Star-Shape CAN-Bus System

Keaton Smith (kps325),
Renjie Zhang(rz1189),
Rayvanth Sabbineni (rs325),
Scott Song (shs474),
Zihui Yuan(zy1028)

Abstract

The Controller Area Network (CAN) present in almost all modern automobiles has been discovered to be vulnerable to a targeted attack. By sending malicious signals to the car's CAN an attacker is able to disrupt, or even disable, critical systems such as the headlights or brakes. Our proposal is to mitigate the effectiveness of a targeted attack through two methods. Thus, we design a **Star-Shape CAN-Bus System.** First, segmenting the CAN so signals from one component cannot disrupt another. Second a system of checking frames sent by each component to prevent the distribution of bad frames. We feel these are the most robust defenses that can be implemented without a total overhaul of the CAN standard.

1 Introduction

Modern automobiles are complex pieces of technology. Within any modern vehicle there are dozens of electric systems which control functionality. Research suggests that these electric subsystems contain 100 million lines of code altogether; modern automobiles have 50 different electric subsystems or electronic control units on average (Zanero 2015). Some of these electric systems are designed to operate autonomously.

However, other electric systems cannot operate independently and require some means of information exchange with each other. The CAN standard provides a solution to this problem; it allows electric systems within an automobile to communicate with each other (Zanero 2015). An important point is that this communication takes place without a host computer. The CAN system is a message-based protocol used to send messages on the CAN Bus. The CAN bus can be thought of as an internal network that ties together the various electronic control units within an automobile (Zanero 2015).

2 Background

As mentioned above the CAN system was introduced to manage the increasing complexity of modern automobiles. There is no doubt that the CAN system has been necessary for many of the advanced features found in present day vehicles. However, the presence of the CAN system increases the attack surface of current day vehicles.

Attacks on the CAN system tend to take advantage of error reports that are generated by electric systems and their underlying devices. If an electric system or device is generating too many errors than the system is placed into a "Bus Off state." When in this state the system is cut off from writing or reading any data into CAN, effectively the underlying system is made inert or inoperable. One principle of secure design that is not being followed is the principle of fail-safe defaults; it also breaks the 'availability' in the principle.

The CAN system does do a good job of

handling errors when there aren’t too many. However, when there are too many errors the underlying system is made inert. This violates the principle of fail-safe defaults because a system being made inert when the automobile is in use is not “fail-safe.” For example, if an automobile breaks or airbags are disabled when it is being used, this is obviously not a safe state. In more general terms, a key component of security that is being attacked is availability. As previously mentioned, important systems within a car can be made unavailable very easily by an attack that generates too many errors.

There are five main types of errors that are commonly associated with the CAN system. These errors are bit errors, stuff errors, CRC errors, form errors and acknowledgment errors (Zanero 2015). Although a malicious attacker can exploit all of the errors listed above, one of the more commonly exploited errors are bit errors (Zanero 2015). An electronic control unit that is sending bits on the CAN bus simultaneously monitors the CAN bus for errors.

A bit error will occur whenever a transmitting node (transmitting electronic control unit), notices that the logical value on the CAN bus is different from the bit value it is trying to send (Zanero 2015). When this occurs the node quickly interrupts the frame transmission, the message being sent, with an error message that includes an error frame. The original frame transmission is then dropped, and the sending node will attempt to retry the transmission. A single incorrect bit in a frame transmission is justification for the frame transmission to be dropped (Zanero 2015).

Therefore, all an attacker needs to do is flip one bit of the transmission frame, and the transmission frame will be dropped. The sending node will attempt to resend the frame transmission, but if the attacker is able to disrupt just one bit in each retransmission then each retransmission will be discarded as well. What ends up happening is that the CAN system gets stuck. The sending node cannot get its message through the can system and in its repeated attempts at retransmission it takes up system resources that other nodes need to send their messages. As a result, these other nodes are unable to send messages as well and the functionality of the entire CAN system is disrupted. Thus by flipping a single bit an attacker can create a very powerful Denial of Service attack.

It has been noted by many papers that it is hard to mitigate the vulnerabilities within the CAN system (Maggi 2017). This is because the vulnerability is inherent to the way the system is designed. That said there are things we can do to make the CAN system safer. We will focus on two techniques to make the CAN system less susceptible to attack. These two techniques are segmentation and detection. Detection involves two sub categories: Frame analysis and Bus load.

For the benefit of the reader we will provide a high-level overview of these two techniques; this information will be helpful when viewing our design and implementation provided later in our report. Segmentation as applied to the CAN system usually involves decoupling safety-critical electronic control units from non-safety-critical electronic control units (Zannero 2015). As a result, errors generated by non-safety-critical electronic control units should not be able to disrupt safety-critical electronic control units.

Error detection is normally handled by an intrusion detection system that monitors network traffic. In this case the intrusion detection system would monitor traffic on the CAN system. These intrusion detection systems try to determine if an attack is in progress, if they detect that an attack is in progress, they generate an alert. One set of features that intrusion detection systems can use to generate alerts is the set of features encompassed by frame analysis. For example, if the same bit within a frame is incorrect in each message that an electronic control unit sends, then there is a high probability there is something malicious going on. This is because it is unlikely for the same bit to be wrong over and over again without malicious activity, errors with no malicious activity behind them tend to be more dispersed and random. Thus, our intrusion detection system could pick up on the same bit being wrong over and over again and respond by generating an alert.

Intrusion detection systems can also consider Bus load and related characteristics as another set of features. For example, if Bus load is very high an intrusion detection system can generate an alert. Of course, our intrusion detection system might suffer from high a rate of false positives if it just uses bus load and related characteristics as a feature set.

3 Threat Model

The following is a rundown of the potential threats, and the proposed solutions, inherent within the CAN system. We also provide an example attack tree for CAN.

Threat:	**Solution:**
Attacker could access CAN system through OBD-II ports.	Use access control for the OBD-II ports or deny physical access to them altogether.
Attackers can access the CAN system directly through the CAN bus.	Segment the CAN system so corrupted devices are incapable of affecting other devices.
Attackers can send malicious frames to make working devices appear malfunctioning.	Encrypt each frame to help ensure only valid frames get read.
An attacker could remotely connect to the CAN bus to disable a connected device.	Build a robust firewall to prevent the installation of uncertified software.

Table 3.1 Threat Model

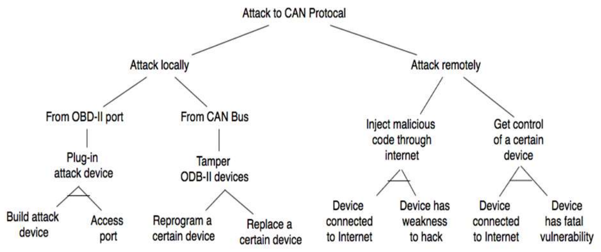
Attack to CAN Protocal
Attack locally
Attack remotely
From OBD-II port
From CAN Bus
Inject malicious code through internet
Get control of a certain device
Plug-in attack device
Tamper ODB-II devices
Build attack device
Access port
Reprogram a certain device
Replace a certain device
Device connected to Internet
Device has weakness to hack
Device connected to Internet
Device has fatal vulnerability

Figure 3.1 Attack Tree

4 Design

The proposed solution for the current CAN-Bus Flaw would be a switch to a star shape network topology architecture with the following security properties:

- **It can detect the DOS attacks before/while/after execution.**
- **It can mitigate the effects of, or impede, DOS attacks with more robust architecture other than the original one-bus architecture.**

Our Star-Shape CAN-Bus System contains the following components:

A **Trusted Center Network Dispatcher** which is in the middle of the star network schema and continuously monitors all Sub-CAN bus networks for incoming DOS attacks.

External Reachable Sub-CAN Bus Networks which are Sub-CAN Networks containing non-critical nodes that are accessible from the outside.

Internal Unreachable Sub-CAN Bus Networks which are Sub-CAN Networks containing critical nodes that are inaccessible from the outside world.

Trusted Mediators contained in the Trusted Center Network Dispatcher which allow for minimal frames passing between sub CAN bus networks to exchange necessary information for the complete functionality of the car.

The Architecture Diagram shows our Star-Shape CAN-Bus System. The number of **External Reachable Sub-CAN Bus Networks** in the system should be as low as possible. (Ideally only one for necessary external accesses). Nodes in the same sub-CAN bus Networks could send frames in the same way as the original one-bus CAN System. Only a few nodes between different Sub-CAN bus Networks could communicate through **Trusted Mediators** in the **Trusted Center Network Dispatcher.** The **Trusted Center Network Dispatcher** monitors all sub networks connected to it and detects whether there are DOS attacks happening in each subnetwork or between adjacent sub networks.

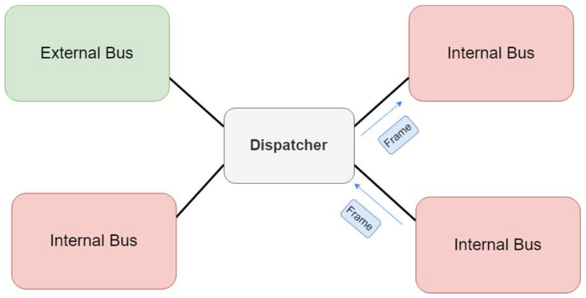

Figure 4.1 Star-Shape CAN-Bus System Architecture

5 Implementation

The Star-Shape CAN-Bus System uses the following technology to detect, mitigate the effects of, or impede DOS attacks.

5.1 DOS Attack Detection:

The **DOS Attack Detection Module** is contained in the **Trusted Center Network Dispatcher**, it detects whether there are DOS attacks happening in each sub network, or between sub networks, with the dispatcher monitoring the whole system. There are two types of DOS Attack Detection in this Module, **Bus Load and Node Integrity Attack Detection** and **Dominant Bit in Error Frame Attack Detection.**

5.1.1 Bus Load and Node Integrity Attack Detection.

One common way for the attackers to perpetrate a DOS attack is to add a surreptitious node into the CAN bus. In our system, all CAN nodes in each Sub CAN-Bus Networks are characterized by a differential internal resistance Rdiff(see Figure below), we will maintain a register table for each Sub CAN-Bus Network which saves each legally registered nodes' Rdiff. Thus, we know the legal bus load for each sub-network. Once an attacker physically adds a surreptitious node into one of the sub bus networks, the bus load for it will be changed and the **DOS Attack Detection Module** will detect the illegal bus load change by checking the original legal register table and it will alert the user there might be an illegal CAN Bus Connection.

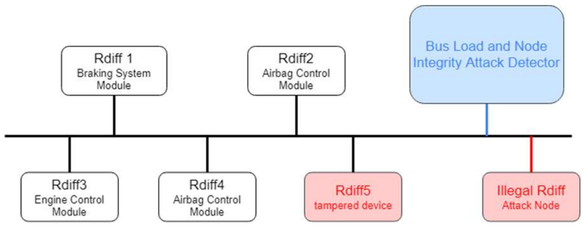

Figure 5.1.1 Bus Load and Node Integrity Attack Detection

To detect the attacks perpetrated by tampering with a legally registered node other than adding an illegal attack node, **DOS Attack Detection Module** will check the integrity of each node whenever the whole system starts and find whether there are any altered nodes in the system and alert the user to prevent DOS attacks caused by the modified node before it begins.

5.1.2 Dominant Bit in Error Frame Attack Detection.

While a DOS attack is in progress, the error frame is always corrupted by the same dominant bit in the frame sent by the node which executes a proposed algorithm. The **DOS Attack Detection Module** will observe such an anomaly and alert the user when a potential attack is in progress.

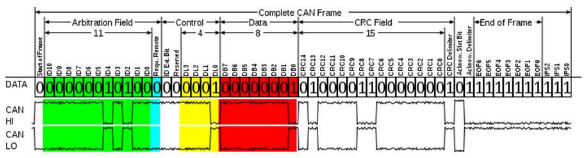

Figure 5.1.2 Structure of a CAN frame

5.2 Star-Shape Network Segmentation

In our **Star-Shape CAN-Bus System**, we use a star network topology architecture, all **Sub-CAN Bus Networks** are connected to the **Trusted Center Network Dispatcher** which continuously monitors all Sub-CAN bus networks for impending DOS attacks with its **DOS Attack Detection Module**.

To apply Network Segmentation to our system, all **Sub-CAN Bus Networks** are separated into two groups **External Reachable Sub-CAN Bus Networks** and **Internal Unreachable Sub-CAN Bus Networks.** Nodes in **External Reachable Sub-CAN Bus Networks** are nodes that are in control of non-critical systems, nodes in **Internal Unreachable Sub-CAN Bus Networks** are critical nodes which will be well protected from any possible outside connection. This will mitigate the effects of DOS attacks and will prevent DOS attacks from tampering with the entire CAN system.

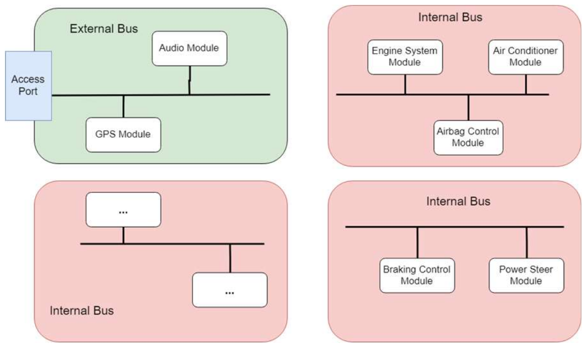

Figure 5.2.1 Idea of Network Segmentation

Furthermore, in our **Star-Shape CAN-Bus System**, connections between nodes from different **Sub-CAN Bus Networks** can only happen through the **Trusted Mediators** contained in **Trusted Center Network Dispatcher**. This will keep the DOS attack in local **Sub-CAN Bus Networks** without further affecting other critical parts of the system.

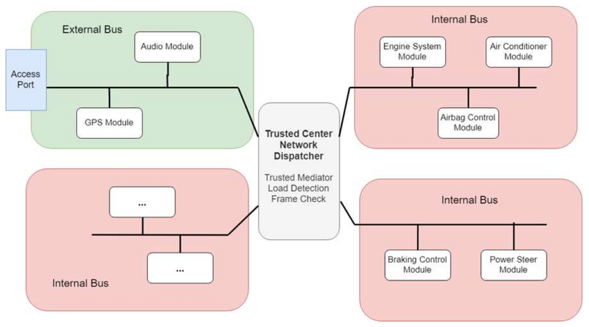

Figure 5.2.2 Detail Architecture of Star-Shape CAN-Bus System

6 Evaluation

Our implementation combines more than one mechanism. However, not every attack will trigger every mechanism. Different attacks will trigger different mechanisms.

When an attacker attempts to plug-in a new device to the CAN bus locally, the central monitor will take on the load detection responsibility and then realize there is an additive device connected to protect against malicious action. The reaction from a load detector happens instantly far earlier before the malicious device start attacking. With this mechanism, new devices are by default disallowed, thus when new devices are needed, some mechanisms are needed to make the process secure and ensure the validity of these new devices.

When one of the existing devices is hacked and starts attacking by sending fake error frames, the attack detection module is likely to catch it through bit checking. In most cases, fake error frames are in a certain pattern, which will be detected immediately and cleaned. In this way the typical attack of forging error frames will no longer threaten the CAN system.

If an attack unfortunately bypasses the detection module, it could not only impact the infected device but also other devices connected to the same bus. However, since we grouped devices into different sub-buses, we can guarantee that only devices within the same domain as the hacked node could potentially be attacked. In this way, even if one device is unfortunately turned malicious, it will never penetrate other buses.

With our defense mechanism, the system is immune to not only typical CAN attacks, but also unexpected attacks, the impact to whole system will be limited within an acceptable range.

7 Related Solutions

There are various other solutions that have been proposed to fix the design flaws that exist in the CAN system. Two other solutions that exist to the design flaw which are related to our solution would be the independent implementation of the attack detection mechanism or network segmentation. Our solution is the combination of these two principles but the independent implementation of either of these would be a solution of its own.

The attack detection mechanism would work by adding a module to the CAN system which scans the bus load of the system and checks the error frames within the system. Observing the bus load will allow for the detection of malicious components that are attached to the system and the components access to the system can be denied. The error frames can be checked to see if continuous error frames are being sent with the same error bit. Multiple error frames with the same error bit would imply that an error frame is being copied. This information would reveal that errors are being forced into the system.

The goal of network segmentation would be to separate the CAN bus into different sections though trusted mediators such as gateways and firewalls. The devices within the CAN bus would separate according to their function. Indiscriminate frames would be prevented from broadcasting throughout the entire system but the minimal frames inter-network exchange necessary would still allow for car functionality.

The advantage of the independent implementations of these solutions would be that they are easier to execute than our solution since our solution is the combination of both of those solutions. The disadvantage of these alternative solutions would be that the attack detection mechanism does not help with attack prevention whereas the segmentation would not be able to detect an attack on the system. Our solution would both be able to prevent attacks that are made to the system and detect attacks in the case that an attack was able to make it through the segmentation of the bus.

A different solution that would be able to stop malicious behavior in the CAN bus system would be the application of a strong encryption to the data fields of the frames made possible by stream ciphers or block ciphers. This solution is similar to the attack detection mechanism, but the difference is that the attack detection analyzes the frames and does not encrypt them. The problem with the encryption solution is that it does not meet the simultaneous low cost and real-time requirements of automotive embedded systems.

8 Conclusions

In the preceding pages we have discussed our solutions to the vulnerabilities inherent in the CAN system present in most automobiles on the road today. Our solutions work to mitigate the effectiveness of an attack that exploits the CAN system. Through segmentation we help mitigate the devices that such an attack is able to affect. Through our process of encrypting each frame the CAN system processes we make attacks not only more difficult but also easier to detect as any malicious program must decrypt each frame. This decryption slows down the malicious program making diagnosis faster and easier. Through these methods we feel the danger to consumers can be reduced until such time as a new CAN standard is adopted by the automobile industry.

References

1. "A Stealth, Selective, Link-layer Denial-of-Service Attack Against Automotive Networks" https://www.politesi.polimi.it/bitstream/10589/126393/1/tesi_palanca.pdf
2. "The Crisis of Connected Cars: When Vulnerabilities Affect the CAN Standard" https://blog.trendmicro.com/trendlabs-security-intelligence/connected-car-hack/
3. "Uncatchable 'Flaw' Affects Most of Today's Modern Cars" https://tech.slashdot.org/story/17/08/17/1825227/unpatchable-flaw-affects-most-of-todays-modern-cars
4. "An uncatchable flaw in CAN protocol expose modern cars to hack" https://securityaffairs.co/wordpress/62100/hacking/can-protocol-flaw.html
5. "A Vulnerability in Modern Automotive Standards and How We Exploited It" https://documents.trendmicro.com/assets/A-Vulnerability-in-Modern-Automotive-Standards-and-How-We-Exploited-It.pdf
6. YouTube Video "A Stealth DoS Against CAN-Based Automotive Networks" https://www.youtube.com/watch?v=oajtDFw_t3Q
7. Wikipedia CAN Bus https://en.wikipedia.org/wiki/CAN_bus