Detecting CAN Attacks on J1939 and NMEA 2000 Networks

Matthew Rogers University of Oxford
Phillip Weigand Shift5 Inc.
Jassim Happa Royal Holloway, University of London
Kasper Rasmussen University of Oxford

Abstract—J1939 is a networking layer built on top of the widespread CAN bus used for communication between different subsystems within a vehicle. The J1939 and NMEA 2000 protocols standardize data enrichment for these subsystems, and are used for trucks, weapon systems, naval vessels, and other industrial systems. Practical security solutions for existing CAN based communication systems are notoriously difficult because of the lack of cryptographic capabilities of the devices involved. In this paper we propose a novel intrusion detection system (IDS) for J1939 and NMEA 2000 networks. Our IDS (CANDID) combines timing analysis with a packet manipulation detection system and data analysis. This data analysis enables us to capture the state of the vehicle, detect messages with irregular timing intervals, and take advantage of the dependencies between different Electronic Control Units (ECUs) to restrict even the most advanced attacker. Our IDS is deployed and tested on multiple vehicles, and has demonstrated greater accuracy and detection capabilities than previous work.

Index Terms—Land Vehicles, Information Security, Intrusion Detection

1 Introduction

For the past decade researchers have practically demonstrated hacking consumer automobiles by injecting traffic across the CAN Bus, pretending to be one of the many Electronic Control Units (ECUs) transmitting data throughout the vehicle. Initial efforts required physical access, but now consumer vehicles can be hacked remotely, albeit with time consuming reverse engineering for each make and model, and on a small scale [24]. From this framing car hacking is an important, but unlikely threat vector for most individuals.

Industrial protocols, such as J1939 and NMEA 2000, take that small attack vector and expand it across millions of trucks, ships, and even military systems by standardizing communication. Now, not only is reverse engineering not necessary, an attack that controls the engine on one make and model of truck also can control the engine of another truck with completely different hardware. This means that any remote attack payload no longer only works on one vehicle, but entire fleets of hetergenous industrial and military vehicles. The value of these assets is far higher than their consumer automobile counterparts [13], [14], while simultaneously being an easier target through their remote attack vectors, and standardized data fields.

A simple example of this is the electronic data logger (EDL). This device is required in US trucks, often plugs directly to the OBD-II port, and then sends data to a fleet management system [42]. A keen observer might note this looks extremely similar to the limited, highly targeted, physical implant style attacks used by car hacking researchers, but now mandatory, and installed across fleets. This is one example among many, the desire for real time access to bus data is growing and has the potential to be a scalable attack vector [40], [35], [41].

The question is then how to secure the CAN bus in a way that supports the decades of systems we already have. One method is adding authentication, but cryptographic mechanisms would impact bus performance, and limit possible bus configurations [26], [31], [16], [12]. This makes it impractical for existing systems, especially with J1939 running at 250kb/s.

The alternative to authentication is adding an intrusion detection system (IDS) which detects attacks as they go across the bus. The proposed solutions in this area range from timing analysis [9], [38], [45] to signal fingerprinting [10], [33], [39] to physical invariance models based on the data from the bus [32], [43]. While these solutions work for some types of messages, they have serious gaps.

Take, for instance, a message which enables and sets the cruise control speed. Our electronic data logger attacker could transmit a single cruise control message indicating the brake is off, the cruise control is enabled, and setting the cruise control speed to 1mph. This causes the vehicle to slow down dramatically, which could create dangerous situations in typical driving environments. In our datasets the cruise control message has three different transmission intervals centered around 35ms, 100ms, and 150ms. Previous timing, data, and ECU fingerprinting techniques do not cover this attack. Prior timing based work fails to account for these three different intervals, meaning they either generate false positives or ignore the message entirely. Data based work relies on detecting changes that violate the physical invariance of the system, but cruise control controls the vehicle using normal mechanisms. Finally ECU fingerprinting techniques fail to account for legitimate devices,and are prone to high false positives from varied external environments. Fundamentally the problem is these techniques do not cover all message / attack combinations when used in isolation.

Message ID	Data Field	CRC	ACK	EOF
29 bits, w/ Priority, PGN, and Source Address	0..64 bits - Variable Length SPNs	16 bits	2	7

Fig. 1. J1939/NMEA 2000 Extended CAN Packet. ACK indicates Acknowledgement, and EOF indicates End of Frame.

Our IDS, CANDID uses a combination of timing, data, and voltage-based techniques in a novel way to ensure we apply some security guarantee to every message, not just ones with regular transmission intervals, from physical implants.

The first step is detecting our EDL sending our cruise control attack. CANDID detects the EDL, or any ECU, spoofing the messages of another ECU by calculating the interval between messages. This interval based detection functions by correlating data to the timing of messages, allowing CANDID to select the appropriate timing for the state of the system instead of assuming fixed timings. Of course, an EDL does not usually transmit messages, meaning we can detect any message it transmits, as it always spoofing another ECU. Even so, all normally transmitting attackers can no longer spoof another ECU's data fields. CANDID leverages this guarantee by drawing dependencies between data fields, such that even if an attacker takes over a critical ECU, their data is restricted to what could be expected given the data of every trusted ECU. The final consideration is how to detect an attacker which has control over the voltage, meaning they could arbitrarily change any and all bits on the bus. Rather than a full voltage fingerprinting mechanism which is prone to false positives and requires constant retraining, CANDID detects these bit flips through CAN error frames and voltage transitions which never occur in benign traffic.

We summarize our contributions as follows:

- An integrated timing and data solution which more accurately predicts message timing based on the state of the system, and dependencies between data. Using these features we are able to detect an adversary spoofing trusted ECUs, and limit attackers in control of legitimate ECUs.
- A practical detection mechanism for signal-level manipulation attacks which doesnt rely on continuous training. Our improvements on bit manipulation attacks to silently reach the bus-off state underscore the importance of this detection mechanism.
- A proof of concept implementation of CANDID. The implementation is tested on multiple vehicles, as well as adversarial data generated on a testbench.

2 Background and Related Work

In this section we provide a background on the CAN and J1939 architectures, as well as existing literature on offensive and defensive techniques affecting these architectures.

2.1 CAN and J1939 Background

CAN is a serial data bus protocol invented in 1986 by Bosch [4], and is used in most modern ground vehicles. J1939 is a software standard created by the Society of Automotive Engineers [37], which tells ECUs how to interpret the contents of an extended CAN packet, as seen in Figure 1. It also adds a networking layer which enables a few attacks [25], and is easily spoofed by an attacker. Importantly, a Parameter Group Number (PGN) identifies the type of message, and allows us to translate the data fields, called Suspect Parameters Numbers (SPNs). NMEA 2000 uses the same structure. For the sake of clarity we will refer to PGN as the message type, the packet ID as the message ID, and SPNs as data parameters, or simply data, for the rest of the paper.

CAN transmits data at regular interval, though messages can have multiple transmission rates, or a wide interval. We can see examples of this in Figure 2, which graphs the observed message intervals from a single data set. The first message is consistent, and would be considered periodic by existing work. The other two have varying transmission rates, and may be considered aperiodic by the same work. Notably J1939 transmits sensor data at a fixed rate. This is not necessarily a single fixed rate, but it will reliably transmit rather than transmitting sporadically on each update.

2.2 Historical Attacks on CAN and J1939

Hacking CAN involves reverse engineering a vehicle to determine the proprietary CAN messages, and then injecting arbitrary messages to manipulate the system into a desired state [11], [36]. Most work on offensive techniques focus on preventing the system from operating, manipulating the brakes, or other safety critical functionality, sometimes by remotely pivoting through a telematic device [24], [7], [21]. Trends indicate more safety-critical systems will use telematic solutions for predictive maintenance [19], [35], providing additional attack vectors. In J1939 hacking becomes simpler as the specification removes the need for reverse engineering, while only requiring the same hardware used when hacking consumer CAN devices [25], [5].

In terms of specific attacks this paper is primarily concerned with attacks that can disable an ECU on the bus. An address hijack attack claims an existing ECU's network address to prevent it from speaking [25]. Other attacks like the bus-off attack and CANStomper solution produce CAN errors to disable individual messages and eventually stop an ECU from transmitting [8], [15], [22]. This paper improves upon this work by cancelling out messages without producing an error frame, and having the transceiver enter an off state within one message, as described in Section 5.

2.3 Existing Defense Mechanisms

Table 1 contains a summary of CANDID versus existing work. We often find timing based solutions to serial data bus exploits [45], [38], [2]. This work relies on trusted ECUs transmitting periodically, making an attacker's message an obvious anomaly. In a similar vein Pawelec et al., Zhou et al. and Butler used machine learning to identify anomalous messages [29], [46], [6]. Lee [23] relies on sending requests on the bus, which requires an IDS capable of transmitting data, something which risks increasing the bus utilization such that legitimate messages fail to transmit on noisy buses. Cho and Shew analyzed clock skew to fingerprint each

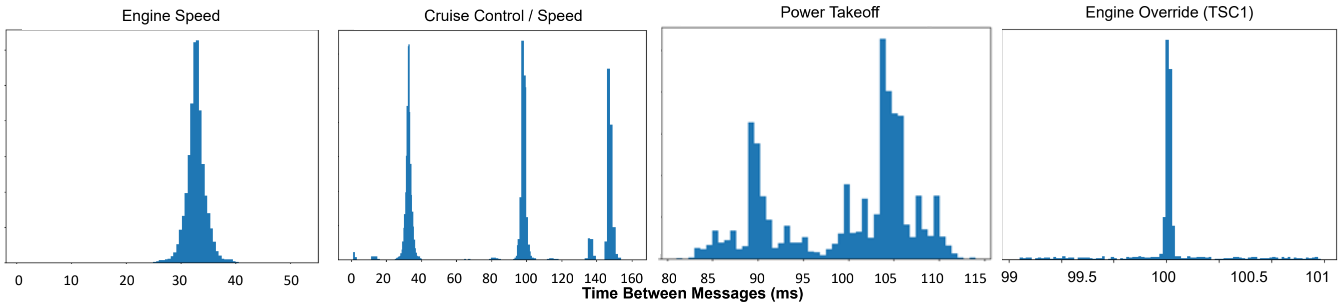


Fig. 2. Interval data from 4 J1939 messages. The engine speed and override messages are easy to predict with their single distributions, the rest require an understanding of when messages are transmitted based on the data going over the bus.

TABLE 1
Comparison between CANDID and related CAN IDS research by technique and attack coverage. Half Circles indicate that we believe their research may cover these attacks, though our adversary models differ. Spoofing attacks pretend to be another ECU on the bus, masquerade attacks control an existing ECU, sending that ECUs normal set of messages, and bit manipulation modifies individual bits from existing messages. Aperiodic refers to aperiodic spoofing and encompasses messages without a single regular transmission interval.

	Techniques				Attack Coverage			
Proposal	Timing	Data	Fingerprinting	Crypto	Spoofing	Aperiodic	Masquerading	Manipulation
Song et. al [38]	✓	-	-	-	●	○	○	○
Cho and Shin [9]	✓	-	✓	-	●	○	○	○
Groza et. al [16]	-	-	-	✓	●	●	○	●
Wasicek et. al [43]	-	✓	-	-	○	○	●	◐
Choi et. al [10]	-	-	✓	-	●	●	○	●
Sagong [33]	-	-	✓	-	○	○	○	●
Zhou et. al [46]	✓	✓	-	-	●	○	○	○
Quinonez et. al [32]	-	✓	-	-	○	○	●	◐
Daily et. al [12]	-	-	-	✓	●	●	○	●
CANDID	✓	✓	-	-	●	●	●	●

ECU and used variations in it for alerts [9]. This work has a 0.055% false positive rate. Based on our capture of 1.5 million CAN packets from a truck in 90 minutes, one would expect nine false positives a minute. A recurring theme in the referenced work is that each of these avoids aperiodic messages. Avoiding some sub-set of messages provides a clear target for an attacker.

In terms of research that currently uses the data field, the closest is physical invariance models [32], and neural networks incorporating the OBD-II pinout [43] [17]. Each of these papers account for primary vehicle functions, but are inherently limited in their data analysis. Quinonez [32] applies physical model restrictions to a small line following robot car, making it unclear how well this approach scales to the magnitude of data coming off a CAN bus. Wasicek et al. [43] references ten data points. Even factoring in recent reverse engineering work [30], only 32 data points are extracted. Hanselmann et al. [17] uses a neural network on individual IDs, but it lacks context for what the data means. This ambiguity, in combination with the general difficulty of actionable alerts from neural networks makes this approach poorly suited for any real time decisions. Detection wise each of these approaches acts in isolation of message timing. We believe physical models are best suited to an integrated solution between data and timing, as this improves timing analysis while ensuring a fast attacker cannot bypass the physical model by intelligently flooding the bus.

Voltage-based intrusion detection, or signal fingerprinting [10], [8] [39], [33], [20] is useful for detecting bit-manipulation attacks, and implants. However, these techniques are ineffective against corrupted devices, and often require continuous retraining to account for shifting environmental factors. Not accounting for corrupted devices means an attacker taking over an existing device can send any messages that device would normally send without any risk of detection.

While the referenced papers are capable of detecting many attacks, they fail to account for every message on the bus, or specific attackers. Through a combination of timing and data analysis, described in Section 4, as well as a more targeted voltage detection mechanism, described in Section 5 we address these shortcomings. We discuss the design advantages of CANDID over the aforementioned research in Section 7.

3 System and Adversary Model

In this section we define our system and adversarial model. The system model defines our CAN bus infrastructure and

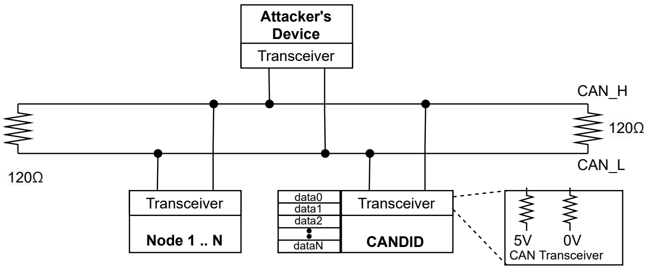


Fig. 3. System Model for a CAN system. Each node represents an ECU with a CAN transceiver, though attackers may have custom hardware.

how CANDID is connected. The adversary model defines the capabilities of the adversary.

3.1 System Model

Our system model is depicted in Figure 3. It is a CAN bus which operates using the differential voltage between two wires designated CAN-L (low) and CAN-H (high). Messages transmitted on the bus are visible to all connected nodes. The specific ECUs vary by vehicle, but some example systems controlled by ECUs include: transmission, engine, brakes, steering, and more. To listen to these ECUs any device with the appropriate connections only has to connect to these two wires.

CANDID is connected to the two wires of our CAN bus, allowing it to monitor all messages transmitted on the bus. CANDID is able to interpret any read data using the J1939 standard, and compares those data values against each other or values within the specification to send alerts. All CAN packets and alerts are stored, compressed, on disk for future analysis. CANDID is secure from wireless threats, and can not be removed from the system. Our IDS is also capable of monitoring the amperage going over the pullup resistors depicted in our system model. Our system requires a brief training period, we assume the attacker is not active during this period.

In this paper we define the state of the system to be CANDID's estimation of how the system is operating. For each message CANDID updates its internal variables, and tracks how those change over time. A full explanation for this process of tracking state is in Section 4.

3.2 Adversary Model

We define the goal of the adversary as changing the state of the system. The state is changed if a dependent data field is modified as a result of the attack. An attack is successful if the state is changed without CANDID detecting it.

An adversary has the power to manipulate voltage over the bus directly, send and receive messages, and modify their attacks based on the state of the system. In a realistic environment the capabilities vary between attack vectors, with remote attackers being limited in their adaptability, and corrupted ECUs having the most control over their victim. However, we deliberately assume the strongest adversary model possible regardless of attack vector and base our security analysis on how that attacker affects the bus.

At a high level an attacker has three options for transmitting an attack across the bus. They can transmit a new message, increasing the overall bus utilization, corrupt an existing device and transmit malicious data as the victim device (effectively replacing a legitimate message with a malicious one), or arbitrarily manipulate the voltage of the bus. In this paper we define these attacks as follows: **Spoofing**, where an attacker transmits a new message for an ECU they do not control. **Masquerading**, where an attacker controls an ECU and transmit the messages it would normally transmit with different data. **Manipulation**, where an attacker controls the voltage to modify individual bits.

This categorization does not limit possible attacks, instead it provides nuance for our detection system. Take a corrupted engine control module, it is able to perform a masquerade attack for engine messages, but if it attempts the same on a trusted ECU, then that trusted ECU is still transmitting and our attacker is increasing bus utilization, making it a spoofing attack. Through this categorization we can define security guarantees for our attacker, regardless of how they connect to the bus.

4 CANDID

This section will outline the design considerations for CANDID, then expand on the chosen designs in the subsequent subsections. The core design is motivated by the three broad attack categories laid out in the adversary model: spoofing, masquerading, and manipulation. Previous work clearly indicates timing is an effective mechanism for detecting spoofing; an attacker transmits a message on the bus, that message is regularly transmitted by a trusted ECU, and the result is an anomalous timing interval. When applied to every ECU on the bus, this technique detects any attack vector pretending to be a trusted ECU. However, if a trusted ECU transmits with multiple timing intervals, then an attacker can easily circumvent the detection system. This indicates a CAN message ID is not enough to detect attackers for all messages. From this we conclude CANDID must be able to use the data it sees on the bus to understand messages with multiple timing intervals.

The next challenge is masquerade attacks, where a corrupted device transmits fake data at the appropriate time. Fake data covers a wide range, from increasing the Engine speed to 8000 Revolutions Per Minute (RPM), to minor tweaks creating small inefficiencies. In some ways always detecting those minor tweaks is implausible with physical systems. Minor fluctuations happen constantly, and calculating the impact of every data field on the system is too computationally expensive for a lightweight IDS to do in near real time. Instead, we take the approach of constraining attackers and limiting damage. Our spoofing detection covering all messages ensures that an attacker cannot transmit the IDs, or data, of another ECU. From this foundation, CANDID can safely examine the data going across the bus and calculate correlations between the data fields, effectively creating a dependency tree. If the data from one dependent ECU does not match the others, CANDID can alert. This limits the scope of an attacker's fake data, without attempting to model the entire system.

Manipulation attacks bypass both of our intended detection mechanisms, as an attacker could overwrite every bit of the bus in a way that changes no timing characteristics, and

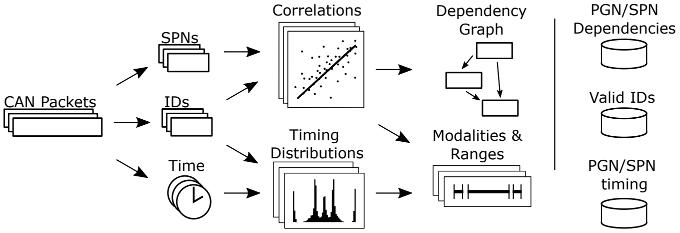


Fig. 4. Training phase of the design. Correlations are drawn between data and IDs, resulting in a dependency graph based on which data fields change in response to a given data parameter changing. Simultaneously a distribution of intervals for each ID is calculated and combined with the correlations to identify the data fields which result in a change in timing. Our training is stored as PGN/SPN dependencies, a list of valid IDs, and PGN/SPN timing.

maintains a consistent narrative in the data field. The first option is a voltage based intrusion detection system which fingerprints every system on the bus. Yet, these solutions can be unreliable in real world environments without retraining, while having the same issues with masquerade attacks as our easier to train timing solution. For CANDID we choose a more targeted approach to manipulation attack detection. To perform a bit flip, i.e., change a logical '0' to a '1', a higher than normal amount of current will flow through the termination resistors in the connected ECUs, which can be detected by CANDID. To avoid complicating the description of our timing and data analysis, we describe how manipulation attacks are executed and detected in Section 5. CANDID's unique approach of integrating and improving upon these three techniques ensures we detect more advanced attackers, more accurately, regardless of if they are trying to avoid our IDS and know how it functions.

4.1 CANDID Training

In order for CANDID to function it must train on a clean vehicle dataset. By training we mean fine tuning CANDID based on the parameters of the vehicle, not a machine learning based solution as seen in recent literature. Without training our solutions risks being too generic, allowing attackers to sneak by as one ECU transmits with a slightly longer delay, or more variance in a data field.

CANDID has 2 main components: determining the dependencies between data parameters, and the expected timing intervals for each ID, as shown in Figure 4. In this subsection we describe how CANDID determines timing distributions, how those distributions are affected by the data going over the bus, and the correlation between parameters. CANDID trains off of a continuous dataset with times calculated based on arrival time in microseconds from the perspective of the process communicating with the transceiver. This process need to be redone for each vehicle.

For high quality training CANDID requires CAN data from the vehicle in normal operations (battery on, idle, driving in various conditions), and importantly the transition between states. Any new types of messages or vehicle states will be seen as anomalous. This is generally fine for rare messages, as we expect an operator would be fine with being alerted to a legitimate error state, but any core vehicle functionality needs to be included in the training set to avoid false positives.

4.1.1 Timing Interval Training

CANDID uses the ID field of each packet to group messages and calculate the interval between each message with the same ID, resulting in a list of intervals, and timestamps for each ID. The list serves two purposes. Firstly it gives us a list of valid IDs, any IDs not seen during training are assumed to be anomalous. The second purpose of the list of ID intervals is determining when a message is going to be transmitted. We do this by taking the maximum and minimum for each list of intervals, defining them as the upperbound and lowerbound respectively. Of course, determining when a message should arrive is not quite so simple. If the space between the bounds is too large then an attacker may be able to sneak in. If they are too small then the model may be prone to false positives.

CANDID's time based training must ensure that for a given period an attacker's message must result in 2 or more messages within an interval, or one legitimate message within an interval, and the attacker's message outside the interval. This bound is defined such that the lowerbound is at least as high as the width of the interval. Bounding based on this width is an improvement on existing timing based work, which assumes messages are periodic [38], and focuses on messages occurring quickly instead of outside of a bounded time range. But not all messages will fit within these bounds as some messages are aperiodic, only occurring on certain conditions, or periodic but with the transmission rate dependent on another variable. The timing training takes the messages which do not meet our bounds criteria, and identifies the locations of their modalities. A modality being the grouping of timing intervals around different times. It does this by transforming the set of intervals for a given message into a Harrell-Davis quantile estimator [18]. This quantile estimator is a way of estimating the shape of the interval dataset for each message ID. We can then apply a low-land detection algorithm [1] to that quantile estimator data. In brief, the algorithm uses the quantile estimator to identify local maxima in our dataset, then metaphorically fills the valleys of our dataset with water to identify any peaks. Then it narrows these down to the modes of the dataset based on how much water filled the different valleys. CAN interval datasets tend to be noisy, with the risk of modes being close together. The low-land detection algorithm allows us to identify the modes of n-modal distributions, and by extension which messages fit within each of those modes.

This gives CANDID n groups of messages, where n is the number of modalities for when a given message occurs. Now CANDID knows that a message can appear at roughly any of these times, but CANDID still needs to identify two things. CANDID does not know how to identify what message is going to be sent at what mode. And CANDID has no knowledge of how to draw a timing interval around this mode such that it captures the appropriate messages without being too encompassing. These require a contextual understanding of why a message changes its transmission rate.

4.1.2 Modality Based Interval Training

At this point in training CANDID has lists of messages with timing intervals at each modality. Since the mode is by definition a common state, CANDID can search for when a message goes from transmitting in one modality to another. Our IDS takes the list of messages that occurred in the timeframe between the modality switch, then grabs the last time CANDID saw that message before the modality switch, and the most recent occurrence after the modality switch for each of those messages. Then CANDID enriches the data fields of those messages to identify what data fields are changing as the modality changes. To pare down the list of potential messages, CANDID starts with smallest time between each modality switch for a given message and ignores any data fields not included in that switch when training on future observations of that switch. We refer to the data fields that influence the timing characteristics of a message as state parameters. We short-circuit the processing to find these state parameters by first examining discrete parameters, as these often reflect the state of the vehicle. If that parameter changes as the modality changes, then we mark it as a state parameter. More than one parameter may change for each modality, which can make it difficult to establish a causal relationship. But CANDID is indifferent as long as the change is consistent. The more modality switches in the training dataset, the more CANDID can whittle down the possible list of state parameters. CANDID stores these state parameters for later.

Now that CANDID has the state parameters for each modality change, it needs to learn the timing interval it applies to. CANDID iterates through the training dataset for each message with state parameters, identifying which state the system is in when the message appears, and lowering the lowerbound or raising the upperbound accordingly. After iterating through the whole dataset each modality for each message with multiple transmission rates now has a base lowerbound and upperbound. From there if the modalities are far apart then we expand the interval to its maximum to be more forgiving of variations, favoring a higher upperbound as attacks are likely to be faster than slower messages. If the modalities are close together and have the same state parameter and ID then they are merged together. With modalities close together they can often have overlapping intervals as a particularly fast message from one interval lines up with a particularly slow message from another. This does not present a security problem for our interval based analysis because we assume that a message does not rapidly flip states. Rapidly being defined as the upperbound of the original transmission interval. That way an attacker cannot bypass our interval based guarantee by pretending the system is in another state. At the end of this modality training CANDID has the base interval training for single transmission interval messages, and for more complicated n-modal messages CANDID has a list of state parameters to determine which of n timing intervals it applies to a given message.

The ramification of this modality training is that an attacker cannot spoof any messages regardless of the ID. The inability to spoof means that an attacker has to corrupt the device they want to send messages as. In part, this is because traditionally adhoc messages, such as the power take off or engine override (TSC1) messages highlighted in J1939 hacking work [5], are typically transmitted at predictable timing intervals. We see both of these messages in Figure 2 as images 3 and 4. We expect that in datasets where this message is only transmitted in emergency situations, CANDID would train on data that indicates that situation. Take an aperiodic message, such as an engine diagnostic light. Our data-based training allows CANDID to see the data which results in that light turning on, and alerts if that data is not true when the engine light message is transmitted. If the attacker is not directly controlling the engine, but instead the gauge cluster displaying the light, then they have no way of setting the engine conditions that result in that diagnostic light without being detected by the timing anomaly from a spoofed engine message.

4.1.3 Data-Dependency Training

The final phase of CANDID's training is identifying how a change in one parameter affects a future parameter. We include this because if an attacker can corrupt an ECU, then CANDID needs a mechanism for detecting when they are sending malicious data. We identify deviations in the data by calculating the correlations between parameters. Afterall, vehicle operations are communicated on the CAN bus because the ECUs rely on this data to function. If the data ECUs output does not match with the flow of data going over the bus, then there is an anomaly. As these relationships are often non-linear we use Spearman correlation to calculate the relationship between parameters. To do this we take our training dataset and pair each data permutation together, duplicating the slower message to pair with each faster message. Duplicating the slower message allows us to determine how much a faster message is concerned with older data as the system continues to run. To reduce our risk of producing false positives we only consider pairs with a correlation an absolute value greater than 0.85, though this value could be changed depending on the desired sensitivity and computational restrictions. The correlation values for these messages are stored by our training process so we can compare the value in our evaluation phase.

The last step to ensuring correlations are effective is accounting for naturally fluctuating scalar values. The RPM of an idling engine is an inherently unstable data point, yet an idling engine will only fluctuate so far, as the physical component attempts to maintain a steady value. Parameters correlated to engine speed will be fluctuating along with engine speed, but these messages are not always transmitted at the same speed, thus we need to ensure an attacker cannot slowly change engine speed such that each individual step looks like normal fluctuations, but never averages back to normal. Our solution is to compute the difference between the correlated and observed value, where we assume an expected difference of zero for dependent messages transmitting more frequently than their correlated message. By applying the CUSUM algorithm [28] to this data, we can alert on both small errors propagating across a large period of time, and large variations from the correlated value. The alert threshold is determined by training on normal data and taking the maximum observed error in either direction. The result is no matter how slowly or quickly an attacker changes the data, we will alert before the system is seriously impacted.

Let us examine the ramifications of this approach for some engine messages. If an attacker corrupts the engine they effectively control most of the vehicle. Of course, many of the messages an engine transmits are dependent upon inputs from the rest of the system, such as the transmission. So if an attacker tries to increase the engine speed without sending the transmission message engine speed is dependent on, then CANDID will alert on the deviation between the expected correlation and what the attacker sent. Our timing guarantees ensure that a corrupted engine control module, could not then spoof a transmission message without being detected. That said, there are high level messages which take control of systems, like the engine override message mentioned in the previous subsection. But the engine sends that message, and it sends it regularly. Because of our improved timing guarantees we can predict when that message will occur, and the attacker has to take over the engine specifically to send an engine override message. The impact of the data correlation guarantees are best demonstrated with the plethora of other ECUs on a CAN bus. It is important to ensure that any cheap ECU cannot start lying about its data to an extent where it has an impact on the vehicle. This forces the attacker to stick closely to what the bus is saying for most ECUs. And for the rare ECUs with override messages the attacker now needs to corrupt that ECU specifically to perform their desired attack.

4.2 Interval Design Considerations

CANDID relies heavily on the idea of fixed timing intervals. At face value an interval sounds overly simplistic and prone to false positives. An easy to imagine alternative is viewing the distribution of timing intervals and determining if a number of messages seen over some time period matches that distribution. The problem is that many CAN attacks are a single CAN packet, making this alternative unlikely to detect the attack. Still, timing is an ideal basis for security as all attacks, outside of manipulation attacks, affect the timing of messages. A timing interval, enhanced by knowledge of the data field, allows the defender to detect attacks without predicting the exact order of messages.

The question is then how we define the interval, and why. Too wide of an interval and an attacker transmitting near the lowerbound may remain undetected as the legitimate ECU transmits closer to the upperbound. When the attacker transmits within the expected interval a new expected interval is defined, and the legitimate ECU falls within this interval. Too small of an interval and CANDID is prone to false positives. We earlier defined the bounds of this interval as the lowerbound being greater than the total width of the interval. This ensures that if an attacker transmits within an expected interval the legitimate ECU's transmission is guaranteed to fall outside of the new interval. With these bounds no matter how many messages an attacker transmits, as long as a legitimate device is transmitting CANDID is guaranteed to raise an alert.

The observed upperbound and lowerbound may be more rigid than necessary. CANDID expands intervals such that the lowerbound is only just greater than the width of the interval. This expansion favors increasing the upperbound. A message is more likely to be delayed as the bus tends to get busier than the baseline. The more comprehensive the dataset the less this expansion matters for reducing false positives, but generally some expansion is desired to disentangle abnormal bus loads from attacks.

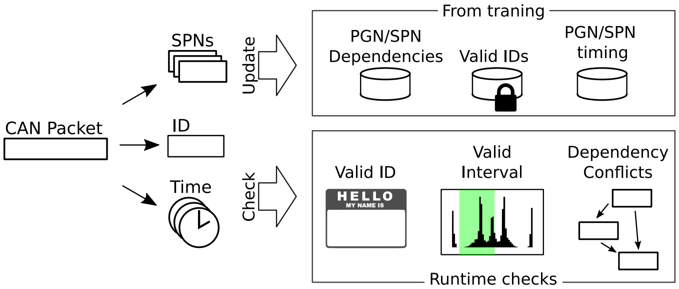


Fig. 5. CANDID Evaluation Phase. On receiving a CAN packet CANDID extracts the ID, time, and parameters. Training is checked to determine if the message has a valid ID, if the interval between the observed ID and the last occurrence matches what CANDID expects from our PGN/SPN timing training given the state of the system, and if the changing data values are correlated according to CANDID's PGN/SPN dependencies training. Using the observed data we update what state our training believes we are in. Valid IDs does not change as the system operates.

For outlier data points, such as a small set of points between two bi-modal distributions, CANDID determines which state parameter is set and includes them in the appropriate distribution. Given that a message is more likely to be delayed than significantly early, this likely increases the upperbound of the interval, making outliers less likely to result in no viable boundary condition. If there is no viable boundary, then CANDID defaults to only using a lowerbound. Such that even if the message can have a wide timing interval, CANDID still detects an attacker transmitting that message too frequently.

4.3 Evaluation Phase

The evaluation phase determines how CANDID functions on each incoming message. A flowchart of this process can be seen in Figure 5. In terms of timing CANDID calculates the interval between the observed ID and the two times it was seen. The appropriate bounds are then selected by querying the last value for the trained state parameter. If the interval is within the lower and upper bounds for the 2nd from last message, or is not within those bounds for the last message, CANDID alerts. CANDID also alerts if the state parameter has changed for each of these messages. This is indicative of an attacker changing the state parameter to change the expected timing bounds, followed by the legitimate ECU transmitting in the original bounds. In terms of data, the expected correlation for data parameters with strong correlations is queried from our training database for each parameter. If the percent change between an observed parameter and percent change in the correlated parameter deviates beyond the trained CUSUM threshold, CANDID alerts. This percent change is then stored in memory. In summary CANDID stores a tuple containing the last timestamp and state parameter for the last two messages from each ID, as well as the percent change of each parameter for each ID. All other necessary data is pulled from CANDID's training.

Taken together, CANDID is a robust intrusion detection system which combines timing and data analysis to detect spoofing attacks, and masquerade attacks where a corrupted ECU is transmitting messages correlated to trusted ECUs.

This leaves the manipulation attack, manipulating arbitrary bits as they are transmitted on the bus. In the next section we will demonstrate how simple it is to perform this attack, and the CAN features that simplify detecting it.

5 Manipulation Attacks

Manipulation attacks can change the contents of a message, without changing the timing metadata. Consequently, if manipulation attacks are not addressed, spoofing attacks are difficult to detect. Additionally, repeated manipulation attacks can lead to a bus-off attack [8] scenario within a single message, without generating a single error frame, or dropping the message, making this attack harder to detect than previous work implies. This section describes how a manipulation attack is done, the possible consequences of them, and how our IDS detects it.

5.1 CAN Mechanisms Supporting Detection

Before we go into how manipulation attacks work, we will go over important features of CAN. These features simplify our detection mechanism, such that we only have to detect ′0′ to ′1′ manipulation attacks to cover all manipulation attacks.

The CAN bus consists of two wires labeled CAN-L and CAN-H. The differential voltage between CAN-L and CAN-H determines the logical output. A ′1′ is the default and referred to as the recessive state, with a differential voltage of 0V. This recessive state means no ECU is actively driving the CAN Bus, meaning it is relatively constant for each CAN Bus. ′0′ is the dominant state with a differential voltage of approximately 2V. If any ECU is transmitting a ′0′ signal it overrides the recessive state, and transmits a ′0′ across the wire. An attacker overriding a recessive bit with a ′0′ after the arbitration field generates a CAN error.

For the purposes of this paper we are concerned with two forms of CAN errors, implemented in all ECUs. We will use these errors to ensure the bus must reach a state where we can detect a manipulation attack, specifically manipulating a dominant bit to a recessive bit.

A Bit Monitoring Error occurs when an ECU transmits one bit, and sees a different value on the bus. The only time bit monitoring errors do not apply is when an ECU transmits a logical 1 during the arbitration period, as a 0 from another ECU is designed to indicate a higher priority message.

A Bit Stuffing Error occurs when 6 of the same bits are observed in a row. After transmitting 5 of the same bits in a row, the transmitting ECU is meant to insert the opposite bit, then resume normal transmissions. All CAN transceivers know to ignore the stuffed bit.

When an ECU detects an error it sends a *CAN Error Frame* which consists of 6 dominant bits (0) in a row. The error frame itself generates a bit stuffing error, alerting every ECU on the bus that it should ignore the message it just transmitted. The bit stuffing error means a ′1′ to ′0′ bit manipulation attack, which generates a CAN error frame, inevitably forces the attacker to perform numerous ′0′ to ′1′ manipulation attacks, lest an error frame is completed and the message is dropped. Notably the victim ECU will retransmit a message interrupted by an error frame, making it an ineffective technique for silencing legitimate ECUs. The bus-off attack [8] complicates this, but is covered in our security analysis.

We make the distinction between ′1′ to ′0′ and ′0′ to ′1′ manipulation attacks as the former is intentionally simple in CAN, and difficult to distinguish from a normal bit change. On the other hand a ′0′ to ′1′ transition requires the attacker to drive the bus against the victim ECU, who is actively providing power to CAN-H, and sinking CAN-L. For the rest of this section we will focus on performing and detecting ′0′ to ‘1′ manipulation attacks, as they are the inevitable consequence of any manipulation attack where the attacker wants the bus to process the manipulated message.

5.2 Performing a Manipulation Attack

An attacker has 3 options to manipulate a 0 to a 1: ground CAN-H to CAN-L, increase CAN-L to CAN-H, or meet somewhere in the middle. In terms of detection, a manipulation attack which significantly deviates from the passive bus voltage is immediately obvious, as no ECU is transmitting during a passive bus, but the attack still works.

In Figure 6 we demonstrate a manipulation attack on a real CAN bus, changing a ′0′ to a ′1′ three times by increasing the voltage of CAN-L such that the differential voltage between CAN-H and CAN-L is 0V. In terms of hardware, we used a standard CAN transceiver to time the attack, and P-channel MOSFETs to source/sink voltage for overriding the existing signal. Each manipulation attack produces an error frame, and manipulating this frame results in another error frame. In Figure 6 we can see these repeated error frames from each manipulation attack. Error frames will stop transmitting a dominant bit after an error count of 127, and ECUs will stop transmitting entirely after an error count of 255 [8]. The errors caused by a manipulation attack increase the count by 8, meaning it only takes 16 bit manipulations for passive bit errors. Once in a passive bit error state, we can simply transmit dominant bits. After 16 dominant bit transmissions the victim ECU will enter a bus-off state, and stop transmitting until reset.

Manipulation attacks allow us to reach a bus-off state much faster, and easier than previous work, which took 32 interrupted messages, and predicted when messages would transmit to have two messages win arbitration instead of simply reading the ID as it goes across the bus [8]. We complete a bus-off attack within the span of a single message, without allowing any error frames to complete, and without having the bus discard the manipulated message.

5.3 Detecting a Manipulation Attack

We detect a manipulation attack by monitoring the current during a dominant to passive bus transition, as shown in Figure 7. Normally this transition is from no ECUs transmitting, but during a manipulation attack the attacker is actively driving the bus towards the passive state. This never occurs on a normal CAN bus. The result of this drive towards passive is an anomalous change in electric potential. CANDID can see this change in potential because each ECU, including CANDID, is connected to the bus in parallel, meaning any change in potential results in a measurable change in current from our pull up resistors. An attacker must perform a 0 to 1 manipulation attack because the initial

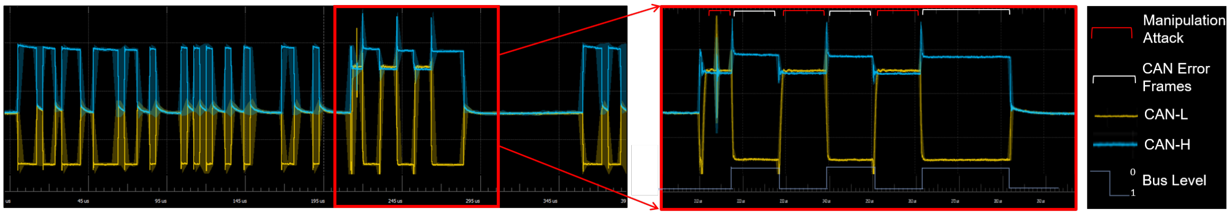


Fig. 6. This shows a manipulation attack, performed 3 times, forcing a dominant to a recessive bit. The right figure, a zoomed in section of the left figure, depicts a 4μs attack, followed by an 8μs pause, 8μs attack, another 8μs pause and attack, before allowing the error frame to complete. When a CAN error frame is interrupted, we see the ECU transmit the error frame again. After a complete error frame is sent, the line goes passive for approximately 60μs before retransmitting the manipulated message.

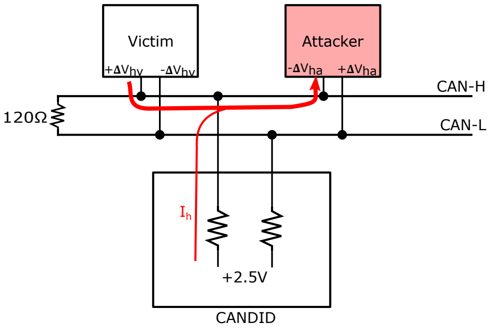


Fig. 7. Depicts active 0 to 1 manipulation attack on CAN Bus. Changing potential from the attacker overriding the victim's transmission from high to low results in current draw from CANDID pullup resistors. This does not occur on a passive bus.

bit flip induces the 6 zero bits of a CAN frame, and if that error frame completes then any standard CAN transceiver easily detects it. We disagree with the premise presented in Cho and Shin's bus-off attack paper [8] that error frames are too unreliable of a detection metric. Presumably the vehicle functions, and the bus-off state is not reached by normal ECUs even if they sometimes produce error frames. By keeping our own error counter for each ECU we can alert when an ECU is approaching the bus-off state. If an attacker attempts to circumvent error frame detection by continuously flipping bits, then they must perform a 0 to 1 bit flip as long as the error counter is below 128. Thus we alert if we see any ECU reaching an error count of 128. To summarize, after a single bit flip the victim will produce 6 zero bits. An attacker can either continue flipping bits to ensure the message is not dropped, at which point we detect the 0 to 1 transition which does not occur normally on the bus, or the attacker can allow error frames to complete until error frames start transmitting 6 one bits instead, which we detect by keeping a counter of errors produced by each ECU. If the attacker decides to let the message drop after one bit flip then there is almost no impact on the bus, as the victim ECU will immediately re-transmit the dropped message. By having our solution take advantage of CAN errors and passive bus levels, we are able to detect these attacks on any CAN system without attempting to fingerprint each ECU and dealing with the complications of electrically fingerprinting electronics, and the cat and mouse game of how much an attacker can recreate a signal.

6 Security Analysis

In the adversary model we defined a successful attack as changing the state of the system without CANDID alerting. To provide a security guarantee against an attacker we first focus on the detection of manipulation attacks and then we analyze spoofing and masquerade attacks separately. Recall that our detection system is connected as a normal ECU to the bus, with an amperage monitor on the pullup resistors in our CAN transceiver. This security analysis assumes the attacker knows CANDID is installed.

6.1 Securing Against Spoofing Attacks

The attacker has somehow added a device to the CAN bus, and wishes to modify the state of the system by impersonating existing ECUs. To perform a spoofing attack, an attacker must send a periodic, or aperiodic message. The security model is the same for both, but our approach for detecting them is different. Let us discuss the approach for an adversary transmitting periodic messages. First the attacker must select a message ID. Because of CANDID's timing training, the attacker must select a periodic ID previously observed on the system. When the attacker transmits their desired periodic ID our IDS checks the transmission interval of that message. This interval is selected based on the state of the system, and defined during training. If the time between the attacker's message and the previous time we saw that ID is not within the known transmission interval, we alert on that message. If it is within the known interval, the original device will send a message, guaranteeing an alert by the end of the interval on whichever message came second. Given this, an attacker must ensure that no other ECUs are sending their desired ID. It follows that to transmit their desired ID the attacker must prevent the original ECU from speaking. An attacker can do this by colliding with the original message, performing an address hijack attack, or uploading malicious firmware. Collisions generate a CAN error frame, which are covered under our security analysis for manipulation attacks. An address hijack attack causes the victim ECU to transmit a specific 'address not found' message, and requires the attacker to send a completely new data field. Mangling this alert message to hide the address hijack attack is secured by the same logic as collisions. Firmware upgrades are trivial to detect if we assume our IDS knows all legitimate firmware versions, and can calculate a hash of each CAN frame involved in the firmware update. This means an attacker can not prevent an ECU from transmitting undetected. Consequently, if a legitimate ECU is transmitting periodic message, we guarantee CANDID detects injected illegitimate periodic messages.

For aperiodic messages there is no existing ECU transmitting the message. Once an attacker injects an aperiodic message CANDID's data correlation training verifies that the provided aperiodic message should have been transmitted based on the state of the system. This is done through a series of data dependency checks. To bypass this check, the attacker must inject additional messages prior to their desired aperoidic message, setting the appropriate data dependencies. However, these data dependencies belong to periodic messages, meaning our security analysis for periodic messages now applies to aperiodic messages. This leaves the attacker unable to inject any messages.

6.2 Securing Against Masquerade Attacks

The attacker has taken over an existing ECU, and aims to send false messages on the bus as the victim ECU. The masquerade attack's security analysis is the same as the spoofing attack's, with three distinct differences: periodic message detection relies on data dependencies, the attacker can lie about the physical defects in the system, and CANDID is unable to detect small changes to messages with no data dependencies.

The attacker transmits messages within the known transmission interval of the victim ECU. As there is no other ECU transmitting this normally periodic message, CANDID's timing features no longer detect this attacker. However, the injected periodic message has data dependencies. To bypass these dependencies the attacker must transmit the parent messages, or transmit within the error bound allowed by the CUSUM analysis for the chosen message. Because the parent message is coming from a trusted ECU, the attacker would need to perform a spoofing attack as that ECU. This attack is protected by the aforementioned timing-based spoofing detection mechanisms. This leaves the attacker with messages within the CUSUM error threshold. By definition, attacks within this CUSUM error threshold have limited impact on the system. This is because the error threshold is determined by training on the innate fluctuations of a normally operating system.

Given that the attacker cannot successfully transmit messages with dependencies, they can only transmit messages with no data dependencies to non-corrupted ECUs. This means a corrupted ECU has control over what its victim transmits, but cannot reach outside of that domain due to data dependencies and CANDID's timing analysis.

The attacker is capable of lying about the state of the system for its independent messages. However, this lie must be physically possible in the system. Take an attacker omitting messages reporting an oil leak. The rest of the system will believe these omissions, until the lie is physically impossible for the rest of the system to maintain. At this point CANDID will alert on the dependent, trusted, ECUs as their data diverges from that expected from the corrupted ECU's messages.

6.3 Securing Against Manipulation Attacks

To perform a manipulation attack the adversary must modify the voltage of CAN-H and CAN-L such that the differential voltage translates to their desired bit. Any attempts at manipulating a bit result in the victim ECU raising a bit monitoring error, as the bit on the bus is different from the transmitted bit. If the error frame completes, then CANDID detects an error frame from the victim ECU, incrementing CANDID's internal error counter for that ECU. CANDID alerts if the error counter for any ECU exceeds 127 (where errors start transmitting recessive bits). This cutoff ensures CANDID does not transmit false positives for one-off legitimate errors, but does alert if any ECU reaches a serious error state. Allowing a number of error frames less than 255 to complete is inconsequential to the state of the bus, as every interrupt message is ignored, then retransmited. Meaning the attacker cannot achieve their goal without allowing enough error frames to complete that CANDID detects it. Now that an attacker cannot allow error frames to complete, they are left with one option - interrupting error frames. Interrupting an error frame requires flipping a '0' to a '1'. Each bit flip during an error frame results in an error count increase which CANDID does not observe. Each manipulated error frame results in the victim ECU transmitting another error frame, for at minimum the length of an extended CAN packet. The necessity of flipping a '0' to a '1' to interrupt error frames ensures CANDID does not need to be able to detect a '1' to '0' bit flip.

Let us discuss how the attacker performs this manipulation attack. When modifying the voltage over CAN-H and CAN-L the attacker can set the voltage of CAN-H and CAN-L to any value. However, the voltage of a passive (1) CAN Bus is consistent. This can be seen in Figure 6, where CAN-H and CAN-L reset after we stop our attack and the bus begins transmitting a recessive bit. If the voltage of CAN-H and CAN-L is significantly different from the passive voltage for longer than a microsecond the attack is obvious. Now the attacker must sink CAN-H and source CAN-L such that the voltage of both are approximately the same as a passive bus.

To sink CAN-H and source CAN-L the attacker must overpower the victim ECU, who is actively driving CAN-H high and CAN-L low to transmit an error frame. The resulting change in potential results in current flowing towards the attacker's CAN-H connection, and away from the attacker's CAN-L connection. We observe these draws in current through the pullup resistors in CANDID. Afterall each ECU is connected in parallel to the CAN Bus. In contrast, the passive bus has no transmitting ECUs, meaning a near 0 change in potential and current. CANDID monitors the current across its pullup resistors whenever the bus is transmitting a '1'. If that current is different from the near-zero current of a passive bus, then we alert. Given that a critical mass of CAN error frames result in detection, the attacker must eventually transmit a '1', and the current will be different from that of a passive bus. This leaves the attacker unable to perform a manipulation attack undetected. Now an attacker must inject new messages onto the CAN bus to perform a successful attack. Our security analysis for spoofing and masquerade attacks covers these new messages being injected onto the bus.

6.4 Security-Analysis Conclusion

We guarantee detecting all periodic messages, and data dependent aperiodic messages for spoofing attacks. For

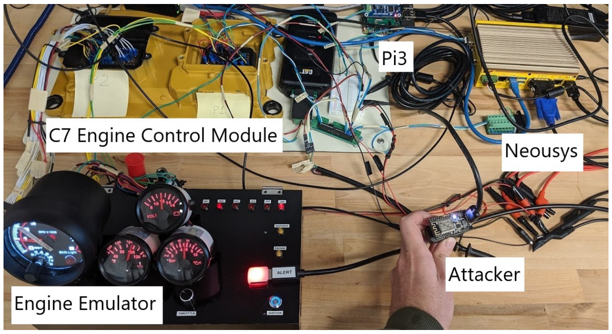


Fig. 8. Testbench with engine emulator in bottom left, C7 Engine Control Module in top left, collection devices in top right, and implant attacker in hand in the bottom right. The red light in the middle left represents a mechanism for alerting the operator of the vehicle.

masquerade attacks, we detect data dependent periodic messages and data dependent aperiodic messages. Our ability to detect any manipulation attack ensures CANDID can rely on the required data from trusted ECUs.

7 Implementation and Experimental Results

This section tests the performance of our implementation, and demonstrates our experimental results.

7.1 Implementation

Our implementation works on both a dump of CAN packets, and live connection from the CAN Bus. That live connection used one of two collection devices, a Neousys ruggedized vehicle computer [27], or a Raspberry Pi 3 CAN Hat [44]. For development and attack testing we used a testbench with a C7 Caterpillar Diesel Engine Control Module, an engine emulator, and an OBD-II computer acting as an implant, and both aforementioned collection devices connected to the same CAN bus. This can be seen in Figure 8. All live testing was done on a 250KB/s CAN Bus.

For our manipulation attack detection we install an amperage monitor in series with the CAN transceiver. Alternatively we could connect a normal oscilloscope to the CAN-H and CAN-L connections with a common ground and measure the difference in voltage. But this requires knowledge of the impedance of the bus line, making it ineffective against implants.

All J1939 specification parsing, rule generation, and IDS functionality is written in Python, primarily for rapid feature development. CAN frame collection is written in C++ and uses Linux PCAN drivers to process incoming messages. We tested the performance of our system using profiling on a data capture, and testing on a fully operational moving vehicle with two CAN busses. By testing to see if the time it takes for CANDID process to a CAN packet is less than the time for another CAN packet to be transmitted we know if CANDID can be used in real world settings. Our profiling indicates CANDID enriches and evaluates an individual CAN packet in less than 500 microseconds, even when duplicating the CANDID process to work on both CAN Busses. On a 250KB/s bus a single CAN packet takes 584 microseconds, meaning CANDID is fast enough to not fall behind the bus. To test CANDID's performance on a real bus we turned off the rest of the bus while a separate device transmitted a test alert message. We observed this test alert instantaneously, indicating CANDID's processing did not fall behind.

7.2 Experiment

Evaluating CANDID requires us to test it for each attack, and against a data set without attacks. To this end we test CANDID against 6 representative attacks, and verify CANDID's behavior against a clean dataset. We do not perform experiments on manipulation attack detection beyond executing the attack and validating repeated CAN Error Frames as described in Section 5.

For our evaluation we have access to datasets from a truck and four tactical vehicles. CANDID is also evaluated while deployed on a full system to ensure CANDID can keep up with real bus speeds, as described above in the Section 7.1. From a detection evaluation perspective CANDID's timing and data capabilities perform the same on a collected data captures as a real bus. We have three hours of driving data from our truck dataset and 145, 55, 170, and 16 data captures for the four tactical vehicle respectively. These captures average 200,000 CAN packets each, with varied driving conditions including: off-road, hill, track, and road testing with faults. Across our datasets we observe 124 different message types, 328 data parameters and 72 million packets. The observed message types are reflected in both datasets, though the tactical vehicles datasets include 41 types not seen in the truck dataset.

For our evaluation we need to create a training set off of these datasets. Based on the differences in observed messages, training is not transferable between vehicles with different ECUs. We trained on roughly one million CAN packets (1-2 hours of driving time) for each vehicle. The most important aspect of this training is ensuring all message types are seen, and that any transitions between vehicles states are included in the traffic. To verify that a normally operating system does not result in false positives we run CANDID against our truck and tactical vehicle datasets. We do not run attacks on vehicles while collecting this data to establish a baseline, and for safety reasons.

For spoofing attacks we inject one of five attacks into a random spot in the aforementioned truck dataset then run the dataset against CANDID. We use randomized spots in the dataset to emulate attacker behavior, but analyze the intervals produced by CANDID's training to ensure they meet the bounding criteria necessary for the security principles around intervals to hold. We only use the truck dataset as the difference between vehicles is inconsequential for detecting attacks once the training mechanism can be trusted. Our five attacks are chosen to demonstrate CANDID's ability to detect messages with one timing interval, messages with multiple timing intervals, aperiodic messages sent as a condition of the data field, completely new messages, and J1939 specific attacks. These attacks cover the entirety of spoofing attacks, and give us confidence in CANDID's ability to detect any other spoofing attacks, in addition to common mechanisms for disabling existing ECUs.

We repeat each of the five spoofing attacks 100 times. The five attacks are: spoofing periodic engine message

TABLE 2
Summary of results of 6 practical attacks. For each attack, a message is injected or replaced randomly in a trace of 1.5 million packets. Each attack is repeated 100 times.

Description	Attack	Precision (%)	Recall (%)
Periodic Engine Message	Spoofing	100	100
Cruise Control Message	Spoofing	100	100
Enable Engine Diagnostic	Spoofing	100	100
New Message Injection	Spoofing	100	100
Address Hijack	Spoofing	100	100
Corrupted ECU	Masquerade	100	93.3

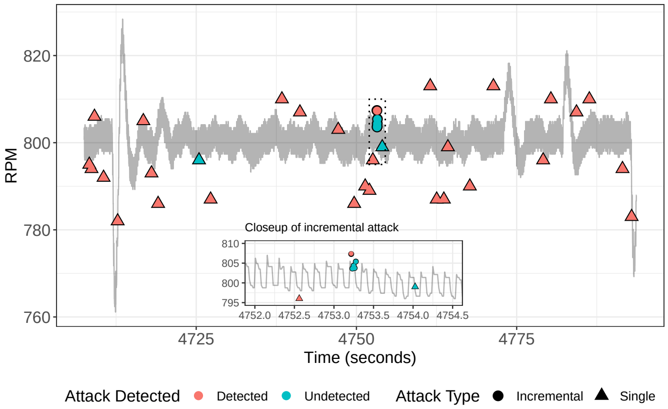


Fig. 9. Masquerade Attack, with a zoom in on the incremental attack. Here an attacker modifies the RPM as a corrupted engine ECU. Attacks are detected using CUSUM on deviations from trained correlations. Undetected attacks have an insignificant impact on the system.

61445, spoofing the multiple timing cruise control message, spoofing an aperiodic diagnostic message which enables the check engine light, transmitting a message type not previously seen by the engine, and an address hijack attack. The aforementioned attacks were tested on our testbench, and work as expected. To test against a masquerade attack we change the engine speed of engine message PGN 61444 at random places in the dataset. We modify 30 data fields, and add an additional case for an attacker incrementally changing the data.

7.3 Results

Our first tests involve running CANDID against truck, and tactical vehicle datasets to test if any alerts are generated on a clean dataset. After training we process each CAN packet for each dataset. We observed no false positives as a result of messages operating outside their normal timing bounds. This indicates predictability and consistency of timing intervals.

Our data analysis implementation uncovered a small number of proprietary message types in our data set (from military vehicles), which led to false positives as CANDID had no way of interpreting their data fields. Proprietary messages are not a problem for the equipment manufacturer, but lacking documentation CANDID ignores them if they do not immediately fit within one timing interval.

We found that 4.76% of IDs in our datasets have data dependent transmission times. These 4.76% of IDs, including messages like the cruise control message from our introduction, would cause false positives in existing timing-only research. At 11.86% our NMEA 2000 data has more data dependent messages than seen in our wheel-based J1939 systems, so here the benefit of CANDID is even more pronounced. CANDID was able to correctly identify the data fields which modify message timing, such that we can safely process these IDs.

To test the success of a spoofing attack we inject a single attack randomly into our truck dataset, scanning the dataset with CANDID, then scrub the dataset. We repeat this experiment 100 times for each of our 5 attacks, for a total of 500 injected messages. In terms of interval training we found that CANDID was able to produce intervals with lowerbounds greater than the width of the interval, such that the security principle holds. For the masquerade attack we modify a data value in a message, rather than inject a new one. We do this 30 times, plus an extra 5 messages in a sequential replacement attack. Table 2 depicts the results of our experiment. For the spoofing attack we achieve 100% recall and precision. For the masquerade attack we achieve a 93.3% recall and 100% precision. Attacks that change the value to something within the variance of the transmission are not detected, but such attacks have a limited impact on the system.

The *Periodic Engine Message* attack demonstrates detecting fixed rate periodic traffic, where each attack causes the victim ID's intervals to diverge from the fixed rate. The *Cruise Control Message* attack uses the same principle as the Periodic Engine Message attack, with multiple intervals. Detecting it requires CANDID to track the state of the system, select the appropriate time interval, then alert on any messages outside that time interval. The *Enable Engine Diagnostic* attack extends the periodicity proven by our periodic engine message to an aperiodic message, as the chosen diagnostic only turns on when the engine speed is over 2400 RPM. The *New Message Injection* attack proves that CANDID can detect unknown IDs, this emulates the heartbeat functionality of a popular CAN cryptography solution [31], but for extended CAN packets and without limiting the system configuration. The *address hijack* attack requires CANDID to save the known addresses for each address, and alert on any new data fields.

Explaining the corrupted ECU attack and what we do and do not detect requires more detail. The results can be seen in Figure 9. In this case our data analysis finds engine speed is dependent on the rate of change of transmission input shaft speed, meanwhile it is in the engine idle state. This state means the engine speed has some natural fluctuations, which we account for with a CUSUM error threshold of 1.2%. A chain of incremental attacks, seen in the zoomed in section of Figure 9, remained undetected over successive smaller changes until the final message where the cumulative error was larger than possible in that idle state time period. CANDID does not detect some small deviations, resulting in false negatives. Even so, the attacker is limited in their desired engine speed, despite controlling the engine control module. Further restricting changes to the data field is likely to result in many false positives, though may be necessary for systems where small changes to scalar values are of consequence.

In summary, tests against a clean dataset results in false positives from enrichment errors with proprietary messages, but has high accuracy on standard traffic. While an implementation can approximate out proprietary values, it depends on how sensitive the defender is about small fluctuations in data. We are able to detect all of our spoofing attacks. This includes periodic, multi-periodic, and aperiodic messages. Identifying these data dependent periodic messages makes the bus more predictable for the defender. Our experiment against a masquerade attack is detected through data dependencies to periodic messages. Detecting multi-periodic, and aperiodic messages from any attacker, as well as masquerade attacks from periodic messages is a significant improvement on previous CAN intrusion detection research.

7.4 Comparison to Existing Work

To contextualize the results of CANDID we will take the three modules of CANDID and compare them to existing work. We define our three modules as a timing module (which encompasses the interval training, and modality based interval training), a data-dependency module, and a manipulation attack detection module. The modality-based timing approach is an improvement on existing CAN timing based IDS approaches as it removes the assumption that messages are strictly periodic. An assumption that is known to be false for some subset of messages [22]. While post-detection-analysis can reveal aperiodic false positives [9], it does not provide a security guarantee to those messages. CANDID provides an improved timing interval bounding approach, which is then extended to aperiodic messages via this modality training. And it does this in a way that is unspoofable in our system model, unlike clock skew analysis which can be faked [34]. An attacker cannot fake an interval as long as the legitimate ECU is still transmitting, and our IDS is designed to catch the attacks that would disable a legitimate ECU. The result is every message has a security guarantee that is non-spoofable. An attacker cannot avoid CANDID simply by choosing a message with a complicated transmission interval.

Now let us examine our data-dependency module. Existing CAN work applies data analysis via an Artificial Neural Network (ANN) which learns how data is expected to deviate in sliding time windows [43]. CANDID's timing analysis exists separately from the data-dependency training, which ensures that an attacker cannot overload the bus with illegitimate messages. Quinonez's physical invariance research has similar timing issues [32]. CANDID's use of dependencies between data fields, with CUSUM applied to the deviation from the expected value, is novel in that it is applied across the entire vehicle. The physical models presented by Wasicek [43] and Quinonez [32] are applied to specific subsystems, and simplistic line following robots respectively. This results in assumptions about which systems are interconnected and risks gaps in detection coverage. Our layering of data contextualized timing analysis further limits the set of messages an attacker can send, while ensuring an attacker is limited when sending malicious data correlated to legitimate ECUs.

Our manipulation attack detection is most similar to voltage based detection systems. These monitor the voltage of the bus to create electrical fingerprints of the system [20], [10]. While not explicitly designed to detect manipulation attacks, the sudden change in voltage required to flip a bit would likely modify the fingerprint of the message, and so trigger the detection system. CANDID offers two novel improvements. The limitation of the monitoring system from fingerprinting the whole system to only 0 to 1 bit flips reduces false positives by removing the need for constant retraining. Overlapping our manipulation attack detection with timing-based spoofing detection ensures we do not lose any of the security guarantees delivered by voltage fingerprinting. The second improvement is using the current in the pull up resistors during 0 to 1 transitions. In a generic sense monitoring only current would create a false positive ridden detection system. However, the nature of the CAN bus and its error process ensures that CANDID knows when a 0 to 1 transition must occur in response to a bit flip, and that when it does any deviations from that particular passive state must be erroneous. This comparison to a passive state ensures that our approach is suited to detecting voltage IDS circumvention techniques like those used by DUET [3], which uses multiple attackers to hide the voltage being changed. Any attempt to overpower the grounding of a CAN line, and thus generate a current in CANDID's pull up resistors, is inherently anomalous during a 0 to 1 bit transition.

8 Discussion

In this section we will further discuss CANDID's effectiveness for detecting attacks with limited training, and what to do after an alert.

8.1 Training

As described in Section 4.1, CANDID has 3 main training components: training on timing intervals, using data to split those timing intervals, and training on a CUSUM error threshold for correlated data fields. This training has to be done for each vehicle, preferably with a representative dataset of the system: powered on, idling, and driving in variable conditions.

The question is then what happens if the vehicle diverges outside of our training. We are unconcerned about this for two reasons. First, CANDID trains on normal behavior, then expands the timing intervals to the maximum possible while still maintaining our outlined security principles. This ensures our timing intervals are not prone to false positives from poor training sets, but are still effective at detecting spoofing attacks. Secondly, CANDID accounts for fluctuations in vehicle data by using CUSUM, as we can safely assume the vehicle will self correct and minor errors will not propagate through a vehicle indefinitely. Our results show that an attacker can make small changes to the data field, but none more significant than the natural fluctuations of normal data. These measures increase our confidence that even if our training did not capture all outliers, it is resilient to future outliers. This makes CANDID an effective and resilient IDS.

8.2 After an Alert

The question with any intrusion detection system is how an alert should be communicated, and what can be done about it. While a full incident response process for CAN is outside the scope of this paper, we can make a few suggestions. Any communication of the alert to the operator of the vehicle must be done through a mechanism external to the compromised CAN bus. In our testbench, seen in Figure 8 we use an LED light, but more context could be helpful. For example, if an alert indicates the communications system is no longer transmitting the appropriate messages, the operator can know to react accordingly. As for broader incident response, CANDID compresses and logs all CAN packets observed on the bus. This makes it a simple process to upload the messages to any traditional incident response platform such as Splunk or ELK. The alerts are logged with the alerting message, and its timestamp, such that analyzing the messages surrounding an alert is trivial.

9 Conclusion

In this paper we proposed CANDID, a novel IDS which detects anomalies on the CAN Bus via timing, data, and voltage techniques. Combining, improving, and developing these techniques ensures CANDID is more accurate than previous work, as it has greater context for how the system is meant to operate, without the cons of any individual approach. CANDID's timing analysis isolates transmission influencing data to more accurately bound when messages are going to appear, capturing safety critical messages, such as cruise control, even if they transmit with multiple timing intervals. CANDID's data analysis takes our much larger set of parameters into account, performing correlation analysis to determine when an attacker is modifying data influenced by the rest of the system. This limits what even the most advanced attacker can do on the system. Finally the voltage analysis avoids traditional fingerprinting to use a more focused approach. It specifically monitors for an attacker manipulating a 0 to a 1, using the physical properties of the CAN Bus to the defender's advantage. This capability is particularly important in the context of our advanced bus off attack contribution, which disables a transceiver within a single message, without producing any error frames. CANDID was tested against multiple vehicles and data captures, and was able to detect all injected attacks without producing false positives beyond small data translation limitations. The success of these tests, our device not requiring the modification of any ECUs, and the performance of our implementation, demonstrates that CANDID is a practical and effective solution to security for existing and future CAN systems.

References

[1] Andrey Akinshin. Lowland multimodality detection, Nov 2020.

[2] Omar Y. Al-Jarrah, Carsten Maple, Mehrdad Dianati, David Oxtoby, and Alex Mouzakitis. Intrusion detection systems for intra-vehicle networks: A review. *IEEE Access*, 7:21266–21289, 2019.

[3] Rohit Bhatia, Vireshwar Kumar, Khaled Serag, Z Berkay Celik, Mathias Payer, and Dongyan Xu. Evading Voltage-Based Intrusion Detection on Automotive CAN. *Ndss 2021*, (February), 2021.

[4] Robert Bosch. CAN specification version 2.0, 1991.

[5] Yelizaveta Burakova, Bill Hass, Leif Millar, and André Weimerskirch. Truck hacking: An experimental analysis of the SAE j1939 standard. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX, 2016. USENIX Association.

[6] Matthew Butler. An Intrusion Detection System for Heavy Duty Vehicle Networks. In *ICCWS*, volume 12, pages 399–405, 2017.

[7] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[8] K. Cho and K Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1044–1055, New York, NY, USA, 2016. Association for Computing Machinery.

[9] K. Cho and K Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 911–927, Berkeley, CA, USA, 2016. USENIX Association.

[10] 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*, 13(8):2114–2129, 2018.

[11] Roderick Currie. Hacking the can bus: Basic manipulation of a modern automobile through can bus reverse engineering. *SANS Institute*, 2017.

[12] Jeremy Daily, David Nnaji, and Ben Ettlinger. Securing CAN Traffic on J1939 Networks. *NDSS Automotive and Autonomous Vehicle Security (AutoSec) Workshop*, (February), 2021.

[13] Jennifer Cheeseman Day and Andrew W. Hait. Number of truckers at all-time high. *US Census Bureau*, July 2019.

[14] Fortune Business Insights Pvt. Ltd. Marine vessel market size, share and industry analysis, by type (commercial vessel, passenger ship, lng/lpg carrier, and special purpose vessel), by system (marine engine system, sensor system, control system, electrical system, auxiliary system, and communication system), and regional forecast, 2019-2026.

[15] Hristos Giannopoulos, Alexander M Wyglinski, and Joseph Chapman. Securing vehicular controller area networks: An approach to active bus-level countermeasures. *IEEE Vehicular Technology Magazine*, 12(4):60–68, 2017.

[16] Bogdan Groza, Stefan Murvay, Anthony van Herrewege, and Ingrid Verbauwhede. LiBrA-CAN: Lightweight broadcast authentication for controller area networks. *ACM Transactions on Embedded Computing Systems*, 16(3), 2017.

[17] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional CAN bus data. *CoRR*, abs/1906.02492, 2019.

[18] FRANK E Harrell and C E Davis. A new distribution-free quantile estimator, 1982.

[19] Chris Hartnoll. Rolls-Royce: Optimising jet engine maintenance with machine learning. *Harvard Business School*, 2018.

[20] Marcel Kneib and Christopher Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 787–800, New York, NY, USA, 2018. Association for Computing Machinery.

[21] 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*, pages 447–462, May 2010.

[22] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. Cannon: Reliable and stealthy remote shutdown attacks via unaltered automotive microcontrollers. *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[23] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *Proceedings - 2017 15th Annual Conference on Privacy, Security and Trust, PST 2017*, Proceedings - 2017 15th Annual Conference on Privacy, Security and Trust, PST 2017, pages 57–66. Institute of Electrical and Electronics Engineers Inc., September 2018. Funding Information: This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TB1403-00.; 15th Annual Conference on Privacy, Security and Trust, PST 2017 ; Conference date: 27-08-2017 Through 29-08-2017.

[24] C. Miller and C Valasek. Remote exploitation of an unaltered passenger vehicle. defcon 23 (2015).

[25] Subhojeet Mukherjee, Hossein Shirazi, Indrakshi Ray, Jeremy Daily, and Rose Gamble. Practical DoS Attacks on Embedded Networks

in Commercial Vehicles. In *International Conference on Information Systems Security*, volume 10063, pages 23–42, 2016.

[26] P. Murvay and B. Groza. Security shortcomings and countermeasures for the SAE j1939 commercial vehicle bus protocol. *IEEE Transactions on Vehicular Technology*, 67(5):4325–4339, May 2018.

[27] Neousys Technology. Poc-351vtc series.

[28] E. S. PAGE. CONTINUOUS INSPECTION SCHEMES. *Biometrika*, 41(1-2):100–115, 06 1954.

[29] Krzysztof Pawelec, Robert A Bridges, and Frank L Combs. Towards a CAN IDS based on a neural-network data field. arXiv, 2017.

[30] Mert D. Pesé, Troy Stacer, C. Andrés Campos, Eric Newberry, Dongyao Chen, and Kang G. Shin. LibreCan: Automated can message translator. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 2283–2300, 2019.

[31] O. Pfeiffer. *Implementing Scalable CAN Security with CANcrypt: Authentication and Encryption for CANopen, J1939 and Other Controller Area Network Or CAN FD Protocols*. Embedded Systems Academy Incorporated, 2017.

[32] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Zhiqiang Lin. SAVIOR: Securing autonomous vehicles with robust physical invariants. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 895–912. USENIX Association, August 2020.

[33] Sang Uk Sagong. *Physics-based Security Analysis of Controller Area Network Protocols*. PhD thesis, University of Washington, 2019.

[34] Sang Uk Sagong, Xuhang Ying, Andrew Clark, Linda Bushnell, and Radha Poovendran. Cloaking the Clock: Emulating Clock Skew in Controller Area Networks. *Proceedings - 9th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2018*, pages 32–42, 2018.

[35] Carol Scheina. Abrams demonstration proves concept for enterprise-level system health monitoring, Sep 2020.

[36] Craig Smith. *The Car Hacker's Handbook: A Guide for the Penetration Tester*. No Starch Press, San Francisco, CA, USA, 1st edition, 2016.

[37] Society of Automotive Engineers.

[38] H. M. Song, H. R. Kim, and H. K. Kim. 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)*, pages 63–68, 2016.

[39] Orly Stan, Adi Cohen, Yuval Elovici, and Asaf Shabtai. Intrusion Detection System for the MIL-STD-1553 Communication Bus. *IEEE Transactions on Aerospace and Electronic Systems*, 9251(c):1–17, 2019.

[40] Adam Tanner. Data monitoring saves some people money on car insurance, but some will pay more.

[41] Joseph Trevithick. U-2 spy plane got new target recognition capabilities in first ever in flight software updates, Oct 2020.

[42] US Federal Motor Carrier Safety Administration, Dec 2015.

[43] A Wasicek, Mert D Pesé, André Weimerskirch, Yelizaveta Burakova, and Karan Singh. Context-aware intrusion detection in automotive control system. *Escar*, 2017.

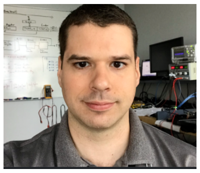
[44] Waveshare. 2-channel can expansion hat for raspberry pi, dual chips solution.

[45] Clinton Young, Joseph Zambreno, Habeeb Olufowobi, and Gedare Bloom. Survey of automotive controller area network intrusion detection systems. *IEEE Design and Test*, 36(6):48–55, 2019.

[46] Aiguo Zhou, Zhenyu Li, and Yong Shen. Anomaly detection of CAN bus messages using a deep neural network for autonomous vehicles. *Applied Sciences (Switzerland)*, 9(15):1–12, 2019.

Matthew Rogers is a PhD student at the University of Oxford on a Rhodes Scholarship. He is focusing on serial data bus security, primarily CAN, MIL-STD-1553, and commercial avionics systems.

Jassim Happa is a lecturer in Information Security at University of London, Royal Holloway.



Phillip Weigand is a founding electrical engineer at Shift5 Inc, designing custom electronics to enable the security of vehicles.

Kasper Rasmussen is an Associate Professor in the Computer Science Department at the University of Oxford. He joined University of Oxford in 2013 and in 2015 was awarded a University Research Fellowship from the Royal Society in London.