Exploiting Transport Protocol Vulnerabilities in SAE J1939 Networks

Rik Chatterjee
Colorado State University
rik.chatterjee@colostate.edu

Subhojeet Mukherjee
Colorado State University
subhojeet.mukherjee@colostate.edu

Jeremy Daily
Colorado State University
jeremy.daily@colostate.edu

***Abstract*—Modern vehicles are equipped with embedded computers that utilize standard protocols for internal communication. The SAE J1939 protocols running on top of the Controller Area Network (CAN) protocol is the primary choice of internal communication for embedded computers in medium and heavy-duty vehicles. This paper presents five different cases in which potential shortcomings of the SAE J1939 standards are exploited to launch attacks on in-vehicle computers that constitute SAE J1939 networks.**

In the first two of these scenarios, we validate the previously proposed attack hypothesis on more comprehensive testing setups. In the later three of these scenarios, we present newer attack vectors that can be executed on bench test setups and deployed SAE J1939 networks.

For the purpose of demonstration, we use bench-level test systems with real electronic control units connected to a CAN bus. Additional testing was conducted on a 2014 Kenworth T270 Class 6 truck under both stationary and driving conditions. Test results show how protocol attacks can target specific ECUs. These attacks should be considered by engineers and programmers implementing the J1939 protocol stack in their communications subsystem.

I. Introduction

Medium and heavy-duty (MHD) vehicles are a part of the US critical infrastructure, transporting goods, supporting emergency services, and so on. Modern MHD vehicles are electronified: most mechanical operations being controlled through embedded computers referred to as Electronic Control Units (ECUs). Within the vehicle, ECUs form networks to communicate mission critical information with each other on a bus topology. For MHD vehicles, the primary choice of communication specifications within these networks is the SAE J1939 standard. SAE J1939 documents [1] are organized in layers much like the ISO/OSI [2] standards for traditional IT networking. At its lowest layers, the SAE J1939 standards utilize the Controller Area Network (CAN) specifications [3] to facilitate the in-vehicle information exchange.

CAN is used widely in automotive networking and aspects of its (in)security has be thoroughly demonstrated. For example, it has been shown, with access to remote and local entry points (vulnerable ECUs) to the CAN network, one can launch attacks on the vehicle to control or disrupt its operations. MHD vehicles also expose similar entry points [4] and, aside from CAN specific attacks, it has been shown that attacks can also be launched on the SAE J1939 protocols. Even so, the number of demonstrated attacks is still limited: Burakova et al. [5] have demonstrated a couple of attacks on the application layer specification of the SAE J1939 standards, Murvay et al. [6] have focused on weaknesses at the network management layer, and Mukherjee et al. [7] have targeted specific protocols at the data-link layer of the specifications.

While the application and network management layers are critical to the cyber-physical operations of the vehicle, important message transportation specifications are made in the data-link layer standards. As such, in this work we demonstrate newer attacks at the data-link layer of the SAE J1939 specifications that broaden the horizon of cyber threats already created by Mukherjee et al. [7]. Moreover, we validate two attacks that Mukherjee et al. demonstrated to work on laboratory test benches. For our validations we use more comprehensive testing setups, as well as a 2014 Kenworth T270 truck; the goal being to demonstrate the applicability and impact of the attacks on different platforms.

The overarching goal of this paper is to enhance the threatscape for in-vehicle networking applications in MHD vehicles. To that end, the rest of the paper is organized as follows. In section II we present a brief overview of SAE J1939, as required to clearly comprehend the contributions made in this paper. In section III we briefly cover the related work in this area. In section IV, we present a description of the testing setup used in this work. In section V, we describe the attack experimentation carried out during the course of the work. Finally, in section VI, we finish with concluding remarks and a brief introspection of the future work.

II. Background on SAE J1939

In-vehicle communication in medium and heavy-duty vehicles is mostly guided by the SAE J1939 standards. SAE J1939 messages carry operational parameters like engine speed, vehicle speed, switch status, etc. These parameters are bundled into logical groups referred to as Parameter Groups (PG). Each PG is identified by a unique number called a Parameter Group Number (PGN), which is also embedded in the message. Information in the J1939 message is carried in a J1939

Symposium on Vehicle Security and Privacy (VehicleSec) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-88-6
https://dx.doi.org/10.14722/vehiclesec.2023.23053
www.ndss-symposium.org

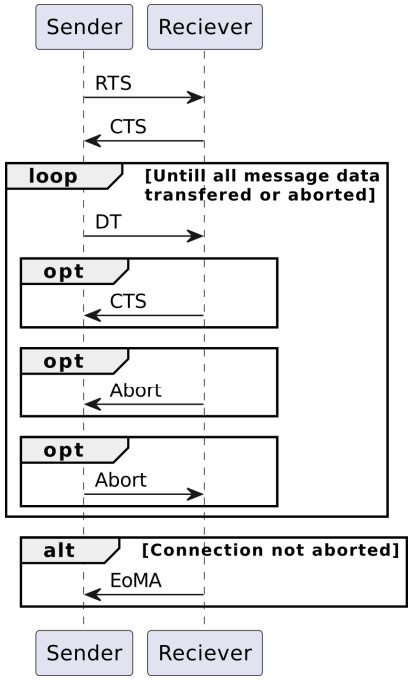


Fig. 1: Logical Point-to-Point or Destination Specific Message using the Transport Protocol

Protocol Data Unit (PDU). A J1939 PDU also bears a source address (SA) identifying the sender, a destination address (DA) identifying the receiver, the priority of the message, the PGN and up to 1785 bytes of data. The priority, PGN, SA, and DA are embedded into the identifier (ID) part of a CAN frame for PDUs with 8 or less bytes. For PDUs that have more than 8 bytes, a transport protocol (TP) is used, and the PGN is located in the last 3 bytes of the TP Connection Management (CM) message data. The TP message has a PGN of 560416 (`0x0EC00`). The CAN ID is used at the time of bus arbitration in case of transmission collisions. As such, the CAN frame identifier with the lowest numerical value wins the arbitration and occupies the bus.

CAN allows a maximum of 64 bits of data transfer in one frame. However, SAE J1939 messages are allowed to carry parameter data equivalent of 64 bits or greater. For message data that is greater than 64 bits, SAE J1939 specifies the transport protocol. Depending on the recipient of the information, SAE J1939 specifies two types of message transport protocols: destination-specific (i.e. from a specific source address to a specific destination address) and broadcast (i.e. from a specific source address to the entire network).

As shown in Fig. 1, in a destination specific transfer, the sending party attempts to open a connection by sending a Request to Send (RTS) message. An RTS message carries information about the total number of packets and the number of bytes to be sent. In response to the RTS, the receiver sends a Clear to Send (CTS) message. CTS messages enforce flow control by limiting the number of packets that can be sent after they are transmitted. CTS messages also aid in missing

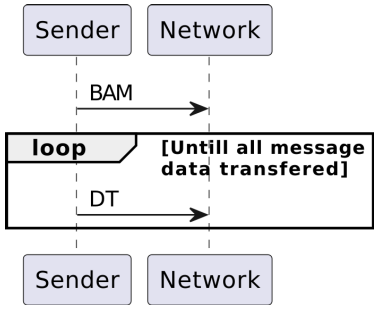


Fig. 2: Transport Protocol for a Broadcast Announce Message (BAM)

packet retransmission by indicating the next packet number to be sent. Upon receiving the CTS, the sender can send message data using data transfer (DT) messages until all data bytes are sent. The PGN of a DT message is 60160 (`0x0EB00`). The first byte of this message is reserved for a sequence number. Sequence numbers are used to reassemble the incoming data bytes. The remaining seven bytes of data in a DT message are used to transport the data bytes from the multipacket message being transferred. The flow of DT messages can be aborted by optionally sending an Abort message that includes information stating the reason for the abort. On successful completion of the transfer (i.e. if not aborted), an End of Message Acknowledgment (EoMA) is sent by the receiver to indicate closure of the connection.

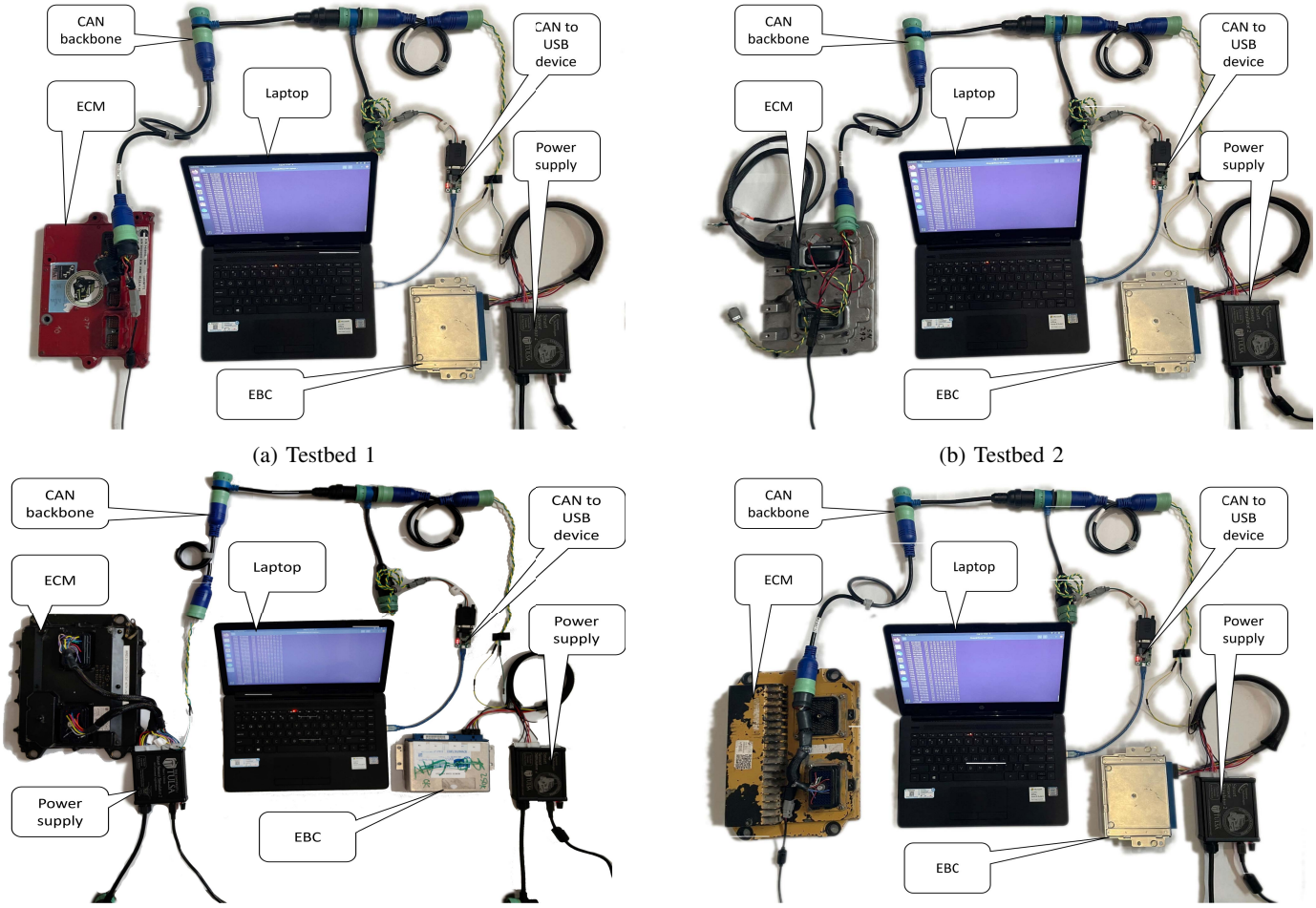
As shown in Fig. 2, the sending party initiates a broadcast message transport by sending a broadcast announcement message (BAM). Similar to the RTS message, the BAM message carries information about the total number of packets and the number of bytes to be sent. Soon after, the message data is transported in data transfer or DT packets that are formatted the same way as in the destination-specific data transfer. No EoMA is required at the end of a broadcast announce message.

Both destination-specific and broadcast message transport can be initiated by sending an SAE J1939 specified request message. This message has a PGN of 59904 (`0x0EA00`) and carries the requested PGN in the last 3 bytes of the data field. Requests can be directed to a specific device or to the global address of 255 (`0xFF`).

III. Related Work

Security aspects of the SAE J1939 Protocol have been largely overlooked. Prior research has demonstrated vulnerabilities at different layers of the SAE J1939 standards. Miller et al. [8] demonstrated that by rapidly injecting frames with CAN ID 0, network bandwidth available to the ECUs can be consumed completely leading to network overload. However, this affects the whole network and is easy to detect.

Burakova et al. [5] demonstrated vulnerabilities in the application layer of the SAE J1939 Protocol. Their research demonstrated that continuous control over the engine can be established by sending an SAE J1939 defined ad hoc message with PGN 0. Additionally, they have demonstrated that the



(a) Testbed 1

(b) Testbed 2

(c) Testbed 3

(d) Testbed 4

Fig. 3: Four bench test setups to include an engine control module (ECM) and an electronic brake controller (EBC).

truck's ability to use engine braking (retarder) can be disabled by commanding a zero percent torque request to the engine retarder in a message with PGN 0 at speeds below 30 mph. Furthermore, they have demonstrated the driver's input to the accelerator pedal can be effectively cut off by sending a command for minimal torque to the engine controller in a message with PGN 0. Murvay et al. [6] focused majorly on weaknesses at the network management layer. Their research demonstrated that an ECU can be rendered defunct on the network by repeatedly sending address claim messages with the target's address in the SA field and all zeros in the data field until all addresses in the ECU's pool are claimed. Additionally, they demonstrated that an attacker can send an abort message during an ongoing multi-packet data transfer and close the connection thereby causing a denial of service. This attack was confirmed by Campo et al. [9] and a potential network level mitigation was proposed. Mukherjee et al. [7] have targeted specific protocols at the data-link layer of the specifications. Their research showed that sending rapid request messages to an ECU can increase its processing load to a point whereby it cannot perform normal functions. Additionally, they showed that connections established with an ECU can be left open, thus denying legitimate connections from being used.

IV. Experimental Testing Setup

To ensure the uniformity of the experiment results, we conducted our experiments on multiple testbed configurations. All experiments were performed on different variations of a local testbed as well as on a research truck. In this section, we describe the different configurations of the local testbed and the research truck.

A. Bench Testbed

The local testbed setup on the bench consisted of four different configurations (Testbed 1, Testbed 2, Testbed 3, and Testbed 4) as shown in Fig. 3. Each of the four testbed configurations hosted a different Engine Control Module (ECM): Testbed 1 had an older generation Cummins 870 ECM, while Testbed 2 had a newer Cummins 2350 ECM. Testbed 3 had an older Caterpillar ADEM 3 ECM while Testbed 4 had a newer generation Caterpillar ADEM 4 ECM. Each ECM had two different controller applications (CA) running on them, one was a CA for Engine #1, and the other was a CA for the

Fig. 4: 2014 Kenworth T270 Research Truck

engine retarder, which is commonly known as the jake brake. Each testbed setup also included a Bendix electronic brake controller (EBC). A CAN backbone connected the ECM to the EBC and a Linux laptop was used to capture and transmit CAN messages through a CAN to USB device accessed by SocketCAN and the can-utils software. The laptop was also used as the attacker's pivot point and, in some cases, as nodes participating in the attack(s). One or more power supplies were also included.

B. Research Truck

Our research truck is a PACCAR PX-7-powered 2014 Kenworth T270. There is a Cummins CM2350 engine ECU (same as Testbed 2), a Bendix EC-60 brake ECU, and an Allison RDS-2000 transmission ECU on the truck. These ECUs communicate with each other on a 250 kbps CAN bus. A picture of this truck along with its dashboard is provided in Fig. 4.

V. Attack Experiments

In this section we describe our attack experiments. For each experiment we first provide the research hypothesis. We follow with a description of the hypothesis testing and a description of the results. Finally, we provide a small discussion of the possible mitigation techniques. All tests were conducted from a black-box perspective and access to the source code or run-time debug information was not possible.

A. Validation of the Request Overload Attack

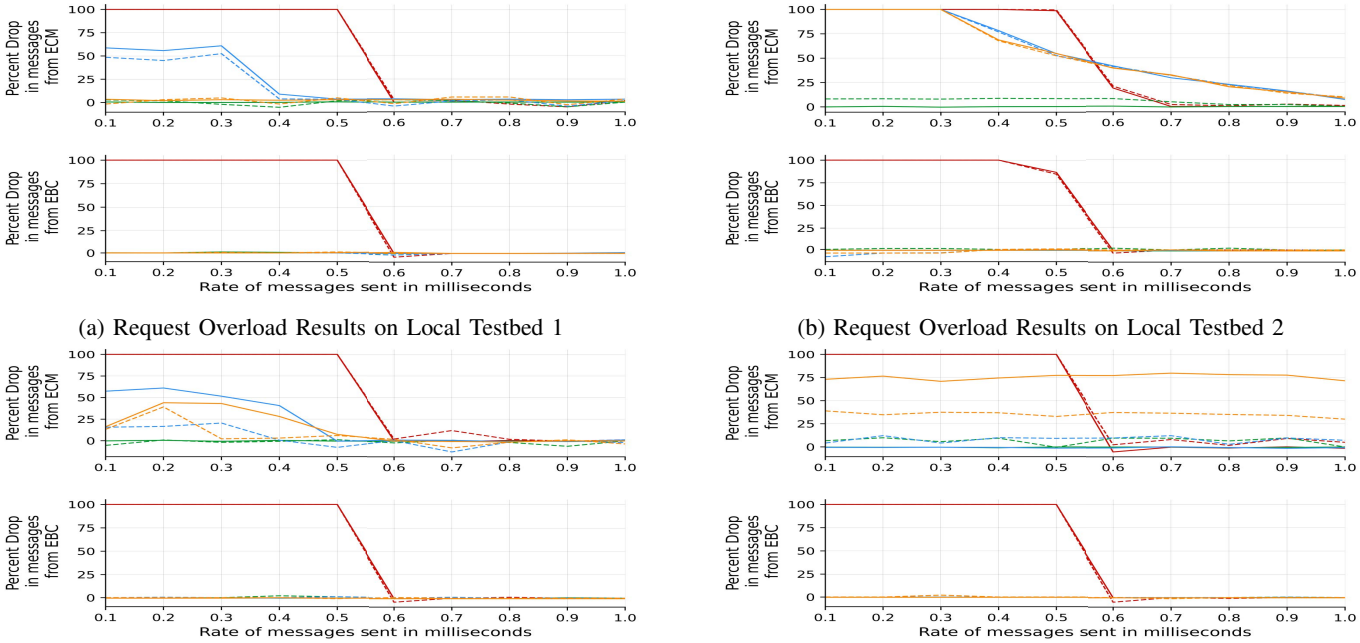
The first of the five attacks was called the "Request Overload" as it involves overloading a target ECU with request messages for attacker-chosen PGNs.

1) Hypothesis: The SAE J1939-21 document specifies that all directed requests to an ECU must be processed. An attack can thus be constructed to send a high volume of J1939 request messages, PGN 59904 (`0xEA00`), to the target ECU with the expectation that, in an attempt to serve the sent requests, the ECU fails to perform regular, more critical tasks like the transmission of periodic messages.

2) Testing: Mukherjee et al. [7] tested the attack by sending a high volume of requests to a target engine control module (ECM). We did the same, albeit across multiple testbed configurations. Additionally, we also addressed a few shortcomings of the previous experimentation process. Firstly, it was not investigated if the drop in count was because of a request overload or messages losing arbitration to higher priority request messages. Secondly, it was not investigated if the rate of injection of the request messages had any relation with the success of the attack. Thirdly, not all PGNs are utilized in a controller application. Therefore, it was not investigated if requesting a Parameter Group Number (PGN) that is not used had any effect on the network traffic.

While conducting our experiments on the local testbed, we addressed the aforementioned shortcomings. We sent four different types of messages on the CAN network with varying intervals of transmission: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1 milliseconds. The first type of message had a CAN ID of `0x00000000`. This is the lowest-valued CAN ID and is expected to flood the entire bus by winning transmission arbitration. If the CAN bus is saturated with these high priority messages, no other message can gain access to the network. The second type of message had the lowest SAE J1939 priority (7), but 0 for the PGN, DA, and SA, which produced an extended CAN ID of `0x1C000000`. The third type of message had the same SAE J1939 priority (7) but was the request PGN 59904 from the engine controller to itself. The equivalent CAN ID was `0x1CEA0000`. The engine controller responded to this request. Also, this request was for the Component Identifier PG (PGN 65259), which was present with all the ECMs used in the experiment. This is a valid request and is often part of a diagnostics routine. Finally, we sent a request for PGN `0xFFFF`, which was not present in any engine controller, so this was an invalid request. The purpose of sending the invalid request was to address the final shortcoming in [7]. The first three types of messages were sent to address the first shortcoming. Messages with CAN ID `0x00000000` were sent to demonstrate network overload and the removal of all normal traffic from the bus. Messages with CAN ID `0x1C000000` were sent to demonstrate that lowering the SAE J1939 priority from 0 to 7 (causing a change in ID from `0x00000000` to `0x1C000000` had no effect on normal traffic. As such, our goal was to observe if messages with CAN ID `0x1CEA0000` (SAE J1939 priority = 7) had any effect on ECM traffic only. If so, we could conclude request overload had an effect on the traffic emanating from the target ECM and not the brake, thereby demonstrating two things: 1) request overload is indeed a targeted attack on broadcast CAN and 2) it works even at the lowest SAE J1939 priority (7).

We conducted five experiments for each type of message transmitted at each interval mentioned in the previous paragraph. The average of the results from the five experiments is plotted. Note that, due to bus contention, CAN controller delay, frame collision, etc., the transmission interval on the CAN bus may not be equal to the exact intervals set for the experiments through the can-utils software.



(a) Request Overload Results on Local Testbed 1

(b) Request Overload Results on Local Testbed 2

(c) Request Overload Results on Local Testbed 3

(d) Request Overload Results on Local Testbed 4

Line color significance:
Red: On flooding with messages of ID 00000000_{16}
Blue: On overloading with valid request messages
Orange: On overload with invalid request messages
Green: On flooding with messages of ID $1C000000_{16}$
Line shape significance:
Solid: High priority ([0,3]) messages
Dashed: Low priority ([4,7]) messages

Fig. 5: Request overload experimental results on different configurations of the local testbed

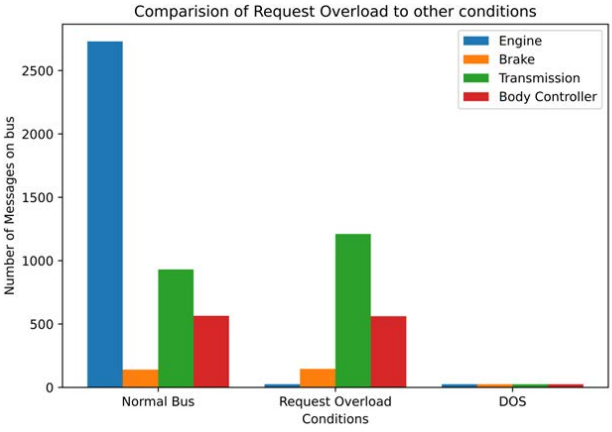


Fig. 6: Comparison of the request overload attack on the Research Truck in other situations

3) Results and Observations: As can be seen from Fig. 5, for all the test cases a drop in message count was observed from all sources when the network was flooded with messages with CAN ID `0x00000000`. Then again, in all the cases of

Fig. 7: Request overload attack on the Research Truck

flooding with messages of CAN ID `0x1C000000`, almost no normal traffic was removed. However, when the request overload attack was conducted, a certain percentage of normal messages transmitted by the ECM was removed. Albeit, the traffic volume from the brake controller remains constant during request overloads, thus indicating that it only affected the performance of the target. On Testbed 1, about 50 percent of both high and low-priority traffic were removed by valid request overloads up to 0.3 milliseconds, but invalid request overloads did not have any effect. On Testbed 2, all traffic transmitted by the ECM was removed by both valid and

invalid request overloads up to 0.3 milliseconds. On Testbed 3, greater than 20 percent of both high and low-priority traffic were removed by valid and invalid request overloads up to 0.2 milliseconds. On Testbed 4, about 75 percent of high and 25 percent of low-priority traffic was removed by invalid request overloads up to a millisecond but valid requests had no effect. We noticed that it handled all valid requests promptly, initiating a multi-packet transfer in all cases. For invalid requests though, it transmitted a series of negative acknowledgments. Overall, we noticed that, in general, a request overload attack launched at 0.3 milliseconds or below, had an effect on the target ECM, even when launched with the lowest priority.

We conducted this attack on the Kenworth T270 research truck. This truck had the same ECM as used in the local Testbed 2. As such, when a request overload attack was conducted at 0.3 milliseconds, all transmissions from the ECM stopped. The physical effect could be seen in Fig. 7 where the truck's dashboard displays erroneous information. Furthermore, the transmission did not shift gears and the engine speed remained high while moving forward. We also compared the effects of the request overload attack to normal bus conditions and a network flood attack(where the J1939 bus is flooded with high priority messages with CAN ID 00000000_{16}). As shown in Fig. 6, we can see that a network flood attack referred to as a denial of service (DOS) removes traffic from all ECUs at 0.3 milliseconds while the request overload attack on the engine controller removes all traffic only from the engine while other ECUs continue to broadcast data. This validates the request overload attack as a targeted denial of service attack.

4) Possible Mitigation: A potential solution to defend against this kind of attack is for ECUs to ignore request messages if they are received faster than a certain rate.

B. Connection Exhaustion Attack

The second of the five attacks was named "Connection Exhaustion" as it exhausts the ability of the ECU to establish legitimate connections for multi-packet data transfer.

1) Hypothesis: According to the J1939-21 standards, there can only be one established connection for multi-packet transfer between a source ECU and a destination ECU at a time. It also states that after data has been transmitted a connection can be kept open for a maximum of 1250 milliseconds by not sending the end-of-message acknowledgment. In addition, a CTS message can be sent to request one or more packets that may have been sent already, but not received by the destination ECU. Using these three specifications, an attack can be crafted to deny legitimate connection attempts to an ECU by creating multiple spoofed connections and keeping them open periodically (typically for less than a second) sending a CTS message but not the end of message acknowledgment.

2) Testing: The attack was carried out on the local testbed as well as on the research truck. A valid connection was established with the ECMs by sending requests and CTS packets from a spoofed source address. After the data packets are transferred by the ECM, the connection is maintained by sending a periodic CTS message every second. After a while, an honest connection is attempted to be established from the same source address.

3) Results and Observations: Fig. 8 shows the "Connection Exhaustion" attack was demonstrated on three of four of our local testbench setups. The attack did not work on testbench 4. From Figs. 8a, 8b and 8c, we can observe a malicious CTS message was sent every second to keep the connection alive. A legitimate node on the network could not receive an RTS message for valid requests it sent. Thus, a legitimate connection could not be established as long the connection was maliciously occupied.

Even though the attack works on the Kenworth T270 truck, it did not have a physical impact. Albeit, a quick investigation of the SAE J1939 digital annex reveals that transport sessions are critical for diagnostic and proprietary communication over SAE J1939. An example of hampering diagnostics is shown in Fig. 9 where the manufacturer diagnostics software tool failed to connect to the ECM. This can also be detrimental to the vehicle if data obtained from the session is used for control purposes. For example, PGN 65251 carries engine configuration information that may be required by more than one legitimate ECU. If this information is not received, those ECUs may malfunction.

4) Possible Mitigation: The apparent solution to this kind of attack is for ECUs to terminate open connections after a certain amount of time if connected ECUs are not exchanging data packets.

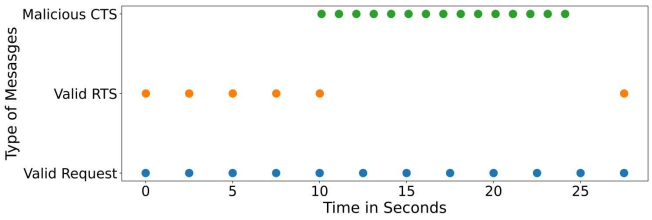
C. BAM Block Attack

The third of the five attacks is coined the name "BAM Block" as it blocks broadcast announcement messages (BAMs) and subsequent data packets for multi-packet data transfer from a targeted ECU.

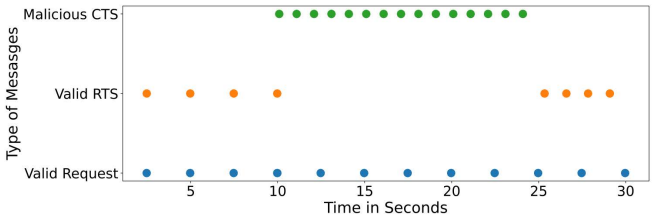
1) Hypothesis: ECUs periodically transmit BAMs which are multi-packet data frames to relay information globally to other ECUs on the network. The SAE J1939-21 standard suggests that an ECU must respond to destination-specific requests. Given this, an attack can be constructed whereby an attacker sends destination-specific requests for PGNs that an ECU broadcasts globally as BAMs with the expectation that this might force the ECU to respond to such a request and, in turn, the global broadcast communication halts denying information to all ECUs on the network.

2) Testing: The attack was tested on the testbed setup. We wrote a bash script that sends a destination-specific request for a PGN the target ECM was broadcasting globally and a CTS message after successfully receiving an RTS message from the target. This process was iterated over a loop for the duration of the attack.

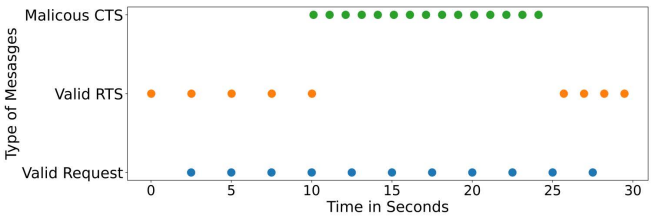
3) Results and Observations: We noticed that this attack was only successful on Testbed 3 and did not have any impact on the other testbeds or the research truck. On execution of our script on Testbed 3, we observed that the Caterpillar ADEM3 ECU responded to the malicious destination-specific



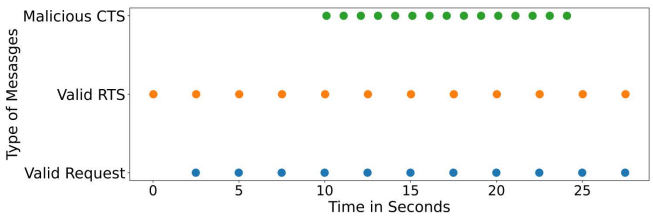
(a) Connection Exhaustion Results on Local Testbed 1



(b) Connection Exhaustion Results on Local Testbed 2



(c) Connection Exhaustion Results on Local Testbed 3



(d) Connection Exhaustion Results on Local Testbed 4

Fig. 8: Connection Exhaustion Experiment Results on Different Configurations of the Local Testbed

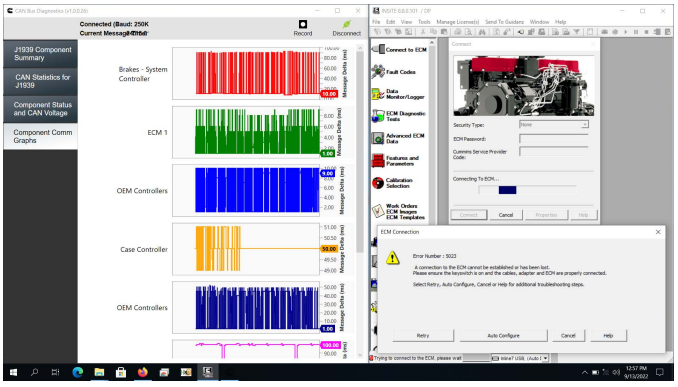


Fig. 9: Connection Exhaustion Attack on Research Truck

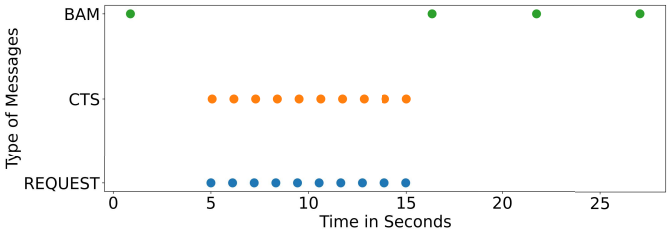


Fig. 10: BAM Block Attack Demonstration

requests by redirecting data packets to the specific destination we requested the data. The ECU did not transmit any BAMs or multi-packet data globally as long as the malicious CTS were being sent by our attack script. This is shown in Fig. 10.

4) Possible Mitigation: The apparent mitigation against this attack is to keep transmitting BAMs for specific PGNs even if destination-specific requests for the same are received.

D. Malicious Clear to Send (CTS) Attack

The fourth of the five attacks is coined the name "Malicious CTS" as it involved sending a malicious CTS packet to hinder

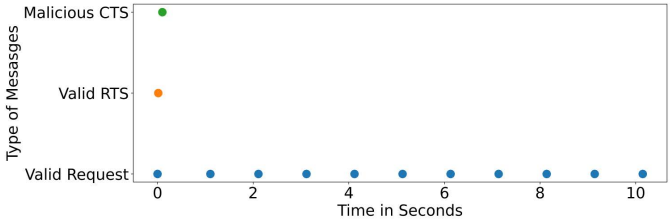


Fig. 11: Malicious CTS Attack Demonstration

J1939 transport protocol communication from a targeted ECU.

1) Hypothesis: The SAE J1939-21 document specifies that an RTS message communicates information about the total number of data packets that an ECU can send over multi-packet data transfer for a requested PGN. Additionally, it specifies that a CTS message should contain information indicating the packet number of the next data packet to be sent. If the value in the CTS message exceeds the total number of data packets in the RTS message for a multi-packet data transfer, it is possible that an ECU may not to be programmed to handle this erroneous information. Given this, an attack can be constructed to send a malicious CTS message with an erroneous value of the next packet to be sent that exceeds the total number of packets that can be sent indicated by the RTS message. This may cause the targeted ECU to enter an unknown state and thus hinder normal operations.

2) Testing: The attack was tested on the testbed setup. We wrote a shell script that sends a destination-specific request for a valid PGN to the target ECU and waits for an RTS. On receiving the RTS message a malicious CTS message is constructed with the value of the next packet to be sent in the CTS message exceeding the total number of packets indicated by the RTS message. This crafted CTS is then sent to the targeted ECU.

3) Results and Observations: We noticed that this attack was only successful on Testbed 3 and did not have any

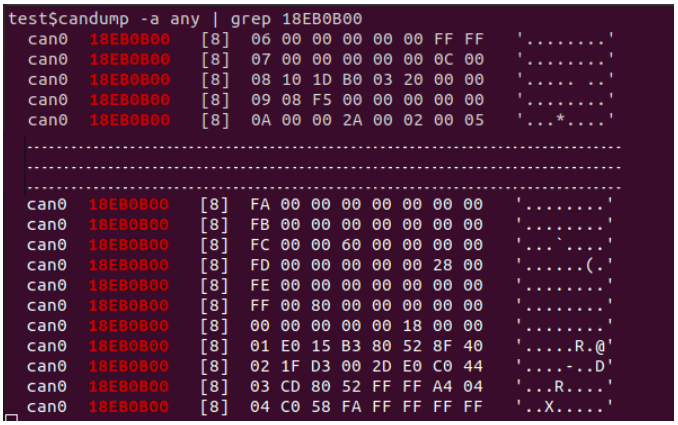


Fig. 12: Memory Leak Attack Demonstration

impact on the other Testbeds or the research truck. On execution of our script on Testbed 3, we observed that the Caterpillar ADEM3 ECM stops responding to any further request messages as shown in Fig. 11. Additionally, all multi-packet communication from the ECM ceases until the ECM is rebooted.

4) Possible Mitigation: The apparent mitigation against this attack is to ensure invalid CTS messages as used in the attack are ignored.

E. Memory Leak Attack

The final of the five attacks is coined as the "Memory Leak" attack as it leads to receiving data from an ECU that was not intended to be sent.

1) Hypothesis: The SAE J1939-21 document specifies that a CTS message should contain information about the number of packets that can be sent over multi-packet data transfer. In theory, this value should not exceed the maximum number of packets indicated by the RTS message. As such, an attack can be constructed by sending a crafted CTS message with the value of the number of packets that can be sent larger value indicated by the RTS message with the expectation this may cause the targeted ECU to leak data packets it never intended to send.

2) Testing: Similar to the Malicious CTS attack, we wrote a script that sends a destination-specific request to the target ECU and waits for a valid RTS. On receiving an RTS from the ECU, a CTS message is crafted with the number of packets to be sent set to a value much larger than indicated by the previous RTS message. This crafted CTS message was then sent to the targeted ECU.

3) Results and Observations: We noticed that this attack was only successful on Testbed 3. As shown in Fig. 12, we look a dump of the J1939 traffic using the 'candump' feature of can-utils. The ASCII representation of the data was also obtained using the '-a' delimiter. The traffic was also filtered to show only multi-packet data leaked from the ECU. This ECU was supposed to send on 6 packets for this specific transfer. Nevertheless, it leaked 255 data packets as requested in the CTS message. The attack did not succeed on the Kenworth T270 research truck carrying a different ECU. The contents of the leaked data has not been investigated as yet; it is reserved for future work.

4) Possible Mitigation: The apparent mitigation against this attack is to ensure invalid CTS messages, as used in the attack, are ignored. This may be implemented as part of the ECU firmware or as a centralized network security solution.

VI. Conclusion and Future Work

This paper presents five different scenarios where ECUs on SAE J1939 networks are subjected to different types of attacks. First, two of the five scenarios demonstrate validations of attacks discovered in prior literature. The validation incorporates a more comprehensive testing setup. The latter three scenarios demonstrate new attack cases. Each of these attacks exploits specifications from the SAE J1939 protocol standards.

At its core, this paper helps in enhancing the existing threatscape of vehicle security for medium and heavy-duty vehicles. Even so, a large part of the networking specifications still remains unexplored for security loopholes. In the future, we aim to investigate these opportunities, the focus still being medium and heavy vehicles. Additionally, we want to explore defense mechanisms to prevent these attacks using a centralized network-based solution.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Naval Information Warfare Center Pacific (NIWC Pacific) under Contract No. N66001-20-C-4021. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or NIWC Pacific.

References

[1] Society of Automotive Engineers, "SAE J1939 Standards Collection." [Online]. Available: https://www.sae.org/standardsdev/groundvehicle/j1939a.htm

[2] International Organization for Standardization, "Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model," Standard ISO/IEC 7498-1:1994. [Online]. Available: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/02/02/20269.html

[3] Robert Bosch GmbH, "CAN Specification," Robert Bosch GmbH, Standard 2.0, 1991.

[4] S. Stachowski, R. Bielawski, and André Weimerskirch, "Cybersecurity Research Considerations for Heavy Vehicles," University of Michigan, Ann Arbor, Transportation Research Institute, Tech. Rep., 2019.

[5] Y. Burakova, B. Hass, L. Millar, and A. Weimerskirch, "Truck Hacking: An Experimental Analysis of the SAE J1939 Standard," in *Proceedings of the 10th USENIX Conference on Offensive Technologies*. Austin, TX, USA: USENIX Association, 2016, pp. 211–220.

[6] P. Murvay and B. Groza, "Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4325–4339, 2018.

[7] S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, "Practical DoS Attacks on Embedded Networks in Commercial Vehicles," in *International Conference on Information Systems Security*. Jaipur, Rajasthan, India: Springer, 2016, pp. 23–42.

[8] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," in *Blackhat USA*. Las Vegas, NV, USA: Blackhat Press, 2015.

[9] M. T. Campo, S. Mukherjee, and J. Daily, "Real-Time Network Defense of SAE J1939 Address Claim Attacks," *SAE International Journal of Commercial Vehicles*, vol. 14, no. 3, Aug. 2021.