Cheap Tools for Hacking Heavy Truck

By Haystack and Six Volts

What we are going to talk about

- Heavy Trucks: similarities and differences from cars
- R&D Problems: Trucks are expensive and the workaround
- Networking Protocols and Standards
- New Hardware Tools
- Adventures in truck hacking

Some Quick Notes

- We assume that you are familiar with basic vehicle networking concepts – e.g. there are computers in cars and they use a network
- We also assume you are familiar with the idea that you can do bad things once you are on those networks
- We are leaving out LOTS of details for time reasons
  - Check out our github (final materials posted by end of next week)
- Safety Disclaimer: Moving vehicles are dangerous. Do not fuzz a rental vehicle while driving, or do anything else stupid

Trucks vs. Cars

- “Trucks” are really any heavy vehicle including but not limited to Over-the-road Semis, Vocation Trucks, Fire Engines, Busses, some Armored Personnel Carriers, Ambulances, Armored cars, boats, diesel generators and agricultural equipment.
  - Exception: Diesel Pickup Trucks (these act more like cars)
- Nearly all heavy vehicles use Diesel engines.
- Different On-board Diagnostic and Networking Standards (J1939/J1708)
  - RP1210 governs workstation->adapter interface

Truck Economics

- Many components from different manufacturers are interchangeable (engine, brakes, etc)
  - Example: Navistar/International Truck can be purchased with either a Cummins or International/Navistar Engine (and Previously CAT also)
  - This means that products from different manufacturers have to be interoperable
- Many trucks operate in Fleets, typically as homogenous as possible
- The industry is incredibly data hungry, lots of data are stored and transmitted
- Data hungry industry + lots of miles = trucks spend (comparatively) more time connected to diagnostic computers

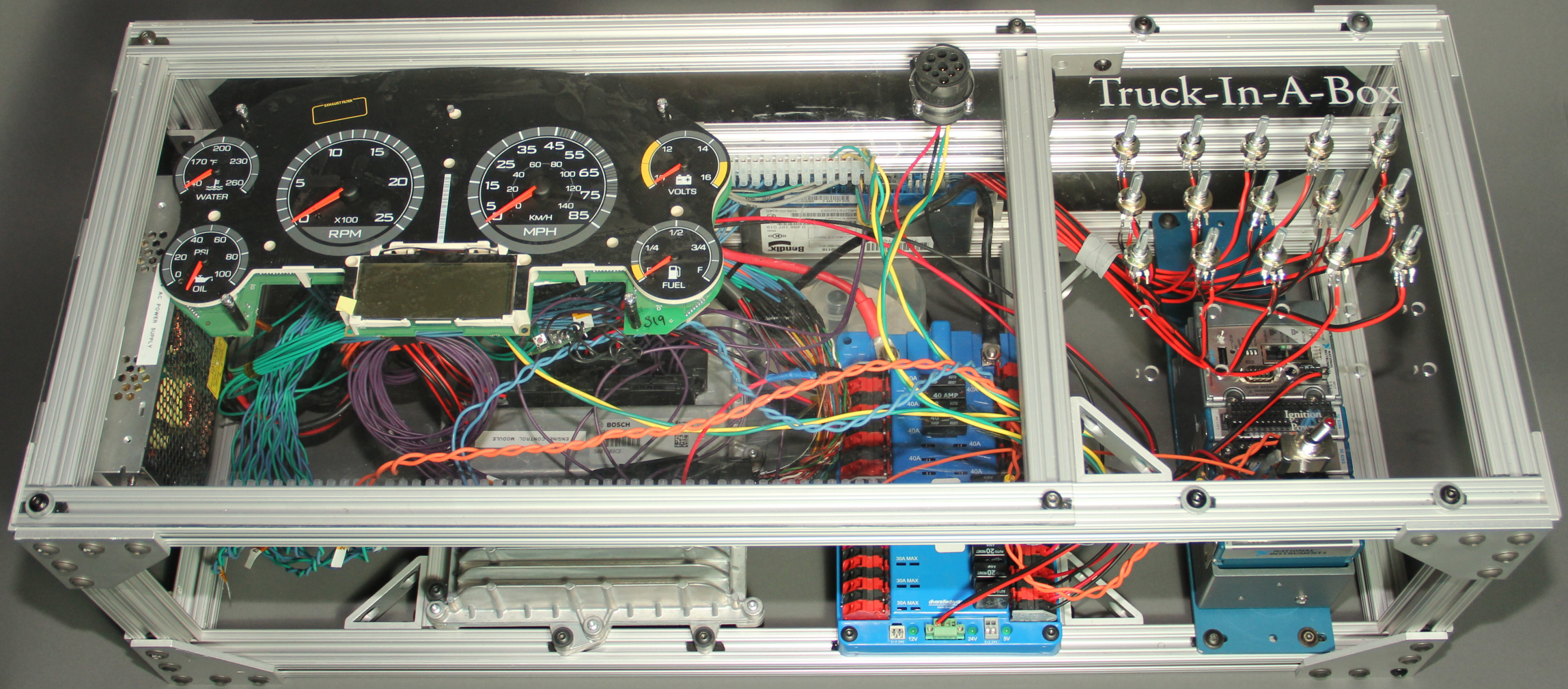
Telematics Attack Surface

- Used for driver logs, navigation, communications between the fleet and driver and emergency info
- They use the cellular network to connect to the provider
- Connects directly to CAN/J1939 bus, J1708 networks
- Many run embedded versions of Windows

Trucks are EXPENSIVE

- A new Truck can cost over $100,000. Ouch.
- For the aspiring hacker - They are big, hard to store, hard to drive and expensive to operate.
- So we didn’t have one (and still don’t)…
- …so how do we experiment? We built a thing.

Truck-In-A-Box, Version 1.0



Truck-In-A-Box (TIB)

- We bought an ECM (Engine Control Module) and built the electronics around it such that it functioned enough for analysis (Key-on, engine off)
- The first one took 6+ months and cost over $10,000
  - However, that's less than the cost of a truck
- Since then, we've built over a dozen of these full-size versions
- Later, compressed the concept into small box with one or two PCBs that hook up to the ECM for each make/model

Truck-In-A-Box Concepts

- Recreate the Vehicle Networks, J1939 (CAN), J1708 (RS485-ish)
- Fake Passive sensor signals (usually just a set voltage or resistance)
- Fake Simple Active Signals (PWM for Accelerator Pedal)
- Generate Complex Analog Signals (Vehicle Speed)


Agilent
Acquisition
Normal
200MSa/s
Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1
Measurements
Freq(1):
903.44Hz
Pk-Pk(1):
33V
Measure Current Mean Min Max Std Dev Count
Freq(1): 903.44Hz 894.69Hz 293.07Hz 913.41Hz 43.599Hz 886
Pk-Pk(1): 33V 31.108V 14V 62V 10.407V 887
Measurement Statistics Menu
Display On
Reset Statistics
Transparent
Relative σ
Max Count
∞

Networking Protocols and Standards

- 2 main protocols: SAE J1939 and J1708
- J1708 is the old one (1985)
  - Based on 9600 baud UART
  - J1587 operates on top of it (transport layer)
- J1939 is the new one (?)
  - Physical & data link layers are 250K CAN
  - Addressing, transport, etc
- ISO15765 also used, but only for diagnostics comms
- (details in whitepaper)

J1708 basics

- Messages are time delimited
- MIDs and PIDs
- Mostly older trucks will have only J1708
  - Some newer ones will have components using it
  - Also, gliders
- Data link escape for proprietary comms (PID 0xFE)
- Message fragmentation & reliable delivery (J1587)

J1939 Basics

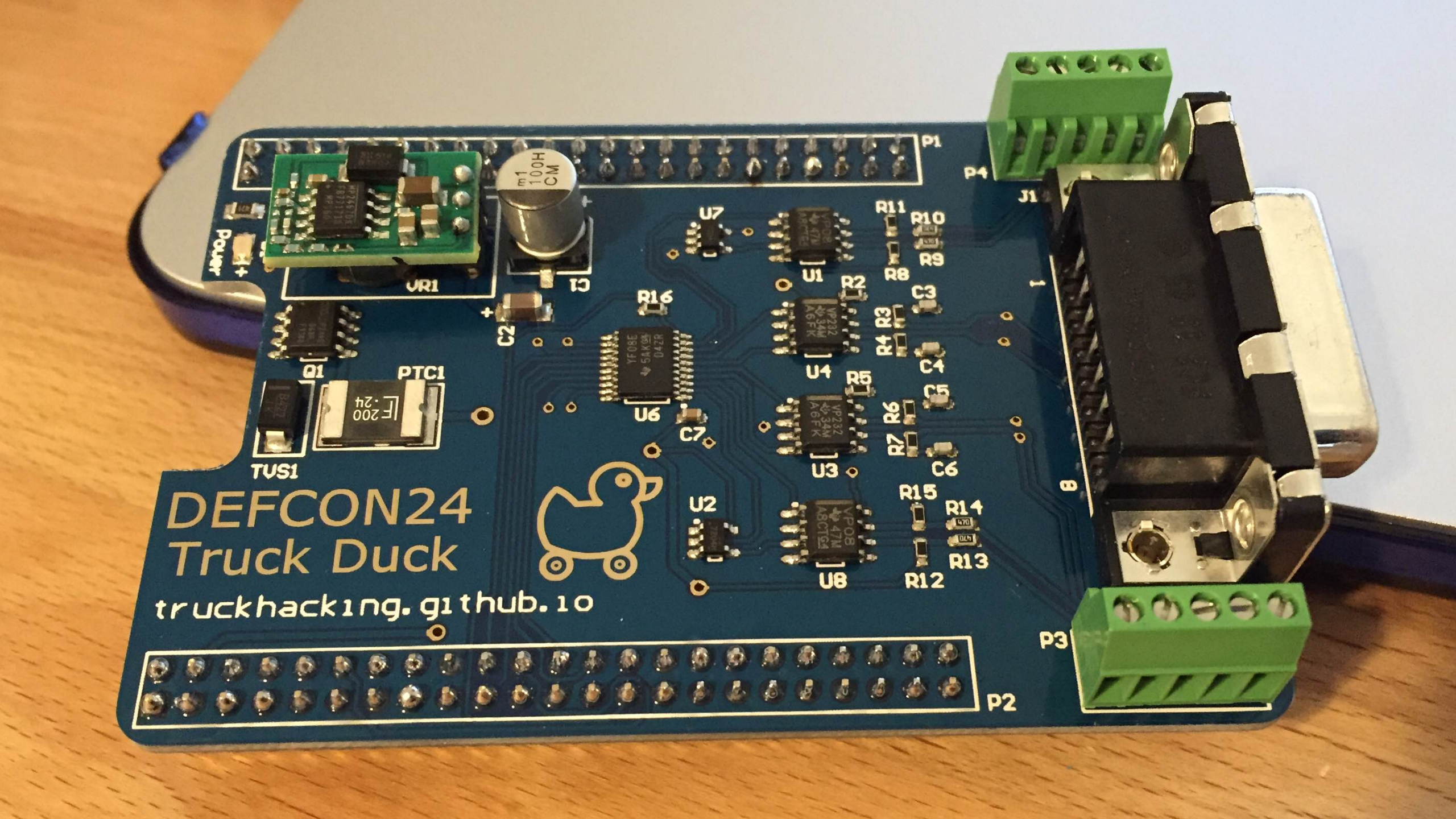
- 250k CAN (500k in the near-ish future)
- Extended CAN ID broken into source, (maybe) destination, etc
- Address management, transport, message fragmentation
  - There’s a bajillion different J1939 standards
- Also a PGN or two reserved for proprietary comms

DLC basics

- Vehicle diagnostic link connector/adapter (DLC/DLA)
  - Similar in purpose to OBD-II scan tools
  - Basically USB/Serial/Ethernet -> J1939/J1708 bridge
- ATA TMC RP1210 governs functions exposed by vehicle diagnostic adapters (VDAs)
- Observing RP1210 driver calls is an excellent strategy for dynamic analysis of OEM software
  - RP1210_SendMessage and RP1210_ReadMessage for actual message traffic

Truck Hacking Tools: Truck Duck

- Cape for a BeagleBone
  - Hardware for CAN and J1708
  - 2 channels of each for potential filtering/modification purposes
- We also have a software stack for doing comms
  - J1939 kernel extensions (plus J1939-enabled Python build)
    - Shoutout to Kurt Van Dijk
  - Homegrown J1708 implementation using AM335x PRU


DEFCON24
Truck Duck
truckhacking.github.io
P1
P2
P3
P4
J1
VR1
C1
C2
C3
C4
C5
C6
C7
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
U1
U2
U3
U4
U6
U7
U8
Q1
PTC1
TVS1
Power

Truck Hacking Tools: RP1210 Tracer

- Currently, best option is to buy a DLC whose driver has debug logging capability
  - Only one we know of costs ~$700
- We rolled our own RP1210 API tracer that logs results of RP1210 function calls
- Works with any DLC (including cheap eBay clones)
- Allows you to decrypt/translate on the fly

But what...is it good for?

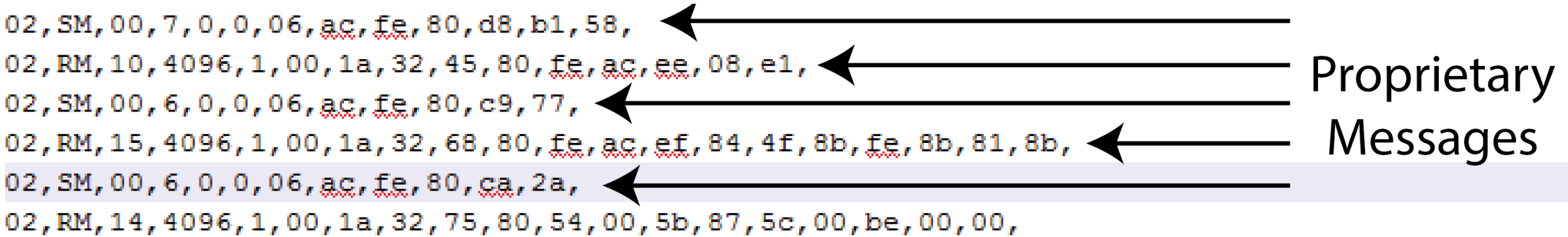
Misconfiguration of Truck ECM

- Wanted a viable attack that could actually have some impact
- But we have no actual truck to test it on
  - Need something that we can do *in vitro*
- Solution: malicious ECM misconfiguration

Screwing with engine parameters

- Most engine parameter configuration is done over proprietary protocol extensions
- We super promised not to give *too* many specifics
- Demonstration of what is possible with TruckDuck & API tracing

First: what does the traffic look like?



Initial notes from analysis

- Same process yields different network traffic every time
  - Ergo: it’s obfuscated/encrypted
- Messages that appear to do same thing are the same length
  - So it’s not *too* obfuscated
- <here’s where I yadda-yadda-yadda past some futzing around with DotPeek and IDA>

Protocol Details: security setup

- This exchange precedes all data downloads
- First three bytes: proprietary exchange between ECM and DLC
- Next byte: identifies security exchange
- Last byte: low nibble is device's key

```
02,SM,00,6,0,0,06,ac,fe,80,f0,02,
02,RM,09,5000,1,00,07,59,5b,80,fe,ac,f0,f5,
```

Protocol details: reads and writes

- msg[3], high nibble: command code
  - 0xC0: Encrypted Read
  - 0xD0: Encrypted Write
  - 0xE0: Encrypted R/W Response
- Low nibble: message code
  - “pre-shared key” + (code % 4) is an index into a char[]
- char ^ msg_byte is encrypted

```
ac,fe,80,d8,b1,58,
80,fe,ac,ee,08,e1,
ac,fe,80,c9,77,
80,fe,ac,ef,84,4f,8b,fe,8b,81,8b,
```

Let’s decrypt

- Used RP1210 API tracer to decrypt these proprietary messages on-the-fly and log them
- Pattern is much easier to deduce now

```
SM ac,fe,80,c1,f
RM 80,fe,ac,e7,f,c2,0,75,0,a,0
SM ac,fe,80,c2,60
RM 80,fe,ac,e8,60,32,31,35,20,20,20,20,20,20,20
SM ac,fe,80,c3,51
RM 80,fe,ac,e9,51,4e,58,53,32,30,36,35,34
SM ac,fe,80,c4,50
RM 80,fe,ac,ea,50,32,38,39,36,36,31,38,32,4a,58
SM ac,fe,80,c5,11
RM 80,fe,ac,eb,11,32,39,32,36,39,36,30,2d,30,30
SM ac,fe,80,c6,a
RM 80,fe,ac,ec,a,46,45,42,30,36
```

What can we do with this?

- Anything with direct bus access can reconfigure engine fairly easily
- Injecting traffic onto the vehicle bus directly requires physical access
  - Could just as easily cut the brake lines or something
- Possible non-boring attack vector: compromised telematics unit
- But let’s do more...

Hijacking OEM software

- Software used in day-to-day operations of fleets
  - Diagnostics
  - Maintenance tracking
  - Driver behavior
  - Trip information
- Over-the-road semis spend *a lot* more time connected to computers than passenger vehicles
- Viable attack vector for broad range of fleet vehicles

API tracer repurposed

- We’re already reading and decrypting ReadMessage and SendMessage function calls on the fly.
- We *should* be able to read, decrypt, modify, and re-encrypt on the fly as well
- Let’s see what that looks like

“Rootkit” demo video

- Screencast b/c showing the ECM would give away brand
- Sorry about the “free version” watermark

Future Work

- RP1210 driver for Truck Duck
  - Allow for easier traffic modification
  - Less costly than even eBay adapter clones
- More Advanced PC-Side Attack
  - Doesn't require as much user input
- Deeper firmware analysis of ECMs
  - Is it exploitable? Let’s find out!
  - Can go beyond simple spoofing of proprietary messages.

Do you want to know more

- truckhacking.github.io will have information/software downloads
  - When we get around to putting them up
- Follow us on twitter at @six_volts and @haystack_ia