Adding Cyberattacks To An Industry-Leading CAN Simulator

Hayward, J., Tomlinson, A. J. & Bryans, J.

Original citation & hyperlink:
Hayward, J, Tomlinson, AJ & Bryans, J 2019, Adding Cyberattacks To An Industry-Leading CAN Simulator. in 2019 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C).. IEEE Computer Society, 19th IEEE International Conference on Software Quality, Reliability and Security, Sofia, Bulgaria, 22/07/19.
https://dx.doi.org/10.1109/QRS-C.2019.00016

DOI 10.1109/QRS-C.2019.00016
ISSN 1368-275X
ESSN 1741-5098

Publisher: Inderscience

Adding Cyberattacks To An Industry-Leading CAN Simulator

Jake Hayward[1], Andrew Tomlinson[2], Jeremy Bryans[3]
Systems Security Group
Institute for Future Transport and Cities
Coventry University, CV1 5FB, UK.
Email: [1]haywar27@uni.coventry.ac.uk, [2]tomlin25@uni.coventry.ac.uk, [3]jeremy.bryans@coventry.ac.uk

***Abstract*—Recent years have seen an increase in the data usage in cars, particularly as they become more autonomous and connected. With the rise in data use have come concerns about automotive cyber-security. An in-vehicle network shown to be particularly vulnerable is the Controller Area Network (CAN), which is the communication bus used by the car's safety critical and performance critical components. Cyber attacks on the CAN have been demonstrated, leading to research to develop attack detection and attack prevention systems. Such research requires representative attack demonstrations and data for testing. Obtaining this data is problematical due to the expense, danger and impracticality of using real cars on roads or tracks for example attacks. Whilst CAN simulators are available, these tend to be configured for testing conformance and functionality, rather than analysing security and cyber vulnerability. We therefore adapt a leading, industry-standard, CAN simulator to incorporate a core set of cyber attacks that are representative of those proposed by other researchers. Our adaptation allows the user to configure the attacks, and can be added easily to the free version of the simulator. Here we describe the simulator and, after reviewing the attacks that have been demonstrated and discussing their commonalities, we outline the attacks that we have incorporated into the simulator.**

***Index Terms*—controller area network, automotive cybersecurity, in-vehicle network, simulation**

I. Introduction

The connectivity of modern cars has made them vulnerable to many different types of attack. The Controller Area Network (CAN) bus, found in all cars and designed to be highly reliable, has repeatedly been demonstrated to be vulnerable to cyber-attacks [1], [2], [3], [4]. Attacks have been demonstrated that incapacitate, disrupt or control car functions, or capture data, or provide misinformation to the driver or other systems. The CAN protocol lacks security features, such as encryption and authentication, that are now expected in computer networks and its vulnerability and appeal to hackers will increase as the autonomy and connectivity of vehicles increases.

On the CAN reside the electronic control units (ECUs) that control the car's performance and safety, making the CAN a prime target for automotive cyber attackers, who might be motivated by sabotage, theft, extortion, mischief, espionage and terrorism [5]. A number of attacking vectors have been documented. These include: compromised, illicitly reflashed ECUs; other devices attached to the CAN; and, external systems accessing the CAN via a gateway accessible through the OBD (On-board Diagnostics) port or IVI (In-vehicle Infotaimment) system [3].

One proposed solution is the development of intrusion detection systems for the CAN bus [6], [7], [8], [4]. Developing and testing an intrusion detection system requires the construction of representative attacks. Test data is needed to validate solutions, while system development requires an understanding of the data patterns signifying normal operation and attack manifestations. However, in the case of the automotive CAN, such data is difficult to acquire. Staging CAN attacks on real cars is costly, dangerous, and should not be done on public roads. Moreover, the vast range of car makes, models and configurations, together with the variety of driving conditions and situations, makes repeatability a massive challenge. An approach taken by some researchers (e.g. [6], [7]), is to capture CAN log data and manipulate this to reflect the likely effects of an attack. This synthetic attack data is safe to generate, but has the disadvantage that it might not accurately capture the response to the attacks of the ECUs on the CAN. An intermediate approach is to carry out the attacks on a simulator (e.g. [8], [4]). This allows attacks to be cheaply developed and tested, and can produce CAN logs that capture the effects of the attack on the bus, including the CAN protocol processes.

In this paper, we present a simulator for mimicking cyber-attacks on the automotive CAN. Our simulator uses archetypal attack scenarios derived from published attacks and is built on industry leading CAN simulation software, ensuring replication of CAN protocol processes. The simulator offers the researcher a tool for testing and manipulating attack scenarios, as well as generating CAN logs for behaviour analysis and the development and testing of intrusion detection systems.

Although some CAN simulators have been developed, they are typically configured for CAN conformance testing or functional testing, rather than for simulating attacks. For example, neither the Vector Informatik's CANoe V11 simulator [9], nor the OCTANE simulator [10] (both of which are discussed below) include CAN attack modules. We therefore extend Vector Informatik's CANoe V11 simulator by adding an attack simulator that offers configurable archetypal attacks. These are representative of the attacks commonly proposed by researchers, and allow packets to be injected either in bulk or in response to specific events. Attacks can be mixed, as well as run across multiple configurations of the CANoe simulator. The user can also set the data values for each of the data fields

in the injected packets. Our simulator module can be used with the free demonstration version of CANoe and requires no extra programming by the operator.

The contributions of this paper are:

1) We present a CAN attack simulator that allows attacks to be modelled on a popular, industry-standard automotive CAN network simulator.
2) We synthesise the published injection attacks into a representative set of archetypal manifestations, which we build into our simulator.

The rest of this paper is organised as follows. Section II presents the anatomy of CAN attacks, including accessing the CAN, carrying out the attack, and detection options. This information is used to inform our simulations. The challenges of acquiring attack data for CAN cybersecurity research, testing and development are discussed in Section III, which also presents the motivation for the CAN attack simulator. Our simulator is compared to existing CAN simulators in Section IV and our simulator is described in Section V, with examples of attacks generated using it, presented in Section VI. Finally, we summarise our conclusions and considerations for future work in Section VII.

II. The Anatomy of an Attack

In this section, we discuss the structure of typical CAN attack and defence, and locate our contribution within that. We consider the following: the opportunities for accessing the CAN; the motives and options for carrying out the attack once access has been achieved; and, the effects on the bus traffic, and the consequent implications for detection and defence.

A. CAN Access

To perpetrate any attack, the attacker will need to gain access to the CAN or its nodes. Although sub-networks in a vehicle should be protected by sufficiently secured gateways, this has been shown to not always be the case. Checkoway *et al*. [3], Hoppe *et al*. [4] and Tencents Keen Security Lab [11] have been able to bypass CAN gateways in production cars, as well as reprogram the gateway functionality. The ODB port has often been used to access the CAN in attacks (e.g. [12] [13] [4], [14]), though other access points have been demonstrated. Checkoway *et al*. [3] were able to inject packets onto the CAN bus via the car's media player, as well as gain access to systems in the car via the car's Bluetooth, FM RDS and Cellular systems, which they could then exploit to compromise the CAN ECUs. Miller and Valasek [15] remotely hacked a Jeep Cherokee via a cellular connection to its infotainment system, enabling them to inject CAN messages which controlled the steering, brakes, and acceleration. Koscher *et al*. [2] demonstrated situations in which CAN ECUs were illicitly reflashed via the OBD port, including whilst the car was being driven, and then used to stage subsequent attacks. They concluded that "many ECUs in our car deviate from their own protocol standards, making it even easier for an attacker to initiate firmware updates or DeviceControl sequences" [2]).

What's clear from these studies is that the impenetrability of the automotive CAN cannot be assumed. Access might be gained through inadequacies or imperfections in gateways; devices might be connected directly to the CAN; or internal ECUs might be illicitly reflashed to act a internal rogue units. Increases in connectivity, and the uptake of over-the-air updating, are likely to present more opportunities for breaching.

B. Attack Types and Their Bus Implications

Whatever the mode of attack access, or the impact of the attack, the manifestation of the attack on the CAN bus can be broadly considered from two dimensions: a) the timing and triggering of the attack; and, b) the effect of the attack on message data contents.

Attacks might be event triggered, such as in response to a broadcast from another ECU, or they might be triggered independently of all other events. Event triggered attacks, in which a vehicle was disrupted by the broadcasting of fabricated packets, were devised by Hoppe *et al.* [4]. An example was the broadcasting of close-window packets whenever an open window packet was detected. The authors also broadcast other counteracting packets when an event such as a speed threshold was exceeded. They also devised attacks in which packets were broadcast at regular intervals such as spoofing periodic normality-check messages for an airbag that had been disconnected. Injecting a fabricated packet to disseminate false information, and counteract the legitimate information, is likely to need to be done at high rates in order to ensure the legitimate data is over-ridden. For example, Taylor *et al.* [6] tested attack injection rates of up to 10x the normal rate, lasting up to 1 second.

Another class of proposed attacks that could be time triggered, rather than event triggered, are attacks staged to happen when the car is likely to be switched off. Attacks with the ignition off might seem pointless at first glance; however, Cho et al. [16] point out that some ECUs have a reserve power supply and will continuously monitor the system, even when the ignition is off. The authors report having been able to therefore start devices such as lights by injecting packets with the ignition off, thus creating malicious effects, such as draining the battery or re-configuring CAN properties. Additionally, Valasek and Miller [12] found a Ford was unable to start if it had previously been flooded with high priority packets whilst the ignition was off.

Some attacks involve the fabrication of data content. Koscher *et al.* [2] demonstrated attacks that altered the legitimate speed sensor readings, causing erroneous feedback on the dashboard. Miller and Valasek [1] staged a variety of attacks that required the broadcast of spoofed packets with fabricated data values that were plausible and capable of deceiving the reading ECU into believing the car was in a state different from the actual. For example, their attack which maliciously activated the Park Assist System and steering, required in addition the broadcasting of false speed and transmission readings.

In some attacks the values of the attack packet payload are irrelevant or will be random. So called fuzzing attacks are often conducted to probe systems to determine weaknesses or

to determine what values might impart a specific response. For these, the data payload might be randomly generated in the hope that some of the values will produce an observable response. Fuzzing attacks are often conducted as a precursor to a more targeted attack. Lee *et al.* [13] determined the function of some CAN packets by systematically fuzzing one of the packet's data fields while filling zeroes in the remaining data fields. They injected the fuzzed packets in batches of 1000 for each field that was fuzzed, with a gap of 10 msec between each injected packet. In another type of attack, denial of service (DoS), the contents of the data fields might be fully irrelevant; the aim being to inject high priority packets at sufficient frequencies to hamper the broadcast of other packets. Such attacks were staged by Froschle and Stuhring [17], who point out that one of the attractions of these attacks is that they can be carried out with no knowledge of the CAN database.

Although there is a broad range of attacks described in the above research, with a range of possible attack vectors; as we have discussed, their manifestation on the CAN bus can be summarised by their effects on the packet data payload, and by their effects on the packet broadcast rates. We are therefore interested in creating archetypal attack simulations that demonstrated these effects and enable control of their parameters; we discuss these in Section V.

C. Prevention and Detection

Modifications to the CAN bus to improve security have been proposed (see [18] for a summary). However, long car life-cycles, industry standards formulation, and long production schedules are strong barriers to their development and adoption [19], [20]. Consequently, there has been research into developing post-production intrusion detection systems that might detect an attack (e.g. [21], [22], [23], [24], [6]), including some industry offerings [25], [26]. Proposals typically monitor the CAN packet traffic to detect changes in packet broadcast rates, or anomalies in the packet data contents, since these are likely to be altered during the attack.

Although signature based CAN intrusion detection has been proposed (e.g. [27]), and potentially offers advantages such as low false positive rates [28], the unpredictability of CAN attacks, as well as the awkwardness of reliably updating signature databases in cars, makes it unfavourable [20] [19]. In contrast, anomaly based intrusion detection systems tend to offer lower accuracy, but offer more scope to cope with attacks that might be difficult to model or predict, as well as offering more self-learning capabilities and not needing signature database updates. Research papers offering CAN intrusion detection proposals are dominated by anomaly detection proposals, usually employing machine learning [29]. These, in particular, require ample, valid, example data sets for training and testing.

III. The challenge of CAN data collection

The collection of sufficient quantities of valid CAN data is a key problem in this domain. Anomaly based intrusion detection methods are developed using "non-polluted" data (CAN data that has not been subject to attack) and "attack data", in which known attacks have taken place. Many researchers lack the resources to test representative attacks in a realistic environment. Staging CAN cyber-attacks on cars is clearly hazardous and potentially damaging to the car. An attack might disrupt the functioning of an ECU or force the car to operate on false information, causing damage and compromising safety. Understandably, it is illegal to stage attacks on public roads in most countries, and many researchers lack private track or rolling roads.

The fact that new CAN attack possibilities to be investigated are still emerging amplifies this lack of resources. The testing of attacks and solutions will need agile test rigs that can quickly, easily and safely be tailored to replicate evolving attack scenarios.

Furthermore, the CAN dictionaries which are used to derive data values, packet IDs and broadcast patterns for a car model, are held confidentially by the manufacturers for security and commercial reasons. Thus, whilst with patience, systematic testing, and reverse engineering, the meaning of some of the CAN IDs can be determined with reasonable accuracy, researchers are unlikely to know for sure the meaning of all of the data broadcast and might rely on speculations and assumptions about the data traffic and payload.

Another solution, and the one adopted most commonly by researchers, is to create "attack data" by gathering non-polluted data from the car and manipulating it in a way considered to be representative of real attacks. The simplicity of this approach makes it appealing, but there are some issues. One drawback is that the CAN protocol processes may not be properly mimicked. For example, the CAN protocol includes arbitration processes that govern the access priority for packet broadcasts, as well as packet error processes that govern the termination and rebroadcasting of packets found to have errors. Additional complications are the broadcast characteristics and frequencies coded for each ECU. For example, Figure 1 shows the CAN arbitration process captured in a log from the simulations we developed. Although algorithms for artificially manipulating CAN logs have been proposed (e.g.[6]), ensuring accurate replicating of such characteristics in an artificially manipulated log is difficult. Also, since the CAN dictionary is unlikely to be known, the nature and outcome of the attack must remain speculative. A final issue is that the readers that can capture accurate records of CAN logs (micro-second level) are expensive.

An intermediate approach (and the one that we adopt) is the use of a simulator. In addition to their safety and relatively low cost, these approaches enable rapid, systematically controlled testing where the attack parameters can be defined and replicated across research projects and institutes. On the other hand, a simulation will always portray a simplification of reality which, in the case of CAN testing, might mean a car system comprising fewer ECUs and a reduced coverage of driving situations. In spite of these drawbacks, simulators clearly offer a potential for cheap, rapid and safe testing and development at stages where testing using actual cars might be costly or impractical.

The costs and benefits of each approach are summarised in Table I.

```
13.151020 2  C9     Tx   d 6 40 00 00 00 5C 28  ID = 201
13.151136 2  3FC    Tx   d 1 03  ID = 1020
13.160134 2  67     Tx   d 2 01 00  ID = 103
13.180134 2  67     Tx   d 2 01 00  ID = 103
13.200134 2  67     Tx   d 2 01 00  ID = 103
13.200270 2  67     Tx   d 2 FF FF  ID = 103
13.200406 2  67     Tx   d 2 FF FF  ID = 103
13.200636 2  64     Tx   d 8 89 11 14 2D B2 02 7A 0D  ID = 100
13.200876 2  66     Tx   d 8 89 11 00 00 B2 02 00 00  ID = 102
13.201012 2  67     Tx   d 2 FF FF  ID = 103
13.201148 2  67     Tx   d 2 FF FF  ID = 103
13.201284 2  67     Tx   d 2 FF FF  ID = 103
13.201486 2  C9     Tx   d 6 40 00 00 00 5B 28  ID = 201
13.201602 2  3FC    Tx   d 1 03  ID = 1020
13.220134 2  67     Tx   d 2 01 00  ID = 103
```

Figure 1. Extract from the CAN log of a simulated attack showing the simulator has applied the CAN protocol arbitration process. Hexadecimal packet IDs are shown in column 3. The attack injected 5 messages for ID 67 with data field values FF FF. In compliance with the CAN arbitration process, lower value IDs (64 and 66) are still broadcast at their allotted times, delaying the broadcast of remaining packets for 67.

Table I
APPROACHES TO CAN ATTACK DATA GENERATION.

Approach	Disadvantage	Advantage
Actual vehicle attack	Expensive. Dangerous. Different cars and environments make cross-project replication difficult.	Validity - data captured from an actual system.
Manipulation of actual CAN logs	CAN protocol processes may not be adequately mimicked. CAN dictionary unlikely to be known. The nature and outcome of the attack is speculative. Accurate capture of CAN logs requires expensive readers.	Safe, with no damage to car. Potential for replication across projects. Known manipulations make data set labelling and testing easier.
Simulator	Simplified model. Systems are typically made for testing conformance and functioning rather than CAN attacks.	Safe. Cheap. Potential for replication across projects.

IV. RELATED SIMULATOR WORK

As already discussed, there have been a few proposals for automotive CAN simulators. However, most of these are configured for testing conformance to the CAN protocol, or testing the functionality of ECU code. An attack using a CAN simulator was shown by Fowler *et al.* [30], who use custom-made cables to connected a dongle to a powered testbed that included Vector's CANoe simulator. Though the dongle they injecting CAN packets to turn the headlights on or off. CANoe was also used by Hoppe, Kiltz and Dittman [4] to demonstrate an attack which sent erroneous window control messages.

Huang *et al.* [31] created ATG (Attack Traffic Generation) which simulates CAN attacks. The tool supports a range of attacks, offers scripting for replicability, and is written to support adaptation and independent contributions. It requires a CAN hardware layer, for which they propose USB2CAN, as well as a configuration of open Python libraries.

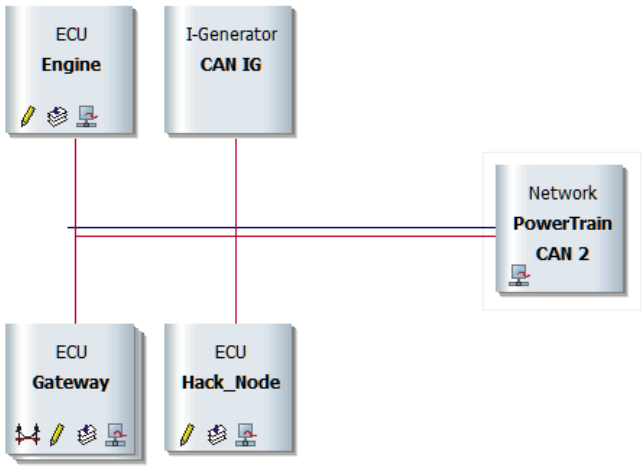

Figure 2. The simulator adds an ECU attack node (Hack_Node) to the existing CANoe ECU nodes. The attacks are instigated and controlled via this new node.

Everett and McCoy [10] developed OCTANE (Open Car Testbed And Network Experiments), a hardware and software testbed that allows CAN packets to be monitored, so their functionality can be observed, and replayed, so their modification can be tested. The tool has a comprehensive architecture stack, enabling flexible interfacing and adaptation. OCTANE does not simulate attacks, though in their testing of intrusion detection, Kang and Kang [7] adapted its logs to mimic the traces that might be generated in an attack.

Although it does not emulate the CAN protocol processes, TORCS (The Open Racing Car Simulator) [32] offers car performance simulation that might be utilised in this field. For example, it has been used to simulate the effects on engine component performance that might stem from an ECU chip tuning attack [8], as well as obtain physics-based feedback from a laboratory network of ECUs built to test an ECU code analyser [33].

Whilst tools such as the above offer potential, they require some effort in configuration and integration, typically requiring the integration of software stacks, APIs and hardware rigs.

V. THE ATTACK SIMULATOR

Our CAN Attack simulator is built on the CANoe simulator available from Vector Informatik [9]. To create and control the attacks, we add an additional Attack ECU (`Hack_Node`) on to the CAN network (Figure 2).

The Attack ECU was created using Communication Application Programming Language (CAPL), an event focused language developed by Vector Informatik to support their CAN tool-set [34]. To launch it the user opens our CANoe configuration file using the CANoe File menu. It is configured to work in conjunction with the rest of the CANoe CAN nodes, and is controlled via an interface in order to launch selected attacks. The simulator contains a series of panels which are used to navigate through and operate each attack. Depending on the attack, our simulator provides the mechanisms for the

user to control the data values broadcast as well as control the size and timings of the packet injection load. While running the attacks the simulator retains a log of the CAN bus that can be used for later analysis. Multiple panels can be used simultaneously, thus allowing different attacks to be merged. These allow the user to study the effects on the simulator of the following types of injection attacks:

- Targeted ECU Attack: broadcast of spoofed packets containing specifically determined data values.
- Counteraction Attack: broadcast of spoofed packets when their legitimate broadcast is detected.
- Denial of Service (DoS): injecting of packets with high priority IDs.
- Data-Fuzzing Attack: injecting packets with any permissible data values.
- Attack with Ignition Off: the effects, if any, of injection attacks can be tested while the ignition is off.

We have developed the attack generator on the licensed version of CANoe, and if the user has access to a full Vector CANoe license the Attack ECU can also be transferred for use in other CAN networks. It can also be operated using a demo version of the CANoe software, although restrictions on the size of the network will apply. The attack simulator module is available on GitHub[1].

VI. Attack Types Supported by the Simulator

In this section, we present the types of attacks that can be generated using the simulator.

A. Message injection

The first attack is a message injection attack which places a number of packets with a user defined composition onto the CAN network. Message injection adheres to CAN protocol arbitration, with each message being injected as soon as the bus is available. To instigate the attack, the user first selects the message ID that they are targeting, and then selects the number of messages that are contained within the attack, and assigns values to be used in the packet's data fields. The attack can then be initiated. The user can attack the CAN bus as many times as desired. Figure 3 shows the message injection attack on the CAN ID responsible for engine speed.

This is a simple attack configuration, yet allows the user to test a variety of attacks that might seek to flood the network (e.g. Overloading or DoS attacks) or target an ECU with packets containing fabricated values. We have created an additional variation of this attack that allows the user to specify the attack start time.

B. Event-triggered action

The second attack type is event based, and is triggered each time a specific message is intercepted by the Attack ECU. In response to this interception, the Attack ECU places a message onto the CAN bus with a composition chosen by the user.

In the example in Figure 4 we re-perform the attack of Hoppe *et al*. in [4]. The passenger window is the attack target.

[1]https://github.com/JakeHayward/Attack-ECU-Simulator-CANoe

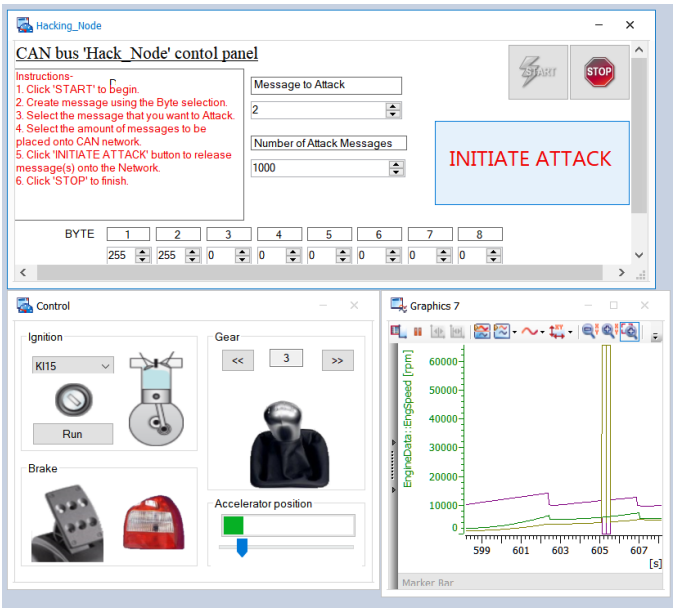

Figure 3. In the message injection attack the upper panel allows the user to select a message (ID 2 above), the number of instances to be injected (1000 above) and their bit-wise composition along the bottom of the upper panel. The INITIATE ATTACK button sends the messages onto the CAN bus in adherence to the CAN arbitration policy. The simulation can be started and stopped using the controls in the upper right. The effects of the attack can be observed in the CANoe output in the lower two panels. The left panel shows an instantaneous view and the right panel (the historic view) shows reported engine speed which spikes at the point the attack takes place.

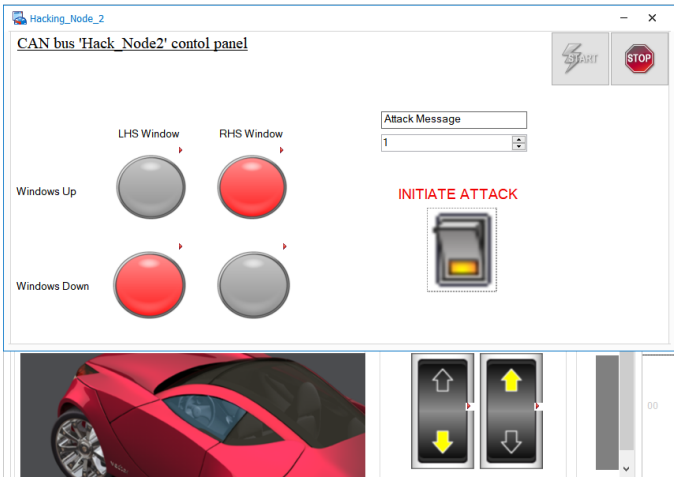

Figure 4. The second attack panel instigates an event-triggered message injection attack. Shown here the attack injects "lower left window" messages whenever a "raise left window" message is detected on the CAN, thus preventing the left window from closing.

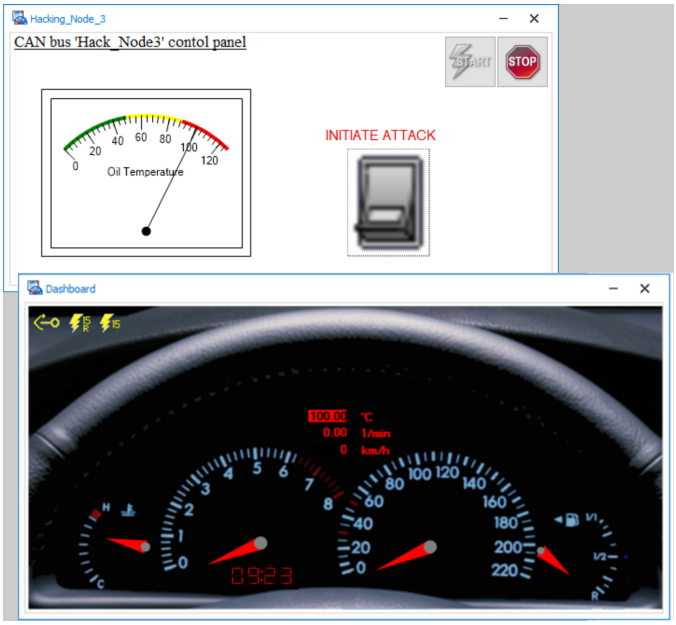

Figure 5. In the data display manipulation above, the upper panel controls an injection attack that manipulates the oil temperature sensor reading. The attack rebroadcasts oil temperature packets, but with increased dangerously hot temperature values (100 degrees, in the lower panel).

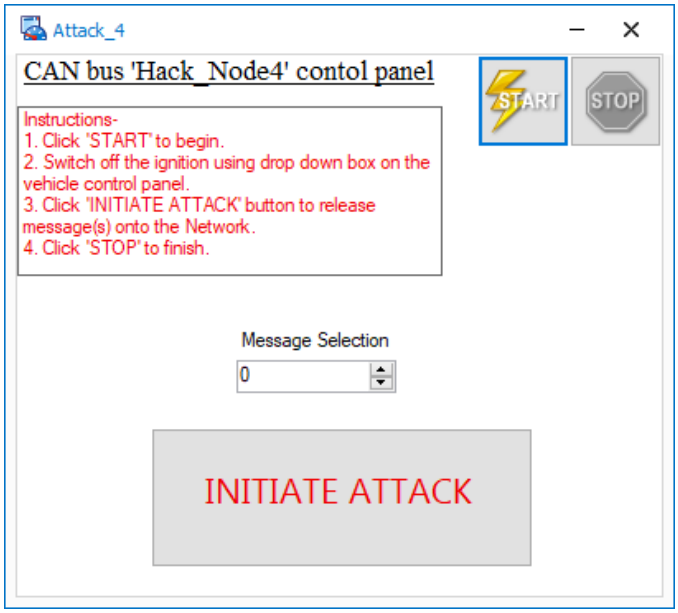

Figure 6. The fourth attack panel enables an ignition-off injection attack to determine if an ECU will receive a CAN packet whilst the ignition is off. The message selection allows the user to select a message ID to use in the attack.

When the attack is initiated the Attack ECU node will wait for the message ID that corresponds to raising the left-hand window of the vehicle. The attack node will then send out a CAN message that corresponds to lowering the window. This will cause the main ECU to receive a "lower window" message immediately after it receives a "raise window" message, meaning the passenger window can't be raised. During the simulation this attack is visually demonstrated in the animation of the vehicle built into the simulator.

C. Data display manipulation

The third attack (Figure 5) systematically increments the data values on a specific CAN ID, resulting in an inaccurate, changing sensor display. The attack uses the CAN ID that broadcasts the value of the vehicle oil temperature sensor. The simulation's predefined behaviour increments the oil temperature from 0x00 to 0x4B (75 degrees). The temperature then goes to steady-state and remains constant throughout the rest of the simulation. The attack will appear to cause the oil temperature to increase over time. It does this by sending out CAN packets of the same ID but with a changing composition which incrementally increases the oil temperature to temperatures higher than the steady state of the simulation seen in Figure 3.

D. Ignition off attack

The purpose of this attack (Figure 6) is to explore the behaviour of the CAN bus when the ignition is switched off. When the ignition is off, a message is injected by the Attack ECU to determine if the remaining ECUs can receive and process CAN packets. Ignition off attacks on real vehicles have resulted in some ECUs being activated [12] or deactivated by being forced into a bus-off state [16]. This attack is similar to the attacks described in the first attack, with the exception that it can be triggered when the ignition is off. The full attack is still under development, and attacks currently have a constant data value (0x01). This could be extended into a full fuzzing attack in which multiple IDs are targeted [30].

VII. Conclusions and Future Work

Simulating attacks, while not a replacement for thorough testing using actual cars and driving situations, is useful, cheap, safe, and can be systematically controlled. In this paper we have presented an attack simulator that extends an openly available, industry accepted CAN simulator by adding attack scenarios similar to those proposed by other researchers. The simulator faithfully replicates the changes in broadcast rates and timings that would result from the CAN arbitration process in an actual car.

The simulator allows the real-time effects of an attack to be observed in the car's body, engine and dashboard renditions, as well as in the graphical and numerical analysis tools built into CANoe. Perhaps, though, the most useful consequence is that it enables the easy generation of CAN log files that can be analysed or used for experimental data. Although the demonstration simulator we used has a smaller number of mimicked ECUs than would be found on a real car, further ECUs, including additional physical hardware nodes, can be integrated using a licensed version of the Vector CANoe simulator.

One of the challenges we faced was in attempting to parameterize the selection of CAN IDs and attack variables to enable the quick and easy selection of permissible options. For example, specifying attack start times prior to starting the

simulation run, has proven trickier than we anticipated, though we will continue to explore options.

Another area that we will continue to explore is the range of attacks replicated. The attacks we have have built so far are black box by nature —forcing change in the car functioning without concern of the internal functioning of the ECUs. Whilst this matches many of the attacks proposed and demonstrated on real cars (such as those discussed in Section II), attacks have also been proposed that force nodes into a dormant error-counter induced bus-off state by illicitly changing bitrates [16] or broadcasting fabricated erroneous messages [35]. The ECU source code and dictionary specifications provided in the CANoe simulation would offer scope to construct such attacks safely, and without the reverse engineering entailed in testing these attacks on real cars with unpublished node configurations. Also, as with monitoring other attacks, the simulation allows the the capturing of parameter values (such as the error counter values) as the attack progresses, facilitating an in-depth analysis of behaviours and responses.

A natural progression of the simulator would be to interface it with experimental intrusion detection systems so that these can be tested in real time. The range of attacks should be also extended since at present our work focuses on packet injection scenarios. Reflashing has frequently been proposed as a method of tuning ECUs into attack vectors. The mimicking of reflashing effects, such as absent broadcasts during the reflash, or even the potential to stage a reflash of simulator ECUs, would be a useful exploration.

VIII. Acknowledgements

The work reported took place during a Research Internship awarded to the first author by Coventry University.

References

[1] C. Miller and C. Valasek, "CAN Message Injection: OG Dynamite Edition," Tech. Rep., 2016. [Online]. Available: http://illmatics.com/can message injection.pdf

[2] 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 *IEEE Symposium on Security and Privacy*, Oakland, CA, 2010, pp. 1–16.

[3] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *20th USENIX Security Symposium*, vol. August. San Francisco: The USENIX Association, 2011, pp. 77–92. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028073

[4] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networksPractical examples and selected short-term countermeasures," *Reliability Engineering and System Safety*, vol. 96, no. 1, pp. 11–25, 2011.

[5] The Institution of Engineering and Technology (IET) and The Knowledge Transfer Network (KTN), "Automotive Cyber Security: An IET/KTN Thought Leadership Review of risk perspectives for connected vehicles," Tech. Rep., 2015. [Online]. Available: https://www.theiet.org/media/2309/iet-automotive-cyber-security-tlr-lr-1.pdf

[6] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," *2015 World Congress on Industrial Control Systems Security, WCICSS 2015*, pp. 45–49, 2015.

[7] M.-J. Kang and J.-W. Kang, "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security," *PLoS ONE*, vol. 11, no. 6, 2016.

[8] A. Wasicek and A. Weimerskirch, "Recognizing Manipulated Electronic Control Units," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, pp. 1–8, 2015.

[9] Vector Informatik GmbH, "CANoe: Product Information (V6.0.04/2018)," Tech. Rep., 2018.

[10] C. E. Everett, "OCTANE : Open Car Testbed And Network Experiments Bringing Cyber-Physical Security Research to Researchers and Students," in *6th Workshop on Cyber Security Experimentation and Test (CSET '13)*. Washington, D.C.: USENIX, 2013, pp. 1–8. [Online]. Available: https://www.usenix.org/conference/cset13/workshop-program/presentation/Everett

[11] Tencent Keen Security Lab, "Experimental Security Assessment of BMW Cars: A Summary Report," Tech. Rep., 2018. [Online]. Available: https://keenlab.tencent.com/en/

[12] C. Valasek and C. Miller, "Adventures in Automotive Networks and Control Units," *Technical White Paper*, p. 99, 2013. [Online]. Available: https://ioactive.com/resources/library/page/2/

[13] H. Lee, K. Choi, K. Chung, J. Kim, and K. Yim, "Fuzzing CAN packets into automobiles," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, vol. 2015-April, pp. 817–821, 2015.

[14] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and Vulnerable: A Story of Telematic Failures," *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, 2015.

[15] C. Miller and C. Valasek, "A Survey of Remote Automotive Attack Surfaces," Tech. Rep., 2014. [Online]. Available: https://sm.asisonline.org/ASIS SM Documents/remote attack surfaces.pdf%0Ahttp://illmatics.com/remote attack surfaces.pdf

[16] K.-T. Cho, Y. Kim, and K. G. Shin, "Who Killed My Parked Car?" 2018. [Online]. Available: http://arxiv.org/abs/1801.07741

[17] S. Fröschle and A. Stühring, "Analyzing the capabilities of the CAN Attacker," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10492 LNCS, pp. 464–482, 2017.

[18] M. Bozdal and I. Jennions, "A Survey on CAN Bus Protocol : Attacks , Challenges , and Potential Solutions," in *IEEE International Conference on Computing, Electronics & Communications Engineering (iCCECE '18)*, University of Essex, Southend, UK, 2018.

[19] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," *2010 6th International Conference on Information Assurance and Security, IAS 2010*, pp. 92–98, 2010.

[20] T. Hoppe, S. Kiltz, and J. Dittmann, "Applying Intrusion Detection to Automotive IT Early Insights and Remaining Challenges," *Journal of Information Assurance and Security*, vol. 4, no. May, pp. 226–235, 2009.

[21] A. Boudguiga, W. Klaudel, A. Boulanger, and P. Chiron, "A simple intrusion detection method for controller area network," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, may 2016, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7511098/

[22] K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *Proceedings of the 25th USENIX Security Symposium*, T. Holz and S. Savage, Eds. Austin, TX: USENIX Association, 2016, pp. 911–927. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho

[23] M. Gmiden, M. H. Gmiden, and H. Trabelsi, "An intrusion detection method for securing in-vehicle CAN bus," *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 176–180, 2016.

[24] A. Tomlinson, J. Bryans, S. Shaikh, and H. Kalutarage, "Detection of Automotive CAN Cyber-Attacks by Identifying Packet Timing Anomalies in Time Windows," in *Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2018*, 2018.

[25] Panasonic Corporation, "Panasonic Develops Automotive Intrusion Detection and Prevention Systems against Cyber Attacks Image of the Automotive Intrusion Detection and Prevention Systems Technical Features :," Tech. Rep., 2017. [Online]. Available: https://news.panasonic.com/global/press/data/2017/10/en171010-3/en171010-3.pdf

[26] Symantec Corporation, "Symantec Launches New IoT Solution to Help Carmakers Protect Against Zero Day Attacks," 2016. [Online]. Available: https://www.symantec.com/about/newsroom/press-releases/2016/symantec_0608_01

[27] I. Studnia, E. Alata, V. Nicomette, M. Ka, I. Studnia, E. Alata, V. Nicomette, M. Ka, and Y. L. A, "A language-based intrusion detection

approach for automotive embedded network," in *21st IEEE Pacific Rim International Symposium on Dependable Computing*, IEEE, Ed., Zhangjiajie, China, 2015.

[28] A. Patcha and J. M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[29] A. Tomlinson, J. Bryans, and S. A. Shaikh, "Towards Viable Intrusion Detection Methods For The Automotive Controller Area Network," in *2nd Computer Science in Cars Symposium - Future Challenges in Artificial Intelligence Security for Autonomous Vehicles*, 2018.

[30] D. S. Fowler, M. Cheah, S. A. Shaikh, and J. Bryans, "Towards a Testbed for Automotive Cybersecurity," *Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017*, pp. 540–541, 2017.

[31] T. Huang, J. Zhou, and A. Bytes, "ATG : An Attack Traffic Generation Tool for Security Testing of In-vehicle CAN Bus," *Proceedings of the 13th International Conference on Availability, Reliability and Security - ARES 2018*, pp. 1–6, 2018.

[32] B. Wymann and E. Espie, "TORCS, The Open Racing Car Simulator." [Online]. Available: http://torcs.sourceforge.net/

[33] U. Drolia, Z. Wang, Y. Pant, and R. Mangharam, "AutoPlug: An automotive test-bed for electronic controller unit testing and verification," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1187–1192, 2011.

[34] Vector Informatik GmbH, "CAPL Documentation." [Online]. Available: https://kb.vector.com/entry/48/

[35] K.-T. Cho and K. G. Shin, "Error Handling of In-vehicle Networks Makes Them Vulnerable," in *23rd ACM Conference on Computer and Communications Security*, 2016.