Teaching Automobile Hacking with the Instrument Cluster Simulator

Dr. Jim Marquardson, jimarqua@nmu.edu
Michael Sauer, msauer@nmu.edu
Northern Michigan University

Outline

- Why auto hacking?
- Simulation Benefits
- ICSim overview
- Setting up ICSim
- Curriculum examples
  - More available at https://jimmarquardson.com

Please stop me at any time if you have questions or comments.

Automotive Cybersecurity

- On-vehicle computers are increasingly becoming attack vectors threatening physical safety and data privacy
- The Controller Area Network Bus (CANBUS) technology used in vehicles was designed for efficiency, not security
- Teaching students about vehicle security will help them consider cybersecurity from a new perspective
  - Go beyond traditional TCP/IP networks

Auto Cyber Teaching Challenges

- Few instructors have access to physical cars in the classroom
- Equipment is costly and hard to scale to large classes

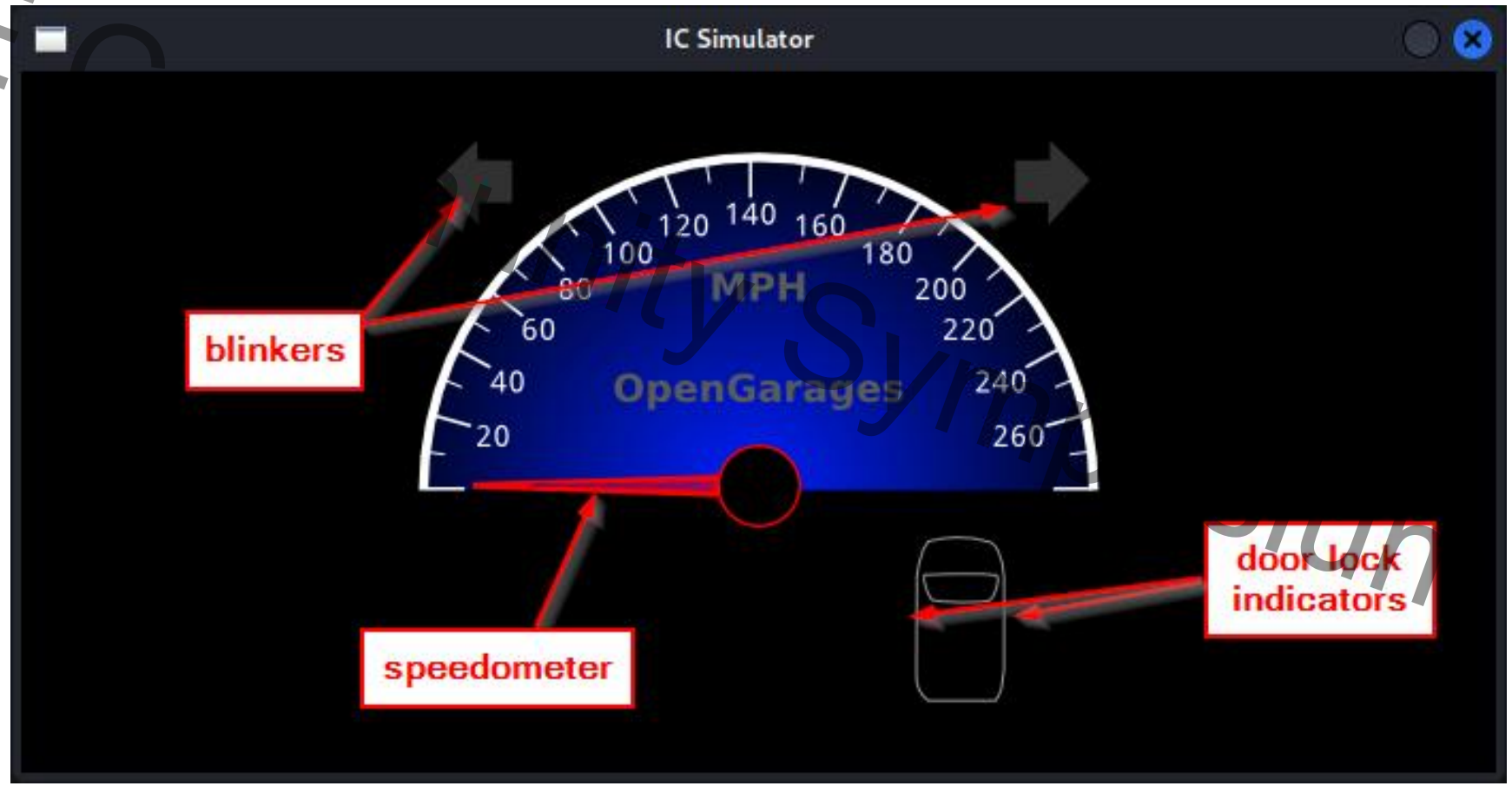
The “Auto Hacking Dashboard” at Northern Michigan University, source:nmu.edu

Instrument Cluster Simulator (ICSim)

- The Instrument Cluster Simulator (ICSim) provides a virtual CANBUS network.
  - https://github.com/zombieCraig/ICSim
- Free and open source (GPL 3)
- Should run on any Debian-based Linux distro with a graphical user interface (e.g., Kali, Ubuntu)
- Students use the same tools to interact with a virtual car that they would use if interacting with a real car

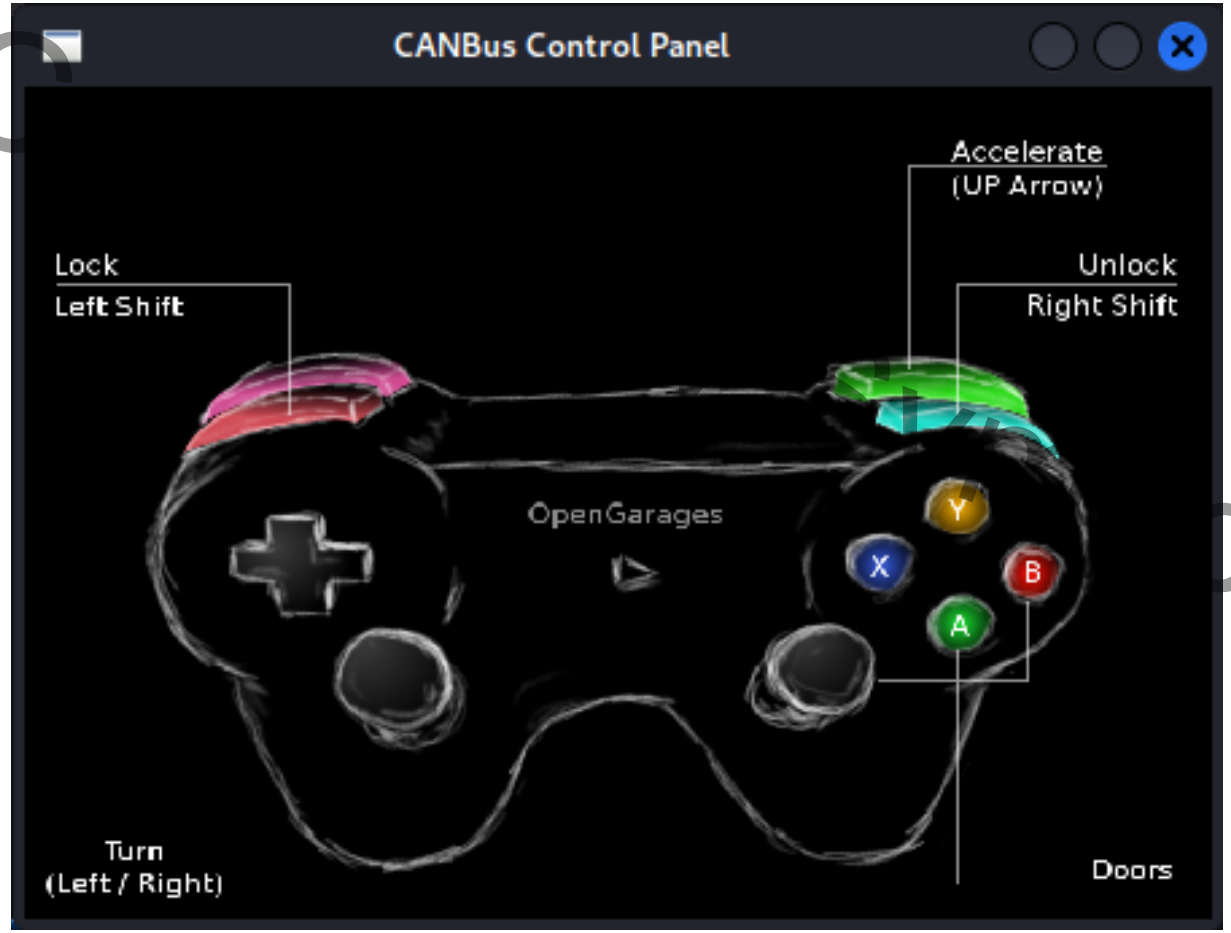
ICSim Dashboard

- Definitely not a racing game
- Can control: blinkers, speedometer, door locks



Controlling the Virtual Car

- A virtual controller operates the car by sending CAN traffic on the virtual CAN network
- Example: ***left shift + b*** locks the right door

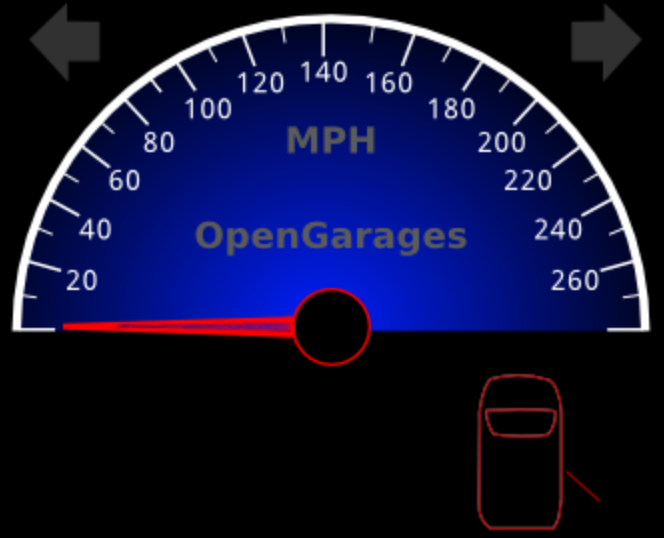


kali@kali: ~/ICSim

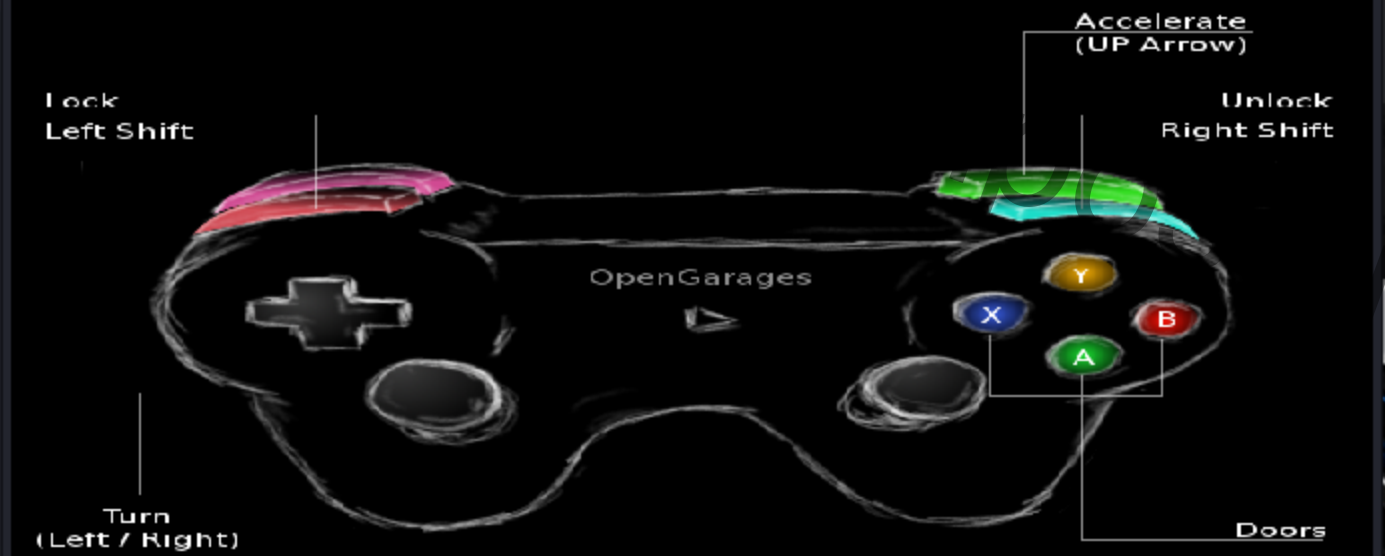
File Actions Edit View Help

```
24014 | 039 | 00 0C                                     ........
00015 | 039 | 00 2A                                     .*.....5
00009 | 095 | 80 00 07 F4 00 00 00 26                   .......&
00009 | 133 | 00 00 00 00 B6                            ........
00010 | 136 | 00 02 00 00 00 00 00 39                   .......9
00010 | 13A | 00 00 00 00 00 00 00 37                   .......7
00010 | 13F | 00 00 00 05 00 00 00 3D                   .......=
00010 | 143 | 6B 6B 00 FF                               kk.....7
00008 | 158 | 00 00 00 00 00 00 00 28                   .......(
00009 | 161 | 00 00 05 50 01 08 00 2B                   ...P...+
00010 | 164 | 00 00 C0 1A A8 00 00 13                   ........
00011 | 166 | D0 32 00 36                               .2.6....
00008 | 17C | 00 00 00 00 10 00 00 30                   .......0
00011 | 183 | 00 00 00 09 00 00 10 3F                   .......?
00008 | 18E | 00 00 7A                                  ..z.A.!?
00009 | 191 | 01 00 10 A1 41 00 1A                      ....A...
00020 | 1A4 | 00 00 00 08 00 00 00 3E                   .......>
00020 | 1AA | 7F FF 00 00 00 00 68 3E                   ......h>
00020 | 1B0 | 00 0F 00 00 00 01 75                      ......u.
00020 | 1CF | 80 05 00 00 00 1E                         ........
00019 | 1DC | 02 00 00 1B                               ....",/.
00040 | 21E | 03 E8 37 45 22 06 10                      ..7E"..8
00011 | 244 | 00 00 00 01 34                            ....4Z.,
00040 | 294 | 04 0B 00 02 CF 5A 00 1D                   .....Z..
00104 | 305 | 80 26                                     .&......
00099 | 309 | 00 00 00 00 00 00 00 B1                   ........
00099 | 320 | 00 00 21                                  ..!....8
00099 | 324 | 74 65 00 00 00 00 0E 29                   te.....)
00100 | 333 | 00 00 00 00 00 00 2D                      ......-8
00101 | 37C | FD 00 FD 00 09 7F 00 29                   .......)
00300 | 405 | 00 00 04 00 00 00 00 0B                   ........
00300 | 40C | 02 36 32 32 39 53 30 39                   .6229S09
00300 | 428 | 01 04 00 00 52 1C 01                      ....R..
00298 | 454 | 23 EF 36                                  #.6
01000 | 5A1 | 96 00 00 00 00 00 62 3E                   ......b>
```

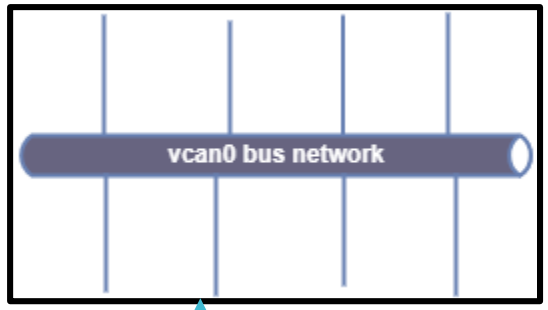
IC Simulator



CANBus Control Panel

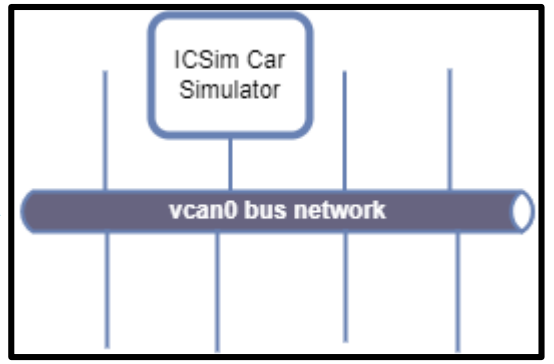


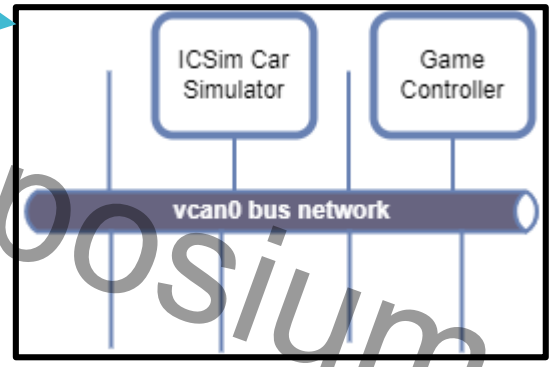
Network Topology

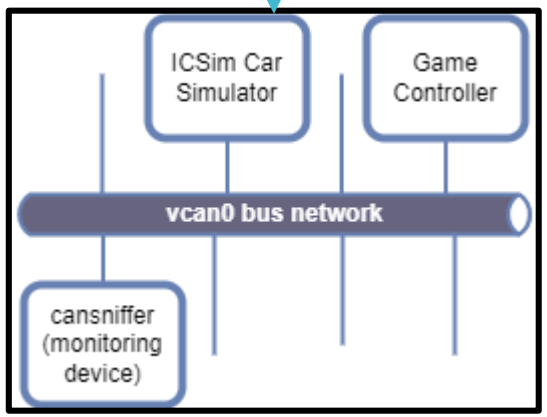


Typical Setup

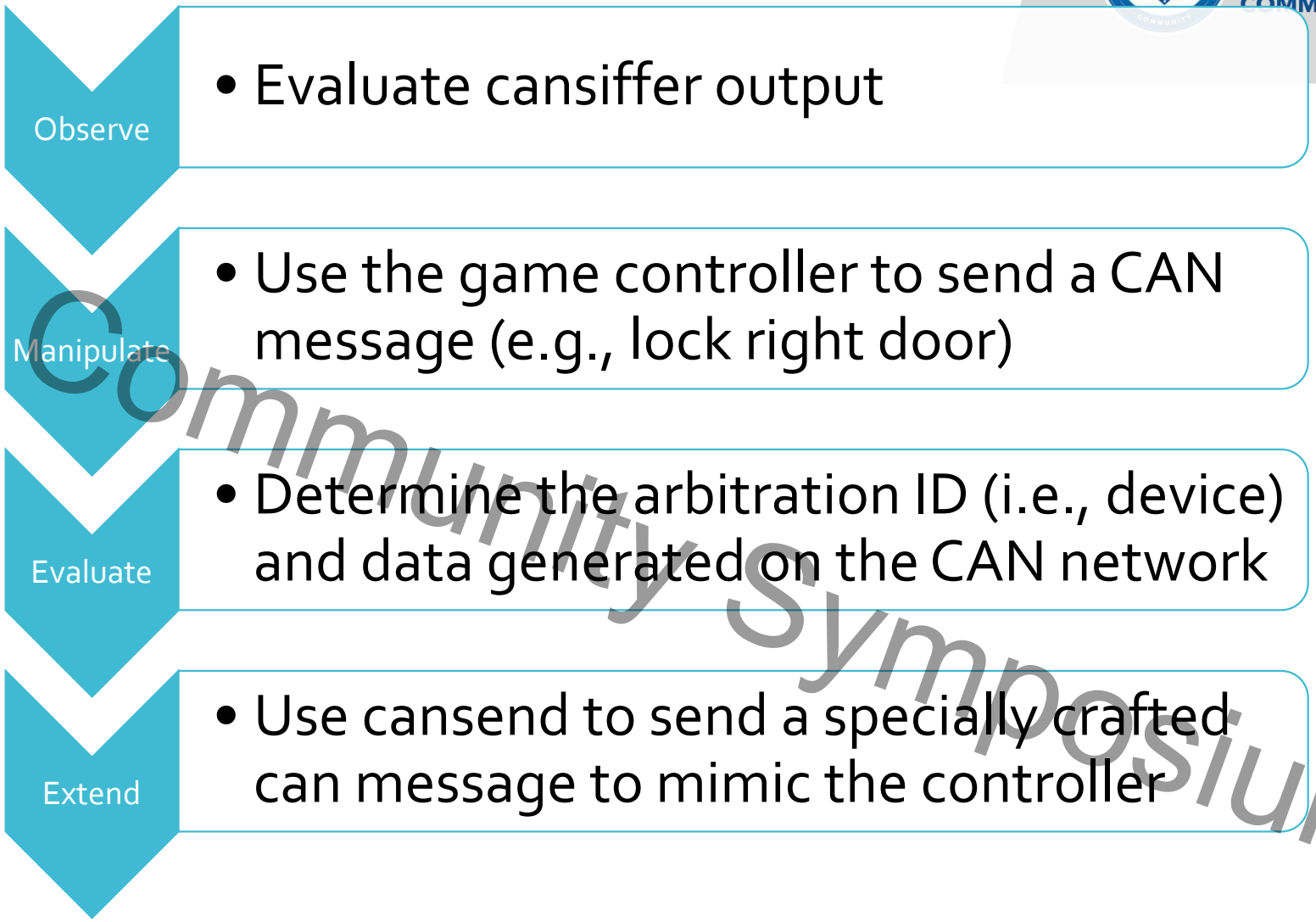
1. Virtual CANBUS network (vcan0) is created
2. The ICSim car simulator is attached to vcan0
3. The game controller is attached to vcan0
4. Can-utils applications monitor or send traffic







Investigation Workflow



can-utils

- The can-utils package has applications to monitor, record, and send CAN traffic
- These work with physical cars as well as ICSim

Application	Description	Example
cansniffer	Display real-time CAN network traffic	`cansniffer -c vcan0`
candump	Record CAN traffic to a file	`candump -l vcan0 -f out.txt`
cansend	Send a single CAN message	`cansend vcan0 188#01`
canplayer	Replay CAN traffic from a file	`canplayer -I test.log`
cangen	Generate random CAN traffic	`cangen vcan0 -I 445`

ICSim Installation

Linux Command	Description
`sudo apt update`	Update package repositories
`sudo apt install libsdl2-dev libsdl2-image-dev can-utils`	Install ICSim dependencies
`cd ~`	Ensure working directory is home
`git clone https://github.com/zombieCraig/ICSim.git`	Download ICSim source code
`git clone https://github.com/linux-can/can-utils`	Download can-utils source code
`cd ~/can-utils`	Enter the can-utils directory
`make`	Compile the can-utils code
`sudo make install`	Installs the can-utils binaries
`cp lib.o ~/ICSim`	Copy dependencies to the ICSim source code directory
`cd ~/ICSim`	Change directories to the ICSim directory
`make clean`	Gets rid of any previously compiled code
`make`	Compiles the ICSim binaries

Demo

- Demo
  - Setup the van0 network
  - Launch ICSim
  - Launch the controller
  - Start cansniffer

Sample Lesson Scenarios

- Level 1: Determine the arbitration ID and data sent on the CAN network to turn the left blinker on.
- Level 2: Determine the hex values for manipulating the accelerometer. Make the car report traveling exactly 120 MPH.
- Level 3: Write a bash script to send accelerometer values in a loop.
- Level 4: Simulate an attacker in the bushes by dumping CAN traffic while the door is unlocked. After locking the door, replay the traffic.
- Level 5: Use cansend to generate random traffic until the door unlocks. Write a Python program to efficiently find the CAN message that unlocks a door message in a candump file.
- Level 6: Write a Python program that filters out noise from candump files.

Tips based on our experience

- Set the scene
  - Ensure that the bus network topology is clear
  - The game controller is the legitimate interface
  - The Linux terminal is the “attacker”
- Don’t spoil the fun
  - Let students struggle to find the arbitration IDs
- Sample code helps

Are these skills useful if students do not plan to pursue careers in automotive security?

- Yes.
- CANBUS networks are found in many places outside of vehicles: mine safety equipment, boats, semi-trucks, and other industrial applications.
- Learning auto hacking can help build critical thinking, reverse engineering, and coding skills.

Questions?

- Thank you!
- Questions?
- Contact
  - Dr. Jim Marquardson, jimarqua@nmu.edu
  - Michael Sauer, msauer@nmu.edu
- Installation instructions, sample exercises, and code are available at https://jimmarquardson.com