Automated Reverse Engineering and Attack for CAN using OBD-II

Tae Un Kang, Hyun Min Song, Seonghoon Jeong, Huy Kang Kim
Graduate School of Information Security
Korea University
Seoul, Republic of Korea
{ktw1332, signos, seonghoon, cenda}@korea.ac.kr

***Abstract*—Controller area network (CAN) is one of the most popular in-vehicle networks. CAN allows electronic control units (ECUs) to communicate with each other. ECUs control various function of vehicle systems such as engine and transmission control. Therefore, CAN and ECUs are the high priority targets by hackers. If the CAN and the connected components are attacked, the vehicle may cause serious malfunction and fatal accidents. However, it is hard to find out the exact CAN messages to send and control the vehicle as intended by hackers. Likewise, vehicle security researchers have the same problem to find out the exact meaning of CAN messages to detect sophisticated attacks as well as attackers. It is relatively easy to detect the simple pattern of attacks such as denial of service (DoS) attack. However, CAN specification information is private information of car OEMs, to reveal the exact meaning of CAN messages, we need to analyze the messages by reverse engineering techniques, which is time-consuming and laborious tasks. To solve this problem, we developed the Automated CAN Analyzer (ACA). The ACA has automated reverse engineering functions which can help to analyze the relationship between the response data from a diagnostic query of on-board diagnostics II (OBD-II) and the related CAN traffic data. Furthermore, it supports the automated attack function that can inject fake messages into CAN bus based on pre-analyzed CAN message information. Researchers can easily confirm whether the reverse engineering results are correctly working or not through the provided automated attack function. As a result, the ACA could lower the barriers to entry to in-vehicle network research. To evaluate the ACA, we applied our approach to two real vehicles, Hyundai YF Sonata (2010 model) and KIA Soul (2014 model). In this paper, we can find out the meaning of CAN messages on both vehicles with the help of the ACA. Additionally, since modern vehicles are all equipped with OBD-II, our approach can be applied to most vehicle widely.**

***Index Terms*—CAN, Reverse Engineering, Vehicle Security, In-Vehicle Network Analysis**

I. Introduction

Modern vehicles have multiple devices to control their various functions such as engine and transmission. These devices are called electronic control unit (ECU), and they communicate with each other over the in-vehicle network. The controller area network (CAN) is a representative in-vehicle network, which is widely used in modern vehicles. As it is already known, the CAN is vulnerable to data pollution by manipulated messages injection. Past vehicles were less threatened because they did not have any communication functions to the outside. Thus, physical access to the vehicle was the only way to inject malicious messages into the CAN bus. However, the connectivity to the outside of modern vehicles enables attackers to perform remote attacks through wireless communication applications such as Over-The-Air (OTA) and infotainment systems [1], [2]. In 2015, Charlie Miller and Chris Valasek presented that how they hacked the Jeep Cherokee's vehicle remotely via telematics and Bluetooth [3]. They successfully controlled the car's brakes, steering wheels, and other critical systems by attacking CAN via wireless communication. To solve such problem, manufacturers try to replace CAN with secure next-generation in-vehicle networks. However, it is infeasible to replace all existing CAN-based in-vehicle network system in a short time. This limitation leads researchers to develop security countermeasure for the CAN-based in-vehicle network system. For the security research, interpretation of the meaning of CAN messages must be preceded in order to classify sophisticated attack traffics from normal traffics. Since vehicle manufacturers keep the CAN specifications of their products confidential, the exact meaning of CAN messages must be found through reverse engineering techniques. However, reverse engineering of the CAN messages is a time-consuming task; and highly dependent on the analyst's knowledge and experience. For effective and efficient reverse engineering of the CAN message, we developed an automated reverse engineering tool called Automated CAN Analyzer (ACA). The ACA has automated reverse engineering functions that can analyze the relationship between the response data of diagnostic query of on-board diagnostics II (OBD-II) and the related CAN traffic data. In this paper, our contributions can be summarized as follows:

- We release the ACA's source code and executable file on GitHub for free (https://github.com/comma1/ACA). Since most CAN analysis tools only provide statistical data of the captured CAN traffic, it is difficult for inexperienced researchers to find out the exact meaning of the CAN messages [4]. As the ACA provides the automated CAN analysis functionality, researchers can figure out the vehicle's CAN specification more easily. We expect that the ACA could lower the barriers and encourage further

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. R7117-16-0161, Anomaly Detection Framework for Autonomous Vehicles)

978-1-5386-6358-5/18/$31.00 ©2018 IEEE

CAN security research because it does not require any prior knowledge of the vehicle to be analyzed regardless of the vehicle's vendor and model. Also, the ACA is applicable to all vehicles adopting OBD-II because OBD-II is equipped with all vehicles of today.

- The ACA can be utilized as a penetration testing tool with its automated attack function. Also, security researchers can use the ACA to evaluate the performance of intrusion detection system (IDS) for vehicles [5], [6].
- Finally, we release our experiment dataset and analysis information for vehicles. Vehicle researchers can use this information for studying the vehicle attack and defense for free.

II. Related works

There are a few research to analyze CAN messages by reverse engineering. In this section, we introduce the previous works that are related to reverse engineering of CAN messages, and we describe the limitations of each work.

Flores *et al.* [7] proposed a system that analyses the CAN messages. The system provides several functions ('CAN frame statistic' and 'Filter') for the analysis of CAN messages. 'CAN frame statistic' calculates the mean, standard deviation and variance of the period of all unique CAN IDs. 'Filter' can filter the recorded CAN messages by given parameters and value. The proposed system provides overall information such as basic statistical information of CAN messages. However, it is still difficult to know the exact meaning of the CAN messages.

CSS Electronics [4] developed 'CLX000 CAN sniffer' for reverse engineering of CAN messages. For each CAN ID, it provides basic statistical information about CAN messages (e.g., count, frame no, time, period). When a particular value in a data field changes, the tool displays the related data byte in specific color. The authors describe that it is useful when comparing data field of the CAN messages versus physical events at that time (e.g., turning on the car wipers). However, about 2,000 of CAN messages are come from the CAN bus per second, it is not suitable to compare data field of the CAN messages versus physical events manually.

Jason [8] presented a reverse engineering methodology for CAN messages based on CAN ID's period. In this work, the author found each CAN ID has a unique period. For example, CAN IDs which are responsible for updating the 'engine speed display gauges' is updated more frequently than other devices. It is quite useful characteristics but it seems to require the analyst's experience and analysis skill. According to the CAN messages analysis information of vehicles (Toyota Camry, Hyundai Sonata and KIA Soul) analyzed by Jeremy [9] and our work, the period value of the CAN ID associated with the 'engine speed display gauges' was not low enough to distinguish CAN IDs. This methodology could be applicable only to specific vehicles.

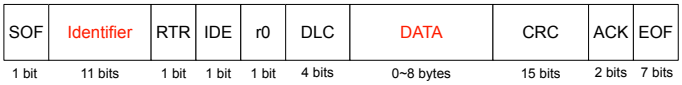

Fig. 1. Structure of CAN frame

Time	CAN ID	DATA
11.565163	2B0	D5 FF 00 07 B4
11.565219	430	00 00 00 00 00 00 00 00
11.565277	4B1	01 01 00 00 00 00 00 02
11.565339	1F1	00 00 00 00 00 00 00 00
11.565398	153	00 00 00 FF 00 FF 00 00
11.565455	002	00 00 00 00 00 08 0E F6
11.569169	0A0	70 7E 88 09 00 26 02 00
11.569288	0A1	80 83 00 00 2A 00 00 00
11.569348	130	F9 7F 00 FF 06 80 0E 49
11.569407	131	DB 7F 00 00 54 7F 0E 00
11.569471	140	00 00 00 00 22 08 2E 42
11.569542	2B0	D5 FF 00 07 A5

Fig. 2. 14 CAN messages extracted from YF Sonata

III. Background

A. CAN bus

CAN bus is currently the most widely adopted network standard among in-vehicle networks. Also, it is a protocol used in the OBD-II vehicle diagnostics. The characteristics of CAN are as follows.

- Priority: CAN ID is used to identify the type of data and the sending node. It is also used to set the priority of the message.
- Broadcast: CAN is basically broadcast type of communication. There is no way to send a message only to a specific node. All nodes can listen all traffic from any nodes in the bus. Fig. 1 shows various fields in a CAN frame. In this paper, we focus on analyzing two fields (Identifier and DATA) to find out the exact meaning of CAN messages.

Fig. 2 shows CAN messages extracted from YF Sonata. The CAN ID (Identifier) field determines the priority of the message delivery when two or more nodes are sending messages at the same time [10]. The DATA field contains zero to eight bytes of data; and it means detailed information about the ID. For example, the data field of CAN ID which is associated with the engine contains information such as 'engine speed' and 'water & oil temperature'.

B. OBD-II

On-Board Diagnostics (OBD) is a term referring to a vehicle's self-diagnostic and reporting capability. By the aid of OBD systems, the vehicle owner or repair technician can retrieve the status information of various vehicle subsystems. OBD-I refers to the first generation of OBD systems which were developed throughout the 1980s, and OBD-II is an improved version over OBD-I in both capability and standardization. OBD-II standard has been mandatory for all vehicles in the U.S. since 1996 [11]. Vehicles currently follow the

OBD-II standard all over the world. OBD-II can monitor parts of the chassis, body and accessory devices, as well as the diagnostic control network of the car. Status of vehicle subsystem is expressed by a number called PID (Parameter Identifier) used to identify the parameter. For example, the standard stipulates that the engine RPM has a PID 0C. Since not all vehicles support all PIDs, it is important to check the supported PIDs by the vehicle.

IV. Automated Reverse Engineering of CAN

The overall process of automated reverse engineering can be summarized as follows:

- To find the supported OBD-II PIDs of the vehicle, we input and request PIDs appeared in Table I as a PID query.
- With the supported OBD-II PIDs obtained in the previous process, we analyze CAN messages to find CAN IDs and data fields that are relevant to each PID.
- To verify the found CAN information is correct, the ACA generates fake messages based on the analyzed CAN information and injects them into the CAN bus.

We observe the vehicle's response to fake messages to verify whether the analyzed CAN information is correct or not. The work of confirming the response of the vehicle must be manually validated.

A. Finding the supported OBD-II PIDs

To obtain the effective OBD-II PIDs of the vehicle, we input PID values appears in Table I one by one; and we analyze the response to check the availability. We can find total 32 PIDs appeared in Table I are available. By sending 7 PIDs in total, we can verify the 224 PIDs are supported or not.

We input the PID in the data field (offset 3) and request an OBD-II query as shown in Fig. 3. A request for PID

TABLE I
SUPPORTED PIDS LIST

PID	Data bytes returned	Description
00	4	PIDs supported [01 - 20]
20	4	PIDs supported [21 - 40]
40	4	PIDs supported [41 - 60]
60	4	PIDs supported [61 - 80]
80	4	PIDs supported [81 - A0]
A0	4	PIDs supported [A1 - C0]
C0	4	PIDs supported [C1 - E0]

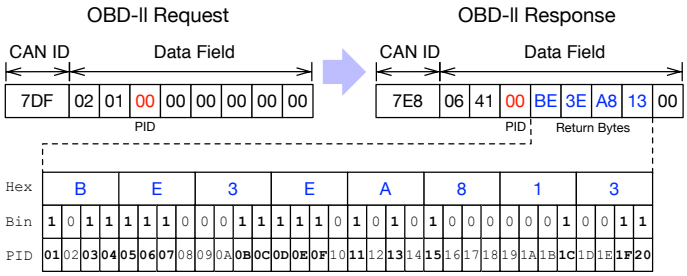

Fig. 3. Interpretation method for OBD-II response's return bytes

No	PID(Dec)	PID(Hex)	Description
1	01	01	Monitor status since DTCs cleared. (Includes malf
2	03	03	Fuel system status
3	04	04	Calculated engine load
4	05	05	Engine coolant temperature
5	06	06	Short term fuel trim-Bank 1
6	07	07	Long term fuel trim-Bank 1
7	11	0B	Intake manifold absolute pressure
8	12	0C	Engine RPM
9	13	0D	Vehicle speed
10	14	0E	Timing advance
11	15	0F	Intake air temperature

Fig. 4. Supported OBD-II PIDs list

returns 4 bytes of data. If return bytes of OBD-II response are BE3EA813 in Fig. 3, it can be interpreted as follows. By representing BE3EA813 value to binary for each digit, we can get the binary value of 32 digits. Each bit from most significant bit (MSB) to least significant bit (LSB) represents a total of 32 PIDs and gives information about if it is supported. We can interpret that supported OBD-II PIDs are 01, 03, . . . , and 20 through PID 00 query as shown in Fig. 3. Each PID of the supported OBD-II PIDs represents vehicle status (e.g., PID 0D: vehicle speed) as shown in Fig. 4. As a result, we can find the supported OBD-II PIDs of YF Sonata and Soul by requesting PIDs (00, 20, 40, 60, 80, A0, C0) as shown in Table I.

B. CAN data analysis by using OBD-II

We request an OBD-II query with the supported PIDs of YF Sonata and Soul. Then, we analyze the relationship between the response data from a query of OBD-II and the CAN traffic data as shown in Fig. 5. We describe two methods of analyzing CAN messages using OBD-II.

The first method is to extract the case where the return bytes in the OBD-II response match some bytes in the data field of CAN messages. For example, if return bytes of PID 0A (fuel pressure) are always mapped to the data field (offset 3-4) of CAN ID 2A0, we can suppose that the data field (offset

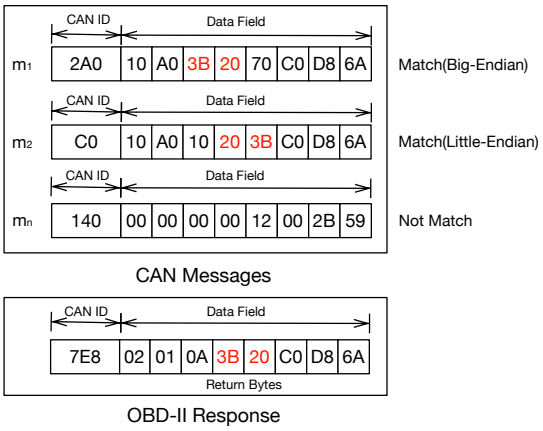

Fig. 5. Finding CAN messages that match the return bytes of the OBD-II response considering byte order

Algorithm 1: Finding a CAN message that matches the return bytes of the OBD-II response in forward and reverse order

Input: CAN Messages(DataField), OBD-II Response(ReturnBytes)
Output: MatchingCANMsg

while *CAN Msg in CAN Messages* **do**
  for $i = 0; i \leq 8 - ReturnBytes.Length; i++$ **do**
    $Matching = 1;$
    for $j = i; j \leq ReturnBytes.Length; j++$ **do**
      if $DataField[j] \neq ReturnBytes[j-i]$ **then**
        $Matching = 0;$
        $break;$
      end
    end
    if $matching = 1$ **then**
      $MatchingCANMsg.append(CANMsg);$
    end
    else
      $Matching = 1;$
      for $j = ReturnBytes.Length; j \geq 0; j++$ **do**
        if $DataField[j] \neq ReturnBytes[j-i]$ **then**
          $Matching = 0;$
          $break;$
        end
      end
      if $matching = 1$ **then**
        $MatchingCANMsg.append(CANMsg);$
      end
    end
  end
end
return $MatchingCANMsg;$

3-4) of the CAN ID represents 'fuel pressure' as shown in Fig. 5. Since the return bytes of the specific PIDs (e.g., PID 0C: engine RPM) are little-endian form, we extract the m_1 and m_2 considering the byte order as shown in Fig. 5 [12].

The second method is to reduce the CAN ID candidates that are mapped to the data field of the OBD-II through the search within results. In Fig. 6, we find the CAN messages associated with the return bytes (20 40) of OBD-II response are m_{n-2} and m_{n-1} in step 1. We request the OBD-II query again and then check the data field of m_{n-2}, m_{n-1} is mapped to the OBD-II response. In step 2, we exclude m_{n-2} because it is not related to the return bytes (2A 3B) of OBD-II response. In step 3, only m_{n-1} is left; and we can find out the data field (offset 3-4) of CAN ID is associated with PID 0A. Because return bytes of specific PIDs are changed along with

Algorithm 2: Search-in-results

Input: CAN Messages(DataField), OBD-II Response(ReturnBytes), count=0
Output: MatchingCANMsg

while *CAN Message in CAN Messages* **do**
  if *ReturnBytes in DataField* **then**
    MatchingCANMsg.append(CAN Message);
  end
end
if $count > 3$ **then**
  return $MatchingCANMsg;$
end
else
  Search-in-results(CAN Messages, ReturnBytes)
end
$count = count + 1;$

the driving condition, we analyzed the CAN messages under various driving environment. For example, PID 0D (vehicle speed)'s return bytes are always zero in the stationary state, the CAN ID candidates do not decrease with the second method. We describe algorithm of above two methods in Algorithm 1 and Algorithm 2.

C. CAN messages injection to real vehicles

To verify the analyzed CAN information is correct, we generate fake CAN messages based on the pre-analyzed CAN information. If we know the data field (offset 4-5) of CAN ID 0x360, which is supposed to be related with the PID 0C (engine RPM), we generate fake CAN messages by changing the byte in the analyzed field from 0x00 to 0xFF. Then we inject them into the vehicle. If the RPM gauge of the vehicle's dashboard is changed, we check that the analyzed CAN information is correct.

V. Experiment Results

A. Setting of experiment environment

We equipped Arduino UNO with CAN Bus Shield as shown in Fig. 7 so that Arduino UNO can communicate through CAN [13]. Then, we connect OBD-II port cable to DB9 port as shown in Fig. 7, and USB cable to USB 2.0 port of a laptop. We can communicate with the CAN bus through Arduino UNO as shown in Fig. 8. Laptop's processor and memory are i7-5500U CPU@2.40Ghz and 8GB, respectively. The ACA program uses the processor and memory about 20% and 0.1%. We conducted experiments on B-CAN (Body CAN) and C-CAN (Chassis CAN) closely related to vehicle safety through OBD-II cable. Then we analyze the meaning of CAN messages on both B-CAN and C-CAN networks. To avoid any possible accident such as sudden unintended acceleration (SUA), we experimented in vacant space safely.

B. Experiment results on real vehicles

In Table II, we summarize the analysis results of two cars. Each column means 'the total number of unique CAN IDs',

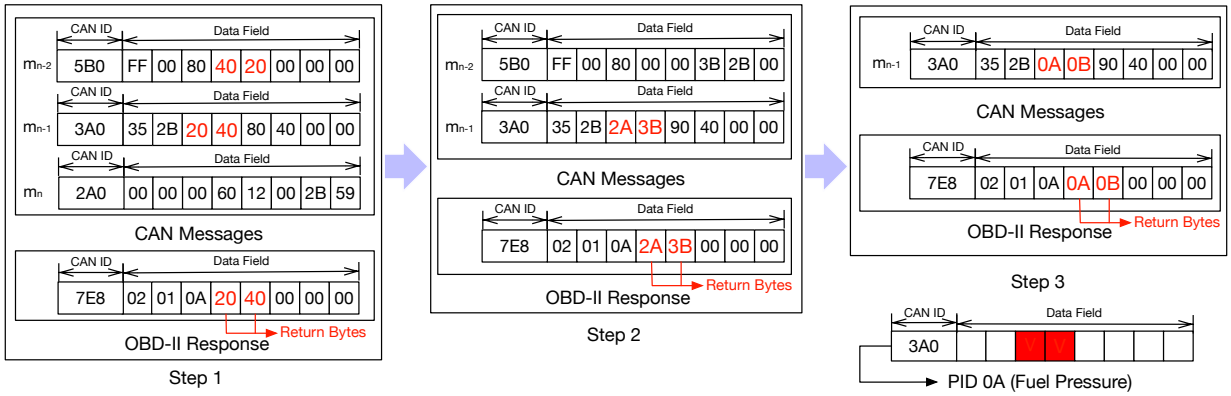

Fig. 6. Finding related CAN messages with OBD-II response

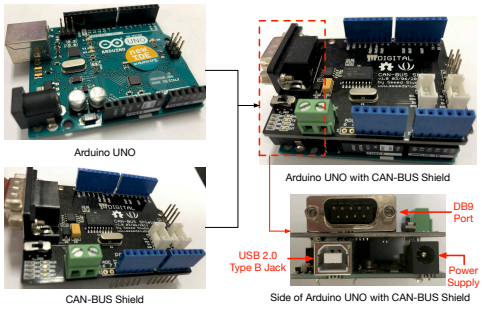

Fig. 7. Arduino UNO with CAN-BUS Shield

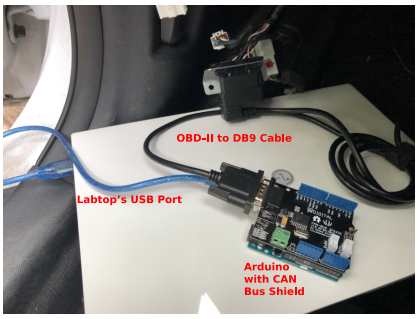

Fig. 8. Setting of experiment environment using Arduino and OBD-II cable

TABLE II
ANALYSIS RESULT FOR CAN ID AND OBD-II PID IN TWO VEHICLES

	Unique CAN ID	Analyzed CAN ID	Supported OBD-II PID	Analyzed OBD-II PID
YF Sonata	27	10	36	15
Soul	45	13	34	15

TABLE III
SUPPORTED OBD-II PIDS LIST FOR YF SONATA AND SOUL

PID	PID range	Supported PIDs of YF Sonata	Supported PIDs of Soul
00	01 - 20	01, 03, 04, 05, 06, 07, 0B, 0C, 0D, 0E, 0F, 11, 13, 15, 20	01, 03, 04, 05, 06, 07, 0B, 0C, 0D, 0E, 0F, 11, 13, 15, **17**, 20
20	21 - 40	22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 31, 32, 34, **35**, **36**, **3D**, 3E	22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 31, 32, 34, 3E
40	41 - 60	X	X
60	61 - 80	X	X
80	81 - A0	X	X
A0	A1 - C0	X	X
C0	C1 - E0	X	X

'the number of CAN IDs analyzed with OBD-II PIDs', 'the OBD-II PIDs supported by the vehicle' and 'the number of analyzed OBD-II PIDs in the third column'. The Soul has 18 unique CAN IDs more than the YF Sonata has. We speculate the 18 CAN IDs are related with cruise control because YF Sonata does not have the function. The rate of analyzed CAN ID from the unique CAN ID of the YF Sonata and the Soul are approximately 37% and 28%. Although the rate is low, we can find out the exact meaning with less effort and time consumption.

As shown in Table III, YF Sonata and Soul support 36 OBD-II PIDs and 34 OBD-II PIDs respectively. Supported OBD-II PIDs of both vehicles are mostly the same. YF Sonata supports 0x35 (Oxygen Sensor 2), 0x36 (Oxygen Sensor 3) and 0x3D (Catalyst Temperature: Bank 2, Sensor 1) PIDs where Soul supports 0x17 (Oxygen Sensor 4) PID.

Table IV shows the information of CAN messages and the response of vehicle when we injected fake CAN messages into the vehicle. By comparing the description of PID with the status change of vehicle, we verify whether the analyzed CAN information is correct or not. For example, the description of PID 0C (engine RPM) matches the response of the vehicle (RPM gauge goes up and down), we confirm the PID 0C is correctly analyzed.

We summarized the findings in Table IV.

The definitions of several terms appeared in Table IV are as follows.

- Vehicle dynamics control (VDC): This detects the slip of road surface. It also controls the pressure of the wheel brake and the engine output [14].
- Engine coolant temperature sensor (ECTS): It is the sensor by measuring the engine coolant temperature. The

TABLE IV
CAN INJECTION RESULT FOR TWO VEHICLES (YF SONATA AND SOUL)

PID	Description	Vehicle	CAN ID	Data Field								Status change due to CAN injection
				1	2	3	4	5	6	7	8	
03	Fuel system status	YF Sonata	140				✓	✓				VDC warning light turns on
		Soul	2A0	✓	✓							Distance to empty value is changed
04	Calculated engine load	YF Sonata	545							✓		Error code (FUSE) appears
05	Engine coolant temperature	YF Sonata	329			✓						ECTS gauge goes up and down
06	Short term fuel trim-Bank 1	YF Sonata	130						✓			VDC warning light turns on
		Soul	153							✓		Direction indicator warning light and VDC warning light turn on
0C	engine RPM	YF Sonata	316			✓	✓					RPM gauge goes up and down
			0A0			✓	✓					RPM gauge goes up and down
		Soul	316			✓	✓					RPM gauge goes up and down, a beep sounds
0D	Vehicle Speed	YF Sonata	316							✓		Immobilizer warning light and VDC warning light turn on, RPM gauge goes up, car door locked
			440			✓						Immobilizer warning light turns on, speedometer gauge goes up
		Soul	165	✓								Airbag warning light turns on, distance to empty value is changed, a beep sounds
			316				✓					RPM gauge goes up and down, a beep sounds
			80				✓					TPMS warning light and airbag warning light are blinking
			220			✓						VDC warning light and airbag warning light turn on
			43F							✓		The vehicle moves slightly, the gear status is changing very quickly, RPM gauge goes up, airbag warning light turns on
0E	Timing advance	YF Sonata	440						✓			Speedometer gauge goes up, car door locked
		Soul	80								✓	Distance to empty value is changed, a beep sounds, brake warning light and seat belt reminder indicator warning light turn off, airbag warning light turns on, error code (FUSE) appears
			4F2								✓	Airbag warning light, seat belt reminder indicator warning light and TPMS warning light turn on, distance to empty value is changed
0F	Intake air temperature	YF Sonata	002								✓	VDC warning light turns on
		Soul	43F						✓			The vehicle moves slightly, the gear status changes very quickly, error code (FUSE and PAUSE) appears, brake warning light turns on, engine oil pressure warning light and airbag warning light turn on
11	Throttle position	YF Sonata	316						✓			RPM Gauge goes up and down, strange noise is generated
		Soul	260								✓	Cruise control warning light and immobilizer warning light turn on, error code (SET) appears
13	Oxygen sensors present (in 2 banks)	YF Sonata	002							✓		VDC warning light and ABS warning light turn on
		Soul	220				✓					VDC warning light turns on
2E	Commanded evaporative purge	YF Sonata	140							✓		VDC warning light and ABS warning light turn on
		Soul	164							✓		Steering wheel is slightly moved, TPMS warning light, airbag warning light and VDC warning light turn on
			165							✓		Distance to empty value is changed
2F	Fuel Tank Level Input	YF Sonata	140							✓		VDC warning light and ABS warning light turn on
		Soul	80								✓	Distance to empty value is changed

engine controls the amount of fuel injection and ignition timing according to the temperature of the cooling water [15].

- Immobilizer warning lights: This warning light is on when the key is not recognized [16].
- Anti-skid braking system (ABS): It allows keeping wheels from locking-up and helps drivers to maintain steering control. If the ABS determines that wheels are in danger of locking-up, it is capable of rapidly activating and deactivating individual brake calipers or wheel cylinders [17].
- Tire pressure monitoring system (TPMS): It monitors tire air pressure and alerts the driver when it falls dangerously low [18].
- Cruise control: It is a system that automatically controls the speed of a vehicle. The system is a mechanism that takes over the throttle of the car to maintain a steady speed as set by the driver [19].

If attackers have enough knowledge about CAN messages and their meaning, they can send well-crafted attacks to control the car. That is, our analysis can be used for the penetration test for vehicles. For example, we can change the warning displays in the dashboard by CAN injection as shown in Fig. 9. It shows the display of 'steering wheel moved', 'vehicle door locked' and 'the turn signal' are flashing regardless of the car's real status. We summarize the status change by CAN injection in Table IV. In the case of PID 0C, both vehicles use the same CAN ID and the analyzed data fields are located at the same offset. It may be a simple coincidence, if the meaning of the CAN messages of the two different vehicles is the same, we can guess the CAN ID and data fields associated with the specific ID (e.g. engine RPM) could be the same. Also, some IDs are related to multiple functions because ECUs are not responding to a single CAN message but a series of CAN IDs. For example, CAN ID 80 and 43F of the Soul can be used for various functions. Although we changed only one byte of the data field when attacking the vehicle, more than five types of status variables of the vehicle changed as a result.

Fig. 9. Dashboard of Soul vehicle

VI. Conclusion

In this paper, we propose an automated reverse engineering method for CAN messages by analyzing the relationship between data field of CAN message and the return bytes of OBD-II response. To our best knowledge, it is the first automatic reverse engineering method for CAN messages. As a proof of concept application, we release the ACA that can analyze CAN messages. With the ACA, vehicle security researchers can easily find out the exact meaning of CAN messages. The ACA can save much time to investigate the CAN traffic without any prior knowledge. Also, with our proposed method, we can reveal the meaning of CAN IDs of the YF Sonata and the Soul up to 37% and 28%, respectively. Finally, we believe that this method will be useful for penetration test to increase the vehicle's safety level.

References

[1] P. Newswire, "Global automotive cybersecurity market 2017-2021 with argus cyber security, harman international, karamba security & symantec dominating," 2017.

[2] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[4] C. Electronics, "Reverse engineering can bus messages with wireshark," 2010.

[5] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 63–68.

[6] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame."

[7] J. Flores-Arias, M. Ortiz-Lopez, F. Quiles-Latorre, V. Pallares, and A. Chen, "Complete hardware and software bench for the can bus," in *Consumer Electronics (ICCE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 211–212.

[8] J. Staggs, "How to hack your mini cooper: reverse engineering can messages on passenger automobiles," *Institute for Information Security*, 2013.

[9] J. Daily, "Interpreting the can data for a 2010 toyota camry," 2010.

[10] P. Apse-Apsitis, "Can bus elements in robotic snake-like movement device control," in *10th International Symposium "Topical Problems in the Field of Electrical and Power Engineering, Pärnu, Estonia*, 2011, pp. 42–45.

[11] T. Miller, "Which obd2 protocol is supported by my vehicle?" 2018. [Online]. Available: "https://www.obdadvisor.com/obd2-protocol-supported-vehicle"

[12] T. Hermans, J. Denil, P. De Meulenaere, and J. Anthonis, "Decoding of data on a can powertrain network," in *16th Annual Symposium on Communication and Vehicular Technology in the Benelux (1)*, vol. 6. Citeseer, 2009.

[13] B. Patil, H. Amrite, K. Gaikwad, J. Dighe, and S. Hirlekar, "Smart car monitoring system using arduino," 2018.

[14] J. Wang and R. G. Longoria, "Coordinated and reconfigurable vehicle dynamics control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 3, pp. 723–732, 2009.

[15] N. El-Awar, D. Geer, T. Krellner, and P. Straub, "Automotive temperature sensing," *Keystone Thermometrics Corporation Application Notes*, 1999.

[16] L. Baolin, W. Tongmin, F. Shuai, and F. Jiangpeng, "Study and design of gateway engine immobilizer based on can-bus," in *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*. IEEE, 2011, pp. 751–755.

[17] J. R. Layne, K. M. Passino, and S. Yurkovich, "Fuzzy learning control for antiskid braking systems," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 2, pp. 122–129, 1993.

[18] R. M. Ishtiaq Roufa, H. Mustafaa, S. O. Travis Taylora, W. Xua, M. Gruteserb, W. Trappeb, and I. Seskarb, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium, Washington DC*, 2010, pp. 11–13.

[19] P. A. Ioannou and C.-C. Chien, "Autonomous intelligent cruise control," *IEEE Transactions on Vehicular technology*, vol. 42, no. 4, pp. 657–672, 1993.