情報保護學會論文誌
第 23 卷 第 2 號, 2013. 4

안드로이드 기반 SNS 어플리케이션의 코드 변조를 통한 취약점 분석 및 보안 기법 연구*

이 상 호,[1†] 주 다 영[2‡]
[1]연세대학교 글로벌융합공학부, [2]연세대학교 미래융합기술연구원

An analysis of vulnerability and the method to secure on Android SNS applications from alteration of the code segments*

Sang Ho Lee,[1†] Da Young Ju[2‡]
[1]School of Integrated Technology, Yonsei University, [2] Yonsei Institute of Convergence Technology, Yonsei University

요 약

최근 안드로이드 OS 기반에서 SNS(Social Networking Service)를 이용한 게임 및 어플리케이션 시장이 급속도로 성장함에 따라 이에 대한 보안 위협도 증가하고 있다. 특히, 근래 카카오톡 플랫폼이 활성화됨에 따라 이를 이용한 어플리케이션이 대거 출시되고 있으나, 이에 대한 보안 대책은 전무한 실정이다. 이와 같이 늘어나는 안드로이드 OS 기반 SNS 어플리케이션의 변조를 통한 보안 위협에 대응하기 위해서는 어플리케이션 제작 단계에서의 코드 보안 기술이 필요하다. 따라서, 본 논문에서는 안드로이드 OS 기반 SNS 어플리케이션의 실제 변조를 통한 분석을 실시하고 적절한 코드 보안 기술을 제안하고자 한다.

ABSTRACT

According to the rapid growth of the number of SNS(Social Networking Service) applications based on Android OS, the importance of its security is also raised. Especially, many applications using KaKaoTalk platform has been released in these days, and these are top ranked in the relative markets. However, security issues on SNS applications have not been resolved clearly. Therefore, it is crucial to provide means to cope with the security threats posed by code-segment modification in the development stage of Android OS based SNS applications. In this paper, we analyze the security threats by modifying SNS application code segments and suggest effective security techniques.

Keywords: SNS, game, application, android, code security, obfuscate

I. 서 론

최근 스마트폰 게임의 숫자가 폭발적으로 증가함에 따라 이에 따른 어플리케이션의 보안 위협도 급속도로 증가하고 있다. 특히, 최근 국내 대표적인 SNS(Social Networking Service) 업체 중 하나인 ㈜카카오가 카카오톡(KaKaoTalk)을 기존의 메세징 기반의 SNS어플리케이션이 아닌, 플랫폼으로

접수일(2012년 11월 9일), 수정일(2013년 2월 14일), 게재확정일(2013년 3월 8일)
* 본 연구는 지식경제부 및 정보통신산업진흥원의 "IT명품 인재양성사업"의 연구결과로 수행되었음. (NIPA-2013-H0203-13-1002).
† 주저자, sangholee@yonsei.ac.kr
‡ 교신저자, dyju@yonsei.ac.kr

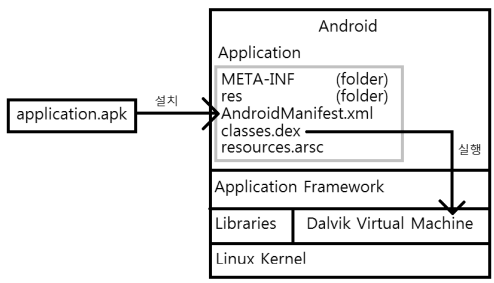
전환하면서 다수의 게임 어플리케이션을 서비스하게 되었다. 이러한 SNS기반의 플랫폼상의 게임들은 상대방과의 정보를 공유하게 되는 SNS어플리케이션의 특성상 조작된 정보를 상대방에게 전달할 수 있으며, 이는 어플리케이션 자체만의 문제가 아닌, '지인' 이라는 믿음을 바탕으로 맺어진 SNS상에서 개인 간의 신뢰를 무너뜨릴 수 있는 사회적인 문제가 될 수도 있겠다. 이렇듯 SNS 어플리케이션의 보안은 소프트웨어측면뿐 아니라 사회공학적인 관점에서도 중요하다〔2, 17〕. 따라서 본 논문에서는 코드 변조 상황에서 어플리케이션이 최대한 보안을 유지할 수 있는 방법을 제안하고자 한다. 이를 위해 II장에서는 어플리케이션 변조 관련 연구들을 소개하고, III장에서는 실제 서비스 되고 있는 가장 최신 버전의 SNS 게임에 탑재되어 있는 유해 어플리케이션 탐지 모듈을 분석하여 변조시키는 과정을 소개함으로써 실제로 어플리케이션의 동작 구조를 변조할 수 있음을 보인 뒤 IV 장에서 본 논문에서 제안하는 분석 기법을 소개한다. 마지막으로 V 장에서는 향후 연구에 대해 논의한다.

II. 관련 연구

자바 가상 머신(Virtual Machine) 위에서 실행되는 안드로이드 어플리케이션은 바이트 코드로 되어 있다.〔3〕 여기에는 원시코드의 정보가 그대로 포함되어 있는데, 이로 인해서 역컴파일이 쉽게 이루어지는 특징이 있다. 이러한 특징은 실제 프로젝트에서 제3자가 작성한 코드에 대한 역분석을 시도해 정보를 얻는데 응용되었으며 이러한 역할 때문에 역컴파일에 대한 연구는 꽤 활발히 진행되었다.〔4, 5〕 이러한 추세에 견주어 이를 이용한 악성 코드에 대한 출현 또한 활발해지고 있으며, 이에 〔1〕과 〔15〕에서는 안드로이드 어플리케이션에서 호출하는 API 목록을 커널 레이어에서 검색하여 악의적인 목적으로 쓰일 가능성이 있는 API들을 분류하는 방식으로 어플리케이션의 유해성을 판단하는 방법을 제안하였으며, 〔16〕에서는 커널 레이어와 프레임워크 레이어 사이에 Aurasium 이라는 패키지를 설치하여 커널 레이어에 호출을 시도하는 API들을 우회하게 만든 후 유해할 가능성이 있는 API 호출을 감지하는 방식을 제안하였다. 하지만 이러한 연구들은 첫째, 어플리케이션의 입장이 아닌 개별 디바이스를 기준으로 한 코드 보안법이며, 둘째, 악의적으로 변조된 어플리케이션을 적발한다 하더라도 이러한 활동을 모니터링 하기 어렵다. 또한 마지막으로, 이러한 연구들은 커널 레이어에 모듈 형태로 장착되거나 해당 어플리케이션이 루트 권한을 가지고 있어야 하는데, 이러한 수정은 디바이스 제조사 외에는 행할 수 없기에 어플리케이션 제작자 입장에선 접근할 수 없는 보안 방식이다. 이러한 연구 결과가 보여주듯, 어플리케이션 제작사의 관점에서 해당 어플리케이션에 대한 역분석 및 변조를 방지하며 동시에 악의적인 시도 자체를 모니터링 할 수 있는 최적의 솔루션이 없었음에도, SNS 어플리케이션이 거의 존재하지 않았던 때에는 클라이언트의 조작된 정보가 상대방에게 알려지지 않았기 때문에 굳이 보안을 강화할 필요성을 느끼지 못했다. 실제로, 2011년에 진행된 안드로이드 OS 기반 멜론 플레이어의 코드 변조를 통한 1분 시간제한 기능 해제에 대한 선행 연구〔7〕에서 멜론 플레이어에는 어떠한 보안도 되어있지 않았지만 소셜 네트워킹 기능이 없어 주변인에게 어떠한 정보도 전파되지 않았기 때문에 1분 시간제한 기능이 해제되어도 사회적으로 큰 문제가 되지 않았다. 하지만 최근 안드로이드 OS 기반의 SNS 어플리케이션이 인기를 끌면서〔2〕 이에 대한 보안 위협도 폭발적으로 증가하였으며, 이들은 점수와 같은 민감한 정보를 다른 사람과 공유하게 되기 때문에〔그림6〕 변조된 코드를 통한 조작된 점수는 상대방으로 하여금 게임에 대한 흥미를 떨어뜨릴 수 있다. 이에 본 논문에서는 실제로 어떤 방법으로 SNS 어플리케이션의 코드가 변조될 수 있는지 시연하고 이러한 악의적인 목적의 코드 변조를 최소화할 수 있는 기법을 소개한다.

III. 어플리케이션 코드변조의 실례

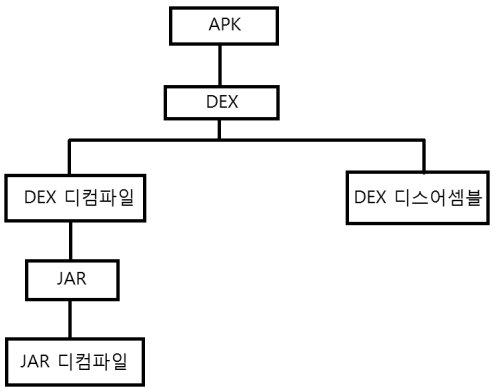
안드로이드 OS 기반 SNS 어플리케이션의 구조와 동작 방법에 대해서는 이미 많은 연구가 진행되었다. 따라서 기본적인 안드로이드 어플리케이션의 동작 원리에 대해서는 〔그림1〕을 참고하도록 한다.〔1〕 본 논



〔그림1〕 안드로이드 어플리케이션 설치 및 실행 과정

문에서는 실제 코드의 분석 및 변조를 위한 디스어셈블 과정을 통해 어떻게 원하는 부분의 코드를 찾고 변조시킬 수 있는지에 대해 중점적으로 소개한다.

3.1 안드로이드 OS 기반 어플리케이션의 역어셈블



〔그림2〕 안드로이드 어플리케이션 분석 로드맵

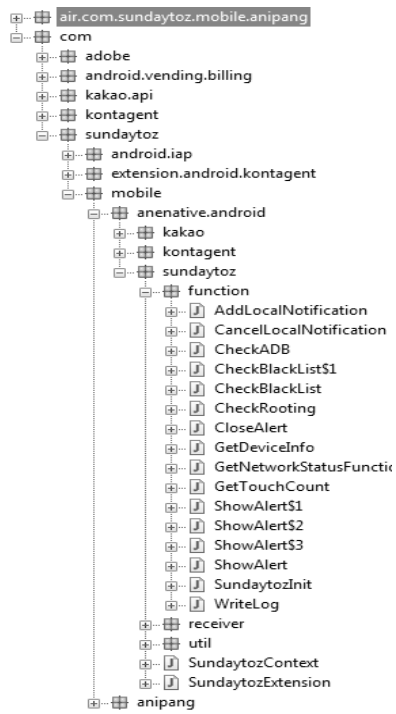
안드로이드에서는 Dalvik 가상 머신을 채용하고 있으며, DEX 포맷으로 변환된 class를 실행한다. 하지만, dex 파일 자체는 바이너리 포맷으로 되어 있어 분석이 어렵기 때문에 실행파일인 Classes.dex 파일의 역어셈블을 통해 분석 및 변조를 실시한다.〔1〕

실제 코드의 분석 및 변조를 위해 자주 쓰는 방법으로는, 〔그림2〕에서 보이는 JAR 디컴파일러와 DEX 디스어셈블러를 병행해서 이용하는 방법이 있다. JAR 디컴파일러는 실제 코드의 구조 및 알고리즘을 분석할 때 유용하게 사용되며, 분석이 끝나고 변조 과정을 시작하게 되면 어셈블리어의 일종인 smali〔10〕 언어로 변환되는 DEX 디스어셈블러를 이용하여 실제 어플리케이션 프로젝트의 코드를 변조하게 된다. JAR 디컴파일러를 이용하여 번역된 코드를 실제 java 컴파일러로 재컴파일 할 수 없는 이유는 번역된 java 형태의 코드가 실제 java 코드와 문법과 구조 측면에서 차이가 발생하기 때문이며, 번역된 java코드가 실제 java 코드와 차이가 발생하는 이유는, DEX 포맷은 여러 가지 방법을 통하여 최적화 작업이 완료된 바이너리 포맷이기 때문이다. 그러므로 재컴파일을 위해서는 어셈블리어 수준에서의 접근이 필요하며, 이 과정에서 DEX 디스어셈블러를 이용하게 된다. 이 논문에서는 DEX 파일을 jar파일로 변환해주는 dex2jar〔11〕을 사용해 JAR파일 형태로 변환시킨 후 JAVA Decompiler〔8〕 라는 JAR 디컴파일러를 이용하여 어플리케이션의 코드를 분석하게 되며, 분석된 정보를 바탕으로 apktool〔9〕라는 툴(tool)을 이용하여 smali 언어로 된 실제 코드의 변조 작업을 수행하게 된다.

3.2 실제 SNS 어플리케이션의 변조 과정

- 선데이토즈 "애니팡(AniPang)"의 예를 중심으로

안드로이드 apk 파일을 unpack 하게 되면 AndroidManifast.xml 파일이 생성된다. 이 파일은 어플리케이션에 관한 설명 및 실행권한 등의 정보를 지니는 정보파일인데, 여기에 어플리케이션의 최초 실행 지점에 대한 Activity 또한 명시되어 있다. 따라서 공격자는 이를 통해 어플리케이션의 시작점을 알게 된다. 또한, 이를 통해 프로젝트의 흐름을 분석하게 되면 해당 어플리케이션의 동작 구조를 알 수 있다. 최근 가장 빨리 성장하고 있는 SNS 게임 중 하나




```
Object localObject10 = localHashMap.put("checkADB", localCheckADB);
CheckRooting localCheckRooting = new CheckRooting();
Object localObject11 = localHashMap.put("checkRooting", localCheckRooting);
CheckBlackList localCheckBlackList = new CheckBlackList();
Object localObject12 = localHashMap.put("checkBlackList", localCheckBlackList);
GetTouchCount localGetTouchCount = new GetTouchCount();
Object localObject13 = localHashMap.put("getTouchCount", localGetTouchCount);
```

〔그림 3〕 애니팡 프로젝트 구성 및 모듈 추가 루틴

인 애니팡[12] 또한 같은 구조로 되어 있으며, Android Manifast.xml 파일에 이와 같은 Activity에 대한 내용이 명시되어 있는데, 이는 모든 프로젝트가 가진 속성이지만 실제 공격자는 이 정보를 가지고 코드의 진입점을 알아낸다. [그림3]에서 볼 수 있듯이 위에서 언급한 Java Decompiler를 이용하여 진입점인 MainApp 클래스를 시작으로 분석해 보면, 애니팡의 보안 모듈은 어플리케이션이 처음 시작할 때 FREFunction의 인터페이스 형태로 컴포넌트 화 되어있는 모듈들을 메시지 디스패칭(dispatching) 방식으로 각각 실행하는 구조로 되어 있는 것을 알 수 있으며 공격자가 찾고자 하는 부분인 유해한 어플리케이션 탐지 모듈뿐만 아니라, 루팅(rooting)확인까지도 이 패키지 안에서 수행하는 것을 확인할 수 있다. 따라서 [그림3]의 패키지(package) 중 CheckRooting 클래스는 루팅이 되어 있는지를 체크하는 부분이고, CheckBlackList 클래스는 유해한 어플리케이션이 있는지를 체크하는 클래스이다. 본 논문에서는 유해한 어플리케이션을 탐지하는 기능을 무력화 시키는데 중점을 두므로 CheckBlackList 클래스를 변조시키도록 한다. 변조시킬 타겟 클래스를 찾으면, smali 언어로 디스어셈블 시키는 apktool을 사용하여 어플리케이션을 디스어셈블 시킨 후 CheckBlackList.smali 파일을 열어 [그림 4]와 같이 마지막 리턴 값을 무조건 0으로 반환하게 변형시킨 후, 재컴파일 하도록 한다. 이는 악성 어플리케이션을 찾지 못하면 0을 리턴하게 되어 있는 CheckBlackList 클래스의 정상적인 로직을 응용한 것이다.

```
const/4 v5, 0x0

invoke-virtual {v4, v0, v5}, Landro
:try_end_0
.catch Ljava/lang/Exception; {:try_

move-result-object v0

.line 135
if-eqz v0, :cond_0

.line 136
add-int/lit16 v0, v2, 0x7d0

.line 144
:goto_1

return v0

.line 138
:catch_0
move-exception v0

.line 129
:cond_0
add-int/lit8 v0, v2, 0x1

move v2, v0

goto :goto_0

:cond_1
move v0, v1
```

```
const/4 v5, 0x0

invoke-virtual {v4, v0, v5},
:try_end_0
.catch Ljava/lang/Exception;

move-result-object v0

.line 135
if-eqz v0, :cond_0

.line 136
add-int/lit16 v0, v2, 0x7d0

.line 144
:goto_1
const/4 v0, 0x0

return v0

.line 138
:catch_0
move-exception v0

.line 129
:cond_0
add-int/lit8 v0, v2, 0x1

move v2, v0

goto :goto_0

:cond_1
move v0, v1
```

[그림 4] 유해 어플리케이션 탐지 루틴 - 변경 전(왼쪽)과 변경 후(오른쪽)

3.3 변형된 SNS 어플리케이션의 쓰임

최근 어플리케이션들은 크로스 플랫폼화 되는 추세에 따라 다른 기반 플랫폼 위에서 스트리밍으로 콘텐츠를 다운받아 실행하는 형태를 취하는 경우가 많은데, 애니팡의 경우에도 플래시로 작성된 콘텐츠를 게임 시작 시 다운로드 받아 adobe AIR[6]로 실행하는 구조로 되어 있다. 하지만, 보안 모듈과 같은 부분은 운영체제 및 Main Activity에 의존적이기 때문에 이런 방법을 쓸 수 없고, 실제 애니팡에서도 이러한 이유 때문에 루팅 검사 및 유해한 어플리케이션 탐색 부분에 있어서는 이와 같은 형태를 취하고 있다. 이러한 경우 단 한 번의 역분석으로 모든 구조를 알 수 있기 때문에 코드에 대한 보안을 신경 써야 했음에도 불구하고, 최근 발매되는 거의 모든 SNS 어플리케이션은 이러한 문제에 대해 관심 가지지 않고 있는 경우가 많으며, 공격자는 이러한 변조를 통해 유해한 기능을 방지하는 기능을 무력화 시킨 후 2차적으로 매크로와 같은 유해한 어플리케이션을 결합하여 점수 조작과 같은 이득을 취하게 된다.

SNS 어플리케이션의 특성상 이렇게 조작된 정보는 상대방에게 그대로 전달될 수 있는데, 실제 본 연구에서 애니팡의 보안 모듈을 무력화 시킨 후에 행한 실험도 Mobizen[14] 이란 휴대폰 원격 접속 프로그램을 설치한 후에 PC에서 원격으로 애니팡을 하면서 실시간 화면 캡처(screen capture)를 하여 이미지 분석을 실시 한 후 적정 알고리즘을 이용하여 매크로 프로그램을 돌리는 방식의 실험을 실시하였다. 만약 변조되지 않은 애니팡 어플리케이션이었다면 [그림5]

[그림 5] 정상적인 애니팡에서의 유해 어플리케이션 감지 메시지.

〔그림 6〕 본인(좌측) 및 상대방(우측)의 SNS 어플리케이션에서 보여지는 조작된 점수 및 순위.

와 같은 오류 메시지와 함께 종료되어야 하는 실험환경이었으나 정상적으로 게임이 실행되었고, 〔그림6〕과 같이 모비즌과 매크로를 이용하여 조작된 점수를 실제 플레이어의 정상 점수와 함께 순위에 올릴 수 있었다.

IV. 안드로이드 OS 기반 SNS 어플리케이션의 코드 변조를 방지하기 위한 기법

안드로이드 OS 기반 어플리케이션의 역분석은 윈도우 기반 실행파일(EXE)의 그것과 비교했을 때 훨씬 더 쉽다고 할 수 있으며, 그 중 가장 큰 이유는 개발단계에서 작성된 클래스, 함수 및 변수가 역어셈블 시에도 그대로 나타나기 때문이다. 또한, 클라이언트에 다운로드 된 어플리케이션 파일은 몇 번이고 변조가 가능하기 때문에 언젠가는 공격자가 원하는 방향으로 어플리케이션을 수정할 수 있다. 따라서 본 논문에서는 코드의 난독화(code obfuscation) 및 비정상 사용자의 프로파일링을 통해 코드 변조를 어렵게 만드는 과정을 소개하며 이 두 방법을 융합하여 코드의 변조 및 공격자에 대해 적극적으로 대처할 수 있는 방안을 제안하고자 한다.

4.1 코드 난독화 기술

코드 난독화 기술은 가상 머신(Virtual Machine)이 소개된 직후 이래로 이슈화 된 기술이다. 이 기술은 역분석을 시행하면 모든 클래스, 함수 및 변수가 그대로 나타나기 때문에 공격자는 그 이름을 가지고 기능을 추측하기 쉽다는 가정으로 소개된 기술이며, 의미가 명확하여 기능 추정이 가능한 이름을 의미가 부여되지 않은 문자열 형태로 바꾸어 주는 것이 핵심이다. Java 뿐만 아니라 가상머신을 채용하여 작동하는 모든 어플리케이션에는 코드 난독화 기술 및 라이브러리가 존재한다. 최근에 소개되는 난독기는 이와 더불어 코드 최적화 기능까지 지원하는 경우가 많다.〔3, 18〕 시중에 존재하는 안드로이드용 코드 난독기는 대부분 위의 소개된 기능을 충족시키며, 가장 대표적인 안드로이드용 난독기는 ProGuard〔13〕 라는 것이 있다. 사용법은 간단하며, GPL 라이센스 채용으로 라이센싱을 위한 비용도 들지 않는 매우 유용한 라이브러리이다.

난독화 작업에 요구되는 기능은 〔표 1〕과 같으며, 최근에 소개되는 기술은 이러한 코드의 난독화와는 별개로 디버거 사용 제한〔13〕과 같은 기능이 추가되어 코드를 추적하면서 분석하는 기능을 제한하는 방향으로 발전하고 있다.

〔표 1〕 난독화 기술 요약

타입	기능	설명
레이아웃		* 변수 및 함수 이름 난독화
컨트롤	계산	* 동작하지 않거나 완전하지 않은 코드를 삽입 * 루프 비교 조건을 늘림 * 최적화 가능한 코드를 불가능하게 변형 * 결과에는 영향을 미치지 않는 무작위 사칙연산을 삽입 * 주석 및 의미 추측 가능한 프로그래밍 용어를 없앰 * 코드의 병렬화
	병합	* 메서드를 끼워 넣음 * 메서드 복제 * 반복문 변형
	순서	* 구문의 순서를 재배치 * 반복문의 순서를 재배치 * 계산식의 순서를 재배치
데이터	저장 및 인코딩	* 인코딩 기법 변경 * 변수들 쪼개기 * static 데이터를 순차적 데이터로 변환
	병합	* 같은 값으로 여러 개 존재하는 상수 및 변수들을 하나로 병합 * 클래스 최적화 * 가짜 클래스 삽입 * 배열 나누기 * 배열 합치기 * 배열 2등분 하기 * 배열을 모두 1차원으로 변경
	순서	* 인스턴스 변수와 메서드의 순서를 재배치함 * 배열 재배치

4.1.1 레이아웃

레이아웃 타입의 난독화 작업에서는 클래스, 메서드, 변수 및 로직의 구조를 바꾸지 않고 이름만 변경하는 작업이 수행된다. 의미가 부여된 변수 및 함수 이름을 a, b 와 같은 의미가 부여되지 않은 이름으로 변경하는 작업은 공격자로 하여금 해당 레이아웃의 의미를 이름을 통해 추측할 수 없게 만들고, 이에 공격자는 해당 모듈의 구조와 로직을 보고 분석해야 하기 때문에 분석 작업을 훨씬 어렵게 만들 수 있다. 하지만, 로직의 정형적인 형태를 분석하게 되면 레이아웃과 상관없이 빠른 분석이 가능하기 때문에 컨트롤 및 데이터 타입의 난독화 작업을 통해 같은 결과를 내는 난독화된 코드로 변형하는 작업을 거치는 것이 일반적이다.

4.1.2 컨트롤

컨트롤 타입의 난독화 작업 단계에서는 실제 코드의 로직에 대한 변형 작업이 수행되며, 의미가 없는 코드를 삽입하던가, 간단한 메서드는 구문을 상위 메서드에 붙인 후 제거하는 방법, 루프의 비교 조건을 난해하게 늘리는 작업, 최적화가 불가능하도록 로직을 변형하는 작업, 결과에는 영향을 미치지 않는 무작위 사칙연산을 삽입하는 작업, 반복문의 형태를 변형하는 작업, 로직의 순서를 영향을 미치지 않는 한도 내에서 변형하는 작업, 반복문의 순서를 재배치하는 작업 등이 포함된다. 이러한 형태의 난독화 작업은 결과 값이 난독화 전과 정확히 일치해야 하기 때문에 미리 정의된 형태의 로직을 비교한 후 분석이 완료된 경우에만 변형 대상으로 선정하며, 이외의 경우에는 컨트롤 타입의 난독화 작업은 수행하지 않는 것이 일반적이다. 컨트롤 타입의 난독화 작업의 장점이라면, 로직 자체가 변형되기 때문에 공격자의 사전 지식과 난독화 전의 정형적인 구문을 대조하여 이해하는 작업이 어려워지며, 레이아웃의 변형과 함께 난독화 작업이 이루어질 경우 공격자가 해당 구문에 대한 기능을 이해하는 것이 훨씬 어려워진다. 또한, 컨트롤 타입의 난독화 작업은 코드의 최적화와 밀접한 관련이 있어 속도 및 어플리케이션 용량을 개선하는 역할을 한다.

4.1.3 데이터

데이터 타입의 난독화 단계에서는 실제 코드 외에 존재하는 상수, 변수 및 클래스의 형태로 존재하는 데이터에 대한 난독화 작업을 수행하며, 대표적인 기법으로는 변수를 쪼갠 후 병합하는 과정을 거치도록 하거나, 가짜 클래스를 삽입하여 혼란을 주는 기법, 배열을 나누거나 병합하는 과정을 통해 난독화 하는 기법, 다차원 배열을 모두 1차원으로 변환하여 다차원 정보에 대한 이해를 어렵게 만드는 기법, 같은 값으로 존재하는 여러 가지 상수 및 변수들을 한 개로 병합하여 난독화 시키는 방법 등이 있다. 이와 같은 작업은 레이아웃 난독화 작업이 완료된 후 컨트롤 난독화 과정과 같이 시행되며, 인코딩 및 디코딩을 위한 암호문이나 인증을 위한 토큰과 같은 데이터에 대해서 난독화 과정이 시행되면 이해하기 어려운 형태의 문장으로 분리 및 병합되기 때문에 중요한 기능을 한다고 말할 수 있다.

〔그림7〕과 〔그림8〕은 Bubble Sort 함수가 원시 코드와 난독화된 코드에서 어떤 차이점이 있는지 보여준다. 위 그림에서 보듯이, SNS 어플리케이션에서 원시 코드가 난독화 되면 클래스명, 함수명 및 변수명이 의미 없는 문자열로 변하기 때문에 공격자가 해당 함수의 기능을 이름을 통해 추측하고 정형화된 코드를 통해 공격자가 알고리즘 추측을 수행하는 행위를 난해하게 만들 수 있다.〔3〕

4.2 비정상적 사용자의 프로파일링

코드를 분석 및 변조하기 위해서 공격자가 처음 확인 하는 일은 타겟 기능이 제대로 동작하며, 어떤 형식으로 작동하는가이다. 그러므로 공격자는 코드를 무력화시키기 전에 타겟 기능이 정상적으로 동작하는지 확인하는 과정을 거치게 되며, 어떤 방식으로 어떻게 작동하는지를 추측하여 실제 분석 및 변조에 들어가게 된다. 또한, 보통 프로그래밍이 그러하듯 실제 변조 과정에서도 많은 시행착오를 거치게 되는데, 이러한 경우 보통 로직을 잘못 파악하거나 함수 호출의 구조를 잘못 파악하게 되어 다른 곳을 수정하게 되는 경우가 많다. 이러한 경우 결국 타겟 기능은 무리 없이 동작하게 된다. 비정상적 사용자의 프로파일링 기법은 코드 변조를 위해 어플리케이션을 실행하는 공격자의 경우 정상적인 사용자와는 다르게 허용될 수 없을 만큼 해당 모듈을 실행시킨다는 전제로 제안하는 기법이다. 위에서 실제로 코드 변조 시연을 보였던 애니팡의 경우, 보안 모듈에 대한 코드 변조를 성공적으로 수행하기 위해서 약 15번 이상의 유해 어플리케이션 발견

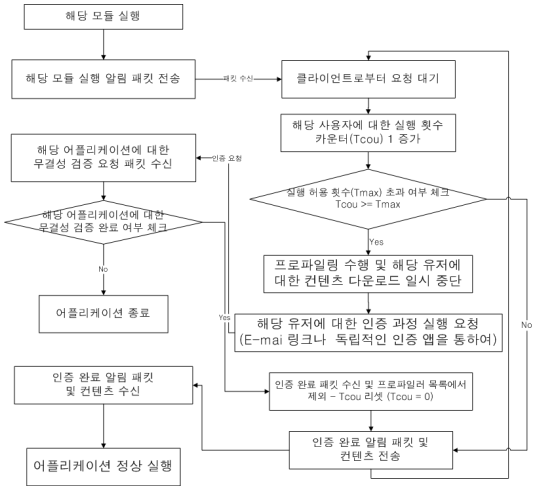
메시지〔그림5〕을 발생시켰다. 〔그림9〕에서 볼 수 있듯이, 적절한 횟수만큼 해당 기능을 실행시킬 수 있게 만들고 이를 초과하면 서버에 해당 사용자에 대한 정보를 전송하여 서버에서 관리하게 만들 수 있다. 서버에서는 프로파일링 된 정보를 이용하여 해당 사용자에 대한 전반적인 정보를 로그 분석 형태로 실시할 수 있으며, 관리자가 분석을 실시하기까지 해당 사용자의 어플리케이션 사용을 일시 중지시킴으로서 악의적인

```
final static int SIZE = 100;

private void SortArray(int array[])
{
    int[] sorted_array = new int[SIZE];

    for(int i=0; i<SIZE; i++)
        sorted_array[i] = array[i];

    for(int i=0; i<SIZE-1; i++)
    {
        for(int j=i+1; j<SIZE; j++)
        {
            if(sorted_array[i] < sorted_array[j])
            {
                int temp = sorted_array[i];
                sorted_array[i] = sorted_array[j];
                sorted_array[j] = temp;
            }
        }
    }

}
```

〔그림 7〕 BubbleSort 의 원시 코드

```
private void a(int[] paramArrayOfInt)
{
  int[] arrayOfInt = new int[100];
  int i = 0;
  int j;
  while (true)
  {
    if (i >= 100)
    {
      j = 0;
      if (j < 99)
        break;
      return;
    }
    int k = paramArrayOfInt[i];
    arrayOfInt[i] = k;
    i += 1;
  }
  int m = j + 1;
  while (true)
  {
    if (m >= 100)
    {
      j += 1;
      break;
    }
    int n = arrayOfInt[j];
    int i1 = arrayOfInt[m];
    if (n < i1)
    {
      int i2 = arrayOfInt[j];
      int i3 = arrayOfInt[m];
      arrayOfInt[j] = i3;
      arrayOfInt[m] = i2;
    }
    m += 1;
  }
}
```

〔그림 8〕 ProGuard를 사용하여 〔그림7〕의 원시코드를 난독화하여 JavaDecompiler로 분석한 모습.

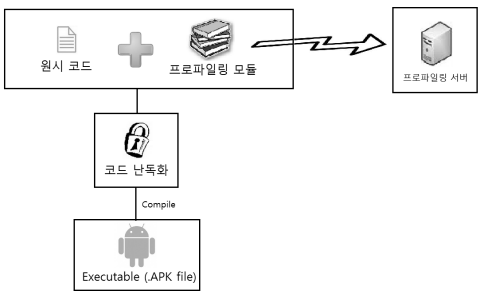


〔그림 9〕 클라이언트(좌측)와 서버(우측)간의 프로파일링 검증 및 해제 프로세스.

목적의 사용자가 더 이상의 변조된 코드에 대한 정상 동작 여부를 확인할 수 없게 하는 것이 가능하다. 그러나 이 방법을 이용하여 프로파일링을 정상적으로 수행하기 위해서는 중요한 조건이 필요한데, 실제 프로파일링이 되어 악의적인 목적의 사용자라고 분류 되었지만 실제로는 정상적인 사용자일 수가 있기 때문이다. 따라서 프로파일링이 된 사용자라 할지라도 정상적으로 게임을 이용할 수 있는 방법을 마련해 놓아야 한다. 이는 이메일 링크를 통해서 재활성화(reactivation) 프로세스를 진행하거나, 혹은 어플리케이션의 변조 여부를 분석하는 독립적인 어플리케이션을 설치하게 하여 검증하도록 할 수도 있겠다. 하지만, 악의적인 사용자가 프로파일링 관련 코드를 변조시켜서 리포트 하지 않게 만든 후에 원하는 부분에 대한 코드 변조를 실시할 수 있으므로, 이런 경우 코드 변조 여부를 알아내는 부분이 변조되어 프로파일링도 이루어지지 않게 된다. 그러므로 우리는 이러한 공격 패턴의 피해를 최소화하기 위해 코드 난독화 기술과 비정상적 사용자의 프로파일링을 융합시켜 좀 더 강력한 보안을 위한 하이브리드(hybrid) 모델을 제안하고자 한다.

4.3 강건한 하이브리드 모델

실제 코드 변조를 위해서는 3.2에서 설명했듯이 AndroidManifast.xml 파일을 통해 시작점을 파악하고, 이를 통해 전체적인 어플리케이션의 구조를 잡아나가는 과정을 거치게 된다. 만약 난독화 되지 않은



〔그림 10〕 제안하는 하이브리드 모델 적용 과정

코드라면, 이 과정을 통해서 비정상적 사용자에 대한 프로파일링을 수행하는 부분의 코드를 쉽게 발견할 수 있을 것이다. 따라서 하이브리드 모델에서는 비정상적 사용자의 프로파일링에 코드 난독화 과정을 추가하여 〔그림10〕과 같이 실제 프로파일링 코드에 접근하게 어렵게 함과 동시에 다른 코드에 대한 난독화 작업 또한 수행하는 방법을 제안한다. 실제로, 난독화 된 코드를 역분석을 하는 과정은 쉽지 않으며, 만약 역분석을 진행하여 코드를 변조한다 하더라도 원하는 부분이 아닐 가능성을 높일 수 있고, 이는 비정상적 사용자에 대한 프로파일링을 실시할 때 해당 모듈의 최대 실행 허용 횟수를 통계의 오차 범위 안에서 늘릴 수 있게 해준다. 그러므로 이 방법을 통해서 얻을 수 있는 장점은 난독화를 통해 프로파일링 모듈을 변조하는 작업을 어렵게 만드는 것이며, 이는 앞에서 설명한 변조 방어를 좀 더 강건하게 할 수 있겠다.

V. 결론

본 논문에서 제안하는 안드로이드 OS 기반 SNS 어플리케이션의 코드 변조를 통한 취약점 분석 및 코드 보안 기법은 코드 난독화와 비정상적 사용자의 프로파일링을 통해 악의적인 목적의 코드 변조의 성공 확률을 낮춤과 동시에 해당 사용자의 프로파일링을 통해 과거에 일어난 비정상적 행위를 적발하고 미래에 일어날 악의적인 코드 변조를 미연에 차단하는 수단으로 유용하게 활용될 수 있다. 비정상적 사용자의 프로파일링 기법은 어느 정도 이상 해당 모듈을 활성화시켜야 코드 변조를 위한 악의적인 사용자로 판단할 것인지에 대한 신뢰성 있는 통계적 데이터 검출이 필요하며, 어플리케이션의 장르 별로 실제 접속자 데이터 및 검출된 데이터의 분석을 통해 프로파일링 기법이 가지는 한계를 극복하는 것이 필수적이다. 또한, 변조된 인증 메시지를 방지하기 위한 무결성을 보장하는 방법 및 프로파일링 모듈 서버가 안전한지 여부를 검증하는 작업이 필요하며, 프로파일링 시스템의 적합성을 판별하기 위해 향후 연구 및 실제 적용을 통한 검증이 요구된다. 제안된 기법의 특성상 콘텐츠를 실시간으로 다운로드 해야 하며 이는 통신 네트워크상에서는 데이터 요금 부과의 원인이 될 수 있으므로 3D 모델과 같은 대용량 데이터가 포함된 고용량/고사양을 요구하는 콘텐츠에 대해서는 제안된 방법을 사용하는데 주의를 기울이는 것이 필요하겠다.

안드로이드 어플리케이션의 코드 영역에서의 가장 큰 보안 위협은 클라이언트에 설치되는 순간부터 해당 어플리케이션을 무제한으로 역분석 할 수 있고 이에 대한 검증 과정이 존재하지 않는다는 것에 있다. 따라서 본 논문에서 제안한 코드 난독화 및 비정상적 사용자의 프로파일링 기법은 악의적인 목적으로 코드를 변조하고자 하는 사용자를 프로파일링하고 이러한 사용자에 대해서는 해당 어플리케이션의 규정에 의거하여 대처할 수 있게 만드는 기반 기술로서 활용할 수 있을 것으로 기대하며, 차후 여러 예를 통해 제안하는 방법을 좀 더 견고하게 발전시키고자 한다.

참고문헌

〔1〕 심원태, 김종명, 류재철, 노봉남, "안드로이드 앱 악성행위 탐지를 위한 분석 기법 연구," 정보보호학회논문지, 21(1), pp. 213-219, 2011년 2월.

〔2〕 김윤영, "국내 소셜 게임 시장 전망," 한국컴퓨터게임학회논문지, 24(1), pp.113-123, 2011년 3월

〔3〕 이병용, 최용수, "Obfuscation 기술의 현황 및 분석과 향후 개발 방향," 보안공학연구논문지, 5(3), pp.219-228 ,2008년 6월

〔4〕 Nomair A.Naeem, Laurie Hendren, "Programmer-friendly Decompiled Java," Proceedings of the 14th IEEE International Conference on Program Comprehension, pp.327-336, Jun. 2006

〔5〕 Jerome Miecznikowski, Laurie Hendren, "Decompiling Java Byte code:Problems, Traps and Pitfalls," Proceedings of the 11th International Conference on Compiler Construction, pp 111- 127, Apr. 2002

〔6〕 Adobe, http://www.adobe.com/kr/ products/air.html

〔7〕 Melon player crack - proof video, http://mu2.nayana.kr/~sangholee/xe/index.php?document_srl=423
〔8〕 Java, http://java.decompiler.free.fr
〔9〕 Android-apktool,http://code.google.com/p/android-apktool/
〔10〕 smali,http://code.google.com/p/smali
〔11〕 dex2jar,http://code.google.com/p/dex2jar
〔12〕 애니팡(Anypang),http://web-m-anipang.sundaytoz. com
〔13〕 ProGuard,http://proguard.sourceforge.net/
〔14〕 모비즌(Mobizen), http://www.mobizen.com/
〔15〕 William Enck, "A study of Android Application Security," USENIX Security Symposium, Aug. 2011
〔16〕 Rabin Xu, "Aurasium: Pracitical Policy Enforcement for Android Applications," USENIX Security Symposium, Aug. 2012
〔17〕 Md Sazzadur Rahman, "Efficient and Scalable Socware Detection in Online Social Networks," USENIX Security Symposium, Aug. 2012
〔18〕 Teodoro Cipresso, "An introduction to software reverse engineering," Springer, 2010.

〈著者紹介〉

이 상 호 (Sang Ho Lee) 정회원
2007년 2월: 한국게임과학고등학교 졸업
2012년 4월: Digipen Institute of Technology 컴퓨터 공학과 학사
2012년 8월~현재: 연세대학교 글로벌융합공학부 석박통합과정
〈관심분야〉 코드보안, 소프트웨어아키텍쳐, 게임엔진, 컴퓨터그래픽스

주 다 영 (Da Young Ju) 정회원
2000년 2월: 홍익대학교 학사
2002년 8월: 서강대학교 미디어공학과 석사
2008년 8월: 영국 University of the Arts London, Digital Arts 석사
2011년 2월: 서강대학교 미디어공학과 박사
2012년~현재: 연세대학교 미래융합기술연구원 조교수
〈관심분야〉 정보보호, 모바일 소프트웨어, 소셜네트워킹서비스, 컴퓨터그래픽스, 게임