CANvas: Fast and Inexpensive Automotive Network Mapping

Sekar Kulandaivel, Tushar Goyal, Arnav Kumar Agrawal, Vyas Sekar

Carnegie Mellon University
Do you know what’s going on in your car?

Electronic Control Unit (ECU)

ECU A  ECU B

ECU C  ECU D

*Koscher et al. IEEE S&P ‘10
*Checkoway et al. USENIX Security ‘11

It’s important to know what’s going on inside your car
Scenario 1: the shady mechanic

Need to identify ECUs in the car
Scenario 2: the radio from Craigslist

craigslist

Radio  Brakes

| Engine |

Need to know who sends each message
Scenario 3: the shut-down attack

*Cho et al., ACM CCS ’16

Need to know who receives each message
We need an automotive network mapper

1. Identify ECUs
2. Identify message sender
3. Identify message receiver(s)
Requirements for a practical tool

- Fast and inexpensive
- Vehicle-agnostic
- Minimally-intrusive
- Non-destructive
Why not ask the automaker?

Confidential database file of messages

Online mechanic subscription

Network inside a car can change
CANvas in a nutshell

A network mapper for cars that leverages message timing and error-handling mechanism

Generates a network map in <30 minutes with <$50 worth of hardware
Outline

• Motivating scenarios
• Background and mapping challenges
• System overview
• CANvas components
• Evaluation
• Conclusions
Controller Area Network (CAN) background

Each ID only sent by one ECU

ECU A
ID 1 @ 20ms
ID 2 @ 120ms

ID 2
ACK
DATA

ECU B
ID 3 @ 25ms
ID 4 @ 40ms

ECU C
ID 5 @ 600ms
ID 6 @ 900ms

priority
simultaneous

ID 2
ACK
CAN makes mapping difficult

ECU A

ECU B

ECU C

CANvas

Can’t tell which ECU is sender or receiver*

Bus traffic:
ID 1 @ t=0.104
ID 2 @ t=0.253
ID 3 @ t=0.350
ID 2 @ t=0.505
ID 3 @ t=0.697
ID 2 @ t=0.757
ID 2 @ t=1.009
ID 3 @ t=1.044

Priority

Simultaneous

ID 2
DATA
ACK
Outline

• Motivating scenarios
• Background and mapping challenges
• System overview
• CANvas components
• Evaluation
• Conclusions
CANvas design overview

1. Identify ECUs
2. Identify message sender
3. Identify message receiver(s)
The source mapping problem

Input:
ID 1 @ t=0.104
ID 2 @ t=0.253
ID 3 @ t=0.350
ID 2 @ t=0.505
ID 3 @ t=0.697
ID 2 @ t=0.757
ID 2 @ t=1.009
ID 3 @ t=1.044
ID 1 @ t=1.114
ID 2 @ t=1.250
ID 3 @ t=1.391

Timestamped traffic log

Source mapping

<table>
<thead>
<tr>
<th>Src. ECU</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>3, 4</td>
</tr>
</tbody>
</table>
Insight: clock offset as a unique identifier

*Cho et al., USENIX Security ‘16

Prior work for IDS

• Clock offset is unique
• Track offset per ID
Limitations: prior work is not sufficient

Not robust to noise in the period

Period-dependent

ECU Y

ID 262 @ 20ms
ID 4C8 @ 980ms
ID 521 @ 300ms
Idea: compare offset at hyper-period

Hyper-period removes period-dependence

ID 1

18

ID 2

6

12

18

24

30

36

42

48

54

Match?

Match?

Match?
Approach: pairwise comparison over time

Hyper-period is period-independent

Measure over time to reduce effect of noise

Practical challenges discussed in paper

Input:
ID 1 @ t=0.104
ID 2 @ t=0.253
ID 2 @ t=0.505
ID 2 @ t=0.757
ID 2 @ t=1.009
ID 1 @ t=1.114
ID 2 @ t=1.250

Match?

ECU Y
ID 1
ID 2
CANvas design overview

1. Identify ECUs
2. Identify message sender
3. Identify message receiver(s)
The destination mapping problem

<table>
<thead>
<tr>
<th>ID</th>
<th>Dst. ECUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A, C</td>
</tr>
<tr>
<td>2</td>
<td>B, C</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
</tr>
</tbody>
</table>

**Source map**

<table>
<thead>
<tr>
<th>Src. ECU</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
</tbody>
</table>

**Physical bus**

**Destination mapping**

**Destination map**
Approach: isolate each ECU

ECU A  ECU B

ECU C

Isolate an ECU to guarantee who sent ACK

ACK indicates some ECU received

<table>
<thead>
<tr>
<th>ID</th>
<th>Dst. ECUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>???</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Dst. ECUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>
Insight: shut-down via error-handling exploit

*Cho et al., ACM CCS ‘16

Prior work for a DoS attack
• Exploit error-handling by causing errors

Not intended to be robust – attack needs just one success

Refer to paper for limitations and our idea for isolation
Outline

• Motivating scenarios
• Background and mapping challenges
• System overview
• CANvas components
• Evaluation
• Conclusions
Evaluation setup

2009 Toyota Prius

2017 Ford Focus

Junkyard ECUs

Ford Engine ECUs

Synthetic topologies

Arduino Due
Key takeaways

- Fast: <30 minutes
- Inexpensive: <$50
- Vehicle-agnostic: Standard CAN
- Minimally-intrusive: OBD-II port
- Non-destructive: No dash lights
Finding an unexpected ECU in a ‘09 Prius

ECU installed during a past vehicle modification

Skid Control

Yaw Rate

Steering Angle

Hybrid

Engine

Power Steering

Battery

Gateway

???
‘17 Focus ECUs enable the shut-down attack

Both Prius and Focus had no filter on what messages an ECU could receive.
CANvas limitations

**Adversarial evasion**
- Timing-aware attacker
- Intentional timing alteration

**Avoiding permanent damage**
- Resetting dash lights
- Limp-home mode

**Multiple CAN buses**
- Accessing unexposed buses

**Message acceptance filter**
- Vendor-specific approaches

**Non-transmitting ECUs**
Conclusions

- Network inside cars can change
- **CANvas**: a network mapper that tells us what’s going on in a car
- Mapping CAN is non-trivial → lack of source or destination info
- Prior work did not solve mapping goals
- A fast and inexpensive design focused on practicality
- Real-world demonstration on two vehicles
- Serves as a basis for many other security applications

https://github.com/sekarkulandaivel/canvas