Mobile Vehicle Security Bus

By: SD MAY 23-14

Presentation Outline

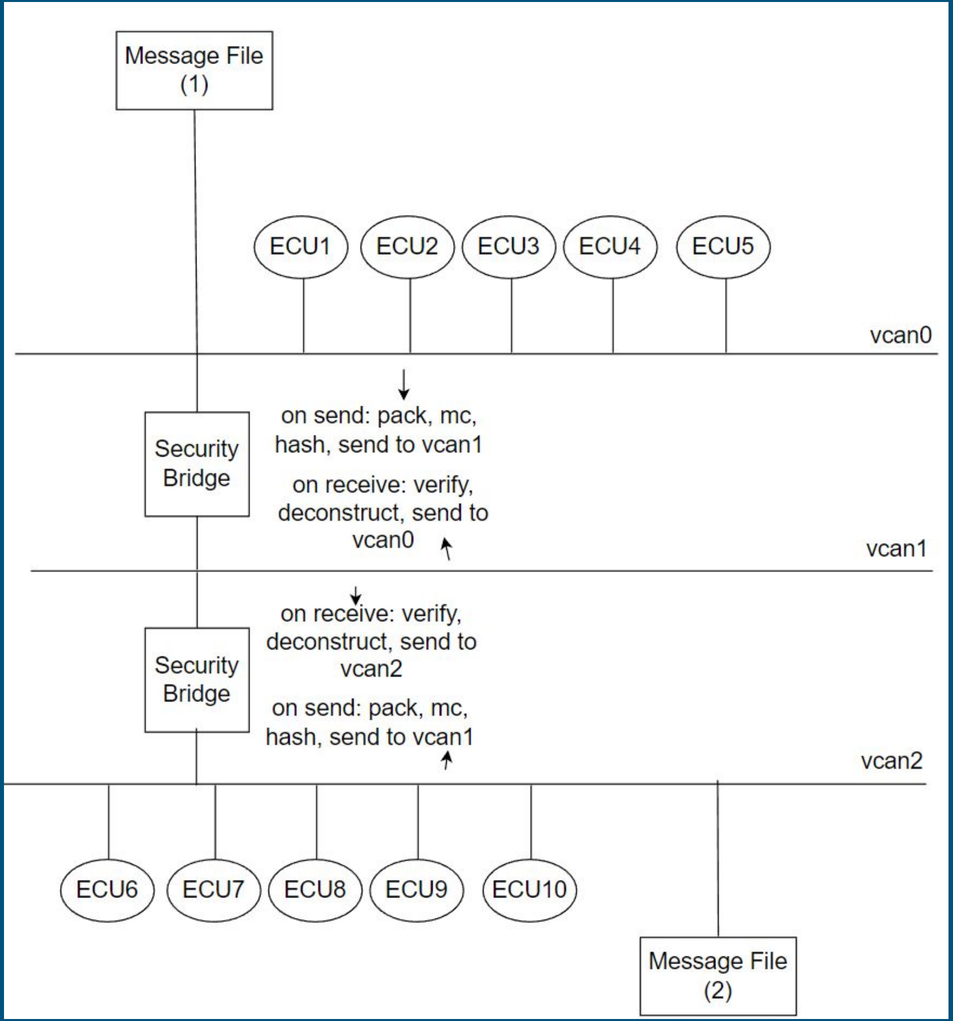
- Intro / Background
- Implementation
- Accomplishments
- Key Contributions
- Challenges + Solutions
- Future Work
- Conclusion
- Live Demonstration

Introduction

- Jeep Hack 2015 (Chrysler UConnect App)
  - Public stunt between journalist + two researchers
  - Root access on car and CAN bus (ECU communications)
  - “Hey Chrysler, please fix this.”
- Fix?
  - Kind of…
    - DIY software download to USB ***or***
    - Bring to dealership

General Architecture

- Message Files
- ECUs
- Virtual Can Buses (vcanX)
- Bridge

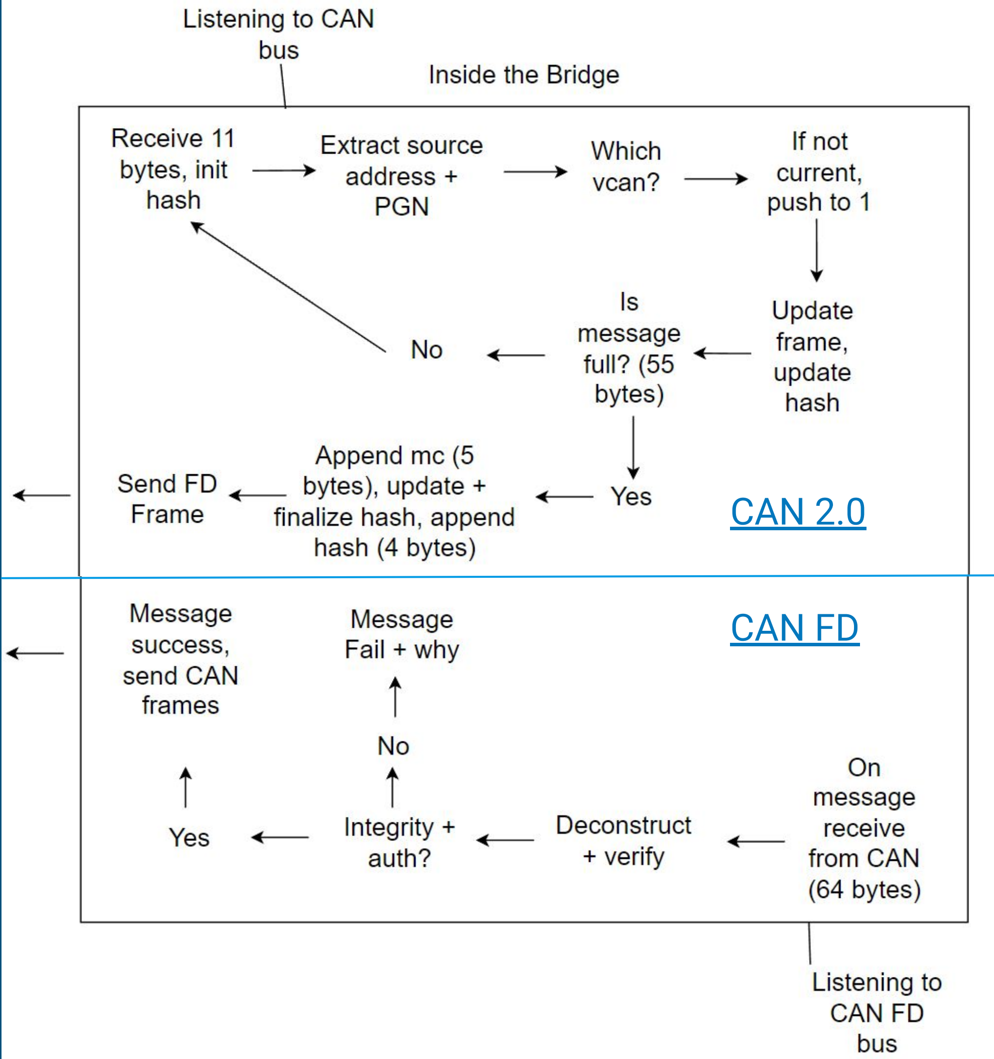


Message File (1)
ECU1
ECU2
ECU3
ECU4
ECU5
vcan0
Security Bridge
on send: pack, mc, hash, send to vcan1
on receive: verify, deconstruct, send to vcan0
vcan1
Security Bridge
on receive: verify, deconstruct, send to vcan2
on send: pack, mc, hash, send to vcan1
vcan2
ECU6
ECU7
ECU8
ECU9
ECU10
Message File (2)

Bridge Architecture

- Listening Can Bus
  - Extract fields
  - Determine destination
  - Pack + Hash
  - Is full?
    - If yes, add mc, hash, send
    - If no, continue listening

- Listening FD Bus
  - Deconstruct, verify
  - Security functions
    - If yes, send messages
    - If no, fail + reason



Listening to CAN bus

Inside the Bridge

Receive 11 bytes, init hash

Extract source address + PGN

Which vcan?

If not current, push to 1

Update frame, update hash

Is message full? (55 bytes)

No

Yes

Append mc (5 bytes), update + finalize hash, append hash (4 bytes)

Send FD Frame

[CAN 2.0](CAN 2.0)

[CAN FD](CAN FD)

On message receive from CAN (64 bytes)

Deconstruct + verify

Integrity + auth?

No

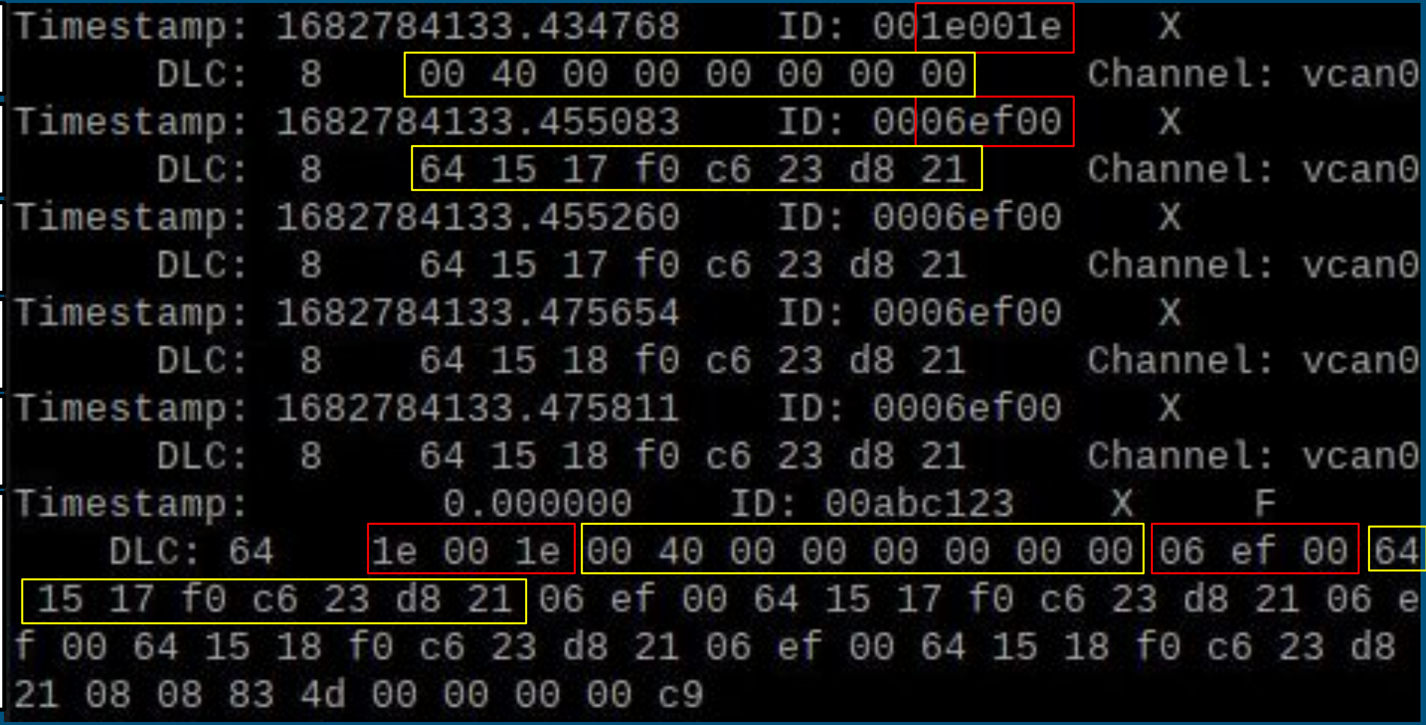
Message Fail + why

Yes

Message success, send CAN frames

Listening to CAN FD bus

Building CAN FD Frames



CAN Frame 1

2

3

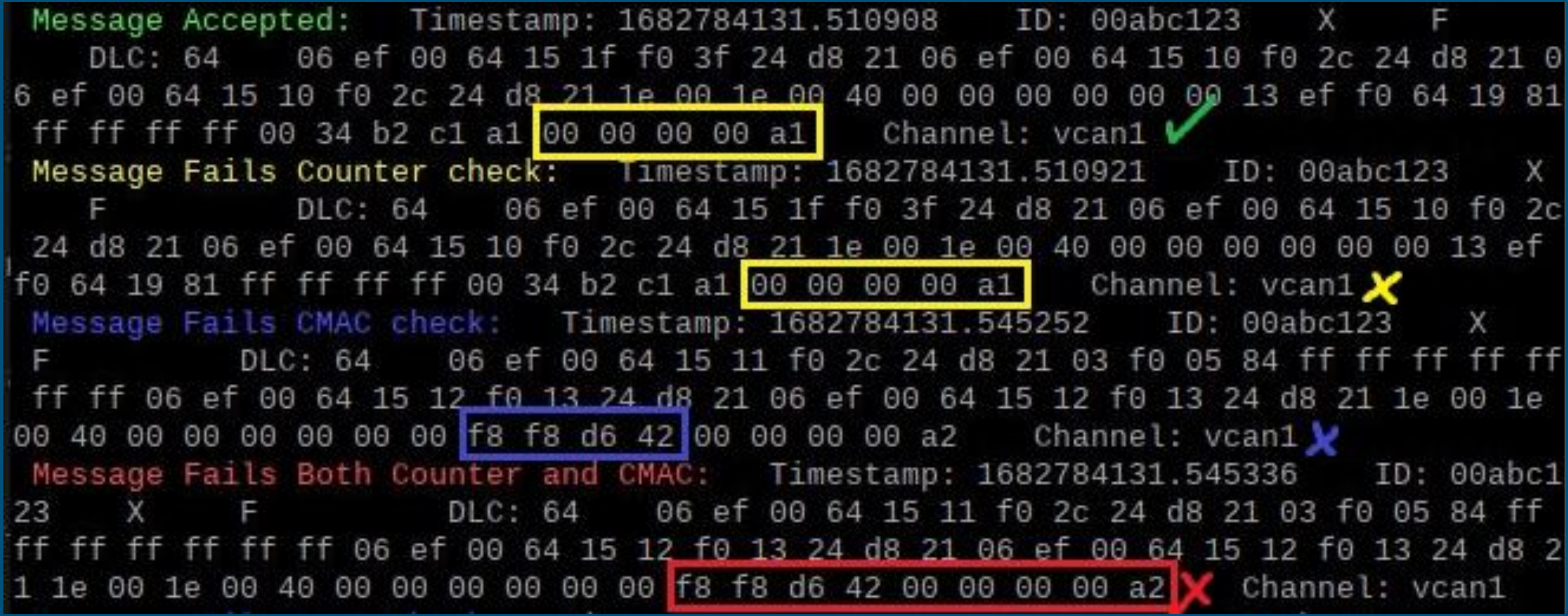
4

5

CAN FD Frame

```
Timestamp: 1682784133.434768        ID: 001e001e    X
          DLC:  8    00 40 00 00 00 00 00 00       Channel: vcan0
Timestamp: 1682784133.455083        ID: 0006ef00    X
          DLC:  8    64 15 17 f0 c6 23 d8 21       Channel: vcan0
Timestamp: 1682784133.455260        ID: 0006ef00    X
          DLC:  8    64 15 17 f0 c6 23 d8 21       Channel: vcan0
Timestamp: 1682784133.475654        ID: 0006ef00    X
          DLC:  8    64 15 18 f0 c6 23 d8 21       Channel: vcan0
Timestamp: 1682784133.475811        ID: 0006ef00    X
          DLC:  8    64 15 18 f0 c6 23 d8 21       Channel: vcan0
Timestamp:          0.000000      ID: 00abc123    X      F
        DLC: 64    1e 00 1e 00 40 00 00 00 00 00 00 06 ef 00 64
 15 17 f0 c6 23 d8 21 06 ef 00 64 15 17 f0 c6 23 d8 21 06 e
f 00 64 15 18 f0 c6 23 d8 21 06 ef 00 64 15 18 f0 c6 23 d8
21 08 08 83 4d 00 00 00 00 c9
```

Accepting / Rejecting Messages



```
 Message Accepted:   Timestamp: 1682784131.510908    ID: 00abc123    X     F
    DLC: 64     06 ef 00 64 15 1f f0 3f 24 d8 21 06 ef 00 64 15 10 f0 2c 24 d8 21 0
6 ef 00 64 15 10 f0 2c 24 d8 21 1e 00 1e 00 40 00 00 00 00 00 00 13 ef f0 64 19 81
 ff ff ff ff 00 34 b2 c1 a1 00 00 00 00 a1    Channel: vcan1 ✓
 Message Fails Counter check:   Timestamp: 1682784131.510921    ID: 00abc123    X
    F           DLC: 64     06 ef 00 64 15 1f f0 3f 24 d8 21 06 ef 00 64 15 10 f0 2c
 24 d8 21 06 ef 00 64 15 10 f0 2c 24 d8 21 1e 00 1e 00 40 00 00 00 00 00 00 13 ef
f0 64 19 81 ff ff ff ff 00 34 b2 c1 a1 00 00 00 00 a1    Channel: vcan1 ✗
 Message Fails CMAC check:   Timestamp: 1682784131.545252    ID: 00abc123    X
 F           DLC: 64     06 ef 00 64 15 11 f0 2c 24 d8 21 03 f0 05 84 ff ff ff ff ff
 ff ff 06 ef 00 64 15 12 f0 13 24 d8 21 06 ef 00 64 15 12 f0 13 24 d8 21 1e 00 1e
00 40 00 00 00 00 00 00 f8 f8 d6 42 00 00 00 00 a2    Channel: vcan1 ✗
 Message Fails Both Counter and CMAC:   Timestamp: 1682784131.545336    ID: 00abc1
23    X     F           DLC: 64     06 ef 00 64 15 11 f0 2c 24 d8 21 03 f0 05 84 ff
ff ff ff ff ff ff 06 ef 00 64 15 12 f0 13 24 d8 21 06 ef 00 64 15 12 f0 13 24 d8 2
1 1e 00 1e 00 40 00 00 00 00 00 00 f8 f8 d6 42 00 00 00 00 a2 ✗ Channel: vcan1
```

Work Accomplishments

- Messages read from file and packed into 64 byte frames.
- Messages sent across multiple CAN busses.
- CMAC and counter validation, harmful or unexpected messages are rejected
- Message routing based on PGN value to reach desired destination.
- Bidirectionality for proper message control.

Key Contributions

Ryan S

- Packing/unpacking CAN FD frames, sending/receiving frames, validating FD frames, timestamp delays

Ryan C

- AES CMAC, Monotonic Counter, Bidirectionality

Cody

- Sending frames, testing tools, git and codebase organization, example ECU design

Levi

- SocketCAN setup, simulated data flow, helped with sending/unpacking of FD frames

Key Contributions

Josue

- Helped develop ideas and prototype for packing CAN frames, Implemented a routing algorithm, and brainstormed ideas for getting full duplex communication between bridges

Drake

- ECU, Helping Pack/Unpack CAN FD Frames, Routing, Bidirectionality

Riley

- Contributed to the ECU, helped research SocketCAN, Send/Receive/Pack CAN FD Frames, CMAC Debugging, Bidirectionality

Challenges & Solutions

- New concepts for all of us
- Only 1 Cybersecurity Engineering major
- C had minimal documentation for SocketCAN
- Switched to Python at 4 weeks into this semester
- Communicating with a large group

Future Work

- Encryption (to provide confidentiality)
- Expanding the number of Bridges + ECUs in the CAN network
- Physical testing on real hardware + vehicle
- Physical live demo

Conclusion

- A solution for vehicle security is possible
  - Of continued importance as CAN will likely be used in vehicles for many years
  - Current and older vehicles can benefit as well
    - Our approach is theoretically backwards compatible with the older CAN standard

- There's still a lot of work to be done
  - Implementing security mechanisms are challenging due to CAN's limitations
  - Vehicle manufacturers would need to update their vehicles and manufacturing processes