The Car Hacker's Handbook: A Guide for the Penetration Tester - Craig Smith (2016)

A. TOOLS OF THE TRADE

This section discusses different tools that you may want to use when researching a vehicle. I've chosen to focus on low-cost devices and software because it's important to me that as many people as possible participate in the research.

Open Garages is willing to showcase and promote tools to aid with automotive research. If your company produces a great product, feel free to contact Open Garages, but unless there's an open way to contribute to your tool, don't expect free publicity.

Hardware

In this section, we'll cover boards, like the ChipWhisperer, as well as dongle-like devices that provide CAN connectivity. We'll first look at lower-cost, open source hardware and then explore some higher-end devices for those willing to spend a bit more money.

Though there are many cost-effective devices for communicating with the CAN bus, the software needed to interact with these devices can be lacking, so you'll often need to write your own.

Lower-End CAN Devices

These devices are useful for sniffing the contents of your CAN bus and injecting packets. They range from hobbyist-level boards to professional devices that support lots of custom features and can handle many different CAN buses simultaneously.

Arduino Shields

Numerous Arduino and Arduino-like devices ($20 to $30, *https://www.arduino.cc/*) will support CAN with the addition of an Arduino shield. Here are some Arduino shields that support CAN:

CANdiy-Shield MCP2515 CAN controller with two RJ45

connectors and a protoarea

ChuangZhou CAN-Bus Shield MCP2515 CAN controller with a D-sub connector and screw terminals

DFRobot CAN-Bus Shield STM32 controller with a D-sub connector

SeeedStudio SLD01105P CAN-Bus Shield MCP2515 CAN controller with a D-sub connector

SparkFun SFE CAN-Bus Shield MCP2515 CAN controller with a D-sub connector and an SD card holder; has connectors for an LCD and GPS module

These shields are all pretty similar. Most run the MCP2515 CAN controller, though the DFRobot shield uses a STM32, which is faster with more buffer memory.

Regardless of which shield you choose, you'll have to write code for the Arduino in order to sniff packets. Each shield comes with a library designed to interface with the shield programmatically. Ideally, these buses should support something like the LAWICEL protocol, which allows them to send and receive packets over serial via a userspace tool on the laptop, such as SocketCAN.

Freematics OBD-II Telematics Kit

This Arduino-based OBD-II Bluetooth adapter kit has both an OBD-II device and a data logger, and it comes with GPS, an accelerometer, and gyro and temperature sensors.

CANtact

CANtact, an open source device by Eric Evenchick, is a very affordable USB CAN device that works with Linux SocketCAN. It uses a DB 9 connector and has the unique advantage of using jumper pins to change which pins are CAN and ground—a feature that allows it to support both US- and UK-style DB9 to OBD-II connectors. You can get CANtact from *http://cantact.io/*.

Raspberry Pi

The Raspberry Pi is an alternative to the Arduino that costs about $30 to $40. The Pi provides a Linux operating system but doesn't include a CAN transceiver, so you'll need to purchase a shield.

One of the advantages of using a Raspberry Pi over an Arduino is that it allows you to use the Linux SocketCAN tools directly, without the need to buy additional hardware. In general, a Raspberry Pi can talk to an MCP2515 over SPI with just some basic wiring. Here are some Raspberry Pi implementations:

Canberry MCP2515 CAN controller with screw terminals

only (no D-sub connector; $23)

Carberry Two CAN bus lines and two GMLAN lines, LIN, and infrared (doesn't appear to be an open source shield; $81)

PICAN CAN-Bus Board MCP2515 CAN controller with D-sub connector and screw terminals ($40 to $50)

ChipKit Max32 Development Board and NetworkShield

The ChipKit board is a development board that together with the NetworkShield can give you a network-interpretable CAN system, as discussed in "Translating CAN Bus Messages" on page 85. About $110, this open source hardware solution is touted by the OpenXC standard and supports prebuilt firmware from OpenXC, but you can also write your own firmware for it and do raw CAN.

ELM327 Chipset

The ELM327 chipset is by far the cheapest chipset available at anywhere (from $13 to $40), and it's used in most cheap OBD device. It communicates with the OBD over serial and comes with just about any type of connector you can think of, from USB to Bluetooth, Wi-Fi, and so on. You can connect to ELM327 devices over serial, and they're capable of sending packets other than OBD/UDS packets. For a full list of commands using the ELM327, see the data sheet at *http://elmelectronics.com/DSheets/ELM327DS.pdf*.

Unfortunately, the available CAN Linux tools won't run on the ELM327, but Open Garages has begun a web initiative that includes sniffing drivers for the ELM327 called CANiBUS (*https://github.com/Hive13/CANiBUS/*). Be forewarned that the ELM327 has limited buffer space, so you'll lose packets when sniffing and transmission can be a bit imprecise. If you're in a pinch, however, this is the cheapest route.

If you're willing to open the device and solder a few wires to your ELM327, you can reflash the firmware and convert it into a LAWICEL-compatible device, which allows your uber cheap ELM327 to work with Linux and show up as an slcanX device! (You'll find information on how to flash your ELM327 on the Area 515 makerspace blog from Des Moines, Iowa, at *https://area515.org/elm327-hacking/*.)

GoodThopter Board

Travis Goodspeed, a well-known hardware hacker, has released an open source, low-cost board with a CAN interface called the GoodThopter. The GoodThopter, based on his popular GoodFet devices, uses MCP2515 and communicates over serial with its own custom interface. You'll need to completely assemble and solder together the device yourself, but doing so should cost just a few dollars, depending on the parts you have available at your local hackerspace.

ELM-USB Interface

OBDTester.com sells a commercial ELM-32x-compatible device for around $60. OBDTester.com are the maintainers of the PyOBD library (see “Software” on page 246).

CAN232 and CANUSB Interface

LAWICEL AB produces the commercial CAN device CAN232, which plugs into an RS232 port with a DB9 connector, and a USB version called CANUSB (the latter goes for $110 to $120). Because they’re made by the inventors of the LAWICEL protocol, these devices are guaranteed to work with the can-utils serial link modules.

VSCOM Adapter

The VSCOM is an affordable commercial USB CAN module from Vision Systems (*http://www.vscom.de/usb-to-can.htm*) that uses the LAWICEL protocol. VSCOM works with the Linux can-utils over serial link (slcan) and provides good results. The device costs around $100 to $130.

USB2CAN Interface

The USB2CAN converter from 8devices (*http://www.8devices.com/usb2can/*) is the cheapest alternative to a nonserial CAN interface. This small, commercial USB device will show up as a standard can0

device in Linux and has the most integrated support in this price range. Most devices that show up as canX raw devices are PCI cards and typically cost significantly more than this device.

EVTV Due Board

EVTV.me (*http://store.evtv.me/*) specializes in electric car conversions. They make lots of great tools for doing crazy things to your historic vehicle, like adding a Tesla drivetrain to it. One of their tools is a $100 open source CAN sniffer called the EVTV Due, which is basically an Arduino Due with a built-in CAN transceiver and handle-screw terminals to interface with your CAN lines. This board was originally written to work solely with their SavvyCAN software, which uses their Generalized Vehicle Reverse Engineering Tool (GVRET), but it now supports SocketCAN as well.

CrossChasm C5 Data Logger

The CrossChasm C5 (*http://www.crosschasm.com/technology/data-logging/*) is a commercial device that supports the Ford VI firmware and costs about $120. The C5 supports the VI, which is also known as the CAN translator, to convert CAN messages to the OpenXC format, and it converts some proprietary CAN packets into a generic format to send over Bluetooth.

CANBus Triple Board

As I write this, the CANBus Triple (*http://canb.us/*) is still in development. It uses a wiring harness designed to support Mazda, but it supports three CAN buses of any vehicle.

Higher-End CAN Devices

Higher-end devices will cost you more money, but they're capable of receiving more simultaneous channels and offer more memory to help prevent packet loss. High-performance tools often support eight channels or more, but unless you're working on racing vehicles, you probably don't need that many channels, so be sure that you need devices like these before dropping any cash.

These devices often come with their own proprietary software or a software subscription at sometimes significant added cost. Make sure the software associated with the device you choose does what you want because you'll usually be locked into their API and preferred hardware. If you need higher-end devices that work with Linux, try Kvaser, Peak, or EMS Wünsche. The devices from these companies typically use the sja1000 chipset at prices starting around $400.

CAN Bus Y-Splitter

A CAN bus Y-splitter is a very simple device that's basically

one DLC connector broken into two connectors, which allows you to plug a device into one port and a CAN sniffer into the other. These typically cost around $10 on Amazon and are actually quite simple to make yourself.

HackRF SDR

HackRF is an SDR from Great Scott Gadgets (*https://greatscottgadgets.com/hackrf/*). This open source hardware project can receive and transmit signals from 10 MHz to 6 GHz. At about $330, you can't get a better SDR for the price.

USRP SDR

USRP (*http://www.ettus.com/*) is a professional, modular SDR device that you can build to suit your needs. USRP is open source to varying degrees at prices ranging from $500 to $2,000.

ChipWhisperer Toolchain

NewAE Technologies produces the ChipWhisperer (*http://newae.com/chipwhisperer/*). As discussed in "Side-Channel Analysis with the ChipWhisperer" on page 134, the ChipWhisperer is a system for side-channel attacks, such as power analysis and clock glitching. Similar systems usually cost $30,000 or more, but the ChipWhisperer is an open source system that costs between $1,000 and $1,500.

Red Pitaya Board

Red Pitaya (*http://redpitaya.com/*) is an open source measurements tool that for around $500 replaces expensive measurement tools such as oscilloscopes, signal generators, and spectrum analyzers. Red Pitaya has LabView and Matlab interfaces, and you can write your own tools and applications for it. It even supports extensions for things like Arduino shields.

Software

As we did with hardware, we'll focus first on open source tools and then cover more expensive ones.

Wireshark

Wireshark (*https://www.wireshark.org/*) is a popular network sniffing tool. It is possible to use Wireshark on a CAN bus network as long as you are running Linux and using SocketCAN. Wireshark doesn't have any features to help sort or decode CAN packets, but it could be useful in a pinch.

PyOBD Module

PyOBD (*http://www.obdtester.com/pyobd*)—also known as *PyOBD2* and *PyOBD-II*—is a Python module that communicates with ELM327 devices (see Figures A-1 and A-

2). It's based on the PySerial library and is designed to give you information on your OBD setup in a convenient interface. For a specific scan tool fork of PyOBD, see Austin Murphy's OBD2 ScanTool (*https://github.com/AustinMurphy/OBD2-Scantool/*), which is attempting to become a more complete open source solution for diagnostic troubleshooting.

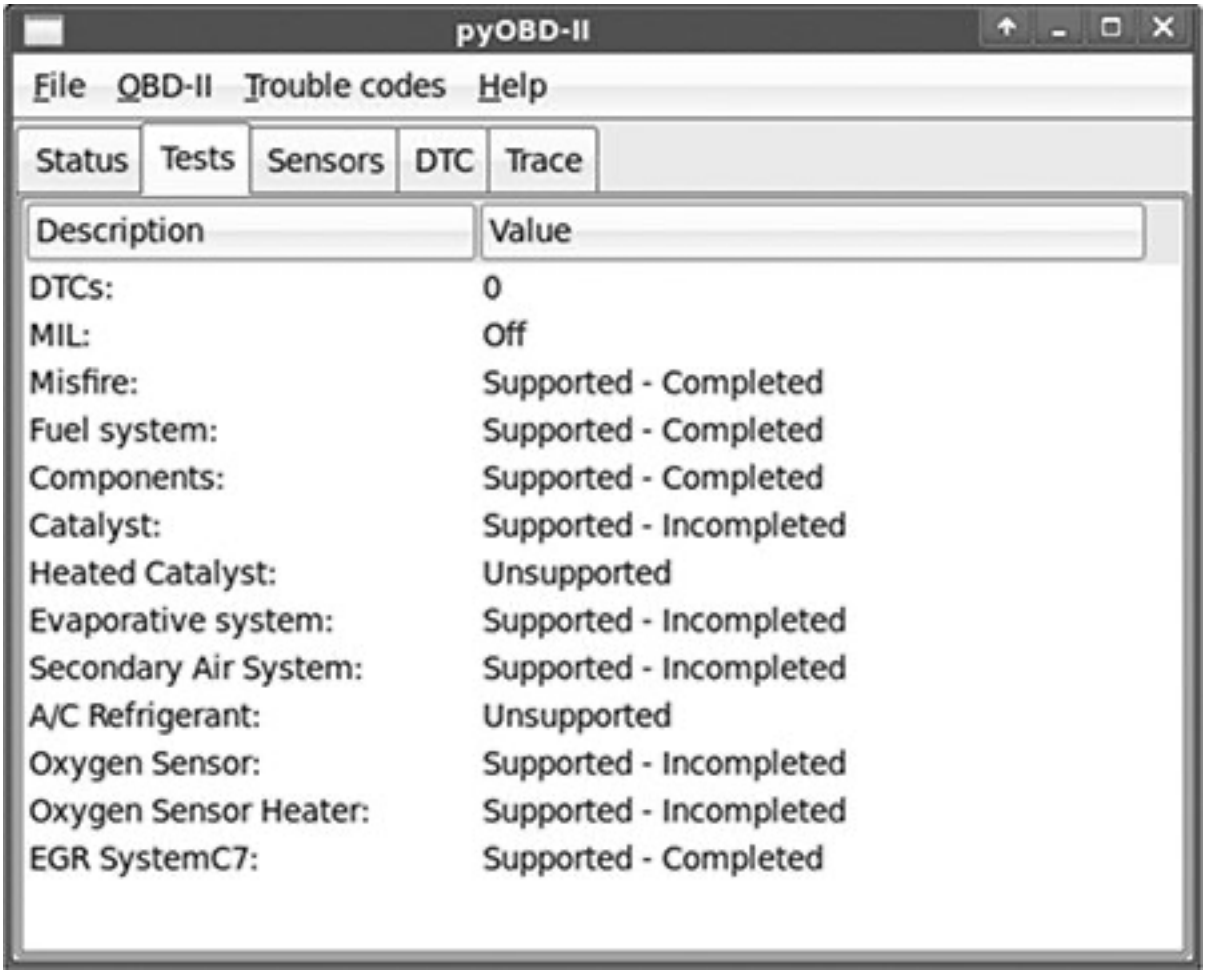

Figure A-1: PyOBD running diagnostic tests

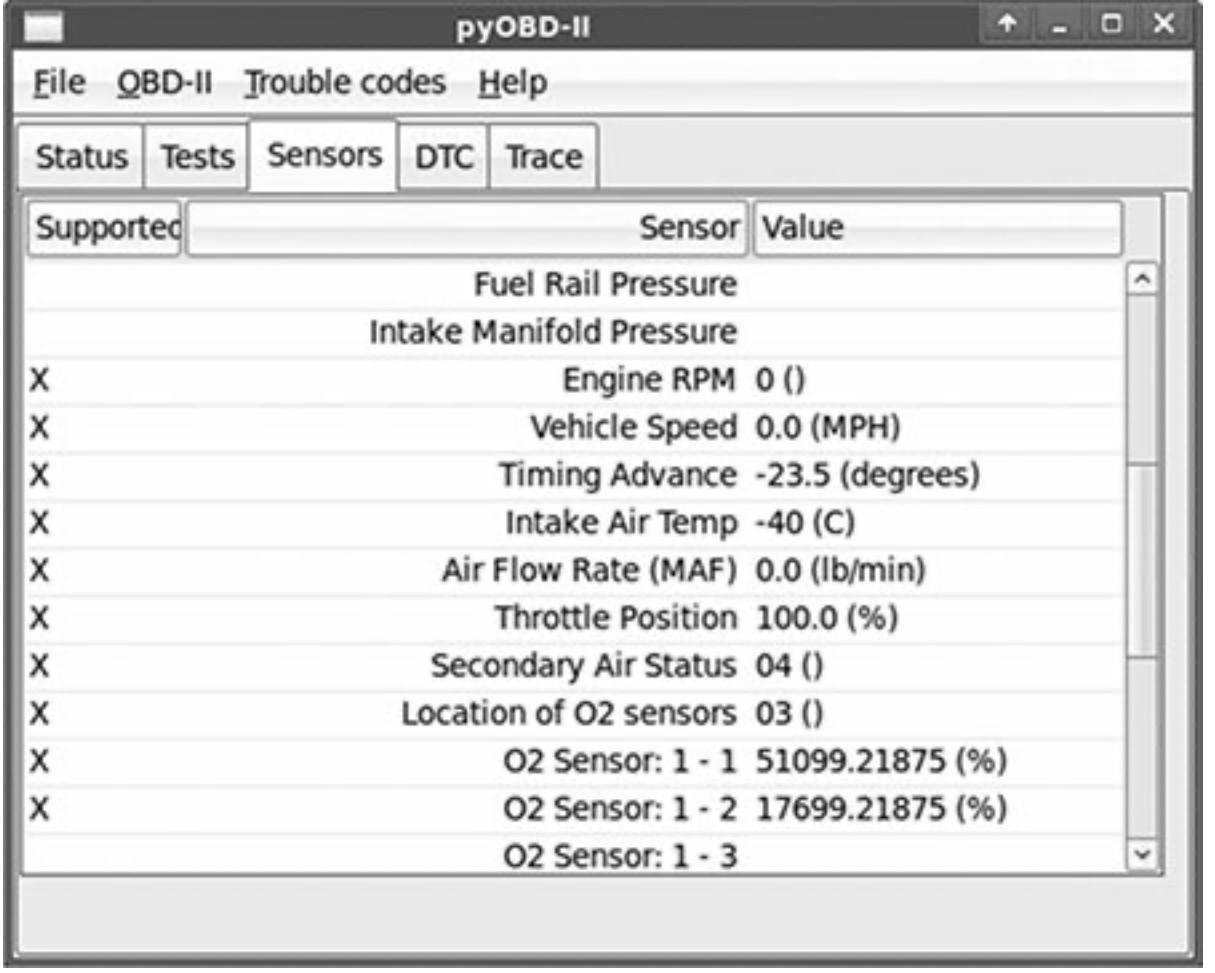

Figure A-2: PyOBD reading sensor data

Linux Tools

Linux supports CAN drivers out of the box, and SocketCAN provides a simple netlink (network card interface) experience when dealing with CAN. You can use its can-utils suite for a command line implementation, and as open source software, it's easy to extend functionality to other utilities. (See Chapter 3 for more on SocketCAN.)

CANiBUS Server

CANiBUS is a web server written in Go by Open Garages

(see Figure A-3). This server allows a room full of researchers to simultaneously work on the same vehicle, whether for instructional purposes or team reversing sessions. The Go language is portable to any operating system, but you may have issues with low-level drivers on certain platforms. For example, even if you're running CANiBUS on Linux, you won't be able to directly interact with SocketCAN because Go doesn't support the necessary socket flags to initialize the CAN interface. (This problem could be addressed by implementing socketcand, but as of this writing, that feature has yet to be implemented.) CANiBUS does have a driver for ELM327 that supports generic sniffing. You can learn more about CANiBUS at *http://wiki.hive13.org/view/CANiBUS/* and can download the source from *https://github.com/Hive13/CANiBUS/*.

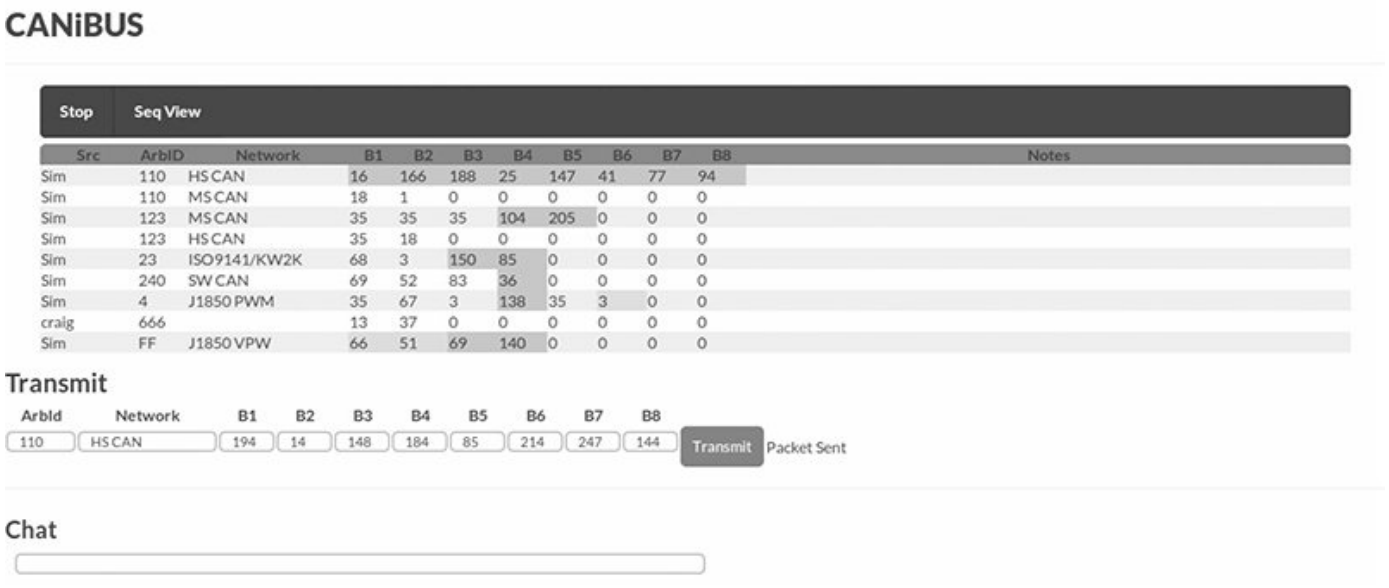

Figure A-3: CANiBUS group-based web sniffer

Kayak

Kayak (*http://kayak.2codeornot2code.org/*) is a Java-based GUI for analyzing CAN traffic. It has several advanced features, such as GPS tracking and record and playback capabilities. It utilizes socketcand in order to work on other operating systems, so you'll need at least one Linux-based sniffer to support Kayak. (You'll find more detail on setup and use in "Kayak" on page 46.)

SavvyCAN

SavvyCAN is a tool written by Collin Kidder of EVTV.me that uses another framework designed by EVTV.me, GVRET, to talk to HW sniffers such as the EVTV Due. SavvyCAN is an open source, Qt GUI–based tool that works on multiple operating systems (see Figure A-4). It includes several very nice features, such as DBC editor, CAN bus graphing, log file diffing, several reverse engineering tools, and all the normal CAN sniffing features you would expect. SavvyCAN doesn't talk to SocketCAN, but it can read in several different logfile formats, such as Bushmaster logs, Microchip logs, CRTD formats, and generic CSV-formatted logfiles.

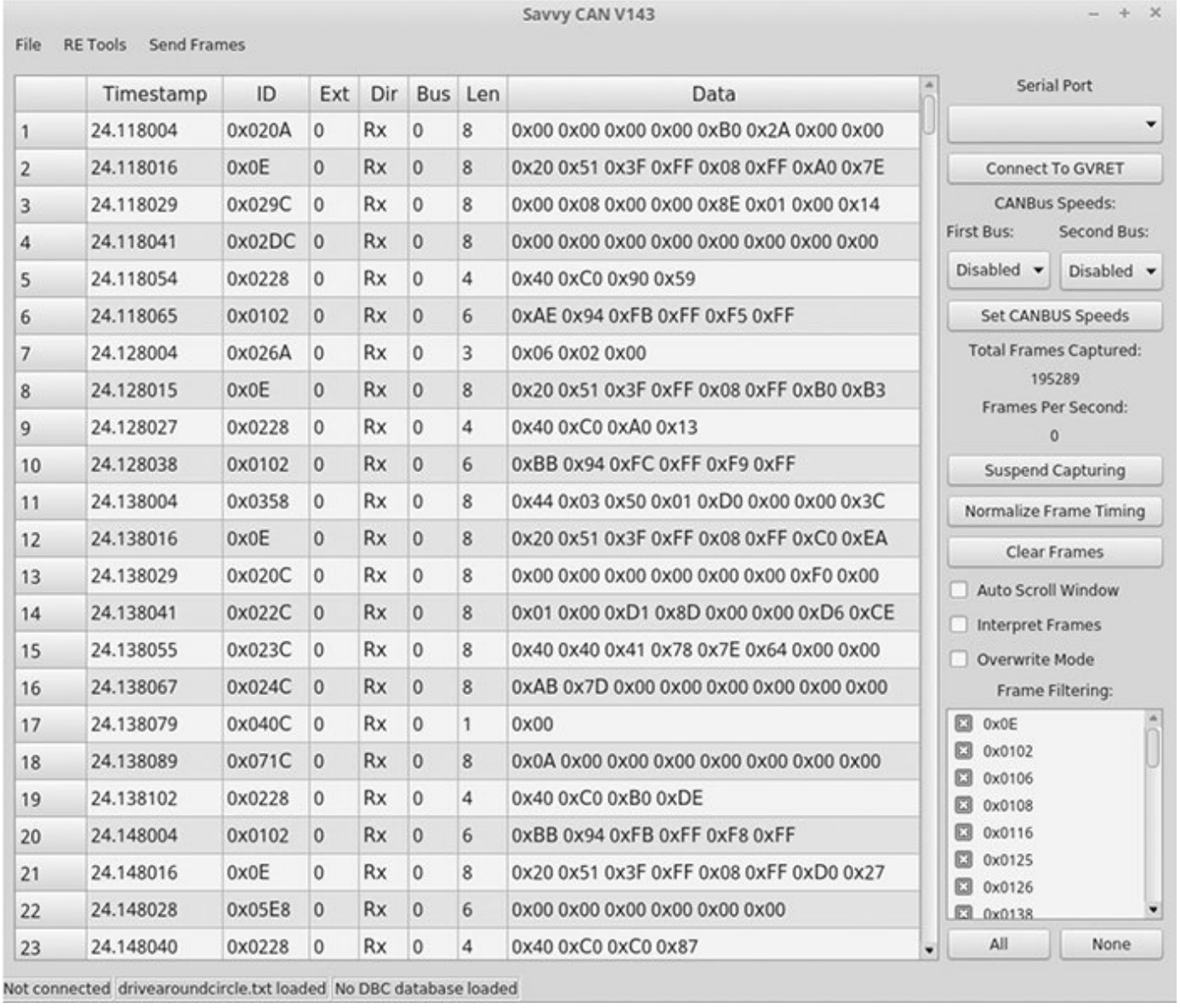

Figure A-4: SavvyCAN GUI

O2OO Data Logger

O2OO (*http://www.vanheusden.com/O2OO/*) is an open source OBD-II data logger that works with ELM327 to record data to a SQLite database for graphing purposes. It also supports reading GPS data in NMEA format.

Caring Caribou

Caring Caribou

(*https://github.com/CaringCaribou/caringcaribou/*), written in Python, is designed to be the Nmap of automotive hacking. As of this writing, it's still in its infancy, but it shows a lot of potential. Caring Caribou has some unique features, like the ability to brute-force diagnostic services, and handles XCP. It also has your standard sniff-and-send CAN functionality and will support your own modules.

c0f Fingerprinting Tool

CAN of Fingers (c0f) is an open source tool for fingerprinting CAN bus systems that can be found at *https://github.com/zombieCraig/c0f/*. It has some basic support for identifying patterns in a CAN bus network stream, which can be useful when trying to find a specific signal on a noisy bus. (See "Using c0f" on page 206 for an example of c0f at work.)

UDSim ECU Simulator

UDSim (*https://github.com/zombieCraig/UDSim/*) is a GUI tool that can monitor a CAN bus and automatically learn the devices attached to it by watching communications (see Figure A-5). It's designed to be used with another diagnostic tool, such as a dealership tool or a scan tool from a local automotive store.

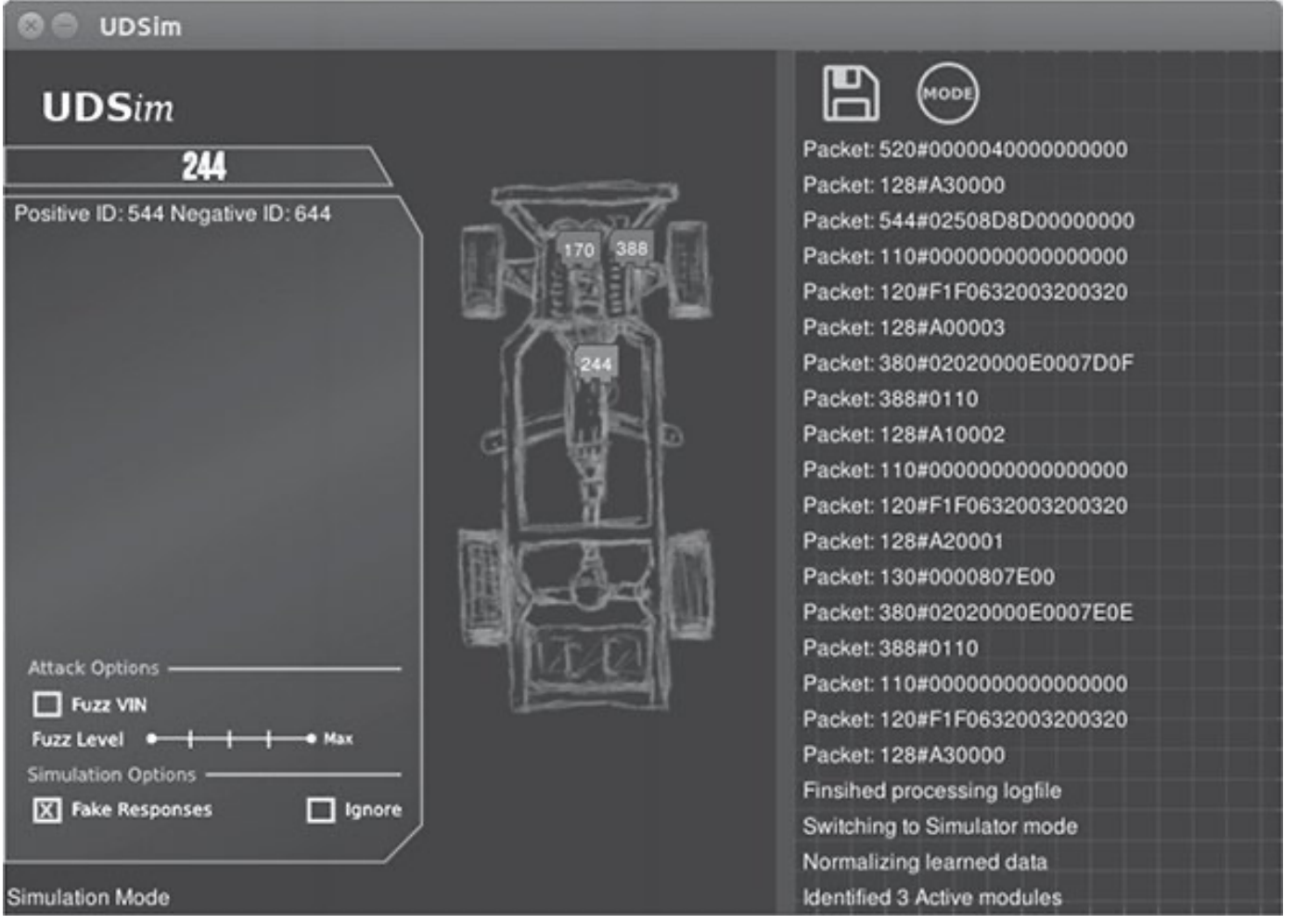

Figure A-5: Sample screen from UDSim as it learns modules off a test bench

UDSim has three modes: learning, simulation, and attack. In learning mode, it identifies modules that respond to UDS diagnostic queries and monitors the responses. In simulation mode, it simulates a vehicle on the CAN bus to fool or test diagnostic tools. In attack mode, it creates a fuzzing profile for tools like Peach Fuzzer (*http://www.peachfuzzer.com/*).

Octane CAN Bus Sniffer

Octane (*http://octane.gmu.edu/*) is an open source CAN bus sniffer and injector with a very nice interface for sending and

receiving CAN packets, including an XML trigger system. Currently, it runs only on Windows.

AVRDUDESS GUI

AVRDUDESS (*http://blog.zakkemble.co.uk/avrdudess-a-gui-for-avrdude/*) is a GUI frontend for AVRDUDE written in .NET, though it works fine with Mono on Linux. You'll see AVRDUDESS in action in "Prepping Your Test with AVRDUDESS" on page 139.

RomRaider ECU Tuner

RomRaider (*http://www.romraider.com/*) is an open source tuning suite for the Subaru engine control unit that lets you view and log data and tune the ECU (see Figure A-6). It's one of the few open source ECU tuners, and it can handle 3D views and live data logging. You'll need a Tactrix Open Port 2.0 cable and Tactrix EcuFlash software in order to download and use the ECU's firmware. Once you've downloaded the flash with EcuFlash, you can edit it with RomRaider. The editor is written in Java and currently works on Windows and Linux, though EcuFlash isn't supported on Linux.

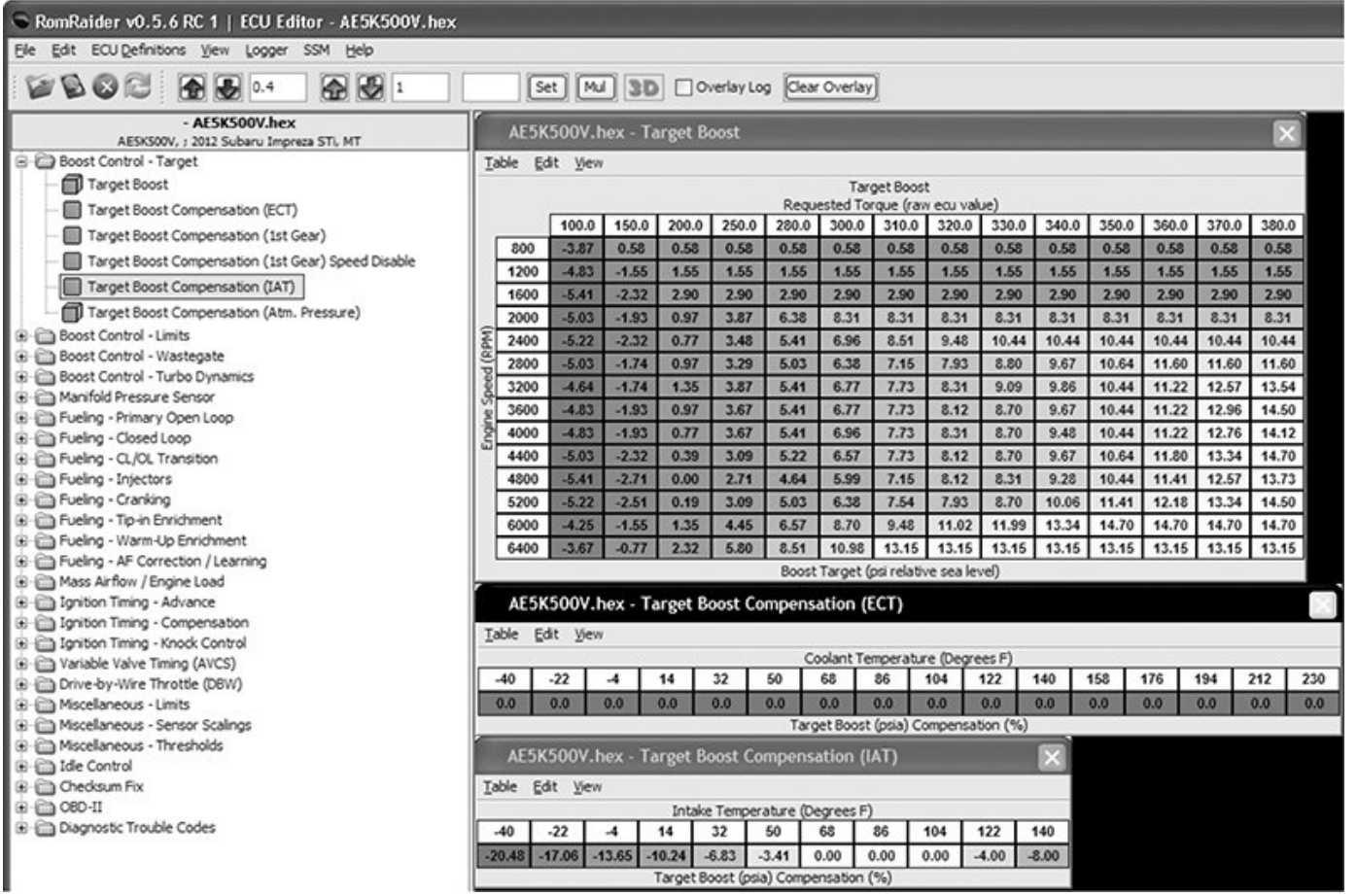

Figure A-6: RomRaider tuning editor

Komodo CAN Bus Sniffer

Komodo is a higher-end sniffer with a nice multioperating system—Python SDK. It costs around $350 to $450 depending on whether you want a single- or dual-CAN interface. Komodo has isolation capabilities to prevent your computer from frying if you miswire something, as well as eight general-purpose IO pins you can configure to trigger actions from external devices. Komodo comes with some decent software to get you up and running, but the real advantage is that you can write your own Komodo software.

Vehicle Spy

Vehicle Spy is a commercial tool from Intrepid Control Systems (*http://store.intrepidcs.com/*) that's specifically designed for reversing CAN and other vehicle communication protocols. The software requires one license per NeoVI or ValueCAN device, both proprietary devices for Vehicle Spy. The ValueCAN3 is the cheapest device that works with Vehicle Spy. It has one CAN interface and costs about $300. Add the Vehicle Spy Basic software and your cost will be about $1,300.

The NeoIV devices are higher end, with multiple configurable channels, starting at around $1,200. A basic package contains a NeoIV (Red) and Vehicle Spy Basic for $2,000, which saves a bit of money. Vehicle Spy Professional costs about $2,600 without hardware. (You'll find several options on Intrepid's site.)

All Intrepid hardware devices support uploading scripts to run on the bus in real time. Vehicle Spy Basic supports CAN/LIN RX/TX operations. You'll need the professional version only if car hacking is going to be a full-time project for you or if you want to use ECU flashing or other advanced features, such as Node Simulation, scripting on the sniffer, or memory calibration.