TRANSNAV
http://www.transnav.eu
the International Journal
on Marine Navigation
and Safety of Sea Transportation
Volume 15
Number 3
September 2021
DOI: 10.12716/1001.15.03.05

The CAN Bus in the Maritime Environment – Technical Overview and Cybersecurity Vulnerabilities

Gary C. Kessler
Fathom5, Ormond Beach, Florida, USA

ABSTRACT: The Controller Area Network (CAN) bus standard was developed in the 1980s and is in widespread use in automobile, vehicular, aviation, and other networks. The CAN bus was introduced in the maritime environment with the adoption of the National Marine Electronics Association (NMEA) 2000 standard in the late-1990s. Many papers have been written about the CAN bus protocols and security vulnerabilities but there is sparse literature about use of the CAN bus in the maritime environment. Part I of this paper is a technical overview, describing CAN bus standards and operation, with particular attention to its use with the NMEA 2000 maritime communications standard. Part II of this paper describes security vulnerabilities in terms of loss of confidentiality, integrity, or availability of information (such as eavesdropping, denial-of-service, and spoofing), and mitigations specific to the maritime environment.

1 INTRODUCTION

The Controller Area Network (CAN) bus was developed in the early 1980s and has been in wide use for intra-vehicular communication since its initial introduction 30 years ago, primarily for automobiles and other land vehicles. The CAN bus has been used in maritime communication standards for the last 20 years, yet there are few papers describing the maritime applications of the CAN bus or the cybersecurity vulnerabilities of the CAN bus in the maritime environment. Part I of this paper is a technical tutorial describing the CAN bus and how it is employed in maritime communications. Part II is less technical, and discusses CAN bus cybersecurity vulnerabilities and mitigations specific to maritime applications. The paper ends with a summary and some conclusions.

2 CAN BUS TECHNICAL DESCRIPTION

In this part of the paper, Section 1 provides an overview of the CAN bus, including its history, origins, and generic use in the maritime industry. Section 2 describes the CAN bus standards, operation, and frame format, followed in Section 3 by a high-level overview of maritime communications employing the CAN bus. Section 4 provides a detailed description of the coding of a CAN bus transmission.

2.1 *CAN bus overview*

The CAN bus standard is a message-based communications protocol developed in the early-1980s for automobile device communication. Unlike the point-to-point and multidrop serial protocols of the day, the CAN bus is a broadcast bus where any device can transmit when it is ready and does not have to wait to be polled by some master station; the

CAN bus standard refers to this as a multimaster protocol because all devices are, essentially, peers. In terms of the Open Systems Interconnection (OSI) reference model, the CAN bus standard provides physical and data link layer (layer 1 and 2) services. Any suitable higher layer protocol can be designed or adapted to run over the CAN bus [7, 10–12].

The CAN bus was originally developed by Robert Bosch Gmbh just as microprocessors were being introduced into the design of automobiles. Adopted by the Society of Automotive Engineers (SAE) in 1986 as the Automotive Serial Controller Area Network, the 1992 Mercedes-Benz W140 was the first production vehicle to employ CAN bus. It is now nearly universally used in automobiles for interconnecting the vehicle's computer controllers with the transmission, airbags, anti-lock braking system, power steering, engine control, traction management, navigation system, and entertainment devices [7, 10, 12].

Use of the CAN bus has grown considerably over the last three decades. Within the transportation sector alone, CAN bus communications are used in [3, 10]:

- Buses, tractor trailers, and agricultural vehicles to interconnect specialized systems and devices.
- Aircraft to interconnect flight state sensors and analyzers, navigation systems, aircraft engine control systems, flight surfaces, fuel systems, and more.
- Railroad equipment such as streetcars, trams, subways (undergrounds), light rail, and long-distance trains.
- Maritime vessels to interconnect such equipment as the wind speed/wind direction/air temperature sensors, Automatic Identification System (AIS), Global Navigation Satellite System (GNSS), gyroscope, compass, navigation display, a large variety of ship status sensors, voyage data recorder (VDR), and ship state dashboard display (Figure 1).

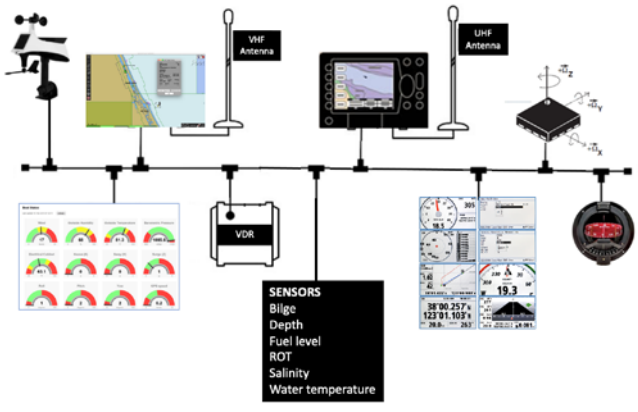


Figure 1. CAN bus interconnection of (clockwise, from upper left) the weather station, AIS and GNSS receivers, gyroscope, compass, navigation display, myriad sensors, VDR, and ship state dashboard display.

Cars represent the very earliest of what we now refer to as industrial control systems (ICS), where computers, sensors, and actuators are interconnected to manage and operate industrial and other mechanical systems. ICS are used on land as well as in the air and on the water. The CAN bus is currently employed in environments as varied as audio/video systems and smart building controls to telescopes and elevators/escalators [3].

2.2 *CAN bus standards and operation*

This section will describe the CAN bus standards, protocol layers, and operation in order to establish a basis for the discussion of security vulnerabilities.

2.2.1 *Standards*

The SAE J1939 family of standards describes both the physical and data link layers of the CAN bus, as well as higher layer protocols for use in a variety of automotive applications [28]. The generic global CAN bus specification is contained in four standards from the International Organization for Standardization (ISO):

- ISO 11519-1 describes a low-speed (125 kbps) serial interface [18].
- ISO 11898-1 describes the data link layer frame format and physical layer signaling [15].
- ISO 11898-2 describes the high-speed (1 Mbps or 5 Mbps) interface [16].
- ISO 11898-3 describes a low-speed, fault-tolerant interface [17].

The CAN bus multimaster architecture supports a multihost, broadcast environment where any node can transmit whenever it has data to send, although rules must be in place to ensure that end devices do not transmit over one another. Higher-layer protocols define the messages that are exchanged via the CAN bus.

2.2.2 *Physical Layer*

The CAN bus physical layer is a two-wire bus (Figure 2). Each node attaches to both wires, called CAN high (CAN_H) and CAN low (CAN_L). By default, the bus signals are driven to a dominant state (0) with $CAN_H > CAN_L$; a signal is passively pulled by resistors to a recessive state (1) with $CAN_H \leqslant CAN_L$. Thus, if more than one node transmits at the same time, the effect is a logical AND (i.e., if any node transmits a 0, then 0 will be the resultant signal on the line; only if all transmitting nodes send a 1 will the resultant signal be a 1). In order to maintain clock synchronization, the CAN bus employs a bit-stuffing mechanism where five consecutive bits of the same value are followed by a single bit of the opposite value [6, 7, 11, 12].

The CAN bus has a relatively simple physical design (Figure 3). The main backbone comprises a series of point-to-point, twisted-pair cables with a CAN bus connector at each end. The entire CAN bus is terminated at each end with a terminating resistor. Devices are attached to the backbone by placing a T-connector on the bus; the backbone, then, is really a sequence of cables that string together the T-connectors. The T-connector also attaches to a two-wire pigtail that connects to the CAN bus interface on compatible devices.

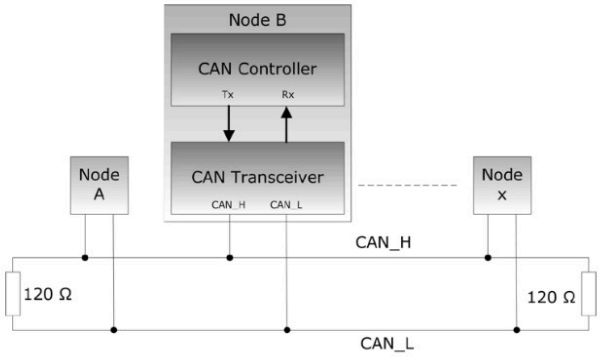


Figure 2. CAN bus nodes connected to the two-wire bus. Note that the bus is terminated with a resister. (Source: https://cdn10.bigcommerce.com/s-7f2gq5h/product_images/uploaded_images/can-bus-network-with-transceiver.jpg)

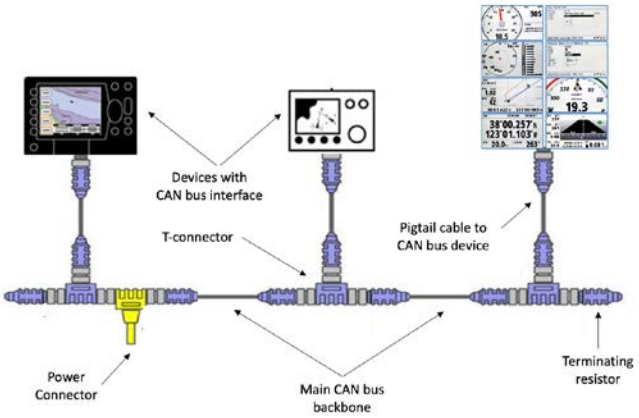


Figure 3. CAN bus architecture.

The CAN bus standard supports speeds up to 1 million bits per second (Mbps) or distances up to 3,300 feet (1,000 meters). As shown in Table 1, the maximum speed decreases as the end-to-end distance – and end-to-end signal delay – increases [7, 11, 12].

Table 1. Cable length and maximum speed trade-off.

Cable Length	Max. Speed
1,000 m (3,300 ft)	50 kbps
500 m (1,660 ft)	125 kbps
200 m (650 ft)	250 kbps
100 m (330 ft)	500 kbps
40 m (130 ft)	1 Mbps

2.2.3 *CAN Bus Frame Types*

The bits on the CAN bus are organized into a protocol data unit called a frame. There are two frame formats; CAN 2.0A, called the base frame format, uses an 11-bit Identifier field while CAN2.0B, called the extended frame format, uses a 29-bit device identifier, split between an 11-bit Identifier field and an 18-bit Identifier Extension field [7, 11].

There are four primary types of frames used by the CAN bus standard. The most common frame type is a data frame which contains up to 8 bytes of data transmitted by a device. A remote frame is used by one device to request data from another; it contains no data. An error frame is transmitted by any device detecting any sort of error in a transmission; this frame causes all other devices to send an error frame after which the original transmitter will automatically resend its message. Finally, an overload frame is sent by a device that is in an overloaded or busy state, and is used to inject a delay between transmissions.

Figure 4. CAN 2.0A frame with 11-bit identification field (top) and CAN 2.0B frame with 29-bit identification field (bottom). (Source: https://cdn.sparkfun.com/assets/learn_tutorials/5/4/1/CAN_PacketStructureFrames_1.png)

Figure 4 shows the CAN 2.0 frame format for both basic and extended frames. The fields are:

- Start-of-Frame (SOF) bit (0)
- Base Identifier (11 bits in length)
- Substitute Remote Request (SRR) bit (1) [CAN 2.0B only]
- Identifier Extension (IDE) bit (0 in CAN2.0A, 1 in CAN2.0B))
- Extended Identifier (18 bits in length) [CAN 2.0B only]
- Remote Transmission Request (RTR) bit (0 = Data, 1 = Remote Request)
- Reserved (RES) bits (0)
- Data Length Code (4 bits in length, value 0-8)
- Data (0-8 bytes in length)
- Cyclic Redundancy Check (CRC) field (16 bits in length)
  - CRC-15 value (15 bits in length)
  - CRC Delimiter bit (1)
- Acknowledgement (ACK) field (2 bits in length)
  - ACK Slot bit (transmitter sends 1, any receiver can send 0)
  - ACK Delimiter bit (1)
- End-of-Frame (EOF) field (1111111)
- Inter-Frame Space (IFS; at least seven 1s)

The CAN bus frame structure is not terribly efficient in terms of data transfer; an extended frame with eight bytes of data is at least 135 bits in length, which means that, at most, 47% of the transmission is data. For the original designers of automotive networks, this was not a high price to pay in a geographically small network with only a few dozen devices, but it does not necessarily scale well to physically larger networks with more devices.

2.2.4 *Arbitration*

The CAN specification employs a broadcast bus so that all devices hear all transmissions. There is no bus controller nor is there a primary device to which all others must communicate; it is essentially a peer-to-peer network. When more than one station is ready to transmit, they resolve the conflict through a process known as arbitration [7, 11]. This process is somewhat similar to the Carrier Sense, Multiple Access with Collision Detection (CSMA/CD) scheme used in IEEE 802.3/Ethernet networks.

When a CAN device is ready to transmit, it listens to the bus to see if the bus is already in use. If another device is transmitting, the station waits until the line becomes idle, represented by a sequence of more than seven consecutive 1 bits (the Inter-Frame space), before it starts transmitting. If more than one device

becomes ready while another station is transmitting, all will start to transmit at the same time after seeing the line go idle.

A device continues to monitor the bus while it transmits. As described in Section 3.2 above, if more than one station transmits at one time, the resultant bit signal on the line is the equivalent of a logical AND of the two inputs. As soon as a station "sees" a bit that is different from one that it sent, it will stop transmitting.

Consider this simple example of CAN bus arbitration. Suppose there two devices that are ready to transmit, where Device 1 has the 11-bit address 0x5C3 (binary 10111000011) and Device 2 has the address 0x598 (binary 10110011000). Per the description of the CAN bus frame, both stations start by transmitting a Start-of-Frame bit, or a 0, so both will "see" a 0 on the line.

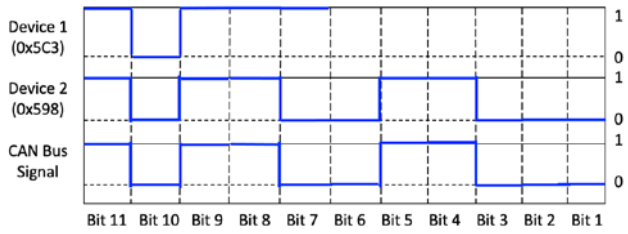


Figure 5. CAN bus arbitration. Devices 1 and 2 start to transmit at the same time; Device 2, with the lower address, "wins" the arbitration.

Next, each station starts to send their 11-bit identifier (Figure 5). The first four bits of both addresses are "1011" so each station sees, in turn, the same bit on the line as the one that they are sending. When they transmit the fifth bit (Bit 7), Device 1 sends a 1 and Device 2 sends a 0; the resultant value on the line is a 0. At that point, Device 1 stops transmitting but Device 2, seeing the same value as it transmitted, continues. This same arbitration process works regardless of how many stations might wish to transit at once. The station with the lowest address will always win the arbitration and, in fact, is never aware that a collision occurred.

2.2.5 *Higher-Layer Protocols*

The CAN bus standard defines a physical layer and frame format for the exchange of data. The actual format of the data is specified in application- and device-specific higher-layer protocols. The format for higher-layer messages are described in parameter group number (PGN) specifications. A PGN defines the function of a message, the format of the CAN bus Arbitration field, and the format of the contents in the Data field.

2.3 *Maritime communications over the can bus*

This section will briefly describe the primary maritime communication standard that employs the CAN bus.

2.3.1 *Maritime Communication Standards*

Just as there are many networks onboard a ship, there are many communication protocols and standards. National Marine Electronics Association (NMEA) standards are the primary specifications employed on boats and ships of all sizes to interconnect instrumentation, from a Global Positioning System (GPS) receiver, navigation display, and engine monitor to the Electronic Chart Display and Information System (ECDIS), myriad sensors, and a variety of controllers (Figure 6).

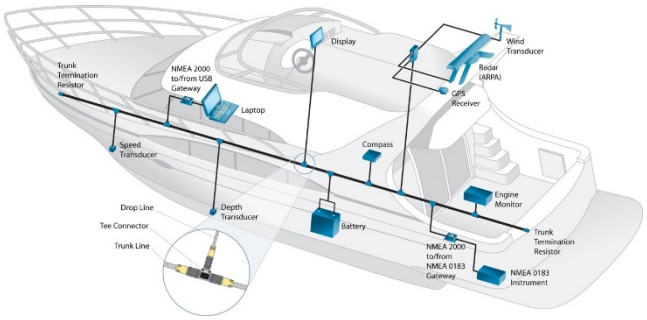


Figure 6. Maritime instrumentation and the NMEA 2000 bus on a boat. (Source: https://upload.wikimedia.org/wikipedia/commons/b/bf/NMEA2000_Modified_motor_yacht.jpg)

There are three dominant NMEA standards for instrumentation communication within a vessel:

- Introduced in 1983, NMEA 0183 operates on Electronic Industries Alliance (EIA)-232/422 serial lines at 4,800 or 38,400 bits per second (bps). This standard is the basis of International Electrotechnical Commission (IEC) standard 61162-1, and is employed in International Telecommunication Union, Radiocommunication Sector (ITU-R) Recommendation M.1371-5 for over-the-air transmissions at a rate of 9,600 bps. Version V4.11 of NMEA 0183 was released in 2018 [22].
- NMEA 2000 was released in 2001 and introduced a streamlined message format and PGN message set to support a large variety of maritime devices. This standard operates at speeds up to 250 kbps over the CAN bus. NMEA 2000 was adopted as IEC 61162-3; Edition 3.101 was released in 2016 [23].
- OneNet is the newest member of the NMEA family of standards. Released in 2020, OneNet employs a superset of the NMEA 2000 message set. OneNet devices exchange data using Internet Protocol version 6 (IPv6) packets running over an Ethernet local area network at speeds up to 10 Gbps and employs IP Security (IPsec) for secure inter-device communication [24].

Since NMEA 2000 is the only one of these standards to operate over the CAN bus, the remainder of this discussion will focus on the interrelationship of these two standards.

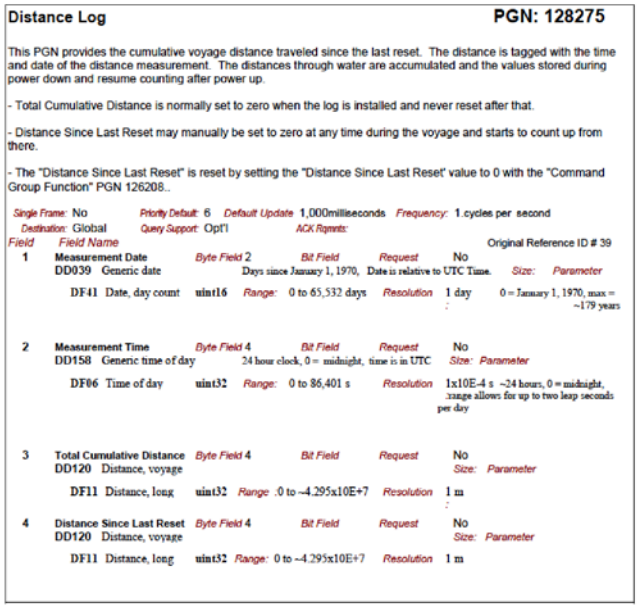
2.3.2 *NMEA 2000*

As mentioned above, the NMEA 2000 (aka N2K) standard is designed for communication between marine electronics on a vessel and employs the CAN bus as the physical layer; this protocol is not used for over-the-air transmissions [2]. Prior maritime communication standards used point-to-point links

between each device and a central controller, whereas the CAN bus employs a single communications bus with which to interconnect multiple devices. Use of the CAN bus yields a much simpler wiring scheme that requires less overall cable, resulting in a noticeable reduction in the cost of the wiring, installation, and maintenance, as well as a lower total weight of the system [6, 10, 12].

N2K messages are denoted by their name and PGN [6, 10, 23]. Each PGN definition describes details about the message, including (Figure 7):

- The binary format of the data
- Message priority (0-7)
- Whether the message fits in a single frame (i.e., it contains eight or fewer bytes of data) or requires multiple frames
- Whether the message is addressed to a specified destination or is a broadcast (global) message [All transmissions are physically broadcast on the CAN bus. An addressed PGN is ignored by all devices other than the one matching the destination address in the CAN bus transmission; global PGNs are received by all devices on the bus.]



Distance Log **PGN: 128275**

This PGN provides the cumulative voyage distance traveled since the last reset. The distance is tagged with the time and date of the distance measurement. The distances through water are accumulated and the values stored during power down and resume counting after power up.

- Total Cumulative Distance is normally set to zero when the log is installed and never reset after that.

- Distance Since Last Reset may manually be set to zero at any time during the voyage and starts to count up from there.

- The "Distance Since Last Reset" is reset by setting the "Distance Since Last Reset" value to 0 with the "Command Group Function" PGN 126208..

Single Frame: No *Priority Default:* 6 *Default Update* 1,000milliseconds *Frequency:* 1 cycles per second
Destination: Global *Query Support:* Opt'l *ACK Rqmts:*

Field	Field Name						Original Reference ID # 39
1	Measurement Date	*Byte Field* 2	*Bit Field*	*Request*	No		
	DD039 Generic date	Days since January 1, 1970,		Date is relative to UTC Time.	*Size:*	*Parameter*	
	DF41 Date, day count	uint16	*Range:* 0 to 65,532 days	*Resolution*	1 day	0 = January 1, 1970, max = ~179 years	
2	Measurement Time	*Byte Field* 4	*Bit Field*	*Request*	No		
	DD158 Generic time of day	24 hour clock, 0 = midnight,		time is in UTC	*Size:*	*Parameter*	
	DF06 Time of day	uint32	*Range:* 0 to 86,401 s	*Resolution*	1x10E-4 s	~24 hours, 0 = midnight, range allows for up to two leap seconds per day	
3	Total Cumulative Distance	*Byte Field* 4	*Bit Field*	*Request*	No		
	DD120 Distance, voyage				*Size:*	*Parameter*	
	DF11 Distance, long	uint32	*Range:* 0 to ~4.295x10E+7	*Resolution*	1 m		
4	Distance Since Last Reset	*Byte Field* 4	*Bit Field*	*Request*	No		
	DD120 Distance, voyage				*Size:*	*Parameter*	
	DF11 Distance, long	uint32	*Range:* 0 to ~4.295x10E+7	*Resolution*	1 m		

Figure 7. A sample NMEA 2000 PGN. [2]

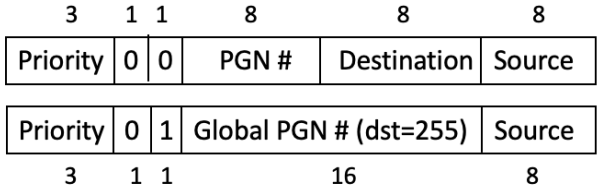


Figure 8. NMEA 2000 format of 29-bit Arbitration Fields in CAN Bus Extended Frame, showing encoding for an addressed (upper) and global (lower) PGN.

NMEA 2000 specifies use of the CAN bus Extended Frame format, employing a 29-bit arbitration field (Figure 8). The first three bits are the message priority field. Bit 4 is reserved and always set to zero [2].

The next bit is the leading bit of the 17-bit PGN identifier. This first bit is, functionally, an addressed/global indicator. If this bit is a 0, the PGN is addressed and, if set to 1, the PGN is global. CAN bus arbitration, then, favors addressed messages over global messages (the overwhelming majority of PGNs are global). The setting of this bit affects the encoding of the next 16 bits, which contain the low-order 16 bits of the PGN identifier.

- In the case of an Addressed PGN, the next 16 bits are the sum of the PGN identifier and the 8-bit address of the intended destination device. As an example, the ISO Address Claim message has a PGN identifier of 060928 (0xEE00). If this message was being sent to a device with address 165 (0xA5), these 16 bits would be encoded as 0xEEA5.
- In the case of a Global PGN, the next 16 bits are merely the remainder of the PGN identifier. As a broadcast message, the destination address is implied to be 255 (0xFF).

The final eight bits of the Arbitration field contain the address of the transmitting device (0-254, 0x00-0xFE). Given the CAN bus arbitration scheme, a lower priority message will always be sent before a higher priority message if multiple devices are ready to transmit at one time. If the priority of multiple transmissions is the same, lower PGNs take precedence over higher numbered PGNs; Addressed PGNs, then, will be sent before Global PGNs. In the case of multiple Addressed PGNs of the same value, a lower destination address wins arbitration over a higher destination address. Finally, if the priority and PGN are the same, the lowest source address wins arbitration.

The next field of importance to NMEA 2000 is the Data Length field, which can take on a value between zero and eight, indicating the length of the Data field, which can be from zero to eight bytes. A PGN containing eight or fewer bytes will be transported in a single CAN bus frame; larger PGNs – referred to as multi-frame PGNs – are fragmented and transported in multiple CAN bus frames. There are two strategies in NMEA 2000 for managing a multi-frame PGN.

- The fast-packet method supports PGNs with up to 223 bytes of data. In this case, the PGN is assigned a serial number (0-7) and can be fragmented into up to 32 CAN bus frames, each of which is assigned a sequence number (0-31). The combination of the serial number and sequence number allows for PGN reassembly.
- The multi-packet method supports PGNs with up to 1,785 bytes of data. This method uses Request to Send (RTS) and Clear to Send (CTS) messages exchanged between the sender and receiver to control when the individual CAN bus frames can be sent.

2.4 *Sample CAN bus frame and PGN encoding*

The complete encoding and transmission of an NMEA 2000 PGN in a CAN bus frame is shown below, with the intent to remove some of the mystery. This example will show the encoding of a Position, Rapid Update message, with a PGN identifier value of 129025 (0x1F801). This is a global message with a priority of 2; in this example, the sending device has the address 150 (0x96) [2].

Figure 8 shows the format of the 29-bit Arbitration field. For this example, the field contains the following values:

- Priority = 0x2 (010)
- Reserved = 0
- High-order PGN identifier bit = 1 (global)
- Low-order 16-bits of PGN identifier = 0xF8-01 (11111000 00000001)
- Source address = 0x96 (10010110)

PGN 129025 has two data fields, namely Latitude and Longitude, each of which is four bytes in length. In this example, we will use the coordinates of the U.S. Coast Guard Research and Development Center in New London, Connecticut:

- Latitude = 41°20'44.1"N (encoded 0xA0-D5-A4-18)
- Longitude = 072°05'45.7"W (encoded 0x00-05-07-D5)

Figure 4 shows the entire CAN bus Extended Frame format. In this example, the frame would be encoded as follows:

- SOF: 0
- Base ID (first 11 bits of the Arbitration field): 01001111110
- SRR: 1
- IDE: 1
- Extended ID (last 18 bits of the Arbitration field): 000000000110010110
- RTR: 1
- Reserved bits: 00
- Data Length: 0x08 (00001000)
- Data: 0xA0-D5-A4-18-00-05-07-D5 (10100000 11010101 10100100 00011000 00000000 00000101 00000111 11010101)
- CRC-15 value: 101001010110010
- CRC Delimiter bit: 1
- ACK Slot bit: 1
- ACK Delimiter bit: 1
- EOF: 1111111
- IFS: 1111111....

Given this data, the transmitted bit stream would appear as follows; stuffed bits, to maintain clock synchronization between the SOF bit and the end of the CRC field, are shown in bold:

00100111111**0**10110000**1**0000110010110100000**1**010001
01000001**11**0101011010010000**11**
10000**1**000001000001010100000**1**111111**0**010101101001
0101100101111111111

The receiving device would interpret this bit stream as shown in Figure 9.

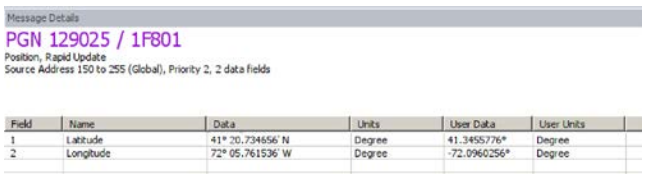


Figure 9. Interpretation of CAN bus frame using the Actisense EBL Reader [1].

3 CYBERSECURITY VULNERABILITIES AND MITIGATIONS

Part I of this paper provided the technical details about the CAN bus and NMEA 2000 standards necessary for an appreciation of the cybersecurity vulnerabilities that are described in this part of the paper. Part II is decidedly less technical than Part I.

3.1 *CAN bus cybersecurity issues*

There have been many papers written about CAN bus security issues. Although these papers typically address vehicular and agricultural applications, CAN bus cybersecurity vulnerabilities in ICS and aviation networks and avionics have also been reported [8, 9]. It is not surprising that papers discussing the maritime environment are hard to find; although the CAN bus protocol has been in place since the late-1980s, it has only been in use in maritime since 2000. Most of the CAN bus cybersecurity papers have been published in the last 15 years, which is when cybersecurity discussions started to become more mainstream and hacking conferences demonstrated attacks on vehicular networks.

This section will explore what kind of vulnerabilities are present that threaten the security of information on CAN bus network. Information security is often discussed in terms of the CIA Triad, , where CIA stands for confidentiality, integrity, and availability. This section addresses CAN bus vulnerabilities in terms of these characteristics of information. Note that some of the scenarios described below are impossible with a properly functioning CAN bus or NMEA 2000-certified device, but might be quite possible if a rogue device was built specifically to violate the CAN or N2K standards.

3.1.1 *Confidentiality*

Confidentiality refers to the secrecy or privacy of information between a sender and receiver. Neither the CAN bus nor NMEA 2000 standards address confidentiality. Both the CAN bus and the NMEA 2000 standard provide a mechanism to send a message addressed to specific receiver; a sender needing product information from another network node, for example, optimizes network bandwidth by sending an addressed message because it does not need a response from all of the devices. Similarly, a command and control message telling a particular actuator or valve to engage should only be sent to the necessary target device.

CAN, however, is a broadcast bus and, therefore every device can hear every transmission [5, 12, 21]. While a properly configured CAN device will not "listen" to an addressed message if it is not the intended receiver, a CAN device could be designed to operate in promiscuous mode and listen to every transmission.

Similarly, the majority of NMEA 2000 PGNs are global messages that employ the broadcast address, so that N2K device sees every global message. Although an NMEA 2000-certified device will not listen to

addressed PGN not intended for this device, a rogue promiscuous N2K node could read all PGNs.

The use of cryptography could be employed to ensure confidential communication between two network nodes [19]. Neither the CAN bus nor NMEA 2000 PGNs have a provision for encryption [2, 4].

At the physical level, it is possible to add a rogue device to the CAN bus [4, 5, 21]. A CAN bus on a car, truck, or passenger coach might have dozens of devices on a network that is 10-100 ft (3-30 m) in length and difficult to surreptitiously access. A ship might have several hundred devices on a network that is more than 1,000 ft (305 m) in length. In any case, it is straight-forward to covertly add a device to a CAN bus; all you need is a short length of backbone pigtail cable, a T-connector to attach to the rogue device, and an obscure access panel on the ship (Figure 10).

Figure 10. Easy physical access to the CAN Bus on a vessel can make it simple to add a rogue device.

The ability to physically insert a rogue node on the CAN bus and the lack of confidentiality in the transmissions provides many ways in which a malicious actor can access and, possibly, exfiltrate information from a vessel's network. An inserted node, for example, can monitor all or a select subset of transmissions on the communications bus; this node can buffer the information for later retrieval by a Bad Actor or transmit the information to another system on or off the vessel in real-time.

3.1.2 *Integrity*

In the vernacular of information security, integrity refers to the correctness of the message; including whether the received message is the same one that was transmitted by the sender, the information is correct, and the purported sender is actually the one who sent the message.

Bit integrity is provided on the CAN bus by use of a CRC calculation; if the receiver calculates a different checksum value than the one contained in the CRC field of the message, then this indicates that a bit error has occurred during transmission. A rogue device can send spurious bits on the line, forcing such transmission errors, causing the receiver to ignore a transmission. A sufficient number of such bit errors could cause other devices to believe that the communication bus is unreliable [4, 5, 21].

The CRC provides bit integrity in the CAN bus only on a per-frame basis. An NMEA 2000 multi-frame PGN is sent in multiple CAN bus frames, but there is no mechanism to ensure bit integrity of the entire PGN message. One of the NMEA 2000 multi-frame mechanisms is the fast-packet method, which allows a PGN of up to 223 bytes sent in up to 32 CAN bus frames; each multi-frame block has a serial number (0-7) and each frame within the block has a sequence number (0-31). Suppose there is a fault on the bus that causes some frames to be lost without the sender knowing. If the sender continues sending multi-frame PGNs, it increments the multi-frame serial number. After eight multi-frames, the serial number wraps back around to match the serial number of the block where the fault occurred. It is possible for a frame from the new block to have a sequence number that matches a missing frame from the old block, which will then be used by the receiver to reassemble the first PGN. This reassembly error scenario is rare under normal circumstances, but such an error could be forced by a rogue node.

A rogue N2K device can also send bogus messages, an attack known as frame injection. As an example, a device on the network could send fake NMEA 2000 messages masquerading as the GPS receiver, compass, or depth gauge, causing false information to be fed to the navigation console. There is no CAN bus or NMEA 2000 mechanism to verify that transmitted data is correct or that the sender is the device that it purports to be [4, 5, 21, 25].

NMEA 2000 and CAN bus standards provide no timestamp in the message body, thus providing no timing integrity. This opens the network to a replay attack, where a legitimate message is stored and retransmitted at a later time by a rogue device. As an example, during an extreme low tide in a narrow channel, an attacker could replay depth gauge information from an earlier extreme high tide, possible causing a ship to run aground [4].

The CAN bus protocol also provides no authentication mechanism, thus providing no way to ensure the integrity of the sender's address. CAN bus nodes derive their device identifier from their serial number, hardware switch settings, or other assignment mechanism, but there is actually nothing that would prevent a device from using an address already in use on the bus. Similarly, a rogue N2K device could use the same source address as another device, and spoof the authentic device and its information [20, 21, 25].

Another attack on message integrity could occur if a rogue device on a CAN bus network uses the same address as another node, in which case it would be impossible for receivers to distinguish between the different transmitters. Indeed, this scenario would allow multiple physical devices to attempt to transmit different data at the same time.

3.1.3 *Availability*

The availability of information refers to the ability of (authorized) users to access data or the network communications channel whenever they need to. There are several attacks on availability possible via the CAN bus.

One of the biggest compromises to availability is a denial-of-service (DoS) or resource exhaustion attack.

These attacks makes a device or the entire network inaccessible by using up all of the available device memory or network bandwidth by bogus requests for service. Consider the example above where a rogue device purposely uses the same address as another node on the CAN bus and continues to transmit when the valid device transmits. The arbitration process is designed to be resolved during the transmission of the Arbitration field; transmitting after losing arbitration is a direct violation of the CAN bus protocol. A receiver detecting a line state different from its transmission while sending its Data field will interpret the event as a transmission error; this will eventually cause the device to enter a BUS OFF state and remove itself from the network. A BUS OFF state is resolved by rebooting the device, the device's CAN bus controller, or, in extreme cases, the entire CAN bus network [5, 8, 25, 27].

Another way to force devices to enter a BUS OFF state is if one or more are operating at different transmission rates. NMEA 2000 requires devices to operate at 250 kbps. If a rogue device was placed on the network persistently operating at a different speed, some or all of the other transmitting devices would eventually enter a BUS OFF state [27].

The CAN bus arbitration process favors devices with lower addresses. By assuming a low address, a rogue device could launch a different form of DoS attack by continually transmitting and blocking out other devices [5, 21].

Another form of DoS attack is for a rogue node to send spurious error or overload frames, or otherwise signal nonexistent error conditions. This can result in unnecessary network or device resets, causing a sequence of network outages [5, 25]. A malicious node can also send false RTS messages and overflow the receiver's input buffer or send bogus CTS messages to hold a connection open and usurp the entire network bandwidth [5].

3.1.4 *Additional Observations*

If there are applicable lessons from the Stuxnet virus – designed to attack centrifuges in Iranian nuclear research facilities and discovered in 2010 – they are that software can be used to attack hardware and malware can be hidden in a working system for years before being activated [30]. It is not too far-fetched to realize a Trojan Horse scenario where a CAN bus controller manufacturer builds malware into a working system. Such a system would work precisely as designed until its malware was activated, which could then launch any one of the attack schemes mentioned above [4, 21].

3.2 *CAN bus security protections and migrations*

A number of mechanisms have been proposed to mitigate or reduce CAN bus security vulnerabilities. While most focus on automotive and vehicular applications, this section will review several that might apply to the maritime environment.

The easiest mitigation strategy is network segmentation, where a network is subdivided into multiple subnetworks, usually based upon their function. NMEA 2000 limits the CAN bus to cable segments of no more than 200 m (650 ft) in length due to employing a 250 kbps transmission speed. CAN bus network designers can add security to the system by separating passenger/crew, entertainment, engineering, navigation, and other functional communications onto different network segments. While this design adds cost to implementation and maintenance, it limits access to critical systems by potential Bad Actors [5, 14, 25, 29].

While the use of cryptography would provide privacy, confidentiality, and authentication for the CAN bus, the protocol issues in the maritime environment make the introduction of encryption methods unlikely. Several methods have been proposed for adding encryption to the CAN bus, but all have focused on relatively small, low-traffic automotive networks that would employ modified, specialized CAN bus nodes. Encryption is also a drain on the limited computational and storage resources of the typical CAN bus node [5, 19, 25, 31]. Shipboard networks are not geographically small and evolving maritime equipment is actually producing an increasing amount of information and network traffic. The protocol overhead added by encryption exacerbates the limitations of the CAN bus protocol that only allows eight bytes of data per transmission; sending larger messages would necessitate adding traffic to an already burdened network. In the maritime environment, it would be possible to encrypt NMEA 2000 PGNs where the use of cryptography would be transparent to the CAN bus protocol, although such methods have not yet been designed. Theoretically, such mechanisms could be added to the N2K standard by defining PGNs for key exchange; use of secret key cryptography would not increase the size of the PGNs although it would add to the processing burden of the node [19].

An intrusion detection system (IDS) is commonly employed on a network to detect, and respond to, cyberattacks. A host-based IDS is generally software that resides on a network node, and analyzes the traffic in and out of the node to detect anomalous behavior; a network-based IDS is a physical device on the network that monitors network traffic to detect aberrant behavior. A host-based IDS would not be feasible in today's maritime environment as it would require changes to the devices themselves and add a computational burden. A network-based IDS, however, could easily be attached to a CAN bus without any impact on the network. A number of IDS methods for the CAN bus have been proposed, all for automotive networks. The research has shown myriad advantages and disadvantages, taking into account the complexity of the algorithms, types of attacks detected, rate of false positives/negatives, and cost [5, 25]. Designing an IDS for maritime CAN bus environments seems to be one area that might bear fruit. Such an IDS would utilize a host-based, passive monitor that can detect anomalous activity on the network without adding any traffic to the bus. Furthermore, it would require no change to the CAN bus protocol.

Fenster, Lee, and Whitfield [13] proposed a machine-learning approach to detecting rogue devices on a CAN bus. Their method identified a subset of

PGNs for a specific application environment. They took advantage of the fact that NMEA 2000 is a proprietary standard, meaning that an attacker will not have complete information when attacking an NMEA 2000 network and, therefore, can be detected. This is not a CAN bus-specific solution, but it suggests an interesting approach to defending the maritime network environment.

4 SUMMARY AND CONCLUSIONS

The CAN bus was developed in the 1980s for the automotive environment. Developed for a trusted network during a time when networks could be trusted, it has no particular security mechanisms or defenses. Indeed, use of the CAN bus shows little sign of diminishing 40-plus years later, yet the security landscape is very different today than it once was; consider that the hacker community has been demonstrating successful attacks on automotive CAN bus networks for more than a decade and CAN bus attack suites employing open-source code and inexpensive hardware are now readily available [26]. We can no longer afford to build networks that are resilient to naturally-occurring errors but not to active attack.

In today's environment of nearly constant cyberwarfare, cyberattacks are planned and scheduled to occur at the convenience of the attacker. Any of the exploits described here might be exacerbated by the fact that a ship at sea has access to a limited pool of personnel and other resources. While a large vessel might have someone trained to administer shipboard information technology (IT) systems and deal with some malfunctions, most are unlikely to have an information security officer trained to recognize and respond to a cyberattack. Furthermore, regardless of the qualifications of the ship's officers and crew, being at sea limits the options for fixing a problem; sometimes the only solution is to power essential devices down.

REFERENCES

1. Actisense: EBL Reader Software, https://www.actisense.com/acti_software/ebl-reader, last accessed 2021/03/01.
2. Anderson, L.C., Luft, L.A.: NMEA 2000® Applied. Presentation at RTCM Meeting, St. Petersburg, FL, May 2002, https://www.nmea.org/Assets/final_rtcm_2002_white_paper.pdf, last accessed 2021/03/01.
3. Applications of Controller Area Network (CAN) Bus: Polytechnic Hub, https://www.polytechnichub.com/applications-controller-area-network-can-bus/, last accessed 2021/03/01.
4. Bozdal, M., Randa, M., Samie, M., Jennions, I.: Hardware Trojan Enabled Denial of Service Attack on CAN Bus. Procedia Manufacturing. 16, 47–52 (2018). https://doi.org/10.1016/j.promfg.2018.10.158.
5. Bozdal, M., Samie, M., Aslam, S., Jennions, I.: Evaluation of CAN Bus Security Challenges. Sensors. 20, 8, (2020). https://doi.org/10.3390/s20082364.
6. Copperhill Technologies: A Brief Introduction to the SAE J1939 Protocol, https://copperhilltech.com/a-brief-introduction-to-the-sae-j1939-protocol/, last accessed 2021/03/01.
7. Corrigan, S.: Introduction to the Controller Area Network (CAN). Texas Instruments Application Report, SLOA 101, https://www.rpi.edu/dept/ecse/mps/sloa101.pdf, last accessed 2021/03/01.
8. CSS Electronics: ICS Alert (ICS-ALERT-17-209-01): CAN Bus Standard Vulnerability. U.S. Department of Homeland Security, https://us-cert.cisa.gov/ics/alerts/ICS-ALERT-17-209-01, last accessed 2021/03/01.
9. CSS Electronics: ICS Alert (ICS-ALERT-19-211-01): CAN Bus Network Implementations in Avionics. U.S. Department of Homeland Security, https://us-cert.cisa.gov/ics/alerts/ics-alert-19-211-01, last accessed 2021/03/01.
10. CSS Electronics: J1939 Explained - A Simple Intro, https://www.csselectronics.com/screen/page /simple-intro-j1939-explained/language/en, last accessed 2021/03/01.
11. Di Natale, M.: Understanding and Using the Controller Area Network. Radical Eye Software, https://inst.eecs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf, last accessed 2021/03/01.
12. Farsi, M., Ratcliff, K., Barbosa, M.: An overview of Controller Area Network. Computing & Control Engineering Journal. 10, 3, 113-120(7) (1999).
13. Fenster, C., Lee, G., Whitfield, W.: Machine Learning in Support of Anomalous Device Detection. U.S. Coast Guard Academy, Electrical Engineering Section (2019).
14. Furuno: Furuno CAN Bus Network Design Guide, https://www.furunousa.com/-/media/sites/furuno/document_library/technical_info/interfacing_and_installation/interfacing_and_installation/furuno_can_bus_network_design.pdf, last accessed 2021/03/01.
15. International Organization for Standardization: Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling. ISO 118981-1. (2015).
16. International Organization for Standardization: Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit. ISO 11898-2. (2016).
17. International Organization for Standardization: Road vehicles — Controller area network (CAN) — Part 3: Low-speed, fault-tolerant, medium-dependent interface. ISO 11898-3. (2006).
18. International Organization for Standardization: Road vehicles — Low-speed serial data communication — Part 1: General and definitions. ISO 11519-1. (1994).
19. Kessler, G.C.: An Overview of Cryptography, https://www.garykessler.net/library /crypto.html, last accessed 2021/03/01.
20. Lin, C., Sangiovanni-Vincentelli, A.: Cyber-Security for the Controller Area Network (CAN) Communication Protocol. In: 2012 International Conference on Cyber Security. pp. 1–7 (2012). https://doi.org/10.1109/CyberSecurity.2012.7.
21. Matsumoto, T., Hata, M., Tanabe, M., Yoshioka, K., Oishi, K.: A Method of Preventing Unauthorized Data Transmission in Controller Area Network. In: 2012 IEEE 75th Vehicular Technology Conference (VTC Spring). pp. 1–5 (2012). https://doi.org/10.1109/VETECS.2012.6240294.
22. National Marine Electronics Association (NMEA): NMEA 0183 Interface Standard, https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard, last accessed 2021/03/01.
23. National Marine Electronics Association (NMEA): NMEA 2000® Interface Standard, https://www.nmea.org/content/STANDARDS/NMEA_2000, last accessed 2021/03/01.
24. National Marine Electronics Association (NMEA): OneNet Standard for IP Networking of Marine

Electronic Devices, https://www.nmea.org/content/STANDARDS/OneNet, last accessed 2021/03/01.
25. Palanca, A., Evenchick, E., Maggi, F., Zanero, S.: A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks. In: Polychronakis, M. and Meier, M. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 185–206 Springer International Publishing, Cham (2017).
26. Payne, B.: Car Hacking: Accessing and Exploiting the CAN Bus Protocol. Journal of Cybersecurity Education, Research and Practice. 2019, 1, (2019).
27. Pfeiffer, O., Keydel, C.: Challenges of CANopen Node ID assignment, avoiding duplicates. Presented at the 1st International Mobile Machine Control (MMC) Conference (2013).
28. SAE International: SAE J1939 Standards Collection on the Web: Content, https://www.sae.org /standardsdev/groundvehicle/j1939a.htm, last accessed 2021/03/01.
29. U.S. Coast Guard: Cyber Incident Exposes Potential Vulnerabilities Onboard Commercial Vessels. Marine Safety Alert 06-19, https://www.dco.uscg.mil/Portals/9/DCO%20Documents /5p/CG-5PC/INV/Alerts/0619.pdf, last accessed 2021/03/01.
30. Zetter, K.: How digital detectives deciphered Stuxnet, the most menacing malware in history, https://arstechnica.com/tech-policy/news/2011/07/how-digital-detectives-deciphered-stuxnet-the-most-menacing-malware-in-history.ars, last accessed 2021/03/01.
31. Zimmermann, T., Bauer, J., Aschenbruck, N.: CryptoCAN – Ensuring Confidentiality in Controller Area Networks for Agriculture. In: Reinhardt, D., Langweg, H., Witt, B.C., and Fischer, M. (eds.) SICHERHEIT 2020. pp. 79–90 Gesellschaft für Informatik e.V., Bonn (2020). https://doi.org/10.18420/sicherheit2020_06.