How to Hack Your Mini Cooper: Reverse Engineering CAN Messages on Passenger Automobiles

Jason Staggs
Who is this guy?

- **Jason Staggs**
  - Graduate Research Assistant
    - Institute for Information Security (iSec)
    - Crash Reconstruction Research Consortium (TU-CRRC)
  - TRUE Digital Security
    - Cyber Security Analyst
Why do we hack cars?

• Related work
  – “Experimental Security Analysis of a Modern Automobile”
  – “Comprehensive Experimental Analyses of Automotive Attack Surfaces”

• Understanding computer and network systems on cars
  – Underlying CAN protocol and components lack of authentication and verification of messages

• Understanding potential points of vulnerability
  – Vehicle network security is in its infancy

• But most importantly…
To prevent this…
From turning into this…
Because of this...
CAN Clock Project

- Research project developed as a proof of concept
  - Manipulating CAN nodes via CAN network
  - Reverse engineering CAN messages
  - 2003 Mini Cooper
Vehicle communication networks

• **Common vehicle protocols**
  – CAN (Most widely used among manufactures)
  – FlexRay
  – LIN
  – MOST
  – J1850 (GM/Chrysler)
  – J1939 (Heavy Trucks)
  – J1708/J1587 (Being phased out due to J1939)

• **2008: All US cars use CAN for mandated EPA diag.**
Interconnected vehicle networks
Controller Area Networks

- **Bosch CAN standard**
  - Developed in the 80s
  - European automotive manufactures were early adopters
  - Multi-master broadcast message system
  - **Standard Format**
    - 11-bit message ID
    - $2^{11}$ or 2048 possible message IDs
    - MFG. use of proprietary IDs for their ECUs
  - **Extended Format**
    - 29-bit message ID
    - $2^{29}$ or 537 million message IDs
    - Used extensively by J1939
CAN Frame

- **SOF** - Start of Frame
- **Identifier** - Unique identifier for message along with priority
- **RTR** - Remote Transmission Request
- **IDE** - Identifier extension (distinguishes between CAN standard and CAN extended)
- **DLC** - Data Length Code (frames have up to 8 bytes of data)
- **CRC** - Cyclic Redundant Check sum
- **ACK** - Acknowledge
- **EOF** - End of Frame
- **IFS** - Intermission Frame Space
Electronic Control Units (ECUs)

- **ECUs designed to control:**
  - Vehicle safety systems
    - Engine control unit
    - ABS braking system
    - Door locks
  - Non safety critical systems
    - Radio deck
    - HVAC system
  - The list goes on...

- **Programmable ECUs**
  - Allows MFGs to update firmware on ECUs

- **Average modern day car has ~70 ECUs**
Reverse Engineering CAN Messages

• **What we want to do:**
  – Manipulate CAN enabled vehicle components (Instrument Cluster)

• **Problem:**
  – Manufactures do not publish CAN message information about specific CAN components (ECUs)
    • Message IDs
    • Payload information (Byte offsets)

• **Solution:**
  – A method for visually correlating physical system interactions with identifiable patterns. (Humans are good at this)
  – Fuzzing (DANGER WILL ROBINSON!!!)
Reverse Engineering CAN Messages

• **Passively captured CAN traffic during a staged test run**
  – In this case it was a staged automotive collision.. 😊
  – Mini Cooper vs. GMC Envoy (Check out TU-CRRC website for killer crash videos)
  – Data capture lasted for roughly 90 seconds

• **Data Log gives us ~106,000 data entries of CAN messages**
Dearborn Group Format x15
Head on Crash for IATAI
Tue Sep 20 16:34:00 2011

Tue Sep 20 16:35:47 2011

106600
  Trigger Frame
  Absolute
  Timestamp, Channel, Frame ID, Frame Acronym, Protocol, DataCount, Data, Tx/Rx

<table>
<thead>
<tr>
<th>Time</th>
<th>Channel</th>
<th>Frame ID</th>
<th>Acronym</th>
<th>Protocol</th>
<th>DataCount</th>
<th>Data</th>
<th>Tx/Rx</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:55:49.668</td>
<td>810.2.316.316.CAN</td>
<td>STD.8.01 00 00 00 00 00 00 00. Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.668</td>
<td>960.2.336.336.CAN</td>
<td>STD.8.00 00 FE 02 6C 12 9C 89.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.669</td>
<td>210.2.322.322.CAN</td>
<td>STD.8.C0 61 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.669</td>
<td>440.2.153.153.CAN</td>
<td>STD.8.10 50 00 00 00 FF 00 80.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.669</td>
<td>690.2.1F0.1F0.CAN</td>
<td>STD.8.A0 20 0A 00 0A 00 0A 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.669</td>
<td>500.2.1F3.1F3.CAN</td>
<td>STD.8.80 80 00 FF 41 7F 00 0E.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.670</td>
<td>190.2.1F3.1F3.CAN</td>
<td>STD.8.00 00 00 FE FF 00 00.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.670</td>
<td>420.2.545.545.CAN</td>
<td>STD.8.12 00 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:55:49.670</td>
<td>660.2.565.565.CAN</td>
<td>STD.8.50 20 66 02 00 02 00 63.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:003</td>
<td>000.2.1F5.1F5.CAN</td>
<td>STD.8.60 80 00 00 80 E2 00 00.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:003</td>
<td>310.2.153.153.CAN</td>
<td>STD.8.10 50 00 00 00 FF 00 80.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:003</td>
<td>550.2.1F0.1F0.CAN</td>
<td>STD.8.A0 40 0A 00 0A 00 0A 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:003</td>
<td>790.2.1F3.1F3.CAN</td>
<td>STD.8.00 81 00 FF 41 7F 00 0E.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:004</td>
<td>040.2.1F3.1F3.CAN</td>
<td>STD.8.00 00 00 00 FE FF 00 00.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:005</td>
<td>620.2.316.316.CAN</td>
<td>STD.8.01 00 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:006</td>
<td>040.2.336.336.CAN</td>
<td>STD.8.00 00 FE 02 6C 12 9C 89.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:006</td>
<td>300.2.329.329.CAN</td>
<td>STD.8.C0 61 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:006</td>
<td>540.2.545.545.CAN</td>
<td>STD.8.12 00 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:006</td>
<td>780.2.565.565.CAN</td>
<td>STD.8.50 20 66 02 00 02 00 63.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:010</td>
<td>360.2.153.153.CAN</td>
<td>STD.8.10 50 00 00 00 FF 00 80.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:010</td>
<td>560.2.1F0.1F0.CAN</td>
<td>STD.8.A0 60 0A 00 0A 00 0A 00.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:010</td>
<td>600.2.1F3.1F3.CAN</td>
<td>STD.6.40 00 00 FF 41 7F 00 0E.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:011</td>
<td>060.2.1F3.1F3.CAN</td>
<td>STD.8.00 00 00 FE FF 00 00.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:011</td>
<td>390.2.1F5.1F5.CAN</td>
<td>STD.8.60 80 00 00 80 F2 04 05.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:015</td>
<td>630.2.316.316.CAN</td>
<td>STD.6.01 00 00 00 00 00 00 00.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:016</td>
<td>060.2.336.336.CAN</td>
<td>STD.6.00 00 FE 02 6C 12 9C 89.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:016</td>
<td>310.2.329.329.CAN</td>
<td>STD.8.C0 61 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:016</td>
<td>550.2.545.545.CAN</td>
<td>STD.8.12 00 00 00 00 00 00 00.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:016</td>
<td>780.2.565.565.CAN</td>
<td>STD.8.50 20 66 02 00 02 00 63.Rx.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:00:017</td>
<td>360.2.153.153.CAN</td>
<td>STD.6.10 50 00 00 00 FF 00 80.RX.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CAN Data Log

- Contained ~106,000 data entries
- Bash “cut –d. –f3 cooperheadion.txt | sort | uniq –c”
  - Only 15 Unique CAN IDs!

<table>
<thead>
<tr>
<th>Message Frequency</th>
<th>CAN IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>12706</td>
<td>153</td>
</tr>
<tr>
<td>12706</td>
<td>1F0</td>
</tr>
<tr>
<td>12706</td>
<td>1F3</td>
</tr>
<tr>
<td>9460</td>
<td>1F5</td>
</tr>
<tr>
<td>12707</td>
<td>1F8</td>
</tr>
<tr>
<td>8899</td>
<td>316</td>
</tr>
<tr>
<td>8899</td>
<td>329</td>
</tr>
</tbody>
</table>
Visually Identifying CAN Messages of Interest

Message ID 0x153 Vehicle Speed
0x153 Byte 2 CAN Message

Vehicle Speed (MPH)

Time (sec)
Reverse Engineering CAN Messages

- **Speedometer and Tachometer Message IDs**
  - 2 methods
    - For each message ID, plot data values vs. timestamp in order to determine physical significance.
    - Given possible CAN IDs, fuzz data fields until needles start moving

<table>
<thead>
<tr>
<th>CAN Message ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x153 Byte 2</td>
<td>Speedometer (Vehicle Speed)</td>
</tr>
<tr>
<td>0x316 Byte 3</td>
<td>Tachometer (Engine Speed)</td>
</tr>
<tr>
<td>0x329</td>
<td>Various indicator lights</td>
</tr>
<tr>
<td>0x61A</td>
<td>Controls the messages being displayed on the tachometer LED screen</td>
</tr>
<tr>
<td>0x61F</td>
<td>Tachometer along with various indicator lights</td>
</tr>
</tbody>
</table>
Building the CAN Clock and Network

- **CAN Bus**
  - 18 gauge wire
  - 2 x 120 ohms terminating resistors
  - 12V DC power source
  - Arduino Uno microcontroller
  - CAN Bus Shield
    - MCP2515 CAN controller
    - MCP2551 CAN transceiver
  - Mini Cooper Instrument Cluster
  - Real time clock module RTC (for clock mode)
CAN Clock Proof of Concept

• Talking CAN with Arduino
  – Arduino and CAN Controller Libraries
    • MCP2515 (Communication with CAN transceiver)
    • SPI (Used for communications between Arduino and CAN shield)

• 2 Modes of operation
  – Clock Mode
  – Demo Mode
Demo
Gaining Physical Access to CAN Bus

- Via OBD2
- Tapping the CAN bus (vampire tap)
  - Under the hood
  - Breaking a powered side view mirror
  - Etc.
- 0 to pwned for less then $100
  - Rogue Arduino CAN node
- Potential conspirators
  - Mechanics
  - Car Rentals
  - Coworkers/Family/Friends/Valets/Ex-girlfriends/etc.
Conclusion / Future Work

• Better access control between vehicle network components
  – ECU to ECU
  – OBD2 to ECU

• Applying conventional NIPS & firewall methods to CAN
  – Message anomaly prevention depending on context?
For more Information

• **TU Research**
  - [http://isec.utulsa.edu/](http://isec.utulsa.edu/)
  - [http://tucrrc.utulsa.edu/](http://tucrrc.utulsa.edu/) ← Check out our research and crash tests 😊
  - [http://tucrrc.utulsa.edu/canclock/](http://tucrrc.utulsa.edu/canclock/)

• **CAN Standards/Docs**
  - [http://www.sae.org/standards/](http://www.sae.org/standards/)
Questions??

• jason-staggs@utulsa.edu