SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Student CyberTruck Experience Manual

Jeremy Daily

May 21, 2020

A special thank you goes out to Urban Jonson at the National Motor Freight Traffic Association, Inc. (NMFTA) for being a champion of the program. His passion for cybersecurity and education was insiprational in creating the content and program called the Student CyberTruck Experience. The Executive Director the NMFTA, Paul Levine, deserves special thanks as he provided resources and guidance on bringing the Student CyberTruck Experience to life. Our initial sponsors at the University of Tulsa include the NMFTA, Geotab, and PeopleNet. The continued support from Geotab through Ryan Brander and Glenn Atkinson are gratefully appreciated.

Contents

Introduction to the Student CyberTruck Experience

In 2016, Urban Jonson of the National Motor Freight Traffic Association, Inc. () reached out to Dr. Jeremy Daily while he was teaching mechanical engineering at the Univeristy of Tulsa () to disuss some research findings related to heavy vehicle cybersecurity. In this conversation, Dr. Daily confirmed many of the hypotheses in the NNFTA whitepaper on the state of heavy vehicle cybersecurity. This converstation led to an invitation for Dr. Daily to attend the firs NMFTA Heavy Vehicle Cyber Security () meeting in Virginia. One of the results of the meeting was a recognition for the need to build the human talent needed to address the challenges assoicated with cybersecurity of heavy vehicles. This initiative is how the Student CyberTruck Experience came into being.

The student Cyber Truck Experience (CyTeX) is a crash course in heavy vehicle networks and cybersecurity with a hands-on approach. It is intended to help a new engineering student with little experience with vehicle systems or cybersecurity develop the skills necessary to work effectively in heavy vehicle cybersecurity. This student handbook provides some reference material to broaden ones understanding of the electrical, computer, and mechanical engineering involved in making and securing these vehicles. It also aims at providing students with the industry background to jumpstart a successful career.

Throughout the handbook there will be several instances which you will be asked to download software. The ideal OS for completion of this lab will be Windows; however, at times, a Linux operating system may be required. In this case, you will be walked through how to install an Ubuntu VM onto your machine before proceeding.

Thank you for taking taking an interest in HV and for utilizing this handbook. This content has been created by students for students at the University of Tulsa to follow during the Student CyberTruck Experience Program and other heavy vehicle cybersecurity programs.

Learning Outcomes and Program Mission

To develop the talent necessary to improve the cybersecurity posture of the heavy vehicle.

Program Objectives

The Taxonomy of Educational Objectives (Bloom's Taxonomy)

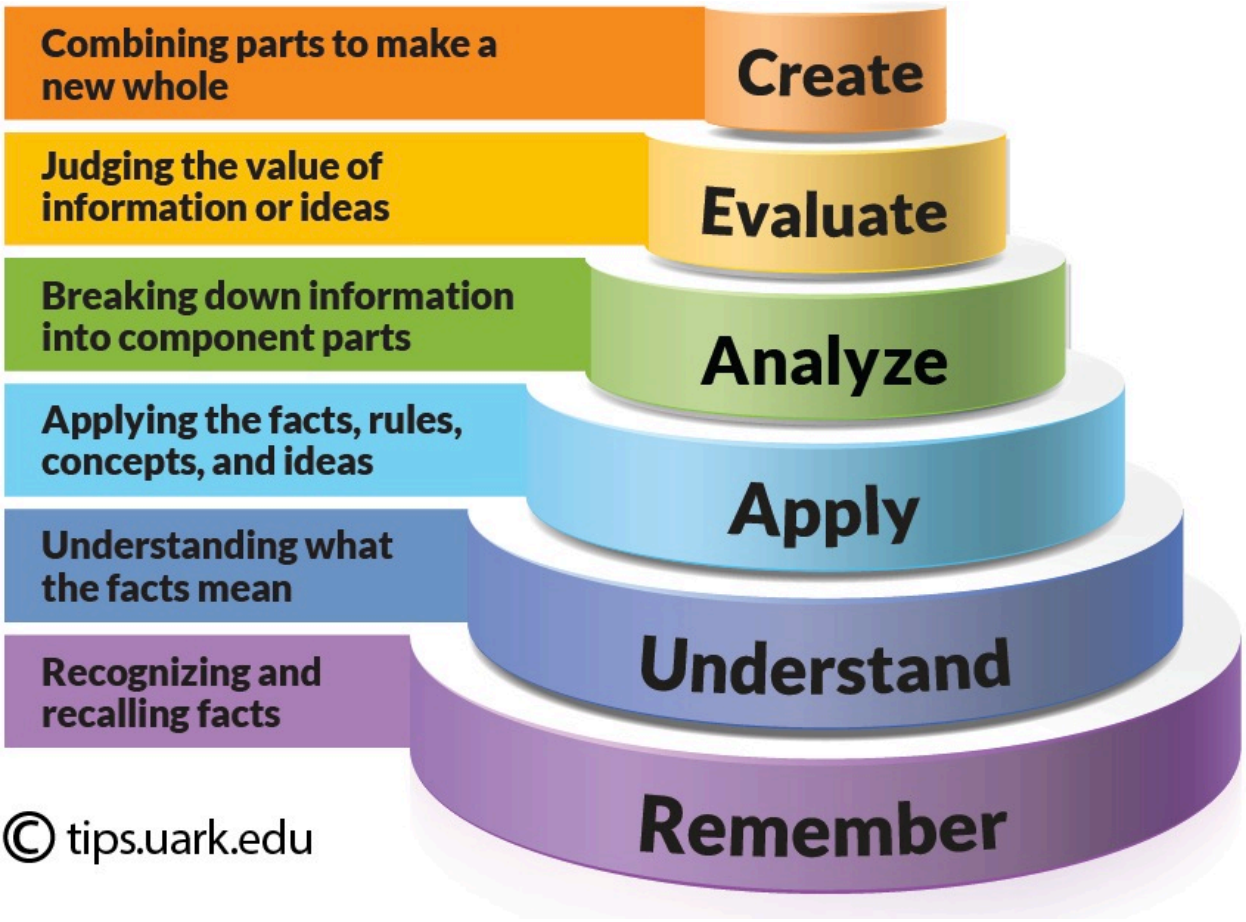

Figure 0.0.1.: Bloom's Taxonomy from Jessica Shabatura.

Exercises and Challenges

David & Ben's Idea: Exercises pertain mainly to their home chapter and rarely expand beyond chapter borders. However, the challenges at the end are the classic "challenges" which build upon eachother. For example a challenge might be: Graph the speed of a can dump using matplotlib. This combines truck system knowledge (Wheel Based Vehicle Speed for Vehicle Speed), CAN Protocol knowledge (PGN, Reverse Byte order, data field. ID field), and python knowledge (parse & graph). If you know all three concepts, the challenge is quite doable. However, if you are missing any of the three, the challenge becomes much more difficult. The proposition:

Truck System Knowledge Exercise: What sensors indicate vehicle speed to the driver?

CAN Protocol Knowledge Exercise: Given a single CAN message, how fast is this vehicle going? (They have to go from the ID to the PGN)

Python 3 Knowledge Exercise: Given this dataset of a string, csv, etc. plot the ____ data versus time.

After doing all these exercises, the challenge becomes much more doable. The important thing is that these exercises have established those three key skills that can be translated to research later on. At the end of certain sections of the manual, the learner is told "You now have the requisite skills to do Challenge X". So after learning CAN in chapter 7 of this iteration, the learner is told that they can do the above exercise.

The exercises are meant to be "sub challenges" that tie into the main summer challenges later on. Given this is a crash course, and by mid summer the learner should be working on actual projects, this manual is meant to be completed in 6 weeks. With ~12 chapters, students should accomplish about 2-3 chapters per week.

Potential Course Pace (super rough outline that I came up with in 15 minutes, might be too fast, might be too slow)

Week 1: Introduction to Trucking, Overview of Heavy Vehicles, possibly begin Electronic Hardware

Week 2: Arduino hardware (plugging in wires, making circuits, resistors, leds, uploading firmware, flashing beaglebones), beginning programming (Arduino code)

Week 3: Arduino Code Continue, Python Code Basics

Week 4: UNIX/Linux (BeagleBone operation), Networking Fundamentals

Week 5: Vehicle Networking Protocols (&Basic Cybersecurity): By the end of week 5, students should be able to competently work with truck data,

Week 6: Cybersecurity: After week 6, students should know enough to use encryption, CAN, arduino, and python. or at least be exposed to it

Week 7+: Specialization and reference chapters like Hardware Design using altium, Heavy Vehicle Forensics, Resources like ordering parts for the project CyberTruck happens around here (assuming starting 3rd week of May).

Exercises are meant to reflect this pace, with challenges meant to take a reasonable amount of time to complete.

Formative

Learning

Assessments are formative hands-on exercises where students

Summative

Cyber Challenges

Students are **strongly** encouraged to participate in the following practicum-based events related to vehicle cybersecurity that take place during the summer.

SAE CyberAuto Challenge

SAE CyberAuto Challenge is an event sponsored by the Society of Automotive Engineers that brings together students and engineers from various backgrounds to collaborate and seek new information on automotive cybersecurity. Registration is typically open until May. To register and for more information visit the website here.

CyberTruck Challenge

The CyberTruck Challenge is an annual event held in June. It is open to students and engineers and is focused on developing the next generation of heavy vehicle cyber professionals. To register and for more information visit the website here.

Research

Research is the top of the pyramid.

1. Introduction to Trucking

1.1. Objectives

What is the trucking industry?

What is a truck?

Who makes trucks?

Cybersecurity for Trucking

1.2. Welcome to the Trucking Industry

Your phone charger, favorite t-shirts and college textbooks all have one thing in common: they all took a trip at some point on one of the 2 million trucks in operation in the US today. In fact, it's difficult to image anything that won't take a ride on a commercial semi-trailer, construction vehicle, or city-service vehicle at some point in its lifespan.

In the absence of these vehicles, the ease of online ordering, walk-in shopping, local gas-stations and numerous other convenient aspects of our lives would not exist.

While a large scale threat to heavy-vehicle infrastructure may not be a present danger, the risk to small fleets is a growing topic within the industry. The threat? A cyber attack carried out as a result of insecure physical or digital systems used to disable one or a fleet of vehicles.

Throughout CyTeX, you will learn how clever engineers have hacked trucks as a means to better understand and hopefully mitigate vulnerabilities. Some techniques include hardware reverse engineering, radio frequency hacking, bus flooding, network data logging and much more. Of course, an understanding of most of the basics is needed before you can reach this point.

1.2.1. Terminology: 'The Truck Dictionary'

1.2.2. Companies and Organizations: 'The Players'

Academia

Ex: Colorado State University

Motor Carriers

Ex:

Vehicle Manufacturing

Ex: Peterbilt, Volvo, Thermo King

Freight and Shipping

Ex: Old Dominion Freight Line

Diagnostic Companies

Ex: DG technologies, Intrepid Control Systems, NEXIQ

Cybersecurity

Ex: Karamba

Government

Ex: NMFTA, DOT,

Telematics Companies

Ex: Omnitracs, Geotab, Samsara, CalAmp

Electronics and Hardware

: Bendix, Deutch

Other industry players include digital forensic companies, law enforcement, small fleets, secondary manufactures.

1.2.3. Regulators

Department of Transportation (DOT)

A federal Cabinet founded in 1966 to ensure the safest, most efficient, and modern transportation system for American workers and businesses. It is the highest governing body relating to transportation in the country and is based in Washington D.C. Notable administrations within the DOT related to HV are the FMCSA and the FTA.

The Federal Motor Carrier Safety Administration (FMCSA)

A DOT administration that regulates commercial drivers' licenses', oversees regulatory compliance and enforcement, provides safety assistance, and collection of motor carrier statistics.

National Motor Freight and Traffic Association, Inc. (NMFTA)

A nonprofit membership organization headquartered in Alexandria, Virginia. Its membership is comprised of motor carriers operating in interstate, intrastate and foreign commerce.

In addition to publishing the NMFC standards, NMFTA also hosts industry workshops exploring cyber security issues such as incident response, risk mitigation, data anomaly detection, and many others. These forums are attended by representatives from motor carriers, academic experts, government agencies, heavy vehicle manufacturers, telematics providers, cyber security firms, and other industry groups or associations.

Society of Automotive Engineers (SAE)

The global professional association for engineers and scientists working in various industries related to consumer automobiles, aerospace, and commercial vehicles as well as other forms of transportation. SAE produces many of the codes and standards for the heavy vehicle industry (i.e J1939 and J1708) to follow and is based the U.S.

Institute of Electrical and Electronics Engineers (IEEE)

The global professional association for educational and technical advancement of electrical and electronic engineering, telecommunications, computer engineering and other related disciplines. It is the world's largest association of technical professionals. In regards to the heavy vehicle industry, IEEE produces many of the standards and codes that make ECUs, sensors, bus communication, and wireless communication possible.

1.3. Introduction to Trucking Exercises

2. Overview of Heavy Vehicles

Weight Classes

2.1. Truck System Fundamentals (Truck Systems for Computer Scientists)

2.2. Electronic Control Units

An ECU is any embedded system in automotive electronics that controls one or more of the electrical systems or subsystems in a vehicle. For example, all commercial trucks have an Engine Control Module (ECM) that manages the everything from the ignition timing, air-fuel ratio, and emissions control. Other modules also exist such as the Transmission Control Module (TCM), Brake Control Module (BCM or EBCM), and Suspension Control Module (SCM). Other ECUs without common acronyms control the airbags, the dash board, power locks and more. When connected together these ECUs form an intricate data exchange system called the vehicle network.

2.3. Heavy Vehicle Exercises

3. Electronic Hardware

The purpose of this chapter is to give students skills necessary to work with modern electronics and build their own hardware tools.

3.1. Intro to Microcontrollers

A **microcontroller (MCU)** is a tiny computer that can run one program at a time, over and over again. More often than not, they are connected to sensors and actuators allowing them to listen and interact with the physical world.

Sensors convert physical phenomena such as wheel speed, fuel level, and oil temperature into electrical signals. **Actuators** convert electrical energy back into physical energy such as linear motion, light, heat, and motor torque.

MCUs listen to sensors and talk to actuators. They decide what to do based on the instructions you have written and stored to its memory. MCUs and the devices connected to them form the basis of the **Electronic Control Unit (ECU)**.

3.2. Arduino and Teensy

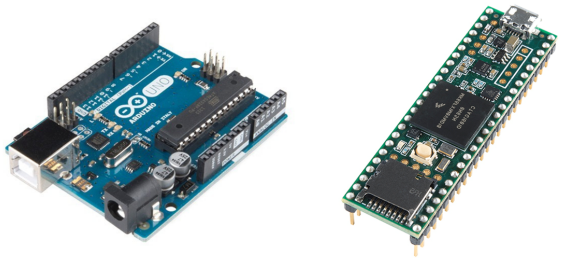

Figure 3.2.1.: The Arduino Uno (left) and Teensy 3.6 (right)

Arduino offers some of the most recognizable open-source microcontroller boards today. This includes the Arduino Uno. When it comes to prototyping, these boards have many advantages including: widespread availability, low cost, thorough documentation, and a global community of users and forums. The Arduino

Uno uses the ATMEGA328P, has 32 KB of flash memory and runs at 16 MHz which are acceptable hardware specifications for simple projects.

Programs are written and compiled in the Arduino IDE program, a special text-editor, and uploaded via USB. When the Arduino is connected to an external power source it will run the instructions you uploaded.

The Teensy is another USB-based microcontroller development system. Although it is slightly more expensive than the Arduino, the 3.2 and 3.6 versions are preferred within CyTeX for their additional capabilities and features. Most notably, they support more communication protocols such as CAN, I2C, SPI, and Ethernet. The Teensy 3.6 footprint is roughly a third the size of the UNO, the MK66 Processor, has 1024 kB of flash memory, and runs at 180 MHz.

All programming is done via the USB port and the Teensy Loader Application. It is run automatically when using Verify or Upload within the Arduino software. Teensy Loader is available for Windows, Mac, and Linux.

Full documentation for the Teensyand Arduino UNOcan be found on their respective websites.

3.3. Intro to Microcomputers

A single-board computer (SBC) is a complete computer built on a single circuit board, with microprocessor(s), memory, input/output (I/O) and other features required of a functional computer. You *could* call it a "micro-computer." SBCs have the ability to run multiple programs, run an operating system, and can often support a simplified version of a typical desktop GUI.

The Raspberry Pi by the Raspberry Pi Foundation in the UK is by far the most common prototyping SCB. For automotive applications, the Beaglebone Black by Texas Instruments and the NXP S32K have been used within CyTeX. Both of these devices are capable of more advanced functionality.

3.4. Recommended Hardware Kit

Each student should have access to the parts in the basic kit. These items are minimum to successfully accomplish the programming and learning exercises.

Qty	Label	Description	Supplier	Supplier Part Number
1	A	Teensy 4.0 Development Board	PJRC	TEENSY40_PINS[1]
2	B	MPC2562 CAN Transceiver	Digi-Key	MCP2562FD-E/P-ND[2]
1	C	Solderless Breadboard	Digi-Key	BKGS-830-ND[3]
5	D	120 Ohm Axial Resistors	Digi-Key	
1	E	Wiz850IO Ethernet Expansion Board	Digi-Key	1278-1043-ND[4]
1	F	30 Pack of Male-Male Breadboard Wires	Sparkfun	PRT-14284[5]
1		20 Pack of Male-Female Breadboard Wires	Sparkfun	PRT-12794[6]
1	G	Ethernet Cat6 Cable, 3 ft.	Sparkfun	CAB-08915[7]
1	H	USB micro Cable, 6 inch	Sparkfun	CAB-13244[8]
1	K	ATECC608A Crypto Authentication Module	Digi-Key	ATECC608A-SSHDA-TCT-ND[9]
1	L	ATECC608 Crypto Co-Processor Breakout	Sparkfun	SPX-15838[10]
1	M	Trimpot 10K Ohm with Knob	Sparkfun	COM-09806[11]
1	N	SOIC8 to DIP Converter Board	Sparkfun	BOB-13655[12]
1	P	Break Away Headers - Straight	Sparkfun	PRT-00116[13]
1	Q	Ambient Temperature Sensor	Sparkfun	SEN-14049[14]
1	R	Conductive 18 Compartment Organizer	Flambeau	C618[15]

Table 3.4.1.: Basic Hardware Kit

3.4.1. Basic Kit

Newer laptop computers may not have a physical Ethernet port, so a USB to Ethernet adapter may be necessary.

3.4.2. Basic Tools

Multimeter Any modern digital multimeter is acceptable to get started. There are many on Amazon that are suitable.

Soldering Iron A

Tweezers

[1] https://www.pjrc.com/store/teensy40_pins.html
[2] https://www.digikey.com/product-detail/en/microchip-technology/MCP2562FD-E-P/MCP2562FD-E-P-ND/4842807
[3] https://www.digikey.com/products/en?keywords=BKGS-830-ND
[4] https://www.digikey.com/products/en?keywords=Wiz%20850
[5] https://www.sparkfun.com/products/14284
[6] https://www.sparkfun.com/products/12795
[7] https://www.sparkfun.com/products/8915
[8] https://www.sparkfun.com/products/13244
[9] https://www.digikey.com/products/en?keywords=ATECC608A-SSHDA-TCT-ND
[10] https://www.sparkfun.com/products/15838
[11] https://www.sparkfun.com/products/9806
[12] https://www.sparkfun.com/products/13655
[13] https://www.sparkfun.com/products/116
[14] https://www.sparkfun.com/products/14049
[15] https://www.flambeaucases.com/18-compartment-box-1095.aspx

Qty	Description	Supplier	Supplier Part Number	URL
1	Teensy 4.1	PJRC	TEENSY41	[16]
1	Beagle Bone Black			
1	Quadrature Encoder with breakout board. Bournes PEC09			

Table 3.4.2.: Advanced Hardware Kit

Qty	Description	Supplier	Supplier Part Number	URL
1	Smart Sensor Simulator 2	DG Technologies	TEENSY41	[17]
1	Beagle Bone Black			
1	Quadrature Encoder with breakout board. Bournes PEC09			

Table 3.4.3.: Vehicle ECU Testing Kit

Wire Cutters

Wire Strippers

Carrying Tote

30 gauge Wire

3.4.3. Advanced Hardware Kits

3.5. Electronic Hardware Exercises

These exercises shouls serve to get you "up to speed" on Arduino hardware. Little to no computer programming experience is needed for this section, that comes later. However, you should know how to upload software in Arduino and view the console.

Exercise 3.1. Write a program to send CAN messages from one node to another.

Hints:

1. Check the continuity of each hookup wire before using it. They can be fragile and break easily.

3.5.1. Getting Started on Teensy Arduino (Stock Code that "should work")

3.5.2. Intro to CAN with Arduino (Move after CAN Section?)

3.5.3. Build a Breadboad Arduino (Move Challenge to Challenges Section)

Challenge: using the pieces below, send a CAN message.

[16] https://www.pjrc.com/store/teensy41.html
[17] https://www.pjrc.com/store/teensy41.html

Materials:

Breadboard Prototyping Wire ATmega328P MCP2551/MCP2562/MCP2558 MCP2515/MCP25625 LED 120ohm (2) or 60 ohm (1) resistor a 16MHz crystal 6 pin straight header a 10k resistor 18-22 pF capacitor (2)

Resources:

Use Datasheets to determine how to connect everything To find datasheets use google, digikey, and mouser Learning how to push code to the arduino: https://www.arduino.cc/en/Tutorial/ArduinoISP For help with SPI communication to MCP2515: http://avrbeginners.net/architecture/spi/spi.html

Challenge:

Send a CAN Message (Verify with oscilloscope) Decode the message from the oscilloscope to ensure that it is the mesage you told your arduino to send This can be done with Python Create an Altium Schematic of the layout that you created

4. Computer Programming Fundamentals (Computer Programming for Engineers)

Computer programming may seem like a daunting field for beginner programmers. It does not need to be. At its core, computer programming is providing a set of instructions to a computer. Each line of code contains one or more instructions. Sometimes instructions interfere with each other and cause errors. Imagine telling someone who has never seen a sandwich before how to make a peanut butter and jelly sandwich. The steps may be:

1. Get Bread, Peanut Butter, Jelly, Knife, and Plate.
2. Place plate.
3. Place two pieces of bread side by side on plate. Call them Slice A and Slice B.
4. Put peanut butter on knife.
5. Use knife to spread peanut butter on Slice A.
6. Clean knife.
7. Put jelly on knife.
8. Use knife to spread jelly on Slice B.
9. Combine Slice A with Slice B by placing Slice B on Slice A.
10. Eat Sandwich.
11. Clean Knife
12. Clean Plate

Provided no external changes, this set of instructions will always lead to a proper peanut butter and jelly sandwich. However, some people may argue about if you should use grape jelly or strawberry jelly, chunky or smooth peanut butter. Some people are adamant that you must put Jelly on Slice A, never peanut butter. Some argue that if one starts with a loaf of bread instead of slices then it produces a better sandwich. These options represent the different ways that people program. There are many different sets of instructions that produce peanut butter and jelly sandwiches. Programming is very smiliar. There are many different ways to code a program that will lead to the same result.

However, there are ways which are more efficient than others, and there are ways to make code easier to understand. When writing code, it is important to write with a consistent programming style that allows others to understand your code. Code must be maintained over time, and may transition between programmers.

This chapter should serve to introduce the basics of computer programming as a platform for the Cyber Truck Experience program.

4.1. Arduino Programming Language

4.2. Python 3 Programming Language

An important tool in any hackers toolkit is the ability to interact with collected data. In this section, we will be using a programming language called Python to help with this.

4.2.1. Background

Python is a general purpose, versatile, and popular programming language. It is great as a first language because it is concise and easy to read. It is also a good language to have in any programmer's stack as it can be used for everything from web development to software development and data science applications.

This is different than what we have done to this point in Arduino. We will generally use the Arduino devices to directly interface with systems. These devices will then send data back to a computer or other device running a python script.

4.2.2. Codecademy

We will begin by having you learn the basics of python. We will be using an external system to aid in this. Navigate over to hereand create an account using your email and fill out any information they require. Please ensure that you do not pay for this service at this time. We will only need the services that the free version offers.

Please complete this course. Take notes throughout the course over functions that may be useful (importing and exporting files, searching through lists, data structures, etc.) Periodically, there will be additional trial material to supplement your learning.

4.2.3. Editors and IDE

4.2.4. Threading

4.2.5. PySerial

4.2.6. matplotlib

4.2.7. Making GUIs with PyQt5

4.3. C++ Programming Language

4.3.1. Codecademy

4.4. Computer Programming Fundamentals Exercises

5. The UNIX/Linux Terminal

5.1. Learn the Command Line

Click the link to go to Codecademy.

5.2. Learn Git

Click the link to go to Codecademy.

5.3. SocketCAN

According to the official documentation, The SocketCAN package is an implementation of CAN protocols (Controller Area Network) for Linux. Basically, it's a lightweight API that allows us to interact with CAN interfaces on Linux based computers. With SocketCAN we can read messages in real-time on a network, send custom messages, parse, and even replay messages with relative ease and speed. The best part is that the package is free, open-source and is powerful enough to use in many non-commercial applications. Additional documentation can be found here?

5.4. Terminal Exercises

6. Networking Fundamentals

6.1. Overview

Serial, USB, Ethernet, SPI, Wifi, What these are, what they do, most popular, basic intro

6.2. Problems of Networking

6.2.1. Encoding

6.2.2. Framing

6.2.3. Data Transmission

Link Access

Media Access Control

Collision Avoidance

6.2.4. Flow Control

6.2.5. Routing

switching

6.2.6. Reliable Delivery

6.2.7. Fault Tolerance

error detection

error correction - Hamming Codes

6.3. OSI Layered Model

6.4. Ethernet

6.4.1. TCP/IP

6.4.2. UDP

6.4.3. Automotive Ethernet

6.5. Wireless Protocols

6.5.1. Wifi

Krack

6.5.2. Cellular

6.5.3. Bluetooth

Blueman

Blueborne

6.5.4. Zigbee

6.5.5. Iridium Satelite

6.5.6. Other Communications

6.6. Networking Fundamentals Exercises

7. Vehicle Networking Protocols

CAN, LIN, I2C, J1708 and J1587, and J1939

7.1. Contoller Area Networks

A short overview of CAN is provided here. If you attend the SAE CyberAuto Challenge you will be provided with more resources to learn about CAN.

7.2. J1939

7.3. J1708/1587 Serial Communications

SAE J1708 and J1587 are the legacy network protocols for heavy vehicles. Together they provide detailed specifications for the electronics, hardware, message structuring, decoding and more. Namely, J1708 deals with the physical layer (the wiring), and J1587 deals with the the message layer or data format.

Often times engineers will refer to the two as simply "J1708" but it is important to remember that they are usually referring to both. After its release in the '90s, the standards were revised intermittently and were widely adopted despite their issues. Most notably, the J1708 network speed of 9600 bits/second limited the amount of data and network capabilities that could be included in a vehicle.

Today, J1708/1587 is now "stabilized" (no more future updates) and has been mostly supplanted by the newer J1939 protocol. However, it is not uncommon to run into the legacy protocol when working on older trucks and trailers. Some OEMs are still designing the older protocol out of their vehicles as well. Thus, it is important to be aware of J1708/1587. Plus, if you learn to send and decode J1587 messages, learning the newer protocols is much easier!

For more info click here.

7.3.1. Physical Layer

As stated earlier, SAE J1708 describes the physical layer of legacy truck networks.

Figure 7.3.1.: Title

7.3.2. Application layer

7.4. PLC4Trucks/J2497

So far we have only discussed the communications on the truck network. But what if we also want data from trailer? The linkage between the tractor and trailer is poer

7.5. RP1210

7.6. Vehicle Networking Protocols Exercises

Exercise 7.1. CAN Frame Decoding

Capture a CAN Frame using an oscilloscpope on a J1939 network and decode it according to SAE J1939.

Exercise 7.2. Read Live Engine RPM Challenge

Using a BeagleBone Black and a Truck Cape, connect to an engine contoller that is broadcasting non-zero engine RPM. Gather this data using candump. Interpret the raw CAN frames and extract information for Engine RPM, or J1939 SPN 190. Plot 20 seconds of changing RPM with matplotlib. Print the properly labeled plot to PDF and show it to your instructor. Objectives

Learn how to interface with Linux SocketCAN and can-utils Be able to look up a signal definition in the J1939 Digital Annex (spreadsheet) Use grep to search for specific strings from a candump Have a reliable CAN datalogger for use in future projects Plot data using matplotlib in Python.

Suggested Materials

This exercise can be run with any Linux device with CAN hardware. An example of a commercial product with these features is the DG Technologies' Beacon device. An example of a hand built project is the BeagleBone Black with a TU TruckCape. Resources

J1939DA Internet Access (you may want to share your PC's connection sharing)

Exercise 7.3. Man in the Middle

Build a man-in-the middle board and box that takes CAN signals into one can channel and sends them out on another. Start a diagnostics session with a PC and RP1210 device to perform maintenance. Create a forwarding system that inspects and forwards network traffic in both directions. Attempt to hijack a a diagnostic session and affect a parameter change started with the PC diagnostics software.

Using DDEC Reports, try to prevent resetting the CPC clock during a data extraction on a CPC.

8. Princples of Cybersecurity

8.1. Introduction to Cryptography

8.1.1. Terminology

At it's simplest, cryptography is packing and unpacking a secret message. The message you are attempting to send is known as the **plaintext**, P. The transformation of the plaintext into non-readable form is called **encryption**, E. The resulting obscured message is called **ciphertext**, C. The transformation of the cipher text back to the original plaintext is called **decryption**, D. This process is illustrated in Fig X.

8.1.2. Randomness

8.1.3. Hashes

8.1.4. Encryption

8.1.5. Diffie-Hellman Key Exchange

8.1.6. AES Encryption

8.1.7. RSA Asymmetric Encryption

8.2. Message Authentication

8.3. Encryption on CAN

8.4. Principles of Cybersecurity Exercises

9. Hardware Design

9.1. PCBs and Wiring Schematics with Altium Designer

9.2. 3D Modeling and Printing with Solidworks

9.3. Hardware Design Exercises

10. Heavy Vehicle Digital Forensics

10.1. Heavy Vehicle Digital Forensics Exercises

11. Colorado State University Resources

11.1. Welcome to the Powerhouse

11.2. Parts and Ordering

11.3. Colorado State University Exercises

12. Challenges

12.0.1. Challenge X: Receive CAN Using Adapter

12.0.2. Challenge X: Send and Receive CAN messages using Arduino

12.0.3. Challenge X: Plot RPM and Wheel Based Vehicle Speed

12.0.4. Challenge X: Hand Decode CAN frame with Oscilloscope

12.0.5. Challenge X: Build an Arduino from Scratch

12.0.6. Challenge X:

12.0.7. Challenge X:

12.0.8. Challenge X:

12.0.9. Challenge X:

A. Data Sheet Snippets