Silently Disabling ECUs and Enabling Blind Attacks on the CAN Bus

Matthew Rogers and Kasper Rasmussen

University of Oxford, Oxford, United Kingdom
matthew.rogers2197@gmail.com

Abstract. The CAN Bus is crucial to the efficiency, and safety of modern vehicles. Electronic Control Units (ECUs) exchange data across a shared bus, dropping messages whenever errors occur. If an ECU generates enough errors, their transmitter is put in a bus-off state, turning it off. Previous work abuses this process to disable ECUs, but is trivial to detect through the multiple errors transmitted over the bus. We propose a novel attack, undetectable by prior intrusion detection systems, which disables ECUs within a single message without generating any errors on the bus. Performing this attack requires the ability to flip bits on the bus, but not with any level of sophistication. We show that an attacker who can only flip bits 40% of the time can execute our stealthy attack nearly 100% of the time. But this attack, and all prior CAN attacks, rely on the ability to read the bus. We propose a new technique which synchronizes the bus, such that even a blind attacker, incapable of reading the bus, can know when to transmit. This increases a limited attacker's chance of success from the percentage of dead bus time to 100%. Finally, we propose a small modification to the CAN error process to ensure an ECU cannot fail without being detected, no matter how advanced the attacker is. Taken together, we advance the state of the art for CAN attacks and blind attackers, while proposing a detection system against stealthy attacks and the larger problem of CAN's abusable error frames.

1 Introduction

The CAN protocol was developed in the 1980s [4], and operates most modern vehicles: from consumer automobiles, to trucking [5], to military systems [17]. Computers, known as Electronic Control Units (ECUs), are connected to the physical components of the vehicle, and share information across a common serial data bus. For over a decade academia has published work on how insecure the CAN bus is [10],[6]. It has no authentication on any messages, this means that any attacker who can transmit on the serial data bus network gains control over the vehicle. In response to this lack of security, researchers published security systems, and intrusion detection systems [2]. In this paper we propose a stealthy attack which bypasses these detection systems by producing no visible impact on the bus: no extra messages, no error frames, and no changes in timing. This new attack abuses the CAN error process in combination with bit flip attacks to silently force an ECU into a state known as 'bus-off'.

ECUs will enter the bus-off state, which turns off their transmitter, after a number of erroneous messages. The number of errors necessary to reach this bus-off state vary, with a minimum of 32, but we will expand on how the entire CAN error process works in Section 2. Regardless, if an attacker has a way to produce errors, then they can control the bus by interrupting messages, and eventually having victims enter the bus-off state. This allows an attacker to effectively replace their victim on the bus.

We propose a novel attack which allows an attacker to push a victim into the bus-off state without producing any errors on the bus. It does this by flipping multiple bits in a single message, rapidly increasing the victim's error count while overriding their victim's error messages, and setting the appropriate bits such that other ECUs register it as a valid CAN message. However, the technology to flip bits is inherently complex and may not be 100% reliable. We introduce a probabilistic attacker, which has a non-100% chance of success at flipping any particular bit. Through this imperfect attacker we can understand how reliable an attacker needs to be at flipping bits to take a victim off the bus, and avoid producing visible errors. We found that even with just 38% accuracy for flipping bits, an attacker could silently take a single ECU off the bus within a single message with over 99% accuracy. More details on this probabilistic attacker are in Section 5.

This all assumes the attacker is in a position where they can read the bus, wait for the appropriate time, then deliver the attack while modifying it on the fly. But this is not true for all attackers. Limited ECU exploits may only allow an attacker to transmit a fixed payload, as they do not have a large enough buffer to adapt the attack to the bus. Or an electromagnetic induction attack may remotely flips bits on the line with no way of knowing the impact of those bit flips, or even if the flips succeeded [15]. As a result many of these attacks rely on the bus being silent when the attack begins so they do not collide with other messages, and mangle their own attack. We propose a new methodology to synchronize the bus for blind (write only) attackers, such that no matter what state the bus is in, the bus is manipulated into the start of a CAN packet. This changes the chance of success for blind attackers from the percentage of dead bus time, to 100%, making blind attacks a viable threat against the CAN bus.

The inherent problem our stealthy bus-off attack is exploiting is that interrupting an error results in another error. We propose changing the error process such that each error only counts as one error. Meaning that if an attacker causes an error on the bus, that error will continue forever until it is allowed to complete. Our small change has no impact to the safety of the bus, but changes how effective and how stealthy the attacker can be. By having error frames go until completion, the attacker must now be 99% accurate at flipping bits to have even an 80% chance at completing a single CAN compliant message. And now has to allow error frames to complete to enter the bus-off state.

We summarize our contributions as follows:

- A novel attack allowing an attacker to silently disable an ECU within one message by flipping bits. As well as a statistical analysis showing that an

attacker only needs 38% accuracy while flipping bits to have over 99% accuracy in silently disabling a victim ECU.
- A novel methodology which synchronizes the bus such that a blind attacker can guarantee the delivery of their malicious CAN messages, regardless of the bus state at the time of the attack.
- A proposed change to the CAN protocol's implementation of error frames to remove the possibility for silently disabling ECUs.

2 Background and Related Work

In this section we briefly summarize the CAN protocol, with the main focus being on how the CAN error process works. We then go into existing attack and defense work, particularly the bus-off attack which uses a similar technique to our stealthy attack.

2.1 CAN Background

CAN is a serial data bus protocol invented in 1986 by Bosch [4], and is used in most modern ground vehicles. The data bus is two 5V differential pair wires, connecting all of the vehicle computers to each others. These vehicle computers, known as Electronic Control Units (ECUs), transmit and receive data on the bus for the purpose of controlling the vehicle in a safer and more efficient way. The packet structure for CAN frames can be seen in Figure 1. The most notable element for the CAN protocol that separates it from most other message schemes is the arbitration process. ECUs all attempt to transmit at the same time, but back off if they observe a 0 on the bus before they transmit. The short hand is if the ID field is a lower value, then that message has a greater priority, and so wins bus arbitration. For this reason a 1 is referred to as a recessive bit, while a 0 is referred to as a dominant bit. The dominant bit overrides the recessive bit.

2.2 CAN Error Frames

At a high level, CAN Error Frames are used to cancel out an erroneous message and inform the rest of the bus that an error occurred. From the bus's perspective, this looks like six consecutive bits, usually six 0s, followed by an error delimiter of eight 1s to inform other ECUs the bus is now free, before finally retransmitting the erroneous message. The CAN error process works by having each ECU monitor the bus for specific problems and flag any errors they see. If an ECU registers enough errors, then that ECU turns off its own transmitter to avoid transmitting more errors onto the bus. After receiving enough error-free messages, that transmitter will eventually turn back on, though some implementations require restarting the vehicle to re-enable the ECU. This mechanism is designed for the safety of the vehicle, such that if any ECU frequently talks over other messages or is unable to properly transmit a signal, then it stops inhibiting the rest of the bus.

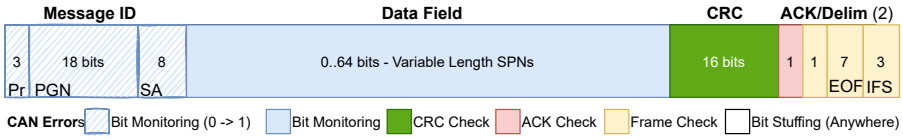

Fig. 1. Extended CAN Packet with error sections highlighted where Pr is priority. Bit Stuffing errors can occur anywhere in the packet while other errors are based more on the location. Generally speaking errors occur in response to bit flips, or CRC checks.

Let us examine what CAN considers an error. There are five possible errors [1]: bit monitoring, checksum check, bit stuffing, frame check, and acknowledge check. Figure 1 depicts where these errors occur. **Bit monitoring** registers an error if the value on the bus is different from what the transmitting ECU believes it sent. **Checksum check** determines if the sixteen bit CRC is valid. **Bit stuffing** checks that whenever five bits of the same value appear, they are 'stuffed'. This means that the opposite bit is transmitted, and the transceiver knows to ignore this bit when assembling the CAN frame. CAN Error Frames use bit stuffing to inform the rest of the bus an error occurred. They do this by transmitting six of the same bit in a row without stuffing, raising the bit stuffing error for every ECU so they drop the victim message. **Frame check** monitors any parts of a CAN frame which are meant to be recessive (1) and transmits an error if they are instead dominant (0). Finally, the **Acknowledge check** transmits an error if the ACK field is not set to a dominant bit. Taken together, these errors are monitoring tools for when an ECU is no longer able to determine when to transmit, consistently set bits, or calculate a checksum.

Each ECU tracks the number of errors it sees via internal error counters. Generally speaking, each error results in that counter increasing by eight for the erroring ECU, while each error-free message decrements the counter by one. If the counter reaches 127, then error frames transmit recessive bits instead of dominant bits. This is called the passive error state. Passive errors make it easy for a transmitting ECU to talk over error frames and complete a message under the assumption the error prone device is in the wrong. Interrupting an error frame increments this counter by eight again. If the error counter reaches 255, the ECU enters the bus-off state, turning off its transmitter. While this mechanism may make sense for safety purposes, it is a clear attack vector.

2.3 Related Work

There is a great deal of work on hacking the CAN bus [10], [6], [5], [14] [9]. Be it remotely [13] [11], or assuming an implanted physical device. Generally it relies on the premise that there is no authentication on the bus, so any message sent has some control of the vehicle. The problem is that then the attacker is conflicting with the existing devices on the bus, and talking louder than them.

As the field of CAN intrusion detection has matured over the last decade this flood of new attacker messages presents a problem. Several pieces of existing

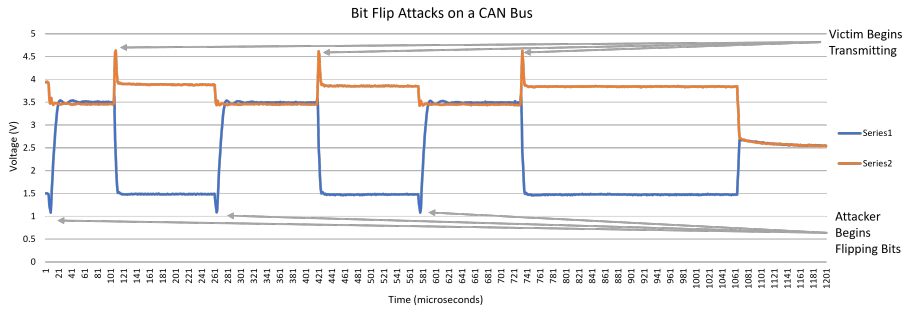

Fig. 2. Three extended bit flips attacks on a CAN Bus. The 2nd and 3rd bit flips occur during an error frame, resulting in another error frame. This demonstrates that an attacker can rapidly produce errors by repeatedly interrupting error frames.

work are designed to detect new messages on the bus, usually through timing characteristics [2], such as the timing intervals between message types. This technique relies on legitimate ECUs transmitting periodically. The way around this is to prevent the legitimate device from speaking, that way the attacker always looks correct. There are a few ways to prevent a legitimate device from speaking. Uploading malicious firmware to the victim device is effective for pivoting, but is simple to detect with a robust maintenance process. Industrial CAN standards allow for single network messages to disable a bus, but the attacker must use a very particular message, with a data field never seen in normal traffic, making it easy to detect [12]. In terms of silencing an existing device, the best way to do it in CAN is through a 'bus-off attack'.

The bus-off attack, introduced by Cho and Shin [7], synchronizes with a victim message during the arbitration ID, then transmits a dominant bit during the message to produce a bit monitoring error. By producing this bit monitoring error continuously, the bus enters the bus-off state, where the transmitter shuts down. This relies on the assumption that most messages are regularly transmitted, so the attacker can predict when a message will occur, and begin transmitting just before they do such that both devices simultaneously win the arbitration process. Significantly, this attack allows error frames to complete. Cho and Shin propose a detection system for bus-off attacks based around error frames, but rightfully point out that an attacker could slowly execute this attack, making individual error frames a poor thing to alert on.

3 Adversary Model

We define a new attacker called a stealthy attacker. The stealthy attacker is capable of reading and writing to the bus, with a goal of disabling and spoofing ECU(s) without being detected. An attacker unable to meet its goal is deemed unsuccessful.

We assume a 0 to 1 bit-flip has a fixed chance of success from 0-100%, and a 1 to 0 bit-flip has a 100% chance of success. The justification for this is based on the CAN protocol's defining of 1 as the recessive state, and 0 as the dominant. Any ECU transmitting a 0 while another transmits a 1 results in a 0. We expand on the idea of attackers with a not-100% chance of success in Section 5.

We make no assumptions about attack vector. It could be remote, an implant, or a supply chain attack as long as it can perform bit-flip attacks. The only restriction is that the attacker's access vector is not the ECU they are attempting to corrupt or take offline. That said, the nature of the attack vector often limits what the attacker can do. Remote attackers especially can have limited payloads which immediately transmit onto the bus. This leads to scenarios where the attacker's payload collides with other messages on the bus and becomes nullified by a CAN error frame. Our solution lies in Section 6 where we propose a new methodology to synchronize the bus. This allows even a blind attacker, unable to read the bus, to execute an attack and transmit a message on the CAN bus 100% of the time.

4 Stealthy Bus-Off Attack

This paper introduces a faster, stealthier bus-off attack for our stealthy attacker. In brief, the original bus-off attack produces errors on the bus by performing a single 1 to 0 flip. Each attack increments an error counter by 8, and by executing the attack 32 times over 32 messages the attacker puts the victim into the bus-off state. The attacker can go slower to make detection more difficult, but the gist is they eventually get the error counter to 255 and the victim transmitter turns off. Now an attacker can spoof that victim without concerns over conflicting messages between themselves and the victim.

Our stealthier version of the attack has the same goal of reaching the bus-off state, but can perform 0 to 1 bit-flips and is unwilling to produce errors on the bus. We also assume the stealthy attacker can arbitrarily flip bits, rather than having to win the arbitration process. The basis of this attack is that a CAN error frame, when interrupted, will produce another error frame, incrementing the victim's error counter by 8 each time it is interrupted. Figure 2 demonstrates this effect by performing 3 extended bits flips. The first produces an error frame which we allow to transmit for a few bits before interrupting the error frame with another bit-flip attack. We then repeat this process. Eventually the error frame transmits six bits in a row and is completed.

It follows that if an attacker can continuously interrupt error frames, then they can reach the bus-off state within a single message by flipping 32 bits. Once the victim is in the bus-off state, it will stop transmitting error frames, regardless of if any of them ever completed. Now, our attacker has turned off an ECU without any error frames transmitted on the bus and no more error frames being sent. The only remaining catch is ensuring no receiver errors are produced, both to avoid detection by an IDS and to ensure the rest of the bus processes their malicious message. Avoiding receiver errors means computing

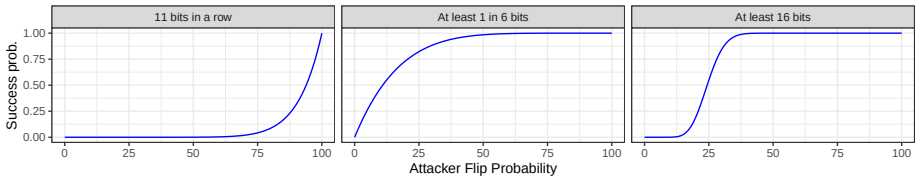

Fig. 3. From Left to Right: The probability of a probabilistic attacker to do 11 bit flips in a row. An attacker unable to do this cannot end a CAN packet without generating errors on non-victim ECUs. The probability of a probabilistic attacker to do at least 1 bit flip per 6 bits. An attacker unable to do this cannot reliably prevent errors from being observed on the bus. The probability of a probabilistic attacker to perform at least 16 bit flips in the 64 bit data field. An attacker unable to do this cannot bypass the need to perform 11 bit flips in a row, or guarantee no errors during the arbitration process.

the CRC for their manipulated message and ensuring the message ends with 11 recessive bits. Both of these are trivial once the bus is in a bus-off state, as 1 to 0 bit-flips are guaranteed in our adversary model. Once the message is done transmitting, the bus continues operating as normal but now with the victim ECU offline. The attacker is free to spoof the victim, taking the victim's place as its receiver remains online and keeping the vehicle functioning while the attacker transmits malicious payloads to other ECUs. All of this happens within a single message, approximately 600 microseconds, with no indicators of compromise communicated on the bus.

Taking a victim offline within a single message, with no errors, ensures no CAN error based detection systems can detect this attack. Detection systems relying on the timing between messages fail, as the attacker can spoof the timing properties of their victim and the attack creates no timing variation in the victim message [8], [16]. Data based detection systems fail to account for small deviations in data controlled by the attacker (i.e an engine control module is the root of truth for engine data) [18]. And an attacker can time their attack for the frequent retraining windows of voltage fingerprinting detection systems to avoid detection [3].

5 A Probabilistic Attacker

Flipping individual bits is an inherently hard problem on a serial data bus, particularly flipping the bus from a dominant state to a recessive state like our attacker is doing. In this section we display how accurate an attacker has to be to successfully take complete control over their victim. Being successful requires the attacker to produce zero indicators of compromise. This means not transmitting new arbitration IDs, not allowing CAN error frames to complete, and not generating errors in receiving ECUs.

Transmitting new arbitration IDs is trivially detected with a basic allowlist. Allowing error frames to complete provides an indicator that something is wrong on the bus, allowing the attacker to eventually be detected. Finally, any errors raised by receivers will also be raised by an IDS connected to the bus, making it immediately obvious an error frame is occurring, regardless of if the attacker interrupts it on the bus. Receiver errors are also problematic because they disable those receiving ECUs. The ECU transmitting the error has no reason to process any data observed while it is sending an error, as errors tell an ECU to drop whatever message it is processing. The consequence of this is that an ECU transmitting an error is no longer processing any messages on the bus. The receiver's errors will continue retransmitting until either the error frame is completed or the bus-off state is reached. Setting every ECU into the bus-off state is not necessarily a problem if the attacker is prepared to emulate the entire vehicle, but otherwise it is equivalent to a denial of service attack. Our question is if an attacker with less than a 100% success rate can avoid either transmitting a new arbitration ID, completing an error frame, or causing errors in other receiving ECUs. Answering this question reveals how realistic, or unrealistic, our stealthy attack is for attackers with less developed capabilities.

We are concerned with three scenarios that lead to our probabilistic attacker being unsuccessful:

- Completing a CAN packet (11 flips in a row)
- Interrupting an error frame (at least 1 flip in 6 bits)
- Transmitting a known arbitration ID

5.1 Completing a CAN Packet

The end of a CAN frame requires 11 recessive bits to be sent. This signals every other ECU that another message can now start. If a receiver sees a dominant bit in this period, then it will transmit a frame check error frame. The result is that if an attacker cannot successfully perform 11 dominant to recessive bit-flips in a row, then the rest of the bus (all receiving ECUs) will stop functioning until an error frame completes or the bus-off state is reached. The first image of Figure 3 demonstrates the likelihood of flipping 11 bits in a row for different attacker probabilities. We can see that while this attack is possible, if the attacker has less than 94% accuracy it quickly becomes worse than a coinflip. However, the attacker has another way to more reliably finish CAN frames. If the victim is put into the bus-off state or error passive state before the final 11 bits, then the victim will be either silent or transmitting a 1. These are effectively the same thing. The result is twofold: the attacker needs to do nothing to end a CAN frame, and bit-flips are easier to execute as the 1 to 0 bit-flip is built into the CAN protocol. A standard CAN transceiver can theoretically do this flip. We assume the attacker can flip a 1 to a 0 with 100% accuracy.

The only caveat is the attacker's ability to perform at least 16 bit-flips to reach the passive state before the CRC check. The number of bit-flips necessary to transmit an updated CRC depends on all prior successful bit-flips. However,

much like the final 11 bits of the CAN frame, transmitting a CRC is trivial if the bus is in a passive error state. A few system specific bits in the arbitration ID aside, this translates to the ability to flip 16 of 64 bits. The third image of Figure 3 demonstrates that most attackers can do this with a high degree of certainty. The data field may be somewhat nonsensical with lower probability attackers, but after one strange message the attacker will completely control the victim ECU.

5.2 Interrupting an Error Frame

The ability to interrupt an error frame represents the ability of an attacker to continue transmitting a message without letting an error frame complete. An attacker needs to interrupt error frames to ensure their manipulated message is not dropped by the bus. Interrupting an error frame allows an attacker to pretend to be an ECU without producing any indicators of compromise. Eventually interrupting the error frame will put the victim ECU into the bus-off state. Fundamentally, interrupting an error frame translates to an attacker flipping at least one of every six bits. The second image of Figure 3 demonstrates that most attackers can interrupt a CAN error frame with a high degree of certainty. However, this comes with two caveats for our stealthy adversary.

The first is ensuring receiver errors (CRC, ACK, Stuffing, Frame Check) do not occur, as an IDS can easily detect it. Caveat two is based on bus arbitration. We will go into more arbitration issues later in this section, but in this case the main issue is interrupting another ECU attempting to transmit a message. If an attacker flips a 0 to a 1 where another ECU is transmitting a 0, they produce an error frame. The error frame continues until the attacker allows it to complete or the non-victim ECU enters the bus-off state. Both of these caveats are avoidable by having the victim ECU in the bus-off state by the arbitration process, or failing that, in the passive error state. For the purposes of avoiding CRC, ACK, Bit Stuffing, and Frame Check errors the passive error state makes it easy for our attacker to transmit the appropriate bits. For avoiding arbitration issues, the passive state also makes the arbitration process function normally. Once again, the final image of Figure 3 demonstrates that reaching the recessive error state is attainable for most attackers by flipping at least 16 bits during the data field. In the case of starting a new message, at least 16 bits in the data field of the previous message.

5.3 Transmitting a Known Arbitration ID

Finally, we must consider the attacker's ability to transmit a realistic arbitration ID. If an attacker transmits a message with an ID which has never been seen on the bus before, it is trivial for any on-looker to identify that something is wrong on the bus. It follows that our attacker must transmit an arbitration ID which is normally transmitted by the victim ECU. For simplicity, we assume the attacker has a list of all known message IDs. The easiest and safest approach to this problem is to assume the victim is in the passive error state, and to perform

arbitration normally. Doing otherwise risks accidentally flipping a non-victim ECU's 0, or failing to flip a particular bit and ending up in a scenario where no valid arbitration IDs exist.

5.4 Summary

From our three scenarios we can conclude that an attacker capable of flipping bits with as low as 38% accuracy can successfully put a victim into the bus-off state, complete the message, then transmit as that now silent victim over 99% of the time. This is done without non-victim ECUs transmitting any error frames, any error frames completing, or any other indicators of compromise being transmitted on the bus. This shows that this stealthy attack is realistic, even with an adversary with under developed technology that works less than half of the time. The existing error frame process is insufficient for detecting attackers flipping bits on the bus. Thus, we conclude this process needs to be redesigned to handle actual attacks. But before discussing potential redesigns, we will further prove out the exploitability of the CAN error process by looking at blind attackers and how errors make them more reliable.

6 Blind Sync Methodology

This paper introduces a new methodology which synchronizes the bus for blind, write-only, attackers. The fundamental problem of attacking a serial data bus without reading it first is that the attacker has no way of knowing what state the bus is in. Let us take the example of a remote attack where the attacker flips arbitrary bits on the bus using electromagnetic interference. Assume we are using a CAN bus with an extended ID field and 70% bus utilization. Thirty percent of the time, the attacker transmits a full message, no messages are being transmitted on the bus, and the attack is successfully transmitted. The other 70% of the time, the attacker message is mangled as it collides with another message and an error occurs. If the attacker continues flipping bits they will likely put that random transceiver into the bus-off state. This is not ideal for actually controlling the victim vehicle, only performing a denial of service attack. If we want to actually inject an attack, then we need to guarantee that our attack results in transmitting a complete message. To do so we need to make that 30% chance a 100% chance by ensuring the bus is silent regardless of what state the bus is in when the attack starts. This means we need to find a series of bits that an attacker can transmit to force the bus to be either silent or at the start of a frame, regardless of what state the bus is in when the attacker transmits.

6.1 Synchronizing the Bus

For CAN we break the possible states into 4 categories: during a message, arbitration (0), arbitration (1), and dead bus. Figure 4 depicts bit timings for each of our 4 states, with red representing an attack and green representing the

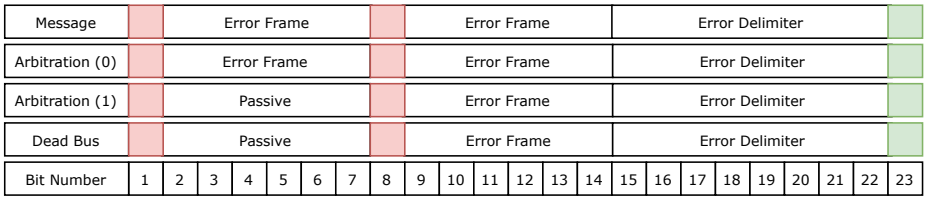

Fig. 4. Blind Sync Methodology which synchronizes the bus without reading any data, regardless of when the attack is executed. The red boxes at bit 1 and 8 indicate bit flip attacks. The green boxes indicate the point where the bus is synchronized.

time to start transmitting. Let us begin with transmitting during a message. Interrupting a message results in an error frame, after this error frame an error deliminator occurs for 8 bits, then normal operation occurs again. Error frames are an ideal way to synchronize the bus. Interrupt a message once, let the error complete, and then the attacker is confident in the state of the bus until the next error occurs. The trick is lining this error up so it syncs up with the other states. Flipping a 0 bit during the arbitration process causes the same result. However, flipping a 1 bit during arbitration results in the attacker winning arbitration. The attacker cannot effectively use this control of the bus, as they have no way of knowing how far into the message they are. Instead, the attacker needs to intentionally produce an error to once again try to synchronize the bus. Because they won the arbitration process, there is no other ECU transmitting a message that can be interrupted to produce an error frame. Instead they need to stop transmitting for six bits. This causes six consecutive 1s to appear, resulting in a bit stuffing error. Summing the attack bits, the error frame, and the error delimiter results in 21 bits. arbitration (0) and message errors only last 15 bits. To sync this up we need to add another bit-flip attack. Interrupting any part of a CAN error frame or the error delimiter results in another error appearing. By adding an attack on bit 8, right as the bit stuffing error frame starts and just after the bit monitoring error frame ends, we extend the length of the bit stuffing error frame by 1 bit and create a new error frame for the message and arbitration (0) case.

Now the message, arbitration (0) and arbitration (1) cases are synchronized. This leaves the dead bus case. The attacker cannot simply transmit a message during the dead bus time, as that only works 30% of the time. Instead, they need to intentionally produce errors to sync with the rest of the bus. This case works the same as the arbitration (1) case, except the initial bit-flip starts a CAN frame. Considering the bus was silent, no other devices are competing to speak. Thus, the attacker can copy the same technique as arbitration (1), where they stay silent for six bits and raise a bit stuffing error. These techniques have the same timing property. Now by flipping a bit, waiting six bits, then flipping the next bit, the entire CAN bus is synchronized until the next error frame. The only evidence of this attack is a single error frame completing, possibly two depending on the level of information provided by the CAN controller. The blind

sync methodology can be used for any attack where the attacker cannot read the bus and cannot queue messages.

6.2 Transmitting An Attack

Once the bus is synchronized, the blind attacker can transmit any message they wish with the caveat that they are likely to produce errors unless they win the arbitration process by setting the first few bits to 0. This will create new arbitration IDs on the bus which makes the blind attacker easy to detect, but only so far as knowing an attack occurred. Traditional CAN logging has no mechanisms for identifying how this attack occurred. The other option is picking a high priority but low frequency message. Now the attacker is likely to win the arbitration process, but will not accidentally share that win with a legitimate ECU. Regardless, our blind sync methodology ensures a blind attacker can have their desired impact on the vehicle 100% of the time, regardless of what is happening when the attack is launched. Our methodology is a vast improvement over the blind attacker's previous 30% chance of success.

7 Bit Flip Attack Detection

From our probabilistic attacker section it is clear that most attackers capable of flipping bits on the bus can silently turn an ECU off, without any errors appearing on the bus. The attacker can then seamlessly pretend to be that device, with on-lookers none the wiser. The critical detail here is that the attack is only easy for the majority of attackers because they can put the victim ECU into a passive error state, making it trivial to finish CAN messages, perform a normal arbitration process, and interrupt CAN errors. If we can prevent this passive state then only highly accurate attackers can properly end messages. Based on these conclusions we require a change to the CAN error process to prevent our stealthy attacker from being able to reliably control their victim. One that ensures the detection of any bit flip attacks and prevents attacks from occurring within a single message, even for attackers with 100% accuracy for flipping bits.

We propose Error Resistant Error Frames (EREFs), which change the error process in CAN such that if an error is interrupted, the victim ECU will re-transmit an error frame without increasing its transmission error counter. EREFS drastically change the 3 scenarios outlined in Section 5. For our first two scenarios, it means the easy way out of having the victim stop transmitting 0s midway through the message is gone. Instead, in our 'completing a CAN frame' scenario the attacker must be able to flip 11 bits in a row, or an IDS will detect the frame check error, and no message will complete. Even if they can drive the entire bus, the transceivers would not actually receive data because every receiving ECU starts transmitting errors. We demonstrate this cascade of errors in Figure 5, where EREFs cause all receiving ECUs to continue transmitting dominant error frames, and normal error frames cause receiving ECUs to enter the bus-off state. These continuing dominant error frames from all receiving ECUs mean that

EREFs make bit flip attacks unreliable for any attacker that cannot perform 11 bit flips in a row with near 100% certainty. Based on Figure 3, we can see that EREFs make ending a CAN Frame without errors unlikely for most attackers, with even a 99% attack chance only having an 80% success rate over 11 bits. The story is similar for the 'interrupting CAN error frames' scenario. A successful attacker would be assembling a reasonable message, with a valid CRC, while also interrupting 1 of every 6 bits and adjusting on the fly for attacks that do not work. This is easier than ending a CAN Frame, but again, prone to errors. Especially once the attacker reaches the CRC and needs a very specific set of 16 bits, and failure means either detection from an error frame completing, or an IDS seeing a receiver error.

EREFs place particularly strong restrictions on the attacker while trying to pick a valid arbitration ID. We can no longer assume the bus is in an error-passive or bus-off mode. This implies the attacker is actively flipping a 0 to a 1 during the arbitration ID process, even when they aren't attempting to transmit a message from their chosen victim ECU. If at any point another ECU attempts to transmit a 0 and the attacker performs a bit flip, then that ECU will begin to transmit a bit monitoring error frame as its 0 is flipped to a 1. This error ensures that ECU will stop receiving data until the error frame completes. Because the ECU cannot complete an error frame without providing an indicator of compromise, they have no way of re-enabling an ECU transmitting errors. But the attacker must perform bit flips during the arbitration, not only to prevent the CAN error from completing, but because otherwise it will transmit previously unseen arbitration IDs which provide an easy indicator of compromise. Differentiating between a 0 transmitted by an error frame, and a 0 transmitted by any number of ECUs on top of that error frame is infeasible. Meaning, the attacker must predict the exact order of messages going over the bus, or risk other transmitting ECUs generating errors, or an unknown arbitration ID. This is true for even an 100% accurate attacker, as their reliability does not affect the cascading error problem. The difficulty in starting a message as a result of EREFs demonstrates that EREFs make bit flip attacks infeasible for our stealthy attacker.

This paper assumes CAN errors are rare enough that an attacker intentionally allowing error frames to complete would be an indicator of compromise. For systems with common error frames we suggest an alternative CAN error detection approach which relies on observing a critical number of CAN errors. By having a security system monitor the bus a defender can count the number of error frames from each ECU and create their own internal error counter. Should that error counter ever reach the passive bus state, there is an anomaly. This is on the assumption that the erroring system typically functions. A system which regularly reaches the bus-off or passive error states would have difficulty maintaining normal system operations.

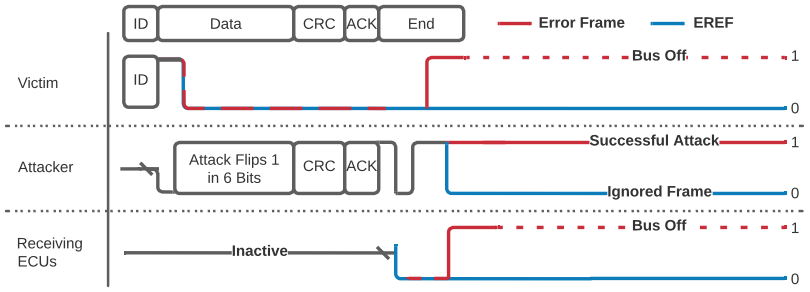

Fig. 5. Demonstrates the effect of an attacker inducing errors onto the bus where red lines indicate normal error frames, and the blue lines indicate error resistant error frames. This shows that an error during the end of a CAN frame results in all receiving ECUs generating errors. The normal error process enters the bus-off state, while EREFs continue transmitting dominant error frames. With receiving ECUs in the bus-off state the bus will process the attacker payload while EREFs cause it to be ignored

8 Conclusion

In this paper we propose a novel attack, a methodology for enabling blind attackers, and a change to the CAN protocol which prevents attackers from stealthily removing ECUs from the bus. Altogether, this paper provides a thorough examination of an attacker capable of flipping bits and what they can do on a CAN bus, even if the attacker is incapable of reliably flipping bits. As the CAN protocol exists currently, an attacker can silently disable any number of target ECUs, even with just a 38% chance of successfully flipping an individual bit. Our addition of Error Resistant Error Frames then puts the attacker in a position where they have no choice but to inevitably complete an error frame or have the bus stop functioning. This allows us to detect the attack. Outside of detection we propose a novel attack methodology which allows blind attackers to always have an impact on their victim, regardless of what the CAN bus is doing as they initiate their attack. This increases the blind attacker's chance of success from the percentage of dead bus time to 100%. Through the totality of this work future researchers can more accurately model against bit flip attacks as they consider the ramifications of more advanced attack vectors targeting the CAN protocol.

References

1. Can bus error handling (Jan 2017), https://www.kvaser.com/about-can/the-can-protocol/can-error-handling/
2. Al-Jarrah, O.Y., Maple, C., Dianati, M., Oxtoby, D., Mouzakitis, A.: Intrusion detection systems for intra-vehicle networks: A review. IEEE Access **7**, 21266–21289 (2019). https://doi.org/10.1109/ACCESS.2019.2894183
3. Bhatia, R., Kumar, V., Serag, K., Celik, Z.B., Payer, M., Xu, D.: Evading Voltage-Based Intrusion Detection on Automotive CAN. Ndss 2021 (February) (2021)

4. Bosch, R.: CAN specification version 2.0 (1991), http://esd.cs.ucr.edu/webres/can20.pdf
5. Burakova, Y., Hass, B., Millar, L., Weimerskirch, A.: Truck hacking: An experimental analysis of the SAE j1939 standard. In: 10th USENIX Workshop on Offensive Technologies (WOOT 16). USENIX Association, Austin, TX (2016), https://www.usenix.org/conference/woot16/workshop-program/presentation/burakova
6. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of the 20th USENIX Conference on Security. pp. 6–6. SEC'11, USENIX Association, Berkeley, CA, USA (2011), http://dl.acm.org/citation.cfm?id=2028067.2028073
7. Cho, K., Shin, K.: Error handling of in-vehicle networks makes them vulnerable. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 1044–1055. CCS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978302, https://doi.org/10.1145/2976749.2978302
8. Cho, K., Shin, K.: Fingerprinting electronic control units for vehicle intrusion detection. In: Proceedings of the 25th USENIX Conference on Security Symposium. pp. 911–927. SEC'16, USENIX Association, Berkeley, CA, USA (2016), http://dl.acm.org/citation.cfm?id=3241094.3241165
9. Currie, R.: Hacking the can bus: Basic manipulation of a modern automobile through can bus reverse engineering. SANS Institute (2017)
10. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: 2010 IEEE Symposium on Security and Privacy. pp. 447–462 (May 2010). https://doi.org/10.1109/SP.2010.34
11. Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. defcon 23 (2015)
12. Mukherjee, S., Shirazi, H., Ray, I., Daily, J., Gamble, R.: Practical DoS Attacks on Embedded Networks in Commercial Vehicles. In: International Conference on Information Systems Security. vol. 10063, pp. 23–42 (2016). https://doi.org/10.1007/978-3-319-49806-5_2
13. Munro, K.: Wi-fi as a weapon: how customer communication networks can be used to take over trains. In: InfraRail (2018)
14. Sagong, S.U., Ying, X., Clark, A., Bushnell, L., Poovendran, R.: Cloaking the Clock: Emulating Clock Skew in Controller Area Networks. Proceedings - 9th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2018 pp. 32–42 (2018). https://doi.org/10.1109/ICCPS.2018.00012
15. Selvaraj, J., Ware, D., Dayanikli, G.Y., Gerdes, R.M., Gaunkar, N.P., Mina, M.: Electromagnetic induction attacks against embedded systems. ASIACCS 2018 - Proceedings of the 2018 ACM Asia Conference on Computer and Communications Security pp. 499–510 (2018). https://doi.org/10.1145/3196494.3196556
16. Song, H.M., Kim, H.R., Kim, H.K.: Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In: 2016 International Conference on Information Networking (ICOIN). pp. 63–68 (2016). https://doi.org/10.1109/ICOIN.2016.7427089
17. Techbriefs Media: The can bus: Driving the future of autonomous military vehicles (May 2019)
18. Wasicek, A., Pesé, M.D., Weimerskirch, A., Burakova, Y., Singh, K.: Context-aware intrusion detection in automotive control system. Escar (2017)