Towards Automatically Reverse Engineering Vehicle Diagnostic Protocols (Appendix: Introduction of Diagnostic Protocols)

Anonymous Author(s)

1 Appendix

1.1 OBD-II Protocol

The OBD-II dongles plugged into the OBD port implement the OBD-II protocol so that users can use mobile apps to read data from vehicle [6,7]. Since many OBD-II dongles use ELM 327 micro-controller, they also support *AT commands* [3]. The AT commands start with the characters "AT". The dongles will interpret the following characters to change the configuration. For instance, "ATZ" means "reset all" and "ATRD" means "Read the stored Data".

Example. To get throttle position (the PID is 0x11), the OBD-II request message is "`01 11`". The corresponding formula to process the response message is: $\frac{100}{255} * DATA\_A$.

1.2 Services of KWP 2000 based on K-Line (ISO 14230-3) and UDS (ISO 14229)

We list the services of KWP 2000 based on K-Line (Tab. 1) and the services of UDS (Tab. 2). The services of KWP 2000 based on CAN (ISO 15765-3 [1]) are not listed since they are almost the same as Tab. 2 (UDS is derived from ISO 14230-3 and 15765-3).

Both KWP 2000 and UDS can group their services into six functional units. We focus on the "Data transmission functional unit" and the "InputOutput control functional unit" since attackers can directly use them to know the current vehicle status or control the vehicle components [2, 5]. Moreover, these two units both contain fields that specified by the vehicle manufactures.

Table 1: Services of KWP 2000 based on K-Line(ISO 14230-3)

Functional Unit	SID	Service Name
Diagnostic management functional unit	$10	StartDiagnosticSession
	$20	StopDiagnosticSession
	$27	SecurityAccess
	$3E	TesterPresent
	$11	EcuReset
	$1A	ReadEcuIdentification
Data transmission functional unit	$21	ReadDataByLocalIdentifier
	$22	ReadDataByCommonIdentifier
	$23	ReadMemoryByAddress
	$2C	DynamicallyDefineLocalIdentifier
	$3B	WriteDataByLocalIdentifier
	$2E	WriteDataByCommonIdentifier
	$3D	WriteMemoryByAddress
	$26	SetDataRates
Stored data transmission functional unit	$13	ReadDiagnosticTroubleCodes
	$18	ReadDiagnosticTroubleCodesByStatus
	$17	ReadStatusOfDiagnosticTroubleCodes
	$12	ReadFreezeFrameData
	$14	ClearDiagnosticInformation
InputOutput control functional unit	$30	InputOutputControlByLocalIdentifier
	$2F	InputOutputControlByCommonIdentifier
Remote activation of routine functional unit	$31	StartRoutineByLocalIdentifier
	$38	StartRoutineByAddress
	$32	StopRoutineByLocalIdentifier
	$39	StopRoutineByAddress
	$33	RequestRoutineResultsByLocalIdentifier
	$3A	RequestRoutineResultsByAddress
Upload download functional unit	$34	RequestDownload
	$35	RequestUpload
	$36	TransferData
	$37	RequestTransferExit

1.3 Diagnostic Frames Analysis: Selected Fields Extraction

After obtaining the payload of diagnostic message, we split the payload to multiple fields according to protocol formats. We extract the ESV and ECR contained in diagnostic messages. (*i*) When using the KWP 2000 protocol to read **ESV**, each positive response message contains one local id and a list of **ESV**. We can directly extract them because the lengths of the local id and **ESV** are fixed values. When using the UDS protocol to read **ESV**, one response message may contain n DID and n **ESV** ($n >= 1$) [4]. Because the length of each **ESV** is variable, we cannot directly extract the DIDs and **ESV** from the response message. To solve this problem, we first extract the DIDs from the request message to assist extracting **ESV** from the response message since the DIDs in the request and response messages are the same. In detail, after getting the request message "`{22} {DID #1: 2 bytes} ... {DID #m: 2 bytes}`", we extract the list of DIDs. When the payload of the corresponding response message "`{62} {DID #1: 2 bytes} {ESV #1: k bytes} ... {DID #m: 2 bytes} {ESV #m: o bytes}`" is obtained, we first locate the DIDs of the request message in the payload of response message. Then, we extract the field value between `DID #i` and `DID #(i+1)` and regard it as the **ESV** of `DID #i`.

(*ii*) When extracting **ECR** from the payload of request messages generated when controlling vehicle components, we extract DIDs and **ECR** according to the formats of request messages [4]. For each **ECR**, we also extract the IO control parameter (first byte) and control state (remaining bytes) from it to discover the semantic meaning of the control state field.

Table 2: Services of UDS (ISO 14229)

Functional Unit	SID	Service Name
Diagnostic and communication management functional unit	$10	DiagnosticSessionControl
	$11	ECUReset
	$27	SecurityAccess
	$28	CommunicationControl
	$29	Authentication
	$3E	TesterPresent
	$83	AccessTimingParameter
	$84	SecuredDataTransmission
	$85	ControlDTCSetting
	$86	ResponseOnEvent
	$87	LinkControl
Data transmission functional unit	$22	ReadDataByIdentifier
	$23	ReadMemoryByAddress
	$24	ReadScalingDataByIdentifier
	$2A	ReadDataByPeriodicIdentifier
	$2C	DynamicallyDefineDataIdentifier
	$2E	WriteDataByIdentifier
	$3D	WriteMemoryByAddress
Stored data transmission functional unit	$14	ClearDiagnosticInformation
	$19	ReadDTCInformation
InputOutput control functional unit	$2F	InputOutputControlByIdentifier
Remote activation of routine functional unit	$31	RoutineControl
Upload download functional unit	$34	RequestDownload
	$35	RequestUpload
	$36	TransferData
	$37	RequestTransferExit

Table 3: IO Control Parameter: Values and semantic meaning

Value	Semantic meaning	Value	Semantic meaning
0x00	Return control to ECU	0x01	Reset to default
0x02	Freeze current state	0x03	Short term adjustment

References

[1] Diagnostics on Controller Area Networks (CAN) - Part 3: Implementation of unified diagnostic services (UDS on CAN). http://read.pudn.com/downloads506/doc/2103567/ISO_15765-3.pdf, 2004.

[2] Zubie: This Car Safety Tool 'Could Have Given Hackers Control Of Your Vehicle'. https://shorturl.at/wAHI3, 2014.

[3] ELM327: OBD to RS232 Interpreter. https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf, 2016.

[4] UDS ISO 14229: Standardized CAN-based protocol for diagnostics. https://automotive.softing.com/en/standards/protocols/uds-iso-14229.html, 2019.

[5] C. Miller and C. Valasek. Adventures in automotive networks and control units. *Def Con*, 2013.

[6] H. Wen, Q. Chen, and Z. Lin. Plug-n-pwned: Comprehensive vulnerability analysis of obd-ii dongles as a new over-the-air attack surface in automotive iot. In *Proc. USENIX Security*, 2020.

[7] H. Wen, Q. Zhao, Q. Chen, and Z. Lin. Automated cross-platform reverse engineering of can bus commands from mobile apps. In *Proc. NDSS*, 2020.