Cyber CAN I haz car secret plz?

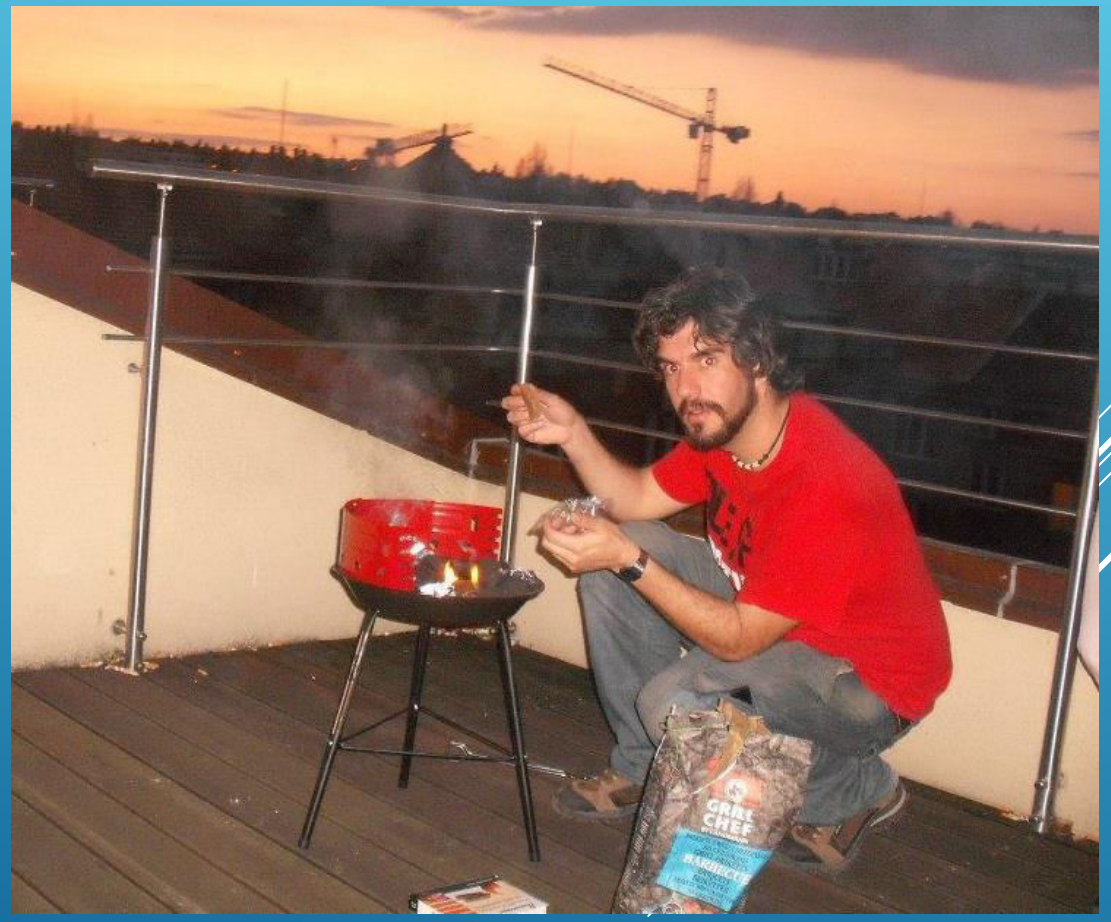


Javier Vazquez Vidal

Henrik Ferdinand Nölscher

Who are we?

- Javier is a Hardware Security Specialist
- Works at Code White as a Product Security Engineer
- He is from Cadiz (Spain)
- Enjoys reversing products that are interesting, or could potentially be more fun
- Likes cake (when it’s not a lie) and bbqs



Who are we?

- Ferdi works as an Information Security Specialist at Code-White in Ulm
- Among many things, he likes pwning stuff, lasers and BBQs
- Likes big busses and he cannot lie



What options did we have?



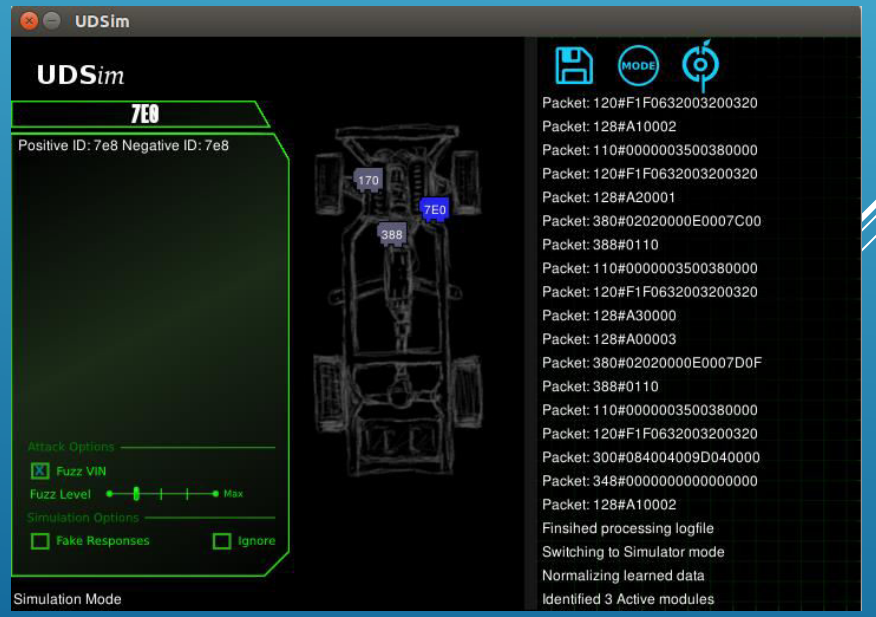


Current Car Hacking status

- Focus is on the CAN bus
- Replay attacks and/or packet injection
- Some researchers also found remote exploits to take control of certain vehicles
- There are some tools to help understand the UDS protocol







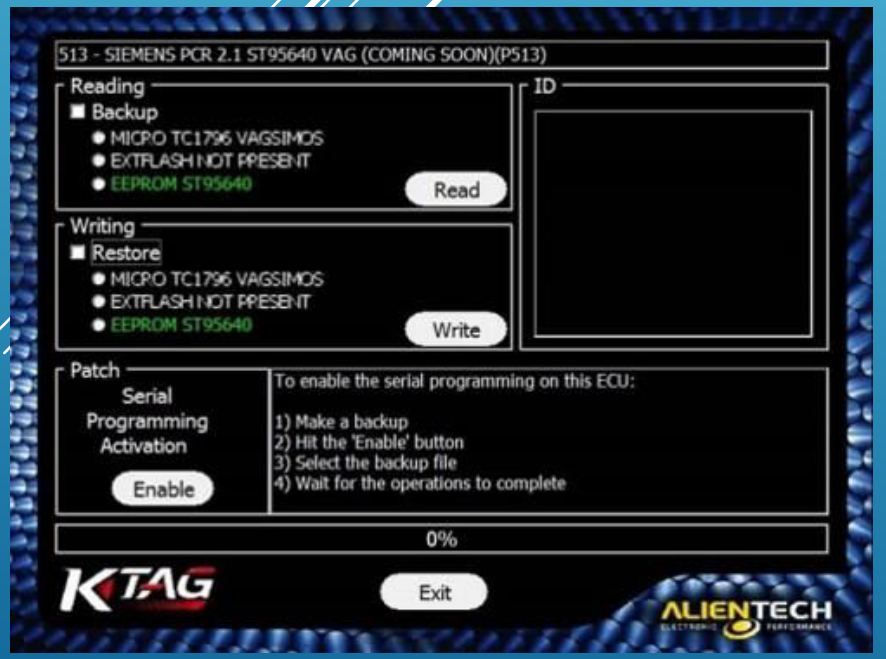
Is there anything else?

- Chiptuning is actually hacking
- ECUs are cloned
- Internal data is manipulated
- OEM specific Diagnostics are more fun than standard ones







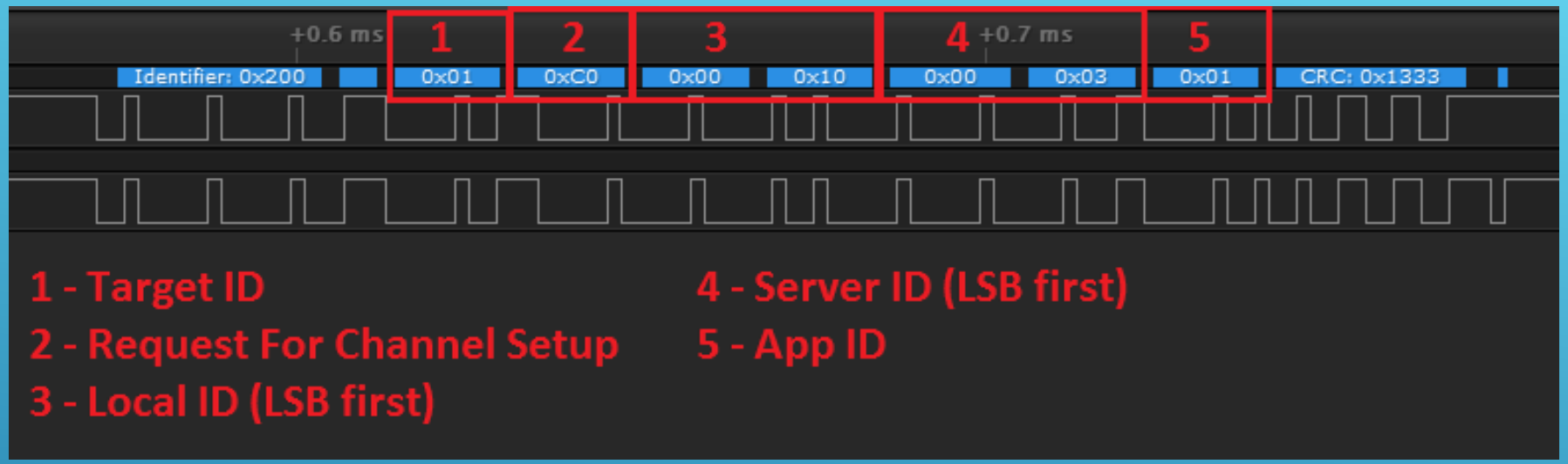


So, what's the secretz?

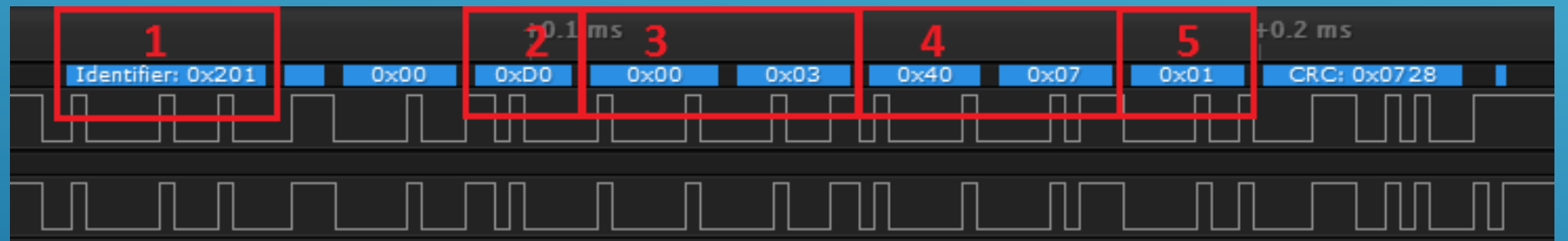
- UDS is not the only protocol
- TP2.0 (Tunneling Protocol) is the CAN version of KWP2000
- Both offer a series of services which are very interesting, but often not taken into consideration.
- With use of these services, a lot of Information can be gathered and modified.



UDS VS TP 2.0 -> TP 2.0 Channel Negotiation

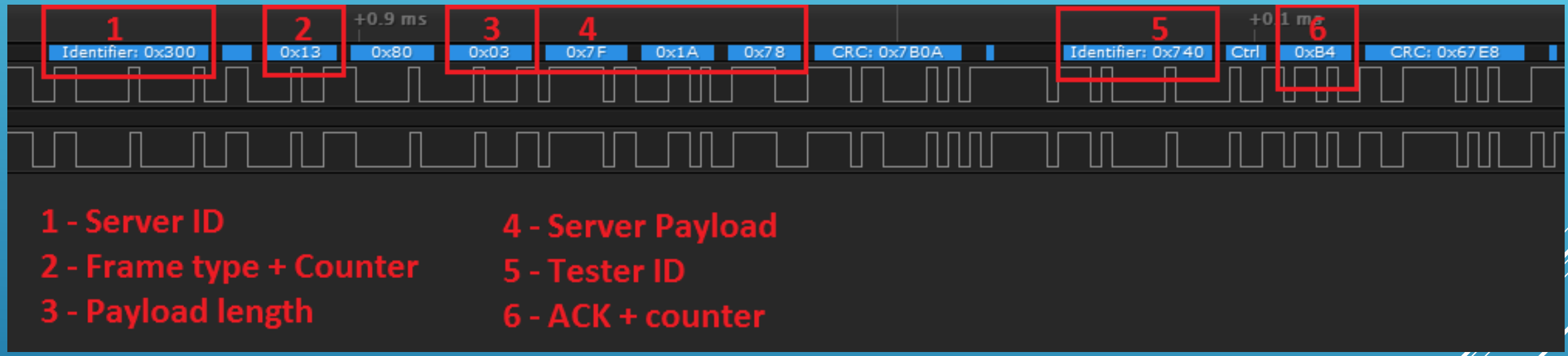


1 - Target ID
2 - Request For Channel Setup
3 - Local ID (LSB first)
4 - Server ID (LSB first)
5 - App ID

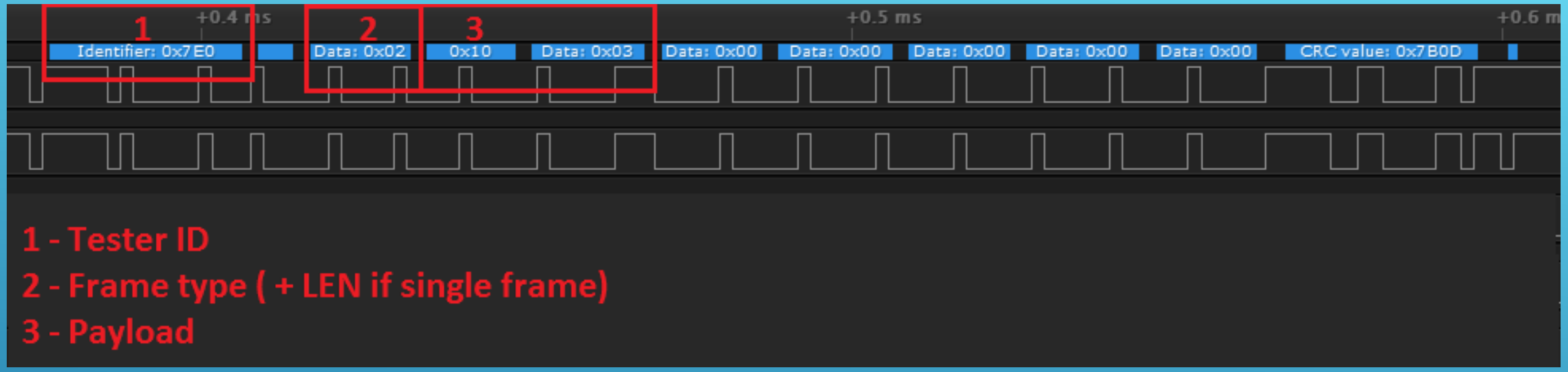


1 -Channel neg. ID + Target ID
2 - Positive reply to Channel Negotiation
3 - Server ID (LSB first)
4 - Tester ID (LSB first)
5 - App ID

UDS VS TP 2.0 -> TP 2.0 Transmission



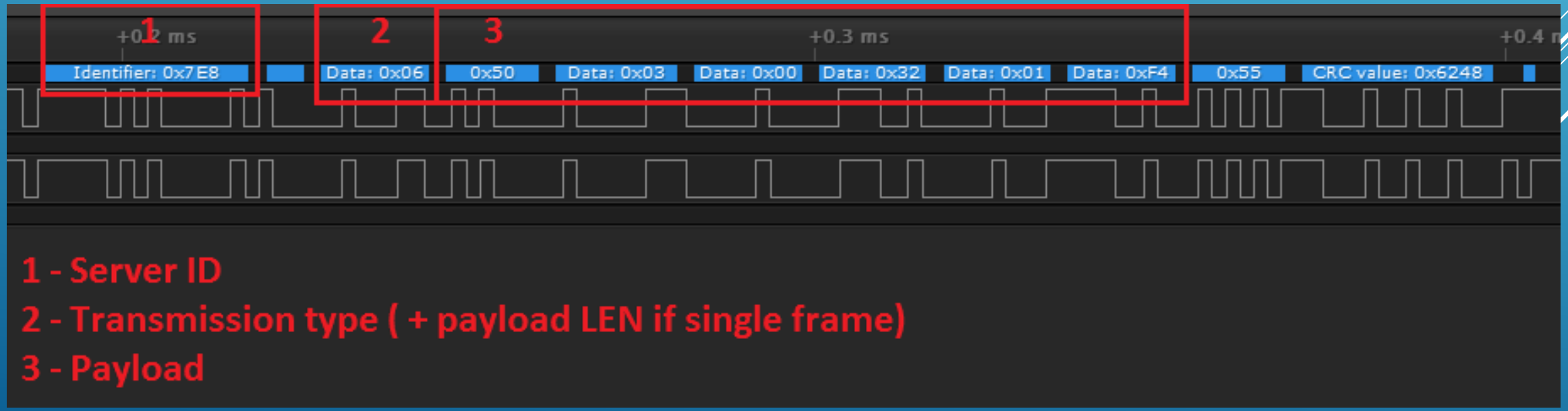
UDS VS TP 2.0 -> UDS Transmission



1 - Tester ID

2 - Frame type (+ LEN if single frame)

3 - Payload



1 - Server ID

2 - Transmission type (+ payload LEN if single frame)

3 - Payload

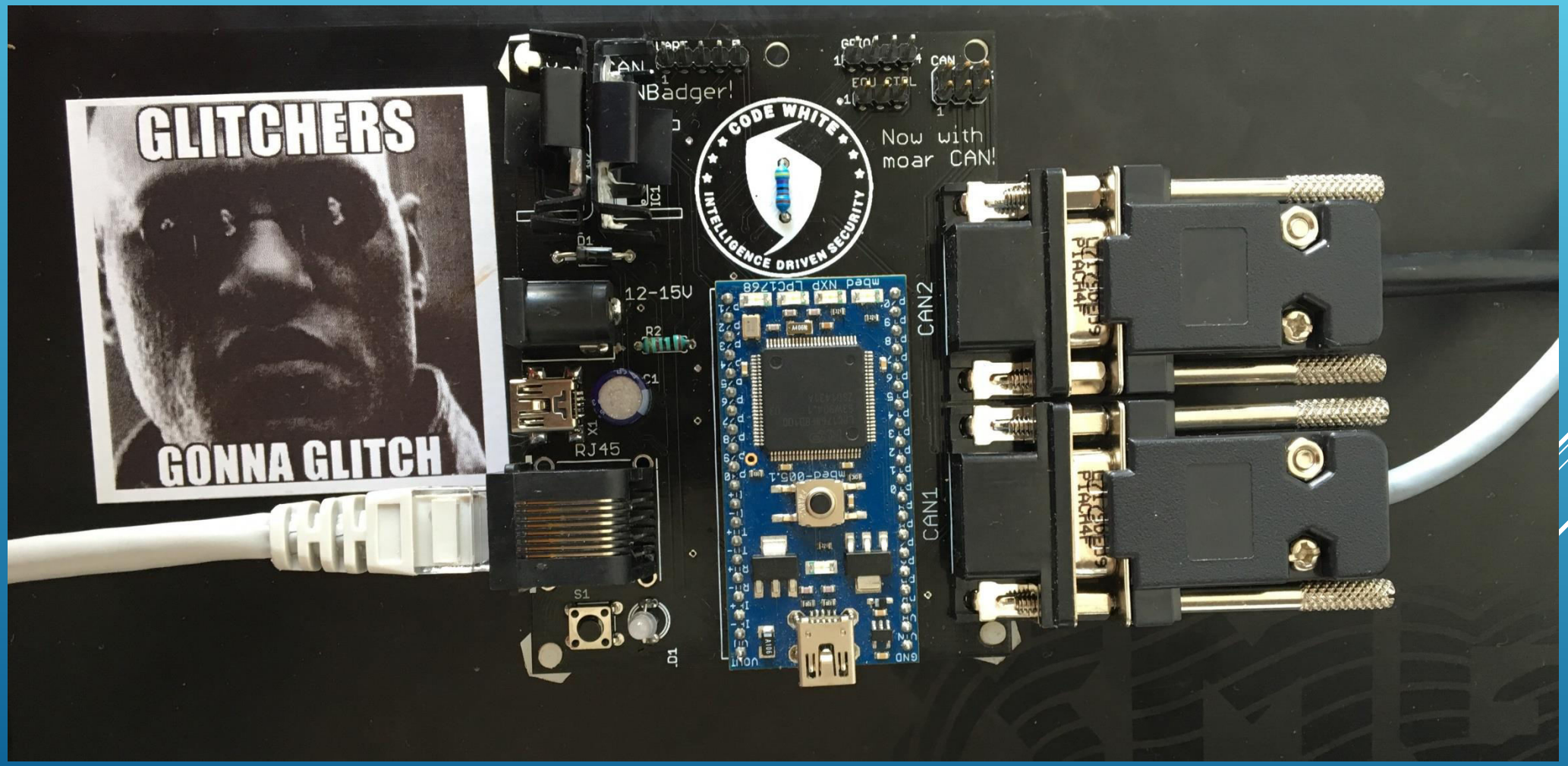
Some of the differences between UDS and TP SIDs

Funktionsgruppe	SID UDS	SID KWP	Default Session	Bezeichnung UDS	Bezeichnung KWP 2000	Beschreibung
Diagnostic and Communication Management	0x10		X	DiagnosticSessionControl	StartDiagnosticSession	Steuerung der Diagnosesitzung
		0x20			StopDiagnosticSession	
	0x3E		X	TesterPresent		
	0x27			SecurityAccess		
	0x11		X	ECUReset		
		0x81	X		StartComm (Verbindungsaufbau)	Steuerung der Buskommunikation
		0x82	X		StopComm (Verbindungsabbau)	
	0x28			CommunicationControl	DisableNormalMessageTransm.	
		0x29			EnableNormalMessageTransm.	
	0x85			ControlDTCSetting		
	0x83			AccessTimingParameters		
	0x87	0x84		LinkControl	NetworkConfiguration	
	0x86		(X)	ResponseOnEventService		
Data Transmission		0x21			ReadDataByLocalIdentifier	Lesen und Schreiben von Messwerten und Steuergerätedaten
	0x22		(X)	ReadDataByIdentifier	ReadDataByCommonIdentifier	
		0x1A	X		ReadECUIdentification	
	0x23		(X)	ReadMemoryByAddress		
		0x3B			WriteDataByLocalIdentifier	
	0x2E		(X)	WriteDataByIdentifier	WriteDataByCommonIdentifier	
	0x3D		(X)	WriteMemoryByAddress		
	0x2A	0x26		ReadDataByPeriodicIdentifier	SetDataRates	
	0x2C		X	DynamicallyDefineDataIdentifier		

The most interesting Services

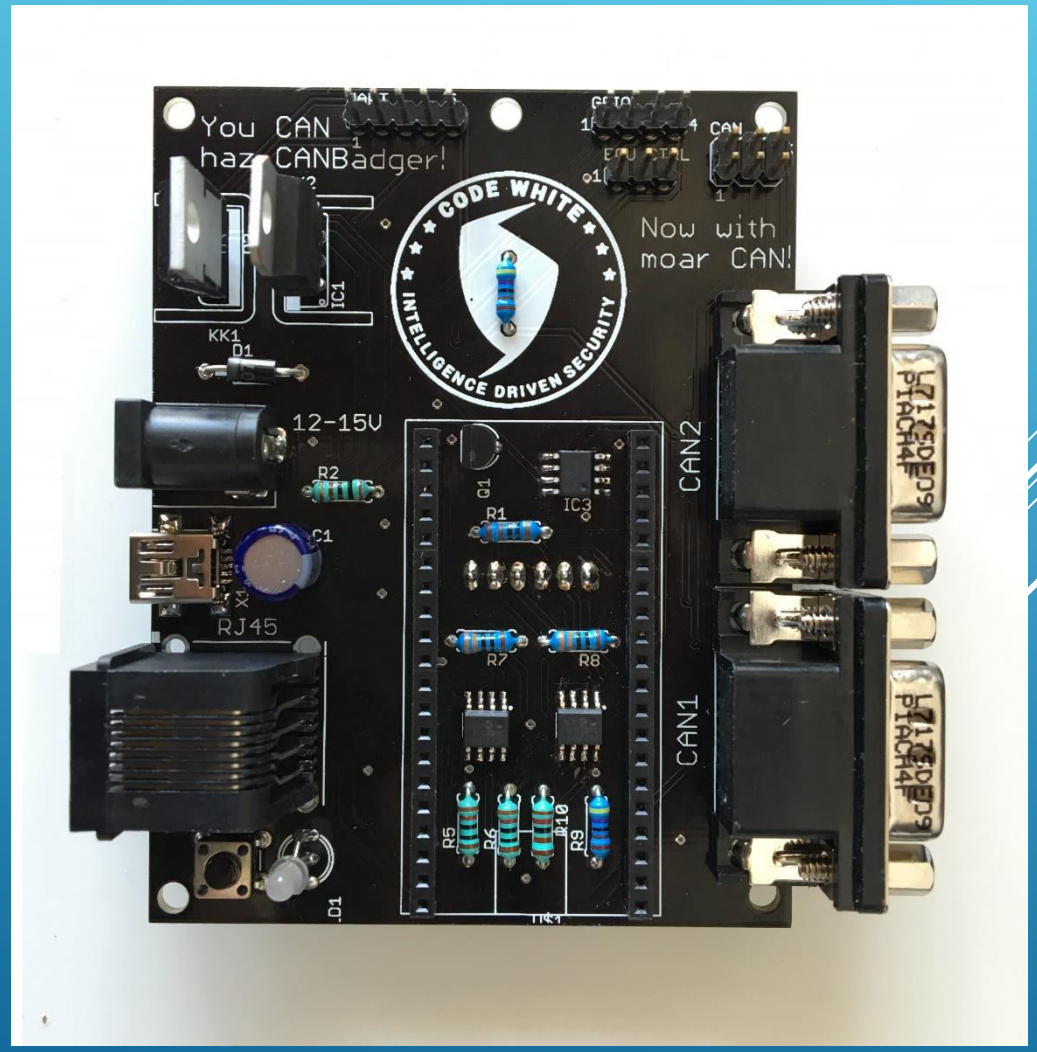
- SecurityAccess (0x27): Allows access to restricted Services
- ReadMemoryByAddress (0x23): Allows to read certain memory addresses
- Read/WriteDataByID(0x22/0x2E): Allows to read/write certain parameters
- RequestUpload(0x35): Service used to retrieve firmware from the ECU
- RoutineControl(0x31): Allows to start “routines”, which have different effects

Say hello to the CANBadger!



CANBadger Hardware Overview

- Powered by mBed LPC1768 or LPCXPresso LPC1769
- 128KB XRAM
- 2x DB9 CAN Interfaces + 2x Debug headers
- SD card
- ECU Power control by software
- UART
- 4 GPIOs
- Standalone mode, USB mode (CDC Device), or Network mode
- Can be powered by PSU, External battery, or OBD2
- Has a blinky dual color LED. Everyone loves blinky LEDs, right?
- Complete board assembly under $25



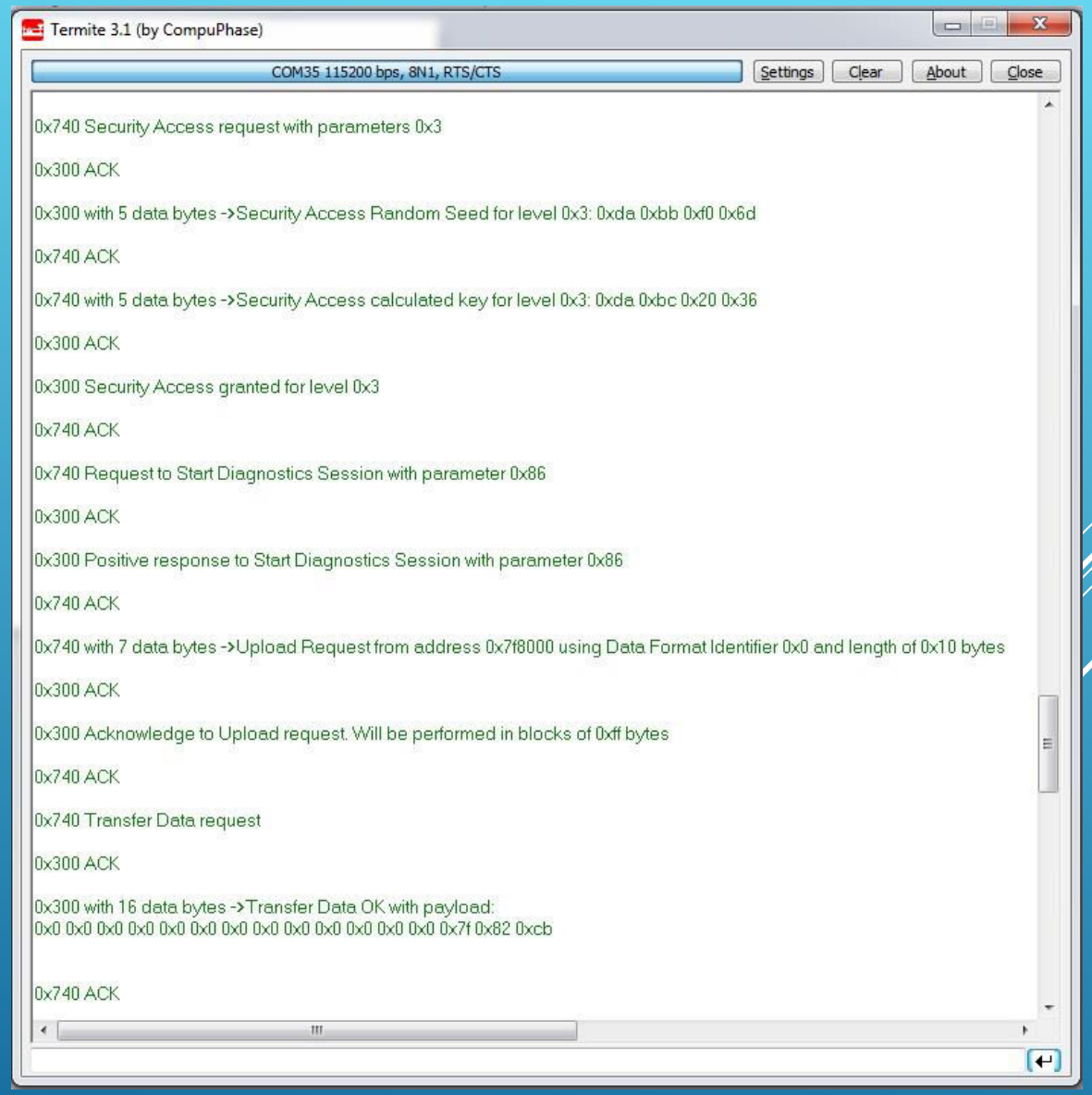
CANBadger POC Firmware Features

- All actions are handled by the CANBadger firmware
- Supports UDS, TP2.0 and RAW CAN
- Diagnostics Sessions are interactive (realtime)
- MITM with rules stored in SD
- Hijack SecurityAccess
- Emulate SIDs/PIDs from log
- Dump Data Transfers to SD
- Log UDS and TP2.0 traffic with very detailed verbosity in standard or bridge mode
- Log RAW CAN traffic in standard or bridge mode
- Detect CAN speed
- Ethernet for use with GUI



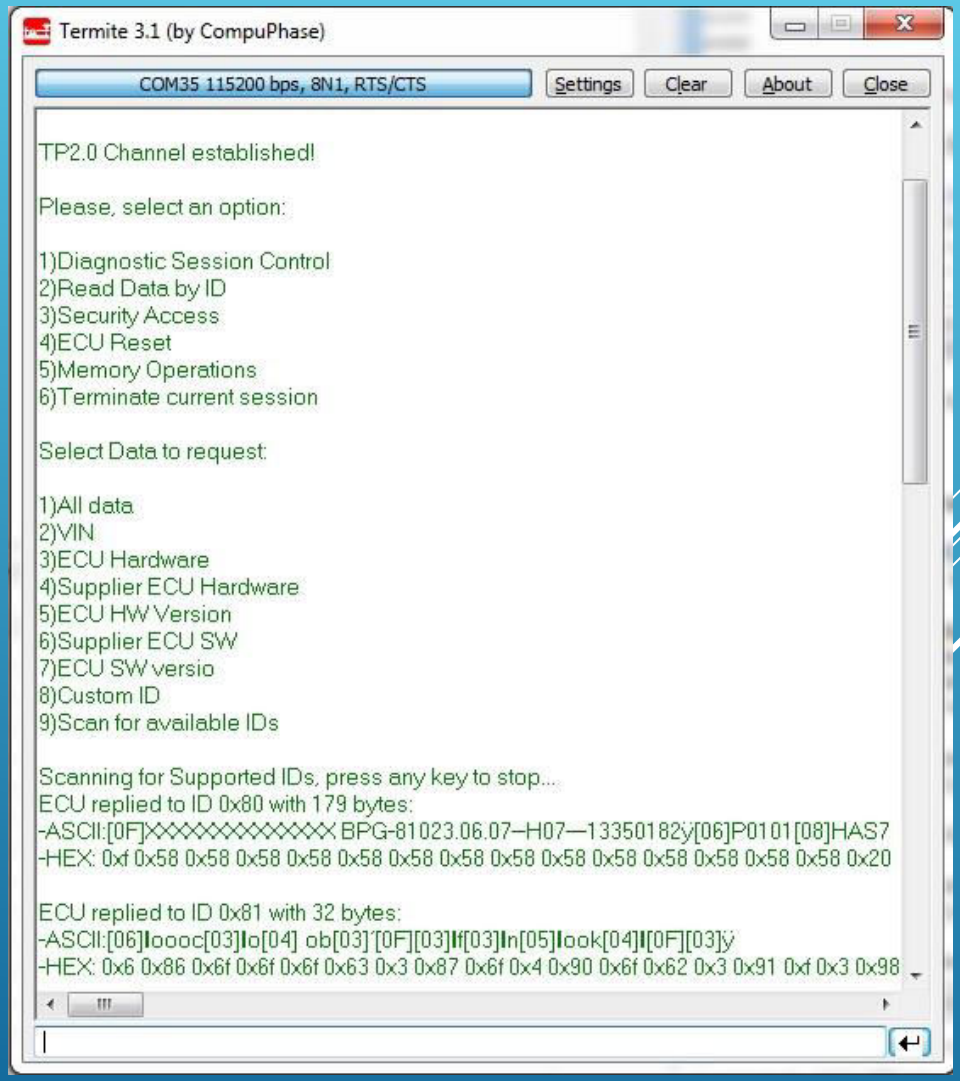
Protocol Analysis

- Many SIDs already included in firmware
- Extremely fast to add support for new SIDs
- High verbosity
- Logging and parsing is done by the CANBadger firmware
- Logs are stored in the SD card and can be viewed and retrieved without removing it
- Works with UDS and TP2.0



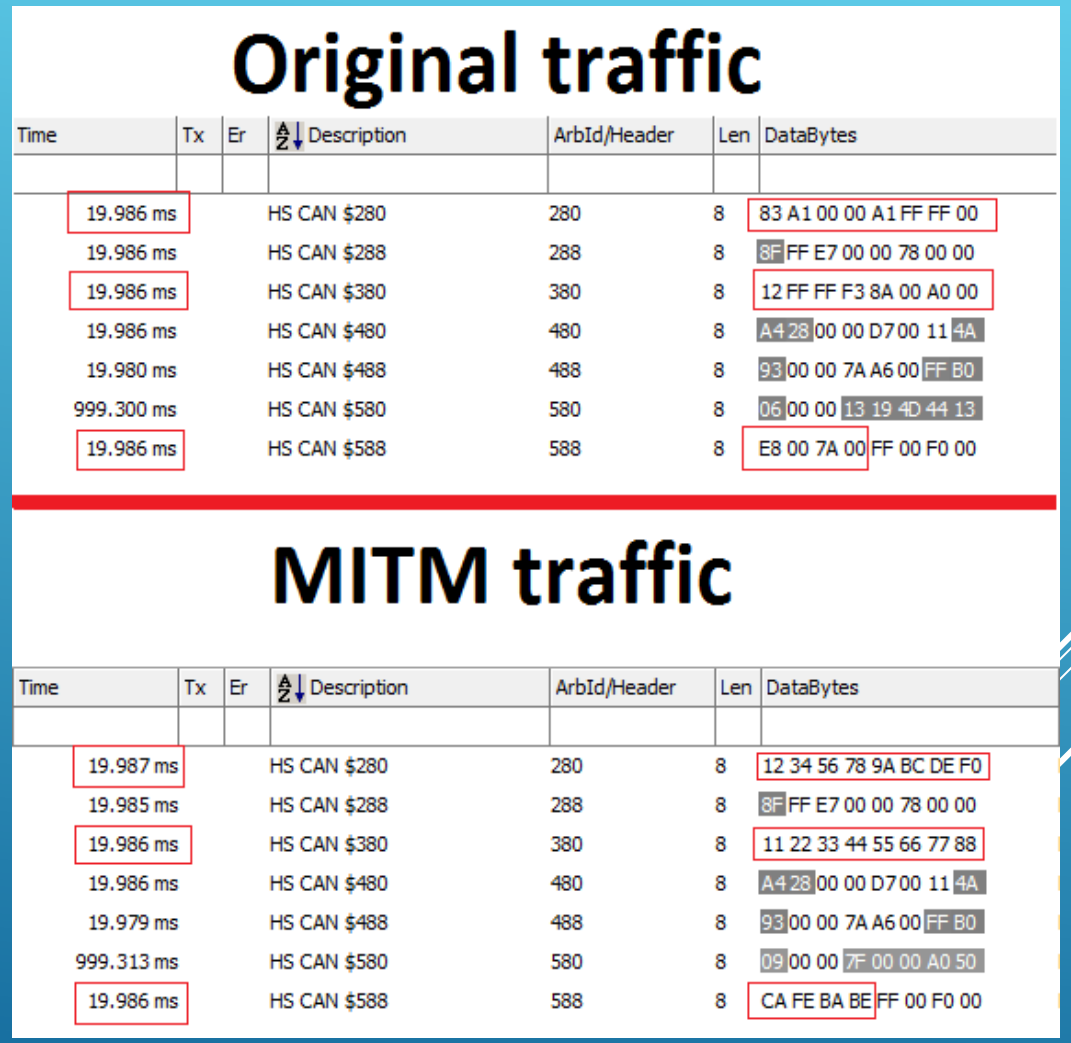
TP2.0 and UDS Interactive Session

- No scripting required
- Allows to perform actions on the go
- Built-in scanners for SID parameters



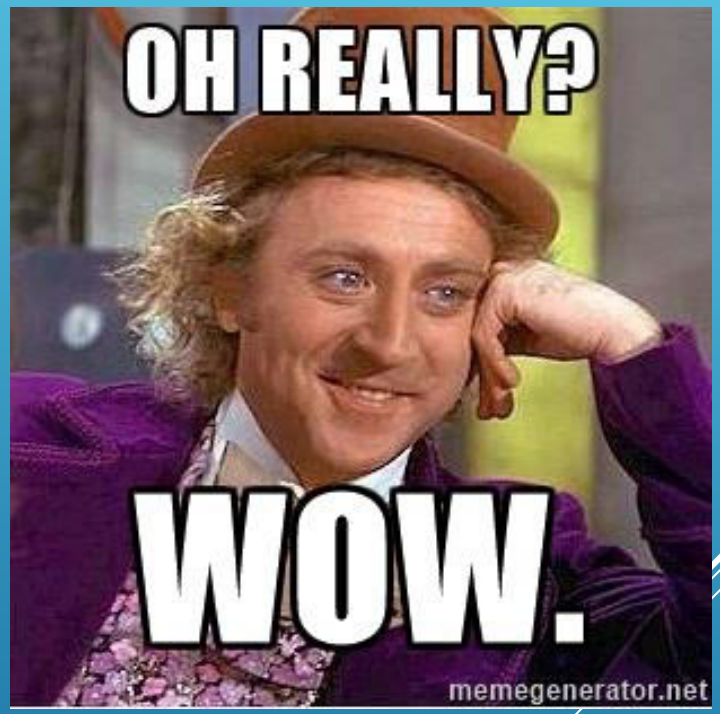
MITM

- Handled by the CANBadger Firmware (real-time!)
- Rules are set for specific IDS
- Manipulate individual bytes:
  - Swap for fixed value
  - Add, substract, multiply, divide
  - Increase or decrease percent
- Conditions can be set to:
  - Whole payload matches
  - Specific bytes match
  - Specific bytes are greater or smaller than X value



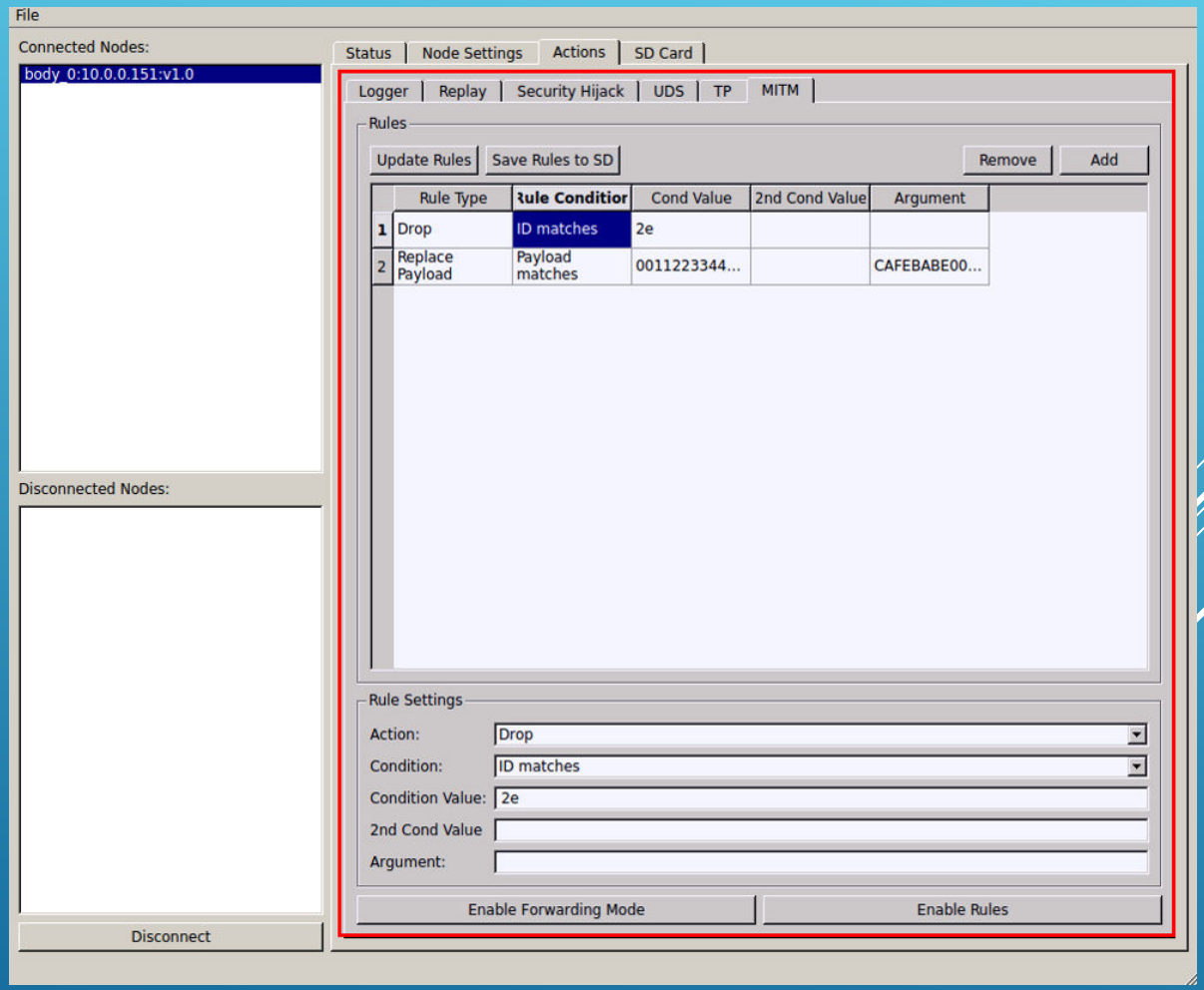
CANBadger Server

- Remotely control multiple CanBadgers!
- Simple UDP Protocol
  - Node2Server
  - Node2Node
- Cross-Platform
  - Python
  - Qt for GUI
  - Hackable Protocol Abstractions



CANBadger Server

- Displays status for connected CANBadgers
- All operations organized in tabs
- Easily modify/add parameters and perform actions
- Exchange data between tabs



SecurityAccess Hijack. Why?

- OEM tools and some third-party tools authenticate themselves to ECUs in order to gain access to restricted features. Who doesn't like restricted stuff?
- These tools have fixed functions, so you have no control over the process other than pressing buttons.
- Because you CAN!



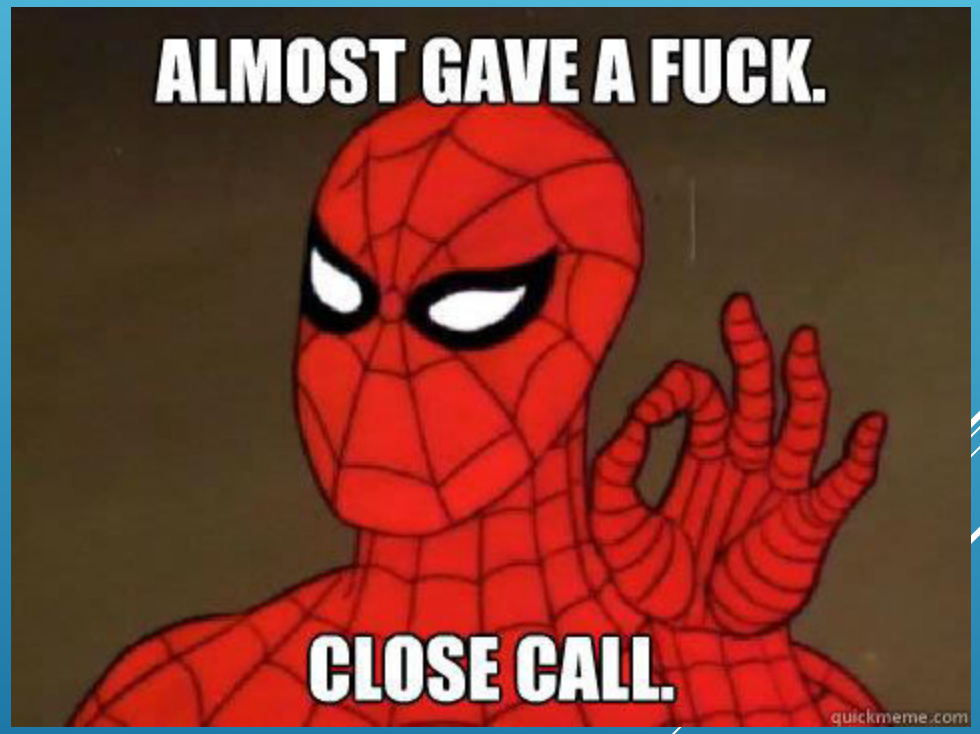
SecurityAccess Hijack. How?

- Wait for a SecurityAccess request in transparent bridge mode
- Switch to the desired type of Diagnostics Session
- SecurityAccess auth is forwarded
- Success? Cut off the external tool and take control over the Session!



So, we survived the demo (hopefully!)

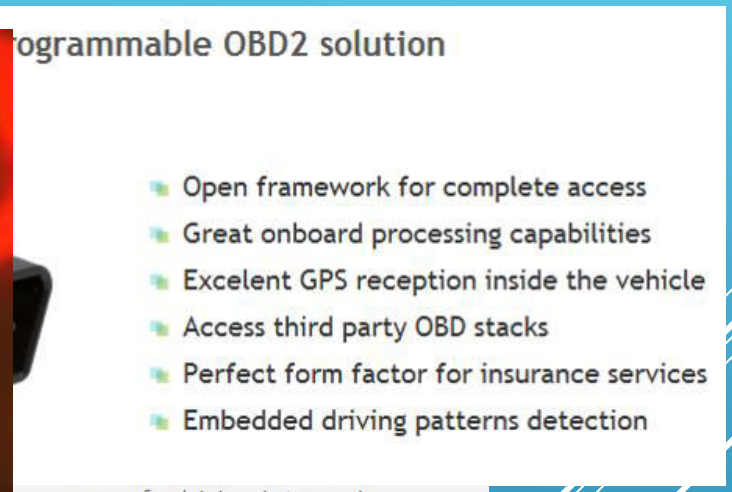
- What else can the CANBadger do?
  - Dump TP and UDS transfers, which are used for firmware updates.
  - Spoof OBD2 data thru MITM and Emulator
  - Use GPIO pins for bootloading (Tricore f.ex)
  - Manipulate GPS signals via UART pins

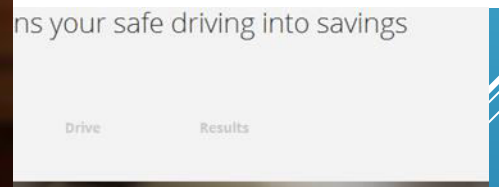


Wait, why did you mention GPS?











Insurance and Tracking dongles

- They implicitly trust the car though they cannot really know where the OBD2 data is coming from
- They are dependent on a GPS module that outputs data via UART

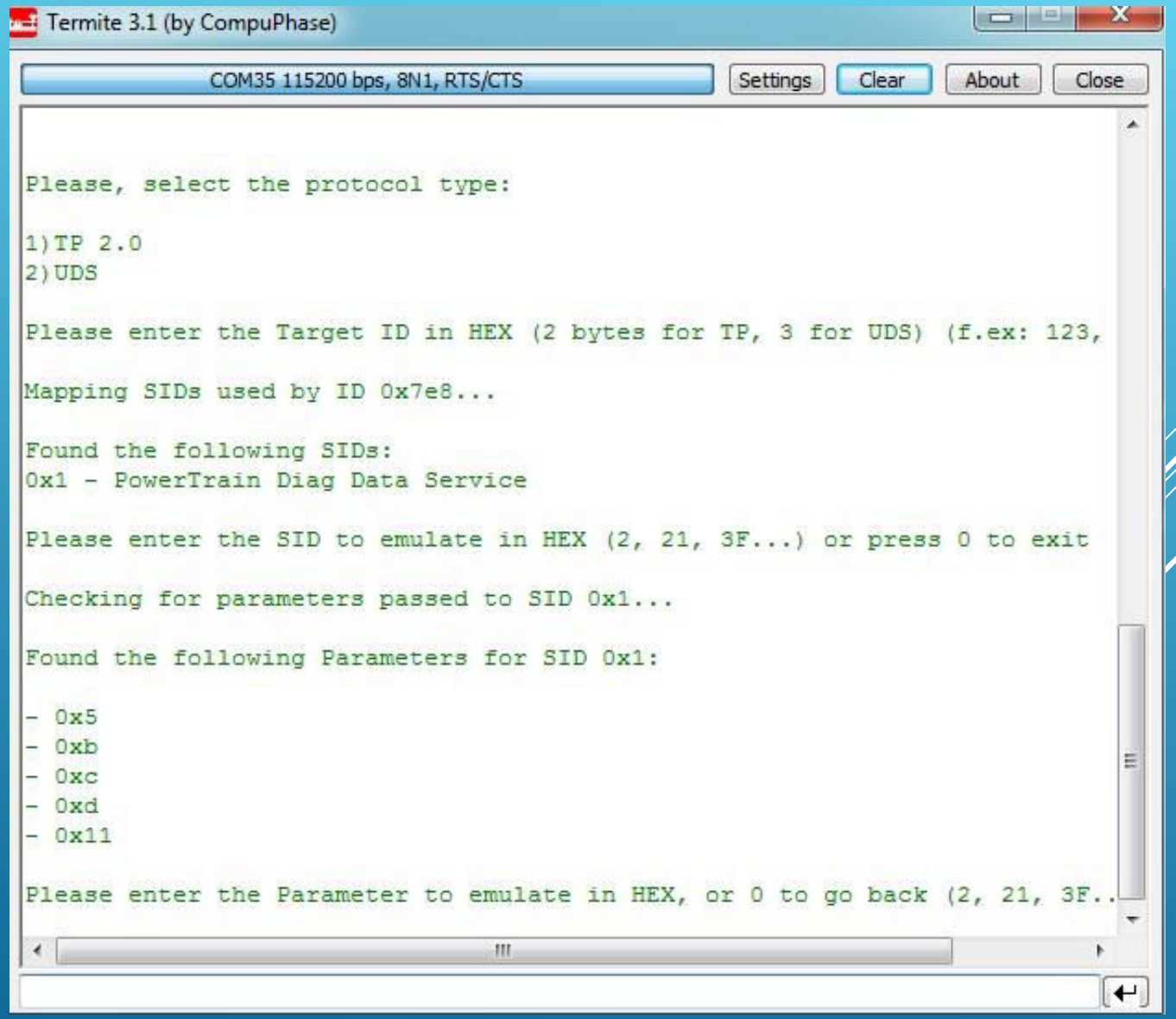
- By spoofing the OBD2 data, you can have your own “driving habits”
- By spoofing your location, well, you can guess… ☺
- Mix both of them
- ???
- Profit!

Insurance and Tracking dongles



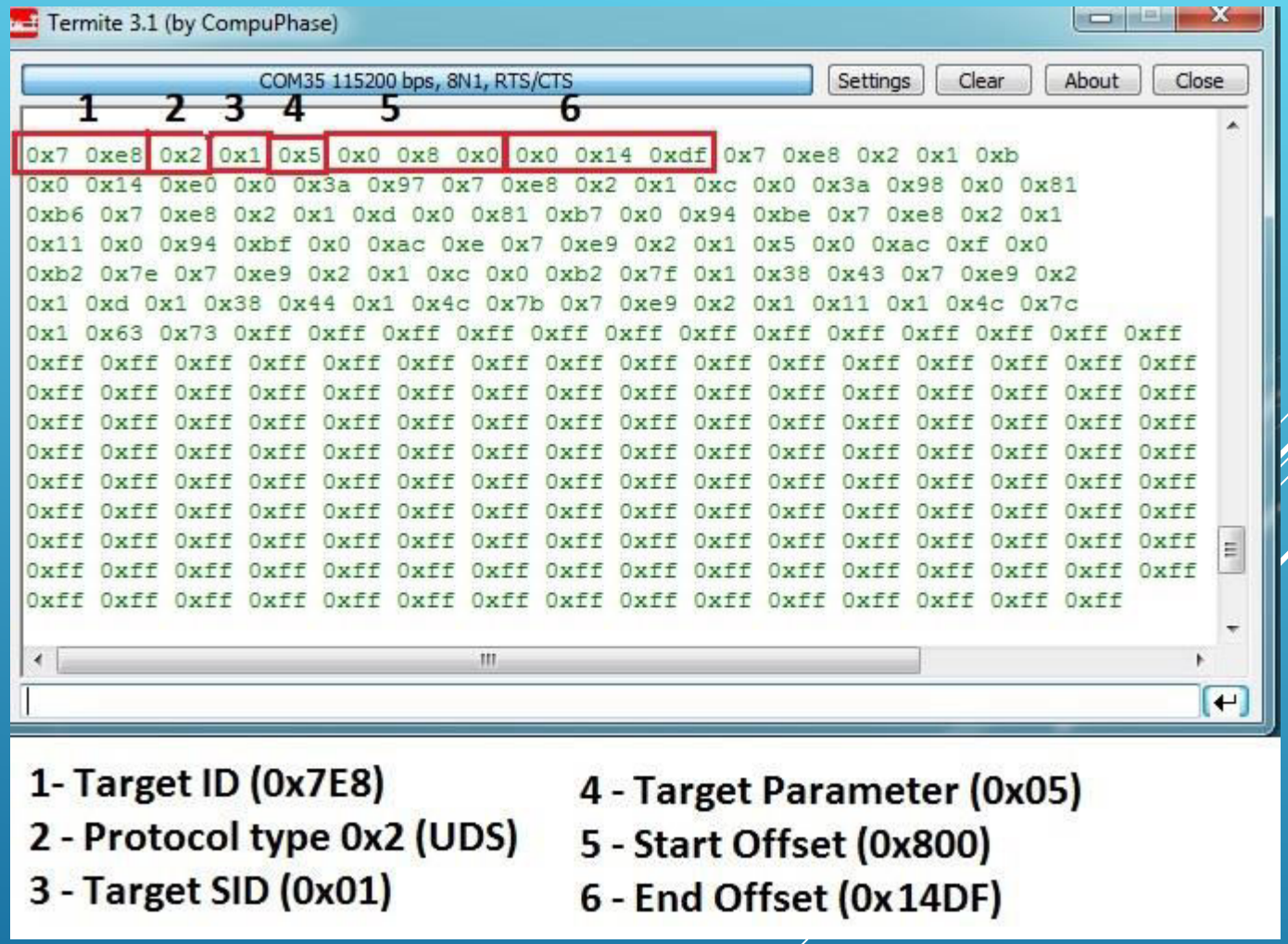
How does the emulator work?

- The CANBadger will make requests for specified PIDs every 10 ms in a loop during desired time, dumping the replies to SD.
- Once the data is stored, you can choose from which IDs, SIDs and PIDs you want to create emulation files.



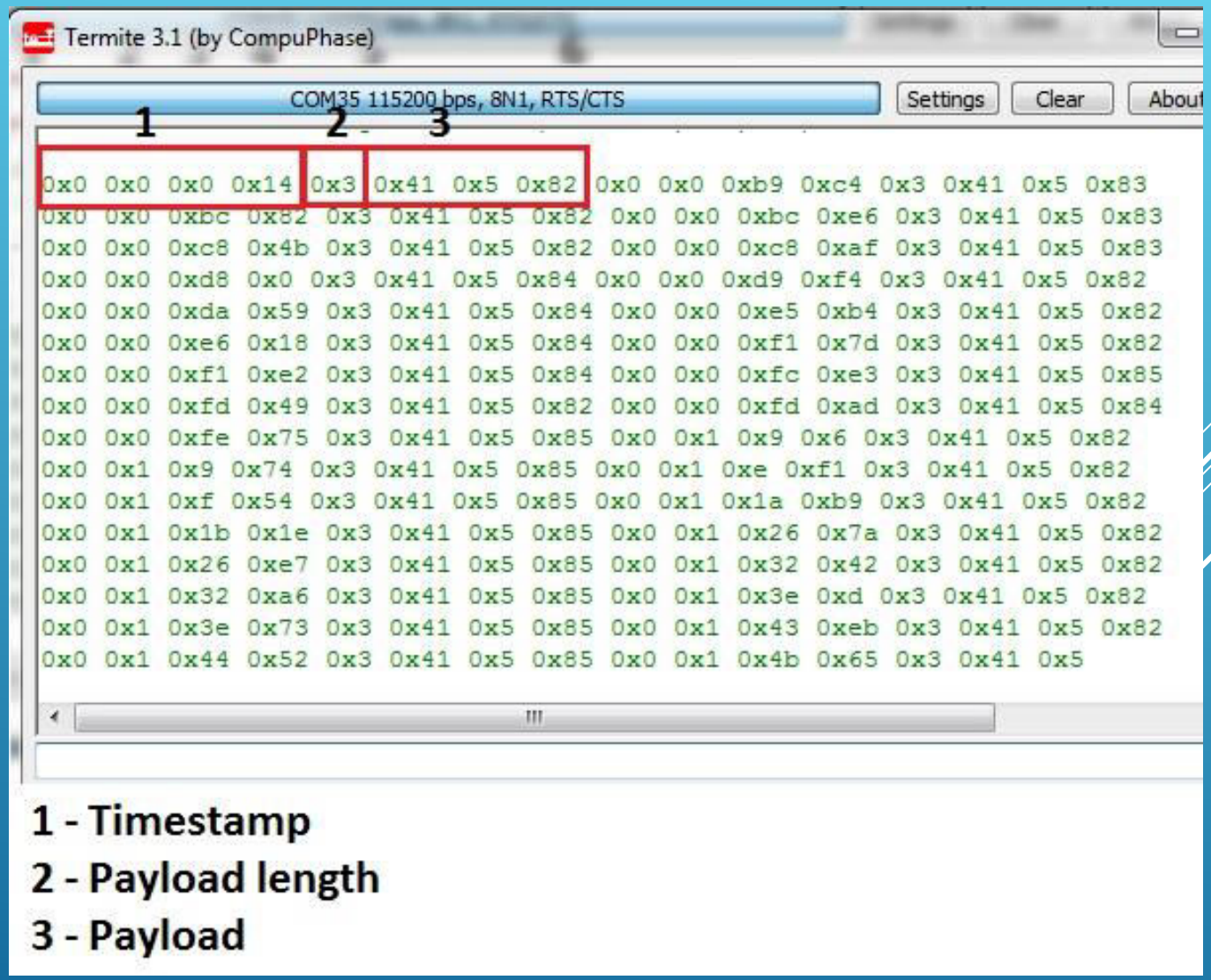
How does the emulator work?

- The emulator data is mapped into the XRAM.
- The first 0x800 bytes are lookup tables which contain ID, Protocol, SID, PID, Start Offset and End Offset
- All requests from the victim are passed to the target system.
- When a reply is received from the target, the CANBadger will lookup these tables to see if an entry is found for it.
- If an entry is found, it will pass the emulated data instead of the original one.



How does the emulator work?

- Emulation data is stored with timestamps.
- The CANBadger will pass the appropriate data for each moment, so even if requests are random, data will still be right.
- The emulation data is optimized, storing only changes within the timeframe.
- When there is no more emulation data left for the current timeframe, the whole emulation will be restarted from the beginning



Wanna see moar?

- We will be waiting for you at the “You CAN haz car secretz” workshop, ready to answer all your questions and to provide more info.
- Do you want to assemble your own CANBadger like, right now? Come visit us!
- Code and schematics are GPL and will be uploaded to github shortly after DC

Thanks!

- To all of you for being here today!
- To Code White for their support and trust in the project
- To our family and friends for supporting us even when we run out of coffee



Javier Vazquez
@fjvva
Javier.Vazquez.vidal@code-white.com

Ferdinand Nölscher
@s1ckcc
Ferdinand.noelscher@code-white.com