RIDWAN HOSSAIN
100747897

2022/06/01

- Received an email from Joelma containing materials on her work regarding the CAN bus.

- **Did some research on what a CAN bus is, and made the following notes:**
    - A CAN (Control Area Network) bus is a vehicle bus designed to allow microcontrollers and devices to communicate with each other's applications without a host computer.

    - It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles to save on copper.

    - The modern automobile may have as many as 70 electronic control units (ECU) for various subsystems. Traditionally, the biggest processor is the engine control unit. Others are used for Autonomous Driving, Advanced Driver Assistance System (ADAS), transmission, airbags, anti lock braking/ABS, cruise control, electric power steering, audio systems, power windows, doors, mirror adjustment, battery and recharging systems for hybrid/electric cars, etc. Some of these form independent subsystems, but communications among others are essential. A subsystem may need to control actuators or receive feedback from sensors. **The CAN standard was devised to fill this need.** One key advantage is that interconnection between different vehicle systems can allow a wide range of safety, economy and convenience features to be implemented using software alone - functionality which would add cost and complexity if such features were "hard wired" using traditional automotive electrics.

    - **My understanding of this is that** it will allow devices and other subsystems to communicate with microcontrollers without a host machine. For example, currently, we need the Linux PC as the host machine for the Arduino microcontroller to be able to communicate data from the Carla simulator to the peripherals, such as the gauges. Utilizing a CAN bus can help streamline this communication.

- According to Joelma, all the necessary files and dependencies are already installed on the Linux machine in the Carla directory. Further information and a video explanation can be found in Joelma's repository: https://github.com/joelmap/Virtual_CAN_bus_with_Carla_simulator

- **After watching Joelma's video, took note of some things:**
    - The CANTools can be used to send data from the Carla simulator to an end device through the CAN bus.

    - Joelma acquired a **dbc** file from the open dbc repository, called **honda.dbc**, which has some preconfigured commands to display information such as wheel speeds, steering angle, gearing data, etc.

    - The Carla client was modified to communicate through the CAN bus by adding functions to send and receive messages from the dbc end.

    - After running the Carla server and the modified client, Joelma used the command **candump vcan0** in order to print all data that is being received by a CAN interface. When she moves the vehicle, the messages being printed change. Afterwards, she used the **candump vcan0 | cantools decode honda.dbc** command in order to visualize the print messages better with respect to the decoding done by the **honda.dbc** file. This helps to make sense of the data being received by the CAN interface, such as speed, steering angle, etc.

    - Afterwards, the **CANdevStudio** software was used to simulate remote attacks on the vehicle by sending messages to the Carla simulator using the CAN bus and performing random actions on the vehicle, such as turning the wheel and accelerating. I presume the **Id** pertains to the area of vehicle control the specific attack is targeting, and the **Data** pertains to the messages (controls) being sent to the Carla end.

- After reviewing the materials given by Joelma and gaining a rough understanding of how all this works, I tried out replicating what Joelma did in the video to make sure it is still working fine.

- I was able to replicate everything Joelma did in her video up until she sent attacks to the Carla simulator using the CAN bus. The files for those attacks do not seem to be on the lab computer or her GitHub repository. I sent her an email to ask if she has those files. The next step for now is to implement those same CAN functions that Joelma had in her modified client in the new Carla simulator bench (testbed.py).

- ‘Testbed.py’ was successfully modified to bind with the socket which communicates with the CAN interface.Using the existing files (vcan.sh and honda.dbc), I was able to have ‘testbed.py’ communicate the following data through the CAN bus, which gets printed to the terminal (same as Joelma’s):
    - Speed
    - Steering angle
    - Current gear (This one was modified and thus different from Joelma’s)

- Maria had gotten back to me and sent me the files for the three attacks. Next step is to get CANdevStudio installed on the Linux machine and attempt to execute those attacks.

- During the installation, got the error during the build process: ‘**By not providing "FindQt5Core.cmake" in CMAKE_MODULE_PATH this project has asked CMake to find a package configuration file provided by "Qt5Core", but CMake did not find one.**’. Found a possible solution in the following source:

https://askubuntu.com/questions/374755/what-package-do-i-need-to-build-a-qt-5-cmake-application

- New error '**By not providing "FindQt5SerialBus.cmake" in CMAKE_MODULE_PATH this project has asked CMake to find a package configuration file provided by "Qt5SerialBus", but CMake did not find one.**'. Looking in how to find missing packages in cmake.
- Spent an extensive amount of time researching this but none of the sources I've found have seemed to be of much help to me. As I am very unfamiliar with qt and cmake, I emailed Joelma for assistance and I'm awaiting her response.