Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.3, Jun. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.3.679

커넥티드 카 환경에서 안드로이드 앱 리패키징을 이용한 자동차 강제 제어 공격*

이 정 호,† 우 사 무 엘, 이 세 영, 이 동 훈‡
고려대학교 정보보호대학원

A Practical Attack on In-Vehicle Network Using Repacked Android Applications*

Jung Ho Lee,† Samuel Woo, Se Young Lee, Dong Hoon Lee‡
Graduate School of Information Security, Korea University

요 약

자동차에 여러 종류의 통신기기가 장착되고 운전자는 통신기기를 이용해 다양한 정보를 자동차에서 수집할 수 있게 되었다. 최근 운전자는 스마트폰을 이용하여 자동차의 상태정보를 실시간으로 획득하고 원격으로 자동차를 제어할 수 있다. 그러나 안드로이드 기반 앱은 변조와 재배포가 쉽다. 운전자는 임의의 공격자가 변조하여 배포한 악성 앱을 사용하게 되면 자동차에서 발생하는 다양한 정보의 유출위협과 운전자의 의도와 다른 제어로 사고를 유발하는 위협에 노출될 수 있다. 더욱이 자동차 내부 네트워크에는 보안기법이 적용되어 있지 않아 보안 위협으로부터 쉽게 노출되어 사고 발생 시 인명과 재산상의 큰 피해를 야기할 수 있다. 본 논문에서는 안드로이드 공식 앱 마켓인 구글 플레이에 배포된 다양한 자동차 진단용 앱의 취약점을 분석한다. 분석된 취약점을 통해 배포된 자동차 진단용 앱에서 발생 가능한 공격 모델을 알아본다. 그리고 실제 자동차에서 실험으로 공격모델에 대한 위험성을 검증한다. 마지막으로 자동차 진단용 앱 리패키징을 사용한 공격자의 악성행위로부터 안전한 보호기법을 제안한다.

ABSTRACT

As vehicle started to contain many different communication devices, collecting external information became possible in IoT environment. In such environment, remotely controling vehicle is possible when vehicle information is obtained by looking in to vehicle network through smart device. However, android based smart device applications are vulnerable to malicious modulation and redistribution. Modulated android application can lead to vehicle information disclosure that could bring about vehicle control accident which becomes threat to drivers. furthermore, since vehicles today does not contain security methods to protect it, they are very vulnerable to security threats which can cause serious damage to users and properties. In this paper, many different vehicle management android applications that are sold in Google Play has been analyzed. With this information, possible threats that could happen in vehicle management applications are being analysed to prove the risks. the experiment is done on actual vehicle to prove the risks. Also, access control method to protect the vehicle against malicious actions that could happen through external network in IoT environment is suggested in the paper.

Keywords: In-Vehicle Network, Control Area Network, Android Application Repackaging

Received(03. 18. 2016), Modified(04. 18. 2016), Accepted(04. 19. 2016)

* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음.[B0101-16-0553, 자동차 전장 ECU간 보안전송기술 개발].

† 주저자, junghlee1987@gmail.com

‡ 교신저자, donghlee@korea.ac.kr(Corresponding author)

I. 서 론

최신 자동차는 100개 이상의 ECU(Electronic Control Unit)라 불리는 전자제어장치를 장착하고 있다[1]. 운전자는 자동차에 장착된 ECU로 주행 중 다양한 편의를 제공받고 있다[2][3]. 그리고 자동차 제조사는 운전자에게 더 많은 기능을 제공하기 위해 자동차에 장착된 ECU의 수를 증가시키고 있다[4]. 한 대의 자동차에 장착된 ECU 간의 효율적인 통신을 위한 자동차 내부 네트워크로는 CAN(Controller Area Network), LIN(Local Interconnect Network), MOST (Media Oriented System Transport), FlexRay가 존재한다[5]. 이 중 대표적으로 사용되는 CAN은 BOSCH에서 개발된 단일 회선의 버스 네트워크이다[6].

최근 ICT(Information and Communication Technology)와 융합된 자동차가 생산됨으로써 자동차는 IoT환경의 일부가 되었다. 따라서 운전자는 운전 중 필요한 정보수집이 가능하게 되었다. 이렇게 ICT와 융합된 자동차를 커넥티드 카 또는 스마트 카라고 부른다. 커넥티드 카 환경을 만들기 위해서는 자동차 제조사에서 자동차 출고 시 기본으로 통신장치를 장착하거나 운전자의 사용 용도에 따라 자동차 출고이후 추가적으로 통신장치를 자동차에 장착할 수 있다. 자동차에 장착된 통신장치를 이용해 운전자는 정보를 수집할 수 있고, 외부에서 자동차를 원격으로 조작할 수 있다. 그러나 자동차에 장착된 통신장치 때문에 기존 ICT환경에서의 보안위협은 자동차로 옮겨지게 되었다[7]. 더욱이 자동차를 제어하는 ECU 간에 통신 프로토콜인 CAN은 보안위협을 고려하지 않고 설계되어 기존 외부 네트워크에서 적용되는 보안 공격기법에 매우 취약하며 이를 사용한 많은 자동차 취약점 연구가 진행되었다[8][9][10].

최근 많은 운전자들은 스마트폰과 자동차 간의 통신을 이용해 다양한 편의를 제공하는 앱을 사용한다. 그러나 안드로이드 앱의 경우 역공학과 리패키징에 취약한 문제점이 널리 알려져 있다[11]. 만약 임의의 공격자가 배포한 자동차 진단용 앱을 분석하고 리패키징으로 자동차에 악성행위가 가능한 앱으로 변조하여 재배포를 하면 공격자의 의도대로 자동차를 제어할 수 있을 것 이다.

본 논문의 분석대상은 구글 플레이에 배포된 자동차 진단용 앱이다. 분석대상이 될 앱은 자동차의 OBD-Ⅱ(On Board Diagnostics)단자에 부착된 ELM327모듈과 통신을 하면서 자동차의 상태를 실시간으로 알 수 있다. 자동차 진단용 앱은 자동차 정비에 이용되는 정비규약으로 자동차의 상태를 실시간으로 사용자에게 보여준다.

본 논문에서는 구글 플레이에 배포된 자동차 진단용 앱의 동작과정과 취약점을 분석한다. 분석된 결과를 사용하여 안드로이드 앱 개발과 역공학에 대한 기본적인 지식만 갖추었다면 쉽게 자동차 진단용 앱을 변조하여 자동차를 임의로 강제 제어가 가능함을 확인할 수 있을 것이다. 그리고 자동차 진단용 앱 리패키징을 사용한 공격자의 악성행위로부터 안전한 보호기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 배경지식을 설명하고, 3장에서는 본 논문에서 다루어지는 관련된 연구를 정리한다. 4장에서는 실제 자동차에서 적용 가능한 공격모델을 제시하고, 5장에서는 제시한 공격모델을 실제 자동차에서 실험을 통해 현실적으로 가능함을 보인다. 6장에서는 보호기법을 제시하고, 마지막 7장을 결론으로 마무리 한다.

II. 배경 지식

2.1 CAN

CAN은 자동차 내부에 탑재되는 ECU 간의 효율적인 통신을 지원하기 위해 1980년대 초반 BOSCH에 의해 개발되었다. CAN은 CSMA/BA (Carrier Sense Multiple Access/Bitwise Arbitration)기법과 버스 네트워크 토폴로지를 지원하는 송신자 ID기반의 브로드캐스트 통신 기법으로 자동차 내부 ECU들 사이의 통신 회선의 복잡성과 길이를 획기적으로 감소시켰다[12]. 이러한 이유로 자동차 업계에서는 신속하게 CAN을 도입하였으며 1993년에는 ISO국제 표준 규격(ISO 11898)으로 제정 되었다. 본 논문에서는 CAN 2.0B 기반의 자동차 내부 네트워크 기준으로 설명하며 메시지 패킷구조는 Fig. 1.과 같다.

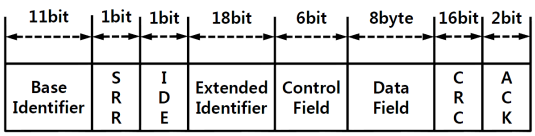

Fig. 1. CAN 2.0 B Packet Frame Format

2.2 커넥티드 카

자동차와 ICT의 융합을 통해 다양한 커넥티드 카 서비스가 상용화 되었다. 대표적인 서비스로는 GM의 Onstar, BMW의 Connected Drive, 그리고 Merceds-Benz의 mbrace2 등이 있다. 일반적으로 커넥티드 카는 자동차 운행 중에도 외부 네트워크가 항상 연결된 상태를 의미한다[7]. 커넥티드 카의 구성요소는 다음과 같다.

- The Vehicle: ECU들이 설치되어 자동차 내부 네트워크를 구성한 자동차
- The Portal: 자동차에 다양한 네트워크 서비스를 제공하는 Portal
- The Communication Link: 자동차와 Portal을 연결 해주는 통신 Link

Communication Link는 Before Market과 After Market 두 가지 형태로 구성된다. Before Market은 자동차 제조사가 자동차에 텔레매틱스 장치와 같은 이동통신 모듈을 자동차 출고 시 설치하여 Communication Link 역할을 수행한다. 그리고 After Market에서는 완성 차가 판매된 후 운전자의 성향에 따라 자동차의 OBD-Ⅱ단자에 무선 통신 모듈을 설치하여 Communication Link를 구축하는 방식으로 이루어져 있다. Portal은 웹 기반의 서비스와 스마트폰의 앱 기반 서비스로 나뉜다. 최근에는 스마트폰의 고성능화, 대중화로 인해 스마트폰과 자동차용 무선 통신 모듈을 통해 커넥티드 카 환경을 구성하는 After Market 서비스들이 급격하게 늘어나고 있다.

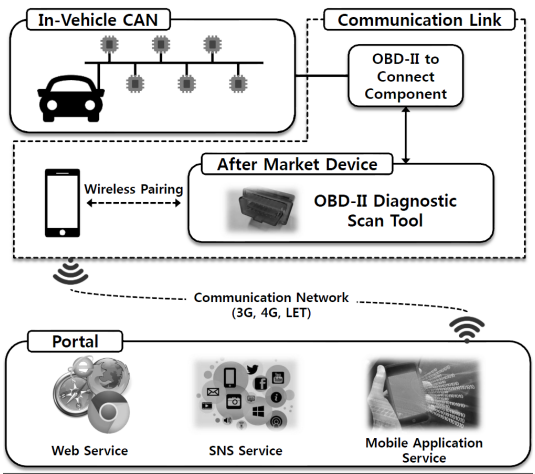

Fig. 2. Connected Car Environment

Fig. 2.는 After Market에서 판매되고 있는 H/W와 S/W를 이용하여 커넥티드 카 환경을 구축한 예를 보여주고 있다.

2.3 OBD 진단 프로토콜

과거 OBD-1은 자동차의 배출가스를 제어하기 위해 사용되었고 현재 OBD-Ⅱ로 발전하면서 배출가스 제어 및 자동차의 모든 ECU의 진단 정보를 얻기 위해 사용되고 있다[13]. 최근 자동차는 ECU의 상태를 진단하기 위해 자동차용 정비기기를 사용한다. 이 기기는 OBD PID(On Board Diagnostics Parameter ID)를 통해 ECU의 상태정보를 요청하며, SAE J1979 표준[14]을 사용한다. 이 때 사용하는 자동차 진단 요청 패킷 프레임의 데이터 구조는 Fig. 3.과 같다. 자동차용 정비기기가 송신한 요청 패킷 프레임의 CAN ID는 0x7DF를 사용한다. Data Field의 3번째 바이트에 자동차 상태정보 요청을 위한 PID 코드를 삽입한다. 자동차 내부에 탑재된 ECU는 요청 패킷 프레임을 수신한 후 응답 패킷 프레임을 송신한다. 자동차 진단 응답용 패킷 프레임의 데이터 구조는 Fig. 4.와 같다. PID protocol을 이용하면 자동차의 ECU들이 측정한 다양한 상태정보들과 ECU의 동작 여부 등을 확인 할 수 있다[15][16].

자동차의 상태정보를 확인하기 위해 요청된 정보에 해당하는 응답 패킷 프레임의 데이터 필드에서 실제 자동차의 상태정보를 계산하기 위한 데이터는 Fig. 4.의 Parameter 1~4에 표현된다. 그리고 이 Parameter값을 Table 1.의 산술식에 적용하면 자동차의 상태정보를 얻을 수 있다[14]. 예를 들어 요청 패킷 프레임의 데이터 필드가 [02 01 0D 00 00 00 00 00]이면 현재 자동차의 속도를 요청하는

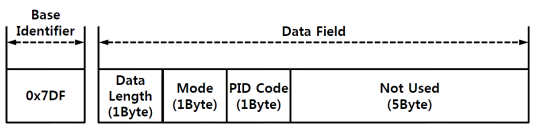

Fig. 3. Diagnostic Request Packet Format

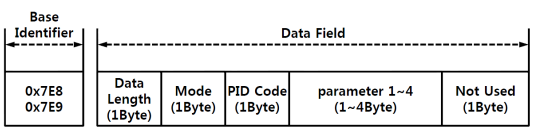

Fig. 4. Diagnostic Response Packet Format

Table 1. OBD PID Code Description

PID	Description	Units	Formula	PID	Description	Units	Formula
04	Calculated engine load value	%	A*100/255	0F	Intake air Temperature	℃	A-40
0A	Fuel Pressure	kPa	A*3	11	Throttle Position	%	A*100/255
0C	Engine RPM	rpm	((A*256)+B)/4	21	Distance traveled with malfunction indicator lamp	km	(A*256)+B
0D	Vehicle Speed	km/h	A	49	Accelerator pedal postion D	%	A*100/255

것이다. 이에 대한 응답 패킷 프레임의 데이터 필드가 [03 41 0D 33 00 00 00 00]이면 4번째 데이터 필드 값으로 현재의 자동차 속도가 51km/h임을 확인 할 수 있다.

2.4 ELM327

ELM327은 자동차 내부 네트워크와 외부의 자동차용 정비기기 사이의 통신을 지원하는 마이크로 컨트롤러이다. ELM327 기반의 무선통신 모듈을 이용하면 운전자 스마트폰과 자동차 내부 네트워크를 무선으로 연결하여 커넥티드 카 환경을 구축할 수 있다. ELM327 기반의 무선통신 모듈들은 OBD-Ⅱ 단자에 직접 연결되어 사용된다. ELM327은 SAE J1850-PWM, SAE J1850-VPW, ISO 9141-2, ISO 14230-4, ISO 15765-4, SAE J2411, SAE J1939, SAE J1939 등[13]의 자동차 내부 네트워크의 다양한 통신 프로토콜을 지원한다. ELM327 마이크로 컨트롤러 제조사인 ELM Electronics[17]는 ELM327을 이용한 각종 H/W와 S/W를 제작할 수 있는 Data Sheet[18]를 제공한다.

III. 관련 연구

3.1 자동차 내부 네트워크 보안 취약점 연구

CAN은 다음과 같은 보안 취약점이 존재한다.

- 약한 접근제어: 악성 ECU를 CAN에 추가 설치하는 것은 간단한 일이다. CAN은 추가된 ECU를 식별하지 못하며 추가된 ECU의 통신 참여를 제어 할 수 없다
- 메시지 암호화/인증 결여: CAN은 각 노드간 통신에 있어 암호화/인증이 지원되지 않는다. 그렇기 때문에 평문의 메시지를 볼 수 있으며 수신된 평문 메시지로 재전송 공격이 가능하다

이러한 취약점을 통해 공격자는 자동차를 임의로 제어 할 수 있다. Karl Kosher 등의[8] 연구는 자동차 내부 네트워크의 취약점에 대해 정리하고 실제 자동차에서 실험으로 위험성을 증명하였다.

3.2 안드로이드 리패키징 공격

안드로이드 기반 기기에서 앱을 설치하기 위해 사용하는 파일은 APK(Android application PacKage)이다. APK의 내부구성은 앱 개발자가 작성한 소스 코드를 컴파일한 결과물인 DEX(Dalvik EXecutable format)파일과 앱이 가지는 권한이 포함된 Manifest, 앱의 Layout을 작성한 XML파일, 앱에서 사용하는 이미지파일이 포함된 리소스로 구성되어 있다. 안드로이드 기기에 설치된 앱은 ADB Shell로 APK를 획득할 수 있다. APK에 포함된 DEX파일을 사용해 원본과 유사한 Java 코드와 Smali 코드를 얻을 수 있다. DEX파일로 Java 코드 복원이 가능한 이유는 가상머신에서 동작하는 바이트코드의 구조적 특징을 가지고 있기 때문이다. 바이트코드에는 클래스 이름, 메소드 이름, 사용하는 변수명 등 다양한 정보가 포함되어 있다. 분석된 결과를 통해 Smali 코드에서 앱을 변경하고 재컴파일을 하여 APK파일로 만드는 것이 가능하다. 또한 안드로이드의 정책상 개발자 스스로 앱에 서명을 하여 배포하기 때문에 변조된 앱 배포 시 공격자

가 임의로 서명하여 배포하는 것이 가능하다 [19][20][21].

3.3 안드로이드 난독화

안드로이드 난독화 기법이란 악의적인 역공학으로부터 APK의 중요 로직이나 코드, 리소스 등을 보호하기 위해 사용되는 기법이다[22]. 프로그램의 기능성은 유지하면서 역공학을 어렵게 하기 위해 다양한 기법들이 사용되고 있다.

식별자 변환(identifier renaming) 기법은 클래스, 메소드, 필드 등의 이름을 의미 없는 문자열로 치환하여 의미정보를 삭제하는 기법이며, 제어 흐름 변환(control flow change)은 소스 코드의 실행 흐름을 변환하거나 코드 사이에 더미 코드(dummy code)를 삽입함으로써 분석을 어렵게 하는 기법이다. 문자열 암호화(string encryption), 클래스 암호화(class encryption), 리소스 암호화(resource encryption) 기법은 암호화 기법을 사용하여 소스 코드 내의 문자열이나 APK 내의 클래스 파일, 리소스 파일을 암호화하여 보호하는 기법이다. 암호화된 정보들은 APK 실행 중에 동적으로 복호화되어 사용된다. API 은닉 기법은 자바 리플렉션(reflection)을 활용하여 특정 라이브러리 또는 메소드의 호출을 감추는 기법이다. 현재 APK에 난독화 기법을 적용하기 위해서 ProGuard[23], DexGuard[24], DexProtector[25] 등 다양한 난독화 도구가 이용되고 있다. ProGuard는 무료 난독화 도구로서 식별자 변환 기법을 제공한다. DexGuard와 DexProtector는 상용 난독화 도구로서 앞서 서술한 기법들을 이용하여 APK의 저작권을 보호하거나 악성 APK로의 리패키징을 방지한다.

IV. 공격 모델

본 장에서는 앱 리패키징을 이용한 자동차 공격 모델을 제안한다. 제안하는 공격 모델은 기존에 배포/판매되고 있는 앱에 자동차를 임의로 제어 가능한 악성코드를 삽입한 후 재배포하는 공격 모델이다. 이미 상용화된 앱을 변조하여 배포하기 때문에 불특정 다수의 사용자들은 해당 앱이 악성 앱이라는 것을 인지할 수 없다. 본 논문이 제안하는 공격 모델에서 공격자의 능력, 타겟의 취약점, 피해자의 행위는 다음과 같다.

4.1 공격자의 능력

공격자는 자동차용 정비기기를 사용하여 특정 ECU를 강제 제어할 수 있는 CAN 데이터 프레임을 획득할 수 있다. 또한 공격자는 앱 마켓에 배포/판매되고 있는 앱에 특정 코드를 삽입할 수 있는 기본적인 능력을 가지고 있으며, 변조된 앱을 개방형 마켓에 재배포 할 수 있다. 공격자가 배포한 변조된 앱은 자동차의 주행 상태를 분석한 후 특정 시점에 ECU 제어용 CAN 데이터 프레임을 자동차 내부 네트워크에 삽입할 수 있다.

4.2 타겟의 취약점

공격 대상 자동차는 ECU간 통신을 위해 CAN을 사용한다. 또한 Fig. 2.와 같이 ELM327기반 무선 통신 모듈과 운전자 스마트폰을 사용하여 커넥티드 카 환경을 구성하고 있다. ECU 간의 통신을 위한 CAN은 자동차의 주행 안전에 초점을 맞춰 설계되었다[26][27]. 따라서 ECU간 통신에 있어서 메시지 암호화와 인증과 같은 보안요소는 배제되어 있기 때문에 커넥티드 카 환경에서는 기존 IT환경에서의 메시지 재전송공격과 위장공격 등의 보안 문제점을 가지고 있다.

4.3 피해자의 행위

피해자는 안드로이드 기반 기기와 ELM327 기반의 무선통신 모듈을 사용하는 커넥티드 카 환경을 구축하고 있다. 피해자는 다양한 커넥티드 카 서비스를 이용하기 위해 구글 플레이에서 자동차 진단용 앱을 다운받아 사용한다. 2015년 10월 기준, 구글 플레이에서 배포/판매하고 있는 자동차 진단용 앱은 약 300여개이며 이 중 ELM327 기반 무선통신 모듈과의 통신을 지원하는 앱은 약 200여개 정도이다. 이 200여개의 앱중 80여개의 앱은 유료로 판매되고 있다. 유료 앱이 개방형 마켓에 무료로 배포된다면 많은 사람들은 이를 이용할 것이다. 하지만 리패키징 후 배포된 앱을 사용한다면 본래 앱 개발자의 서명이 없으므로 앱의 업그레이드 또는 같은 개발자의 서명이 있는 앱들 간의 데이터 공유가 불가 하다[28].

Table 2. ELM327 AT Command Description

AT Command	Description	AT Command	Description
AT D	Set all to Defaults	AT H0*/AT H1	Header off* or on
AT Z	Reset All	AT R0/AT R1*	Responses off or on*
AT BRT hh	Set Baud Rate Timeout	AT SH xyz	Set Header to xyz
AT E0/AT E1*	Echo off or on*	AT SP h	Set Protocol to h and save it
AT AL	Allow Long(>7byte) message	AT CAF0/AT CAF1*	Automatic Formatting off or on*
AT AR	Automatically Receive	AT WM[1-6bytes]	Set the Wakeup Message

4.4 공격 시나리오

본 논문에서 제시하는 공격 시나리오는 크게 두 가지이다.

- 변조된 앱 실행 시 ECU강제 제어 데이터프레임을 자동차 내부 네트워크인 CAN에 삽입하는 공격
- 변조된 앱이 자동차 주행 동안 관련 정보를 분석한 후 자동차의 특정 조건에 ECU강제 제어 데이터 프레임을 자동차 내부 네트워크인 CAN에 삽입하는 공격

V. 실 험

본 장에서는 실제 자동차를 사용하여 4장에서 제시한 공격 시나리오가 현실적으로 공격 가능함을 증명한다. 공격 실험은 앱 분석을 위한 사전 준비단계와 가능함을 검증하는 검증 단계로 나뉜다.

5.1 사전 준비단계

본 절에서는 ELM327의 동작 원리와 ELM327 기반의 무선 통신 모듈을 사용하는 자동차 진단 앱을 분석한다.

5.1.1 ELM327통신 프로토콜 분석

자동차의 OBD-Ⅱ단자에 장착되어 자동차 진단용 앱과 함께 사용되는 ELM327모듈의 CAN ID는 정비용 ID인 '7DF'로 설정되어 있다. 그래서 ELM327 모듈을 통해 임의로 자동차를 제어하기 위해서는 자동차 제어에 필요한 CAN ID로 변환하여 데이터를 전송해야 한다. ELM Electronic[17]에서 제공하는 ELM327 Data Sheet에는 ELM327모듈을 사용자의 의도대로 제어 할 수 있는 AT Command가 존재한다[18]. Table 2.에 ELM327 Data Sheet를 참조하여 ELM327모듈을 제어할 수 있는 다양한 AT Command를 정리하였다. AT Command를 이용해 ELM327을 제어하여 사용자는 실제 CAN으로 원하는 CAN ID의 데이터 전송이 가능하다. 이 명령어를 통해 실제 CAN으로 원하는 데이터 전송이 가능하다. Fig. 5(A).은 실제 자동차에서 실험한 환경이다. Fig. 5(B).은 ELM327을 제어할 수 있는 터미널 앱으로 사용자 임의로 원하는 CAN ID로 변환하여 Data를 전송할 수 있음을 Fig. 5(C).으로 확인할 수 있다. 이를 통해 공격자는 ELM327 모듈로 임의의 CAN ID로 데이터 전송이 가능함을 확인 할 수 있다.

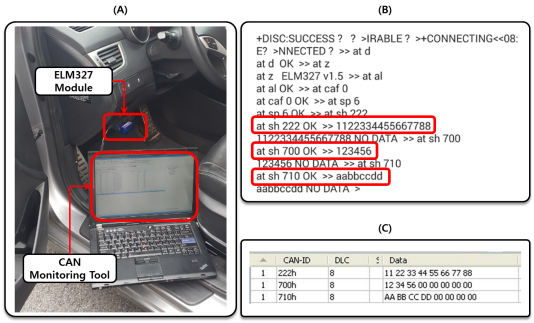

Fig. 5. ELM327 Analyzing Implementation

5.1.2 자동차 진단 앱 분석

현재 구글 플레이에서 ELM327로 검색된 앱은 무료 140개, 유료 85개가 존재한다. 여기서 실제자동차의 OBD-Ⅱ단자에 장착된 ELM327과 통신 가능한 무료 자동차 진단용 앱은 75개이다. 나머지 무

료 자동차 진단용 앱 65개는 사용이 불가함을 확인했다. 본 논문에서는 동작 가능한 무료 앱 75개와 유료 앱 중 가장 많이 배포된 10개의 앱을 선택하여 실험을 수행하였다.

자동차 진단용 앱은 ELM327과 블루투스를 사용해 통신한다. 이 때 앱에서 발생하는 통신 메시지를 확인하기 위하여 다음 Fig. 6.과 같은 실험 환경을 설정하였다. 이 실험 환경을 통하여 자동차의 상태를 점검하기 위한 앱과 ELM327모듈간의 통신 프로세스가 Fig. 7.과 같이 동작함을 확인하였다. 각 단계의 동작에 대한 설명은 다음과 같다.

- Step 1: 앱과 ELM327간의 블루투스 페어링 연결
- Step 2: 앱에서 ELM327의 통신 프로토콜 초기 설정
- Step 3: 앱에서 PID를 사용한 자동차의 상태를 요청

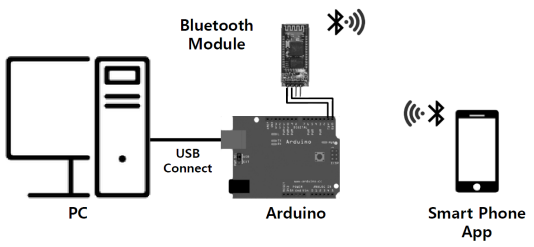

Fig. 6. Application Analyzing Experiment Environment

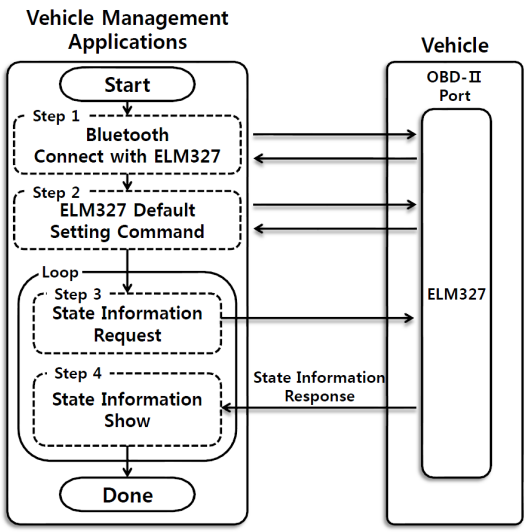

Fig. 7. Diagnostic Application Communication Process

- Step 4: 응답 받은 자동차의 상태정보를 사용자에게 전달

만약 각 단계의 Request-Response 메시지를 공격자가 획득하고 이를 검색하여 Smali 코드에서 변조한다면 앱을 사용해 쉽게 자동차를 제어할 수 있다. 앱에서 전송하는 메시지 탐색은 Fig. 6.의 실험 환경을 통해 얻을 수 있다. 이 실험환경에서 간단히 획득한 메시지는 "ATZ", "ATD", "ATSP6"와 같은 Step 2의 Request 메시지이다. Step 2의 메시지는 실험대상인 모든 앱의 Smali 코드에서 단순 문자열 검색만으로도 찾을 수 있다.

공격자는 검색된 Request 메시지 대신 Smali 코드에 공격자가 원하는 메시지로 변경 및 추가 할 수 있다. 그리고 앱 동작 중 자동차 상태정보를 수신하여 처리하는 Smali 코드의 식별이 가능하다면 공격자는 자동차의 특정 상태정보 수신 이후 임의제어 가능한 명령을 Smali 코드에 삽입할 수 있다.

5.1.3 자동차 진단 앱 통신 메시지 분석

본 절에서는 자동차 진단 앱과 ELM327의 통신 메시지를 사용한 앱 분석방법에 대해 설명할 것이다. Fig. 7.과 같은 자동차 진단 앱의 통신 프로세스의 분석을 통해 Step 4에서 자동차의 상태정보가 전달되는 것을 확인하였다. 이때 전달 받은 데이터는 2.3절에서 언급된 ODB PID를 따르며 데이터의 표시형식은 16진수로 표현된다. 하지만 앱에서 사용자에게 표현될 때는 10진수의 형태이다. 10진수로 표현되기 위해 앱은 수신한 자동차 상태정보를 16진수에서 10진수로 변환을 해준다. 이를 위해 안드로이드 자동차 진단용 앱 개발자들은 Java에서 제공하는 기본함수를 다양하게 사용한다. 그 중 대표적으로 사용되는 기본함수는 parseInt, parseLong과 같은 함수가 존재한다. 배포된 앱의 Smali 코드에서 공격자가 원하는 자동차의 상태정보를 확인하기 위해 Fig. 8.과 같이 Log를 삽입하면 확인이 가능하다. 10진수로 변환된 데이터는 PID의 규약을 따르는 자동차의 상태 정보로서 공격자는 이 의미를 쉽게 판별할 수 있다. 그렇기 때문에 공격자가 자동차의 상태정보를 획득한다면 자동차상태의 특정 조건에 자동차를 제어할 수 있다. 이와 같은 Java 기본함수 탐색을 통하여 공격자는 Smali 코드에 자동차 강제제어 코드를 삽입할 곳을 쉽게 찾을 수 있다.

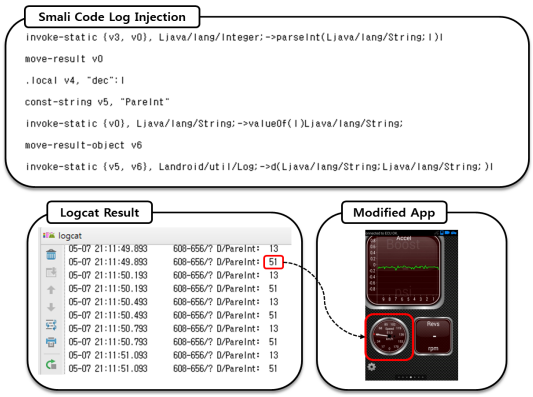

Fig. 8. Smali Code modification

5.1.4 공격 코드 삽입 방법

Java 코드가 컴파일된 결과물은 DEX 바이너리이다. DEX 바이너리를 사람이 읽을 수 있게 표현한 것은 Smali 코드이며, 이 코드에 직접 공격 코드를 직접 작성하는 것은 어려운 작업이다. 그렇기 때문에 공격자는 앱을 분석하기 위해 DEX 파일을 사용하여 Java 코드을 추출한다. 만약 Java 코드 분석과정 없이 단순히 악성 Smali 코드를 획득하고 앱에 삽입한다면 공격자는 쉽게 앱을 변조 할 수 있을 것이다. Fig. 9.에서는 공격자가 제작한 악성 앱에서 필요한 Smali 코드를 추출하여 공격 대상의 앱에 적용하여 악성 앱으로 변조과정을 설명한다[29]. 이 방법으로 공격자가 원하는 악성 Smali 코드를 작성하여 공격 대상 Smali 코드에 쉽게 적용 가능하다.

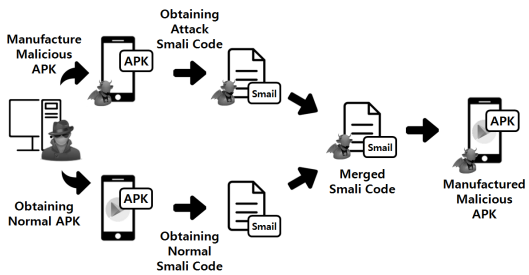

Fig. 9. Malicious Smali Code Injection Method

5.2 검증 단계

5.2.1 자동차 진단 앱 리패키징

본 절에서는 분석된 자동차 진단 앱의 특징으로 앱 리패키징을 사용한 자동차 강제제어 공격 수행이 가능함을 검증한다. 첫 번째 검증단계로 앱에서 발생하는 통신데이터를 변경하여 자동차를 강제 제어한다. 앱에서 발생하는 블루투스 통신 데이터를 탐색하는 환경은 Fig. 6.과 동일하다. 이 실험환경에서 앱 실행 이후 처음 발생하는 통신 데이터인 ELM327 제어 명령어 획득은 간단하게 가능하다. 획득된 데이터를 공격 대상의 앱에서 획득한 Smali 코드에 단순 문자열 검색 결과는 Fig. 10(A).로 확인 할 수 있다. 검색된 Smali 코드에 공격자가 원하는 CAN ID의 데이터를 삽입하기 위한 명령을 Fig. 10(B). 와 같이 넣을 수 있다. 이를 리패키징하여 디바이스에 설치하고 실제 자동차에서 사용하면 공격자가 의도한 CAN ID의 데이터가 자동차으로 전송됨을 CAN전용 모니터링 장비를 사용해 확인 할 수 있다.

두 번째 검증단계는 수신된 자동차 상태정보를 사용하여 자동차를 강제 제어한다. 변조하고자 하는 앱의 Smali 코드에서 수신데이터의 형식을 10진수로 변환하기 위한 Java 기본함수인 parseInt 또는 parseLong이 검색가능하다. 검색된 결과가 실제

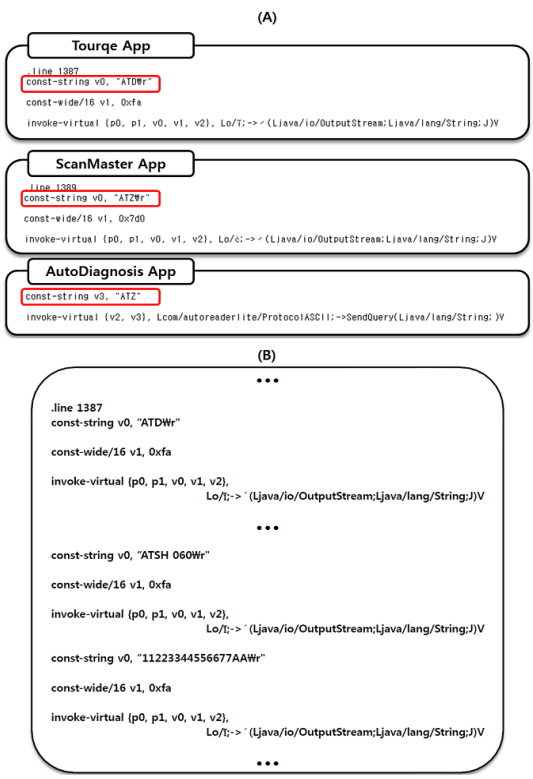

Fig. 10. Command Search and Code Injection

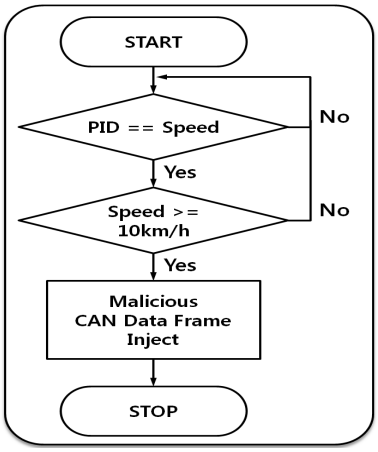

Fig. 11. Malicious Code Injection Flow

자동차 상태정보로 확인된 이후에는 자동차 강제제어를 위한 명령어를 삽입한다. 자동차 강제제어 명령을 위한 Smali 코드를 삽입하기 위해서는 사전에 공격자가 별도로 제작한 앱에서 강제제어 명령어에 해당하는 Smali 코드를 추출한다. 추출된 Smali 코드를 공격 대상의 앱에 삽입하고 리패키징하여 실제 자동차에서 사용하면 공격자가 의도한 CAN ID의 데이터가 자동차로 전송됨을 CAN전용 모니터링 장비를 사용해 확인 할 수 있다. ODB PID의 정비용 응답 패킷 프레임을 시용해 자동차의 속력이 10km/h이상일 경우 자동차 강제제어 명령이 들어가기 위한 조건문은 Fig. 11.과 같다.

5.2.2 실차 실험

앞서 실험된 내용으로 변조된 앱에서 공격자가 원하는 CAN ID의 데이터가 자동차로 전달됨을 확인하였다. 실제 앱 변조를 통해 4.4절의 공격 시나리오를 적용하여 실험하고 검증한다. 본 논문의 실험환경은 Table 3.과 같이 자동차, Hardware, Software로 구분하여 설정하였으며 실험 결과는 [30]의 동영상에서 확인이 가능하다. 동영상에 나와있는 실험은 구글 플레이에 배포된 앱을 변조하여 진행하였다. 동영상의 첫 번째 실험은 첫 번째 공격 시나리오에 해당하며, 앱 실행 시 엔진에 연료공급이 중단되어 자동차의 시동이 꺼짐을 확인할 수 있다. 그리고 두 번째 실험은 두 번째 공격 시나리오에 대한 결과로 공격자가 지정한 자동차 속도를 초과하면 자동차의 엔진에 연료공급이 중단되어 시동이 꺼짐을 확인할 수 있다. 실제 실험에서는 안전을 위하여 변조된 악성 앱이 반응하는 자동차의 속도는 저속인 10km/h로 설정하였다. [30]에서는 자동차의 엔진 정지를 위한 CAN 메시지를 사용하면 저속일 경우 엔진이 바로 정지하여 시동이 꺼지지만 60km/h이상의 속력에서는 엔진이 즉시 정지하지 않고 자동차에서 심각한 흔들림을 느낄 수 있다. 그리고 이후 브레이크를 밟게 되면 엔진이 정지되어 시동이 꺼짐을 확인하였다. 이 실험을 통해 4장에서 제시한 공격 모델이 현실 가능함을 실험으로 증명할 수 있다.

5.2.3 안드로이드 난독화 적용 분석

안드로이드 난독화는 악의적인 역공학으로부터 앱을 보호하기 위해 사용한다. 자동차 진단용 앱의 역공학을 통한 앞선 실험에서 자동차 운행 중 운전자에게 사고위험이 발생 가능함을 확인 하였다. 따라서 악의적인 역공학의 위협으로부터 자동차 진단 앱도 보호를 받아야한다. 대표적인 안드로이드 난독화 도

Table 3. Implementation Environment

Classification	Model Name	Functionality
Vehicle	Omitted	• Target vehicle • Analysis CAN data & Monitoring target
Hardware	Arduino Uno	Application analyze tool
	Bluetooth module	Bluetooth message receiver
	ELM327(Version 1.5)	Diagnostic wireless device
	Samsung Galaxy S5(OS Version 4.4.2)	Target smartphone device
Software	Smali & BakSmali (Version2.1.0)	APK assemble and disassemble tool
	Signapk	APK Self-sign tool
	PCAN Explorer	CAN monitoring and capture tool
	Microsoft Visual Studio 2008	Searching code and modify code tool

Table 4. Obfuscation Option Result

Option	Scenario 1	Scenario 2
String Obfuscation	X	O
Class Obfuscation	X	X
Resource Obfuscation	O	O
Hide Access Obfuscation	O	X

구인 DexProtector의 다양한 난독화 옵션을 사용하여 자동차 진단용 앱을 난독화하고 각 난독화 옵션에 대한 본 논문의 공격 분석 적용가능 여부는 Table 4.와 같다. DexProtector의 난독화 옵션은 문자열을 난독화하는 String 난독화와 APK의 DEX파일을 암호화하는 Class 난독화, 그리고 APK의 assets폴더의 리소스를 암호화하는 Resource 난독화 등으로 구성되어 있다[32]. 본 논문의 분석방법이 일부 난독화 옵션에서 가능한 이유는 분석의 대상이 되는 주체가 개발상 사용되는 일부 문자열과 함수이기 때문에 이를 보호하지 않는 난독화 옵션에서는 쉽게 분석이 가능하다.

VI. 보호기법

커넥티드 카 환경에서 자동차는 다양한 보안 위협으로 노출되어 공격대상이 될 수 있다. 특히 본 논문의 연구를 통해 ELM327 모듈과 통신 가능한 앱을 리패키징으로 자동차 강제 제어가 가능함을 확인하였다. 따라서 커넥티드 카 환경에서 이러한 위협으로부터 보호 가능한 기법이 필요하다. 본 논문에서 제안하는 보호기법은 다음과 같다.

6.1 ELM327 모듈 사용제어

자동차 진단을 위해 사용하는 ELM327 모듈의 CAN ID는 기본으로 '7DF'로 자동차 정비용 ID를 사용한다. 사용자는 ELM327 모듈의 제어 명령으로 원하는 CAN ID의 데이터를 생성할 수 있다. 본 논문의 실험에서 제어 명령을 사용하여 자동차 강제 제어가 가능함을 확인했다. 이와 같은 자동차 강제 제어가 가능한 근본적 원인은 ELM327 모듈에서 '7DF' 이 외의 CAN ID를 사용가능하기 때문이다. 따라서 ELM327 모듈 자체에서 정비용 ID인 '7DF'만 사용하도록 모듈을 설계한다.

6.2 자동차 관리 앱 난독화

안드로이드 앱의 난독화로 악의적인 역공학으로 앱 분석을 여렵게 할 수 있다. Table 4.의 결과와 같이 Resource 난독화와 같이 앱을 분석하는데 있어 어려움이 없는 옵션을 선택한다면 앱을 보호하기 어려울 것이다. 그렇기 때문에 앱의 특성을 파악하고 분석이 어렵도록 적절한 난독화 옵션을 설정하여 앱을 보호해야 한다. 자동차 진단앱을 개발하는 개발자는 본 논문에서의 실험과 같은 위협을 인지하고 앱을 악의적인 역공학으로 보호해야한다. 자동차 진단앱 난독화 적용으로 앱의 분석을 어렵게 하여 리패키징으로부터 보호한다.

6.3 자동차 OBD-II 접근제어

자동차 OBD-Ⅱ 단자는 ELM327를 포함하여 다양한 통신 모듈이 장착되어 사용될 수 있다. 이 다양한 모듈은 자동차 강제제어와 같은 자동차 위협의 원인이 될 가능성이 있다. 현재 OBD-Ⅱ 단자는 CAN 데이터에 대한 접근제어기술이 적용되어 있지 않다. 따라서 위협의 해결을 위해 자동차가 주행 중일 때 특정 CAN 데이터를 제외하고 접근제어를 수행해야 한다. Fig. 12.는 ODB-Ⅱ 단자에서의 접근제어 방법을 나타낸다. 접근제어하기 위한 방법은 자

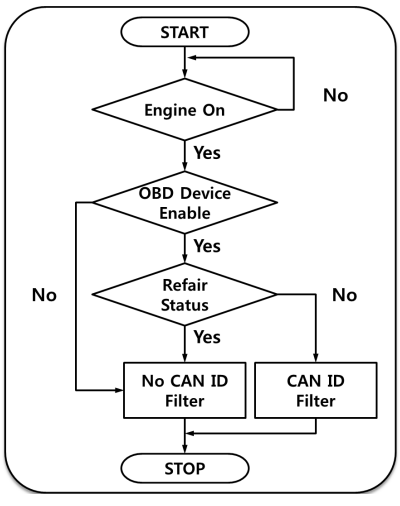

Fig. 12. Countermeasure Data Flow

동차에 통신기기가 장착된 경우 자동차를 정비 중과 주행 중으로 나눈다. 정비 중일 경우 모든 CAN ID의 접근제어를 해제하며, 주행 중일 경우에는 특정 CAN ID 데이터만 접근 가능하도록 필터링 한다. 이와 같은 접근제어를 통해 외부 네트워크에서의 자동차 해킹으로부터 안전한 환경을 만들 수 있다.

VII. 결 론

본 논문에서는 커넥티드 카 환경을 구축하기 위해 시중에서 쉽게 구매 가능한 ELM327 모듈과 앱 마켓에 배포된 다양한 자동차 진단용 안드로이드 앱을 분석하였다. 분석을 통해 공통된 특징과 취약점을 확인하고 이를 사용해 자동차를 임의로 제어 가능한 공격 모델을 만들어 실제 자동차에서 공격 가능성을 검증하였다. 또한 안드로이드 앱 난독화 적용 이후에도 일부 난독화 옵션에서 이전과 동일한 과정으로 분석이 가능함을 확인하였다.

본 논문의 분석 방법은 안드로이드 기반 앱 개발 능력과 역공학에 대한 기초 지식만 갖추었다면 쉽게 자동차 진단용 앱을 분석하고 취약점을 찾아 앱을 변조하여 쉽게 자동차를 임의로 제어 가능함을 확인 하였다. 그리고 본 논문에서 제안하는 보호기법을 통해 앱 리패키징 공격으로부터 자동차를 안전하게 보호할 수 있다.

References

[1] Robert n. charette. this car runs on code. http://www.spectrum.ieee.org/feb09/7649

[2] A. Saad and U. Weinmann, "Automotive software engineering and concepts," GI. Jahrestagung., vol. 34, pp. 318 - 319, 2003.

[3] E. Nickel, "IBM automotive software foundry," in Proc. Conf. Comput. Sci. Autom. Ind., Frankfurt, Germany, 2003.

[4] M.Wolf, A.Weimerskirch, and T.Wollinger, "State of the art: Embedding security in vehicles," EURASIP J. Embedded Syst., vol. 2007, no. 5, p. 1, 2007.

[5] T. Nolte, H. Hansson and L.L. Bello, Automotive communications-past, current and future, in Proceedings of ETFA(Emerging Technologies and Factory Automation), 2005.

[6] K.H. Johansson, M. Törngren, L. Nielsen, "Vehicle applications of controller area network," D. Hristu-Varsakelis, W.S. Levine (Eds.), Handbook of Networked and Embedded Control Systems, Springer (2005) ISBN: 0-8176-3239-5

[7] P. Kleberger, T. Olovsson, and E. Jonsson. Security aspects of the in-vehicle network in the connected car. In Intelligent Vehicles Symposium (IV), 2011 IEEE, pages 528{533, June 2011.

[8] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In Security and Privacy (SP), 2010 IEEE Symposium on, pages 447{462, May 2010.

[9] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In Proceedings of the 20th USENIX Conference on Security, SEC'11, pages 6{6, Berkeley, CA, USA, 2011. USENIX Association.

[10] https://www.blackhat.com/us-15/speakers/Charlie-Miller.html

[11] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of android application security. In Proceedings of the 20th USENIX Conference on Security, SEC'11, pages 21{21, Berkeley, CA, USA, 2011. USENIX Association.

[12] Bosch can, 2004. www.can.bosch.com.

[13] A. Tahat, A. Said, F. Jaouni, and W. Qadamani. Android-based universal vehicle diagnostic and tracking system. In

Consumer Electronics (ISCE), 2012 IEEE 16th International Symposium on, pages 137{143, June 2012.

〔14〕 SAE standard, e/e diagnostic test modes. http://standards.sae.org/j1979 201408/

〔15〕 C. Lin, C.-C. Li, S.-H. Yang, S.-H. Lin, and C.-Y. Lin. Development of on-line diagnostics and real time early warning system for vehicles. In Sensors for Industry Conference, 2005, pages 45-51, Feb 2005.

〔16〕 C. Furmanczyk, D. Nufer, B. Sandona, and B. Ullom. Integrating odb-ii, android, and google app engine to decrease emissions and improve driving habits.

〔17〕 Elm electronics. http://www.elmelectronics.com/

〔18〕 Elm327 data sheet. http://www.elmelectronics.com/DSheets/ELM327DS.pdf

〔19〕 T. Bla?sing, L. Batyuk, A.-D. Schmidt, S. Camtepe, and S. Albayrak. An android application sandbox system for suspicious software detection. In Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on, pages 55{62, Oct 2010.

〔20〕 Google android application developer self-sign. http://developer.android.com/tools/publishing/app-signing.html.

〔21〕 J.-H. Jung, J. Kim, H.-C. Lee, and J. Yi. Repackaging attack on android banking applications and its countermeasures. Wireless Personal Communications, 73(4):1421{1437, 2013.

〔22〕 Patrick, S. 2012. Code Protection in Android. Technical report, University of Bonn.

〔23〕 ProGuard, http://proguard.sourceforge.net/

〔24〕 DexGuard, https://www.guaerdsuare.com/dexguard

〔25〕 DexProtector, https://dexprotector.com/

〔26〕 T. Hoppe and J. Dittman. Sning/replay attacks on can buses: A simulated attack on the electric window lift classied using an adapted cert taxonomy.

〔27〕 T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive can networks | practical examples and selected short-term countermeasures. In Proceedings of the 27th International Conference on Computer Safety, Reliability, and Security, SAFECOMP '08, pages 235{248, Berlin, Heidelberg, 2008. Springer-Verlag.

〔28〕 Taenam Cho, Seung-Hyun Seo, "A Strengthened Android Signature Management Method", KSII Transactions on Internet &Information Systems, Vol. 9 Issue 3, p1210-1230. 21p. 6

〔29〕 J. Xu, S. Li, and T. Zhang. Security analysis and protection based on smali injection for android applications. In X.-h. Sun, W. Qu, I. Stojmenovic, W. Zhou, Z. Li, H. Guo, G. Min, T. Yang, Y. Wu, and L. Liu, editors, Algorithms and Architectures for Parallel Processing, volume 8630 of Lecture Notes in Computer Science, pages 577{586. Springer International Publishing, 2014.

〔30〕 Implementation result. http://52.27.28.182/Attack.html.

〔31〕 S. Woo, H. J. Jo, and D. H. Lee. "A practical wireless attack on the connected car and security protocol for in-vehicle CAN". Intelligent Transportation Systems, IEEE Transactions on, 16(2):993{1006, April 2015.

〔32〕 Se Young Lee,Jin Hyung Park,Moon Chan Park,Jae Hyuk Suk,Dong Hoon Lee. "A Study on Deobfuscation Method of Android and Implementation of Automatic Analysis Tool". Journal of the Korea Institute of Information Security and Cryptology. Vol.25 No.5. pp.1201-1215

〈저자소개〉

이 정 호 (Jung Ho Lee) 학생회원
2014년 2월: 한국외국어대학교 컴퓨터공학과 졸업
2014년 9월~현재: 고려대학교대학교 정보보호대학원 석사과정
〈관심분야〉 In-vehicle Security, 안드로이드 보안

우 사 무 엘 (Samuel Woo) 학생회원
2006년: 단국대학교 컴퓨터공학과 졸업 학사
2006년: ㈜EOTECHNICS근무
2010년: 단국대학교 전자계산학 석사
2011년~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉무선 네트워크 보안, In-Vehicle Security

이 세 영 (Se Young Lee) 학생회원
2014년 2월: 서울시립대학교 수학과 졸업
2016년 2월: 고려대학교 정보보호대학원 석사 졸업
2016년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 모바일 앱 분석, 소프트웨어 난독화

이 동 훈 (Dong Hoon Lee) 종신회원
1983년 8월: 고려대학교 경제학사 졸업
1987년 12월: Oklahoma University 전산학과 석사 졸업
1992년 5월: Oklahoma University 전산학과 박사 졸업
1993년 3월~1997년 2월: 고려대학교 전삭학과 조교수
1997년 3월~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 암호프로토콜, 암호이론, USN이론, 키교환, 익명성 연구, PET 기술