UDS over CAN

CAN Bus laboratory - Cyber Physical System and IoT Security - 23/24
Alessandro Brighente

Tommaso Bianchi - tommaso.bianchi@phd.unipd.it

SPRITZ
Security & Privacy
Research Group

UDS over CAN

- UDS On CAN Bus (UDSonCAN)
- Diagnostic over CAN (DoCAN)

Standard: ISO 14229

Standardized across both manufacturers *and* standards

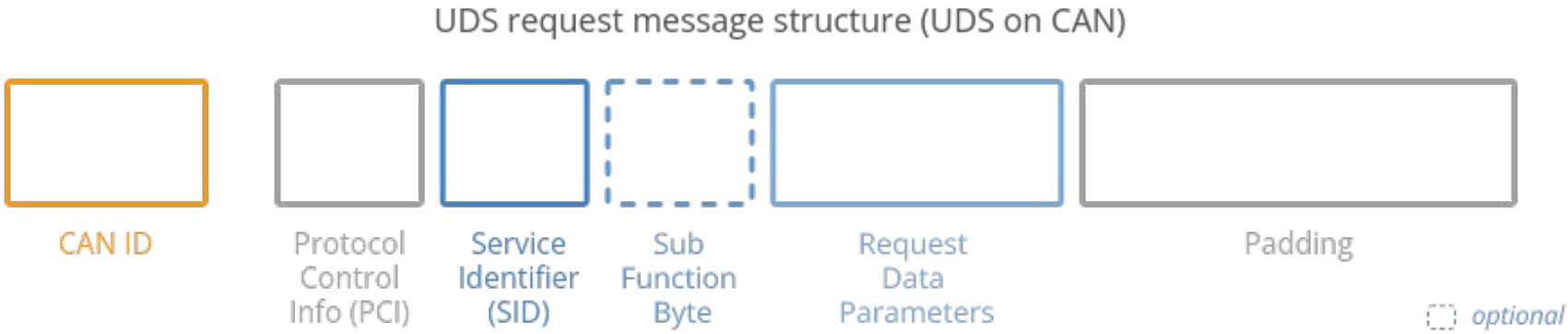
Communication in Client-Server fashion (server = ECU)

Possibilities:

- Read/clear trouble codes,
- Extract parameter data values,
- Initiate diagnostic sessions,
- Modify ECU behavior via resets, firmware flashing and settings modification

https://www.csselectronics.com/pages/uds-protocol-tutorial-unified-diagnostic-services#:~:text=Unified%20Diagnostic%20Services%20(UDS)%20is,2000%2C%20Ethernet%2C%20LIN

UDS Message Structure



- PCI: required for diagnostic, not UDS specific. 1-3 Bytes long with information related to the messages that don't fit a single CAN Bus packet
- SID: identity the specific UDS service
- Request Data Parameters: further information to provide for further configuration of a request

UDS Standards

It uses Application and Session layers: consecutive frames with data (segmentation and data stream)

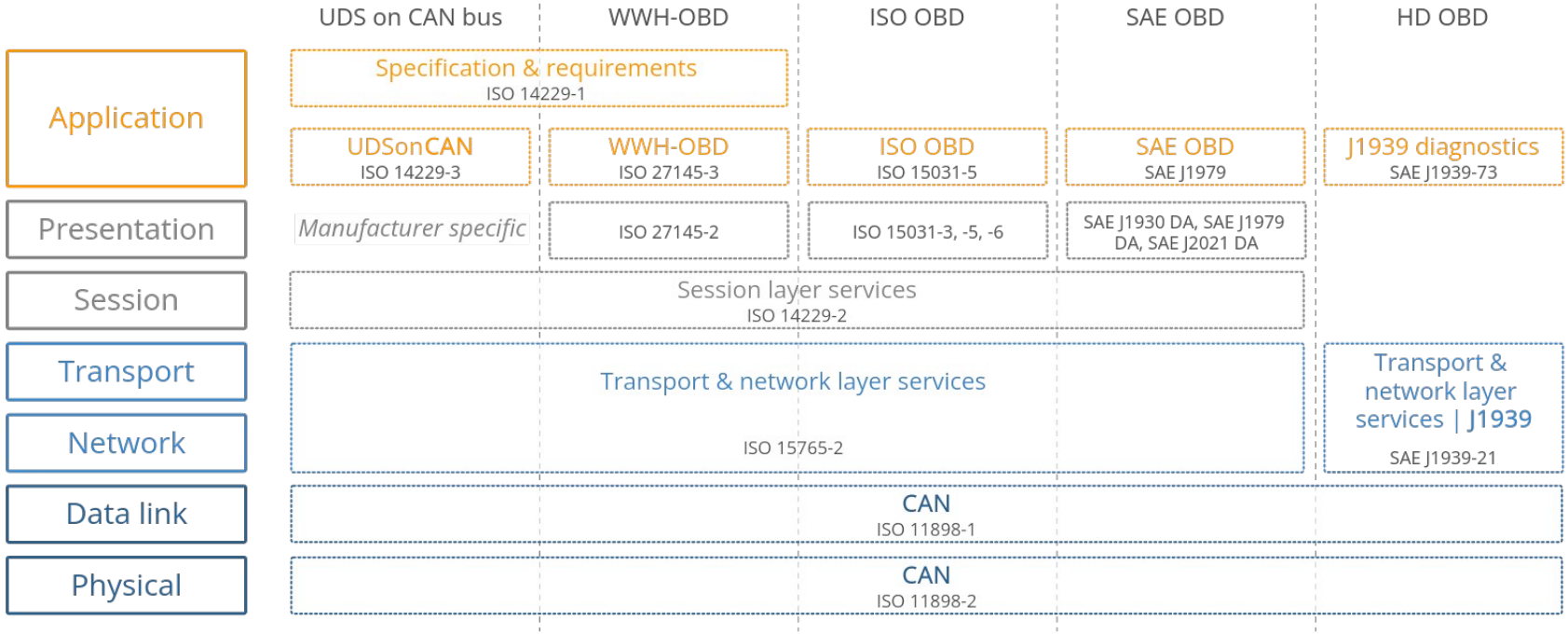
Transport and Network layer is ISO-TP for CAN

Data and Physical: CAN Bus

ISO 14229-1 (Application Layer)
ISO 14229-3 (Application Layer for CAN)
ISO 14229-2 (Session Layer)
ISO 15765-2 (Transport + Network Layer for CAN)
ISO 11898 (Physical + Data Link Layer for CAN)

UDS Standards

7 layer OSI model | **Diagnostic Protocols** (on CAN)



UDS with Python

- Library ***udsoncan***

  https://udsoncan.readthedocs.io/en/latest/

"This project is an implementation of the Unified Diagnostic Services (UDS) protocol defined by ISO-14229 written in Python 3. [...]"

"The goal of this project is to provide with a set of tool to interact with a UDS server by building/interpreting UDS payload and detecting malformed messages. All of this, with minimal effort and comprehensive code. It can be useful to develop a tester unit, debugging a server code, **searching for security flaws or just messing with your car.**"

Nice Features

- Bypass CAN Bus Gateways
- Access by OBD-II
- Most likely supported by every vehicle and OEMs
- ECU firmware and settings manipulation

Drawback → Security Feature (but possibility to bypass it or not implemented by manufacturers):

- Security Access Function (14229:1 - Section 10.4)
- Authentication (14229:1 - Section 10.6)
- Security Sub-Layer (14229:1 - Section 16, also Session Layer)

https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf

Experiments

Use of caring caribou integration in python3 scripts to automate the process and the analysis

```
$./cc.py -i can- uds discovery
```

```
Sending Diagnostic Session Control to 0x06f3
 Verifying potential response from 0x06f3
   Resending 0x6f3...  No response
   Resending 0x6f2...  No response
   Resending 0x6f1...  No response
   Resending 0x6f0...  Success
Found diagnostics server listening at 0x06f0, response at 0x0611
Sending Diagnostic Session Control to 0x07ff
```

Querying each ID for open services

```
./cc.py', '-i', f'{interface}', 'uds', 'services', str(src), str(dst)
```

Finally, try to get information through these services

Experiments

SPRITZ Security & Privacy Research Group

UDS service identifiers (SIDs)

	UDS SID (request)	UDS SID (response)	Service	Details
Diagnostic and Communications Management	0x10	0x50	Diagnostic Session Control	Control which UDS services are available
	0x11	0x51	ECU Reset	Reset the ECU ("hard reset", "key off", "soft reset")
	0x27	0x67	Security Access	Enable use of security-critical services via authentication
	0x28	0x68	Communication Control	Turn sending/receiving of messages on/off in the ECU
	0x29	0x69	Authentication	Enable more advanced authentication vs. 0x27 (PKI based exchange)
	0x3E	0x7E	Tester Present	Send a "heartbeat" periodically to remain in the current session
	0x83	0xC3	Access Timing Parameters	View/modify timing parameters used in client/server communication
	0x84	0xC4	Secured Data Transmission	Send encrypted data via ISO 15764 (Extended Data Link Security)
	0x85	0xC5	Control DTC Settings	Enable/disable detection of errors (e.g. used during diagnostics)
	0x86	0xC6	Response On Event	Request that an ECU processes a service request if an event happens
	0x87	0xC7	Link Control	Set the baud rate for diagnostic access
Data Transmission	0x22	0x62	Read Data By Identifier	Read data from targeted ECU - e.g. VIN, sensor data values etc.
	0x23	0x63	Read Memory By Address	Read data from physical memory (e.g. to understand software behavior)
	0x24	0x64	Read Scaling Data By Identifier	Read information about how to scale data identifiers
	0x2A	0x6A	Read Data By Identifier Periodic	Request ECU to broadcast sensor data at slow/medium/fast/stop rate
	0x2C	0x6C	Dynamically Define Data Identifier	Define data parameter for use in 0x22 or 0x2A dynamically
	0x2E	0x6E	Write Data By Identifier	Program specific variables determined by data parameters
	0x3D	0x7D	Write Memory By Address	Write information to the ECU's memory
DTCs	0x14	0x54	Clear Diagnostic Information	Delete stored DTCs
	0x19	0x59	Read DTC Information	Read stored DTCs, as well as related information
	0x2F	0x6F	Input Output Control By Identifier	Gain control over ECU analog/digital inputs/outputs
	0x31	0x71	Routine Control	Initiate/stop routines (e.g. self-testing, erasing of flash memory)
Upload/ Download	0x34	0x74	Request Download	Start request to add software/data to ECU (incl. location/size)
	0x35	0x75	Request Upload	Start request to read software/data from ECU (incl. location/size)
	0x36	0x76	Transfer Data	Perform actual transfer of data following use of 0x74/0x75
	0x37	0x77	Request Transfer Exit	Stop the transfer of data
	0x38	0x78	Request File Transfer	Perform a file download/upload to/from the ECU
		0x7F	Negative Response	Sent with a Negative Response Code when a request cannot be handled