Intrusion Detection System on Automotive CAN Bus

DESIGN DOCUMENT

sdmay24-39
Advisor/Client:
Manimaran Govindarasu
Students:
Cole Burkle - Lead Vulnerability Tester/Car Testbed Lead
Trace Haage - Client Liaison/Pi Testbed Lead
Tiffanie Fix - Vulnerability Research and Development Lead
Alec Cose - Testbed Design/IDS Rule Developement
sdmay24-39@iastate.edu
https://sdmay24-39.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

- SecurityOnion SIEM Environment
- Snort IDS/IPS
- Raspberry Pi 3 Model B+
- PICAN2 CAN Bus Hat
- ECUSIM2000
- Python Library - canutils, cantools, PyQt5
- Raspbian OS

Summary of Requirements

- Design testbeds with CAN channels that accurately simulate how passenger vehicles use the CAN bus protocol.
- Introduce attack vectors used on CAN networks to our testbeds.
- Create rules for the Intrusion Detection System to properly detect these known attacks in our testbeds.
- Ensure the Intrusion Detection System is both accurate and efficient when placed into any CAN channel.

Applicable Courses from Iowa State University Curriculum

- CYB E 230
- CYB E 231
- CYB E 331
- CPR E 288
- CPR E 430
- CPR E 489
- COM S 309

New Skills/Knowledge acquired that was not taught in courses

- General knowledge about control systems and CAN bus protocols
- Python Libraries - canutils, cantools, PyQt5
- Raspberry Pi
- Arduino UNO/Arduino UNO IDE
- CAN bus attacks
- Project Management

Table of Contents

1 Team

1.1 Team Members

Trace Haage

Cole Burkle

Tiffanie Fix

Alec Cose

1.2 Required Skill Sets For Our Project

- Programming
  - Python
  - C
- Networking
- Packet Capturing
- Packet Disassembly
- Reverse Engineering
- Real Time Operating Systems
- Real-time Documentation
- Technical Writing
- Vulnerability Research

1.3 Skill Sets Covered By The Team

Trace, Tiffanie, Cole, Alec - Programming

Tiffanie, Cole - - - - - - - - - - Local Networking

Alec, Tiffanie, Cole - - - - - - Packet Capturing

Trace, Cole - - - - - - - - - - - - Packet Disassembly

Trace, Tiffanie, Cole, Alec - - Reverse Engineering

Cole - - - - - - - - - - - - - - - - - Real Time Operating Systems

Alec - - - - - - - - - - - - - - - - - Real-time Documentation

Trace, Tiffanie - - - - - - - - - - Technical Writing

Tiffanie - - - - - - - - - - - - - - - Vulnerability Research

1.4 Project Management Style Adopted By The Team

Majority vote, final decision goes to our Faculty Advisor

1.5 Initial Project Management Roles

Trace - Client Liaison/Pi Testbed Lead

Alec - Testbed Design/IDS Rule Development

Tiffanie - Vulnerability Research and Development Lead

Cole - Lead Vulnerability Tester/Car Testbed Lead

2 Introduction

2.1 Problem Statement

We are attempting to solve the problem of automotive cyber security in regards to CAN bus protocol. We will be designing testbeds and a corresponding Intrusion Detection System (IDS) to monitor a CAN network and alert of anomalies or malicious traffic.

2.2 Requirements & Constraints

Functional Requirements:

- Data Collection: The system should be capable of monitoring and capturing real-time traffic on the CAN bus.
- Analysis Engine: Develop an algorithm to analyze CAN bus data for potential intrusions.
- Alert Generation: When an intrusion or anomaly is detected, the system should generate an alert with relevant details.
- Reporting: The system should be able to produce reports showing detected intrusions, patterns, and trends.
- Interoperability: The IDS should be capable of integrating with other systems, such as an Intrusion Prevention System (IPS) or a Security Information and Event Management (SIEM) system.
- False Positives/Negatives: The IDS will need to maintain a low amount of false positive alerts and missed false negative allowances.

Performance Requirements:

- Real-time Processing: The IDS must analyze CAN bus data in real-time or near real-time.
- Scalability: The system should be scalable to handle a varying number of CAN messages per second.

Usability Requirements:

- User Interface: A dashboard or interface that provides an overview of the system status, alerts, and detailed reports.
- Configurability: Administrators should be able to update the detection rules or train the model with new data.

Security Requirements:

- Data Integrity: Ensure that the data collected from the CAN bus is not tampered with during analysis.
- Authentication and Authorization: Only authorized users should have access to the IDS interface and configuration settings.
- Adversarial Attacks: Any machine learning approaches used should be preventing any possible attacks against the model.

Constraints:

- Hardware Limitations: Depending on the platform or tools being used, there may be restrictions related to processing power, memory, or storage.
- Data Privacy and Ethics: When using real-world data, ensure that no private or sensitive information is accessed or disclosed.
- Complexity of the CAN bus: The sheer volume and speed of messages of the CAN bus may pose challenges for real-time analysis.
- Budget: There may be costs associated with acquiring hardware, software, or datasets.
- Availability of Data: Getting access to real-world CAN bus data can be challenging. Simulated data might not capture all intricacies of real-world traffic.

2.3 Engineering Standards

IEEE 802.10/IEEE 802.1Q: Standards for LAN/MAN security implementation

We are going to be directly working with LAN's/WLAN's as this is how devices on our network will be connected. Our protocol we are using is CAN-over-IP so Ethernet standards directly relate to our project, however this standard has been largely replaced by IEEE 802.1Q.

IEEE 829-2008: Standard for Software and System Test Documentation

As we'll be working with a system (ICS), the eight defined stages for system testing will be a basis for us to ensure our documentation is up to standards in terms of security. As it was not made mandatory to complete each of the documents, we can look at each and select the ones that would cover what we need to test in our system.

IEEE C37-2040: Standard Cybersecurity Requirements for Substation Automation, Protection, and Control Systems

Again, we are working with a control system, so any standards or requirements will need to be known to cover all types of attacks and defenses.

IEEE 802.11: Wireless Networking - "WiFi"

Devices can communicate via Can-to-WiFi using a gateway, so this standard will be used throughout while looking at security.

2.4 Intended Users & Uses

This IDS would benefit users of vehicles which rely on the CAN channel for their car to communicate, and the information about detected intrusions can be useful to vehicle manufacturers on creating better security.

Users of passenger vehicles that communicate with the CAN bus protocol will benefit from the use of our IDS through the added security this can provide by being able to detect possible issues or attacks that could threaten their safety while using the vehicle. This IDS could be utilized by vehicle manufacturers to view the attacks that are being used to exploit vulnerabilities in their CAN networks and make fixes based on what they learn from the IDS.

3 Project Plan

3.1 Task Decomposition

Task 1: Project Planning and Research for Requirements and Equipment

- Research CAN bus documentation
    - Communication method
    - Hardware/wiring needed to create a CAN bus channel
    - CAN bus frame formatting
    - Understand the characteristics of normal and malicious CAN traffic
- Research requirements for functional and practical IDS
- Gather prior designs and implementations of an IDS on a CAN bus system
- Search possible car parts in nearby junkyard
    - Send in order and acquire internals from a car
- Search for possible compatible ECU Simulators

Task 2: Raspberry Pi Testbed Design and Implementation

- Configure Raspberry Pi with PICAN2 CAN bus HAT
- Set up CAN channels using software libraries to emulate passenger vehicle use
- Link Arduino to CAN network
- Link multiple Arduinos to CAN network
- Link an ECU simulator to the testbed
- Hardware and Software Inventory

Task 3: Raspberry Pi Testbed Validation Testing

- Send CAN messages from the Raspberry Pi, log and trend to ensure frame validity
- Send CAN messages from one Arduino, log and trend to ensure frame validity
- Send CAN messages from multiple Arduinos, log and trend to ensure no traffic issues
- Send CAN messages from the ECU, log and trend to ensure frame validity

Task 4: Pontiac G6 Testbed Design and Implementation

- Power and connect components to bus
- Set up monitoring device to view and log CAN messages
- Set up and connect MiTM device to the CAN network

Task 5: Pontiac G6 Testbed Validation Testing

- Send CAN messages to monitoring device, log and trend
- Inject and Alter CAN messages using MiTM device, log and trend

Task 6: CAN IDS Design and Implementation

- System Configuration
    - Define what tools will be used for network monitoring
    - Decide between signature based and anomaly based detection
- Implementation
    - Deploy the IDS on the chosen platform
    - Integrate with the CAN network to monitor traffic
    - Implement alert mechanisms (e.g., visual, auditory, logs)

Task 7: Integrate IDS into Pi and Car Testbed

- Assign appropriate network settings (e.g., IP addresses, if needed)
- Configure the IDS to monitor the CAN bus traffic specifically
- Input any necessary CAN message formats or templates into the IDS

Task 8: Functional Testing and Performance Evaluation

- Network Test
    - Validate that the CAN network is working properly and as expected
- Initialization Test
    - Ensure the IDS starts up and initializes without errors
- Configuration Test
    - Change various IDS settings and ensure they apply correctly
- Alert Mechanism Test
    - Ensure that when an anomaly is detected, the alert mechanism (be it a log, sound, visual prompt, etc.) works as intended
- Performance Evaluation
    - Throughput Testing
        - Assess the maximum amount of CAN bus traffic the IDS can handle without missing any alerts or causing delays
    - Resource Usage
        - Monitor CPU, memory, and other resource utilizations under varying loads to ensure the IDS doesn't overwhelm the system
    - Latency Measurement
        - Measure the time taken for the IDS to generate an alert once a malicious message is introduced
    - Scalability Test
        - Increase CAN bus traffic volume and complexity to see if the IDS scales effectively
    - Recovery & Resilience Test
        - Introduce faults or simulate system crashes and observe how quickly and effectively the IDS can recover

Task 9: Attack/Defense Testing and Evaluation

- Planning & Strategy Development
    - Define the scope, objectives, and desired outcomes for the cybersecurity testing.
    - Determine which attack vectors/scenarios will be simulated
    - Ensure a safe testing environment to avoid unintended disruptions
- Establishing Baselines
    - Monitor and record typical "benign" CAN bus traffic to understand normal patterns.
    - Configure the IDS with current settings and definitions
- Attack Simulation
    - Common Attacks: Simulate well-known attacks on the CAN bus
    - Novel Attacks: Try newer or lesser-known attack techniques to test the IDS's resilience against unexpected threats
    - Sustained Attacks: Launch prolonged attacks to test the IDS's endurance
    - Stealth Attacks: Introduce subtle, slow, or low-volume malicious activities to check the IDS's sensitivity
- False Positive Evaluation
    - Inject benign traffic variations that could potentially be misconstrued as malicious.
    - Monitor and record instances where the IDS incorrectly flags these as threats.
- False Negative Evaluation
    - Analyze the IDS logs and compare against the introduced attacks
    - Identify instances where the IDS failed to detect and alert on genuine threats
- Impact & Effectiveness Assessment
    - Determine how quickly the IDS responds to threats
    - Assess the accuracy of the IDS's alerts and logs
    - Evaluate the IDS's resource consumption during attack simulations (CPU, memory, etc.)
- Recovery & Resilience Testing
    - After simulating attacks, test how well the system recovers
    - Determine if the IDS maintains its integrity or requires resets/configuration adjustments after attacks
- Iterative Adjustments & Retesting
    - Tweak IDS configurations, signatures, or settings based on testing outcomes
    - Re-run specific attack scenarios to see if detection and response improve

Task 10 (Stretch Task): Scale Up Both Networks/Testbeds

3.2 Project Management/Traffic Procedures

We will be using the agile-scrum style of project management for our project. We chose to use this style because our tasks will need to be completed in a mostly linear order, so putting each task into a sprint to ensure we finish that in the given amount of time to allow ample time to complete dependent tasks. This allows us to identify the progress we are making while establishing a well-versed line of communication in what has, is and needs to be done. This includes regular meetings and tracking what each member has done to assure we are meeting expectations and moving forward.

Another solution we are considering is doing our progress documentation using Gitlab and Discord. All changes made in software will be documented using issues in Gitlab, while other non-code related issues will be tracked using Discord.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Task 1: Project Planning and Research for Requirements and Equipment

- Comprehensive understanding of the CAN bus protocol
  - CAN frame format
  - How CAN messages are sent on the Bus channel
- Comprehensive understanding of Intrusion Detection System and its uses
- Equipment fully gathered for virtual and physical testbed

Task 2: Raspberry Pi Testbed Design and Implementation

- Raspberry Pi is bootable and recognizes PiCan2 bus hat
- CAN channel is initialized on Raspberry Pi
- Raspberry Pi recognizes Arduino UNO's

Task 3: Raspberry Pi Testbed Validation Testing

- Successful transmission of CAN messages from Raspberry Pi
- Successful transmission of CAN messages from single Arduino UNO
- Successful simultaneous transmission from two Arduino UNOs
- Successful transmission of CAN messages from ECU Simulator

Task 4: Pontiac G6 Testbed Design and Implementation

- Successful integration of car components into the CAN bus network
- Successful integration of monitoring device into network
- Successful integration of MiTM device into network
- All components can recognize each other

Task 5: Pontiac G6 Testbed Validation Testing

- Monitoring device correctly logs all CAN messages, no missed frames
- MiTM device properly injects message in real time, within 0.5 seconds

Task 6: CAN IDS Design and Implementation

- Identify several different attack vectors that the IDS should detect
- The IDS should be programmed to detect these found CAN bus attack vectors

Task 7: Integrate IDS into Pi and Car Testbeds

- Network can properly communicate with IDS using correct static IP address
- IDS configured on monitoring device to view testbeds
- IDS is portable (physical and virtual)

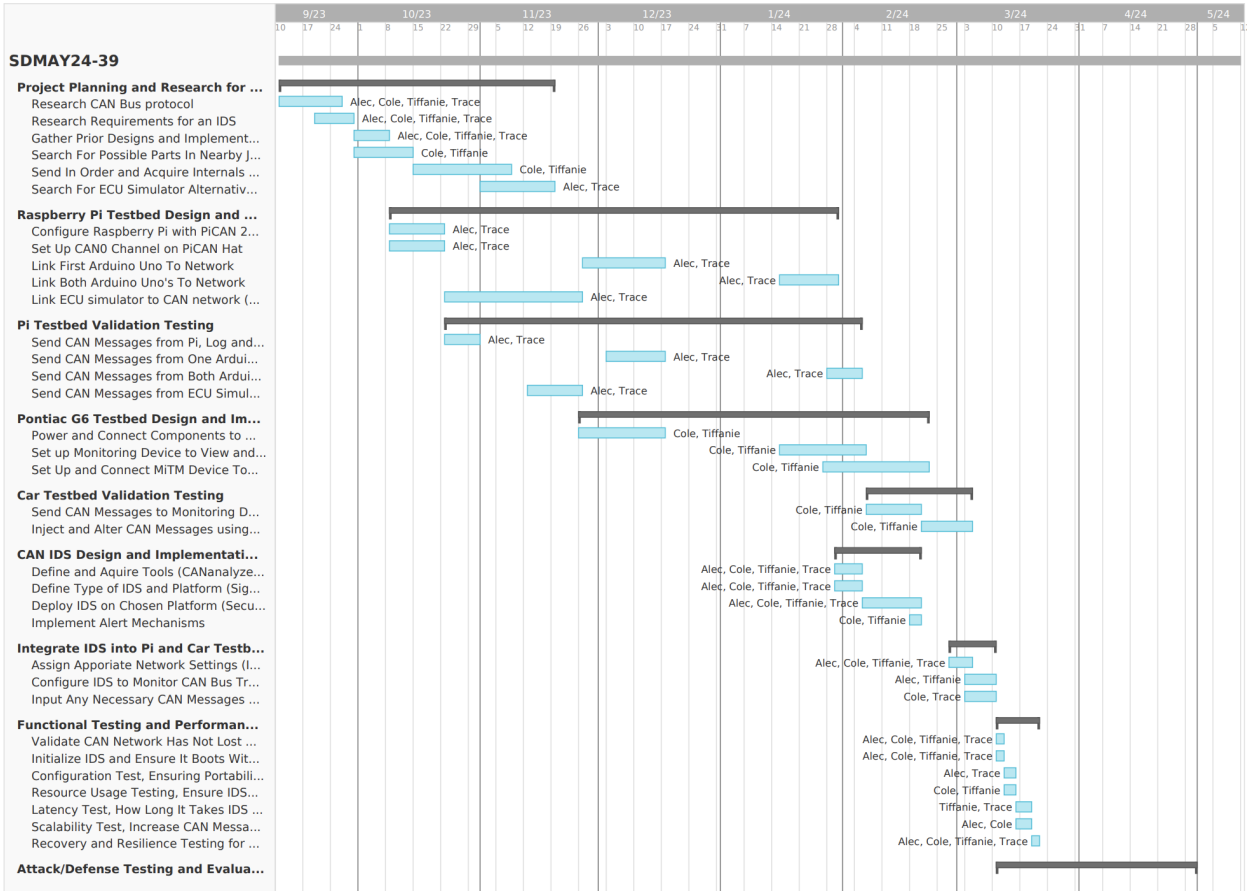
Task 8: Functional Testing and Performance Evaluation

- The IDS should start up and initialize without errors or crashes
- IDS settings and other configurations should all apply correctly
- When anomalies are detected, alerts are properly sent out
- Ensure the IDS can handle large amounts of CAN bus traffic
- Test that performance in resource usage remains stable while IDS is in use

Task 9: Attack/Defense Testing and Evaluation

- The IDS should have a 90% detection rate for each designated attack vector
- The IDS should have less than a 5% false positive rate for non-malicious CAN bus messages

3.4 Projected Timeline/Schedule



3.5 Risks And Risk Management/Mitigation

- Virtualized network does not properly emulate real signals being sent in passenger vehicles.
  - 0.5%
- Physical network does not properly send and receive real signals that are present in passenger vehicles
  - 0.5%
- ECU simulator(s) are damaged and cannot send CAN bus data resembling passenger vehicles.
  - 0.5%
- ECU's (TCM, PCM, or the ECU's MCU) and sensors are damaged and cannot send CAN bus data resembling passenger vehicles.
  - 2.0%
- Product shipment takes an extended period of time or other issues preventing setting up hardware.
  - 3.0%
  - Mitigation: Order parts early to give maximum amount of time for delivery, find alternate methods of obtaining CAN bus data to use for attack vectors and IDS.
- Safety hazard when dealing with electronic components of ICS Testbed.
  - 0.5%
- New feature or requirement is necessary to complete the project that was not known prior to project start.
  - 0.5%
- IDS host can not properly analyze CAN traffic due to being set up for TCP/IP traffic
  - 3.0%
  - Look into other options for hosting our IDS or developing a different method for hosting IDS on the network

3.6 Personnel Effort Requirements

Task 1: Project Planning and Research for Requirements and Equipment

Task	Estimated Hours	Explanation
Research CAN bus documentation	8	Explore basics of CAN bus protocol and frame formatting. Understand structure of functional CAN channels. Research characteristics of both normal and malicious CAN traffic.
Research requirements for functional and practical IDS	6	Strengthen knowledge of uses and applications of intrusion detection systems. Determine estimates for accuracy and speed of IDS.
Gather prior design and implementation of an IDS on a CAN bus system	4	Gather references for design and opportunities for our IDS. Research methods to program and deploy IDS to the CAN environment.
Search for possible car parts in a nearby junkyard	6	Find possible equipment that can be used to create a physical testbed. Determine plausibility of creating a

		testbed out of car parts based on tool accessibility and prior skills. Ensure IDS can be deployed to this testbed.
Search for compatible ECU Simulators	6	Find possible equipment that can be used to create a virtual testbed. Ensure that acquired components are compatible with the components being used as the CAN channel and controller. Ensure IDS can be deployed to this testbed.

Task 2: Raspberry Pi Testbed Design and Implementation

Task	Estimated Hours	Explanation
Configure Raspberry Pi with PICAN2 CAN bus HAT	4	Attach PICAN HAT to the Raspberry Pi for joint use. Download and update all necessary software libraries required for components to properly interact. Set up proper configuration and authentication settings, including allowing remote access to the system.
Set up virtualized CAN channels using software libraries to emulate passenger vehicle use	6	Configure basic CAN channel on the PICAN2 HAT to allow nodes to connect to the virtual network. Utilize can-utils and cantools libraries for setup and configuration.
Link Arduino Pros to virtualized CAN network	6	Introduce first nodes to the CAN network. Implement code within the Arduinos to be able to emulate ECUs on the network. Send signals across the CAN channel using the newly designed nodes.
Link an ECU simulator to the testbed	6	Expand the scope of the current testbed by adding new nodes to the network. Introduce signals from the ECU simulator to broaden the type and structure of signals being sent along the CAN channel.
Hardware and Software Inventory	6	Continuously document all hardware and software being used in the testbeds. Log all changes being made to software settings and permissions. Allows for better visibility and reproducibility of the system.

Task 3: Raspberry Pi Testbed Validation Testing

Task	Estimated Hours	Explanation
Send CAN messages from the Raspberry Pi	4	Ensure that the CAN channel has been properly set up and configured by manually generating signals to be sent across the channel.

Send CAN messages from one Arduino	4	Verify that a node can successfully be added to the CAN channel and all frames generated can be properly logged and viewed.
Send CAN messages from multiple Arduinos	4	Confirm that by creating a CAN network with multiple nodes, that all messages are still being properly sent across the channel and no collisions occur causing problems with the protocol.
Send CAN messages from the ECU	4	Make certain that frames being created from the ECU node function on the network, similar to previous Arduinos and no errors arise from having different nodes on the network.

Task 4: Pontiac G6 Testbed Design and Implementation

Task	Estimated Hours	Explanation
Power and connect components to bus	12	This process will involve extensive labor of identifying, splicing, and connecting necessary components while also testing the circuits.
Set up monitoring device to monitor CAN messages	8	For our device, we will be implementing a teensy 4.0 and the set up will involve installing, and configuring drivers as well as connecting it to the network.
Set up man in the middle device	8	For our device, we will be implementing a second teensy 4.0 and the set up will involve installing, and configuring drivers as well as connecting it to the network.

Task 5: Pontiac G6 Testbed Validation Testing

Task	Estimated Hours	Explanation
Verify that messages are being logged by CAN monitor device	10	Connecting the device to the testbed network and to a laptop to verify that messages are being read by the device.
Verify that man in the middle device is able to inject/alter messages on the network	10	Connect both the MITM and monitor device to the network and inject messages with the MITM, if they show as being logged by the monitor device then it is working.

Task 6: CAN IDS Design and Implementation

Task	Estimated Hours	Explanation
System Configuration	10	Find most effective ways of monitoring traffic and decide what method of detection will be used within the

		IDS.
Rules and Alert Implementations	12	Integrate with CAN messages to teach IDS how to detect malicious signals. Implement alert mechanisms when an intrusion is detected.

Task 7: Integrate IDS into Pi and Car Testbeds

Task	Estimated Hours	Explanation
Assign appropriate network setting to IDS	8	Ensure that the IDS can properly identify the sender and receiver nodes for each CAN message. Prevent the IDS from interrupting the traffic across the CAN channel.
Configure the IDS to monitor CAN bus traffic specifically	6	Equip the IDS application with the ability to monitor a protocol like CAN bus. Apply the TCP/IP capabilities of the IDS application to a physical communication protocol.
Input any necessary CAN message formats or templates into the IDS	8	Teach the IDS about CAN packet formatting found in the testbeds. Instruct the IDS about the channel structure and its associated nodes.

Task 8: Functional Testing and Performance Evaluation

Task	Estimated Hours	Explanation
Network Testing	4	Validate that the CAN network is working properly and as expected.
Initialization Testing	4	Ensure the IDS starts up and initializes without errors
Configuration Testing	4	Verify that all settings within the IDS can be changed and combined and will still properly function.
Alert Mechanism Testing	4	Ensure that when an anomaly is detected, the alert mechanism works as intended.
Performance Evaluation	20	Assess the maximum amount of CAN bus traffic the IDS can handle without missing any alerts or causing delays. Monitor CPU, memory, and other resource utilizations under varying loads to ensure the IDS doesn't overwhelm the system. Measure the time taken for the IDS to detect an anomaly and generate an alert once a malicious message is introduced. Increase CAN bus traffic volume and complexity to see if the IDS scales effectively. Introduce faults or simulate system crashes and observe how quickly and effectively the IDS can recover.

Task 9: Attack/Defense Testing and Evaluation

Task	Estimated Hours	Explanation
Planning and Strategy Development	8	Define the scope, objectives, and desired outcomes for the cybersecurity testing. Determine which attack vectors and scenarios will be simulated. Ensure a safe testing environment to avoid unintended disruptions.
Establishing Baselines	6	Monitor and record typical benign CAN bus traffic to understand normal patterns. Configure the IDS with current settings and definitions.
Attack Simulation	12	Simulate well known attacks on the CAN network. Try lesser known attack techniques on the CAN network. Launch prolonged attacks to test the endurance of the IDS. Introduce subtle, slow, or low volume malicious activities to check the sensitivity of the IDS.
False Positive Evaluation	6	Inject benign traffic variations that could potentially be misconstrued as malicious. Monitor and record instances where the IDS incorrectly flags these as threats.
False Negative Evaluation	6	Analyze the IDS logs and compare against the introduced attacks. Identify instances where the IDS failed to detect and alert on genuine intrusions.
Impact and Effectiveness Evaluation	6	Determine how quickly the IDS responds to threats. Assess the accuracy of the IDS's alerts and logs. Evaluate the IDS's resource consumption during attack simulations.
Recovery and Resilience Testing	6	After simulating attacks, test how well the system recovers. Determine if the IDS maintains its integrity or requires resets/configuration adjustments after attacks.
Iterative Adjustments and Retesting	12	Tweak IDS configurations, signatures, or settings based on all testing outcomes. Re-run specific attack scenarios to see if detection and response improve.

3.7 Other Resource Requirements

Technical documentation of CAN bus, any existing research/solutions/technologies that may be or perform as an IDS for CAN bus, personnel well versed in the automotive industry that does cyber security on CAN bus.

4 Design

4.1 Design Content

Our project consists of the implementation of an intrusion detection system on CAN bus networks observed within passenger vehicles. Our design is to be carried out within three stages; designing a testbed for a Raspberry Pi CAN Bus Network, designing a testbed for a car CAN bus network emulating vehicle behavior, and finally deploying the IDS onto both CAN bus networks. By having multiple systems to test our IDS on, we will have more options for testing different attacks and ensuring the IDS will work on many different CAN network environments.

4.2 Design Complexity

Our designs incorporate multiple components used together in order to create a network or a system.

For our first testbed, we will use a Raspberry Pi and a PiCAN Hat 2 in order to create a bus channel. The CAN bus protocol is widely accepted and used in almost all vehicles and many machines, so this testbed keeps up with industry standards. We are going to use Arduino UNO's in order to create multiple nodes to adeptly simulate what the network inside a car would look like. The engineering principles applied with this will be network integration, as we need to ensure each component of our system can communicate with each other, and that that communication can be viewed as well. This will involve both software and hardware troubleshooting.

For our second testbed, we will be purchasing parts from a vehicle. The vehicle will have many ECUs connecting to many different components. These components will be taken out of the vehicle and reconnected in a smaller network outside of the vehicle. To power this network we will use a 12 volt power supply. We will use a CAN sniffer on the network to monitor the traffic and share that data with the IDS. Another device will be connected to the network with the ability to send, receive, and modify CAN messages. This MITM device will be used to run attacks on the network which will alert the IDS of possible malicious activity.

For the IDS, we plan to implement an open source intrusion detection system, Snort. Snort rules will have to be set for known malicious behavior as well as the use of machine learning to catch malicious behavior that has not yet been observed. An issue we may face is to integrate Snort into our testbed networks. Snort is based on TCP/IP traffic and CAN bus is not. If we are unable to make this work we will pivot into another open source tool or we will build our own.

4.3 Modern Engineering Tools

Python - The main structure of our Intrusion Detection System (IDS) will be programmed using Python. We will be utilizing libraries like cantools and can-utils in order to collect CAN messages and analyze them for potential intrusions.

C - We will utilize C for the Arduinos in order for them to emulate electronic control units (ECU) and create CAN channels that accurately represent the channels used in passenger vehicles.

Wireshark - We will use Wireshark as a tool to capture and monitor CAN frames. This will allow us to see the traffic we are generating and observe frames we inject into our systems.

Git - This will be our primary repository for sharing code and progress throughout our project. We will be creating issues to track progress as we work through the project and keep our code organized.

Arduino UNO - We will use these components to be nodes on our CAN network to send signals along the channel to other Arduinos and the Raspberry Pi.

ECUSIM2000 - This will be used in our testbed to simulate ECUs used in cars in our CAN network.

Raspberry Pi - We will program one of these to be the main node and controller in our testbed.

CAN Transceiver - This small board will be used on top of other PCBs to give them the functionality of sending and receiving CAN messages.

Car - A car will be stripped to make our second testbed, ECUs, wiring, and components will be taken from the car and made into a smaller CAN bus network

Figma - This tool will be used to design the layout of the GUI for the IDS

4.4 Design Context

Area	**Description**	**Examples**
Public health, safety, and welfare	Our project works towards ensuring the safe usage of passenger vehicles by detecting and alerting operators of potential intrusions occurring. Problems that result from not secure CAN bus channels could cause harm to passenger vehicle users.	Detecting possible hacks into the CAN channels of passenger vehicles causing the vehicle to not function correctly. Denial of Service attack.
Global, cultural, and social	Globally, most individuals commute to their workplace or school using an automotive vehicle whether that is passenger vehicle, or bus transit many rely on cars to live. An IDS would assure malicious actors would not leave people without reliable transportation.	In 2021, kia challenge trend caused an uptick of 19% in car theft where on average of 17 Kias and Hyundais were stolen every day in Columbus due to an exploit where malicious actors use a usb cable to start the vehicle.
Environmental	Testing the IDS may result in damage to the testbed or cause faulty parts in which replacement parts would be required that would have had a negative impact on the environment. We lead a carbon neutral footprint in utilizing recycled material.	Replacement parts require the processing of raw materials, usage of fossil fuels for shipment so we thought it is in our best interest to utilize parts from salvaged vehicles in the area or borrowed from the ETG.
Economic	IDS would enact a positive measure in assuring their product's reliability as well as the customer's privacy and safety.	Companies are greatly impacted by the financial losses experienced in recalls and security breaches so there would be a huge selling market to manufacturers in being the only of its kind on the market.

4.5 Prior Work/Solutions

We have inspiration for both of our testbeds we are designing to send CAN messages. One was also utilizing a Raspberry Pi and Arduinos as the nodes on the network to create the environment. This project will be used as a baseline for our simple network, however, we will be using the PICAN2 Hat for our CAN channel which the other project did not use.

That testbed using the car parts had previously been seen at car hacking competitions, which provided us proof that this method of creating a CAN network would be possible. We have been in contact with the creators of the competitions unit for advice and questions about how we could use the car parts for our own project.

There have been many different intrusion detection systems created and designed in the cyber security world. Our differs from most IDSs in that types of signals we are analyzing from the network, while most analyze TCP/IP traffic, our deals with a completely different communication protocol. There have been some previously designed CAN intrusion detection systems, but these are not the most complete and do not provide the best documentation.

4.6 Design Decisions

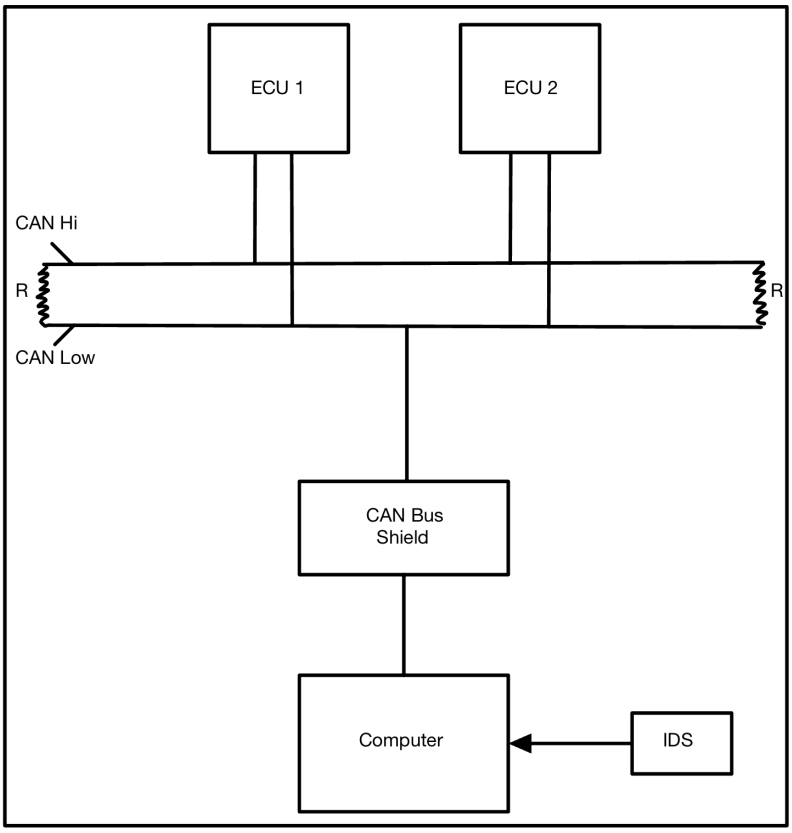
For the Raspberry Pi testbed, we were initially going to implement a system using MCP2515 transceivers in order to initialize the CAN bus channel. We would need one transceiver for each node in the system. Instead, we chose to use the PiCAN2 Hat in order to create the bus channel, allowing for easier integration and scaling.

We have also chosen to utilize Python in order to program our IDS because of the several libraries that are useful when dealing with CAN bus. These libraries will be useful for reading signals from the testbed and injecting signals into the channel. Having these tools will allow for our IDS to easily obtain data and analyze it while also allowing easier testing with the injecting of messages.

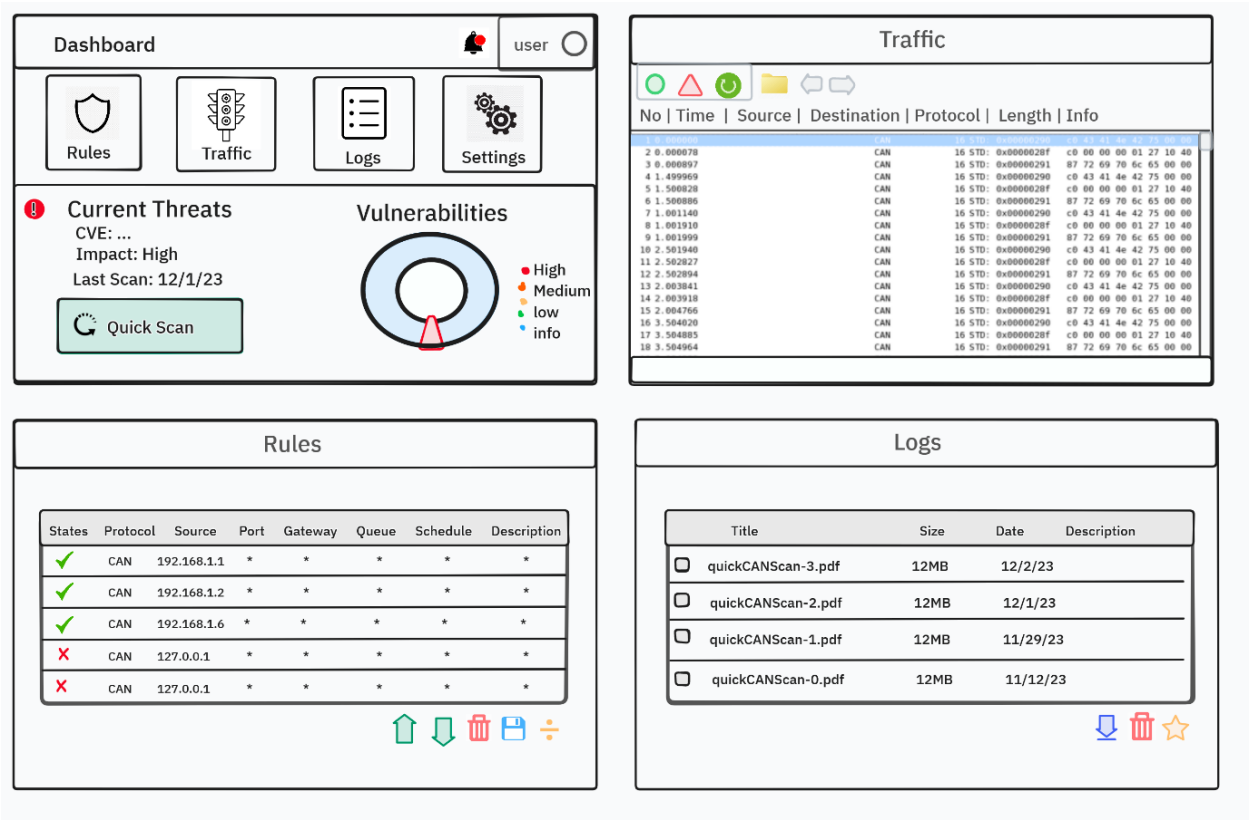
Lastly, for the second testbed, the choice to pursue the purchase of car parts was a difficult decision and at first we had decided against this idea. However, after speaking with multiple professionals in the field of vehicle security who have all built their own testbeds for the purposes of the career, we decided that pursuing the idea of using a car to build our testbed was an important step in this project. A testbed built from a car is not only the most real world testbed that we could have, but it is also more economical and easier to build a testbed from a car. The ECUs, components, and wiring are already built specifically for each other. The wiring is already designed. There is no necessary programming of ECUs or components, because they are already programmed. There is only initializing and monitoring the network.

4.7 Proposed Design

4.7.1 Design 0 (Initial Design)



This is the basic idea of which components on both testbeds will need to have and how they will have to be connected to each other. Beginning at the top, we will start with microcontroller units that will be used to simulate nodes in the network (ECU's in a car's control system). They will be attached to a CAN bus channel, having one connection to the CAN high wire and one to the CAN low. There will also be two resistors on each end of the wires, fully creating the channel. This channel architecture will be created by a CAN bus shield (MiTM device replaces CAN bus shield in car testbed), which in turn will be attached to a computer of some kind. On this computer is where the Intrusion Detection System will be deployed. The components being used for each testbed will vary, but all of the pieces will be included as one of the basic components listed in this diagram.



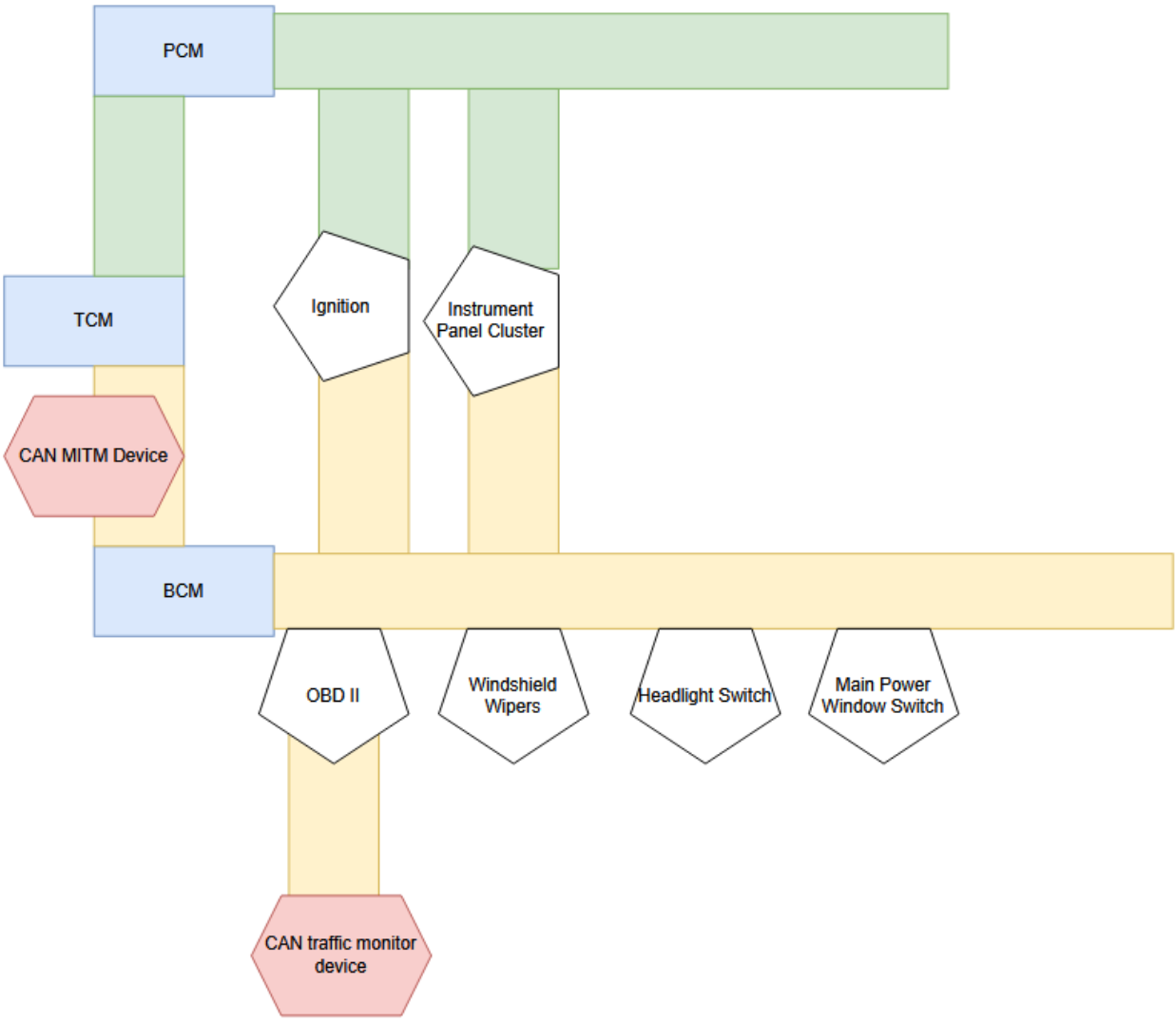
This is a simplified design of what the GUI of the IDS may look like and highlights features required of the system. Some of the features that will be present within our IDS, are areas where the current rules can be viewed, enabled or disabled, and changed. The user will also have a section where the logs of the signals that the IDS analyzes can be monitored and filtered. The interface will require a section for configurability where administrators will have the ability to change the settings the IDS runs on. All of these areas will fulfill requirements requested of our intrusion detection system including configurability, authentication, and data collection.

Functionality

All vehicles come equipped with an instrument cluster that features a set of warning lights that are used to alert users of a component needing service such as the gas is running low to provide the driver live updates of the conditions in which the vehicle operates. Modern cars that come equipped with a multimedia unit or digital display that will further detail the issue persisting while also providing insight on how soon the issue needs to be addressed (i.e., oil life remaining, miles until empty) but often in older vehicles for indicators such as a check engine light you need a diagnostic tool to scan the ODB II port to find out. This has been taken into consideration as we wanted to provide a solution that enables users the same luxury of modern vehicles in receiving live updates with a graphic interface detailing the issue persisting. Our IDS would come preloaded onto a small device that connects directly by listening to the CAN bus traffic and notify users of any malicious traffic or attempts of CAN based attacks. Beyond our current implementation this design has the potential in being a device that connects to your vehicle's OBD II port with a mobile app enabled to send

4.7.2 Design 1 (Design Iteration)

This iteration was brought about by the ambition of our group. Although it may be possible to complete this project without a real non-simulated test bed, we have found it to be economically more efficient, assembly-friendly, and realistic. With this design we are using real CAN traffic, on a real production vehicle that is in use by consumers today. This design allows our project to be overall much more impactful to the community.



This is a high level view of the testbed network that will be built from the 2007 Pontiac G6. The CAN traffic monitoring device will be connected to the OBDII port, OBDII is a commonly used connection port for diagnostics tools in vehicles. This monitoring device will be built from a teensy 4.0 chip; the monitoring device will capture traffic from this network and forward it to the IDS that will be deployed on it. The CAN MiTM (man-in-the-middle) device will be a teensy 4.0, the same as the CAN monitoring device, it will have the ability to alter messages on the network in real time, as well as simply injecting CAN messages into the network. This will function as our threat actor. The windshield wipers, headlights switch, and main power window switch are all things that require a driver to interact physically with them, when this is done the windows will open/close, the headlights indicator, or turn indicator will appear on

the instrument panel cluster. Changes made by the powertrain control module (a.k.a ECU, ECM) will appear on the instrument panel cluster as well. This would include things such as oil temperature, RPMs, MPH, or anti-brake locking system indicator.

4.8 Technology Considerations

While deciding on how we could use our IDS and test its uses and effectiveness, we had to consider the different options and hardware that could properly demonstrate the signals being sent around a CAN channel in passenger vehicles. Some options would be very difficult to implement, but more accurate, while others were easier to build, but the messages were further from actual signals. Ultimately, we came to the conclusion that we could do both of those options. With other projects creating a simple CAN network using a Raspberry Pi and Arduinos, we could develop a similar system to test the basics of our IDS on simpler communication methods. This would be accompanied by a testbed using components from a real car, and while more difficult, we believe that with assistance from people who have developed similar systems, we could also create a network with these components. This would allow more accurate messages to be sent along with more real world attacks being introduced and would let us build up our IDS after making it work with the messages and attacks on the simpler testbed.

4.9 Design Analysis

So far, we have been able to prove that we can make the Raspberry Pi testbed functional for sending CAN messages across the channel. Within the Pi and using the PICAN2 Hat, we have already been able to create the CAN channel, can0, and send signals along the channel. Just as important as creating the channel, we have been able to log messages being sent which will be important for allowing the IDS to analyze the messages going across the channel. Some changes may be made as to what nodes and the combination may be used, but they will all send CAN frames across can0.

5 Testing

For our testing strategy, we have 6 main sections. These include Unit, Interface, Integration, System, Regression, and Acceptance testing. Unit testing will include specific test cases for each individual component, such as tools to simulate and analyze network traffic. As for interface testing, ensuring accurate CAN messages/frames are being sent and received between testbeds and their accompanying software is the main aspect of this section. Integration testing's main challenge will be confirming that the IDS can be deployed on both the Pi and the Car testbed, containing some portability. System testing is the combination of the first 3 stages, having a unique issue of continuous testing of each system as they are being individually scaled up, while keeping the integrity of the overall system. Keeping with this same idea, regression testing will be crucial and natural as we will need to ensure the function of previous components before and after adding new components to the 3 different systems. Lastly, we will go over all of this testing with our advisor and confirm that the pre-decided expectations and requirements were met for every layer of our testing, and the accuracy, latency, and speed of the two testbeds and the IDS are all up to standards.

5.1 Unit Testing

Our unit testing will focus primarily on the IDS itself but also on the software used for the CAN monitoring device and the CAN MITM device. The tests conducted on the monitoring and man-in-the middle device will target their functionality, to ensure they are performing as intended. This will include assessing communication and evaluating the ability to properly monitor, intercept, handle errors, and forward data without unexpected behavior or data loss. A variety of testing platforms and CAN bus

specific testing tools will be utilized, such as CANtact/CANable and SocketCAN to test how the devices interact with the network, and fuzzing tools to assess the response to unexpected input.

Unit testing the IDS will involve evaluating the accuracy and performance of the IDS's capabilities. This would be carried out in a series of steps by first generating traffic and observing how the IDS responds to different types of messages using tools such as CANtact/CANable and can-utils. Tools UDSim and CANSim will be used to simulate malicious traffic to test the capability of the IDS to recognize abnormalities and attacks. Finally we would simulate CAN bus attacks such as frame spoofing, or DoS using the car testbed to evaluate its response.

5.2 Interface Testing

As there are multiple testbeds and an IDS, there will need to be multiple interfaces tested. For the Raspberry Pi testbed, there will be two interfaces between the Pi/PiCAN hat and the ECUsim 2000; ScanMaster-ELM and ECUsim Master. The ScanMaster-ELM is what actually connects the ECUsim to the Pi, so testing for this will include ensuring that all of the ECU's that the ECUsim 2000 provides are accurately shown in the software. There are a lot of sensors in these ECU's, so it will be important to complete total tests where we check that every single sensor is outputting "real" data on the software, and each time we use the ECUsim we should pre-inspect each of the sensors that we are going to be utilizing for the session to confirm that the data is not corrupted. This procedure will be similar to how the interface testing for the actual vehicle testbed will go. Each of the parts, like the TCM, PCM, etc, will be initially tested and have their function be confirmed on a laptop or a MITM device. After, only the components that will be used in a certain intrusion testing session will be evaluated and double checked, ensuring the CAN frames are being accurately sent and received. Since the integrity of the CAN messages/frames will already be established by the time it gets to the IDS interface, the only testing necessary will be testing if the messages are malicious. This is the purpose of our IDS, so this part will be constantly running and evaluated.

5.3 Integration Testing

The most critical integration in our design is ensuring that our IDS can properly be implemented into our testbeds. The most important aspect that will need to be tested is that our IDS can properly detect any intrusion found in our testbeds. To properly confirm that the IDS has been integrated into our testbeds, we will inject attacks into the system and monitor the accuracy of the detection system when analyzing the frames. It is also necessary to show that the IDS runs quickly once it has been integrated. This can be tested by having the nodes on the network send signals and checking how fast the IDS is analyzing all the frames being sent across the channel.

5.4 System Testing

To ensure the robustness of our CAN bus IDS, we implement a multi-layered testing strategy tightly coupled with system requirements. Unit testing is conducted using the Unity framework to validate the functionality of individual pieces of embedded system code. Tools like CANalyzer and CANTools facilitate interface testing, sending both well-formed and malformed packets to assess the resilience and correctness of communication within our testbeds. Integration testing is performed in an incremental fashion; as we introduce new components to our virtualized and physical networks, we can efficiently isolate and address issues. At the system level, we conduct comprehensive testing by monitoring network operations under standard conditions, verifying that the testbeds function seamlessly and the IDS accurately logs traffic in real-time. Additionally, we execute controlled attack scenarios on both networks to evaluate the effectiveness of the IDS in detecting and responding to threats promptly. This approach allows for a thorough validation of the entire system's reliability and security performance.

5.5 Regression Testing

Version control for software is in place with Gitlab and peer reviews will be done to ensure accurate coding practices. We have two CAN bus networks that will be backups for each other. Design decisions are not made by any one person. We have extensive documentation on what the design of the testbed networks will look like. Benchmark tests will be done on the Intrusion Detection System to ensure a 90% level of accuracy in reporting is met and continues to be met as the project progresses.

5.6 Acceptance Testing

Our acceptance testing will need to involve both testbeds and the IDS to ensure they meet requirements. We will test our testbeds to make sure that the signals being sent through the channel meet what the requirements state by ensuring accuracy and speed of packets being sent. The IDS will undergo tests from above that check accuracy, latency, and speed as a few examples, and the specific threshold that each of these aspects needs to meet will be decided by our advisor.

5.7 Results

At this time the project is not in a stage suitable for testing so our results are up to speculation. Nonetheless, we aim to ensure that each component, IDS functions correctly and meets specified requirements. Results are expected to confirm functionality, detect errors, assess performance metrics, ensure system robustness, and satisfy acceptance criteria. Compliance with requirements is achieved by addressing detected issues, maintaining stability through regression testing, and aligning results with project documentation.

6 Implementation

Implementation of the testbed will involve building and programming many small pieces that will come together to complete this project. The CAN monitoring device and our CAN MITM device will be built with a Teensy 4.0 as the board. The board will have two MCP 2562 CAN transceivers, and a female DB9 port soldered onto the board. A male DB9 port connects to the board and will have two wires the CAN high and CAN low connecting to the testbed network on the other side. Both the CAN monitor and the CAN MITM will be able to connect to any computer through a USB mini connection on the board to a USB A connection to a computer. The CAN network will be built from stripped electronics of a 3.5 L V6 automatic 2007 Pontiac G6. This includes the Engine Control Module (ECM), Transmission Control Module (TCM), Body Control Module (BCM), engine fuse box, instrument panel cluster, main power window switch, headlights switch, turn indicator switch, steering column, ignition switch, engine wiring harness, and body wiring harness. All of these components and control modules will be connected with the wiring harnesses and it will be powered by a 12 volt power supply.

The IDS will be built using the open-source tools Security Onion and Snort. Security Onion is a Linux distribution for intrusion detection, network security monitoring, and log management. It contains a suite of tools that work in unison to monitor network traffic and alert users of potential threats. Snort, on the other hand, is a widely used open-source network intrusion prevention and detection system (NIDS/NIPS) that is capable of performing real-time traffic analysis and packet logging on IP networks. In this project, Snort will be configured to analyze the network traffic flowing through the CAN network, identifying suspicious patterns that could indicate a potential intrusion. To ensure effective detection capabilities, custom Snort rules will be developed based on the typical communication patterns and messages of the CAN network. These rules will help in identifying anomalies that deviate from normal operations, such as unusual message sizes, unexpected request rates, or unauthorized access attempts.

The system will be tested with various known attack scenarios, such as message injection, flooding, spoofing, and fuzzing to validate the effectiveness of the intrusion detection setup.

7 Professionalism

7.1 Areas of Responsibility

Area of Responsibility	IEEE Code of Ethics	NPSE Code of Ethics
Safety, Health, and Welfare	To hold safety, health, and welfare of our team members and the public as predominant standard	Engineers shall hold paramount the safety, health, and welfare of the public in the performance of their professional duties.
Improve understanding	To use our project to improve upon the current technical understanding	Engineers shall perform services only in the areas of their competence and continue their professional development to stay current in their field.
Avoid conflicts of interest	Avoid any conflicts of interest and bring them to attention if they should occur	Engineers must act for their employers or clients as faithful agents or trustees and avoid conflicts of interest.
Avoid unlawful conduct and reject bribery	Do not use our work to do anything illegal and to always refuse bribes to do so	Engineers must avoid deceptive acts and conduct themselves honorably, ethically, and lawfully.
Maintain and improve technical competence	Work to improve and maintain the technical competence that currently exists	Engineers should seek to advance the integrity and prestige of the profession and support the professional development of their colleagues.
Treat all persons fairly and with respect	To treat all people with respect and fairness	Engineers shall treat all persons with dignity, respect, fairness, and without discrimination.
To not bully or sexually harass	We have a duty to never bully or sexually harass anyone	Engineers shall build their professional reputation on the merit of their services and not resort to unfair or improper solicitation of their professional employment.
To avoid injuring others, their property, or their reputation	We have a duty to avoid injuring others, their property, or their reputation	Engineers shall avoid all conduct or practice that deceives the public and are responsible for

		their actions towards others' property and reputation.
Strive to uphold the code, support those who uphold the code, and to ensure that retaliation on those who report never happens	We must always strive to uphold these ideals, and support others in upholding these ideals, and to always support those who report violations of this code.	Engineers shall uphold and enhance the honor, integrity, and dignity of the engineering profession.

7.2 Project Specific Professional Responsibility Areas

Public Health, Safety, and Welfare:

- Highly applicable. The primary aim of the project is to enhance the safety of passenger vehicles by preventing cybersecurity threats. Ensuring the secure functioning of vehicles directly impacts public health and safety.
- High performance. The project directly contributes to vehicle safety by detecting potential intrusions, which is crucial for the well-being of passengers and pedestrians.

Global, Cultural, and Social:

- Highly applicable. Transportation is a global need, and ensuring the security of vehicles is vital across cultures and societies. Our project has a direct impact on the reliability of a universally used mode of transport.
- Medium performance. While the project addresses a global need, the extent to which it considers varied cultural and social contexts in its design and implementation could be further explored.

Environmental:

- Moderately applicable. While the primary focus is on cyber security, the project's approach to using recycled materials and salvaged parts shows an awareness of environmental responsibility.
- Medium performance. The team is making efforts to minimize environmental impact through sustainable practices, but this is not the main focus of the project.

Economic:

- Highly applicable. This project has significant economic implications, both in terms of potential market value and in preventing economic loss due to vehicle theft or damage from cyber attacks.
- High performance. This project addresses an important economic concern for vehicle manufacturers and owners, showing a high degree of professional responsibility in this area.

7.3 Most Applicable Professional Responsibility Area

The most applicable professional area for the project focused on building an Intrusion Detection System (IDS) for automotive CAN bus systems is unequivocally "Public Health, Safety, and Welfare." This project's core objective is to fortify the safety of passenger vehicles by proactively identifying and mitigating cybersecurity threats, a goal that aligns directly with the imperative of ensuring public safety in the context of automotive transportation. By concentrating on securing the CAN bus systems, which are integral to the functioning of modern vehicles, the project addresses a critical vulnerability that could, if

exploited, lead to dire consequences for vehicle occupants and pedestrians alike. This focus on cyber security in automotive systems is not just a matter of technological advancement but a crucial safeguard for public health and safety. Vehicles are an everyday part of life, and any compromise in their electronic systems could lead to catastrophic failures, posing significant risks to public safety.

8 Closing Material

8.1 Discussion

The results of our project will be a proper intrusion detection system that can be placed within a CAN bus network in order to quickly and accurately find any intrusions that may occur within the protocol. Our work with designing testbeds that can accurately resemble how CAN bus networks communicate and creating packets that electronic control units in passenger vehicles would actually send, allows the IDS to be accurate because it was tested on these systems. We will have trained and taught the IDS to be able to detect several different types of CAN bus attacks that are commonly used when attacking the protocol, these attacks will be tested through the use of both testbeds we will have prepared.

8.2 Conclusion

With our plan for completing this project complete and a head start in the design of both testbeds, we can look forward to our goals and how they will be achieved. With the parts and components of our two testbeds in hand, our short term goal is to get the basics of a CAN network established by connecting single nodes to the channels along with the controller, so that we may begin sending signals between devices. This first goal will also allow us to include the IDS on the controller and monitor the messages the single node is sending, thus providing the data to begin IDS development. Beyond that we will expand each network with more nodes and begin introducing attacks to these networks for IDS testing and training, this will be followed by making adjustments to the systems configuration and rules to improve the accuracy and speed to fit our requirements. Ultimately, our work in creating real vehicle emulating testbeds will result in an intrusion detection system that will correctly assess if threats have entered the system.

8.3 References

[1] J. Staggs, "How to Hack Your Mini Cooper: Reverse Engineering CAN Messages on Passenger Automobiles," University of Tulsa, Institute for Information Security, Crash Reconstruction Research Consortium, [Online]. Available: https://www.engr.colostate.edu/~jdaily/tucrrc/CANClock/DEFCON21_Staggs_Paper.pdf

[2] K. Lade and R. J, "CANalyse 2.0: A vehicle network analysis and attack tool," presented at DEF CON 30, Aug. 12, 2022, [Online]. Available: https://youtu.be/Py_1I-GtUxw Accessed on: Oct. 1, 2023.

[3] I. Tabor, "From an 'IVI in a box' to a 'CAR in a box'," presented at ROOTCON 2020, [Online]. Available: https://www.youtube.com/watch?v=4Ptk9ikfpAQ Accessed on: Sep. 21, 2023.

[4] SK Pang Electronics Ltd, "Raspberry Pi CAN2 HAT User Guide," 2016. [Online]. Available: https://raspberry-valley.azurewebsites.net/ref/Raspberry-Pi-PICAN2-Hat-User-Guide.pdf. [Accessed: Insert date accessed].

[5] S. Bhattacharya, "CPRE 558 Project Final Report," Iowa State University, 2023.

8.4 Team Contract

Team Name sdmay24-39

Team Members:
1) Cole Burkle
2) Alec Cose
3) Tiffanie Fix
4) Trace Haage

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings: Two meetings per week, one which includes advisor and graduate student, one with only senior design students

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e mail, phone, app, face-to-face): Discord, email

3. Decision-making policy (e.g., consensus, majority vote): majority vote & advisor veto

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived): Rotate note taking of meetings, notes posted on git & discord

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings: No unexcused absences.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines: Equal levels of responsibility between team members.

3. Expected level of communication with other team members: Keep team members updated on progress through discord and weekly updates/reports. Notify the team of meetings that member will be absent for.

4. Expected level of commitment to team decisions and tasks: 100% committed

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.): All team decisions will be made through majority vote with adviser's input and final decision.

2. Strategies for supporting and guiding the work of all team members: If one is struggling with their assigned work they will ask others for help. If someone already knows/ has experience with something they should take the lead on a given task.

3. Strategies for recognizing the contributions of all team members: Tasks completed and level of

difficulty

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. Cole Burkle
        i. Worked with CAN bus, cyber security research, programming in C, Java, and Python
    b. Tiffanie Fix
        i. Cyber Threat intelligence and Incident Responder for IBM XForce, researched zero-day exploits and detection on MFTs presented at Blackhat c.
    Alec Cose
        i. Programming experience in Python, Java, and C, IDS experience
    d. Trace Haage
        i. Python, Java, C. Worked with control systems so I know Modbus, Bacnet, and SCADA software like Ignition.
2. Strategies for encouraging and supporting contributions and ideas from all team members: Daily affirmations, don't be mean, be kind, and understanding when a team member is struggling
3. Procedures for identifying and resolving collaboration or inclusion issues: Let the team know if they are having issues with someone or the current team environment.

Goal-Setting, Planning, and Execution

1. Team goals for this semester: Plan out project and set the team up for success
2. Strategies for planning and assigning individual and team work: Gitlab and Discord
3. Strategies for keeping on task: Reporting progress and logging missed or late deadlines

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
Communicate struggles of team members infractions with class professor and advisor
2. What will your team do if the infractions continue? Discuss with the class professor and advisor about possibilities or removing member from the team.