanism of CAN to drive an ECU to the bus-off state. When an ECU reaches the bus-off state, it loses its access to the bus and cannot transmit any message. Apart from the work of Cho and Shin [5], different approaches have been used to drive a victim ECU into bus-off. Rogers and Rasmussen [25] introduce a technique that enables an attacker to push a victim into bus-off by manipulating several bits in a data frame, increasing the victim's error count. The authors assume that the attacker can arbitrarily invert the bits of the victim's data and error frames to cause an interruption and increase the victim's error counter. Murvay and Groza [26] exploit the vulnerability of the wired-AND design of the CAN bus to force a victim ECU into bus-off based on bit injection by using bit banging to control the CAN transceiver. With the ability to modify the physical representation of bits on the bus, the attacker monitors the bus and forces the transmitted bits at a specific frame location into a dominant state. The manipulation causes the sending node to detect a bit-error, transmit an error frame, and increment its error counter. Serag et al. [27] exploit the deterministic recovery vulnerability of CAN, discovered through their CANOX tool, to continuously force an ECU into a bus-off state. The attacker targets a victim's message ID and deliberately causes errors through collisions. When the victim has transitioned into the bus-off state, the attacker reinitiates the attack to prevent the victim's recovery process. Iehira et al. [28] leverage the bus-off attack to spoof messages on the CAN bus without detection by the authorized ECU. Once the victim ECU is in the bus-off state, the attacker injects spoofed messages into the bus. The attacker initiates the attack by transmitting an error frame when a victim ECU transmits a data frame on the bus. The victim detects bit-error and increases its TEC. These attacks show the vulnerability of CAN's error handling mechanism, which an attacker can leverage to drive a victim ECU into bus-off, but they all use dominant bit errors whereas our work uses a recessive bit error to generate a dominant error flag.

C. WeepingCAN Bus-off Attack

The WeepingCAN attack is a variation of the original bus-off attack by Cho and Shin [5]. One limitation of the original bus-off attack is that it exhibits detectable features that allow intrusion detection systems to flag the attack. The WeepingCAN attack improves the bus-off attack by introducing variations that ensure the attack's success without the detectable features. By disabling the retransmission of the attacker's message, WeepingCAN eliminates the first detectable feature, which relies on the cascade of retransmissions and consecutive errors. Additionally, the WeepingCAN strategy causes the attacker rather than the victim to initiate generating an error flag by introducing recessive bit errors instead of dominant bit errors. This error flag causes a bit error at the victim. This approach prevents any overlap between successful transmissions and the victim's passive error flag, thus eradicating the second feature that allows attack detection. However, the disadvantage of disabling retransmission is that it prevents the attacker's message from being successfully transmitted, potentially pushing the attacker's ECU into a bus-off state. To mitigate this risk, the attacker can identify additional messages that it can successfully transmit to maintain a TEC lower than the victim's ECU. Concretely, the attacker must transmit at least 1 more message than the victim for each injection of the attacker's message. The victim's ECU must also transmit fewer than 7 other messages between injections to keep its TEC from resetting to zero. This strategy helps the attacker maintain control while avoiding going into bus-off.

Prior to the attack, the attacker identifies a specific, periodic message v that the victim ECU transmits. To keep the attacker's TEC below the victim's, Bloom provides Eq. 1 [12], which we reproduce here:

$$\exists v \in M_{\mathcal{V}} \ s.t. \ 8 > \sum_{m' \in M_{\mathcal{V}}} \frac{v_T}{m'_T} < \sum_{m \in M_{\mathcal{A}}} \frac{v_T}{m_T}. \qquad (1)$$

Eq. 1 specifies a constraint on the victim ECU based on the period of v (i.e., v_T) for the attack to succeed, where $M_{\mathcal{V}}$ and $M_{\mathcal{A}}$ are the set of (authentic) messages transmitted by the victim and attacker, and m_T is the period of some message m. In addition to this constraint, the message v must also have a *preceded* message, which is some message that transmits immediately before v either due to winning arbitration because of higher priority or being in transmission when v is ready to transmit, i.e., interfering or blocking v's bus access. The success rate of the (original) bus-off attack without preceded messages was found to be about 1%, and reliance on preceded messages for synchronizing the attacker with the victim increases the success rate to 90–100% [5]. Engineering real-time systems exhibit the behavior of batching multiple periodic events (messages) at the same period and using periods that are multiples of other periods [29]–[31]. Thus, we expect genuine preceded IDs exist in practice that an attacker can leverage, which has been shown to be the case by Bloom [12] and by Hounsinou et al. [32].

Given that the attacker can identify some v that satisfies the above constraints, which can be done via offline analysis, then the attack can proceed. The attacker fabricates an attack message $v_{\mathcal{A}}$ that has an identical prefix (including ID) as v until a random location in the DLC or data field of the frame where v is dominant and $v_{\mathcal{A}}$ is recessive. When both v and $v_{\mathcal{A}}$ transmit synchronously, this intentional insertion of a recessive bit creates a mismatch between what the attacker's ECU intended to transmit and what it actually detects on the bus since the message with the dominant bit will transmit. Consequently, the discrepancy causes a recessive bit-error for the attacker. This error causes the attacker's controller to transmit an error-active flag that increases the attacker's TEC by 8. The victim sees the error flag, increments its TEC by 8, and attempts retransmission.

The attack proceeds in an iterative fashion. First, the attacker synchronizes with the victim ECU by using the periodic approach described in the original bus-off attack [5]. Then, the attacker injects $v_{\mathcal{A}}$ during the preceded message of v with retransmissions disabled. Bloom [12] identifies two approaches to disabling retransmissions: (i) disable automatic retransmissions for all messages, and (ii) abort transmission on transmit error. The automatic disabling approach requires turning retransmission off prior to enqueueing $v_{\mathcal{A}}$ and on again after its transmission fails. Aborting on transmit error uses the CAN controller's capability to generate an interrupt on a TX error and to flush the CAN message buffer within the bus idle time before the retransmission starts. The net effect is that, depending on the other messages sent by the victim and attacker ECUs, the victim will reach the bus-off state without any transmission of the attack message succeeding on the bus. Hence, the attack is considered stealthy. Note that, however, satisfaction of Eq. 1 limits the applicability of the attack; also, depending on how tightly the equation is satisfied, the attacker may need to employ a *skipping attack strategy* that purposefully avoids injecting $v_{\mathcal{A}}$ to allow the attacking ECU to recover its TEC faster than the victim does [12].

Groza and Murvay [33] disregard the bus-off attack for several reasons, including its detectable features and the fact that (by their calculations) the victim node can recover and rejoin the network within 1.5 ms. Although we agree about

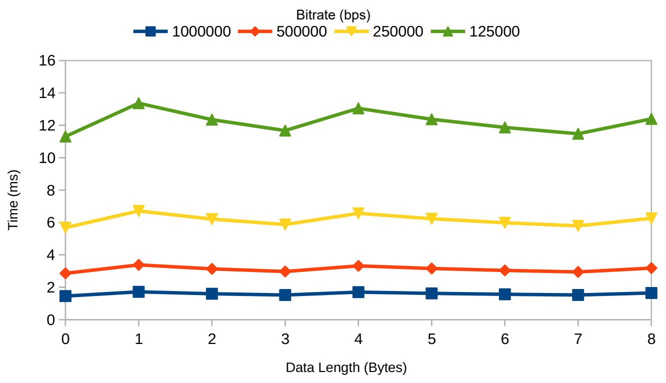


Fig. 3. Realistic CAN Bus-off Recovery Time with 50% Bus Utilization. The latency from start to finish to recover from the bus-off state with increasing data lengths at commonly used CAN bitrates (125000 bps to 1000000 bps).

the detectable features, the analysis of victim recovery time is optimistic: the bus-off recovery time varies depending on bus utilization, bitrate, and the node's recovery policy [34]. The estimate of 1.5 ms is feasible for the 1 Mbps bus speed below 40% utilization (for a 500 kbps bus below 30% utilization, a 250 kbps bus below 10% utilization, and a 125 kbps bus below 5% utilization, respectively) and an automatic reset on bus-off. However, the automotive industry generally discourages using an automatic reset—software should initiate the reset, e.g., in an interrupt handler or application logic, and often hardware does not even support an automatic reset capability [35] [36, p. 86]. The bus-off recovery occurs after 128 transmissions of 11 consecutive idle bits, which is guaranteed to happen between consecutive messages. However, when the bus is heavily used, there may not be additional idle bus signals beyond the inter-frame space between messages that contribute to a node's bus-off recovery.

Fig. 3 shows how the recovery time increases with (average) message data length and bitrate with a 50% bus utilization assuming uniformly distributed (non-bursty) messages. The number of idle bits in the gap between each data frame is $d*(1-u)/u$ as a function of the utilization u and data frame size d, where u represents the utilization of the bus, which is calculated as the percentage of the total bits transmitted out of the maximum bus bandwidth, i.e., the elapsed time (wall clock) divided by the bitrate of the bus. Without using extended IDs, frame sizes vary between 44 to 108 bits based on the DLC of 0 to 8 bytes, plus additional bits for stuffing; the maximum CAN data frame size is 129 bits, although designers avoid such maximal stuffing in practice. With 8 byte messages at 1 Mbps bitrate the CAN bus-off recovery time is 1.6 ms, while slower speed busses recover with proportionally longer times.

IV. Attack Optimizations

We now describe three improvements to the Weeping-CAN [12] attack that increases the attack effectiveness and the range of victim targets. First, we introduce *zero-phase synchronization* as a better method to synchronize the attacker and victim ECUs than the periodic approaches used in prior work. Second, we introduce an alternative method to disable

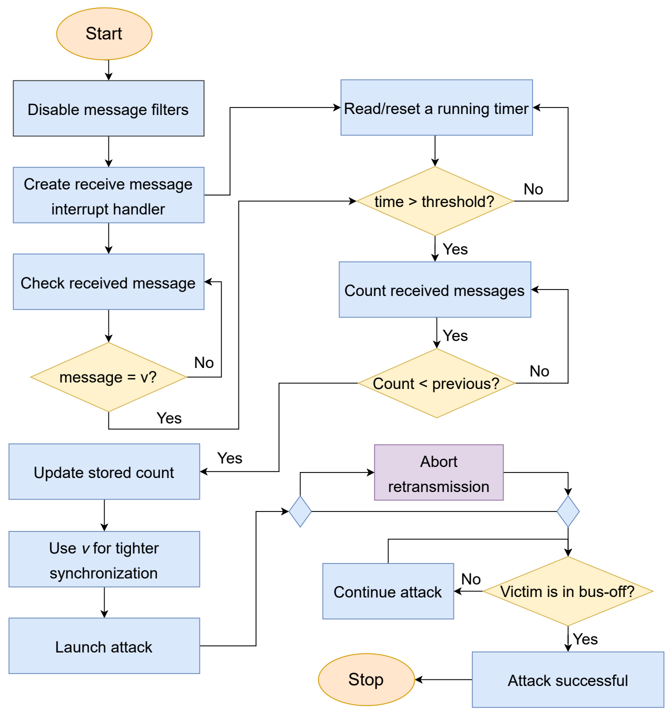


Fig. 4. Flow diagram of bus-off attack optimizations showing zero-phase synchronization and abort transmission on receiving preceded message.

retransmissions of $v_{\mathcal{A}}$ that enables the attack to be launched regardless of the type of CAN controller used by the attacker's ECU. Third, we identify a novel *transitive* attack strategy that allows an attacker to attack multiple ECUs in succession.

A. Zero-Phase Synchronization

The original and WeepingCAN bus-off attacks used periodic synchronization of attack injections [5], [12]. The periodic approach uses the transmission of v (or its preceded message) to program a periodic timer that will enqueue the attack message each time it gets triggered. However, this approach ignores that the attacker can adapt to resynchronize with the victim in response to online observations of bus behavior. We use such an adaptive approach in our attack implementation that synchronizes to a zero-phase transmission of v.

Our approach is inspired by the strictly periodic approach. However, instead of synchronizing with some arbitrary transmission of v, we attempt to re-synchronize on v when it transmits with less bus interference or blocking. Such transmissions occur with a smaller phase from v's period, and the ideal case is when there is no bus interference, and the transmission occurs with zero-phase (i.e., no offset, aligned) with respect to the period. Also, jitter—variations in the true or expected periods—is ignored due to measurement constraints, and we conspicuously expect no significant impact on the performance of this approach.

As shown in Fig. 4, the attacker accomplishes zero-phase synchronization by disabling message filters and creating a receive message interrupt handler that is active for all received messages. In this way, the interrupt handler will fire for every message on the CAN bus. The handler logic determines if the received message is v or not, and whether the received message was transmitted while the bus was idle. To track the bus idle time, the interrupt handler reads and resets a running timer (counter) for each interrupt. If the timer exceeds the threshold needed to transmit the received message, then we assume the bus was idle. The handler keeps a count of how many messages are received without an idle bus. The first receipt of v stores the count of messages since the bus was idle, and each subsequent receipt of v updates the stored count with the current count if it is smaller. Each time the stored count is updated indicates greater confidence that v encountered less bus interference, and therefore has a smaller phase and should be used for tighter synchronization.

We use the receipt of v to program a periodic timer that controls the injection of $v_{\mathcal{A}}$ with the (known) period of v minus an offset to account for v transmission time and software processing overhead. The periodic timer triggers an interrupt during the transmission of the message that the attacker expects precedes the transmission of v, and the attacker will use this interrupt handler to inject $v_{\mathcal{A}}$. This approach does rely on there being *some* preceded message that transmits before v, but it does not need to be unique. A problem is that if the bus is idle when the timer fires, then $v_{\mathcal{A}}$ will transmit before v does, and therefore will be detectable. To avoid this detection, an attacker could track the time between received messages and skip an injection if the timer indicates the bus might be idle. (We did not, however, implement such a strategy.) Furthermore, attackers can observe arbitrarily many periodic transmissions of v keeping track of the reduction in phase until it is stable.

A challenge for this approach is that the interval may fluctuate due to bus interference and jitter. We posit, however, that the interval will only *increase* due to these factors. If v transmits later than expected, it will do so because of either interference by higher priority messages or a substantial jitter. In case of high priority interference, $v_{\mathcal{A}}$ will be subjected to the same interference and will therefore still synchronize successfully. In case of large jitter, it is possible the transmission of $v_{\mathcal{A}}$ succeeds before v transmits. The jitter would need to be large enough that v does not attempt to arbitrate immediately after the preceded message overlapping with the enqueueing of $v_{\mathcal{A}}$.

Zero-phase synchronization requires the capabilities to: receive all messages (disable filters), trigger interrupts on any message, and program two timers: one periodic and one running timer. All the microcontrollers and CAN controllers we have examined support these capabilities from software and thus are viable within the assumptions of the threat model.

B. Abort Transmission on Receiving Preceded Message

We discovered another approach to disable retransmissions of $v_{\mathcal{A}}$ that is even more widely supported by commercial CAN controllers than the two approaches described in the original WeepingCAN attack [12]. In particular, disabling automatic retransmissions is only available on CAN controllers that support time-triggered CAN (TTCAN), while aborting transmissions on transmit error is actually the most reliable approach for attack (in the absence of other transmit errors), but not all CAN controllers may support this approach, or may support it imperfectly. Our novel approach instead is supported by all CAN controllers, and therefore removes a significant limitation of the prior work.

Every CAN controller has the ability to generate an interrupt when a message is received. Furthermore, the interrupts can be filtered by ID, or the ID that caused the interrupt may be queried. This ability can be used to generate receive interrupts on the set of preceded messages for v, assuming that some genuine preceded messages exist. By coupling this interrupt with the zero-phase synchronization strategy that already uses the receive interrupts as shown in Fig. 4, the adversary can simply assume the next receive interrupt after enqueueing $v_{\mathcal{A}}$ for transmission is caused by the preceded message. (In case the attacking ECU transmits the preceded message, then the attacker can instead use a transmit succeeded interrupt to identify when the preceded message's transmission is successful. Without loss of generality, we say the preceded message is received.) The success of this approach depends on whether the interrupt is in fact triggered during the actual preceded message of v, and therefore the abort happens during the transmission of $v_{\mathcal{A}}$.

When the (presumed) preceded message is received, the attacker issues a request to abort the transmission of messages in the CAN controller's transmission buffers. The key here is to delay requesting the abort until *after* the first transmission of $v_{\mathcal{A}}$ has already started but not finished. Typical behavior for an abort request does not preempt the message that is currently transmitting because CAN controllers have their own internal processor to transmit the message from the message buffer. Note that this behavior is in contrast to requests to reset or halt the controller, which preempts the internal processor and therefore aborts the current transmission. Aborting a specific message while it is in transmission however does prevent retransmissions of that message. The behavior of aborting messages is dependent on the CAN controller implementation, but it is consistent across common controllers that the message is allowed to complete transmission if it started prior to the abort request reaching the controller. Hence, we program a delay of about 20 bit times to allow for the bus idle time, SOF, and arbitration of $v_{\mathcal{A}}$ to occur, and then flush the transmit buffer for the attack message.

C. Transitive Attack Strategy

The attacker can satisfy the requirement of Eq. 1 by targeting an ECU to compromise initially based on it sending messages at a higher frequency or in a greater quantity than the intended victim. Even if no unique such ECU can be found, we find a clever attacker can still *transitively attack* its victim by first using a WeepingCAN bus-off attack against a set of intermediate victim ECUs that individually satisfy Eq. 1 before attacking the final victim. The basic idea of this attack is that the attacker will iteratively bus-off and then *masquerade* as one or more ECUs by transmitting their authentic messages prior to conducting a bus-off attack against the final victim. Thus, the attacker will increase the number of messages it transmits and therefore recover its TEC faster.

The transitive attack requires the attacker to identify a set of ECUs $\mathcal{A}^*$ that satisfy Eq. 1 with replacement of $M_\mathcal{A}$ by $M_{\mathcal{A}^*}$. To maintain low detectability, the attacker should also be able to transmit reasonable forgeries of the messages belonging to the ECUs in $\mathcal{A}^*$. We assume that, based on the offline analysis, an attacker can construct such forgeries, and can determine a set of ECUs for conducting the transitive attack. We also assume that an ECU that goes to bus-off does not recover—if it does, then the attacker may need to detect the recovery and attack it again. Note however that transmitting forgeries in the transitive attack may reduce stealthiness and increase the detectability of this attack: the attacker will inject non-authentic message frames that rightly belong to some other ECU in $\mathcal{A}^* \setminus \mathcal{A}$.

Eq. 1 can be generalized to the case that the attacker targets multiple messages in the set $M_\mathcal{V}$ for conducting the bus-off attack. In this case, the attacker could also rotate its attack to target different messages in $M_\mathcal{V}$ to further obfuscate the attack behavior. We observe that the existence of genuine preceded IDs can be coupled with the transitive attack strategy by putting the ECU that transmits the genuine preceded ID in $\mathcal{A}^*$. Then the attacker can inject the preceded message at a convenient time of its choice. We provided an evaluation of this approach under different bus speeds and varying bus loads to attest to its viability.

Example (Transitive Attack): Consider a hard case when $|M_\mathcal{V}| = |M_\mathcal{A}| = 1$, i.e., the victim and attacker both transmit one authentic message, and the attacker's authentic message is sent with the same period as the victim's. Eq. 1 does not hold as $1 \nless 1$. Suppose however that there exists some ECU $\mathcal{R}$ that transmits a single message with double the period of the attacker's message, i.e., $m \in M_\mathcal{R}$ and $m_T = 2.0 * v_T$. Then the attacker can target $\mathcal{R}$ with a WeepingCAN attack, because Eq. 1 holds as $8 > 1 < 2$, although the attacker has to use a skipping attack strategy. Now $\mathcal{R}$ is in $\mathcal{A}^*$ and $|M_{\mathcal{A}^*}| = 2$ with two messages having periods equal to $1.0 * v_T$ and $2.0 * v_T$. The attacker can attack the intended victim now, using a skipping attack strategy, because $1 < 1.5$.

V. Evaluation

We use a benchtop CAN bus for experimental validation. The CAN benchtop setup uses off-the-shelf hardware and in-house/open-source[1] software. Fig. 5 depicts the bus setup. The platform includes five microcontrollers: two BeagleBone Black (BBB), and three TM4C129EXL (TM4C) boards. BBB and TM4C boards have built-in hardware CAN controllers integrated with their processor chips. We connected the BBBs and TM4Cs to SN65HVD23x CAN transceivers. The transceivers are connected via 3.3v CAN implemented in a breadboard.

We flashed the BBB with updated U-Boot (2018.09) firmware and Debian Stretch 9.5 (BeagleBoard.org Debian Image 2018-10-07). We use Python with SocketCAN to interact with the CAN bus using version 3.2.0 of python-can that provides support for multiple periodic messages. Python on Linux gives the BBB jitter especially to start communicating,

[1]GitHub URL with open-source license will be provided.

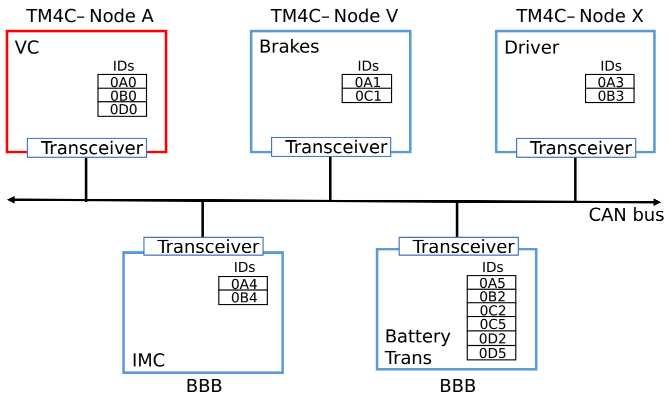


Fig. 5. 5-node CAN Benchtop Setup. The TM4C Node A acts as the attacker in all experiments.

but periodic messages are supported by the kernel so they exhibit reasonably predictable, regular message timing after the first transmission.

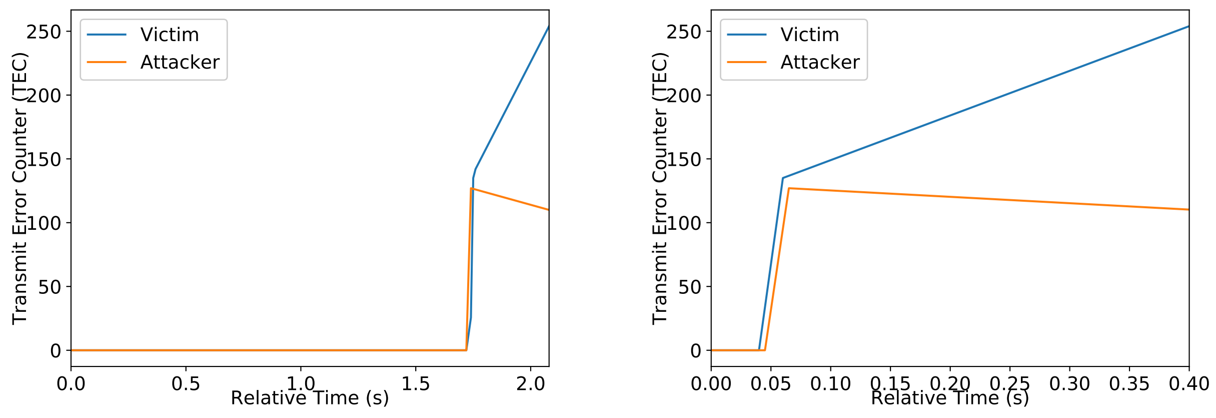
We run custom baremetal code on the TM4C boards that includes both normal and attacker behavior. Baremetal provides the least jitter, the most control over the software behavior, and easily programmed interrupt handlers.

1) F1 Attack Detector: We used the approach described by Cho and Shin [5] to identify consecutive errors (feature F1) as indicative of a bus-off attack. We report the performance of this F1 attack detector in each experiment.

A. Bus-Off Attack Experiments

We conducted two experiments using the original bus-off attack with dominant bit errors and a reproduced synthetic benchmark from prior work [5], [12]. The first experiment examines zero-phase synchronization, and the second evaluates across approaches to disable retransmissions. We implement a 500 kbps CAN bus in our benchtop setup with nodes X, V, and A in TM4C boards to recreate a similar setup as the prior work. Node X sends messages with ID 7 and 9 at a period of 10 ms, and node V sends a message with ID 17 when it receives the message with ID 7 thus also having a period of 10 ms The attacker is implemented on node A. The message ID 9 is the genuine preceded ID of 17. Both messages are transmitted with a period of 10 ms.

1) Zero-Phase Synchronization: Here, we examine the efficacy of zero-phase synchronization with respect to periodic synchronization at aligning the attacker's injected message with the victim's normal transmission. The attacker relies on the approach described in Section IV-A to synchronize the transmission of its attack message from node A with the transmission of ID 17 by node V. A uses an interrupt handler to track the transmissions on the bus, and programs a timer to 10 ms minus the transmission time of message ID 17 and one-half the transmission time of a 1-byte message at the 500 kbps bus speed. This extra half transmission time ensures that the timer fires before a message with ID 17 is being transmitted. This precaution allows adequate time to enqueue a message that will synchronize seamlessly with the subsequent transmission of message ID 17. The message with ID 17 never transmits with 0-phase, but always transmits with the same interference caused by IDs 7 and 9.



(a) zero-phase

(b) periodic

Fig. 6. TEC increase of original bus-off attack with zero-phase and periodic synchronization.

We examined 100 bus-off attacks using the zero-phase synchronization approach. We also reproduce the direct periodic approach proposed by Cho and Shin [5] and used in the original WeepingCAN [12]. Fig. 6 shows the growth of TEC with both synchronization approaches averaged over the 100 trials. (All results were nearly identical.) As expected, the F1 detector identified every attack with each synchronization approach and detected the 15 retransmissions caused by the dominant bit error.

The zero-phase synchronization takes longer to align with the victim as it waits for a transmission of v with minimal interference. Note that the direct periodic approach works well due to the presence of genuine preceded IDs and extremely low bus utilization in this benchmark. These results show that zero-phase and periodic synchronization are both viable options when there do exist genuine preceded messages. The zero-phase approach takes longer to synchronize with the victim because it waits to observe a transmission of v that has minimal bus interference. However, as we show in the following experiments, the quality of the synchronization achieved by the zero-phase approach can be better than the periodic approach.

2) Disabling Retransmissions: We evaluate the three approaches for disabling retransmissions of $v_{\mathcal{A}}$: (1) disable automatic retransmissions, (2) abort transmission on transmit error, and (3) abort transmission on receiving preceded message. We implemented each approach with the TM4C boards. They have TTCAN capability and support interrupts on bit or stuff errors and aborting messages selectively.

We again use the Synthetic Benchmark with the original bus-off attack. Node X reliably transmits both preceded messages of 17, therefore attacker and victim can be perfectly synchronized, which isolates the variable of retransmission approach. We examined 100 bus-off attacks with each approach and measured the number of successful attacks and visible transmissions by the attacker based on the TEC increases on each node.

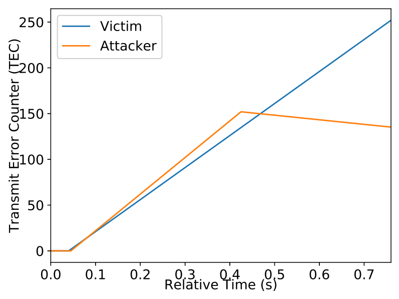
Our approach to disable automatic retransmissions and aborting transmission on transmit error is similar to prior work [12]. To disable automatic retransmissions we globally configure the CAN controller during its initialization, and to abort the transmission on transmit error, we clear the attack message from its transmit message buffer in the CAN interrupt handler on an error code. For aborting the transmission on receiving the preceded message we clear the transmit message buffer after the interrupt handler returns to normal processing with a slight processing delay of approximately 15 bit transmission times to ensure the attack message is in transmission.

Fig. 7 shows the increase in TEC for victim and attacker when retransmissions are disabled, which shows that our new approach to disable retransmissions when the preceded message is received has similar attack performance as the prior approaches. More important, Fig. 8 shows the number of transmissions of $v_{\mathcal{A}}$ by the attacker.

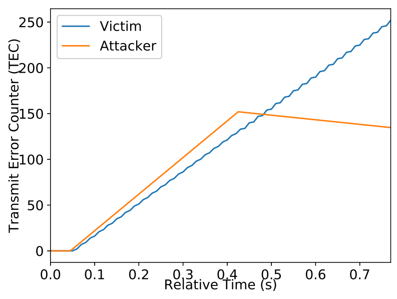
The attacker transmits messages successfully because of the dominant bit error: both disabling automatic retransmission and aborting on the transmit error result in 18 successful transmissions, while aborting on the receipt of the preceded message results in zero successful transmissions. This result shows that in fact the dominant bit error injection can potentially be used stealthily with the strategy of aborting transmission upon receiving the preceded message. In addition, the F1 attack detector did not identify any of the attacks when using these approaches to disable retransmissions.

B. WeepingCAN Attack Experiments

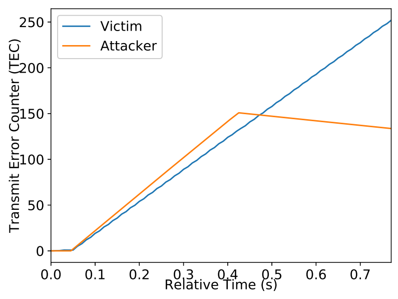
We conducted two experiments using the WeepingCAN attack with the improvements introduced in this paper. These experiments use the modified SAE benchmark described by Tindell et al. [37]. The original benchmark is not specified for CAN, and the authors did not suggest CAN IDs to use, so we arbitrarily use the rate monotonic approach to define message priorities inversely to their periods (shorter period, higher priority). Although message sizes are specified, the contents are not; we used fixed data fields in our experiments. Each subsystem's periodic messages are transmitted synchronously from the same clock, and therefore the subsystem enqueues messages it transmits in batches. Ties caused by identical periods are broken as follows in decreasing priority order:



(a) auto-disable



(b) abort transmit error



(c) abort receive preceded

Fig. 7. TEC increase of original bus-off attack without retransmissions.

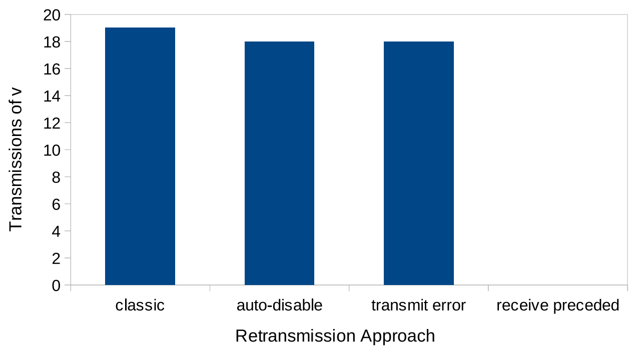


Fig. 8. Transmissions of the victim's ID with four approaches for retransmissions using the original bus-off dominant error injections. Classic uses retransmissions and the other three approaches disable retransmissions.

TABLE I
PARAMETERS FOR MODIFIED SAE BENCHMARK [37]

Sender	CAN ID (hex)	Size (B)	Period (ms)
VC	A0	1	5
	B0	6	10
	D0	1	1000
Brakes	A1	2	5
	C1	1	100
Battery	B2	1	10
	C2	4	100
	D2	3	1000
Driver	A3	1	5
	B3	2	10
IMC	A4	2	5
	B4	2	10
Trans	A5	1	5
	C5	1	100
	D5	1	1000

vehicle controller (VC), brakes, battery, driver, inverter/motor controller (IMC), and transmission (trans). Table I shows the parameters for the modified SAE Benchmark arranged according to the sending subsystem in priority order. We placed the VC, Driver, and Brakes each on a separate TM4C, the IMC on its own BBB, and we pair the Battery and Trans subsystems together on one of the BBB boards.

1) Skipping Attack with Zero-Phase Synchronization: We conducted 75 WeepingCAN attacks skipping 5 iterations per attack with VC attacking Brakes using zero-phase and periodic synchronization. With zero-phase synchronization, 86.7% of the attacks succeeded in pushing the Brakes to bus-off, while periodic synchronization succeeded in 78.1% of the attacks. The minimum and median number of transmissions

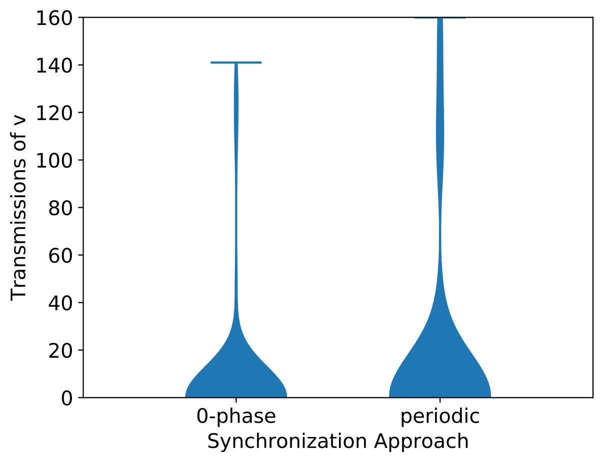
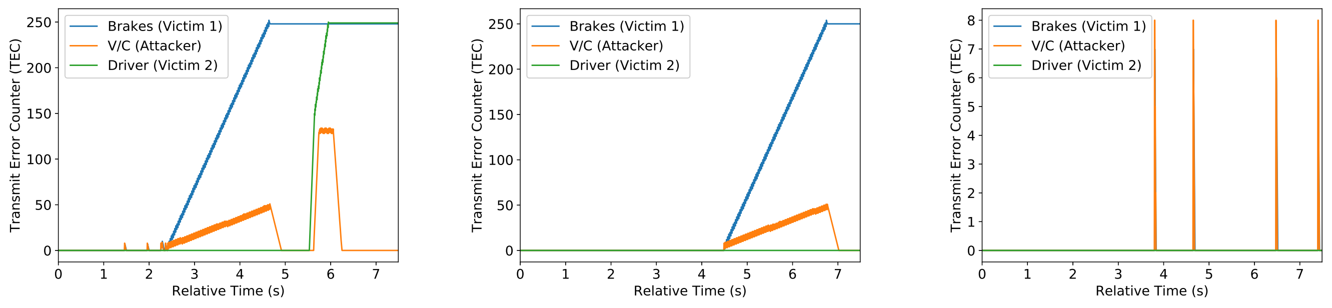


Fig. 9. Transmissions of the victim's ID with the SAE Benchmark and skipping 5 messages per injection. Each violin shows the distribution of transmission counts over 75 attacks.

of $v_{\mathcal{A}}$ was 0 for both cases, with a maximum of 141 and 160 transmissions, respectively. The trials that failed did not manage to synchronize with the victim correctly and resulted in mis-timed injections. Figure 9 shows a violin plot depicting the distribution of transmissions of $v_{\mathcal{A}}$. Zero transmissions indicates good synchronization and stealthiness, while higher numbers are less stealthy and have been caused by poor/lost synchronization between the attacker and victim. The F1 attack detector identified 6.7% of the attacks with zero-phase synchronization and 2.7% of the attacks with period-based synchronization. Fig. 9 shows the number of transmissions prior to bus-off for the SAE Benchmark was often but not always 0. These results show that, although the zero-phase synchronization achieves a higher attack success rate, it may reduce stealthiness. Based on preliminary investigation, we suspect this contradiction is due to early transmissions of the attack message with the zero-phase approach, and suggest that an attacker who can carefully tune synchronization to the bus behavior can likely achieve much better attack results.

2) Transitive Attack: Here we use the SAE Benchmark to evaluate using the transitive strategy to disable ECUs that are otherwise unassailable by the original WeepingCAN attack. Based on Eq. 1, the VC, Driver, and IMC subsystems cannot be directly attacked, but any subsystem that first compromises the Battery, Brakes, or Trans subsystem can then compromise



(a) Success (b) Partial (c) Fail

Fig. 10. Representative TEC increase of successful and unsuccessful transitive attacks.

any other subsystem by the transitive attack strategy (with skipping). We use the VC subsystem to first conduct a stealthy WeepingCAN attack against the Brakes, as in the preceding experiment, and then launch the transitive attack against the Driver subsystem, which does not require any skipping when combining the VC and Brakes messages. Fig. 10 shows representative results for a successful transitive attack, an attack that succeeds against the Brakes but fails against the Driver, and an attack that fails. The primary reason for failure is loss of synchronization with the victim, which is a challenge in the SAE Benchmark due to high bus utilization.

We conducted 200 attacks with each approach for disabling retransmissions. Table II summarizes the key results of attack success rates and number of transmissions by the attacker with respect to the specific approach for disabling retransmissions. Fig. 11 shows the violin plots for the number of injections of $v_{\mathcal{A}}$ prior to the bus-off of the victim, which again is an indicator of the stealthiness of the attack. The original (Classic) bus-off attack using periodic synchronization—but with recessive injections—is shown as a baseline for comparison; we found that it succeeds roughly as often in disabling one ECU, does poorly with the transitive step, and has a high count of detectable transmissions of the victim message prior to attack success. Of course, it also exhibits high F1 detection rates.

An important point here is that disabling automatic retransmissions yielded zero successful attacks (regardless of synchronization approach), which we attribute to the excessive interference in the SAE Benchmark and that this approach will not retransmit the attack message if it loses arbitration. The other two approaches show adequate yet imperfect attack capability. Zero-phase synchronization can achieve higher attack success rates than periodic synchronization, although not consistently. Lower attack success rates tended to exhibit more transmissions of the injected message by the attacker and higher F1 detection rates, which we again suspect is due to poor synchronization with the victim. Fig. 11(a) shows the transmissions of the brake message by the attacking VC ECU, and Fig. 11(b) shows transmissions of the driver message.

3) Transitive Attack Under Different Bus Speeds: Using the SAE benchmark, we conduct the transitive attack in Section V-B2 under different bus speeds. In this experiment, we implemented the zero-phase synchronization and abort message transmission when transmit error is detected. We first carry out a stealthy attack with the VC against the Brakes before launching a transitive attack against the Driver subsystem. Fig. 12(a) shows the representative TEC growth of the attacker and victims using 125 Kbps bus speed, Fig. 12(b) shows the result when the bus speed is 250 Kbps, and Fig. 12(c) depicts the result of 500 Kbps bus speed. Although the attack against the Brakes started a bit later for 500 kbps bus speed, the three experiments show identical TEC changes. In each experiment, the attacker's TEC remains in the error-passive region, then recovers while driving the victim ECUs to the bus-off state. These results indicate that the transitive attack approach applied to varying bus speeds yields similar results.

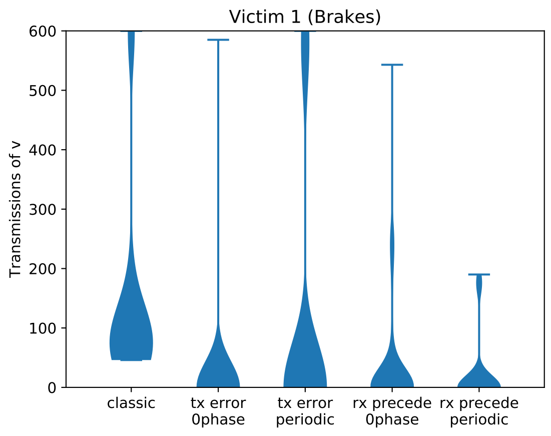
4) Transitive Attack Under Different Bus Loads and Varied Message Sets: Here, we implement a 500 kbps CAN bus, varying the number of messages on the bus from 800 to approximately 1400 messages per second, and conduct WeepingCAN attacks using the transitive strategy to drive victim ECUs to the bus-off state. To achieve this, we changed the modified SAE benchmark message periods as shown in Table III and performed 75 attacks using zero-phase synchronization. We use the VC to stealthily attack the Brakes before employing the transitive strategy to attack the Driver subsystem. In addition, we randomly varied the messages transmitted by each ID before launching the attacks against the victims. We evaluate the delay experienced by the victims before each successful attack under each bus load. For each experiment, the results are nearly identical. Fig. 13 shows the average delay of each victim ECU before entering the bus-off state. We noticed that the time to bus off state increases with increasing bus load and posit that this is due to higher priority messages winning arbitration before the victim's messages get transmitted on the bus. The success of the bus-off attacks under various bus loads and different message sets demonstrates the feasibility of using the transitive attack approach against ECUs.

C. Performance of In-Vehicle IDSs Against the Transitive Attack

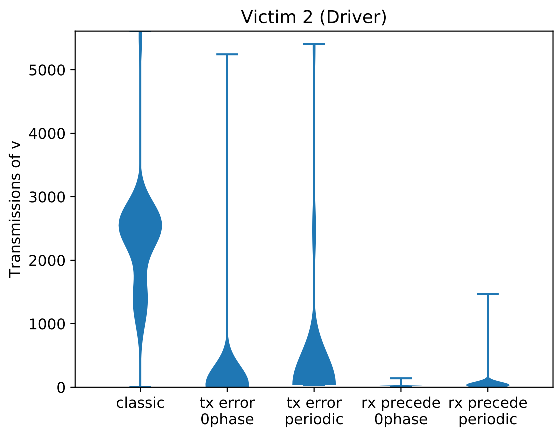
Existing defenses based on the timing of messages, statistical thresholds, and machine learning (ML) based have been proposed for detecting attacks on the CAN bus [38]–[41]. These IDSs use the inter-arrival time of CAN signals or the frequency of messages over a specific time window to identify anomalies on the bus. Although these IDSs have shown positive results in detecting attacks such as denial

TABLE II
TRANSITIVE ATTACK SUCCESS RATES AND STEALTHINESS

Retransmit Policy	Brakes Bus-Off	Driver Bus-Off	VC Transmits min	VC Transmits med.	F1 Det. Rate Brakes	F1 Det. Rate Driver
Classic Bus-off	84.5%	19.5%	46	2604	98.0%	85.5%
Disable Automatic	0%	0%	0	188	N/A	N/A
TX Error, 0-Phase	90.5%	84.5%	0	38	0.5%	0.0%
TX Error, Periodic	80.5%	74.0%	38	38	0%	1.5%
RX Preceded, 0-Phase	95.8%	74.1%	0	0	0.5%	11.4%
RX Preceded, Periodic	91.0%	97.5%	0	0	0.5%	0.0%

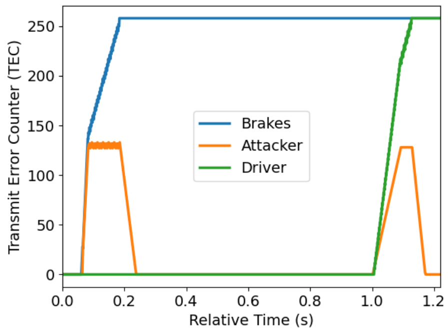


(a) Brakes Victim

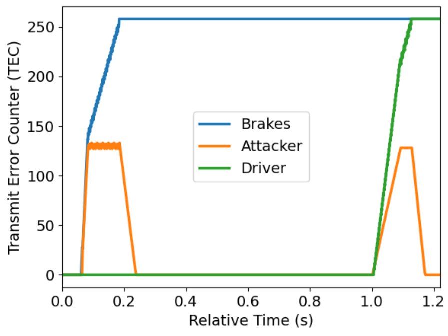


(b) Driver Victim

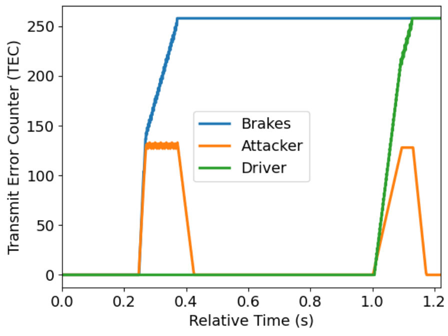
Fig. 11. Transmissions of the victim message by the attacker with the SAE Benchmark and a transitive attack strategy. Each violin shows the distribution of transmission counts over 200 attacks.



(a) 125 Kbps



(b) 250 Kbps



(c) 500 Kbps

Fig. 12. Representative TEC increase of successful transitive attacks under different bus speeds.

of service, fuzzy, and spoofing attacks, they are unreliable when deployed against our transitive attack. We implement the transitive attack against timing-based IDSs using the inter-arrival time and frequencies of messages. We also analyzed the attack's performance against the ML detection approach based on the recurring patterns of message IDs and statistical-based IDSs that monitor the entropy change in CAN IDs [40], [42], [43]. We implemented the 500 Kbps CAN bus and performed 100 different attacks with zero phase synchronization. We find that all the IDSs cannot distinguish between legitimate and attack messages. In addition, the IDSs are unable to detect when the brakes or driver subsystems move into a bus-off state. Since the proposed attack uses zero-phase synchronization to align the attack message with the transmission of the victim, these IDSs are unreliable for accurately detecting deviations caused by WailingCAN on the bus.

D. Secure Transceiver Countermeasure

A promising countermeasure to CAN attacks is to use secure CAN transceivers such as the NXP TJA115x [44]. Secure transceivers are in early stages of production, and it is not clear if they will be adopted ubiquitously by industry.

We attached a secure CAN transceiver (prototype) to the attacker VC node in the CAN benchtop setup replacing the SN65HVD23x CAN PHY. We configured the transceiver with transmission pass-list (TPL) filters to allow the attacker to transmit only its authentic message IDs. The TPL generates an error flag in case a message transmission completes a full frame (through the CRC) successfully with an ID that is not in the allowed list. We attempted the following attacks: a classic bus-off attack with the dominant bit-error injection, the original WeepingCAN attack with recessive bit-error injections and disabled retransmissions, and the transitive attack introduced

TABLE III
PARAMETERS FOR MODIFIED SAE BENCHMARK FOR DIFFERENT BUS LOADS

Sender	CAN ID (hex)	Size (B)	Period (ms)			
			800 messages	1000 messages	1200 messages	1400 messages
VC	A0	1	5	5	5	5
	B0	6	100	10	10	10
	D0	1	1000	1000	1000	1000
Brakes	A1	2	5	5	5	5
	C1	1	1000	100	100	100
Battery	B2	1	10	10	10	10
	C2	4	100	100	100	100
	D2	3	1000	1000	1000	1000
Driver	A3	1	5	5	5	5
	B3	2	1000	100	10	10
IMC	A4	2	100	10	5	5
	B4	2	1000	100	100	10
Trans	A5	1	10	10	10	5
	C5	1	100	100	100	100
	D5	1	1000	1000	1000	1000

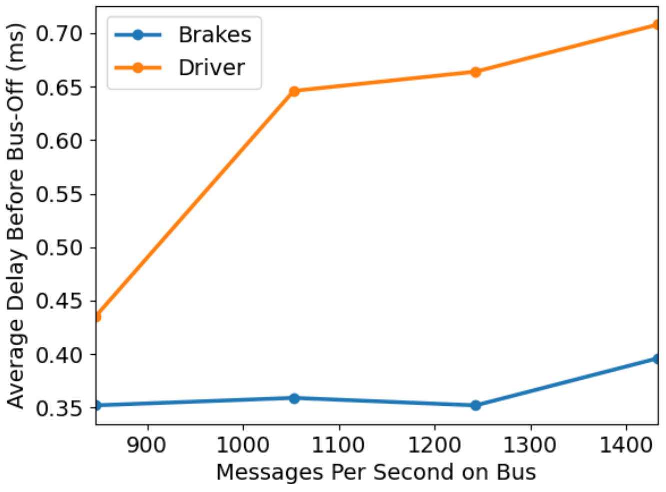


Fig. 13. Average delay of victims before bus-off

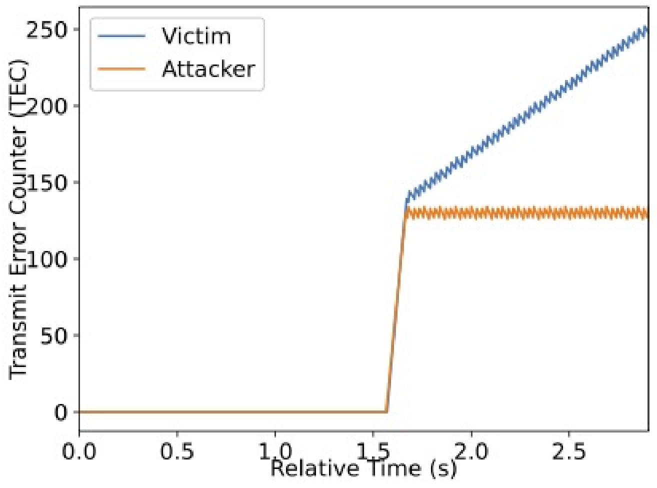


Fig. 14. TEC Growth for WeepingCAN Attack with secure transceiver on attacker node using transmission pass-list filters.

in this paper. We found that the secure transceiver prevents the classic bus-off attack from working, but just setting the TPL filters does not prevent the original WeepingCAN bus-off attack. Fig. 14 shows the increases in the attacker and victim TEC for the WeepingCAN attack when the TPL filters are in place. (This example used the zero-phase synchronization with abort on receiving preceded message and skipping one message per injection, but other attack configurations are similar.) Instead of going to bus-off, the attacker reaches the error-passive state and stays just above the passive threshold going between TEC 134 and 127 while continuing to cause bus errors synchronized with the victim. Eventually, the victim reaches the bus-off state. Despite the success of a direct WeepingCAN attack, the attacker is unable to employ the transitive attack strategy, because the TPL prevents spoofing the first victim's messages. Thus, the attacker ends up in the bus-off state while attacking the second victim.

VI. DISCUSSION

The optimizations to WeepingCAN require similar capabilities as the original attack and our approach is generalizeable: the assumptions of the threat model and required capabilities for the attack are satisfied by all modern vehicles that use CAN. Using the techniques we have introduced in this paper, an attacker can improve the success rate of the bus-off attack while reducing detectability. In addition, attackers can combine these optimizations to produce even more devastating and effective attack scenarios.

We encourage industry to develop and adopt secure technology in the transceiver logic, which is uniquely capable of implementing reference monitor functions between a compromised microcontroller and the bus. For protection against bus-off attacks, it seems that the secure CAN transceivers must be used by all nodes that an attacker may compromise.

Another hardware-based approach that may work to detect these attacks due to the transmission of active error frames is CopyCAN [45], which uses a custom CAN controller to track protected ECUs' TECs and detect if an ECU reaches the bus-off state. This approach might detect the effect of the stealthy bus-off attack because the error flags are visible on the bus, but it does require special hardware. Interestingly, CopyCAN does not detect the original bus-off attack because the passive error flags are not observable on the bus. A clever attacker might switch between recessive and dominant error injections to hide from such detection.

MAuth-CAN [46] is a CAN bus authentication mechanism that aims for resilience of the authenticator from bus-off attacks using off-the-shelf hardware. The authors propose the use of two CAN interfaces and switching from one to the other in case the TEC exceeds 96. In this way, the attacker's TEC will reach 128 and become error-passive before the authenticator. The authenticator furthermore can prevent the fabrication of messages following a bus-off attack against other CAN nodes, although it does not prevent or detect the bus-off attack itself. It is also not clear if the MAuth-CAN authenticator is resilient to the less-greedy WeepingCAN approach with attack injection skipping and the proposed transitive attacks. Also, MAuth-CAN copies the remaining messages in the transmission and reception buffers of the affected CAN controller to the micro-controller whenever there is a switch from one CAN interface to another. However, with new messages being received from the bus while copying from one buffer to the other, the absence of an explicit buffer management technique leads to priority inversion and latency issues for incoming messages.

As with the original bus-off attack, physical-layer mechanisms might detect the simultaneous transmissions of identical bits by multiple ECUs [47]–[52]. Adoption of these mechanisms however may be challenging due to the need for custom hardware, and they can be attacked themselves [9], [11]. One open problem for such mechanisms in the presence of both stealthy and original bus-off attacks however is how to know which ECU to blame for the attack. Another possibility is to combine the physical-layer attacker identification with active error flag transmission by detecting that a single ECU (based on its physical characteristics) is sending repeated active error flags during transmission by another ECU. This combination of techniques may be prone to false positives due to noisy bus conditions, and it still requires special hardware to implement.

In designing countermeasures for the bus-off attack, intrusion detection systems that can recognize essential features that adequately represent the relationship among ECUs may help detect the improved bus-off attack. One way to recognize these features may be to model the probability of an ECU going to bus-off. Once modeled, standard optimization or classification techniques can be used to determine parameters or features for an IDS. In addition, machine learning techniques can be explored to develop a predictive model that can detect real-time bus-off attacks. The model can be trained on the historical data to recognize the relationship among ECUs and identify the patterns and anomalies in message transmission behavior that indicate attack-induced bus-off states.

Although we have ignored jitter and normally occurring faults in our model—as jitter is a feature of the clock signal we cannot measure—the performance impact is negligible in the experiments. We have quantified the model performance and assume some inaccuracies could be due to random jitter or other assumptions that do not hold in practice. Furthermore, a vehicle may have highly variable fault rates depending on operating conditions and normal wear and tear. An aggressive yet practical fault model may consider approximately 30 faults per second, or roughly one fault every 33 milliseconds [53]. Faults provide opportunities for the attacker to hide its behavior as plausibly naturally-occurring errors, and detection techniques that ignore the fault model will incur false positives that undermine their effectiveness and utility for triggering recovery. Modeling faults—for attack and defense—is an important area to consider, especially in the design and evaluation of countermeasures.

The proposed zero-phase synchronization relies on the assumption that there will be a preceding message on the bus before the transmission of v. However, the attack could be detected if there are no preceding messages or if the bus is idle when the timer fires. We suggest that the attacker could track the time between received messages and skip an injection if the timer indicates the bus might be idle. Future work can validate the attack's effectiveness with such an attack strategy or quantify the effect of bus interference and jitter on the effectiveness of the zero-phase synchronization technique. Although we posit that the attack interval will only increase due to the factors, more rigorous experiments can be performed to substantiate these claims.

Although disabling transmission on receiving preceded messages may not be as reliable as the prior approach of aborting transmissions on transmit error, we do not see this as a limitation. Instead, we view it as a trade-off between effectiveness and feasibility. Furthermore, our approach is supported by all CAN controllers, making it a practical and viable option for attackers. In contrast, the prior approaches require specific CAN controllers to work. However, there are still areas for improvement, such as exploring more sophisticated synchronization methods or developing countermeasures to detect and prevent these attacks.

The proposed transitive attack strategy requires the attacker to construct forgeries that are reasonable enough to maintain low detectability. While this assumption is plausible, future work can explore techniques for constructing forgeries that are harder to detect or that do not require knowledge of the victim's ECUs. In addition, classification or optimization techniques might reduce the time or difficulty of the identification of $\mathcal{A}^*$ ECUs that can be used to satisfy a transitive attack.

VII. Conclusion

In this paper, we identified three improvements to the WeepingCAN stealthy CAN bus-off attack. We evaluated these improvements with reproducible benchmarks and have shown that some methods can achieve high attack success rates (75%–97%) under realistic bus loads. The transitive attack strategy makes it possible for an attacker to target many more victims with the stealthy WeepingCAN attack than previously known. We examined the efficacy of straightforward adoption of secure transceiver hardware, and found that they can prevent the classic bus-off attack and the transitive optimization we introduced, but that they may not address the WeepingCAN attack directly by themselves. We discussed several methods that may prove useful to help thwart the attack, but WeepingCAN and its variants remain open problems to solve.

References

[1] H. Olufowobi and G. Bloom, "Chapter 16 - Connected Cars: Automotive Cybersecurity and Privacy for Smart Cities," in *Smart*

Cities Cybersecurity and Privacy, D. B. Rawat and K. Z. Ghafoor, Eds. Elsevier, Jan. 2019, pp. 227–240. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128150320000160

[2] N. Zhao, X. Zhao, M. Chen, G. Zong, and H. Zhang, "Resilient distributed event-triggered platooning control of connected vehicles under denial-of-service attacks," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[3] L. Yang, A. Moubayed, and A. Shami, "Mth-ids: a multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2021.

[4] P. Agbaje, A. Anjum, A. Mitra, E. Oseghale, G. Bloom, and H. Olufowobi, "Survey of interoperability challenges in the internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22 838–22 861, 2022.

[5] 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: Association for Computing Machinery, 2016, pp. 1044–1055, vienna, Austria. [Online]. Available: https://doi.org/10.1145/2976749.2978302

[6] 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.

[7] 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 *20th USENIX Security Symposium (USENIX Security '11)*. San Francisco, CA, USA: USENIX Association, 2011, pp. 447–462.

[8] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[9] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "SIMPLE: Single-Frame Based Physical Layer Identification for Intrusion Detection and Prevention on in-Vehicle Networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, ser. ACSAC '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 229–244, event-place: San Juan, Puerto Rico. [Online]. Available: https://doi.org/10.1145/3359789.3359834

[10] S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, "Practical DoS Attacks on Embedded Networks in Commercial Vehicles," in *Information Systems Security*, ser. Lecture Notes in Computer Science, I. Ray, M. S. Gaur, M. Conti, D. Sanghi, and V. Kamakoti, Eds. Cham: Springer International Publishing, 2016, pp. 23–42.

[11] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the Clock: Emulating Clock Skew in Controller Area Networks," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, Apr. 2018, pp. 32–42, porto, Portugal.

[12] G. Bloom, "WeepingCAN: A Stealthy CAN Bus-off Attack," in *Workshop on Automotive and Autonomous Vehicle Security*. Internet Society, Feb. 2021.

[13] P. Bajpai, R. Enbody, and B. H. Cheng, "Ransomware Targeting Automobiles," in *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*, ser. AutoSec '20. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 23–29. [Online]. Available: https://doi.org/10.1145/3375706.3380558

[14] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of Automotive Controller Area Network Intrusion Detection Systems," *IEEE Design Test*, vol. 36, no. 6, pp. 48–55, Dec. 2019.

[15] B. Groza and P.-S. Murvay, "Security Solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, Mar. 2018.

[16] O. Ikumapayi, H. Olufowobi, J. Daily, T. Hu, I. C. Bertolotti, and G. Bloom, "Canasta: Controller area network authentication schedulability timing analysis," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 10 024–10 036, 2023.

[17] H. Wen, Q. A. Chen, and Z. Lin, "Plug-N-Pwned: Comprehensive Vulnerability Analysis of OBD-II Dongles as A New Over-the-Air Attack Surface in Automotive IoT," 2020, pp. 949–965. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/wen

[18] S. Kulandaivel, T. Goyal, A. K. Agrawal, and V. Sekar, "CANvas: Fast and Inexpensive Automotive Network Mapping," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 389–405. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/kulandaivel

[19] H. Olufowobi, U. Ezeobi, E. Muhati, G. Robinson, C. Young, J. Zambreno, and G. Bloom, "Anomaly Detection Approach Using Adaptive Cumulative Sum Algorithm for Controller Area Network," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, ser. AutoSec '19. New York, NY, USA: ACM, 2019, pp. 25–30, event-place: Richardson, Texas, USA. [Online]. Available: http://doi.acm.org/10.1145/3309171.3309178

[20] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "SAIDuCANT: Specification-Based Automotive Intrusion Detection Using Controller Area Network (CAN) Timing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1484–1494, Feb. 2020, conference Name: IEEE Transactions on Vehicular Technology.

[21] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "LibreCAN: Automated CAN Message Translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 2283–2300, london, United Kingdom. [Online]. Available: https://doi.org/10.1145/3319535.3363190

[22] U. Ezeobi, H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Reverse Engineering Controller Area Network Messages using Unsupervised Machine Learning," *IEEE Consumer Electronics Magazine*, pp. 1–1, 2020, conference Name: IEEE Consumer Electronics Magazine.

[23] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. New York: Springer-Verlag, 2012. [Online]. Available: https://www.springer.com/gp/book/9781461403135

[24] T. Hu, "Deterministic and flexible communication for real-time embedded systems," PhD Thesis, Ph. D Thesis, 2015.

[25] M. Rogers and K. Rasmussen, "Silently disabling ecus and enabling blind attacks on the can bus," *arXiv preprint arXiv:2201.06362*, 2022.

[26] P.-S. Murvay and B. Groza, "Dos attacks on controller area networks by fault injections from the software layer," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017, pp. 1–10.

[27] K. Serag, R. Bhatia, V. Kumar, Z. B. Celik, and D. Xu, "Exposing new vulnerabilities of error handling mechanism in {CAN}," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 4241–4258.

[28] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ecu of the can bus," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2018, pp. 1–4.

[29] A. Minaeva, B. Akesson, Z. Hanzálek, and D. Dasari, "Time-triggered co-scheduling of computation and communication with jitter requirements," *IEEE Transactions on Computers*, vol. 67, no. 1, pp. 115–129, 2017.

[30] H. Chishiro, "Multiprocessor semi-fixed-priority scheduling," *Journal of Information Processing*, vol. 26, pp. 202–211, 2018.

[31] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, vol. 130, 2015.

[32] S. Hounsinou, M. Stidd, U. Ezeobi, H. Olufowobi, M. Nasri, and G. Bloom, "Vulnerability of controller area network to schedule-based attacks," in *2021 IEEE Real-Time Systems Symposium (RTSS)*, 2021, pp. 495–507.

[33] B. Groza and P.-S. Murvay, "Efficient Intrusion Detection With Bloom Filtering in Controller Area Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, Apr. 2019, conference Name: IEEE Transactions on Information Forensics and Security.

[34] H. Olufowobi, S. Hounsinou, and G. Bloom, "Controller Area Network Intrusion Prevention System Leveraging Fault Recovery," in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, ser. CPS-SPC'19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 63–73, event-place: London, United Kingdom. [Online]. Available: https://doi.org/10.1145/3338499.3357360

[35] "M_can Busoff Recovery Handling," Robert Bosch GmbH, Application Note M_CAN_AN004, Nov. 2019. [Online]. Available: https://www.bosch-semiconductors.com/media/ip_modules/pdf_2/m_can/m_can_an004_v1-0_busoff_recovery_handling.pdf

[36] "TCAN4550-Q1 Automotive Controller Area Network Flexible Data Rate (CAN FD) System Basis Chip with Integrated Controller and Transceiver," Texas Instruments, Tech. Rep. sllsez5d, Jun. 2022.

[37] K. Tindell, A. Burns, and A. J. Wellings, "Calculating controller area network (can) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163–1169, Aug. 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0967066195001128

[38] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus

signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, 2017, pp. 1–4.

[39] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno, "Automotive intrusion detection based on constant can message frequencies across vehicle driving modes," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, 2019, pp. 9–14.

[40] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *Ieee Access*, vol. 8, pp. 58 194–58 205, 2020.

[41] P. Agbaje, A. Anjum, A. Mitra, G. Bloom, and H. Olufowobi, "A framework for consistent and repeatable controller area network ids evaluation," in *Fourth International Workshop on Automotive and Autonomous Vehicle Security*, 2022.

[42] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 1110–1115.

[43] A. Paul and M. R. Islam, "An artificial neural network based anomaly detection method in can bus messages in vehicles," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*. IEEE, 2021, pp. 1–5.

[44] B. Elend and T. Adamson, "Cyber security enhancing CAN transceivers," in *Proceedings of the 16th International CAN Conference*, 2017.

[45] S. Longari, M. Penco, M. Carminati, and S. Zanero, "CopyCAN: An Error-Handling Protocol Based Intrusion Detection System for Controller Area Network," in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, ser. CPS-SPC'19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 39–50, event-place: London, United Kingdom. [Online]. Available: https://doi.org/10.1145/3338499.3357362

[46] H. J. Jo, J. H. Kim, H.-Y. Choi, W. Choi, D. H. Lee, and I. Lee, "MAuth-CAN: Masquerade-Attack-Proof Authentication for In-Vehicle Networks," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2019.

[47] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 911–927.

[48] ——, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1109–1123. [Online]. Available: https://doi.org/10.1145/3133956.3134001

[49] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, Aug. 2018.

[50] S. Halder, M. Conti, and S. K. Das, "COIDS: A Clock Offset Based Intrusion Detection System for Controller Area Networks," in *Proceedings of the 21st International Conference on Distributed Computing and Networking*, ser. ICDCN 2020. Kolkata, India: Association for Computing Machinery, Jan. 2020, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3369740.3369787

[51] M. Kneib, O. Schell, and C. Huth, "EASI: Edge-Based Sender Identification on Resource-Constrained Platforms for Automotive Networks," in *Network and Distributed Systems Security Symposium (NDSS '20)*, San Diego, CA, Feb. 2020.

[52] J. Zhou, P. Joshi, H. Zeng, and R. Li, "BTMonitor: Bit-time-based Intrusion Detection and Attacker Identification in Controller Area Network," Nov. 2019. [Online]. Available: https://doi.org/10.1145/3362034

[53] I. Broster, A. Burns, and G. Rodríguez-Navas, "Timing Analysis of Real-Time Communication Under Electromagnetic Interference," *Real-Time Systems*, vol. 30, no. 1, pp. 55–81, May 2005. [Online]. Available: https://doi.org/10.1007/s11241-005-0504-z

Paul Agbaje is a Ph.D. student in the Computer Science and Engineering Department at The University of Texas at Arlington. He obtained his bachelor's degree in Electrical and Electronics Engineering from the University of Ilorin, Nigeria. His current research focuses on interoperability issues in internet-of-vehicles and cyber-physical system security.

Habeeb Olufowobi received his Ph.D. in computer science from Howard University in 2019. He joined the University of Texas at Arlington as an Assistant Professor of Computer Science and Engineering in 2020. His research focuses on embedded systems security and privacy challenges in emerging network technologies for connected autonomous vehicles, the Internet of Vehicles (IoV) in a smart city ecosystem, and vehicular cloud network.

Sena Hounsinou received her Ph.D. in electrical and computer engineering from Southern Illinois University Carbondale in 2018. She joined Metro State University as Assistant Professor in 2023. Her research interest is in computer architecture, cyber-physical systems, real-time systems, and reconfigurable computing.

Gedare Bloom (SM'19) received his Ph.D. in computer science from The George Washington University in 2013. He is Associate Professor at the University of Colorado Colorado Springs in Computer Science, which he joined in 2019. He was Assistant Professor of Computer Science at Howard University from 2015-2019. His research expertise is computer system security with emphasis on real-time embedded systems. He is an associate editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.