ENUMERATING VULNERABILITIES IN AN ECU

HANNAH SILVA

ABOUT ME

- Senior Security Consultant at Leviathan Security
- Application security enthusiast
- 5 years of heavy vehicle security research experience
- Instructor and mentor for CyberTruck Challenge
- CHV volunteer
- Bachelor's in Computer Science

SECURITY ASSESSMENT ON AN ECU

- Threat model & attack surface
  - Remote attack vectors
  - Severity of compromise
- Get connected
- Discovery phase
  - Features, proprietary protocols, diagnostics…
- Testing all functionality that accepts input
  - Authentication/authorization on sensitive functions?
  - Input validation and handling
  - Logic bypass

TESTING FRAMEWORK

- TruckDevil is an open-source testing framework where collaborators can add various kinds of modules
- It's in early development

github.com/LittleBlondeDevil/TruckDevil

```
Welcome to the truckdevil framework
(truckdevil)help

Documented commands (type help <topic>):
========================================
add_device  help  list_device  list_modules  run_module

(truckdevil)list_modules
custom
ecu_discovery
j1939_fuzzer
read_messages
send_messages
(truckdevil)
```

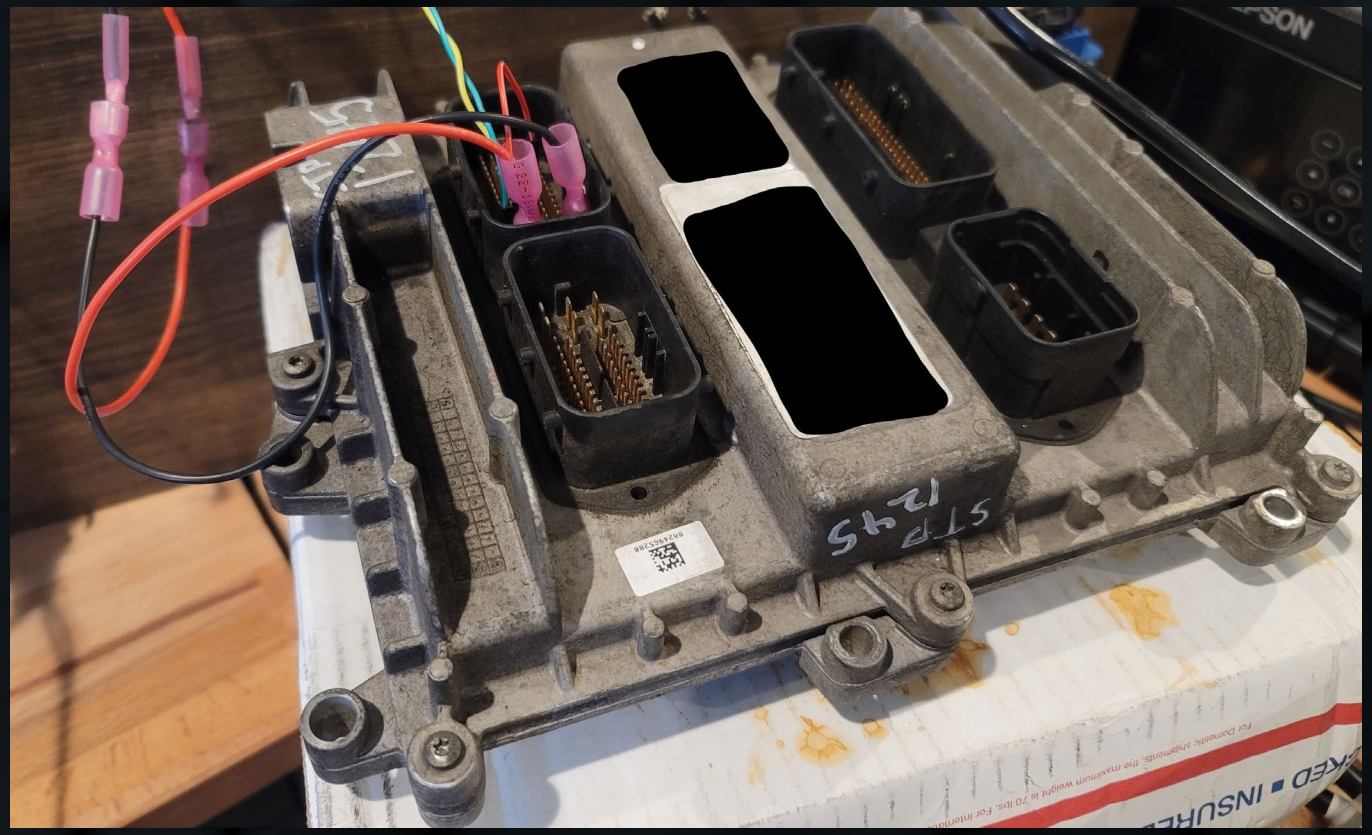
SET UP SOCKETCAN

```
$ sudo ip link show # should report can0

$ sudo ip link set can0 down

$ sudo ip link set can0 type can bitrate [250000 or 500000]

$ sudo ip link set can0 up
```

INSTALL TRUCKDEVIL

```
$ sudo apt install python3.10-venv
$ git clone https://github.com/LittleBlondeDevil/TruckDevil.git
$ cd TruckDevil
$ python3 –mvenv venv
$ source ./venv/bin/activate
$ python3 setup.py install
$ cd truckdevil
$ python3 truckdevil.py
```

THE TARGET ECU

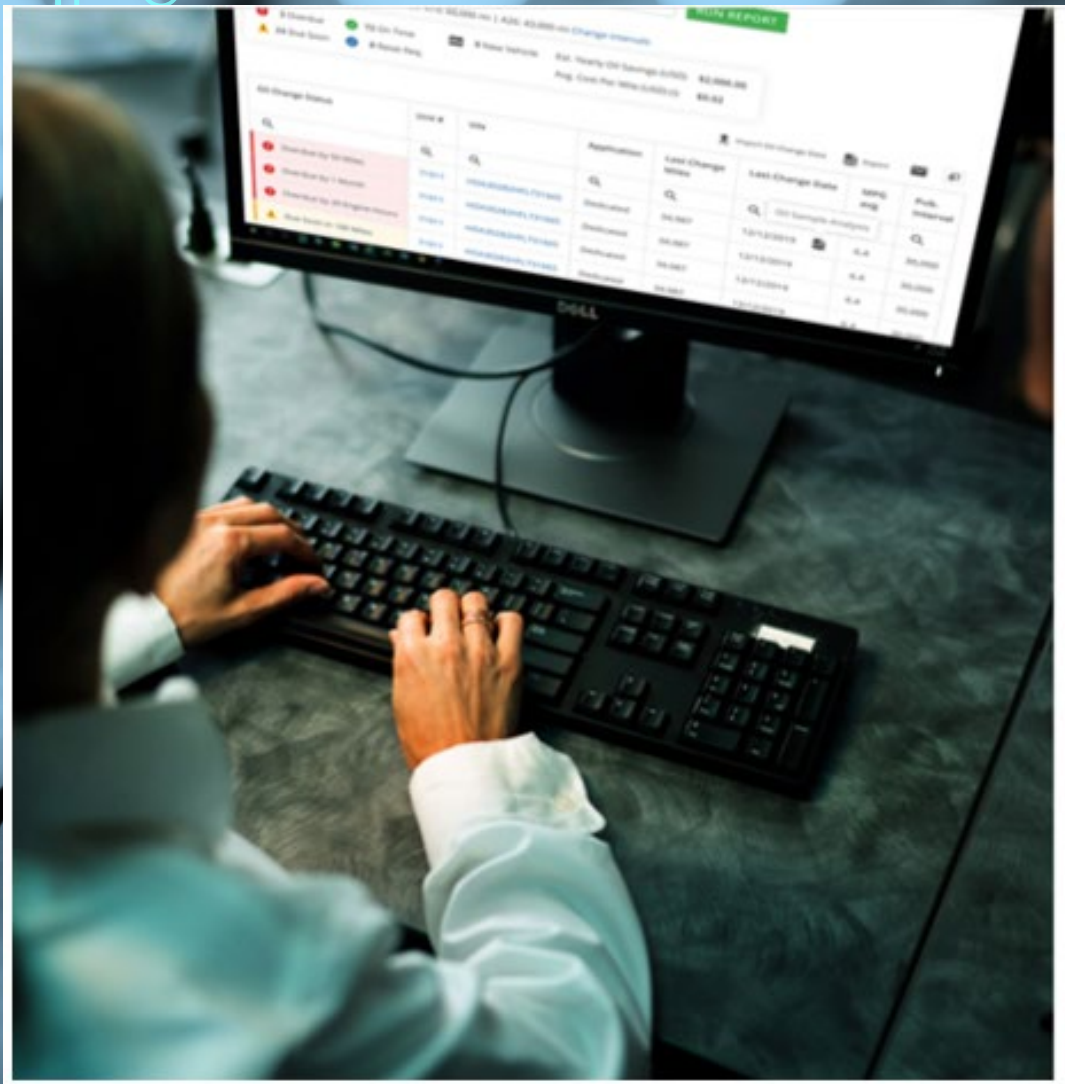


THREAT MODELING

- Exploitability
  - Remote attack vectors (telematics systems, ELDs, over-the-air programming)
  - Mitigations (isolated network, gateway, authentication)
- Impact of compromise
  - Confidentiality, integrity, availability

TARGET ECU THREAT MODEL

- Manufacturer's website lists convenience features of their vehicles
  - Built-in telematics devices standard on all newer vehicles
  - Advanced remote diagnostics
  - Over-the-air programming
  - Fleet health maintenance
  - TPMS reporting



[redacted]

Google Facebook

or

Email Address

FORGOT YOUR PASSWORD?

Password

SIGN IN

Don't have an account? SIGN UP NOW

Email Address

SEND VERIFICATION CODE

New Password

Confirm New Password

First Name

Last Name

Cancel

* Required Field

Your Information

Verify or enter your personal information below.

User ID *
janedoe4953@gmail.com

First Name *
Jane

Last Name *
Doe

Email Address *
janedoe4953@gmail.com

Phone Number *
(713)555-5555

ZIP / Postal Code *
77001

Country *
United States of America

Carrier / Company Information

Search for an existing carrier before attempting to enter a new carrier.

Search for Existing Carriers/Companies

DOT Number
DOT Number

Carrier / Company name
walmart

Search

Reset

	DOT Number	Carrier / Location	Match Quality
○		Enroll a new company.	New
○	63585	WAL-MART TRANSPORTATION LLC / **Walmart** Fleet Services BENTONVILLE, US-AR	Partial

Enroll

	DOT Number	Carrier / Location	Match Quality
◉		Enroll a new company.	New
○	63585	WAL-MART TRANSPORTATION LLC / **Walmart** Fleet Services BENTONVILLE, US-AR	Partial

DOT Number (Optional)
DOT Number

Carrier / Company name *
JaneDoeTrucking

Company Country *
United States of America

Primary Street Address *
123 Rainbow Drive

Additional Address Information

Company ZIP Code *
77001

City / Town *
Houston

State / Province *
Washington

Company Email Address *
janedoe4953@gmail.com

Company Phone Number *
(713)555-5555

Time Zone *
Central (Chicago, Winnipeg, Mexico City)

[redacted]

Enroll

JaneDoeTrucking Enrollment Pending [redacted] Inbox ×

[redacted]

to me

[redacted]

Hi Jane Doe,

Thank you for choosing [redacted]
We have received your enrollment request.

We are actively working on creating JaneDoeTrucking and enrolling you as an administrator.
You will receive an email once your company is approved and activated.

JaneDoeTrucking Is Now Available [redacted] Inbox ×

[redacted]

to me

[redacted]

Hi Jane Doe,

Thank you for choosing [redacted]
An account for JaneDoeTrucking has been created and is now available.

Login
To login to JaneDoeTrucking click the below link and enter your credentials.
https://[redacted]

Remember to login using the same method you used to create your user login. For help logging in, click this link: http://bit.ly/

Dashboard Vehicle Scans Mapping Reports Admin

Dashboard

Data as of 1:06 PM today

JaneDoeTrucking - Locations

1 location selected

Last 24 hours

Include: ☑ At dealership ☑ At known location ☑ With inactive fault codes ☑ No fault codes Apply

0 Results
out of 0

Fault Severities

Stop Now	0
Service Immediately	0
Service Soon	0
No Actionable Faults	0

Fault Indicators

Derate Condition	0
Maintenance Related	0
Safety Related	0

Telematics (Includes ALL Vehicles)

[redacted]
IDLE MILES

[redacted]
BRAKING EVENTS

[redacted]
ACCELERATING EVENTS

Currently Showing: 0 Vehicles
* Live/Speed Tracking

Unit/Chassis United States All Vehicles

Accounts for:

JaneDoeTrucking

Locations | Watch Lists

Search

- JaneDoeTrucking
  - JaneDoeTrucking
    Houston

JaneDoeTrucking

JaneDoeTrucking

General | Users (1) | Vehicles (1) | Asset Verification (0) | Devices (0) | Dealers (0) | Telematics (1) | Import

⚠ Your company does not have any Company Admins assigned, or those admins have not logged in for more than 30 days. This new role is very important to the proper administration of your account(s).
To assign a user to this role, please download and fill out this document and mail it to [redacted].
If you need assistance, please contact us via email or call [redacted]

Search...

Full Name	User ID	Last Login (CT)	Role Type	Account Admin	Programmer	Pending
				(All)	(All)	(All)
Jane Doe	janedoe4953@gmail.com	10/12/2021 06:53 PM	Customer	✓		

Delete | Add | Copy | Move | Refresh | Save Settings

"Company Admin" roles and capabilities

Setting up a Company Admin is important in order to access all of the features of Self-Service such as:

- Giving a user "Programmer" access, which grants the ability to send OTA and Programmable Parameters updates to a programmable vehicle.
- Creating new location accounts anywhere in the Company structure.
- Administrative ability over all location and watchlist accounts.
- Sharing lists of vehicles with other organizations.
- Inviting Dealer Advocates to manage your organization through sharing

Close

[redacted] **OTA Programming Services**

Company Administrator Authorization

As the owner ("Customer") of vehicle(s) enrolled in [redacted] I designate the individual identified below as Company Administrator with authority to assign permissions related to over-the-air (OTA) programming of engine calibrations and programmable parameters to [redacted] with access to the Company Vehicles ("OTA Programming Services"), including

- Assign and revoke programmer access
- Enable and disable Auto-Deploy engine calibration updates
- Deploy engine calibration updates
- Create and deploy programmable parameter update profiles

Through the OTA Programming Services, I will receive the [redacted] OTA Programming Services offer, including fast, timely updates no matter where the vehicle is located, greater flexibility to modify engine parameters for the right balance of performance, safety and efficiency, and more, as set forth in the [redacted]

I understand that OTA Programming Services are available for the Company Vehicles as long as they are enrolled in [redacted]

I represent that I have authority on behalf of the below Customer to make this designation and agree to the terms of the [redacted] which I have had the opportunity to read and review.

Name, email and telephone number of the person who will serve as Company Administrator of the Company Vehicles:

Name*:_____

Email address*:_____

Telephone number*:_____

[redacted]:_____

Signature on behalf of Customer*:_____ Date*:_____

Name (printed)*:_____

Title* :_____

Customer Name*:_____ DOT Number:_____

Customer Address*:_____

Customer CIS Number:_____

List of Company Vehicles *(enter VIN of one or more Company Vehicles)**:

_____

_____

_____

_____

_____

** Required field*

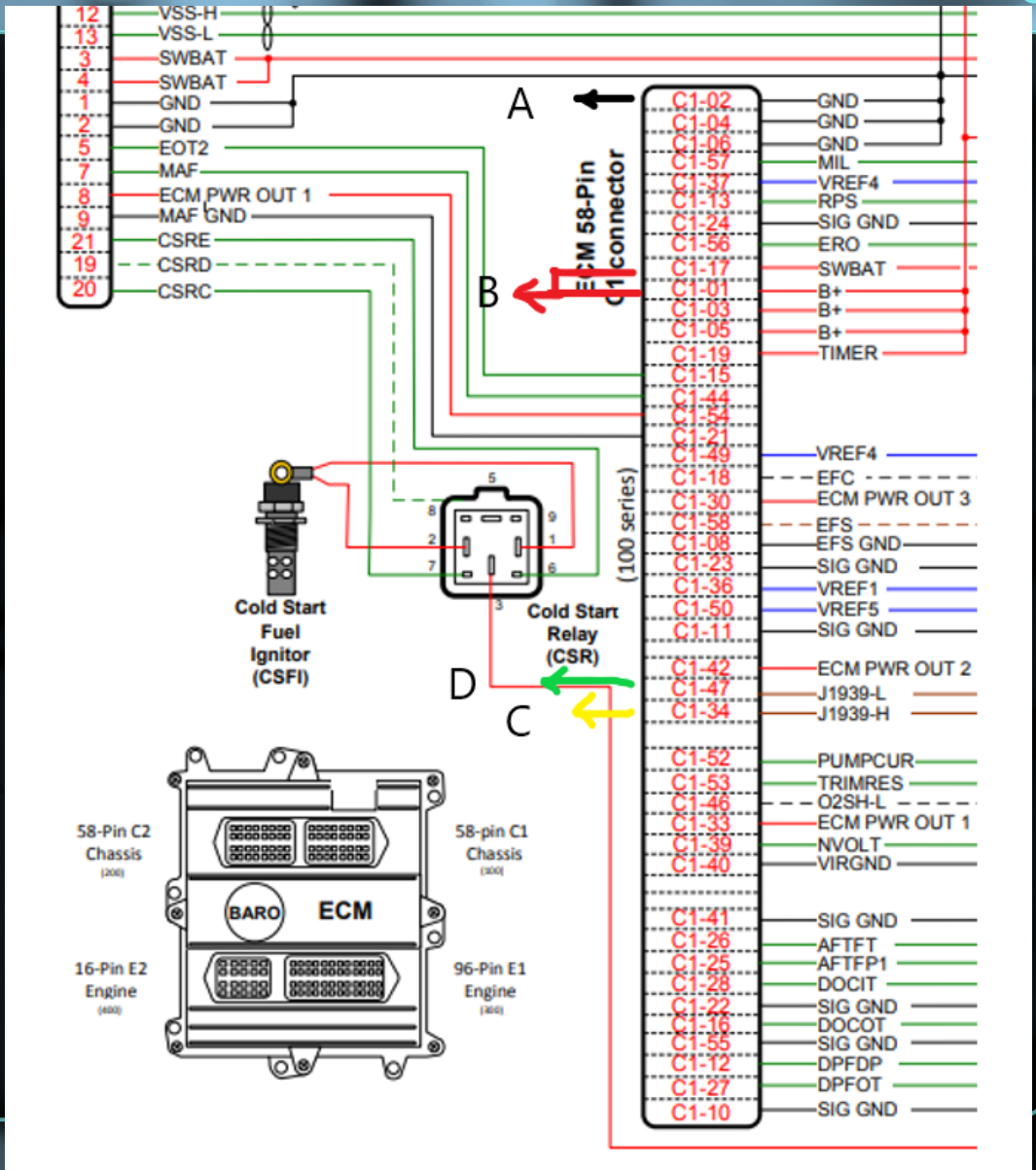
TARGET ECU THREAT MODEL

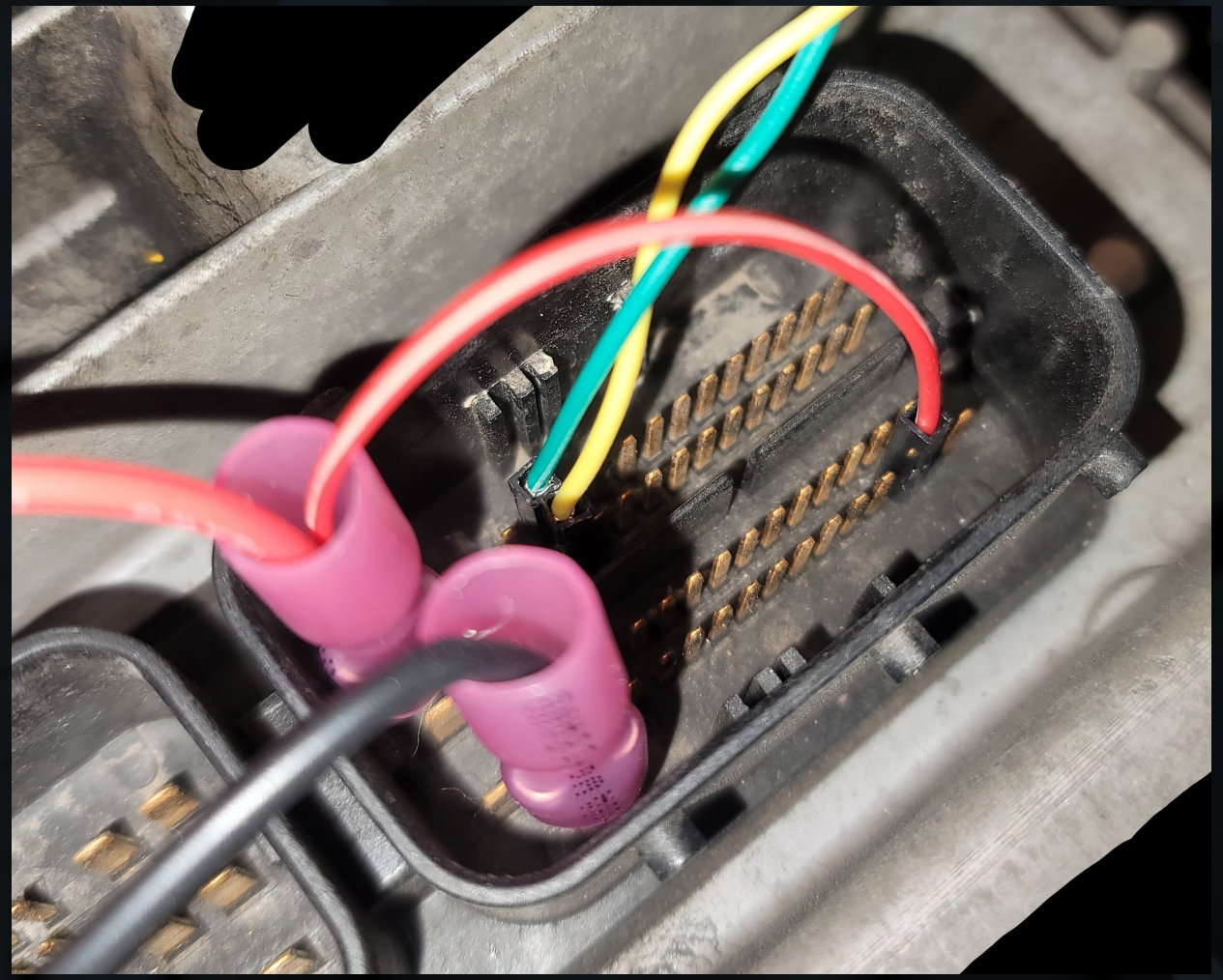
- Remote attack vectors are likely, and possibly on whole fleets of vehicles
- The impact of vulnerabilities found on the ECU could be critical, ranging from information disclosure to vehicle takeover

GETTING CONNECTED

- Isolate ECU on a test bench for initial discovery and testing
  - Power the ECU with a power supply, and connect to data pins (e.g., CAN high/low)
  - Attach to other signal pins as needed for testing various conditions (e.g., wheel speed sensor)
- Place ECU in a truck for testing node-to-node communication, gateways, and features only enabled when vehicle is in motion (Note: dangerous! Use a simulator if possible!)

Pin	Type 2 (Green)
A	Ground
B	Battery
C	J1939 + 500kb
D	J1939 - 500kb
E	J1939 Shield
F	J1708+ / J1939 + 250kb
G	J1708- / J1939 - 250kb
H	OEM Specific
J	OEM Specific





DISCOVERY

- Address and Name
- Status messages
- Proprietary messages
- Diagnostics

- Determine ECU's address

```
> python .\truckdevil.py add_device m2 can0 250000 COM5 run_module ecu_discovery
```

```
Welcome to the ECU Discovery tool.
(truckdevil.ecu_discovery) active_scan
scanning...
scanning complete.
added 2 new ecus.
(truckdevil.ecu_discovery) passive_scan
scanning...
scanning complete.
added 1 new ecus.
(truckdevil.ecu_discovery) view_ecus
address: 0     NAME: 0000004005000000
address: 11    NAME: 0000400b00090000
address: 15    NAME: unknown
(truckdevil.ecu_discovery)
```

- Look for messages in the proprietary range

```
(truckdevil.ecu_discovery) ? find_proprietary

        Provide the address of the ECU to discover the proprietary messages it's sending.
        Performs passive and active scanning techniques.

        usage: find_proprietary <address>

(truckdevil.ecu_discovery) find_proprietary 0
Scanning...
discovered 8 new unique proprietary messages.
Proprietary messages for address 0:
0x18ff3500  06 FF35 00 --> FF [8]  FFFFFF00FFFEFFFF
0x18ff1800  06 FF18 00 --> FF [8]  00FFFFFFFFFFFFFF
0x18ff9b00  06 FF9B 00 --> FF [8]  FFF0F0F0FE00FEF2
0x18efff00  06 EF00 00 --> FF [8]  FFFFFFFF00011980
0x18ff5000  06 FF50 00 --> FF [8]  FFFFFFCCFFFFFFFF
0x18ffb000  06 FFB0 00 --> FF [8]  7F7F7F7F7F7FFFFF
0x18ff9a00  06 FF9A 00 --> FF [8]  0000000000F0FFFF
0x18ff3800  06 FF38 00 --> FF [8]  000000FFFF96FFFF
```

- Look for the presence of UDS for diagnostics

```
(truckdevil.ecu_discovery) ? find_uds

        Provide the address of the ECU to determine if it responds to a UDS session.
        Performs passive and active scanning techniques.

        usage: find_uds <address>

(truckdevil.ecu_discovery) find_uds 0
Scanning...
ECU did not respond to any tester present requests.
```

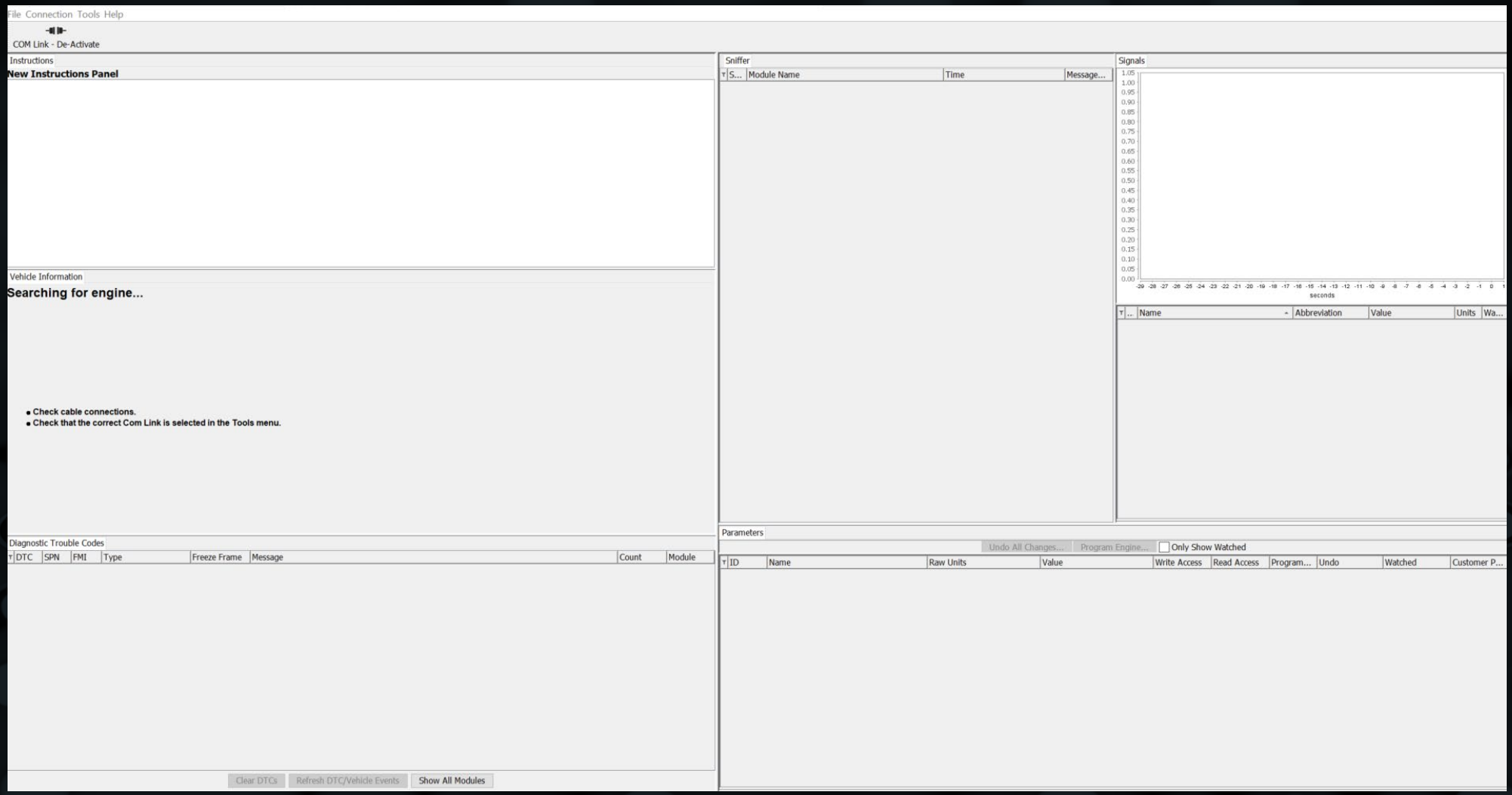
- Example of what the tool would find on an ECU that does offer UDS

```
(truckdevil.ecu_discovery) find_uds 11
Scanning...
Tester present responses:
0x18daf90b  06 DA00 0B --> F9 [8]  027E00FFFFFFFFFF
```

- What is the ECU's reboot message?

```
(truckdevil.ecu_discovery) ? find_boot_msg

        Provide the address of the ECU to discover it's reboot message in order to detect crashes.
        ECU must be reset during this test.

        usage: find_boot_msg <address>

(truckdevil.ecu_discovery) find_boot_msg 0
please shut down the ECU, enter y when done or q to quit: y
waiting for messages to stop transmitting...
please power on the ECU, enter y when done or q to quit: y
reboot message for ECU 0:
18EEFF00    06 EE00 00 --> FF [0008] 0000004005000000
```

- Are there any engineering or diagnostic tools available that interact with the ECU?
  - Often these expose functionality for reading/writing parameters and firmware
- I found several tools sold from the manufacturer that can reprogram blank modules, read parameters, perform diagnostics, data log, etc.
  - beware sketchy free versions; these are usually old or don't have all the features and probably contain malware
  - Just buy the < $500 product key or get one as part of the assessment


File Connection Tools Help
COM Link - De-Activate
Instructions
New Instructions Panel
Vehicle Information
Searching for engine...
Check cable connections.
Check that the correct Com Link is selected in the Tools menu.
Diagnostic Trouble Codes
DTC
SPN
FMI
Type
Freeze Frame
Message
Count
Module
Clear DTCs
Refresh DTC/Vehicle Events
Show All Modules
Sniffer
S...
Module Name
Time
Message...
Signals
seconds
Name
Abbreviation
Value
Units
Parameters
Undo All Changes...
Program Engine...
Only Show Watched
ID
Name
Raw Units
Value
Write Access
Read Access
Program...
Undo
Watched
Customer P...

- Before connecting, record a baseline so you know what messages the ECU sends on a regular interval

```
> python .\truckdevil.py add_device m2 can0 250000 COM5 run_module read_messages
```

```
Welcome to the Read Messages tool.
(truckdevil.read_messages) set log_to_file true
(truckdevil.read_messages) set log_name baseline.txt
(truckdevil.read_messages) set abstract_TPM true
(truckdevil.read_messages) print_messages
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
10FCE300    04 FCE3 00 --> FF [0008] FE0000FFFFFFFFFF
18F0090B    06 F009 0B --> FF [0008] FFFFFFFFFFFFFFFF
18FD9400    06 FD94 00 --> FF [0008] 0000FFFF0000FFFF
10FCFD00    04 FCFD 00 --> FF [0008] FFFFFFFF2964FFFF
18FEDB00    06 FEDB 00 --> FF [0008] FFFF00FEFFFFFFFF
18FEEF00    06 FEEF 00 --> FF [0008] FEFFFFFE8080FFFA
18F0000F    06 F000 0F --> FF [0008] 517D7DFF007DFF7D
18FE6900    06 FE69 00 --> FF [0008] 00FEFFFFFFFF00FE
18FEF200    06 FEF2 00 --> FF [0008] 000000FEFFFFFEFF
```

- Then record all the traffic that occurs upon connecting with the tool

```
(truckdevil.read_messages) set log_name diag_connect.txt
(truckdevil.read_messages) print_messages
```

- While recording, press the “Activate COM Link” button in the diagnostic tool
- Perform a passive scan in ECU discovery to determine the diagnostic tool’s address

```
Welcome to the ECU Discovery tool.
(truckdevil.ecu_discovery) passive_scan
scanning...
scanning complete.
added 4 new ecus.
(truckdevil.ecu_discovery) view_ecus
address: 0     NAME: unknown
address: 11    NAME: unknown
address: 15    NAME: unknown
address: 249   NAME: unknown
```

File Connection Tools Help

COM Link- Connection

Instructions

New Instructions Panel

Vehicle Information

Engine Type: [redacted]
Software Identification: OECAATMA
Vehicle Identification Number: 3HSDJSJR6CN582020
Engine Serial Number: 125HM2Y4114626
EDC Customer Unit Number: 0
Transmission Type: Manual
Rated Power: 430.0 hp
Total Miles: 568,425.2 miles
Total Fuel Used: 93,440.0 gal
Engine On Time: 18,780.71 hr

Diagnostic Trouble Codes

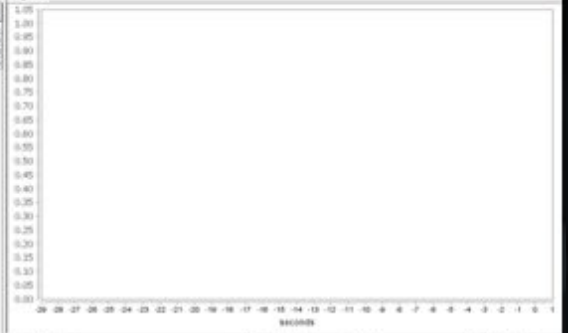
↑	DTC	SPN	FMI	Type	Freeze Frame	Message	Count	Module
	R/A	1479	4	Pending	Open	AFT Fuel Doser Valve High Side Short circuit	1	Engine Co...
	R/A	157	3	Pending	Open	FRP signal Out of Range HIGH	1	Engine Co...
	R/A	633	5	Pending	Open	FPCV open load/circuit	1	Engine Co...
	R/A	110	3	Pending	Open	ECT1 signal Out of Range HIGH	1	Engine Co...
	R/A	4076	3	Pending	Open	ECT2 signal Out of Range HIGH	1	Engine Co...
	R/A	3480	3	Pending	Open	AFTTP1 signal Out of Range HIGH	1	Engine Co...
	R/A	4077	3	Pending	Open	AFTTP2 signal Out of Range HIGH	1	Engine Co...
	R/A	3251	4	Pending	Open	DPFDP signal Out of Range LOW	1	Engine Co...
	R/A	412	3	Pending	Open	EGROT signal Out of Range HIGH	1	Engine Co...
	R/A	105	3	Pending	Open	IMT signal Out of Range HIGH	1	Engine Co...

Clear DTCs | Refresh DTC/Vehicle Events | Show All Modules

Sniffer

↑	S...	Module Name	Time	Message...
	11	ABS Control Module	01:38:56:740	101.791
	0	Engine Control Module	01:38:57:242	343.205
	15	Retarder - Engine	01:38:32:444	29.493
	249	OBD Board Diagnostic-Service Tool	01:38:30:523	280.365

Signals



↑	..	Name	Abbreviation	Value	Units	Wa...
	3304	A/C System Refrigerant Monitoring Ena...	A/C Mon Enabled	Disabled		
	3305	A/C System Refrigerant Monitoring Status	A/C System Mon S...	Complete		
	1223	A/C system refrigerant monitoring Status	A/C	test complete, no f...		
	1222	A/C system refrigerant monitoring Sup...	A/C	test not supported ...		
	5...	AC Demand	ACD	Off		
	991	AESC Accelerate Switch	AESCASw	Not Available		
	993	AESC Coast/Decelerate Switch	AESCCSw	Not Available		
	995	AESC Enable Switch	AESCESw	Off		
	974	AESC Remote Accelerator Position	RAP	Not Available%		
	976	AESC Remote Signal Status #1	AESC RSS #1	Off		
	978	AESC Remote Signal Status #2	AESC RSS #2	Off		
	974	AESC Remote Switch State	R-AESC-St	Off/Disabled		
	992	AESC Resume Switch	AESCRSw	Not Available		
	994	AESC Set Switch	AESCSSw	Not Available		
	3236	AFT Exhaust Gas Mass Flow	EGMF	0.43kg/hr		
	3479	AFT Fuel Doser CTL	AFTFDC	0.00%		
	3480	AFT Fuel Pressure 1	AFTFP1	Error psi		
	4077	AFT Fuel Pressure 2	AFTFP2	Error psi		
	3481	AFT Fuel Rate	AFTFR	0.00gal/hr		
	3482	AFT Fuel Shutoff CTL	AFTFSC	Disable		
	3484	AFT Fuel Temperature	AFTFT	Error F		
	3734	AFT Number of Active Regens (Trip)	AFTITAR	3,320		
	3736	AFT Regen Inhibit - AESC Active	AFTRIAESCA	Not inhibited		
	3737	AFT Regen Inhibit - Accelerator Off Idle	AFTRIAOI	Not inhibited		

Parameters

Undo All Changes... | Program Engine... | Only Show Watched

↑	ID	Name	Raw Units	Value	Write Access	Read Access	Program...	Undo	Watched	Customer P...
	91232	Crankshaft Position Learning Reset Request		No	Fleets	Available				
	03691	Smart Torque Enable		Enable	Fleets	Available				
	03670	Trip AftertreatmentFuel Used	L/10	[illegible]	Unavailable	Available				
	99332	Remote AESC Variable Speed Switch Input ...		Use hardwired input	Fleets	Available				
	62470	Total Distance with Fan On	m	[illegible]	Engineering	Available				
	57351	Total AESC Mobile Fuel Used	L/10	[illegible]	Engineering	Available				
	65001	Vehicle Identification Number		[illegible]	Program Support	Available				
	59411	Trip Vehicle Overspeed 1 Time	s	[illegible]	Engineering	Available				
	59111	Transfer Case Input Mode Select		Driveline Engaged	Fleets	Available				
	62490	Trip Fan Time in Slip Zone	s	[illegible]	Unavailable	Available				
	57341	Total AESC Stationary Fuel Used	L/10	[illegible]	Engineering	Available				
	59411	Trip Vehicle Overspeed 2 Time	s	[illegible]	Engineering	Available				
	99352	Remote Accelerator Pedal Input Selection		Use CAN input 1	Fleets	Available				
	62490	Trip Fan Time in On/Off Zone	s	[illegible]	Unavailable	Available				
	57371	Total AESC Mobile Time	s	[illegible]	Engineering	Available				
	95141	Enable the fan on with engine braking fe...		Disable	Dealers	Available				
	59421	Trip Loaded AESC Mobile Time	s	[illegible]	Engineering	Available				
	99362	Injection Quantity Adjustment - Injectio...		A7A7A4AAAA	Fleets	Available				
	62500	Trip Fan Time at Maximum Fan Speed	s	[illegible]	Unavailable	Available				
	57381	Total AESC Stationary Time	s	[illegible]	Engineering	Available				
	59431	Trip Drive Time	s	[illegible]	Engineering	Available				
	61000	Trip Engine Oil Temp Warning Time	s	[illegible]	Engineering	Available				
	99372	Injection Quantity Adjustment - Injector...		A1A4A0AAIA	Fleets	Available				
	62510	Trip Distance with Fan On	m	[illegible]	Unavailable	Available				
	57391	Total Drive Fuel Used	L/10	[illegible]	Engineering	Available				
	65061	Vehicle Model		[illegible]	Program Support	Available				

- Review the recorded log file for communications from 0xF9

```
F9 -->                                                    Cc W .*   12/133
2744  18EFFF00   06 EF00 00 --> FF [0008] FFFFFFFF00011980
2745  10F01A00   04 F01A 00 --> FF [0008] 0000FFFF0000B004
2746  18F0090B   06 F009 0B --> FF [0008] FFFFFFFFFFFFFFFF
2747  0CF00400   03 F004 00 --> FF [0008] F87D7D000000F07D
2748  08FE6E0B   02 FE6E 0B --> FF [0008] FFFEFFFEFFFEFFFE
2749  18F0090B   06 F009 0B --> FF [0008] FFFFFFFFFFFFFFFF
2750  18EC00F9   06 EC00 F9 --> 00 [0008] 130A0002FFDAFE00
2751  0CF00400   03 F004 00 --> FF [0008] F87D7D000000F07D
2752  18F0090B   06 F009 0B --> FF [0008] FFFFFFFFFFFFFFFF
2753  18FFDDF9   06 FFDD F9 --> FF [0008] 72550000FFFFFFFF
2754  10FE6F00   04 FE6F 00 --> FF [0008] FFFFFFFFFFFFFFFF
```

- Record the same connection again, but filter out only comms between the target ECU and the diagnostic tool

```
(truckdevil.read_messages) set filter_src_addr 0,249
(truckdevil.read_messages) set log_name diag_connect_filtered.txt
(truckdevil.read_messages) print_messages
```

```
18EC00F9    06 EC00 F9 --> 00 [0008] 130A0002FFDAFE00
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
18FFDDF9    06 FFDD F9 --> FF [0008] 6D30494E5414FFFF
10FE6F00    04 FE6F 00 --> FF [0008] FFFFFFFFFFFFFFFF
18FF3500    06 FF35 00 --> FF [0008] FFFFFF00FFFEFFFF
18FF1800    06 FF18 00 --> FF [0008] 00FFFFFFFFFFFFFF
18FEF100    06 FEF1 00 --> FF [0008] F300FE10FF0000C5
18F00100    06 F001 00 --> FF [0008] FFFFFFCFFFFFFFFF
10FCE300    04 FCE3 00 --> FF [0008] FE0000FFFFFFFFFF
18FD9400    06 FD94 00 --> FF [0008] 0000FFFF0000FFFF
18FFDD00    06 FFDD 00 --> FF [0008] 6D301400FFFFFFFF
18FFDDF9    06 FFDD F9 --> FF [0008] 72580101FFFFFFFF
18ECF900    06 EC00 00 --> F9 [0008] 100F000303DDFF00
18EC00F9    06 EC00 F9 --> 00 [0008] 110301FFFFDDFF00
18EBF900    06 EB00 00 --> F9 [0008] 0172580101000288
18EBF900    06 EB00 00 --> F9 [0008] 0230303030303030
18EBF900    06 EB00 00 --> F9 [0008] 0330FFFFFFFFFFFF
18FFDD00    06 FFDD 00 --> FF [0015] 7258010100028830303030303030 30
18EC00F9    06 EC00 F9 --> 00 [0008] 130F0003FFDDFF00
18FFDDF9    06 FFDD F9 --> FF [0008] 6D31494E5414FFFF
18FFDD00    06 FFDD 00 --> FF [0008] 6D311400FFFFFFFF
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
```

- The tool sends a message with PGN 0xFFDD from the proprietary range

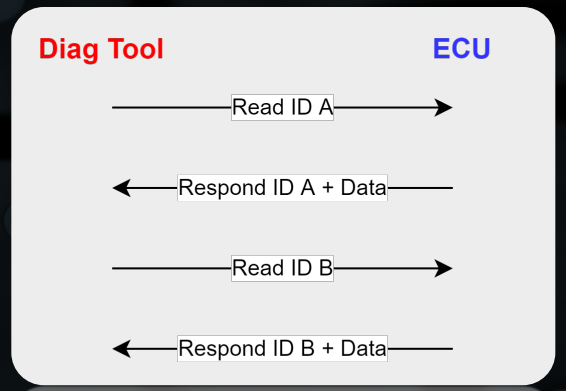
- **Note 1**: this PGN was not found in the baseline, so it's very likely associated with the diagnostic session:

FFDD Cc W .* 0 results

- **Note 2**: the engine stops sending all other messages – the initial message appeared to put the ECU into a diagnostic session and the last message stopped the session

```
18FFDDF9    06 FFDD F9 --> FF [0008] 72520505FFFFFFFF
18ECF900    06 EC00 00 --> F9 [0008] 1009000202DDFF00
18EC00F9    06 EC00 F9 --> 00 [0008] 110201FFFFDDFF00
18EBF900    06 EB00 00 --> F9 [0008] 0172520505000288
18EBF900    06 EB00 00 --> F9 [0008] 023EA1FFFFFFFFFF
18FFDD00    06 FFDD 00 --> FF [0009] 7252050500028 83EA1
18EC00F9    06 EC00 F9 --> 00 [0008] 13090002FFDDFF00
18FFDDF9    06 FFDD F9 --> FF [0008] 72530202FFFFFFFF
18ECF900    06 EC00 00 --> F9 [0008] 100B000202DDFF00
18EC00F9    06 EC00 F9 --> 00 [0008] 110201FFFFDDFF00
18EBF900    06 EB00 00 --> F9 [0008] 0172530202000288
18EBF900    06 EB00 00 --> F9 [0008] 0236869CFFFFFFFF
18FFDD00    06 FFDD 00 --> FF [0011] 72530202000288 36869CFF
18EC00F9    06 EC00 F9 --> 00 [0008] 130B0002FFDDFF00
18FFDDF9    06 FFDD F9 --> FF [0008] 72530000FFFFFFFF
18ECF900    06 EC00 00 --> F9 [0008] 100B000202DDFF00
18EC00F9    06 EC00 F9 --> 00 [0008] 110201FFFFDDFF00
18EBF900    06 EB00 00 --> F9 [0008] 0172530000000288
18EBF900    06 EB00 00 --> F9 [0008] 020035F8C2FFFFFF
18FFDD00    06 FFDD 00 --> FF [0011] 72530000000288 0035F8C2
18EC00F9    06 EC00 F9 --> 00 [0008] 130B0002FFDDFF00
18FFDDF9    06 FFDD F9 --> FF [0008] 72530101FFFFFFFF
```

- Then there's a series of messages from F9 and 00 with the same 0xFFDD PGN
- This message is used to read various data from the engine



- What do we know so far?

Uses proprietary PGN 0xFFDD for diagnostics

Start a diagnostic session:

6D	30 49 4E 54 14	FF FF

Stop a diagnostic session:

6D	31 49 4E 54 14	FF FF

Read data by identifier:

72	Data Identifier (3 bytes)	FF FF FF FF

DYNAMIC TESTING

- Create test cases that challenge intended logic

Does it actually work the way you think it does?

- Let’s create a module in the testing framework to attempt to start our own diagnostic session with the ECU
- It’s called “custom.py”

```
> python .\truckdevil.py add_device m2 can0 250000 COM5 run_module custom
```

```
(truckdevil.custom) read_by_identifier 0x550000
Request:
18FFDDF9    06 FFDD F9 --> FF [0008] 72550000FFFFFFFF
Response:
18FFDD00    06 FFDD 00 --> FF [0024] 72550000000A14334853444A534A5236434E353832303230
Value of ID 550000: 334853444A534A5236434E353832303230
```

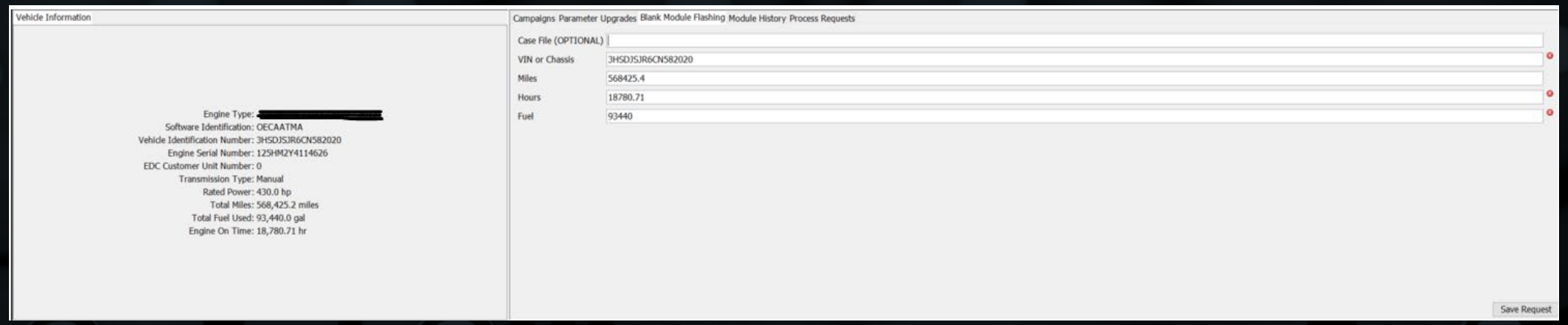
```python
class ECUInteraction:
    def __init__(self, device):
        self.devil = J1939Interface(device)

        self.ecu_addr = 0
        self.diag_addr = 0xF9
        self.can_id_from_ecu = 0x18FFDD00
        self.can_id_from_diag = 0x18FFDDF9

        self.diag_sess_id = 0x6D
        self.read_id = 0x72
        self.write_id = 0x77

    def start_diag_sess(self):
        data = "6D30494E5414FFFF"
        msg = J1939Message(self.can_id_from_diag, data)
        params = {"data_contains": "6D301400"}
        self.devil.send_message(msg)
        self.devil.read_messages_until(**params)

    def stop_diag_sess(self):
        data = "6D31494E5414FFFF"
        msg = J1939Message(self.can_id_from_diag, data)
        params = {"data_contains": "6D311400"}
        self.devil.send_message(msg)
        self.devil.read_messages_until(**params)

    def read_by_identifier(self, identifier: int):
        self.start_diag_sess()
        data = "{:02x}{:06x}FFFFFFFF".format(self.read_id, identifier)
        rqst_msg = J1939Message(self.can_id_from_diag, data)
        params = {"data_contains": "{:02x}{:06x}".format(self.read_id, identifier),
                  "can_id": self.can_id_from_ecu}
        self.devil.send_message(rqst_msg)
        rsp_msg, msgs = self.devil.read_messages_until(rts_response_addr=self.diag_addr, **params)
        return rqst_msg, rsp_msg
```

- Create various functions for the actions you can take

```python
class CustomCommands(cmd.Cmd):
    intro = "Welcome to the Custom tool."
    prompt = "(truckdevil.custom) "

    def __init__(self, argv, device):
        super().__init__()
        self.inter = ECUInteraction(device)

    def do_read_by_identifier(self, arg):
        """
        example: read_by_identifier 0x4A0404
        """
        argv = arg.split()
        identifier = int(argv[0], 16)

        rqst, rsp = self.inter.read_by_identifier(identifier)
        print("Request: \n{}".format(rqst))
        print("Response: \n{}".format(rsp))


    def do_start_diag(self, arg):
        self.inter.start_diag_sess()
        print("diagnostic session started.")

    def do_stop_diag(self, arg):
        self.inter.stop_diag_sess()
        print("diagnostic session stopped.")


def main_mod(argv, device=None):
    cli = CustomCommands(argv, device)
    cli.cmdloop()
```

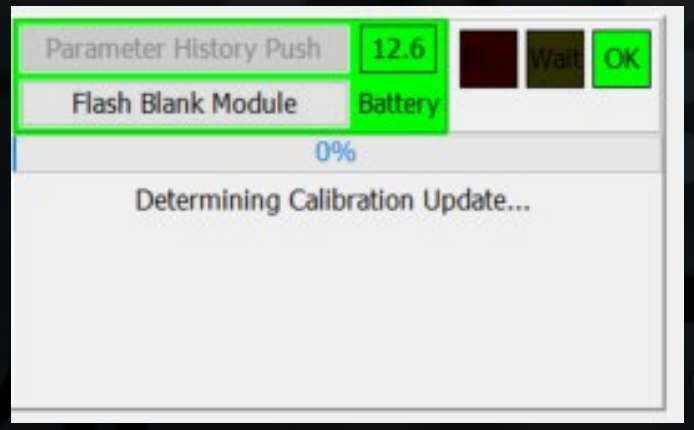
- Module is called in a command loop to easily accept user input and add more functionality and test cases

- How do we write data?
- Other views of the tool include “parameter upgrades” and “blank module flashing”

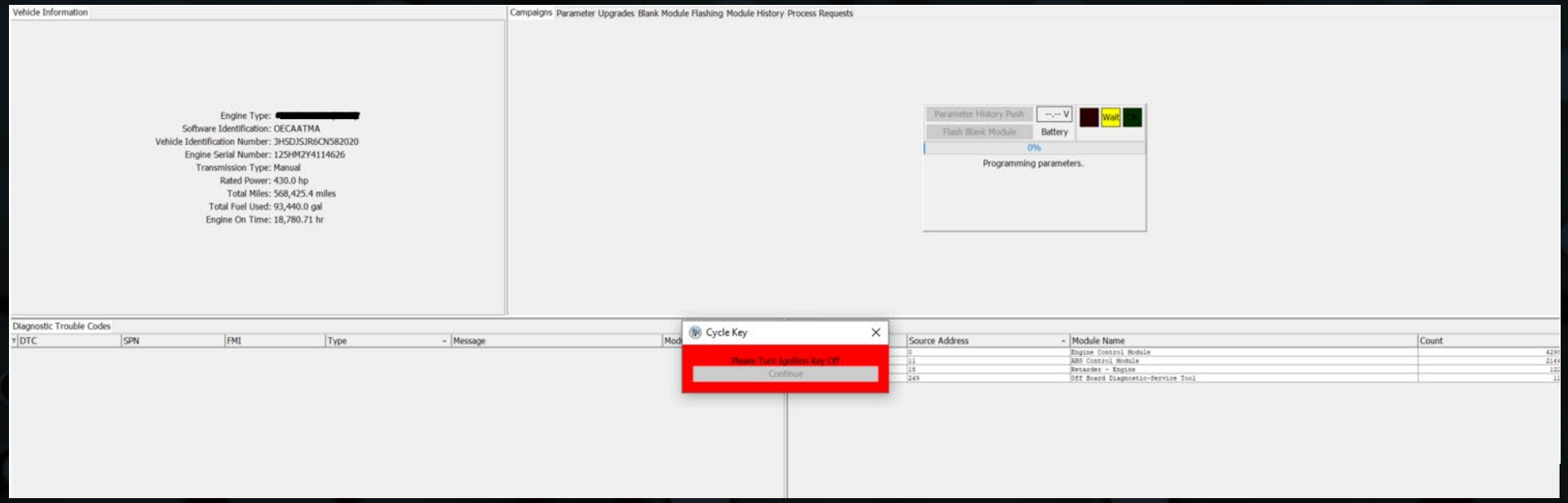



```
(truckdevil.read_messages) set filter_src_addr 0,249
(truckdevil.read_messages) set log_name write_miles_param.txt
(truckdevil.read_messages) set log_to_file true
(truckdevil.read_messages) set abstract_TPM false
(truckdevil.read_messages) print_messages
```

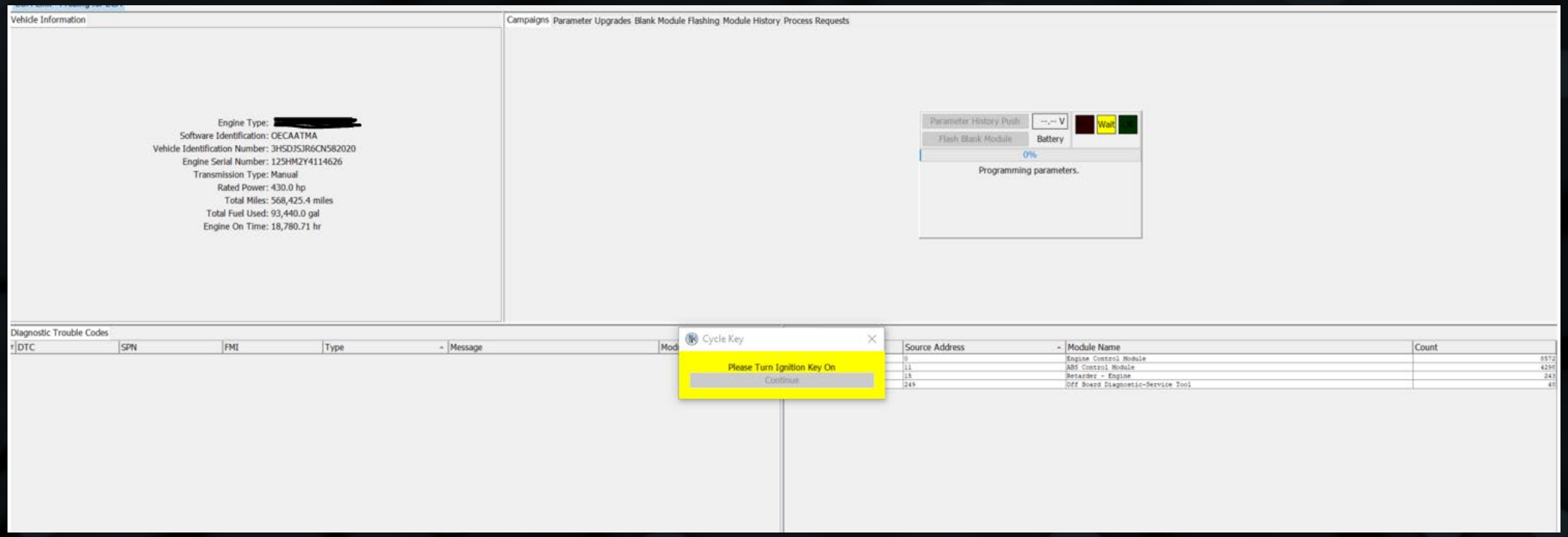
- Step 1 – select “flash”



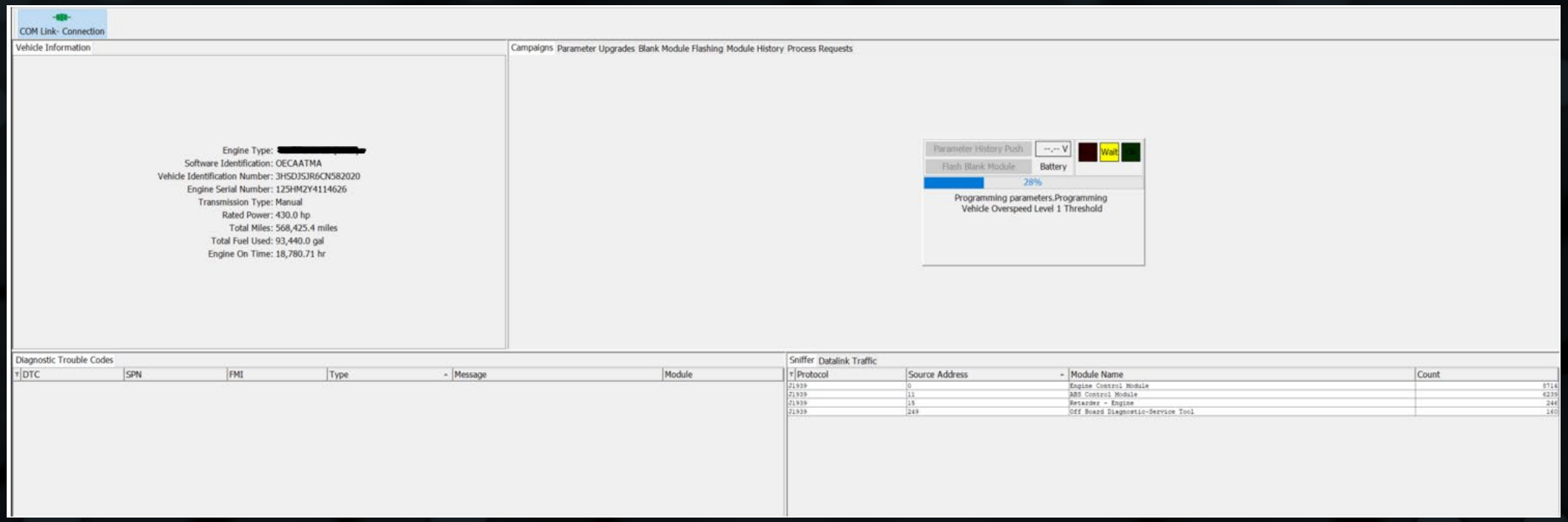
- Step 2 – turn off ignition



- Step 3 – turn ignition back on



- Step 4 – write all parameters



- While reviewing the log file, look for the ECU's reboot message
- We know that writing occurred soon after this message

```
18EA00F9    06 EA00 F9 --> 00 [0003] DAFE00
18EEFF00    06 EE00 00 --> FF [0008] 0000004005000000
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
```

```
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
18FFDDF9    06 FFDD F9 --> FF [0008] 6D30494E5414FFFF
18FFDD00    06 FFDD 00 --> FF [0008] 6D301400FFFFFFFF
18ECFF00    06 EC00 00 --> FF [0008] 20120003FFECFE00
18FFDDF9    06 FFDD F9 --> FF [0008] 72420101FFFFFFFF
18EBFF00    06 EB00 00 --> FF [0008] 01334853444A534A
18EBFF00    06 EB00 00 --> FF [0008] 025236434E353832
18EBFF00    06 EB00 00 --> FF [0008] 033032302AFFFFFF
18FEEC00    06 FEEC 00 --> FF [0018] 334853444A534A5236434E3538323032302A
18FFDDF9    06 FFDD F9 --> FF [0008] 72420101FFFFFFFF
18ECF900    06 EC00 00 --> F9 [0008] 1013000303DDFF00
18EC00F9    06 EC00 F9 --> 00 [0008] 110301FFFFDDFF00
18EBF900    06 EB00 00 --> F9 [0008] 0172420101000288
18EBF900    06 EB00 00 --> F9 [0008] 0208101344027206
18EBF900    06 EB00 00 --> F9 [0008] 030211024123FFFF
18FFDD00    06 FFDD 00 --> FF [0019] 7242010100028808101344027206021102412 3
18EC00F9    06 EC00 F9 --> 00 [0008] 13130003FFDDFF00
```

- Next, a diagnostic session is started
- Then identifier 0x420101 is requested, which was not in the previous recording

```
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0000FF426C756549515147414F464A03
18FFDD00    06 FFDD 00 --> FF [0008] 774A0000000029403
18FFDDF9    06 FFDD F9 --> FF [0021] 774A0101FF426C7565515051534F41495200015F90
18FFDD00    06 FFDD 00 --> FF [0011] 774A0101000295000015F90
18FFDDF9    06 FFDD F9 --> FF [0019] 774A0202FF426C75654F4D534643434B4F00D2
18FFDD00    06 FFDD 00 --> FF [0009] 774A0202000296000D2
18FFDDF9    06 FFDD F9 --> FF [0019] 774A0303FF426C7565534A49434D444C410000
18FFDD00    06 FFDD 00 --> FF [0009] 774A0303000297 0000
18FFDDF9    06 FFDD F9 --> FF [0021] 774A0404FF426C75654D4C4F4F45414C4600007530
18FFDD00    06 FFDD 00 --> FF [0011] 774A0404000298000075 30
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0505FF426C7565435251414C4D4D4900
18FFDD00    06 FFDD 00 --> FF [0008] 774A0505000299 00
18FFDDF9    06 FFDD F9 --> FF [0019] 774A0606FF426C75654A4A505344474B530096
18FFDD00    06 FFDD 00 --> FF [0009] 774A0606000 29A0096
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0707FF426C75654351424C4741424700
18FFDD00    06 FFDD 00 --> FF [0008] 774A0707000 29B00
```

- Next, the diagnostic tool writes all the parameters to the ECU, incrementing some count after each write

- Engine Serial Number flashed (ID 0x580000)

```
18FFDDF9    06 FFDD F9 --> FF [0034] 77580000FF426C7565494D52444F4947453132354 84D3259343131343632360000 00
18FFDD00    06 FFDD 00 --> FF [0024] 7758000000035E31323548 4D3259343131343632360000 00

426C7565494D52444F4947453132354 84D3259343131343632360000 00 =
BlueIMRDOIGE125HM2Y4114626
```

Vehicle Identification Number: 3HSDJSJR6CN582020
Engine Serial Number: 125HM2Y4114626

- Attempting to write the VIN by replaying
- Write VIN from recording:

```
18FFDDF9    06 FFDD F9 --> FF [0034] 77550000FF426C75654950504F434C524E334853444A534A5236434E353832303230
18FFDD00    06 FFDD 00 --> FF [0024] 77550000 00035A334853444A534A5236434E353832303230
```

- Write VIN with script:

```
(truckdevil.custom) write_by_identifier 0x550000 334853444A534A5236434E353832303230 FF 4950504F434C524E
Request:
18FFDDF9    06 FFDD F9 --> FF [0034] 77550000FF426C75654950504F434C524E334853444A534A5236434E353832303230
Response:
18FFDD00    06 FFDD 00 --> FF [0008] 77550000020A14FF
```

00 = success!

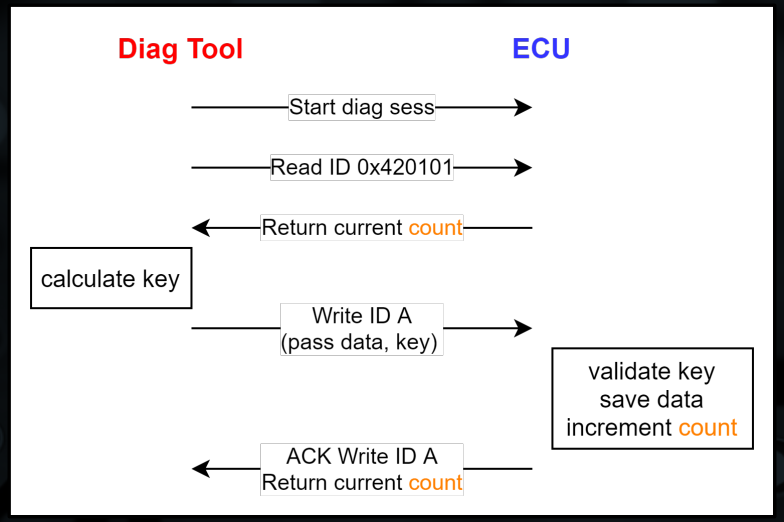
02 = fail ☹

```
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0000FF426C756549515147414F464A03
18FFDD00    06 FFDD 00 --> FF [0008] 774A000000029403
18FFDDF9    06 FFDD F9 --> FF [0021] 774A0101FF426C756551505153 4F41495200015F90
18FFDD00    06 FFDD 00 --> FF [0011] 774A010100029500015F90
18FFDDF9    06 FFDD F9 --> FF [0019] 774A0202FF426C75654F4D534643434B4F00D2
18FFDD00    06 FFDD 00 --> FF [0009] 774A02020002960 0D2
18FFDDF9    06 FFDD F9 --> FF [0019] 774A0303FF426C7565534A49434D444C410000
18FFDD00    06 FFDD 00 --> FF [0009] 774A03030002970000
18FFDDF9    06 FFDD F9 --> FF [0021] 774A0404FF426C75654D4C4F4F45414C4600007530
18FFDD00    06 FFDD 00 --> FF [0011] 774A040400029800007530
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0505FF426C756543525141 4C4D4D4900
18FFDD00    06 FFDD 00 --> FF [0008] 774A050500029900
18FFDDF9    06 FFDD F9 --> FF [0019] 774A0606FF426C75654A4A505344474B530096
18FFDD00    06 FFDD 00 --> FF [0009] 774A060600029A0096
```

- Why didn’t the write work?

It may be due to these 8 bytes of data that differ between messages

- Simple replay does not work because an authentication mechanism has been implemented to write values



- Can you trick the diagnostic tool into calculating the next key for you?

Current count = 0x06D4

Spoof the ECU and respond to diagnostic tool with the count of our choice to get key

```
(truckdevil.custom) get_key
sent: 18EEFF00    06 EE00 00 --> FF [0008] 0000004005000000
received: 18FFDDF9    06 FFDD F9 --> FF [0008] 6D30494E5414FFFF
sent: 18FFDD00    06 FFDD 00 --> FF [0008] 6D301400FFFFFFFF
received: 18FFDDF9    06 FFDD F9 --> FF [0008] 72420101FFFFFFFF
sent: 18FFDD00    06 FFDD 00 --> FF [0019] 7242010100006D4081013440272060211024123
received: 18FFDDF9    06 FFDD F9 --> FF [0034] 77550000FF426C7565524F5243444D44413334853444A534A5236434E353832303230
```

- Replaying this Write VIN request to the ECU with the calculated key worked!

Problem: when I attempted to change one character in the VIN, it failed…

This means the calculate_key function probably also takes the data as input

- How is the key calculated?

-8 characters

-all upper-case letters (A-T only)

-repeats sometimes, but never for the same identifier

-appears random (no character appears significantly more often)

calculate_key(curr_count, data, ?) -> key

- The tool also allowed for individual parameters to be written to

Sniffer Datalink Traffic Parameters

Undo All Changes... Program Engine... Only Show Watched

ID	Name	Raw Units	Value	Write Access	Read Access	Progra...	Undo	Watched	Customer ...
95232	Crankshaft Position Learning Reset Request		No	Fleets	Available				
89091	Smart Torque Enable		Enable	Fleets	Available				✓
83970	Trip Aftertreatment Fuel Used	L/10	999.0	Unavailable	Available				
99332	Remote AESC Variable Speed Switch Input Selection		Use hardwired input	Fleets	Available				✓
62470	Total Distance with Fan On	m	88,588.42	Engineering	Available				
57351	Total AESC Mobile Fuel Used	L/10	0.0	Engineering	Available				
85001	Vehicle Identification Number		3HSDJSJR6CN582020	Program Support	Available				
59401	Trip Vehicle Overspeed 1 Time	s	88.31	Engineering	Available				
89101	Transfer Case Input Mode Select		Driveline Engaged	Fleets	Available				✓
62480	Trip Fan Time in Slip Zone	s	13,966.01	Unavailable	Available				
57361	Total AESC Stationary Fuel Used	L/10	453.2	Engineering	Available				
59411	Trip Vehicle Overspeed 2 Time	s	77.30	Engineering	Available				
99352	Remote Accelerator Pedal Input Selection		Use CAN input 1	Fleets	Available				✓
62490	Trip Fan Time in On/Off Zone	s	17.34	Unavailable	Available				
57371	Total AESC Mobile Time	s	0.00	Engineering	Available				
90141	Enable the fan on with engine braking feature		Disable	Dealers	Available				✓
59421	Trip Loaded AESC Mobile Time	s	0.00	Engineering	Available				
99362	Injection Quantity Adjustment - Injector 1 (Physi...		AIACA4AAAA	Fleets	Available				
62500	Trip Fan Time at Maximum Fan Speed	s	3,127.86	Unavailable	Available				
57381	Total AESC Stationary Time	s	289.98	Engineering	Available				
59431	Trip Drive Time	s	6,915.10	Engineering	Available				
41000	Trip Engine Oil Temp Warning Time	s	00:00:00	Engineering	Available				
99372	Injection Quantity Adjustment - Injector 2 (Physi...		AIAABDAAIA	Fleets	Available				
62510	Trip Distance with Fan On	m	88,588.42	Unavailable	Available				
57391	Total Drive Fuel Used	L/10	63,372.9	Engineering	Available				
85041	Vehicle Model		LF68700	Program Support	Available				
59441	Trip Engine Brake Activations		91,233	Unavailable	Available				
41010	Total Engine Oil Temp Warning Time	s	00:00:00	Engineering	Available				
89141	Remote Accelerator Enable Switch		Disable	Fleets	Available				✓
99382	Injection Quantity Adjustment - Injector 3 (Physi...		AAAAAZAAIA	Fleets	Available				
62520	Trip Fan on Time in AESC	s	280.64	Unavailable	Available				
57401	Total Drive Time	s	6,915.10	Engineering	Available				
83003	Total Fuel Used	L * 0.1	93,440.0	Program Support	Available				
90171	Fan Output Pin Configuration		Fan output pin 118 (MIL ou...	Fleets	Available				
59451	Trip Engine Brake Percent Time	%/81.967	18.18	Unavailable	Available				
41020	Trip Engine Oil Temp Critical Time	s	00:00:00	Engineering	Available				
89151	Minimum Gear Number to Enable Extra Torque		16	Program Support	Available				
99392	Injection Quantity Adjustment - Injector 4 (Physi...		DIAGBMABIA	Fleets	Available				
62530	Trip Fan On Time due to fault	s	0.00	Unavailable	Available				
57411	Total Full Load Operation Time	s	2.35	Engineering	Available				
83012	Engine On Time	s	18,780.71	Program Support	Available				
59461	Trip Number of Maximum Accelerations with Vehicle...		491,424	Engineering	Available				
41030	Total Engine Oil Temp Critical Time	s	00:00:00	Engineering	Available				
89160	Firm Brake Deceleration Rate Threshold	kph/s * 0.00015	4.97	Fleets	Available				

(Disable (0), Enable (1))

```
18FFDDF9    06 FFDD F9 --> FF [0018] 77590E0E00426C7565303030303030303000
18FFDD00    06 FFDD 00 --> FF [0008] 77590E0E0006D700
```

```
(truckdevil.custom) read_by_identifier 0x590E0E
Request:
18FFDDF9    06 FFDD F9 --> FF [0008] 72590E0EFFFFFFFF
Response:
18FFDD00    06 FFDD 00 --> FF [0008] 72590E0E0006D900
Value of ID 590E0E: 00
(truckdevil.custom) write_by_identifier 0x590E0E 01
Request:
18FFDDF9    06 FFDD F9 --> FF [0018] 77590E0E00426C7565303030303030303001
Response:
18FFDD00    06 FFDD 00 --> FF [0008] 77590E0E0006DA01
New value of ID 590E0E: 01
(truckdevil.custom) read_by_identifier 0x590E0E
Request:
18FFDDF9    06 FFDD F9 --> FF [0008] 72590E0EFFFFFFFF
Response:
18FFDD00    06 FFDD 00 --> FF [0008] 72590E0E0006DA01
Value of ID 590E0E: 01
(truckdevil.custom)
```

- Modify “Remote Accelerator Enable Switch” from Disabled to Enabled

- It used a static key “3030303030303030”
- 00 in 5th byte place instead of FF; this may be “access type” or similar

- Some parameters are “customer programmable” and use a static key
  - Max vehicle speed, setting brake/clutch/parking switches to be CAN-controlled or hardwired, low and high idle engine speed, tire revolutions per mile, and many more…
- Others are protected by this key calculation
  - We need to disassemble and debug the diagnostic tool while key is calculated

- What do we know so far?

Uses proprietary PGN 0xFFDD for diagnostics

Start a diagnostic session:

6D	30 49 4E 54 14	FF FF

Stop a diagnostic session:

6D	31 49 4E 54 14	FF FF

Read data by identifier:

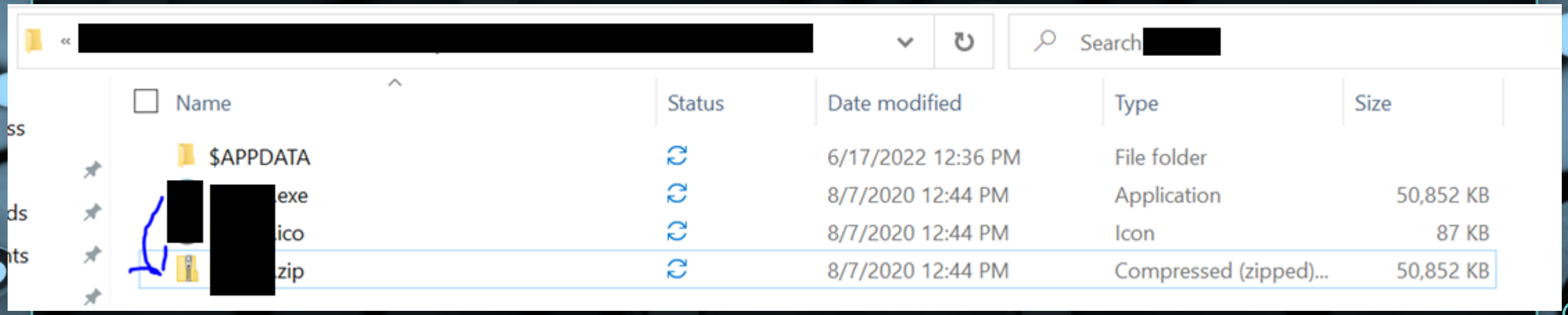
72	Data Identifier (3 bytes)	FF FF FF FF

Write data by identifier:

77	Data Identifier (3 bytes)	Access type (1 byte)	42 6C 75 65	Key (8 bytes)	Data to write (1...n bytes)

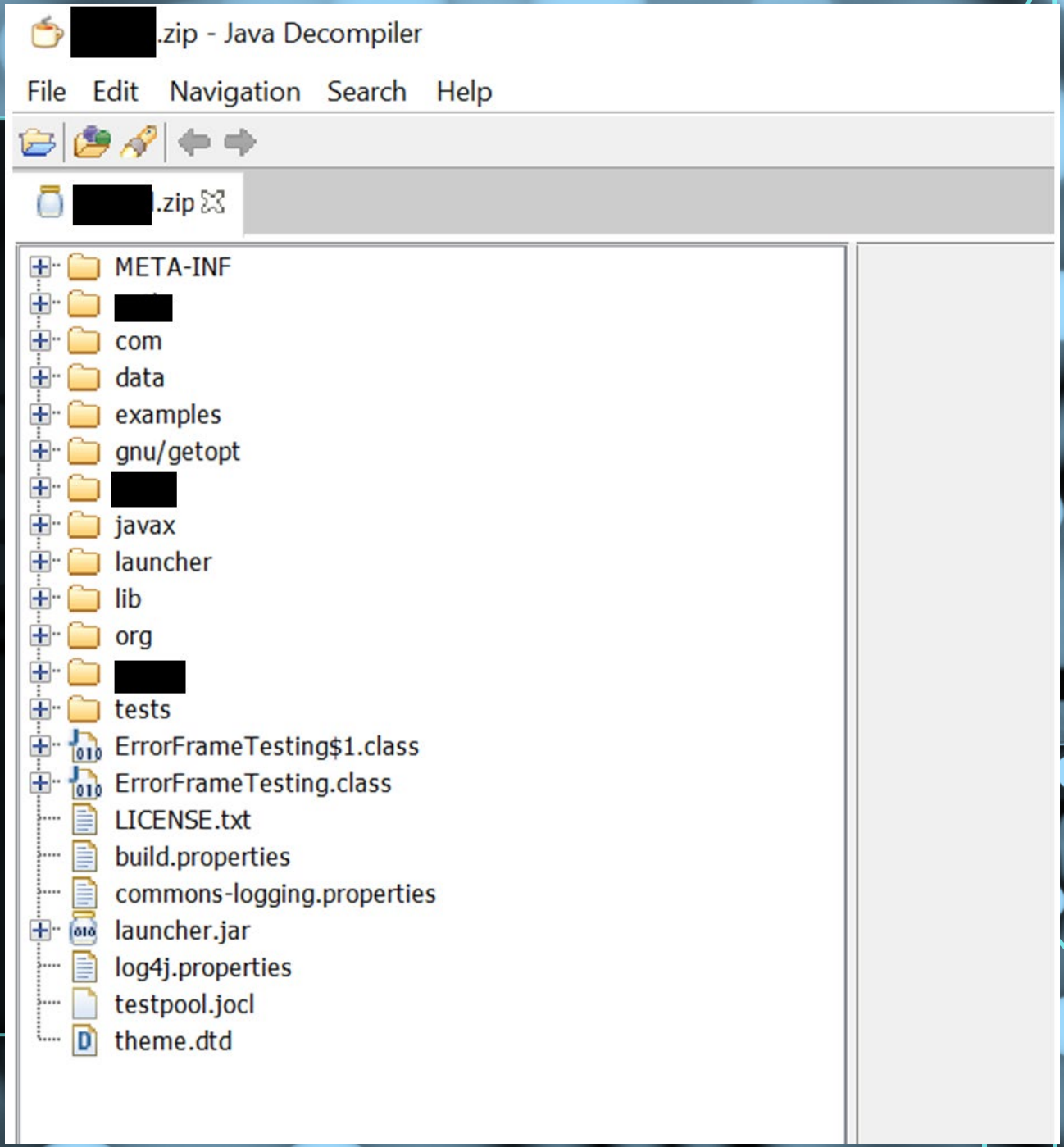
- Reverse engineering the tool

Change the file extension from .exe to .zip



- Reverse engineering the tool

Open [file].zip in jd-gui to decompile into java and export



- Reverse engineering the tool

Found what looks like the code to write a parameter in BlankEngineImpl.java!

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
} public [redacted]Response programParameter([redacted]Parameter parameter, byte[] toolSn, byte[] totalTattleTale, byte[] ecmSN)
{
    Object value;
    [redacted]Engine.EnginePacket enginePacket;
    try
    {
        byte[] valueAsBytes = parameter.getValueAsBytes();
        value = parameter.getValue();
        enginePacket = createParameterProgrammingPacket(parameter, getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(),
            (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
    }
    catch (MarshalException e)
```

- Reverse engineering the tool

The previous code calls into a.a():

```
public static byte[] a(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte firstLevel, byte lastLevel, byte[] data)
{
    return a(53159L, ecmSn, toolSn, totalTt, cmd, block, firstLevel, lastLevel, data);
}
```

After a custom hashing algorithm to calculate a crc, the calculated key is returned:

```
private static byte[] b(long crc, byte[] toolSn)
{
    byte[] password = new byte[8];
    for (int j = 0; j < 2; j++)
    {
        for (int i = 0; i < 4; i++)
        {
            crc = a(crc, toolSn[i]);
            password[i + j * 4] = (byte)(int)(crc % 19L + 65L & 0xFFL);
        }
    }
    return password;
}
```

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- ecmSN = parameter 420101 = “08101344027206021102412 3”

```
} public [redacted]Parameter getECMSerialNumberParameter()
{
    return findSParameter(66, 1);
```

```
18FFDD00    06 FFDD 00 --> FF [0019] 724201010004AE08101344027206021102412 3
```

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- toolSn = “Blue” = 426C7565

```
public static final byte[] TOOL_SN = "Blue".getBytes();
```

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- totalTt = the current count / parameter 560000

```
} private [redacted] Parameter getTotalTattleTaleParameter()
{
    return findSParameter(86, 0);
```

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- cmd = 119

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- block = first byte of parameter identifier

```
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0505FF426C756547424B524344524400
```

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- level = second (and third) byte of parameter identifier

```
18FFDDF9    06 FFDD F9 --> FF [0018] 774A0505FF426C756547424B524344524400
```

- Reverse engineering the tool

```
} protected byte[] getPPPassword(byte[] ecmSn, byte[] toolSn, byte[] totalTt, byte cmd, byte block, byte level, byte[] data)
{
    return a.a(ecmSn, TOOL_SN, totalTt, (byte)119, block, level, level, data);
```

```
getPPPassword(ecmSN, TOOL_SN, getTotalTattleTale(), (byte)119, (byte)parameter.getBlock(), (byte)parameter.getLevel(), valueAsBytes));
```

- data = the data to be written

- Reverse engineering the tool

Attempting to calculate the key from a previously recorded write command:

```
public static void main(String[] args) {
    //18FFDD00    06 FFDD 00 --> FF [0011] 774A040400 04BE 00007530 (current count)
    //18FFDDF9    06 FFDD F9 --> FF [0018] 774A0505FF426C7565 47424B5243445244 00 (write
    //it should calculate this key: 47424B5243445244
    byte[] ecmSn = HexFormat.of().parseHex("08101344027206021102412 3");
    byte[] toolSn = HexFormat.of().parseHex("426C7565"); //Blue
    byte[] totalTt = HexFormat.of().parseHex("04BE"); //04BA

    byte cmd = (byte)119;
    byte block = (byte)74; //0x4A
    byte firstLevel = (byte)5;
    byte lastLevel = (byte)5;
    byte[] data = HexFormat.of().parseHex("00");

    byte[] password = a(ecmSn, toolSn, totalTt, cmd, block, firstLevel, lastLevel, data);
    System.out.println("expected: 47424B5243445244\n"
            + "calculated: " + HexFormat.of().formatHex(password));
```

- Reverse engineering the tool

It worked!

```
expected: 47424B5243445244
calculated: 47424b5243445244
```

- Reverse engineering the tool

Transferring the key calculation to our python script. Writing the VIN worked!

```
(truckdevil.custom) write_by_identifier 0x550000 334853444A534A5236434E353832303230
4d48434d46465243
18FFDDF9    06 FFDD F9 --> FF [0034] 77550000FF426C75654D48434D4646524333485344A534A5236434E353832303230
Request:
18FFDDF9    06 FFDD F9 --> FF [0034] 77550000FF426C75654D48434D4646524333485344A534A5236434E353832303230
Response:
18FFDD00    06 FFDD 00 --> FF [0024] 77550000000A1733485344A534A5236434E353832303230
New value of ID 550000: 334853444A534A5236434E353832303230
(truckdevil.custom)
```

```
(truckdevil.custom) fuzz_test_cases 6
baselining...
18FFDDF9    06 FFDD F9 --> FF [0008] 1000000000000000
18FEFF00    06 FEFF 00 --> FF [0008] FCFFFFFFFFFFFFFF

18FFDDF9    06 FFDD F9 --> FF [0008] 1600000000000000
18FE9400    06 FE94 00 --> FF [0008] 0000000800000000

18FFDDF9    06 FFDD F9 --> FF [0008] 2300000000000000
18FEFF00    06 FEFF 00 --> FF [0008] FCFFFFFFFFFFFFFF

18FFDDF9    06 FFDD F9 --> FF [0008] 3700000000000000
18FEFF00    06 FEFF 00 --> FF [0008] FCFFFFFFFFFFFFFF

18FFDDF9    06 FFDD F9 --> FF [0008] 4B00000000000000
18FEFF00    06 FEFF 00 --> FF [0008] FCFFFFFFFFFFFFFF

18FFDDF9    06 FFDD F9 --> FF [0008] 5F00000000000000
18FEFF00    06 FEFF 00 --> FF [0008] FCFFFFFFFFFFFFFF

18FFDDF9    06 FFDD F9 --> FF [0008] 6D00000000000000
18FFDD00    06 FFDD 00 --> FF [0008] 6D00C80BFFFFFFFF

18FFDDF9    06 FFDD F9 --> FF [0008] 7000000000000000
18FFDD00    06 FFDD 00 --> FF [0008] 700000000A0A16FF

18FFDDF9    06 FFDD F9 --> FF [0008] 7200000000000000
18FFDD00    06 FFDD 00 --> FF [0008] 720000000B0A16FF
```

- Are there other service identifiers besides read/write/diagnostic session control?

```
(truckdevil.custom) fuzz_test_cases 10
baselining...
18FF41F9    06 FF41 F9 --> FF [0008] FFFFFFFFFFFFFFFF
18E8FF00    06 E800 00 --> FF [0008] 00FFFFFFF941FF00

18FF5CF9    06 FF5C F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FE9400    06 FE94 00 --> FF [0008] 0000000700000000

18FF5DF9    06 FF5D F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FE9400    06 FE94 00 --> FF [0008] 0000000700000000

18FF94F9    06 FF94 F9 --> FF [0008] FFFFFFFFFFFFFFFF
18E8FF00    06 E800 00 --> FF [0008] 01FFFFFFF994FF00

18FFA9F9    06 FFA9 F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FFA900    06 FFA9 00 --> FF [0008] FF0101FFFFFFFFFF

18FFAAF9    06 FFAA F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FFAA00    06 FFAA 00 --> FF [0008] FF0101FFFFFFFFFF

18FFABF9    06 FFAB F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FFAB00    06 FFAB 00 --> FF [0008] FF0101FFFFFFFFFF

18FFACF9    06 FFAC F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FFAC00    06 FFAC 00 --> FF [0008] FF0101FFFFFFFFFF

18FFBEF9    06 FFBE F9 --> FF [0008] FFFFFFFFFFFFFFFF
18FE9400    06 FE94 00 --> FF [0008] 0000000700000000
```

- What are the other proprietary messages?

(PGNs FF00-FFFF)

- Let’s fuzz these protocols!

Pros:

- will find cases of improper input validation and error handling
- limited setup and knowledge of protocol needed
- can run 24/7 with little oversight

Cons:

- cannot traverse all program paths
- only tests cases that would cause the ECU to crash
- fuzzing an ECU is a lot slower than fuzzing a native or remote application

- Fuzzing the protocol on PGN 0xFFDD

```
Welcome to the truckdevil J1939 Fuzzer.
(truckdevil.j1939_fuzzer) target add 0 60928
(truckdevil.j1939_fuzzer) target
address: 0   reboot_pgn: 60928 reboot_data_snip: not set
(truckdevil.j1939_fuzzer) set mode 1
(truckdevil.j1939_fuzzer) set generate_data_option 2
(truckdevil.j1939_fuzzer) set message_frequency 0.2
(truckdevil.j1939_fuzzer) set test_case_can_id 0x18FFDDF9
(truckdevil.j1939_fuzzer) record_baseline
Baselining for 60 seconds...
Baselining complete.
(truckdevil.j1939_fuzzer) generate_test_cases
Creating 5000 messages to fuzz...
(truckdevil.j1939_fuzzer) start_fuzzer
Sending: [.......................................] 11/5000
```

- Fuzzing all protocols that may be running in the proprietary range (PGN FF00-FFFF)

```
Welcome to the truckdevil J1939 Fuzzer.
(truckdevil.j1939_fuzzer) target add 0 60928
(truckdevil.j1939_fuzzer) target
address: 0   reboot_pgn: 60928 reboot_data_snip: not set
(truckdevil.j1939_fuzzer) set mode 1
(truckdevil.j1939_fuzzer) set generate_data_option 2
(truckdevil.j1939_fuzzer) set message_frequency 0.1
(truckdevil.j1939_fuzzer) set test_case_priority 6
(truckdevil.j1939_fuzzer) set test_case_reserved_bit 0
(truckdevil.j1939_fuzzer) set test_case_data_page_bit 0
(truckdevil.j1939_fuzzer) set test_case_pdu_format 0xff
(truckdevil.j1939_fuzzer) set test_case_src_address 0xf9
(truckdevil.j1939_fuzzer) record_baseline
Baselining for 60 seconds...
Baselining complete.
(truckdevil.j1939_fuzzer) set num_messages 20000
(truckdevil.j1939_fuzzer) generate_test_cases
Creating 20000 messages to fuzz...
(truckdevil.j1939_fuzzer) start_fuzzer
Sending: [.......................................] 46/20000
```

- When the fuzzer detects a crash, it looks like this:

```
(truckdevil.j1939_fuzzer) start_fuzzer
Sending: [####.................................] 2022/20000
    source: 0
        interval messages/second: 254.7
        baseline messages/second: 280.31666666666666
        Reason: targets reboot message was detected.
    Stored previous interval fuzzed messages to: crashReport_1636066351_previous_1
    Stored current interval fuzzed messages to: crashReport_1636066351_current_1
Please restart the ECU. Once complete, enter 'y' to continue / 'q' to quit fuzzing:
```

- Other things to investigate:

- How does modifying various parameters affect the vehicle when running?
- In a running vehicle, what other data is the ECU normally accepting from other ECUs?
- The diagnostic tool also allows for running diagnostic tests
- The ECU is requesting PGN 0xFEE6 (date/time) from node 0x96

```
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
18EA9600    06 EA00 00 --> 96 [0003] E6FE00
0CF00400    03 F004 00 --> FF [0008] F87D7D000000F07D
```

- Future work:

- Reverse engineer the diagnostic tools, set up a debugger, look for other possible admin functionality
- Open the ECU, pull firmware off and look for hard-coded secrets and bugs, look for remote vectors
- Acquire the manufacturer’s telematics unit, and perform additional work on the back-end systems to find remote access vector to ECU
- Perform assessment on manufacturer’s full running vehicle, review any gateways or mitigations that may be in place

Thank you!

Questions?