안드로이드 보안카드 앱 취약점 분석 및 OTP를 통한 해결*

최원섭, 김동규**
한양대학교 전자컴퓨터통신학부
e-mail : wschoi@esslab.hanyang.ac.kr

The vulnerability analysis of android application for security card and the solution using OTP

Won Seop Choi, Dong Kyue Kim**
Dept of Electronics and Computer Engineering, Hanyang University

요 약

안드로이드 보안카드 앱은 보안카드를 암호화하여 스마트폰에 저장하고 관리하는 앱으로 사용자의 편의성을 향상 시켜주지만 안드로이드의 특성상 앱 설치 파일로부터 키 노출의 위험이 있다. 본 논문은 안드로이드 보안카드 앱의 디컴파일로부터 키를 추출하는 취약점을 설명하고 보안카드를 대체하기 위해 OTP 사용을 제안한다.

1. 서론

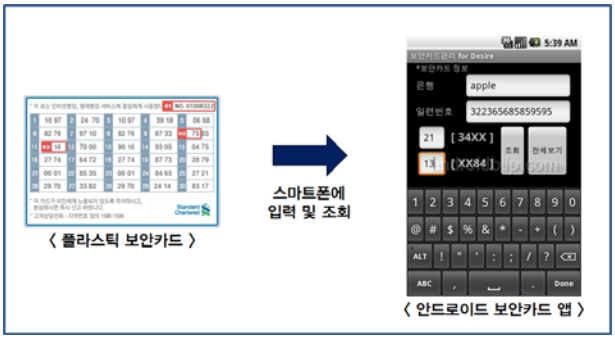
국내 인터넷뱅킹에서는 거래인증을 위해 공인인증서, 비밀번호, 보안카드를 요구한다. 보안카드는 카드 형태로 제공되어 물리적으로 사용자 컴퓨터 및 스마트폰과 분리되어 있어 공인인증서와 비밀번호가 악성코드로부터 해킹되어도 최종적으로 사용자를 보호하는 역할을 수행하였다. 그러나 보안카드는 항상 소지해야 하는 불편함 때문에 최근에는 보안카드를 스마트폰에 전자적 형태로 보관하고 관리해주는 앱(APP)들이 등장하였다. 보안카드 앱은 보안카드 DB를 AES로 암호화 하여 저장하여 쉽게 접근할 수 없지만 안드로이드에서는 앱 설치파일로부터 앱 소스코드를 디컴파일 할 수 있어 키 노출의 위험이 있다. 본 논문은 안드로이드 보안카드 앱의 취약점을 설명하고 이를 보완하기 위해서 OTP 사용을 제안한다.

2. 안드로이드 보안카드 앱 개요

보안카드는 30(또는 35)개의 숫자가 적혀있는 카드로 사용자 인증 시에 시스템이 요구하는 카드에 있는 특정 번호를 입력하여 인증을 하는 인증 수단이다. 그러나 보안카드의 소지가 불편하여 최근에는 보안카드를 스마트폰에 저장하고 필요시 조회할 수 있는 앱이 등장하였다. 보안카드 앱은 사용자가 직접 보안카드 번호를 입력해 저장하는 방식과 사진을 찍어 이미지처리 후 자동으로 입력되는 방식이 있다.

* 본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음 (NIPA-2013-H0401-13-1008)
** 교신저자

안드로이드 보안카드 앱은 안전하게 앱을 사용할 수 있도록 여러 보안장치를 해놓았다. 보안장치에는 DB폴더 접근통제, 앱 실행 시 비밀번호 요청, 보안카드 DB 저장 시 AES[1]로 암호화, 외부 메모리에 백업 시 AES로 암호화가 있다.



(그림 1) 안드로이드 보안카드 앱

3. 보안카드 앱 디컴파일 및 보안카드 정보 획득

위에서 설명한 보안장치에도 불구하고 보안카드 앱은 안드로이드 OS 자체의 취약점으로부터 위험에 노출된다. 안드로이드는 기본적으로 사용자가 안드로이드 앱의 데이터파일에 접근할 수 없도록 되어있지만 공격자는 사용자의 스마트폰에 악성코드를 감염시킴으로써 앱의 데이터폴더에 접근할 수 있는 권한을 얻을 수 있다. 그러나 공격자가 데이터폴더에 접근할 수 있는 권한을 얻어 보안카드 DB파일을 얻었다 해도 DB파일이 AES로 암호화가 되어 있어 쉽게 DB파일 정보를 얻을 수 없다. 그러나 AES의

키가 저장되어 있는 안드로이드 보안카드 앱 설치파일이 디컴파일이 가능해 결국 키가 노출 가능하다.

안드로이드는 앱 설치를 위해 APK 파일 형식을 사용한다. APK 파일 내부는 안드로이드 앱 설정 정보가 저장되어 있는 AndroidManifest.xml 파일과 안드로이드 가상머신(Dalvik VM)이 실행하는 dex파일 등으로 구성된다. dex 파일은 컴파일된 자바 클래스로 구성되어 있는데 이때 (그림2)와 같이 dex를 jar로 변환해 주는 툴을 이용하여 jar 파일을 생성한 뒤 jar파일을 디컴파일 도구를 이용하여 디컴파일 하면 내부 소스를 읽을 수 있다. 내부소스에 암호화 키는 특정 변수에 할당되어 있으며 소스를 주의 깊게 분석하면 (그림 3)과 같이 암호화키 값을 얻어낼 수 있다.

(그림 2) 안드로이드 dex파일 디컴파일

```
public class Encryption
{
  public static final String AESEncryptKey = "SecurityCardData";
  public static final String DESEncryptKey = "CardData";
```

(그림 3) 안드로이드 dex파일 디컴파일 후 키 값 추출

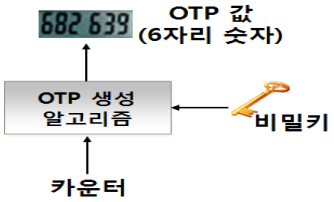
얻어낸 키로 공격자는 보안카드 DB파일을 복호화 시키고 보안카드 DB 내부의 보안카드 번호를 얻어낼 수 있다. 보안카드 DB 내부에는 모든 보안카드 번호 30(또는 35)개가 저장되어 있어 노출 시, 공격자는 정상적인 사용자로 가장하여 사용자에게 심각한 피해를 줄 수 있다.

4. 보안카드 취약점 개선을 위한 OTP 사용

보안카드의 가장 큰 단점은 보안카드 번호가 고정되어 있다는 것이다. 숫자가 고정되어 있으므로 공격자는 키로그 혹은 모니터 감시 해킹 툴을 통해 사용자의 인터넷뱅킹을 반복하여 관찰함으로써 보안카드 번호를 대부분 알아낼 수 있다. 또한 보안카드는 사용자가 편의성을 위해 고정된 보안카드 번호를 앱이나 사진으로 저장하여 사용할 수 있는 여지를 제공하여 사용자의 모든 보안카드 번호가 한 번에 노출 되도록 만들 위험이 있다.

이러한 취약점을 가진 보안카드를 대신하여 안전한 금융거래를 위해서 One time password(OTP)[2]를 사용할 수 있다. OTP는 금융 거래 시 매번 새로운 OTP 값을 생성하여 인증을 수행하는 인증 수단이다. OTP는 매번 OTP 값이 변하므로 공격자가 OTP값을 중간에 습득한다 해도 다음 거래 시에 OTP값이 변하므로 정상적인 사용자로 위장이 불가능하다. 또한 사용자는 반드시 물리적으로 OTP토큰을 소지해야 하므로 공격자가 OTP 토큰에 접근할 수 없어 어떠한 소프트웨어적 형태의 공격도 할 수 없다.

OTP는 (그림4)와 같이 OTP 생성 알고리즘에 비밀키 및 인증서버와 동기화 된 카운터 값을 입력 값으로 하여 6~8자리의 OTP 값을 생성한다.



(그림 4) OTP 생성 모듈

OTP는 보안카드와 달리 OTP 값이 매번 바뀌므로 인증서버에서는 OTP값의 인증을 위해 사용자 OTP 토큰과 동일한 OTP값을 생성해야 한다. 이를 위해 인증서버는 사용자 OTP 토큰과 OTP 생성 알고리즘의 시드 값을 동기화하여 동일한 OTP값을 생성한다. 동기화방식에는 S/Key 방식[3], 시간동기화 방식[4], challenge-response 방식[5], 이벤트 동기화 방식[6] 등이 있다.

OTP는 기본적으로 엄지손가락 크기의 토큰형이나 카드형으로 되어 있어 기존 보안카드보다 휴대성이 떨어진다. 그러나 최근에는 스마트폰의 USIM 기반 모바일 OTP[7]가 개발되었다. USIM 기반의 모바일 OTP는 USIM 카드 내부에 OTP 관련 데이터를 저장하고 USIM카드에서 OTP값을 생성하여 안전하게 단말로 전송하는 모바일 OTP이다. 사용자가 USIM 기반의 모바일 OTP를 사용한다면 OTP토큰을 따로 휴대할 필요 없이 금융 결제가 가능하여 기존 보안카드에서 문제가 되었던 보안성과 휴대성을 모두 해결할 수 있다.

5. 결론

본 논문은 보안카드 앱의 취약점을 설명하고 이로부터 보안카드의 고정된 숫자를 사용하는 방식에 대한 문제점을 분석하였다. 또한 보안카드의 문제점을 개선하기 위해 보안카드를 대신하여 OTP의 사용을 제안하였다. 보안카드는 원칙적으로 비밀번호를 입력하는 스마트폰과 분리되어 관리되어야 한다. 공인인증서와 비밀번호가 스마트폰으로부터 탈취되어도 보안카드가 외부에 있어 안전할 수 있기 때문이다. 그러나 보안카드 사용의 편의성을 위해 보안카드를 스마트폰 내부에 앱이나 사진으로 저장하여 관리하면 공인인증서, 비밀번호와 함께 스마트폰에서 해킹 될 위험성이 있다. OTP는 보안카드를 대신하여 금융거래시 인증을 위해 사용될 수 있다. OTP는 매번 다른 비밀번호를 제공하고 스마트폰으로부터 물리적으로 분리되어 있어 기술적으로 보안카드 시스템의 문제점을 해결할 수 있다.

참고자료

[1] United States National Institute of Standards and

Technology (NIST), "Announcing the ADVANCED ENCRYPTION STANDARD (AES)". Federal Information Processing Standards Publication 197, 2001.
[2] N.Haller, C.Metz, "A One-Time Password System", Network Working Group RFC 1938, 1996
[3] N.Haller, "The S/KEY One-Time Password System", Network Working Group RFC 1760, 1995
[4] D.M'Raihi, S.Machani, M.Pei, J.Rydell, "Time-Based One-Time Password Algorithm", Internet Engineering Task Force RFC 6238, 2011
[5] D.M'Raihi, J.Rydell, S.Bajaj, S.Machani, D.Naccache, "OCRA: OATH Challenge-Response Algorithm", Internet Engineering Task Force RFC 6287, 2011
[6] D.M'Raihi, M.Bellare, F.Hoornaert, D.Nacc-ache, O.Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", Network Working Group RFC 4226, 2005
[7] "미래테크놀로지-Any OTP 개요",
http://www.mirae-tech.co.kr/solution/solutions4.php