안드로이드 어플리케이션 역공학 방지 기술 분석

박민건*, 오정희**, 류환일*** 김수민****
*인하대학교 컴퓨터정보공학과
**부산대학교 정보컴퓨터공학부
***경찰대학 법학과
**** 한국과학기술원 수리과학과
e-mail : diadld2@naver.com

An analysis on Technics for Preventing Android Reverse Engineering

Min-Gun Pak *, Jeonghui Oh **, Hwahn-il Lyoo ***, Soo Min Kim ****
* Dept. of Computer Science & Information Technology, Inha University
** Dept. of Computer Science & Engineering, Pusan National University
*** Dept. of Law, Korean National Police University
**** Dept. of Mathematical Sciences, Korea Advanced Institute of Science and Technology

요 약

역공학 방지 기법이 적용되지 않은 어플리케이션은 악의적인 역공학에 취약할 수밖에 없다. 악의적인 역공학은 사회적으로 여러가지 손실을 가져온다. 그러므로 역공학 방지 기법을 적용하여 어플리케이션을 보호해야 한다. 역공학 방지 기법은 다양하며, 크게 자바 소스 난독화, Smali 코드 조작, Dex 파일 포맷 조작, 그리고 Zip 파일 포맷 조작 기법으로 나눌 수 있다. 자바 소스 난독화는 코드의 가독성을 떨어트려 안드로이드 어플리케이션의 역공학을 어렵게 하는 기법이다. Smali 코드 조작 기법은 Goto 문 추가, 예외 처리 재귀 기법 등을 통하여 역공학을 막는 기법이다. Dex 파일 포맷 조작 기법에는 클래스명 길이 변경, 헤더 크기 변경 등을 통해 역공학을 어렵게 만드는 기법이다. Zip 파일 포맷 조작 기법은 Zip 파일 포맷에서 헤더 값을 조작하여 마치 암호화된 것처럼 보이도록 만드는 기법이다. 본 논문에서는 이러한 다양한 기법들에 대해서 설명하고 이를 비교·분석한다.

1. 서론

스마트폰 시대가 도래함에 따라 일상의 여러 일들이 스마트폰을 통하여 이루어지고 있다. 스마트폰에는 다양한 어플리케이션들이 설치되어 있으며, 사용자가 원하고 필요로 하는 어플리케이션도 손쉽게 설치하여 이용할 수가 있다. 그래서 사람들은 스마트폰을 통해 다양한 일들을 할 수 있게 되었다. 이제 스마트폰과 사람은 떼어낼 수 없는 관계가 되었고 앞으로도 이 관계는 변함없이 지속될 것이다.

그런데 우리는 이 편리함에 때문에 한 가지 사실을 간과하고 있다. 우리에게 다양한 기능을 제공하는 스마트폰 어플리케이션들은 대게 취약하다. 왜냐하면 역공학 도구들을 통해 손쉽게 취약점을 찾을 수 있기 때문이다. APK 파일을 스마트폰에서 추출하여 역공학 도구에 의해 원본 소스코드와 유사한 자바 소스코드를 분석가에게 보여줄 수 있다.

얻어낸 소스 코드는 어플리케이션의 취약점을 찾는데 중요한 정보를 공격자에게 제공한다. 공격자는 취약할 것 같은 부분에 특정 값이나 함수 등을 추가하여 악의적인 행동을 할 수 있다. 또한, 악성 어플리케이션을 만들어 마켓에 배포할 수도 있다. 그 외에 소스 코드를 무단으로 도용하여 개발자의 지적 재산권도 침해될 수 있다. 이 같은 내용들은 그저 가정이 아니라 현실에서 현저하게 나타나는 일들이다. 그러므로 우리는 어떻게 대처하고 해결할 것인지를 고민해야 한다.

본 논문에서는 안드로이드 어플리케이션 역공학을 방지하기 위한 기법들을 설명한다. 역공학을 방지하기 위한 기법들은 다양하며, 자바 소스 난독화, Smali 코드 조작, Dex 파일 포맷 조작, Zip 파일 포맷 조작으로 나눌 수 있다. 이러한 기법의 간단한 소개와 원리, 적용하는 방법 그리고 적용 시 얻을 수 있는 효과를 설명한다.

본 논문의 구성은 다음과 같다. 2 장에서는 자바 소스 난독화와 Smali 코드 조작을 통해 역공학을 방지하는 방법을 설명한다. 3 장에서는 Dex 파일 포맷 조작, Zip 파일 포맷 조작을 통해 역공학을 방지하는 방법을 설명한다. 마지막으로 4 장에서는 결론을 맺는다.

2. 소스 코드 조작

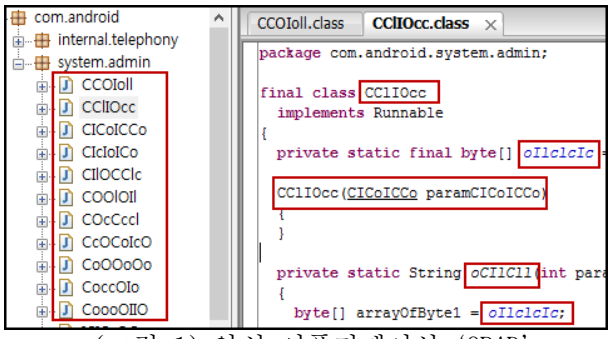
소스 코드 조작 기법에는 자바 소스 난독화 기법과 Smali 코드 조작 기법이 있다.

2.1. 자바 소스 난독화

자바 소스 난독화는 코드의 가독성을 떨어트려 안드로이드 어플리케이션의 역공학을 어렵게 하는 기법이다. 자바 소스 난독화 기법은 크게 레이아웃 난독화, 제어 난독화, 데이터 난독화로 나눌 수 있다[1].

2.1.1. 레이아웃 난독화

레이아웃 난독화는 변수 이름, 클래스 이름, 함수 이름 등을 읽기 어려운 문자로 변경하거나 서로 비슷한 문자들로 바꾸는 기법이다. 이 기법을 적용한 예로 악성 어플리케이션 'OBAD' 를 들 수 있다.



(그림 1) 악성 어플리케이션 'OBAD'

'OBAD'는 I, 1, l, 0, O, o 와 같이 유사한 형태의 문자들로 변수, 클래스, 함수 이름을 작성하여 가독성을 떨어트렸다.

Smali 코드에서 함수나 클래스의 전체 경로를 한자나 특수 문자같이 읽기 어려운 문자로 변환하는 것 역시 레이아웃 난독화에 해당한다.

2.1.2. 제어 난독화

제어 난독화는 제어의 흐름을 바꾸는 기법이다. 제어 난독화는 연산 난독화, 코드 집적 난독화, 재배치 난독화로 나눌 수 있다.

연산 난독화는 실행되지 않는 코드를 삽입하거나, 반복문의 종료 조건을 복잡하게 만들거나, 불필요한 연산을 추가하는 등의 작업을 통해 코드를 읽기 어렵게 만드는 기법이다. 코드 집적 난독화는 관련 없는 여러 코드를 임의의 지점에 집적시키거나 코드를 여러 곳으로 분산시키는 기법이다. 재배치 난독화는 변수나 코드의 위치를 본래의 위치가 아닌 다른 곳에 재배치하는 기법이다.

2.1.3. 데이터 난독화

데이터 난독화는 데이터 값을 사용자가 읽기 어렵게 하는 기법이다. 데이터 난독화의 기법에는 하나의 변수를 두 개 이상의 변수에 나누어 저장하거나, 연관 없는 변수들을 하나의 배열로 관리하는 것 등이 있다.

2.2. Smali 코드 조작

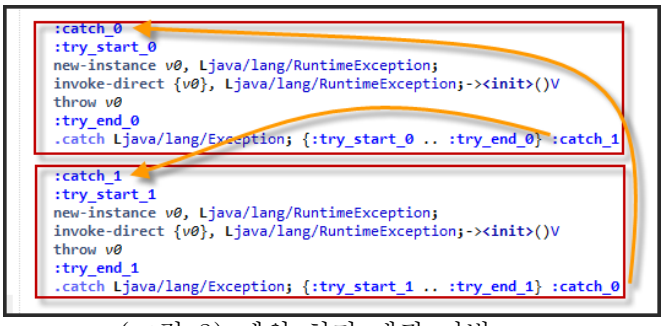
Smali 코드란 Dalvik 에서 사용되는 Dex 포맷의 어셈블리어다. Smali 코드를 조작하는 기법에는 Goto 문 추가, 예외 처리 재귀 기법이 있다.

2.2.1. Goto 문 삽입

프로그램은 일반적으로 위에서 아래로 코드를 해석한다. 하지만 Smali 코드에 Goto 문을 삽입하면 흐름이 비순차적으로 변한다. JD-GUI 나 JAD 와 같은 패턴 매칭 방식으로 디컴파일링하는 도구들은 이러한 경우를 분석하지 못한다[2].

2.2.2. 예외 처리 재귀 기법

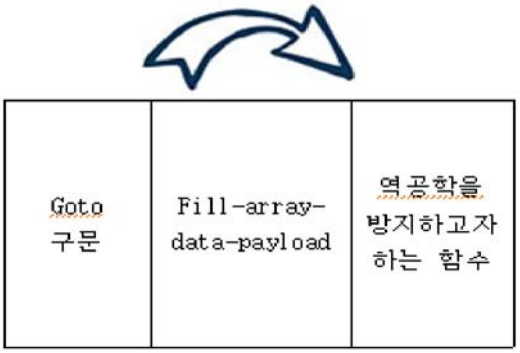
예외 처리 재귀 기법이란 Java 에서 사용하는 예외 처리문인 Throw 와 try/catch 를 응용해 재귀적으로 예외 처리를 하게끔 하는 기법이다. 먼저 사용하지 않는 함수를 만든다. 함수의 내부에는 두 개의 catch 문이 존재하는데, 각각의 catch 문은 다른 catch 문에 예외를 넘겨주는 throw 문을 가지고 있다. 그래서 한 번 예외가 발생하면 두 catch 문은 무한히 서로를 호출하는 상태에 놓인다. 사용하지 않는 함수라서 실행에는 아무런 문제가 없고 역공학도 방지할 수 있다[3].



(그림 2) 예외 처리 재귀 기법

2.2.3. Fill-array-data-payload 의사 명령어 사용

Fill-array-data-payload 는 Patrick Schulz 가 제안한 기법[4]이다. Fill-array-data 는 Dalvik OP 코드 중 하나로, 배열에 데이터를 넣는 기능을 수행한다. Dalvik 은 이 명령어가 가리키는 주소의 코드들을 무의미한 데이터로 인식한다. 그러므로 역공학을 방지하고자 하는 중요 함수를 가리키게 하고 Fill-array-data 명령어 앞에 Goto 문을 넣어 Fill-array-data 명령어가 실제 런타임에서 실행되지 못하게 한다. 그러면 어플리케이션은 정상적으로 실행되고 역공학을 방지할 수 있다.



(그림 3) Fill-array-data-payload 의사 명령어 기법

2.2.4. Smali 코드 조작을 통한 안드로이드 역공학 방지 기술 비교

<표 1>은 Smali 코드 조작 기법들이 각각 어떤 역공학 프로그램의 분석을 방지하는가를 나타낸다.

<표 1> Smali 코드 조작을 통한 안드로이드 역공학 방지 기술 비교

구분	Goto 문 삽입	예외 처리 재귀	Fill-array-data-payload
Apktool	X	X	O
Androguard	X	X	O
Dex2jar	O	O	O
Dexdump	X	X	O
IDA Pro 6.1	X	O	O

O 은 해당 열의 역공학 방지 기술이 해당 행의 역공학 프로그램을 성공적으로 우회한다는 것을 의미한다. 반면 X 는 역공학 프로그램이 어플리케이션을 제대로 분석할 수 있다는 것을 의미한다.

3. 파일 포맷 조작

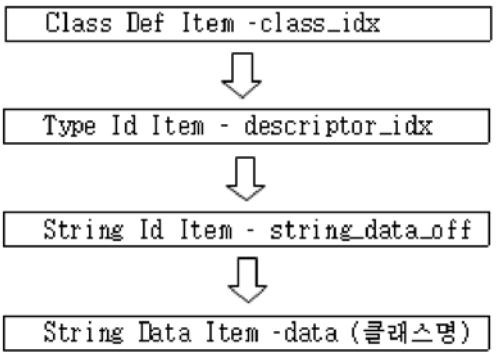
파일 포맷 조작 기법에는 Dex 파일 포맷 조작 기법과 Zip 파일 포맷 조작 기법이 있다.

3.1. Dex 파일 포맷 조작

Dex 파일은 안드로이드 가상 머신이 실행할 수 있는 파일을 말한다. Dex 파일 포맷 조작 기법에는 클래스명 길이 변경, 헤더 크기 변경, Invalid OP 코드 삽입, Invalid offset 참조가 있다.

3.1.1. 클래스명 길이 변경

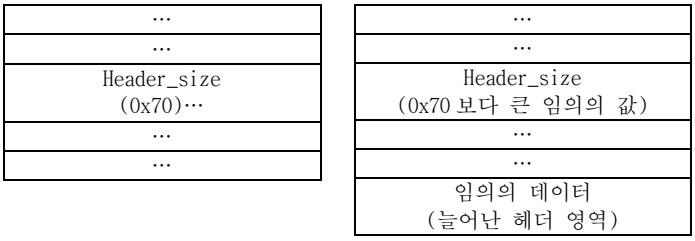
Tim Strazzere 가 제안한 이 기법[5]은 classes.dex 파일에서 클래스 이름에 해당하는 필드를 수정하여 클래스 이름의 길이가 255 자 이상이 되도록 변경하는 기법이다. 이 기법은 Apktool(baksmali)에 한해서 디컴파일링을 방지할 수 있다.



(그림 4) Dex 파일 내 클래스명 위치

3.1.2. 헤더 크기 변경

Dex 파일에서 일반적으로 헤더 크기는 0x70 이고, 헤더의 크기를 가리키는 필드에 있다. 이 헤더 크기의 값을 더 큰 값으로 변경하고, Dex 파일의 헤더 끝 부분에 임의의 바이트 값들을 추가하는 것이다.



(그림 5) 헤더 크기 변경 기법

이 기법[6]의 원리는 간단하지만, 헤더의 크기가 증가하면 그 이후의 모든 테이블이 뒤로 밀려 위치가 변경되므로, 테이블의 Offset 들을 모두 수정해 주어야 하기에 많은 시간이 소요된다. 이 기법 또한 Apktool(baksmali)에 한해서 디컴파일링을 방지할 수 있다.

3.1.3. Invalid OP 코드 삽입

Dalvik 에 정의되지 않은 OP 코드를 삽입하는 기법이다. 단, 코드를 넣을 클래스는 런타임에서 실행되지 않아야 한다. 그러므로 사용하지 않는 클래스 파일을 만들어 함수의 지시어를 Invalid OP 코드로 대체하여 역공학을 방지할 수 있다[7].

이 기법의 예로 사용하지 않는 클래스에 12 01 38 01 03 00 FE FE 의 바이트를 삽입하는 것을 들 수 있다.

<표 2> 12 01 38 01 03 00 FE FE 바이트 코드 의미

12 01	V1(Register) = 0
38 01 03 00	V1 == 0 인 경우, 현재 위치에서 3 바이트(0x0003)만큼 이동
FE FE	실제로 실행이 되지 않는 코드

위 바이트코드는 If 문을 항상 참으로 만들어 3 바이트만큼 점프한다. 그러면 FE FE 는 실제로 실행되지

않아 프로그램의 실행에는 영향을 끼치지 않는다. 하지만 디컴파일링 도구는 FE FE 명령어를 해석할 수 없어 에러가 발생하게 된다.

3.1.4. Invalid Offset 참조

Dex 파일은 String Pool 로 문자열들을 관리하고 Offset 을 이용하여 String 값을 가져온다. Invalid Offset 참조 기법[8]은 이 Offset 이 존재하지 않는 영역을 가리키도록 변경하여 역공학을 방지하는 기법이다.

이 기법의 예로 사용하지 않는 클래스에 12 01 38 01 03 00 1A 00 70 17 의 바이트를 삽입하는 것을 들 수 있다.

<표 3> 12 01 38 01 03 00 1A 00 [2h] 바이트 코드 의미

12 01	V1(Register) = 0
38 01 03 00	V1 == 0 인 경우, 현재 위치에서 3 바이트(0x0003)만큼 이동
1A 00 [2h]	String 을 가리키는 String_id (=[2h])를 V0(=this)에 삽입

표 2 에서 String_id 가 실제 Dex 상에 존재하지 않는 값을 참조하도록 바이트 값을 설정하면 역공학을 방지할 수 있다.

3.1.5. Dex 파일 포맷 조작을 통한 안드로이드 역공학 방지 기술 비교

<표 4>는 Dex 파일 포맷 조작 기법들이 각각 어떤 역공학 프로그램의 분석을 방지하는가를 나타낸다.

<표 4> Dex 파일 포맷 조작을 통한 안드로이드 역공학 방지 기술 비교

구분	클래스명 길이 변경	헤더 크기 변경	Invalid OP 코드	Invalid Offset
Apktool	O	O	X	O
Androguard	X	X	O	O
Dex2jar	X	X	O	O
Dexdump	X	X	O	O
IDA Pro 6.1	X	X	X	X

3.2. Zip 파일 포맷 조작

Zip 파일 포맷에서 헤더 값을 조작하여 마치 암호화된 것처럼 보이도록 만드는 기법이다. ‘General purpose bit flag’ 의 bit 값을 00 00 에서 09 08 로 수정하면 APK 파일 자체에 암호화가 되어있는 것처럼 보여 일반적으로 압축을 풀 수 없다[9].

공격자는 역공학 분석할 때 기본적으로 압축을 풀어 classes.dex 파일을 얻는다. 그렇지만 apk 파일의 압축을 풀 수 없으면 역공학 분석 시도를 하기 힘들어진다

<표 5> 암호화를 위한 Zip 파일 포맷 변경 기법

변경 전	50 4B 01 02 14 00 14 00 **00 00** 08 00…
변경 후	50 4B 01 02 14 00 14 00 **09 08** 08 00…

4. 결론

안드로이드 어플리케이션은 쉽게 역공학이 가능하며, 악의적인 공격자는 이를 이용하여 어플리케이션의 취약점을 탐색하고 소스 코드를 무단으로 도용하고 있다. 이러한 상황에서 역공학 방지 기법을 아는 것은 필수라고 할 수 있다. 본 논문에서는 자바 소스, Smali 코드, Dex 파일 포맷, Zip 파일 포맷과 관련하여 각각 어떠한 역공학 방지 기법들이 있는지, 이러한 기법들이 각각 어떤 역공학 프로그램의 분석을 방지하는지 설명하였다.

안드로이드 버전 4.4 킷캣부터 새로운 런타임인 ART 가 등장하였다. 앞서 설명한 기법들은 모두 ART 에서도 적용 가능하다. 그러므로 본 기법들을 제대로 익히고 숙지하여 더 많은 어플리케이션들의 보안성을 높여야 할 것이다.

참고문헌

[1]가드프리 놀란, “디컴파일링 안드로이드”, pp.165-188, 2012 년 9 월

[2]Emre TINAZTEPE, Doğ an KURT, and Alp GÜLEÇ, “Android OBAD Technical Analysis Paper”, http://www.comodo.com/resources/Android_OBAD_Tech_Reportv3.pdf, pp.4, Jul. 2013.

[3]Tim Strazzere, “Dex Education : Practicing Safe Dex”,http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf, pp.56-66, Jul. 26. 2012.

[4]Patrick Schulz, “Android Bytecode Obfuscation”, http://www.Dexlabs.org/blog/bytecode-obfuscation, Jul. 21. 2012.

[5]Tim Strazzere, “Dex Education : Practicing Safe Dex”,http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf, pp.31, Jul. 26. 2012.

[6]Tim Strazzere, “Dex Education : Practicing Safe Dex”,http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf, pp.69-83, Jul. 26. 2012.

[7]Tim Strazzere, “Dex Education : Practicing Safe Dex”,http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf, pp.33-45, Jul. 26. 2012.

[8]Tim Strazzere, “Dex Education : Practicing Safe Dex”,http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf, pp.46-52, Jul. 26. 2012.

[9]NSHC, “악성코드 분석 보고서”, pp.4-5, 2013 년 10 월 3 일