Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.23, NO.6, Dec. 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.6.1049

안드로이드 스마트폰 암호 사용 앱 보안 분석 및 대응*

박 상 호,[1†] 김 현 진,[2] 권 태 경[3‡]
[1]세종대학교, [2]ETRI 부설연구소, [3]연세대학교

On Security of Android Smartphone Apps Employing Cryptography*

Sang-ho Park,[1†] Hyeonjin Kim,[2] and Taekyoung Kwon[3‡]
[1]Sejong University, [2]Attached Instituteof ETRI, [3]Yonsei University

요 약

스마트폰은 사용자가 필요한 응용프로그램(이하 앱)을 선택하여 설치할 수 있어서 그 활용도가 점차 확대되고 있다. 앱에 따라서 계정정보, 금융정보 등 민감한 정보가 저장되며, 반드시 안전하게 암호화되어야 한다. 안드로이드는 리눅스 커널 기반으로 메모리와 스토리지에 대한 보안을 수행하지만 루팅 공격으로 인하여 보안이 무력화될 수 있다. 본 논문에서는 안드로이드에 사용되는 보안 기법을 분석하여 문제점을 지적하였다. 문제점을 바탕으로 상용 앱 분석을 통해 취약점을 보인 후, 대응 방안을 제시한다.

ABSTRACT

Smartphones are rapidly growing because of easy installation of the apps (application software) that users actually want. There are increasingly many apps that require cryptographic suites to be installed, for instance, for protecting account and financial data. Android platform provides protection mechanisms for memory and storage based on Linux kernel, but they are vulnerable to rooting attacks. In this paper, we analyze security mechanisms of Android platform and point out security problems. We show the security vulnerabilities of several commercial apps and suggest appropriate countermeasures.

Keywords: Android, Smartphone, Reverse-Engineering, Security

I. 서 론

스마트폰 시장의 성장은 멈출 줄 모르며, 2013년 2/4분기 기준, 전 세계에 약 435백만대의 스마트폰이 판매되었고 이 중 약 80%가 안드로이드 플랫폼을 탑재하고 있다[1]. 안드로이드 플랫폼은 2008년 10월 첫 단말기를 출시한 이후, 개방성을 무기로 빠르게 성장하여 2011년 3/4분기에는 전체 스마트폰 OS 시장의 절반 이상을 차지하였다[2]. 우리나라는 특히 안드로이드 점유율이 높아서 약 90%에 이르는 것으로 조사된다[3]. 높아진 점유율만큼 다양한 앱이 출시되고 있으며, 이 중 일부는 중요한 정보들을 저장한다. 많은 앱이 온라인 서비스에 접속하기 위한 계정 정보를 저장하며, 계좌 및 신용카드 등 금융 정보를 저장하기도 한다. 개인 정보 보호를 위하여 앱에서 암호화를 적용하고 안드로이드가 보안 기능을 제공하나, 실제로 안전한지는 알 수 없다. 안드로이드 보급률이 높은 만큼, 앱과 데이터의 안전성을 파악하고 취약점을 분석하여 대책을 마련해야 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구에 대해 소개하고, 3장에서 안드로이드 보안 요소

접수일(2013년 10월 17일), 게재확정일(2013년 10월 23일)
* 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음(NIPA-2013-H0301-13-1003). 또한, 2012년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2012R1A1B3000 965)
† 주저자, superh1@gmail.com
‡ 교신저자, taekyoung@yonsei.ac.kr(Corresponding author)

및 문제점을 지적한다. 4장에서 상용 앱에 대한 분석을 수행하고 5장에서 대응 방안을 제시한다.

II. 관련 연구

스마트폰 보급으로 BYOD(Bring Your Own Devices)가 확산되면서 보안 문제는 더욱 대두되었다[4]. 사용자 정보와 회사 정보가 서로에게 유출될 수 있으며, 일부 기관은 스마트폰 사용을 통제하고 있다[5]. 기관에서 제작한 보안 앱을 설치하도록 하고, 그러지 못한 경우 사용을 금지한다.

일부 스마트폰 제조사들은 기업환경을 목표로 자체적으로 보안 솔루션을 도입하였다. BlackBerry는 자사의 스마트폰에 Balance technology를 도입하였다[6]. BlackBerry Balance를 활성하기 위해서는 기업에서 BlackBerry Enterprise Server를 갖추어야 한다. BlackBerry 단말의 앱과 저장공간은 업무용과 사용자용으로 나뉘어 독립된다. BlackBerry 제공 앱들과 일부 공용 기능 앱은 양쪽 모두에 접근 가능하나, 인스턴스 분리 등을 통하여 데이터가 오가는 것을 방지한다. Fig. 1.은 BlackBerry Balance의 간략한 구조도이다[7]. BlackBerry Enterprise Server와의 데이터 송수신은 모두 암호화된다.

삼성은 안드로이드 플랫폼에 기반한 KNOX를 발표하였다[8]. 이는 미국 국가안보국(National Security Agency)에서 안드로이드 플랫폼을 위해 제시한 Security Enhanced Android[9][10]에 기반하며, 그 밑에 하드웨어에 따라 수정 가능한 Secure Boot와 TrustZone-based Integrity Measurement Architecture(TIMA)가 존재한다. TrustZone은 ARM CPU에서 제공되는 기술로, 보안모드와 일반모드를 분리하여 안전한 지불결제 환경 등을 제공한다. 삼성은 KNOX 기술로 미국 국방부(Department of Defense)의 보안인증을 통과했다[11]. Fig. 2.는 KNOX의 계층 개요이다.

사용자 환경의 보안으로써, LG는 게스트 모드를 제공한다[12]. 게스트 모드에서는 미리 설정한 앱에만 접근할 수 있도록 하여 사용자 정보를 보호한다. 사용자 주의를 위한 안내 역시 여러 곳에서 발표되고 있다. 안철수연구소는 스마트폰 보안 수칙 10계명[13], 한국인터넷진흥원은 스마트폰 이용자 10대 안전 수칙[14], 금융감독원은 스마트폰 금융거래 10계명을 발표하였다[15]. 내용은 대동소이하다. 신뢰할

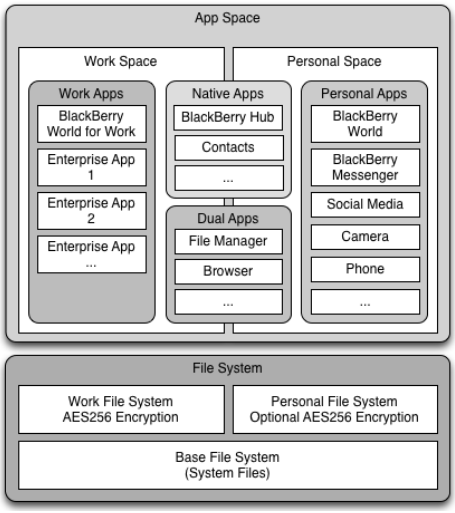


Fig.1. BlackBerry Balance Architecture

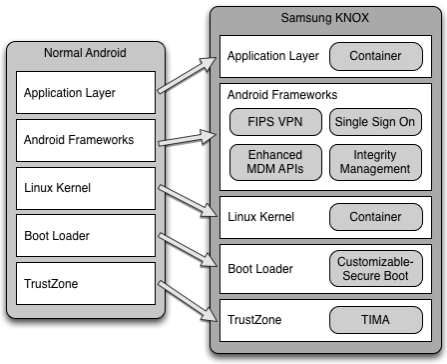


Fig.2. Samsung KNOX Layer

수 있는 앱마켓 이용하기, 백신 설치하기, 개인/금융정보 저장하지 않기, 임의개조 하지 않기 등이 있다.

III. 배경 및 문제점

3.1 연구 배경

안드로이드 사용자가 늘어남에 따라, 공격 역시 증가하고 있다. 위에서 소개한 자체 기술은 ① 제조사, 단말 등에 따라 지원 여부가 다르고, ② 보안 솔루션 API가 적용되지 않고 개인 영역에 설치된 앱에 대해서는 여전히 공격이 가능하다. 따라서 앱과 앱 데이터 보안은 여전히 필요한 과제이다. 안드로이드는 보안 기법을 제공하고 있으나 무력화 될 수 있다. 안드로이드 보안 기법과 공격법을 소개한다.

3.2 안드로이드 보안 요소

안드로이드는 비인가된 접근으로부터 메모리와 스토리지를 보호하는 기법을 기본으로 제공하고 있다[16]. 앱에서 시스템의 중요한 기능을 사용하기 위해서는 사용을 명시해야 하며, 앱 설치시 이를 사용자에게 알림으로써 사용자가 앱의 위험성을 판단할 수 있게 한다. 본 연구에서는 사용자 관리의 측면이 아닌, 시스템 측면을 분석한다.

1) 앱 샌드박스: 안드로이드는 리눅스 커널 위에서 동작하는 플랫폼이다. 사용자가 새로운 앱을 설치하면, 안드로이드는 해당 앱에 대한 유일한 리눅스 사용자 ID(UID)를 발급한다. 해당 앱은 패키지명과 동일한 이름의 디렉토리에 설치되며, 이 디렉토리 또한 해당 UID만이 읽기/쓰기 권한을 갖도록 설정된다. 앱에서 생성한 파일은 기본적으로 이 디렉토리 안에 생성된다. 앱의 실행 파일, 생성 파일 등 자원에 다른 앱이 접근할 수 없도록 보호받으며, 반대로 다른 앱의 자원에 접근할 수 없다. 앱 런타임은 앱에 할당된 유일한 UID 계정으로 실행되므로 앱 프로세스는 리눅스 커널에 의해 다른 앱 프로세스의 메모리 등 자원에 접근할 수 없다.

2) Secure Element: 스마트폰에서 전자 결제 서비스 요구가 높아짐에 따라 NFC(Near Field Communication)기능이 기본 탑재되는 단말기가 증가하고 있다. NFC는 자체 CPU와 SE(Secure Element)를 갖추고, 카드 정보 등과 같은 중요한 정보를 SE에 안전하게 저장한다. NFC 통신을 위한 안테나는 단말기에 위치하며, CPU와 SE는 단말기에 내장될 수도 있고, USIM(Universal Subscriber Identity Module)에 내장될 수도 있다[17]. 자체 CPU를 가지고 있다는 것은, 안드로이드 플랫폼과 분리되어 있음을 의미한다. SE로의 접근은 운영체제 차원에서 엄격하게 제어된다.

3.3 보안 요소의 문제점

안드로이드가 위와 같은 여러 보안 기법을 제공함에도 불구하고, 중요한 보안 문제를 안고 있다. 이 문제점은 앱에서 저장한 중요 정보를 노출시킬 수 있으므로 앱 개발시 반드시 고려되어야 한다.

1) 루팅: 앱 샌드박스는 리눅스 커널의 권한 제어에 기반하므로, 사용자가 루트 계정을 획득할 경우 무력화될 수 있다. 사용자에게 배포되는 펌웨어에는 로그인 계정 변경할 수 있는 su 명령어가 존재하지 않으므로 사용자가 루트 권한을 획득하기란 불가능하다. 시스템의 버그를 이용하거나, su가 포함된 펌웨어를 설치하는 방법으로 su 명령어를 단말기에 설치할 수 있으며, 이를 루팅(rooting)이라고 한다[18]. 루팅된 단말기, 즉 su명령어가 설치된 단말기에서는 루트 권한을 획득할 수 있으므로 앱 샌드박스와 무관하게 다른 앱의 자원에 접근할 수 있다.

2) SE 사용 제약: SE는 NFC 하드웨어 모듈이 탑재된 안드로이드 단말기에서만 사용 가능하다. 또한, 안드로이드에서 SE에 데이터를 입력하기 위한 API는 공개되어 있지 않다. 앱 개발자가 SE에 데이터를 넣기 위한 함수를 구현했다 하더라도, SE가 USIM에 내장된 경우에는 통신사의 승인을, 단말에 내장된 경우 단말 제조사의 승인을 받은 경우에만 사용 가능하다. 따라서, 임의의 앱에서 데이터를 SE에 저장하기란 사실상 불가능하다.

3) 암호화 저장 기법 부재: 앱에서 저장한 데이터는 앱 샌드박스에 의해 보호되므로, 안드로이드 API에는 안전한 데이터 저장을 위한 별도의 기법을 제공하지 않는다. 그러나 위에서 지적한 바와 같이 루팅이 된 단말기인 경우 저장된 데이터가 보호되지 않으므로, 암호화되지 않은 데이터는 노출될 수 있다.

4) 역공학: 안드로이드 앱은 자바로 쓰여진다. 자바는 다른 프로그래밍 언어에 비하여 역공학이 용이하다. 단말기에 설치된 앱을, 컴퓨터 등으로 추출하기 위한 방법 역시 용이하다. 암호화 키가 앱 내에 고정되어 있는 경우, 역공학을 통해 암호키가 노출될 수 있다.

IV. 상용 암호 소프트웨어 분석

본 장에서는 역공학 과정과 필요 도구를 간단히 소개한 후, 구글 플레이에 등록된 앱 중 카드 정보 등 개인정보를 다루거나 암호 기능을 제공하는 앱을 역공학하여 분석한다.

4.1 역공학 과정 및 도구

안드로이드 앱 역공학 과정은 다음과 같다. apk파일 획득은 구글 플레이에서 쉽게 설치할 수 있는 무료 탐색기 앱으로 가능하다. 나머지 과정 PC 환경에서 이루어지며, 관련 도구는 무료로 구할 수 있다.

① apk 파일 획득
② 압축 해제
③ dex 파일 변환 - dex2jar[19] 이용
④ jar 파일 압축 해제
⑤ class 파일 디컴파일 - jad[20] 등 이용

4.2 상용 앱 분석 결과

구글 플레이, 즉 안드로이드 앱 마켓에서 개인 정보를 사용하거나 암호 기능을 제공하는 상용 앱을 설치한 후, 이를 분석하였다. dex2jar 0.0.9.13, jad 1.5.8g 버전을 사용하였다. 데이터 파일에 접근하기 위하여 루팅된 단말을 사용하였다.

1) CJ One card (2.7.0): 제품, 컨텐츠 등을 구매할 때 포인트를 적립하고, 쿠폰 등을 발행하는 앱이다. 이 앱은 자동 로그인 기능을 지원하므로, 계정 정보가 단말기에 저장된다.

AES 클래스 디컴파일 결과, AES 알고리즘을 사용하며, 암호키가 "cjonemobilecard1"로 고정되어 있음을 확인하였다. SetCardPW 클래스에서 사용자로부터 입력받은 패스워드를 위 AES 클래스의 메쏘드를 호출하여 암호화함을 확인하였다. 앱이 설치된 디렉토리의 shared_prefs/CjOneCarePref. xml에서 autoLoginUserPass 항목 값이 2230e31f-942970fb463b3168b9734f84임을 확인했다. 해당 파일은 루팅된 단말에서만 접근이 가능하다. 이 값을 복호하여 Fig. 3.과 같은 값을 얻어냈다. 이는 로그인 패스워드와 동일하다.

2) Gallery Lock Lite (4.6): 갤러리로부터 사진 및 동영상을 숨겨주는 앱이다. 앱 실행시 패스워드를 입력해야 하며, 앱에서 미디어 파일을 선택하면, 다른 갤러리 앱에서 해당 미디어파일이 보이지 않는다. shared_prefs/com.morrison.gallery locklite_preferences. xml 파일에 앱 실행 비밀번호가 암호화되지 않고 그대로 저장되어 있었다. 하지만 앱은 난독화되어 있어 쉽게 분석되지 않았다. 주 패키지의 util/ia 클래스에서 "DESede" 문자열은 난독화되지 않아 3-DES를 사용함을 알 수 있었다. 암호키는 자바 API를 통해 랜덤하게 생성했으나, 저장 방법을 찾지 못했다. 앱을 난독화하여 분석이 어려웠으나, 앱에 암호 기능은 구현되어 있지만 실제로 파일이 암호화되지 않았다. 확장자가 변경되고 숨김 폴더로 이동되어 드러나지 않을 뿐이었다.

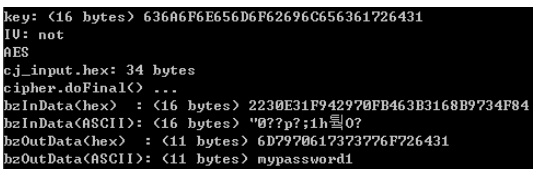


Fig.3. Decrypting user's password

3) Encrypt File Free (1.0.6): 사용자가 선택한 파일을 암호화하는 앱이다. 앱 구동 시 비밀번호를 물어보며, 암복호 시에는 비밀번호를 묻지 않는다. 앱 구동 비밀번호는 Login 클래스에서 MD5되어 databases/optionscrypt 파일과 암호화된 파일의 첫 부분에 기록된다. Fig. 4.는 앱 구동 비밀번호가 "mypassword1"일 때 저장된 모습이다.

암호화 방식으로는 공개 알고리즘을 사용하지 않고 내부에서 독자적인 알고리즘을 사용하고 있다. Main 클래스와 Crypt 클래스를 분석하여 복호 알고리즘을 완성하였다. 디컴파일된 소스를 활용하여 복호 프로그램을 제작한 후, 암호화한 파일에 수행하여 원본을 획득할 수 있었다.

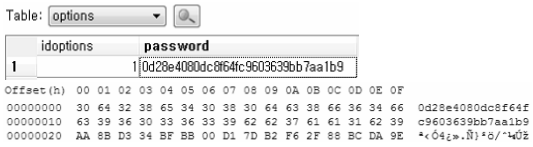


Fig.4. Stored password using MD5

4) Secret Space Encryptor (1.4.7): 문자열, 파일 등을 암호화하는 앱이다. 암호시 패스워드는 사용자로부터 입력받는다. 난독화는 되어 있지 않으나, 다양한 알고리즘이 포함되어 있고 복호 패스워드를 저장하지 않고 사용자에게 물어보는 방식으로서 정적 분석만으로는 파일을 복호할 수 없었다.

V. 대응 방안

위와 같은 공격을 막기 위한 대응 방안을 제시하고 이를 위한 개발 프로세스를 제안한다.

5.1 단위 보안 기능

1) 데이터 암호화는 반드시 적용해야 할 기술이다. 암호화 키 역시 안전하게 관리되어야 한다. 소스 코드에 고정되어 있거나 파일 등에 기록한 경우 키가 노출될 수 있다. 〈사용자 입력값〉 및 〈기기의 고유 하드웨어 정보〉를 조합 및 변형하여 사용한다면, 해당 키는 정적 분석만으로는 알기 어려워지는 동시에 기기에 종속된다. 다른 기기에서 동일한 사용자 입력 값을 입력해도 복호되지 않는다.

2) C/C++로 작성된 JNI 코드는 자바 바이트 코드보다 역공학이 어렵다. 민감한 데이터는 JNI 내부에서 처리하고, 자바 실행 코드에서 필요한 데이터만 가져와서 사용하도록 하면 역공학에 보다 강인해질 수 있다. 암호 라이브러리를 직접 구현할 수도 있고, OpenSSL과 같은 기존 C/C++ 라이브러리를 빌드하여 사용할 수도 있다.

3) 역공학에 대응하기 위하여 난독화를 적용할 수 있다. 난독화는 소스 혹은 머신 코드를 사람이 이해하기 어렵게 만드는 기법이다[21]. 변수와 연산을 추가하고 제어 흐름을 바꾸어 알고리즘을 알아보기 어렵게 만드나, 최종적으로 동일한 결과가 출력되도록 한다. 자바는 주로 소스 코드가 아닌 바이트 코드, 즉 class 파일에서 난독화가 이루어진다. 안드로이드는 class 파일을 직접 실행시키는 것이 아니라 dex 파일을 실행시키므로 일반적인 난독화 도구는 사용할 수 없다. 안드로이드를 지원하는 난독화 도구를 사용해야 하며, 안드로이드 SDK에 ProGuard 난독화 도구가 포함되어 있다[22]. 개발 환경에서 ProGuard를 활성화하면, 설치 파일 생성 단계에서 자동으로 난독화를 수행한다. 난독화는 역공학을 불가능하게 만드는 것이 아니라, 분석에 소요되는 시간을 증가시켜 공격자로 하여금 중단하게 만든다. 난독화를 해제 및 공격하는 기법 또한 존재하므로, 난독화만으로는 공격에 대비할 수 없다[23].

5.2 데이터 보호 프레임워크

위에서 제시한 3개의 기법은 동시에 적용되어야 한다. 이들을 유기적으로 결합하여 앱에 적용해야 한다. 앱 설계 과정에서 체계적으로 보호 기법을 적용할 수 있도록 데이터 보호 프레임워크를 제안한다. Fig. 5.는 데이터 보호 프레임워크의 계층도이다.

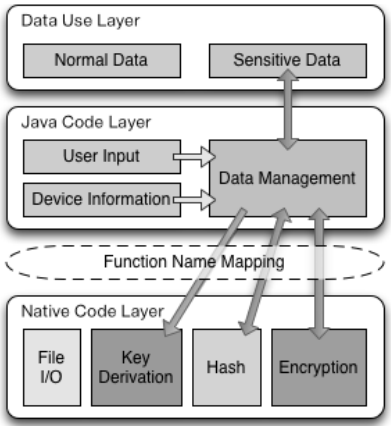


Fig.5. Data Protection Framework

1) 데이터 사용 계층: 앱에서 사용할 데이터 중, 암호화가 필요 없는 일반 데이터와 암호화가 필요한 민감한 데이터로 나눈다. 민감한 데이터를 저장, 전송해야 하는 경우에는 반드시 하부 계층을 통하여 보안을 적용해야 한다.

2) 자바 코드 계층: 안드로이드 SDK로 구현되는 계층이다. 키 생성에 활용할 〈사용자 입력값〉과 〈기기의 고유 정보〉는 이 계층에서 획득한다. 데이터 관리 모듈은 자바 코드 계층에서 보안에 가장 민감한 부분이다. 평문, 암호문을 하위 계층으로 전달하고 암호, 복호, 해쉬 결과를 받아 UI, 네트워크 등으로 전달한다. 보안 연산을 단위 기능으로 분리하여 하위로 호출하는 경우, 중간 값을 자바 코드 계층에서 임시로 가지고 있어야 한다. 정적 분석만으로는 이를 알아낼 수 없지만 동적 분석에서는 일부 정보가 노출될 수 있다. 그러므로 앱에서 수행하고자 하는 큰 기능 단위로 하위 계층에 요청하도록 한다. 기능을 나누어서 호출할 경우에는, 중간에 더미 값을 포함시켜 분석으로부터 강인하도록 설계한다. 자바 코드 계층은 앱 배포 전에 반드시 난독화를 적용해야 한다.

기능 설계 시, 가령 AES와 SHA1연산을 수행해야 하는 경우, 하위 계층의 aes()와 sha1()을 각각 호출하는 것이 아니라, getData()와 같은 하나의 함수만을 호출하도록 한다. getData()내에서 내부의

aes()와 sha1()을 호출함으로서 자바 코드 계층을 분석하는 것만으로는 내부를 알기 어렵도록 한다.

3) 함수 이름 사상: 자바 계층을 난독화하여 JNI 호출 함수 이름이 변경될 경우, 정상적으로 JNI 함수를 호출할 수 없다. JNI 함수 명이 encrypt()과 같이 쉽게 내용을 추측 가능한 경우는 보안에 취약할 수 있다. 따라서 JNI 함수 이름을 aa, bb 와 같이 의미를 알 수 없거나 getProcess() 등과 같이 다른 의미의 이름으로 사상하여 구현한다.

4) 데이티브 코드 계층: 암복호, 해쉬, 키 유도, 파일입출력 등 보안 기능을 구현한다. 자바 코드 계층에서 요구하는 기능을 수행하고 결과를 반환한다. 유도된 키는 자바 코드 영역으로 전송하지 말고, 내부에서만 사용해야 한다.

Fig. 6.은 데이터 보호 프레임워크 구현 절차이다. 데이터의 분류를 시작으로 키 유도, 필요한 암호 기능을 정의한다. 이에 따라 JNI로 구현하고 함수 이름을 사상한다. 데이터 관리 모듈에서는 사상된 JNI를 통하여 데이터를 관리하도록 한다.

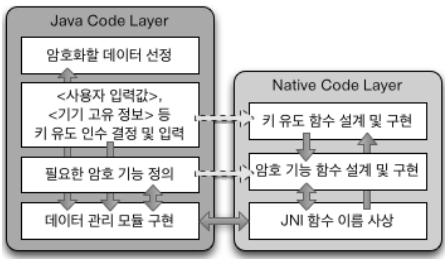


Fig.6. Implementation Method

VI. 결 론

스마트폰의 다양한 기능에 따라, 더 많은 개인 정보가 단말기에 저장되고 있고 이는 보안 위협으로 다가오고 있다. 본 연구에서는 이러한 위협이 실제함을 보였다. 앱의 알고리즘은 역공학을 통해 드러날 수 있고, 저장한 데이터 역시 루팅에 의해 노출될 수 있다. 개인 데스크탑에서 관리자 권한이 일반적이듯, 스마트폰에서도 루팅을 고려한 대응을 해야 한다. 사용자에게 루팅의 위험성을 알리고 주의를 당부하는 일 역시 중요하지만, 앱을 제작할 때에는 본 연구에서 제시하는 데이터 보호 프레임워크에 따라 앱의 알고리즘 및 데이터를 안전하게 보호해야 한다.

References

[1] Gartner, "Market Share Analysis: Mobile Phones, Worldwide, 2Q13," http://www.gartner.com/newsroom/id/2573415, Aug. 2013.

[2] Gartner, "Market Share: Mobile Communication Devices by Region and Country, 3Q11," http://www.gartner.com/newsroom/id/1848514, Nov. 2011.

[3] icrossing, "2013 Mobile Market Share," http://connect.icrossing.co.uk/2013-mobile-market-share-infographic_10062, Jan 2013.

[4] Wikepedia, "Bring Your Own Device," http://en.wikipedia.org/wiki/Bring_your_own_device

[5] News1, "From 15th, blocking smartphone features in the building of Ministry of National Defense," http://news1.kr/articles/1239175

[6] BlackBerry, "Balance technology," http://us.blackberry.com/business/software/blackberry-balance.html

[7] Jason Foy, "Understanding BlackBerry Balance," BlackBerryLive, 2013.

[8] Samsung, "KNOX," https://www.samsungknox.com/en/

[9] SELinux Wiki, "SEforAndroid," http://selinuxproject.org/page/SEAndroid

[10] Centrify, "Samsung to OEM Centrify for Single Sign-On and Mobile Management," http://www.centrify.com/blogs/tomkemp/samsung_oems_centrify_for_sso_and_mdm.asp

[11] Samsung, "Samsung KNOX available for use by consumers," http://www.samsung.com/us/news/21651

[12] LG Electronics, "Guest Mode," http://www.lgmobile.co.kr/microsite/LGG2/features/features03.jsp

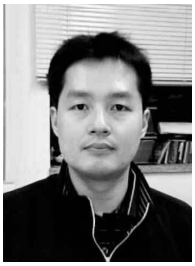
〔13〕 AhnLab, Inc., "10 Commandments of smartphone security policy," http://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?menu_dist=2&seq=16012
〔14〕 Korea Internet & Security Agency, "10 safety regulations for smartphone users," http://boho.or.kr/kor/private/private_02.jsp
〔15〕 Financial Supervisory Service, "10 Commandments of smartphone banking," http://www.fss.or.kr/fss/kr/promo/bodobbs_view.jsp?page=1&seqno=14941
〔16〕 Google, "Android Security Overview," http://source.android.com/devices/tech/security/index.html
〔17〕 Smart Card Alliance, "Mobile/NFC Security Fundamentals," http://www.smartcardalliance.org/resources/webinars/Secure_Elements_101_FINAL3_032813.pdf
〔18〕 Wikipedia, "Android rooting," http://en.wikipedia.org/wiki/Android_rooting
〔19〕 dex2jar, https://code.google.com/p/dex-2jar/
〔20〕 jad, http://varaneckas.com/jad/
〔21〕 Wikipedia, "Obfuscation," http://en.wikipedia.org/wiki/Obfuscation_(software)
〔22〕 ProGuard, http://developer.android.com/tools/help/proguard.html
〔23〕 "A look inside Dexguard," http://www.android-decompiler.com/blog/2013/04/02/a-look-inside-dexguard/

〈저자소개〉

박 상 호 (Sang-ho Park) 학생회원
2004년 2월: 세종대학교 컴퓨터공학과 학사
2006년 2월: 세종대학교 컴퓨터공학과 석사
2008년 2월: 세종대학교 컴퓨터공학과 박사과정 수료
2008년~2011년: 잉카엔트웍스 전임연구원
2011년 3월~현재: 세종대학교 컴퓨터공학과 박사과정
〈관심분야〉 암호프로토콜, 스마트폰 보안 등

사 진

김 현 진 (Hyeonjin Kim) 정회원
1992년 2월: 서울대학교 수학과 졸업
1994년 2월: 포항공과대학교 수학과 석사
2008년 8월: 한국과학기술원 수리과학과 박사
1994년 2월~1999년 9월: 한국전자통신연구원
1999년 10월~현재: ETRI부설연구소 책임연구원
〈관심분야〉 정보보호, 암호분석, 불함수이론



권 태 경 (Taekyoung Kwon) 종신회원
1992년 2월: 연세대학교 컴퓨터과학과 학사
1995년 2월: 연세대학교 컴퓨터과학과 석사
1999년 8월: 연세대학교 컴퓨터과학과 박사
1999년~2000년: U.C. Berkely Post-Doc.
2001년~2013년 8월: 세종대학교 컴퓨터공학과 교수
2007년~2008년: Univ. Maryland at College Park 교환교수
2013년 9월~현재: 연세대학교 정보대학원 부교수
〈관심분야〉 암호프로토콜, 네트워크 프로토콜, 센서네트워크 보안, HCI 보안 등