Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.1, Feb. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.1.167

CAN 트레이스 분석을 통한 데이터 필드 형식 추론 방법 연구*

지 청 민,[1†] 김 지 민,[1] 홍 만 표[2‡]
[1]아주대학교 컴퓨터공학과, [2]아주대학교 사이버보안학과

Method for Inferring Format Information of Data Field from CAN Trace*

Cheongmin Ji,[1†] Jimin Kim,[1] Manpyo Hong[2‡]
[1]Dept. of Computer Engineering, Ajou University,
[2]Dept. of Cyber Security, Ajou University

요 약

최근 차량에 대한 공격 사례가 늘어남에 따라 CAN 기반의 보안 기술에 대한 연구가 활발히 진행되고 있다. 그러나 CAN의 상위 계층 프로토콜은 차량 제조사 및 모델 별로 상이하므로 이상 탐지 기술 또는 ECU 대상의 취약점 탐지를 위한 연구에는 큰 어려움이 따른다. 본 논문에서는 이러한 문제를 완화하기 위하여 CAN 트레이스의 분석을 통해 데이터 필드 영역의 세부 구조를 추론하는 방법을 제안한다. 기존 인터넷 환경에서는 이미 프로토콜 역공학을 위한 연구가 다수 진행되었으나, CAN 버스는 기존의 프로토콜 역공학 기술을 그대로 적용하기 어려운 구조를 지닌다. 본 논문에서는 CAN 프레임 내 데이터의 특성을 이용한 낮은 계산 비용의 필드 구분 방법 및 기존의 CAN 데이터필드 내 필드 분류 방법을 이용한 새로운 추론 방법을 제안한다. 본 논문에서 제안하는 방식은 실제 차량의 CAN 트레이스 및 시뮬레이션으로 생성된 CAN 트레이스를 대상으로 검증되며, 기존 방식 대비 더 낮은 계산 비용으로 더 높은 정확도의 필드 구조 추론 결과를 보인다.

ABSTRACT

As the number of attacks on vehicles has increased, studies on CAN-based security technologies are actively being carried out. However, since the upper layer protocol of CAN differs for each vehicle manufacturer and model, there is a great difficulty in researches such as developing anomaly detection for CAN or finding vulnerabilities of ECUs. In this paper, we propose a method to infer the detailed structure of the data field of CAN frame by analyzing CAN trace to mitigate this problem. In the existing Internet environment, many researches for reverse engineering proprietary protocols have already been carried out. However, CAN bus has a structure difficult to apply the existing protocol reverse engineering technology as it is. In this paper, we propose new field classification methods with low computation-cost based on the characteristics of data in CAN frame and existing field classification method. The proposed methods are verified through implementation that analyze CAN traces generated by simulations of CAN communication and actual vehicles. They show higher accuracy of field classification with lower computational cost compared to the existing method.

Keywords: vehicle security, CAN, protocol reverse engineering

Received(11. 22. 2017), Modified(01. 11. 2018), Accepted01. 11. 2018)

* 본 연구는 국토교통부 및 국토교통과학기술진흥원의 연구비 지원(17TLRP-B117133-02)으로 수행되었습니다.

† 주저자, ooops222@gmail.com
‡ 교신저자, mphong@ajou.ac.kr(Corresponding author)

I. 서 론

차량 내부에는 여러 ECU(Electronic Control Unit)가 존재하며 각 ECU는 버스 형태의 차량 내부 네트워크를 통해 센서 정보, 제어 정보 등의 메시지를 서로 주고받는다. 현재 대부분의 차량 내부 통신은 1986년 Bosch社가 개발한 CAN(Controller Area Network)을 통신 규격으로 사용한다. 그러나 CAN은 프로토콜 설계 당시 보안 기능이 전혀 고려되지 않아 최근의 다양한 차량 대상 공격사례[1-6]의 근본적인 원인이 되고 있다. 이러한 문제점을 극복하기 위해서는 CAN-FD, FlexRay, MOST, ethernet 등으로 CAN을 대체해야 할 필요가 있다. 하지만 현재까지 개발된 CAN 기반기술을 단번에 다른 기술로 전환하거나 이미 도로 상에 존재하는 이전 모델의 차량들을 단 시간 내에 모두 폐기하기는 힘들다. 따라서 CAN 기반 차량내부통신 환경 하에서의 보안에 대한 연구 또한 병행되어야 할 필요가 있다[8].

CAN 기반 차량내부통신 보안에는 한 가지 어려움이 존재한다. CAN 자체는 데이터 링크 계층 프로토콜로서 명세가 모두 알려져 있으나 CAN 프레임 내 데이터 필드 영역은 상위 계층으로서 차량 제조사 및 모델 별로 메시지 형식이 서로 상이하다. 각 제조사 및 차량 모델 별 메시지 형식은 쉽게 공개되지 않으므로 해당 정보 없이는 CAN 기반의 새로운 보안 기술을 연구하는 데 어려움이 따른다. 이상 탐지(anomaly detection)를 통해 정상적인 패킷이 아닌 외부로부터의 공격을 탐지하거나 퍼징(fuzzing) 테스트를 통해 특정 차량의 내부통신이 가진 외부 공격에 대한 보안 수준을 객관적으로 평가하고 취약점을 탐지하는 등의 차량 보안 연구는 명확한 한계를 가지게 된다.

본 논문에서는 이러한 문제를 완화하기 위하여 CAN 트레이스의 분석을 통해 데이터 필드 영역의 세부 구조를 추론하는 방법을 제안한다. 제안하는 방법은 크게 두 단계로 구성된다. 첫 번째 단계에서는 전체 트레이스에서 블록 단위로 일부 기록을 선정하고 각 블록마다 독립적으로 정적 필드 구분을 수행한다. 두 번째 단계에서는 각 블록의 중간 구분 결과를 통해 최종적인 필드 구분 결과를 추론한다. 추가로, 제안하는 방법과 기존 방법의 혼합된 방식이 제시된다. 기본적인 추론 방법은 몇 가지 CAN 프레임 내 데이터 필드에 포함되는 값의 특성을 이용하며, 이를 통해 기존의 필드 구분 방법과 비교하여 더 낮은 계산 비용으로 더 높은 정확도의 필드 추론 결과를 보인다. CAN 프레임 내 데이터 필드의 세부 형식에 대한 더 정확한 필드 추론 결과는 앞서 언급한 이상 탐지 기술의 성능을 높이거나 생성 기반 퍼징(generation-based fuzzing)을 통해 차량 내 ECU에 탑재된 소프트웨어에 대한 테스트 수준을 높일 수 있다.

논문의 구성은 다음과 같다. 먼저 2장에서 CAN 프로토콜의 배경에 대해 서술하며 CAN 프로토콜과 관련된 기존 공격 사례들과 프로토콜 역공학 관련 연구에 대해 소개한다. 3장에서는 본 논문에서 제안하는 기법인 필드 구분 방법에 대해 서술하며, 4장에서는 실제 차량에서 취득한 CAN 트레이스와 시뮬레이션 기반으로 생성된 CAN 트레이스를 대상으로 실험을 수행한 결과에 대해 서술한다. 마지막으로 5장에서 결론 및 추후 연구에 대해 서술한다.

II. 연구 배경

2.1 Controller-Area Network

CAN(Controller Area Network) 프로토콜은 현재 자동차 산업 분야에서 널리 사용되고 있는 버스 기반 네트워크 표준 통신 방식으로서, 1983년 독일의 Bosch사에 의해 개발이 시작되었다. CAN 프로토콜은 CAN 버스 네트워크상에서 노드 간 데이터 전달 방법을 명시하는데, CAN 네트워크에서 노드는 ECU(Electronic Control Unit)를 의미한다. ECU란 센서와 엑츄에이터가 부착된 임베디드 장비를 지칭한다. 자신의 주변 환경에 대한 정보를 센서를 이용해 읽어 들이고, 엑츄에이터를 통해 이에 따른 적절한 행동을 수행한다. 차량 내부에는 이러한 수많은 ECU들이 버스 네트워크에 연결되어 통신 환경을 구성한다. CAN 버스의 데이터 라인은 CAN_H와 CAN_L로 구성되는 트위스트 페어 선을 사용하며, 네트워크상의 모든 노드들이 두 선에 연결되어 있어 버스 상에서 전송되는 모든 메시지들을 수신할 수 있다. CAN 버스에서 각각의 노드들은 프레임을 단위로 통신을 수행한다.

각 CAN 메시지는 버스 상에서 유일한 식별자를 가지는데, CAN 표준 형식에서는 그 크기가 11비트이고 확장 형식에서는 29비트의 크기를 가진다. 식별자는 해당 메시지에 대한 우선순위를 부여하는 의

미도 가진다. 식별자 번호가 낮을수록 더 높은 우선순위를 가지게 되며, 우선순위가 높은 메시지의 순서대로 CAN 버스의 사용 권한을 가지게 된다. CAN 버스에서 데이터 전송 노드는 수신 노드가 필요로 하는 메시지의 식별자를 데이터 프레임의 식별자 필드에 명시하여 브로드캐스트 방식으로 데이터를 송신하며, 수신 노드는 자신이 필요로 하는 정보에 해당하는 식별자가 포함된 메시지의 경우에만 수신하고 그렇지 않은 메시지는 무시한다.

2.2 관련 연구

2.2.1 CAN 버스 보안

CAN 프로토콜은 그 특성 상 보안에 대한 고려 없이 설계되었기 때문에, 현재까지 CAN 프로토콜을 대상으로 하는 수많은 공격 사례들이 보고되고 있다. 자동차의 CAN 네트워크로 접속할 수 있는 가장 대표적인 방법은 OBD-II 포트를 이용하여 물리적으로 접근하는 것이다. 이를 통해 공격자는 CAN 네트워크상의 패킷들을 수집하고 역공학을 통해 패킷들의 기능을 분석할 수 있다. 또한 자신이 임의로 만든 CAN 패킷을 버스로 주입하여 차량의 오동작을 발생시키기도 한다. [1], [5]에서는 이와 같은 과정을 통해 공격자가 CAN 버스에 침입하여 차량 계기판과 같은 특정 ECU에서 출력되는 정보 값을 변조하거나, 브레이크의 동작을 멈추게 하고, 특정 ECU의 메모리로부터 값을 읽어오거나 값을 쓰는 것이 가능함을 보였다. 2000년대에 이르러서는 차량에 주행 기능 외에 편의성 측면이 중요하게 부각되면서 차량 내 컴포넌트들이 인터넷에 연결되기 시작했다. 이에 따라 [4]에서는 차량에 물리적으로 접촉하지 않더라도 외부에서 발생할 수 있는 공격 위협들에 대한 연구 결과를 제시하였다. 해당 연구에서는 차량 외부에서 들어올 수 있는 공격 유형을 공격이 일어나는 범위에 따라 직접 접근(OBD-II), 근거리 접근(Bluetooth, Wi-Fi), 장거리 접근(Cellular) 등으로 구분하여 각각의 공격 유형 별 수행 가능한 공격과 공격의 범위, 공격에 따른 권한의 획득 정도 등에 대한 결과를 도출하였다. 또한 [6]에서는 2014년 Jeep Cherokee 모델의 헤드 유닛을 통해 실제 차량에 직접적으로 접근하지 않더라도 Wi-Fi, Cellular network를 통해 원격에서 차량에 대한 공격이 가능함을 실험하였다. 또한 차량에 탑재된 인포테인먼트 시스템을 통해서도 공격이 발생할 수 있는데, [2]에서는 차량 내부의 인포테인먼트 시스템과 스마트폰 어플리케이션 사이에 사용되는 프로토콜인 MirrorLink의 안전성에 대한 분석 연구를 수행하였다. 그 결과 공격자가 MirrorLink 프로토콜의 취약점을 이용하는 악성 앱을 통해 차량의 내부 네트워크로 악의적인 메시지를 전송할 수 있음을 보였다. 한편 현재 최신 차량에 자율주행 기능이 추가되기 시작하면서 자동차에 탑재되는 수많은 센서들에 대한 공격에 대해서도 연구가 활발히 이루어지고 있다. 일례로, [3]에서는 테슬라 모델 S를 대상으로 레이더, 초음파 센서, 카메라 등 주행에 사용되는 센서를 대상으로 재밍 공격과 스푸핑 공격이 가능함을 보였다.

차량을 대상으로 하는 주요 공격 사례는 대부분 임의의 차량을 제어하거나 일부 기능 정지 및 오동작을 야기하기 위해 차량 내부 CAN 네트워크로 이상 데이터를 주입하려는 시도를 한다. 차량 관련 기술이 발전하면서 차량과 외부와의 통신 인터페이스가 늘어나고 차량 내 전자기기 및 소프트웨어 탑재가 늘어나는 것은 불가피한 반면 이에 대한 사이버 공격을 완전히 막아낸다는 것은 불가능에 가깝다. 따라서 외부 공격을 통해 차량내부 네트워크로 공격 패킷이 주입되었을 때 이를 식별하기 위한 이상탐지기술의 연구가 몇 가지 존재한다[8,12,13,14,15,16]. 차량내부 통신 환경을 대상으로 하는 대부분의 이상탐지 연구는 메시지의 주기 및 순서, 네트워크의 엔트로피 등 차량내부 네트워크로부터 특정 정보를 추출하고 이를 바탕으로 모델을 설립하여 이상 패킷을 감지한다. 특히 M. Markovitz 등[8]은 CAN 트레이스로부터 ID 별 데이터필드에 대한 모델을 만들고 해당 모델을 벗어나는 패킷을 공격 패킷으로 간주하는 방식으로 이상 패킷을 탐지하는 방법을 제안한다.

2.2.2 프로토콜 역공학

프로토콜 역공학은 표준이 공개되지 않은 사유 프로토콜에 대해서 표준 문서, 명세서 등 해당 프로토콜의 정보가 없는 상태에서 해당 프로토콜의 형식이나 문법을 추론하는 것을 목표로 한다[7]. 프로토콜 역공학은 대표적으로 특정 프로토콜이 구현된 소프트웨어의 안전성을 테스트하는데 활용되며, 역공학을 통해 추론된 프로토콜 모델을 바탕으로 유의미한 테스트 케이스를 생성하여 블랙박스(black-box) 테스트 대비 더 효과적인 테스트를 가능케 한다. 그 밖에

악성 소프트웨어의 통신 방식을 분석하거나 사유 프로토콜을 시뮬레이션하기 위해서 활용될 수 있다. 가장 대표적인 프로토콜 역공학 도구로는 Netzob[17]이 있으며, 현재 소스코드가 Github[18]에 공개되어있다.

그러나 기존의 일반적인 인터넷 환경을 대상으로 하는 프로토콜 역공학 연구[19, 20, 21]가 존재함에도 불구하고, 이러한 방식을 차량내부네트워크 환경에 그대로 적용하기에는 어려움이 존재한다. 기존의 프로토콜 역공학 방식은 주로 클라이언트-서버 모델에서 대상들 간에 주고받는 메시지를 기반으로 프로토콜의 문법이나 FSM(Finite State Machine) 등을 추론한다. 반면 CAN 프로토콜의 경우 수많은 ECU가 하나의 버스 네트워크상에서 서로 메시지를 주고받으며, 각 메시지에는 송신자 및 수신자 주소 대신 해당 메시지의 우선순위 및 메시지 별 구분을 위한 식별자 정보만 존재하기 때문에 각 메시지의 송·수신 ECU를 명확히 식별하기 힘들다. 또한 차량 내부통신의 메시지는 특별한 경우를 제외하면 대부분 바이너리 정보만을 담기 때문에 텍스트 기반 메시지에 쓰이는 역공학 기술의 효과를 기대하기 힘들다.

이러한 어려움으로 인해 [8]에서는 기존 프로토콜 역공학과는 다른 방법으로 CAN 버스 네트워크만을 위한 역공학 방법을 제안하였다. 해당 연구는 앞서 언급한 바와 같이 CAN 버스를 위한 이상 탐지 기술의 일부로서 CAN 프레임 내 데이터 필드에 대한 필드 구분 및 모델링 방법을 설명한다. 이를 위해 해당 연구에서는 실체 차량(Ford Escape 및 Toyota Prius)의 CAN 트레이스를 분석하여 CAN 프레임 내 데이터 필드의 세부 필드 타입을 상수(constant), 다중 값(multi-value), 카운터(counter), 센서(sensor) 네 가지로 정의한 뒤, 성능평가를 위해 해당 네 가지의 세부 필드를 가진 가상의 CAN 트레이스를 시뮬레이션하고 이를 대상으로 그리디(greedy) 알고리즘 형태의 필드 구분 알고리즘을 적용하여 각 ID 별 세부 필드를 추론한다. 개략적인 필드 추론 알고리즘은 다음과 같다. 먼저 8바이트 내 데이터 필드에 존재 가능한 $(64+1)\cdot 64/2 = 2{,}080$ 개의 모든 후보 필드에 대해 각 필드마다 전체 트레이스에서의 유일한 값의 개수를 계산한다. 그 뒤 각 후보필드를 유일한 값의 개수에 따라 상수, 다중 값, 카운터/센서 중 하나의 타입으로 분류하고 필드 타입 및 크기 별로 우선순위와 점수를 매겨 가장 높은 우선순위와 점수를 가진 필드부터 선택한다. 후보 필드 하나를 선택할 때마다 겹치는 후보 필드는 버려지며 더 이상 남은 후보필드가 없을 때까지 반복하여 최종 필드 구분을 도출한다. 해당 연구에 따르면, 임의로 시뮬레이션된 10개 ID의 CAN 트레이스에 대하여 그리디 알고리즘을 수행했을 때 약 26.1%의 필드 분류 거리(field classification distance)를 보인다. 여기서 필드 분류 거리는 해당 연구에서 정의하는 정확도 측정기준으로서 0에 가까울수록 추론 결과가 실제와 정확히 일치함을 의미하며 상세 기준은 5.2에 서술한다.

```
;   Message Number
;   |       Time Offset (ms)
;   |       |        Type
;   |       |        |       ID (hex)
;   |       |        |       |       Data Length
;   |       |        |       |       |   Data Bytes (hex) ...
;   |       |        |       |       |   |
;---+--   ----+----  --+--  ----+---  +   -+ -- -- -- -- -- -- --
     1)    342732.6  Rx        02B0  5   7E FD 00 07 2E
     2)    342734.5  Rx        04F0  8   80 00 1A 00 00 DD CD 10
     3)    342734.8  Rx        0350  8   00 20 B2 77 79 00 00 9C
     4)    342735.0  Rx        02C0  8   01 00 00 00 00 00 00 00
     5)    342735.7  Rx        0580  8   00 00 00 00 00 00 00 00
     6)    342736.5  Rx        0316  8   31 1E E8 0A 1E 17 00 7F
     7)    342736.8  Rx        018F  8   00 2C 1E 00 00 43 00 00
     8)    342737.0  Rx        0080  8   00 17 E8 0A 1E 17 1E 11
     9)    342737.2  Rx        0081  8   80 84 63 00 00 00 00 21
    10)    342737.4  Rx        0260  8   05 1E 00 30 28 92 4F 10
    11)    342737.7  Rx        02A0  8   A2 00 63 9C 5D 04 F5 00
    12)    342737.9  Rx        0329  8   0F A8 7F 0C 12 2A 00 10
    13)    342738.2  Rx        0545  8   D8 11 00 8B 39 00 3A 00
    14)    342738.5  Rx        043F  8   00 40 60 FF 4A 80 09 00
    15)    342738.8  Rx        0370  8   FF 20 00 00 FF 00 00 20
    16)    342739.0  Rx        0440  8   FF 00 00 00 FF 84 09 00
    17)    342742.1  Rx        02B0  5   7C FD 00 07 1F
    18)    342744.7  Rx        0350  8   00 20 C2 77 79 00 00 EC
    19)    342744.9  Rx        02C0  8   01 00 00 00 00 00 00 00
    20)    342747.3  Rx        0316  8   31 1E CE 0A 1F 17 00 7F
    21)    342747.5  Rx        0080  8   00 17 CE 0A 1E 17 1F B2
    22)    342747.7  Rx        0081  8   80 84 63 00 00 00 00 12
    23)    342748.0  Rx        018F  8   00 2C 1F 00 00 43 00 00
    24)    342748.2  Rx        0260  8   05 1F 00 30 28 93 51 2A
    25)    342748.5  Rx        02A0  8   C2 00 63 9C 5D 04 F5 00
    26)    342748.7  Rx        0329  8   0F A8 7F 0C 12 2A 00 10
    27)    342748.9  Rx        0370  8   FF 20 00 00 FF 00 00 E4
    28)    342749.2  Rx        043F  8   00 40 60 FF 4A 80 09 00
    29)    342749.4  Rx        0440  8   FF 50 00 00 FF 84 09 00
    30)    342749.6  Rx        0545  8   D8 11 00 8B 38 00 3C 00
    31)    342751.7  Rx        02B0  5   7C FD 00 07 E0
    32)    342754.6  Rx        04F0  8   80 00 1C 00 00 DD CD 10
```

Fig. 1. An example of CAN trace from Hyundai Avante 12' (generated by PCAN-view).

III. 필드 구분

본 장에서는 CAN 버스의 트레이스를 분석하여 각 ECU ID 별 데이터 필드 내 세부 필드를 구분하는 방법에 대해 설명한다. 본 논문에서 제안하는 방법은 크게 두 단계로 구성된다. 첫 번째 단계에서는 전체 트레이스에서 블록 단위로 일부 기록을 선정하고 각 블록마다 독립적으로 정적 필드 구분을 수행한다. 두 번째 단계에서는 각 블록의 중간 구분 결과를 토대로 CAN 데이터의 특징을 이용하여 최종적인 필드 구분 결과를 추론한다. 각 단계에 대한 구체적인 내용은 3.2 및 3.3 절에 서술한다.

3.1 CAN 데이터 수집

CAN 버스의 트레이스를 수집하기 위해서는 일반적으로 차량 내 위치한 OBD-II(On-Board Diagnostics) 포트를 이용한다. OBD-II 포트를 통해 CAN 데이터를 PC로 전달받기 위해서는 별도의 하드웨어 인터페이스가 필요하다. 본 연구에서는 PCAN-USB 및 Kvaser OBD-II to Dsub9 케이블[9, 10], 그리고 PCAN-USB와 함께 제공되는 PCAN-view 소프트웨어를 통해 CAN 버스의 네트워크 트레이스를 수집하였으며, 대상 차량은 현대 아반떼 2012년 모델 및 기아 쏘렌토 R 2009년 모델이다.

일반적인 인터넷 환경에서 Wireshark[11] 등의 도구를 통해 수집된 네트워크 트레이스(network trace)처럼, CAN 버스의 트레이스에는 각 프레임별 주요 필드 값 정보가 Fig.1과 같이 저장된다. 각 CAN 프레임에는 해당 프레임이 수집된 시간, 중재 필드(arbitration field), DLC(Data Length Code), 데이터 필드 등의 값이 포함된다. 중재 필드는 ID 필드로도 불리며 ID 값은 각 차량 제조사 또는 차량 모델 별로 ECU 간에 주고받는 각 메시지에 할당하는 우선순위의 의미를 가진다. 참고로, 본 연구에서 수집된 CAN 트레이스에서는 같은 ID 값을 가지는 CAN 프레임은 모두 같은 값의 DLC 및 같은 길이의 데이터 필드를 가진다.

본 연구에서는 기본적으로 각 CAN 프레임의 ID 및 데이터 필드만을 필드 구분에 활용한다. 수집된 CAN 트레이스에는 차량 별로 약 22~25개의 서로 다른 ECU ID가 존재하며 절반 이상의 ID가 약 10ms의 주기로 전송되어, 전체 네트워크에서 초당 약 1,500~2,000개의 프레임이 기록된다.

수집된 CAN 트레이스의 형태는 동일한 차종이라고 하더라도 어떠한 환경 하에서 데이터를 취득하였

```
function split_static(d_0, d_1, ..., d_{m-1})
    S ← 0
    for i in [0:m-1]   # m-1 exclusive
        S ← S ∨ (d_i ⊕ d_{i+1})
    end for
    return groupby(S)
```

Fig. 2. Static partitioning algorithm.

```
function subdivide(pos_l, pos_r, sen)
    # pos_l: position of the left-most bit
    # pos_r: position of the right-most bit
    # sen: sensitivity
    fields ← []
    if pos_r = pos_l then
        fields.append(Field(pos_l, pos_r))
        return fields
    end if
    ptr_l ← pos_l
    ptr_r ← pos_r
    while ptr_r ≤ pos_r do
        n_pos_r = get_n_pos_r(ID, pos_r)
        # get_n_pos_r: get the number of identical
                       pos_r from R_0,...,R_{k-1} of
                       a given ID
        if n_pos_r / k > 1-sen or
           ptr_r = pos_r do
           fields.append(Field(ptr_l, ptr_r))
           ptr_l ← ptr_r + 1
        end if
        ptr_r ← ptr_r + 1
    end while
```

Fig. 3. subdivision of variable field algorithm

는지에 따라 달라질 수 있다. 예를 들어 차량이 시동만 켜지고 제자리에 정지된 상태에서 수집된 트레이스와 가속 및 감속뿐만 아니라 와이퍼, 도어락, 기어 변속 등 다양한 동작을 수행하면서 수집된 트레이스가 있을 때, 둘 모두 같은 시간동안 수집되었다고 하더라도 기록된 데이터의 종류에는 차이가 발생한다. 즉 필드 구분을 수행하기 위해서는 최대한 다양한 종류의 데이터가 기록된 트레이스를 대상으로 하는 것이 실제와 더욱 가까운 결과를 얻을 수 있다. 본 연구에서는 자동차 내에 갖춰진 여러 기능을 최대한 동작시키면서 수집된 CAN 데이터 트레이스를 실험 데이터로 사용한다.

3.2 블록 별 정적 구분

본 절에서는 전체 트레이스의 일부 블록에 대해 정적 필드 구분(static field-partitioning)을 적용하는 과정을 서술한다. 전체적인 과정은 다음과 같

다. 전체 t개의 CAN 프레임 정보를 담은 트레이스에서 각각 n개의 연속된 프레임을 포함하는 k개의 블록 $B_0, B_1, ..., B_{k-1}$를 선정한다. 그 뒤, 선정된 각각의 블록에 대해 독립적으로 정적 필드 구분을 수행한다. CAN 트레이스에 대한 정적 필드 구분은 특정 블록 내 같은 ID 값을 가진 CAN 프레임의 데이터 필드에 대해, 해당 블록 내에서 고정된 값을 가지는 필드와 그렇지 않은 필드를 비트 단위로 구분하는 것을 의미한다. 여기서 고정된 값을 가지는 필드는 상수(constant) 필드, 변하는 값을 가지는 필드는 변수(variable) 필드로 표현한다. 구분은 각 필드가 최대한의 크기를 가지도록 하여 상수 필드와 변수 필드 각각이 서로 같은 종류의 필드와 인접하지 않도록 수행한다.

비트열로 구성된 데이터 필드에 대해 정적 필드 구분을 수행하는 방법은 다양하며, 본 논문에서는 XOR과 OR연산을 이용하여 상수 필드와 변수 필드를 구분한다(Fig. 2). 한 블록 내에서 동일한 ID 값을 가진 m개의 CAN 프레임의 l-비트 크기 데이터 필드 값(비트열) $d_0, d_1, ..., d_{m-1}$ 및 모든 비트가 0인 l-비트 크기의 비트열 S에 대해, 순차적으로 각 d_i와 d_{i+1}의 XOR값에 S와의 OR 비트연산을 누적하여 수행하면 상수 필드는 0, 변수 필드에는 1의 값을 가진 비트열이 생성된다. 해당 비트열에서 연속적인 0, 1 값을 하나의 그룹으로 해석하면 각 그룹의 범위가 상수 또는 변수 필드의 범위를 표현하게 된다. 이와 같은 정적 필드 구분 과정을 각 블록 내 서로 다른 ID값을 가진 프레임별로 적용하여 블록 별 중간 필드 구분 결과 R_i를 도출한다.

$$R_i = \{field_0, field_1, ...\}, (0 \le i < k) \quad (1)$$

3.3 최종 필드 구분

최종적으로 세부 필드 경계를 도출하기 위한 기본 개념은 몇 가지 CAN 통신의 특성에 기반을 둔다. 우선 CAN 프레임 내 데이터 필드에는 각 ECU에서 측정된 각종 센서 값 및 제어를 위한 설정 값 등이 포함된다. 이 중 센서 값은 일정한 범위를 가지며 각 ID 별 CAN 프레임의 DLC는 고정되어있으므로, ECU 간 프로토콜 정의 시 일반적인 인터넷 프로토콜과 같이 데이터 내에 따로 세부 필드들의 길이를 명시하는 필드가 사용되지 않는다. 즉 ECU 간 프로토콜에는 일반적으로 여러 센서 값을 최대 64비트 크기의 데이터 필드에 담기 위한 비트단위의 위치와 범위를 미리 할당한다. 특정 센서 값이 최대 l개의 정보를 표현해야 한다면 해당 센서 값을 담기 위해 데이터 필드 내에 최소 $\lceil \log_2 l \rceil$ 비트가 고정적으로 할당될 필요가 있으며 대부분의 네트워크 프로토콜에서는 빅엔디언(big-endian) 방식을 따르므로 이 범위 내에서 특정 시점에 작은 값이 들어가면 왼쪽의 상위 비트들의 값은 0의 값을 가지게 된다. 이러한 특성으로 인해, 3.2의 블록 별 정적 구분 시 각 블록의 크기 n을 적절히 조절하면 센서 값들이 변하는 폭이 일정 범위 이내로 한정됨을 알 수 있다.

이러한 CAN 통신의 특성을 이용하여, 3.2의 방법으로 도출된 각 블록의 ID 별 정적 필드 구분 결과를 통해 최종 필드 구분 결과를 추론한다. 필드 구분의 기본 개념은 정적 필드 구분 결과에서 각 변수 필드의 오른쪽 경계부분을 이용하여 세부 필드를 구분하는 것이다. 특정 ID의 최종적인 세부 필드를 구분하기 위한 구체적인 절차는 다음과 같다. 우선 블록 $B_0, B_1, ..., B_{k-1}$에서 특정 ID에 해당하는 모든 프레임의 데이터 필드 값에 대해 정적 필드 구분을 수행하여 전체 블록에 걸쳐 변하는 부분과 변하지 않는 부분(상수 필드 및 변수 필드)을 도출한다. 그 뒤, 각 변수 필드에 대해 Fig.3과 같은 방법으로 세부 필드를 구분한다. 주어진 변수 필드의 범위(pos_l~pos_r) 내에서 가장 왼쪽의 1비트 크기의 후보 필드($pos\_l \sim pos\_l$)부터 오른쪽으로 1비트씩 크기를 늘려가며 해당 후보 필드를 최종 경계로 확정할 것인지 여부를 판단한다. $R_0, ..., R_{k-1}$의 정적 필드 구분 결과에서 해당 후보 필드의 오른쪽 경계와 같은 오른쪽 경계를 가진 변수 필드가 존재하는 블록의 개수를 u라고 할 때, 해당 후보 필드의 u/k가 특정 임계값($1-sen$)을 넘으면 최종 필드 경계로 결정한다. sen은 필드 구분의 민감도(sensitivity)로서 [0,1]의 범위를 가지며 해당 값이 클수록 필드가 더 잘게 구분되는 경향을 보인다.

마지막으로, 여기까지 도출된 민감도별 필드 구분 결과에서 각 변수 필드 앞에 모든 비트가 0의 값을 가진 특정 길이 T_m 이하의 상수필드가 존재할 경우 이 두 필드를 하나의 필드로 병합한다. 이러한 병합은 임의로 생성된 가상의 CAN 트레이스에서보다 실제 차량의 CAN 트레이스에서 필드 구분의 정확도를 높이는데 도움이 될 수 있다. 예를 들어 특정

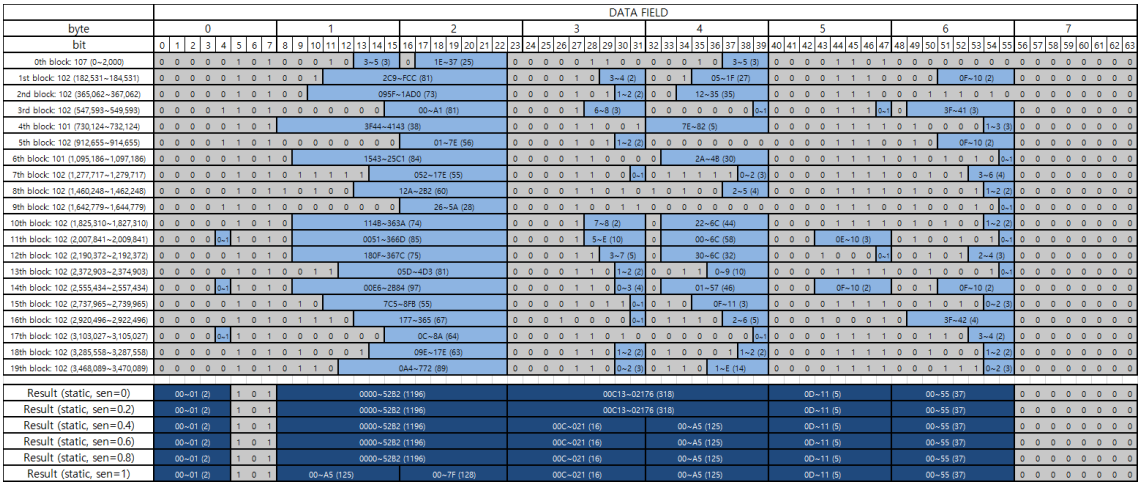

Fig. 4. An example of static field-partitioning results from CAN ID 0x316 of KIA SorentoR. Grey area represents constant field and blue area represent variable field. ($t = 3650621, n = 2000, k = 20$, intervals between blocks$= 180531$) In each variable field, the minimum value, the maximum value, and the number of unique values in the field are specified.

ID의 64비트 크기 데이터필드 내 세부 필드가 모두 8비트 크기이고 각 필드에 작은 값만 포함된 트레이스를 분석한다면 이러한 병합은 실제 필드의 구분 결과와 더욱 가까운 결과를 도출할 수 있다.

IV. 실험 결과

본 장에서는 3장에 서술한 필드 구분 방법을 실제 CAN 트레이스 및 시뮬레이션으로 생성된 CAN 트레이스를 대상으로 적용한 결과를 제시한다. 실험은 Python 구현을 통해 이루어지며, Intel(R) Core(TM) I5-4278U 2.60GHz 프로세서, 16GB 메모리 사양의 노트북 및 Windows 10 Home 64비트 운영체제 환경에서 수행된다.

4.1 실제 CAN 트레이스 대상 필드 구분

본 절에서는 실제 CAN 트레이스를 통해 도출된 결과의 일부를 제시한다. Fig.4는 기아 쏘렌토R의 CAN ID 0x220에 대한 블록 별 정적 필드 구분을 수행한 결과(위) 및 민감도 별 최종 필드 추론 결과(아래)를 나타낸다. 8~22번째 비트 범위를 제외한 나머지 영역은 앞서 언급한 바와 같이 빅엔디언의 특성에 따라 각 블록에서 변화된 영역의 오른쪽 경계 위치가 일정한 것을 알 수 있다. 23~39번째 비트 범위의 결과는 민감도에 따라 필드 경계가 다르게 도출되나, 해당 범위에서는 변수 필드의 범위가 서로 겹치는 부분이 없기 때문에 높은 확률로 31번째 비트와 32번째 비트 사이가 경계임을 추측할 수 있다. 반면 8~22번째 비트 범위에서는 민감도에 따라 최종 필드 구분 결과가 다르게 도출될 뿐만 아니라 상수 필드의 범위가 서로 겹치는 부분이 존재한다. 0번째 블록의 16번째 비트가 1비트 크기의 상수 필드로 구분됨으로 인해, 민감도가 1일 경우 0번째 블록의 13~15번째 비트 범위의 변수 필드에 의해 15번째 비트와 16번째 비트 사이가 필드 경계로 도출된다.

4.2 시뮬레이션 CAN 트레이스 대상 필드 구분

실제 차량에서 쓰이는 CAN 상위 프로토콜의 명세는 알려져 있지 않으므로 실제 CAN 트레이스를 대상으로 수행한 실험 결과의 정확도를 판단하기 힘들다. 따라서 본 연구에서는 [8]에서 제시된 시뮬레이션 모델에 따른 가상의 CAN 트레이스를 생성하여 필드 구분 결과의 정확도를 검증한다.

CAN 프레임 내 세부 필드는 [8]에 정의된 바와 같이 상수(const.), 다중 값(multi-value), 카운터(counter), 센서(sensor) 4 가지로 분류하여 생성한다. 즉 가상의 CAN 트레이스에는 임의로 생성된 각 ECU ID 별 프레임의 데이터 필드마다 위 4가지 필드가 임의의 크기와 순서로 할당된다.

본 실험에서는 총 10개의 ID를 가진 CAN 버스

Table 1. Type and length of the randomly generated sub-fields in each CAN message in the simulation. Information of only three out of ten IDs are presented. Each number in parenthesis means the length of a sub-field (bits).

ID	Fields
1	const (14), sensor (5), counter (10), const (6), sensor (4), multi-value (13), sensor (9), const (3)
2	multi-value (7), sensor (10), const (4), sensor (15), multi-value (15), sensor (12), counter (1)
3	counter (15), multi-value (6), counter (7), const (9), sensor (11), multi-value (12), multi-value (4)
...	...

Table 2. Properties of ID 1's message format in the simulation. The noise parameter of sensor field means the maximum value of random noise.

Field type	length	Properties
const	14	constant value=0x107D
sensor	5	amplitude=24, period=39095, base=8, phase=38489, noise=4
counter	10	start=3
const	6	constant value=0x17
sensor	4	amplitude=6, period=15289, base=1, phase=11543, noise=1
multi-value	13	items={3060, 892, 3860, 6052, 7892, 7436, 6653, 4832, 3672, 7473, 3487, 706}
sensor	9	amplitude=376, period=2288, base=67, phase=410, noise=75
const	3	constant value=0x3

의 네트워크 트레이스를 생성하였다(Table 1 및 2 참조). 데이터 필드는 모두 64비트 크기이며 각 ID 별 데이터 필드 내 세부 필드의 크기와 종류는 모두 임의로 생성된다. 세부 필드가 너무 크거나 너무 작을 경우 실제 CAN 메시지와는 동떨어진 형식이 생성될 수 있으므로 각 필드는 최소 4비트에서 최대 16비트의 크기로 제한한다. 각 ECU는 평균 10ms 마다 한 프레임을 전송하며, 전체 300초의 시뮬레이션 시간 동안 총 약 30만개의 트레이스를 생성하였다.

필드 구분의 성능을 평가하기 위한 기준은 [8]의 필드 분류 거리(field classification distance)와 동일하게 적용한다. 필드 분류 거리는 실제 필드 구분과 추론된 필드 구분 간의 거리를 두 가지 기준으로 측정한다. 첫 번째 기준은 CAN 프레임 내 데이터 필드의 0에서 63번째까지 각 비트 중에서 실제 필드와 추론된 필드 간 분류가 서로 다른 비트의 개수이다. 두 번째 기준은 실제와 다르게 추론된 필드 경계의 개수이다. 실제 필드 경계가 아님에도 불구하고 추론된 경계에 존재하는 경계 또는 실제 필드 경계임에도 불구하고 추론된 경계에는 존재하지 않는 경계가 개수에 포함된다. 이 두 거리의 합산을 s라고 할 때, s의 최대값은 127이며, 최종 필드 분류 거리는 $s/127$로 표현한다. 필드 분류 거리는 값이 작을수록 더욱 정확한 추론 결과임을 의미한다.

본 실험에서는 총 세 가지 알고리즘을 수행한 결과를 비교한다. 첫 번째는 3.2와 3.3장에 설명된 정적 구분 방법, 두 번째는 [8]에서 제안된 그리디 알고리즘, 세 번째는 앞의 두 방법을 함께 적용하는 혼합(hybrid) 방식이다. 첫 번째 정적 구분 방법을 통해 도출된 필드 구분 결과는 상수 필드와 변수 필드 두 종류이므로, 본 실험에서는 정확한 필드 분류 거리 측정을 위해 추가적인 필드타입 분류과정을 수행하여 이전 연구와 동일하게 네 가지 타입의 필드를 도출한다. 이를 위해 각 변수 필드는 해당 범위 내 유일한 값의 개수에 따라 [8]과 동일한 기준을 적용하여 다시 다중 값, 카운터/센서 중 하나의 타입으로 분류된다. 세 번째 방식은 첫 번째 방식을 먼저 적용하여 필드 구분 결과를 도출하고 해당 결과에서 각 변수 필드만을 대상으로 독립적으로 그리디 알고리즘을 부분 적용하여 다시 세부 필드를 구분한다.

Fig.5는 전체 트레이스에서 5000개의 프레임을 대상으로 각각 세 가지 버전의 알고리즘을 적용하여 도출된 ID 별 필드 구분 거리를 나타낸다. ID 3과 4를 제외한 모든 결과에서 정적 구분과 혼합 방식이 단순 그리디 알고리즘보다 좋은 성능을 보인다. 특히 혼합 방식은 절반 이상의 ID에 대해 가장 좋은 결과를 보인다. Fig.6은 분석되는 메시지의 개수에 따른 전체 10개 ID에 대한 알고리즘별 평균 필드 분류 거리를 나타낸다. 세 알고리즘 모두 분석되는 메시지

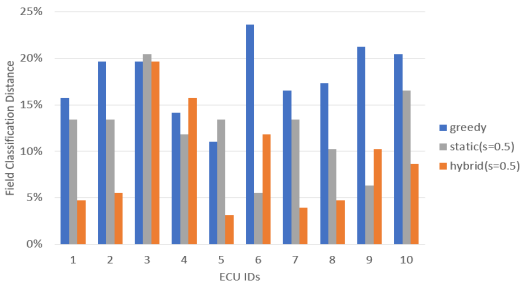

Fig. 5. Field classification distances by 10 different simulated ECU ID traces. Each algorithm analyzed 5,000 frames (10 blocks for static algorithm).

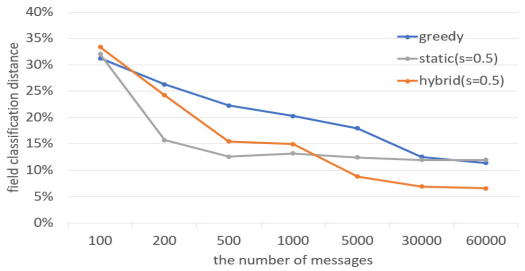

Fig. 6. Average field classification distances for each algorithm by the number of analyzed frames.

의 수가 늘어남에 따라 더 낮은 필드 분류 거리를 나타낸다. 정적 구분의 경우 분석되는 메시지의 수가 500개를 넘으면 12~13% 수준의 필드 구분 거리 값을 유지하는 반면 그리디 알고리즘과 혼합 방식은 메시지의 수가 늘어남에 따라 더 낮은 필드 구분 거리 결과를 보인다. 특히 혼합 방식은 5000개 이상의 메시지에 대해 나머지 두 방식보다 더 좋은 결과를 보인다. 그리디 알고리즘, 정적 분석 방법, 혼합 방식은 60000개 메시지에 대해 각각 11.42%, 11.97%, 6.61%의 필드 분류 거리 결과(Fig.6 참조)를 보인다.

4.3 계산 비용

Fig.7은 분석 대상 메시지의 수에 따른 각 알고리즘별 수행 시간을 나타낸다. 그리디 알고리즘과 혼합 방식의 경우 메시지의 수가 늘어남에 따라 수행 시간이 늘어나는 반면 정적 분석의 경우 메시지의 수에 큰 영향을 받지 않는다. 정적 분석은 XOR 및 OR 비트연산만을 수행하므로 매우 빠른 결과 도출

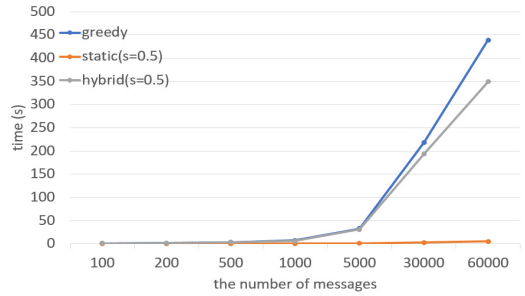

Fig. 7. Time elapsed for each algorithm by the number of frames analyzed.

이 가능하다. 그리디 알고리즘의 경우 상수 필드 도출까지는 정적 분석과 결과가 동일하나 같은 결과를 도출하기까지 불필요한 계산 및 메모리 사용이 존재한다. 이는 혼합 방식의 수행 시간이 그리디 알고리즘 방식보다 짧음을 통해 확인할 수 있다. 혼합 방식은 상수 필드를 도출하기까지 정적 구분과 동일한 방식이 적용되므로 해당 계산비용만큼의 오버헤드를 줄이는 효과를 볼 수 있다. 실험 결과에서 총 6만개의 메시지에 대해 그리디 알고리즘은 438초, 정적 분석 방식은 5.1초, 혼합 방식은 349초가 소요되었다.

V. 결 론

본 연구에서는 차량 제조사 및 차량 모델 별로 다른 CAN 상위 프로토콜, 즉 데이터 필드 내 세부 필드의 경계와 각 필드의 타입을 추론하기 위한 방법을 제안하였다. 제안한 방법은 정적 구분을 통한 중간 추론 결과를 토대로 데이터 필드 값의 특성을 이용하여 최종적인 필드 경계를 추론한다. 해당 방식은 기존에 비해 더 낮은 계산 비용을 보이며, 이전 연구에서 제안된 방식과 제안하는 방식을 혼합한 방식을 통해 이전 연구의 결과에 비해 필드 분류 거리(field classification distance)를 기준으로 약 42% 향상된 필드구조 추론 결과를 보인다.

CAN 프레임 내 데이터 필드의 세부 형식에 대한 더 정확한 필드 추론 결과는 다양한 기술에 활용될 수 있다. CAN 트레이스 분석을 통해 도출된 필드구조 모델을 이용하여 이상 패킷을 탐지하는 기술[8]의 경우 더 정확한 필드 추론 결과를 통해 이상 탐지의 오탐율을 낮출 수 있을 것으로 기대된다. 추가로, 차량의 내부통신 안전성을 평가하기 위한 방법으로 퍼징을 활용할 경우 제조사에서 공개하지 않는 데이터 필드에 대해 생성 기반 퍼징(generation-based

fuzzing)에 가까운 방식으로 테스트를 가능케 하여 전체 테스트 케이스의 수 및 테스트 시간을 줄이는 효과를 보일 것으로 기대된다.

마지막으로, 본 연구에서 제안하는 방식은 이전 연구와 마찬가지로 4가지 종류의 필드만을 정의하며 실제 CAN 데이터에 비해 단순화된 형태의 시뮬레이션을 통해 검증되었다. 차후에는 더 다양한 형태의 CAN 데이터 모델이 필요할 것으로 사료되며, 필드 구분 방법에서도 데이터마이닝 또는 기존 프로토콜 역공학 기법 중 일부 기술을 추가로 적용할 경우 더 높은 정확도를 보일 수 있을 것으로 기대된다.

References

[1] C. Miller and C. Valasek, "Adventures in automotive networks and control units," DEF CON 21, Aug. 2013

[2] S. Mazloom, M. Rezaeirad, A. Hunter, and D. McCoy, "A Security Analysis of an In-Vehicle Infotainment and App Platform," WOOT 2016, Aug. 2016

[3] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," DEF CON 24, Aug. 2016

[4] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," USENIX Security Symposium, pp. 77-92 Aug. 2011

[5] K. Koscher, A. Czeskis, F. Roesner, S. Patel, and T. Kohno, "Experimental security analysis of a modern automobile," Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, pp. 447-462, May 2010

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

[7] Julien Duchene, Colas Le Guernic, Eric Alata, Vincent Nicomette, and Mohamed Kaaniche, "State of the art of network protocol reverse engineering tools," Journal of Computer Virology and Hacking Techniques, 2017

[8] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown CAN bus networks," Vehicular Communications, vol. 9, pp. 43-52, Jul. 2017

[9] PCAN-USB, https://www.peak-system.com/PCAN-USB.199.0.html?L=1

[10] Kvaser OBD II to Dsub9 Adapter Cable, https://www.kvaser.com/product/kvaser-obd-ii-to-dsub9-adapter-cable/

[11] Wireshark, https://www.wireshark.org

[12] K. Cho and K. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," USENIX Security Symposium, pp. 911-927 Aug 2016

[13] A. Taylor, S. Leblanc and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on. IEEE, pp. 130-139, Oct. 2016

[14] S. Mukherjee, J. Walker, I. Ray and J. Daily, "A Precedence Graph-Based Approach to Detect Message Injection Attacks in J1939 Based Networks," 15th International Conference on Privacy, Security and Trust(PST), Aug. 2017

[15] D. Vasistha, "Detecting Anomalies in Controller Area Network for Automobiles," PhD Thesis. Texas A&M University, Aug 2017

[16] M. Muter, N Asaj, "Entropy-based anomaly detection for in-vehicle networks," Intelligent Vehicles Symposium (IV), 2011 IEEE, pp.

1110-1115, Jun. 2011
〔17〕 G. Bossert, F Guihéry and G Hiet, "Towards automated protocol reverse engineering using semantic information," Proceedings of the 9th ACM symposium on Information, computer and communications security, pp. 51-62, Jun. 2014
〔18〕 Netzob, https://github.com/netzob/netzob
〔19〕 C. Leita, K. Mermoud, and M. Dacier. "Scriptgen: an automated script generation tool for honeyd," In Proceedings of ACSAC, pp. 203-214, Dec. 2005
〔20〕 W. Cui. "Discoverer: Automatic protocol reverse engineering from network traces," In Proceedings of USENIX Security Symposium, pp. 1-14, Aug. 2007
〔21〕 Y. Wang, X. Yun, M. Z. Shaq, L. Wang, A. X. Liu, Z. Zhang, D. Yao, Y. Zhang, and L. Guo. "A semantics aware approach to automated reverse engineering unknown protocols," In Proceedings of ICNP, pp. 1-10, Oct. 2012

〈저자소개〉

지 청 민 (Cheongmin Ji) 학생회원
2012년 2월: 아주대학교 정보컴퓨터공학과 학사
2012년 3월~현재: 아주대학교 컴퓨터공학과 석·박사 통합과정
〈관심분야〉 차량 보안, 임베디드/IoT 보안, 블록체인 보안, 영상데이터 보안

김 지 민 (Jimin Kim) 학생회원
2015년 2월: 아주대학교 정보컴퓨터공학과 학사
2015년 3월~현재: 아주대학교 컴퓨터공학과 석·박사 통합과정
〈관심분야〉 임베디드/IoT 보안

홍 만 표 (Manpyo Hong) 종신회원
1981년 2월: 서울대학교 계산통계학과 학사
1983년 8월: 서울대학교 계산통계학과 석사
1991년 2월: 서울대학교 전산과학과 박사
1985년 3월~2016년 2월: 아주대학교 정보컴퓨터공학부(과) 교수
2016년 3월~현재: 아주대학교 사이버보안학과 교수
〈관심분야〉 기반시설보안, 차량보안, 임베디드/IoT 보안, 금융보안, 병렬처리