RMI 방식을 이용한 안드로이드 애플리케이션의 데이터 암호화 프레임워크

김성수○ 홍충선*
경희대학교 컴퓨터공학과
{ mjs0514, cshong }@khu.ac.kr

A Framework for Data Encryption of Android Application Using RMI Scheme

Sung Soo Kim○ Choong Seon Hong*
Department of Computer Engineering, KyungHee University

요 약

안드로이드 애플리케이션 개발자들은 소비자들의 요구를 만족하기 위해 실용성과 창의성에만 집중하는 경향을 보이고 있다. 그로인하여 암호화 체계를 갖춘 애플리케이션은 매우 드문 것으로 나타나고 있으며, 지속적인 개인정보 유출 및 보안 사고들이 발생하고 있다. 이제 소비자들은 더 이상 보안에 취약한 애플리케이션을 원하지 않는다. 따라서 본 논문에서는 개발 시 데이터 암호화가 필수적인 이유를 설명하고, 암호화 프레임워크가 애플리케이션의 기능 및 구조상 달라질 수밖에 없다는 점 때문에 발생하는 문제점을 제시하며 그에 대한 해결방안으로 RMI 방식을 이용한 암호화 프레임워크를 제안한다.

1. 서 론

스마트 폰의 애플리케이션 시장에는 사용자들의 요구를 만족하기 위해 수많은 애플리케이션들이 나타나고 있다. 하지만 사용자들의 흥미를 끌지 못하면 도태되어 버리는 시장의 특성 때문에 개발자들은 실용성과 창의성에 집중하고 있다. 이런 환경 탓에 자연스럽게 보안상의 허점들이 존재하게 되어 악의적인 의도를 가진 사용자들에 의해 악용되고 있으며, 그로 인하여 금전적인 피해나 개인정보 유출 등의 문제점이 발생하고 있다.

세이프넷에서 발표한 전 세계 데이터 유출/침해 통계를 따르면 2014년 2분기 총 237건의 개인정보 등의 민감한 정보가 유출되었다고 한다.[1] 이는 1분기에 비해 줄어든 수치이지만 여기에서 주목할 점은 2분기에 보고된 237건이 발생한 기관들 중 데이터 암호화와 강력한 인증 수단을 기반으로 한 접근 제어 체계를 갖춘 곳은 겨우 1%밖에 되지 않는다는 점이다. 그 외에 얼마 전 카카오 톡의 사생활 보호에 대한 신뢰가 깨져버려 사이버 망명사태가 일어난 사건만 봐도 기업들이 소비자들의 개인정보보호에 대하여 별다른 문제점을 인식하고 있지 않다고 여겨질 수 있다.

그러나 현재 소비자들은 개인정보에 대해 매우 민감하게 받아들이고 있다. 그만큼 보안은 권장사항이 아닌 필수사항이며, 이를 위해 할 수 있는 가장 쉬우면서 확실한 방법이 데이터 암호화라는 것을 우리는 모두 알고 있다. 하지만 애플리케이션의 기능 및 구조상 암호화 방식이 달라야만 하는 경우가 존재하는데, 본 논문에서는 문제가 되는 경우를 설명하고 이 때문에 달라져야 하는 암호화 모듈의 프레임워크를 하나로 통합하기 위한 해결방안으로 RMI방식을 이용한 방법을 제안한다.

본 논문은 산업통상자원부 산업핵심기술개발사업으로 지원된 연구결과입니다[10049079, 퍼스널 빅데이터를 활용한 마이닝 마인즈 핵심기술 개발] *Dr. CS Hong is the corresponding author

2. 관련 연구

2.1 RMI(Remote Method Invocation)

Java RMI는 개발자로 하여금 다른 호스트나 다른 Java VM에 있는 자바 기반의 분산 객체들을 원격으로 호출 할 수 있도록 만든 기술이다.[2] 이런 RMI가 원격 메소드를 호출하는 과정을 이해하기 위해서는 몇 가지 개념을 알아야 한다. 그 중에 스텁(Stub)[3]이라는 것이 있는데, 스텁이란 클라이언트가 원격지의 메소드를 호출하려고 할 때 그 메소드의 존재를 알려주고 추상적으로 나타내는 것이며, 다른 말로는 원격 인터페이스라고도 한다.

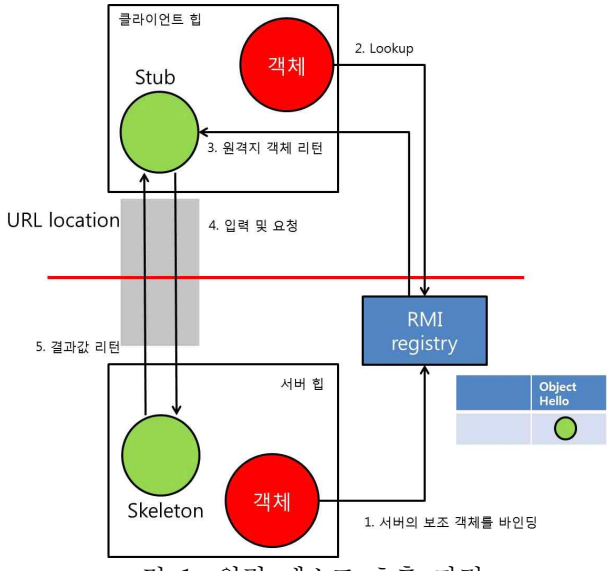

그림 1. 원격 메소드 호출 과정

그림 1은 클라이언트에서 서버 힙에 존재하는 원격 메소드를 호출하는 과정을 나타낸다. 먼저 클라이언트 내부를 보면 객체와 스텁 객체가 존재하는데, 스텁 객체는 서버에 있는 객체의 메소드가 마치 클라이언트 내부에 있는 것처럼 사용하기 위한 원격 인터페이스 역할을 한다. 이렇게 사용하기 위해서는 첫 번째로 서버에 존재하는 RMI registry에 클라이언트와 통신하기 위한 스켈레톤(Skeleton) 객체가 바인딩 되어 있어야 한다. 사실 스켈레톤은 스텁과 마찬가지로 원격 인터페이스 역할을 하는 보조 객체지만 구분을 위해 다르게 정의됐다. 두 번째로 클라이언트에서 서버에 있는 레지스트리에 Lookup 명령어를 통해 호출하려는 메소드의 객체를 찾는다. 세 번째로 Lookup에 대한 결과값으로 객체를 돌려준다. 그럼 클라이언트는 이 객체를 통해서 마치 클라이언트 내부에 있는 것처럼 서버 쪽 객체의 메소드를 사용한다. 하지만 실제로는 스텁 객체는 코드 내부에서 입력한 파라미터 값을 서버에게 전송한 다음 결과값을 되돌려 받는 과정을 거친다. URL location은 자바 JDK에 포함된 JSSE(Java Secure Socket Extension) API에 구현된 SSL을 통해 통신할 수 있다.[4]

2.2 앱 샌드박스[5]

안드로이드는 Linux 위에서 동작하는 플랫폼이지만 기존 Linux와 다르게 각각의 앱에 고유의 User ID를 할당한다. 그래서 앱이 설치 된 폴더는 해당 앱의 UID만이 읽기/쓰기 권한을 갖도록 설정된다. 그렇기 때문에 앱에 의해 생성되는 파일들은 자신의 고유 폴더에만 저장되며, 이 자원들은 다른 UID를 가진 앱들에 의해 접근할 수 없도록 보호받는다. 앱 샌드박스란 이렇게 앱의 자원을 고립하고 식별된 사용자만이 사용할 수 있도록 하는 보호방법을 말한다.

3. 배경 및 문제점

3.1 연구 배경[6]

안드로이드는 앱 샌드박스에 의해 앱에서 저장한 데이터를 저장한다. 하지만 이러한 구조 때문인지, 안전한 데이터 저장을 위한 별도의 암호화 기법을 제공하지 않는다. 하지만 악성 사용자에 의해 안드로이드 디바이스가 루팅 된 경우라면 루트 계정을 획득한 사용자에 의해 암호화 되지 않은 데이터가 유출되는 문제점이 발생한다. 여기에서 루팅이란 루트 계정을 획득하기 위해 하는 행위를 말한다.[7] 일반적으로 사용자에게 배포되는 안드로이드 펌웨어에는 로그인 계정을 변경할 수 있는 su 명령어가 없다. 그렇기 때문에 사용자가 루트 계정을 획득하는 것은 불가능하다. 하지만 아직까지 시스템의 보안 허점을 이용해 루트계정을 획득할 수 있는 방법들이 존재한다.

3.2 문제점

위와 같은 배경에 의해 우리는 필수적으로 데이터를 암호화해야 한다. 그러나 이러한 암호화 과정에서 구조적인 문제를 가지는 경우가 있다. 예를 들면 어떤 서비스에 자동로그인을 하는 기능을 만드는 경우 우리는 ID와 PW를 암호화 시켜서 저장한다. 이런 데이터를 암호화하기 위해 일반적으로 공개된 알고리즘을 사용하며, 이 때 필요한 개인키를 소스코드 내부에 저장하여야만 한다. 왜냐하면 자동로그인이라는 기능이 패스워드를 입력하지 않고 로그인하기 위한 것인데, 패스워드를 암/복호화 하려고 암호화에 필요한 다른 키 값을 또 입력하게 하는 것은 기능상에 모순이다. 그렇기 때문에 키를 자체적으로 저장하는 구조를 가지게 된다. 하지만 키 값을 저장하게 되면 역공학(Reverse-Engineering)에 의해 소스코드가 노출 될 수 있고, 이로 인해 암호 키는 제 역할을 하지 못하게 된다. 여기서 역공학이란 안드로이드 실행파일인 apk파일을 디컴파일을 통해 역으로 소스코드를 추출하는 방법이다. 역공학을 막기 위한 방법으로 소스코드 난독화가 있지만, 결국 소스 코드 분석을 어렵게만 할 뿐이기 때문에 키가 노출 될 수 있다.

4. 제안 사항

우리는 3.2절에서 제시한 사용자의 입력 없이 저장된 키를 사용해야 되는 경우와 반드시 사용자로부터 입력을 받아야만 하는 경우인 두 가지로 나누어 정의했다. 본 제안사항에서는 전자의 경우를 해결하기 위한 구조를 설계했다.

4.1 구조

그림 2는 서버내부에 존재하는 암호화와 복호화 모듈에 저장된 키가 있음을 보여주고 그것을 가지고 암/복호화 하는 구조를 나타낸다.

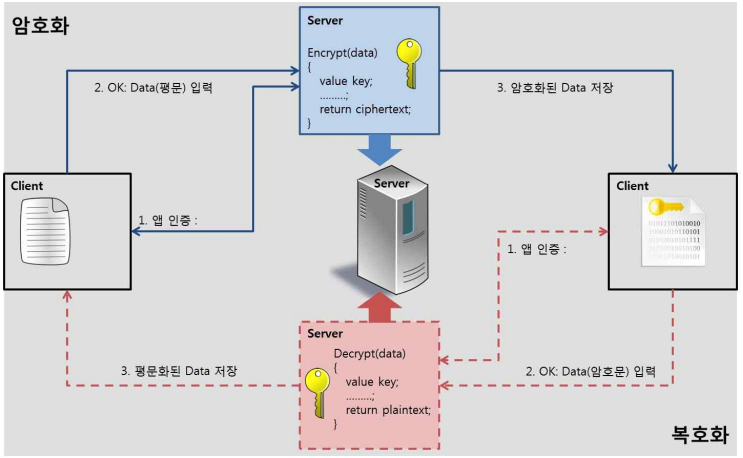

그림 2. 제안하는 아키텍처

첫 번째로 암/복호화 공통적으로 서버에게 데이터를 보내기 전에 반드시 앱 인증을 거친다. 그리고 올바른 접근임이 확인되면 데이터를 서버에 입력한다. 그 다음 서버 내부에서 입력 값을 가지고 결과 값을 도출해서 클라이언트 앱에 할당된 저장소에 데이터를 저장하는 구조로 동작하게 했다. 여기서 앱 인증이란 서버에게 요청하는 클라이언트가 반드시 개발자가 서비스하기 위한 앱이라는 것을 확인하는 것인데, 이 인증과정을 거치지 않으면 암호화된 데이터를 수집한 공격자가 모방 앱을 통해 복호화를 요청하는 방법을 통해 평문이 노출되는 방법이 성립한다. 그러므로 반드시 앱을 인증하도록 해야 한다. 인증 방법으로는 apk파일의 해쉬 값을 서버에 등록하고

그 값을 비교하도록 하는 방법 및 패키지 이름을 비교하는 등 다양한 방법을 사용 할 수 있다.

4.2 구현

자바에서는 기본으로 RMI 라이브러리를 지원하지만, 안드로이드에서는 지원하지 않기 때문에 안드로이드용으로 제작된 lipermi를 사용했다.[8]

표 1. 서버/클라이언트 클래스 명세

구분	클래스명	설명
서버	RemoteService	원격인터페이스 (Skeleton 객체)
	RemoteServiceImpl	인터페이스의 구현부(서버객체)
	RemoteServer	메인 클래스
클라이언트	RemoteService	원격인터페이스 (Stub 객체)
	MainActivity	메인 클래스

서버/클라이언트 모두 lipermi 라이브러리를 추가한 뒤 각각 자바와 안드로이드 프로젝트를 생성하고 표 1과 같이 클래스들을 작성했다. 그림 3은 암/복호화 기능을 구현한 모습이다.

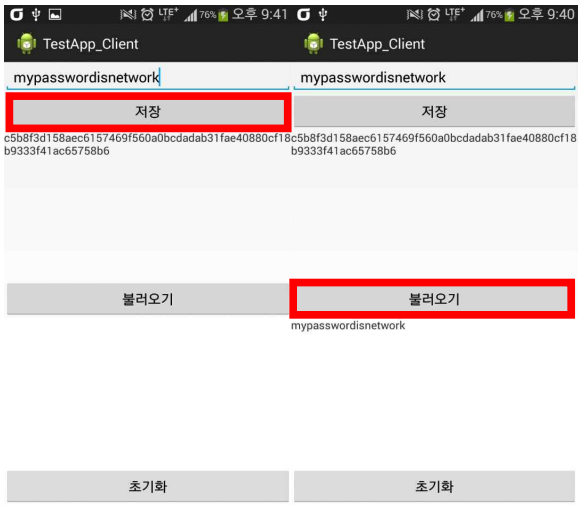

그림 3. 구현한 앱의 암/복호화 기능

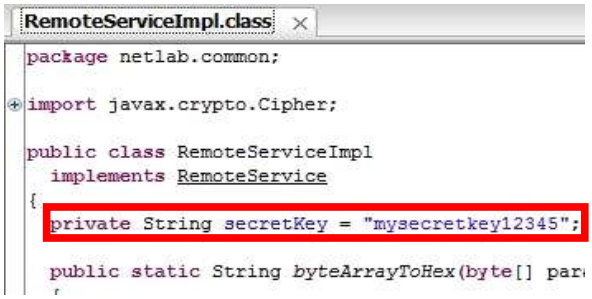

그림 4. 앱 내부에 저장된 키 노출

4.3 성능 평가

제안하는 방법이 역공학을 통해 저장된 키 값이 노출되지 않는 것을 비교하는 것과 다른 공격들에 대한 취약점은 없는지에 대한 분석으로 성능 평가를 진행했다. 디컴파일 툴로는 dex2jar[9]와 jd-gui[10]를 사용했고, 그 결과 그림 4에서 키가 포함된 메소드의 구현부가 앱 내부에 있을 때만 소스코드 상에 저장된 키 값을 찾을 수 있었다. 다음으로 표 2와 같은 취약점들을 대상으로 대안여부를 확인했다. 중간자와 재전송 공격 시나리오에 대해서는 인증절차를 통해 안전하지만 스니핑 같은 경우는 lipermi에서 SSL의 지원여부가 불투명해서 네트워크 상에 전송되는 평문이 도청당할 수 있다는 취약점이 존재했다.

표 2. 다른 공격들에 대한 취약점 분석과 대안 여부

기법	시나리오	여부
중간자 공격	모방 앱을 통해 수정 된 형태의 코드로 암호화된 데이터 유추	O
재전송 공격	세션을 획득 후에 재전송 시도	O
스니핑	전송되는 평문을 도청	X

5. 결 론

본 논문에서 제안한 방법으로 역공학과 같은 방법에는 저장된 키가 보호되는 것을 확인했다. 하지만 표 2에서처럼 스니핑에 대한 취약점이 존재한다. 따라서 연구를 보완하기 위해 lipermi에서 SSL 통신을 지원하도록 하는 연구가 필요하다. 또한 사용자 입력을 꼭 받아서 암호화해야 하는 경우, 어떻게 해야 본 논문에서 제안한 프레임워크와 크게 다르지 않게 할 것인지에 대한 연구도 필요하다.

6. 참고문헌

[1] 정규문, “소매 업계를 노린 개인정보 유출 시도 급증”, 보안뉴스, 2014, http://www.boannews.com/media/view.asp?idx=42271
[2] Java RMI API[Online], Available: http://docs.oracle.com/javase/8/docs/technotes/guides/rmi/index.html
[3] Method stub[Online], Available: http://en.wikipedia.org/wiki/Method_stub
[4] Using Java RMI with SSL[Online], Available: http://docs.oracle.com/javase/8/docs/technotes/guides/rmi/socketfactory/SSLInfo.html
[5] Android Security Overview[Online], Available: http://source.android.com/devices/tech/security/index.html
[6] 박상호 외 2명, “안드로이드 스마트폰 암호 사용 앱 보안 분석 및 대응”, 정보보호학회, Vol.23, No.6, 2013
[7] Android rooting[Online], Available: http://en.wikipedia.org/wiki/Rooting_(Android_OS)
[8] jorgenpt, "LipeRMI", GitHub, https://github.com/jorgenpt/lipermi
[9] dex2jar[Online], Available: https://code.google.com/p/dex2jar/
[10] Java Decompiler[Online], Available: http://jd.benow.ca/