Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.6, Dec. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.6.1175

안드로이드 암시적 인텐트의 보안 취약점에 대한 연구*

조 민 재,† 신 지 선‡
세종대학교

Study on Security Vulnerabilities of Implicit Intents in Android*

Min Jae Jo,† Ji Sun Shin‡
Sejong University

요 약

안드로이드는 어플리케이션간의 메시지 전달을 위해 인텐트 메커니즘을 지원한다. 인텐트는 어플리케이션간의 통신을 용이하게 하지만 사용에 따라서 보안상 취약점을 가질 수 있다. 특히, 암시적 인텐트는, 메시지를 전달 받을 컴포넌트를 명시하는 명시적 인텐트와 달리, 메시지를 전달 받는 컴포넌트를 지정하지 않기 때문에, 인텐트를 가로채는 인터셉트 공격이나 인텐트를 변조하는 공격에 취약할 수 있다. 본 논문에서는 암시적 인텐트의 취약점에 대하여 기존에 연구된 공격 방법들과 대응방안을 다시 살펴본다. 개발자 정의 액션을 사용한 인텐트를 이용하는 공격 방식이 많이 연구가 되어 있지만, 안드로이드 표준 액션을 사용한 인텐트을 이용한 공격은 아직 구체적으로 발견한 연구가 없다. 본 논문에서 안드로이드 표준 액션을 사용한 인텐트에 대한 새로운 공격을 소개하고, 이러한 공격으로부터 스마트폰을 보호하는 방법을 논의하고 제안한다.

ABSTRACT

Android provides a message-passing mechanism called intent. While it helps easy developments of communications between intra and inter applications, it can be vulnerable to attacks. In particular, implicit intent, differing from explicit intent specifying a receiving component, does not specify a component that receives a message and insecure ways of using implicit intents may allow malicious applications to intercept or forge intents. In this paper, we focus on security vulnerabilities of implicit intent and review researched attacks and solutions. For the case of implicit intent using 'developer-created action', specific attacks and solutions have been published. However, for the case of implicit intent using 'Android standard action', no specific attack has been found and less studied. In this paper, we present a new attack on implicit intent using Android standard action and propose solutions to protect smart phones from this attack.

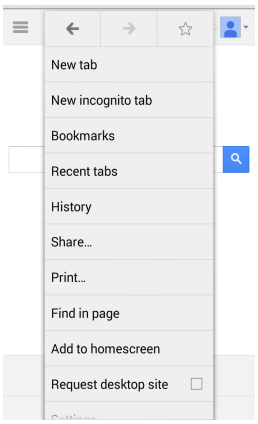
Keywords: Android, Intent Vulnerability, Implicit Intent, Smart Phone Security

I. 서 론

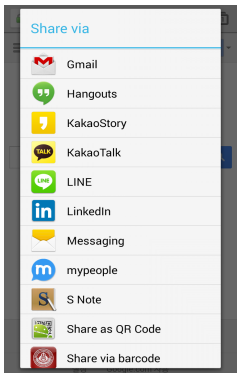
스마트폰 사용자가 많아지면서 스마트폰을 대상으로한 공격이 점차 증가하고 있다. McAfee 보고서에 의하면 2014년 1분기에만 700,000개가 넘는 모바일 멜웨어(malware)가 새로 발견되었다[18]. 특히, 안드로이드 스마트폰은 스마트폰 시장에서 약 84%를 점유하며[8] 가장 많이 사용되는 스마트폰 운영체제(OS) 플랫폼으로 2013년도에 발생한 모바일 악성 앱의 약 96%가 안드로이드 플랫폼에서 발생하였다[9]. 안드로이드는 오픈소스를 기반으로한 개방형 플랫폼으로 개발자들의 접근이 쉽고,

접수일(2014년 8월 21일), 수정일(2014년 10월 24일), 게재확정일(2014년 11월 19일)
* 본 연구는 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2013R1A1A3009163)
† 주저자, eklim@sju.ac.kr
‡ 교신저자, jsshin@sejong.ac.kr(Corresponding author)

(a) A Web Page to share



(b) Menu List of Chrome Application



(c) Captured Screen when a message is sent via Kakaotalk Application

Fig. 1. Screen Captures of Chrome Application when trying to share a Web Page

Google Play외의 다양한 마켓이 존재한다. 이러한 안드로이드의 특징은 어플리케이션 개발과 사용을 활성화함과 동시에 안드로이드 스마트폰이 많은 공격의 대상이 되게 하기도 한다.

안드로이드는 어플리케이션간의 메시지 전달을 위하여 인텐트 메커니즘을 지원한다. 인텐트는 어플리케이션을 구성하는 컴포넌트가 다른 컴포넌트에 자료를 전달할 때 사용된다. 인텐트는 내부 어플리케이션의 컴포넌트들의 통신(intra-application communication)에 사용될 수도 있고, 서로 다른 어플리케이션에 속한 컴포넌트들의 통신 (inter-application communication)에도 사용될 수 있다.

인텐트는 메시지 전달 방식에 따라서 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent)로 나눠진다. 명시적 인텐트는 수신 컴포넌트를 명시하여 그 컴포넌트에게만 인텐트가 전달된다. 암시적 인텐트는 수신 컴포넌트를 명시하지 않고, 인텐트를 전달받을 수 있는 컴포넌트의 특성을 정의한다.

인텐트는 어플리케이션 간의 통신을 용이하게 하는 장점을 지니고 있지만, 그 사용에 따라서 보안 취약점을 지닐 수 있다[5-7,11-17]. 특히 암시적 인텐트는 인텐트를 받는 컴포넌트를 명시하지 않기 때문에 이를 이용한 공격들이 가능하다[2, 5-7].

이 논문에서는 암시적 인텐트의 취약점에 초점을 두고, 이에 대한 공격방법들과 대응방안을 소개한다: 좀 더 구체적으로, 이 논문은 (1)기존에 연구된 암시적 인텐트의 스푸핑 공격과 대응방안을 소개하고, (2)인텐트 인터셉트 공격을 개발자 정의 액션의 경우와 안드로이드 표준 액션의 경우로 세분화하여 각각을 소개한다. 특히, (3) *안드로이드 표준 액션*과 관련하여 이미 발견되지 않은 인터셉트 공격의 구체적인 사례를 보여주고, 그 대응 방안을 논의한다.

1.1 인텐트의 사용

컴포넌트는 어플리케이션을 구성하는 기본 요소로 안드로이드에는 액티비티(Activity), 서비스(Services), 브로드캐스트 리시버(Broadcast Receiver), 컨텐츠 프로바이더(Content Provider)라는 4가지 종류의 컴포넌트가 있다: *액티비티*는 화면에 보이며 사용자와 상호작용(interaction)하는 컴포넌트이다. *서비스*는 액티비티와 달리 사용자와 상호작용하지 않고 백그라운드에서 동작하는 컴포넌트이다. *브로드캐스트 리시버*는 브로드캐스트 이벤트를 수신하고 그에 따른 적절한 수행을 담당하는 컴포넌트이다. *컨텐트 프로바이더*는 데이터 관리와 공유를 담당하는 컴포넌트이다.

인텐트에 포함되는 정보는 컴포넌트 이름, 액션(action), 카테고리(category), 데이터(data), 타입(type), 부가정보(extra)등이 있다: *액션*은 수신 컴포넌트가 수행할 행동을 정의한다. *데이터*는 인텐트를 통해 전달되는 자료의 주소(Uniform Resource Identifier)이며 *타입*은 자료의 종류를 명시한다. *카테고리*는 수신 컴포넌트의 종류에 대한 정보로 액션, 데이터 그리고 타입과 함께 인텐트 라

우팅에 사용된다. 엑스트라는 액션, 데이터, 카테고리 이외에 부가적인 정보로 인텐트 라우팅에는 사용되지 않는다.

인텐트를 사용하기 위해서는 안드로이드 메니페스트(AndroidManifest.xml) 파일에 인텐트를 명시해야 한다. 메니페스트 파일은 어플리케이션의 구성과 관련된 모든 정보를 담고 있는 파일이며 어플리케이션의 이름, 구동을 위한 SDK 버전 정보, 실행에 필요한 권한 등 정보들을 정의하고 있다. 모든 어플리케이션은 메니페스트파일에 그 어플리케이션을 구성하는 컴포넌트들을 명시한다.

1.2 인텐트의 종류 : 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent)

서론에서 간단히 소개하였듯이 인텐트에는 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent) 두 가지 종류가 있다. 명시적 인텐트는 인텐트가 전달되어야 하는 어플리케이션을 구체적으로 명시한다. 즉, 명시적 인텐트는 받는 어플리케이션의 이름을 명시한다. 이름으로 어플리케이션 컴포넌트의 클래스 명이 사용되므로 내부 어플리케이션 사이의 통신에서 주로 사용된다.

명시적 인텐트와 다르게 암시적 인텐트는 받는 어플리케이션을 명시하지 않는다. 암시적 인텐트는 주어진 작업을 수행할 수 있는 다양한 어플리케이션들이 수신 어플리케이션의 후보가 된다. 예를 들어, 크롬(Chrome) 어플리케이션 사용 중 특정 웹 페이지 링크를 친구와 공유한다고 하자. 이때, Fig 1에서처럼 “공유”을 클릭하여 공유하는 데 사용할 어플리케이션을 선택할 수 있다. 사용자는 Fig 1-(c)에 나오는 어플리케이션들 중에서 공유하는데 사용할 어플리케이션을 클릭하여 선택한다. 이를 구현하기 위해서

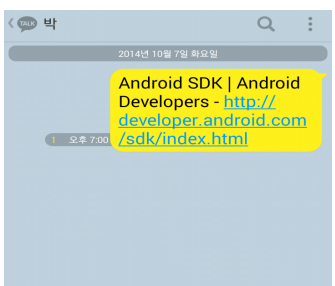


Fig. 2. Captured Screen when a message is sent via Kakaotalk Application

인터넷 어플리케이션은 암시적 인텐트를 전달한다. 또한, 공유 기능을 지원하는 어플리케이션들은 암시적 인텐트를 전달 받기 위해서 인텐트 필터를 등록한다. 위의 예에서는 Fig 1-(c)에 나오는 어플리케이션 중에서 사용자가 클릭하여 선택한 어플리케이션이 암시적 인텐트를 전달받게 된다(Fig 2).

1.3 인텐트 전달

수신하고자 하는 컴포넌트는 인텐트 필터를 등록하는데 인텐트 필터는 어플리케이션을 구성하는 각 컴포넌트가 원하는 암시적 인텐트의 수신을 위해 시스템에 공지하는 역할을 한다. 인텐트 필터는 안드로이드 메니페스트 파일에 작성되며 자신이 처리해야 할 인텐트의 특성을 명세 한다. 인텐트 필터는 어플리케이션을 구성하는 각 컴포넌트 별로 정의하며, 인텐트의 구성 요소 중 액션, 데이터, 카테고리를 명시한다. 하나의 컴포넌트가 여러 개의 인텐트 필터를 정의할 수 있고 각 인텐트 필터는 독립적으로 작동한다. 만약 인텐트에 액션과 데이터가 동시에 명시되어 있다면 인텐트 필터가 이 두 요소를 모두 만족해야 전달되고, 여러 개 인텐트 필터가 있는 경우에, 이 중 하나라도 만족하면 전달된다.

Table 1에서 볼 수 있듯이 액션에는 두 가지 종류가 있다. 위의 웹 페이지 링크 공유 예에서 사용된 액션은 “ACTION_SEND”이다. 이는 안드로이드 표준 액션에 속한다. 안드로이드는 약200개 정도의 표준 액션을 정의하여 제공한다. Table 2에 그 대표적인 예들을 나열하고 있다. 표준 액션 외에 개발자가 직접 정의한 액션을 개발자 정의 액션(developer-created action)이라고 한다.

인텐트의 수신자를 결정하는 작업은 명시적 인텐트의 경우 인텐트에 명시된 컴포넌트가 존재하면 바로 전달되고, 해당하는 컴포넌트가 존재하지 않으면 에러 처리를 하게 된다. 암시적 인텐트의 경우 안드로이드는 존재하는 모든 컴포넌트의 인텐트 필터를

Table 1. Two categories of Android Intent Action

Developer-Created Action	Developer defines an Intent Action.
Android Standard Action	Android provides a list of Standard Actions. Some important actions are introduced in Table 2.

Table 2. Android standard Actions

Action Name	Action Description
ACTION_SEND	Send some data to someone
ACTION_MAIN	Start activity as a main activity
ACTION_CALL	Call someone specified by the data
ACTION_SENDTO	Send a message to someone specified by the data
ACTION_SYNC	Run a data synchronization
ACTION_VIEW	Display the data to the user
ACTION_CHOOSER	Choose an activity

확인하여 전달한다. 이 때 수신 컴포넌트가 여러 개 존재하면, 사용자에게 수신 컴포넌트 리스트를 보여주며 컴포넌트를 선택하게 한다. 사용자가 컴포넌트를 선택하면 인텐트를 해당하는 컴포넌트에게 전달한다. 액티비티 컴포넌트가 아닌 서비스 컴포넌트는 사용자에게 보이지 않고 백그라운드에서 동작하기 때문에 안드로이드가 수신할 서비스를 선택한다.

II. 암시적 인텐트의 취약점

암시적 인텐트는 그 내용이 공개적(public)이고, 그에 따른 인텐트 필터를 정의하면 어느 어플리케이션이나 인텐트를 수신할 수 있다는 점에서 공격에 이용될 수 있는 취약성을 가진다.

암시적 인텐트의 취약점을 이용한 공격은 그 방식에 따라 크게 2가지로 나눌 수 있다. 첫 번째는 인텐트를 위조하여 수신하는 어플리케이션이 가짜 인텐트를 받도록 유도하는 공격이고, 두 번째는 인텐트가 악의적인 어플리케이션에 전달되도록 유도하는 공격이다. 첫 번째 공격을 인텐트 스푸핑(Intent Spoofing)이라고 하고 두 번째 공격을 인텐트 인터셉트(Intent Interception)라고 한다(Table 3).

Table 3. Two types of Implicit Intent Attack

Intent Interception	A malicious Application intercepts an Implicit Intent.
Intent Spoofing	A Malicious Application can launch malicious behavior by sending a forged Intent to a Normal Application.

2.1 인텐트 스푸핑

2.1.1 공격

암시적 인텐트를 이용한 인텐트 스푸핑의 대표적인 예는 과거 버전의 페이팔(Paypal)을 이용하여 소개되었다[2]. 안드로이드 페이팔 앱은 사용자가 상대방의 이메일 주소만을 이용하여 상대방에게 돈을 보낼 수 있는 기능을 제공한다. 사용자가 돈을 수신할 사람의 이메일 주소와 보내는 금액을 입력하고 "송신(Send)"를 누르면 이메일 주소의 주인에게 돈이 보내지게 된다. 이러한 페이팔(Paypal) 어플리케이션을 이용한 인텐트 스푸핑 공격은 다음과 같이 작동한다: 공격자가 공격자의 이메일 주소와 금액이 담긴 인텐트를 가짜 어플리케이션을 통해 페이팔(Paypal) 어플리케이션 내의 SendMoneyActivity 컴포넌트에 전달한다. 그러면 페이팔(Paypal) 어플리케이션은 가짜 어플리케이션에서 받은 이메일 주소와 금액을 화면에 보여주고 사용자가 "송신"을 누르면 공격자에게 해당 금액이 전송되게 된다.

2.1.2 대응방안

위의 예에서 개발자는 내부 어플리케이션 간의 통신에 개발자 정의 액션을 이용한 암시적 인텐트를 사용하여 데이터를 전달하고 있다. 디컴파일(decompile) 등을 통해 이 액션 값을 얻은 공격자가 가짜 인텐트를 만들어 자신이 만든 악성 어플리케이션에서 데이터가 전달되도록 하는 스푸핑 공격을 할 수 있게 된다.

따라서, 개발자의 인텐트 사용 의도가 내부 어플리케이션 간의 통신을 위한 것이라면 외부 어플리케이션과의 통신을 막도록 설정하여 인텐트 스푸핑을 피할 수 있다[2-5]: 즉, 메니페스트 파일(AndroidManifest.xml)에 인텐트 필터를 명시할 때 exported 값을 false로 설정하면 된다. 메니페스트 파일에 exported값이 명시되지 않은 경우에는 기본적으로(by default) exported 값은 true이고, 이 경우에는 외부에서 보내는 메시지도 전달받게 된다.

메니페스트 파일에 exported값을 false로 설정하는 것은 개발자가 인지하고 설정해야 한다. 즉, 안전한 코딩(secure coding)의 방안이 된다. 한편 안드로이드 플랫폼을 수정하여 시스템에서 어플리케이션 간의 통신을 컨트롤하도록 하는 방안도 있는데[7], 이 방안은 암시적 인텐트의 경우에는 특별한 조건이 아니면 내부 어플리케이션 간의 통신만을 허용하도록 플랫폼을 수정한다.

2.2 인텐트 인터셉트

인텐트 인터셉트 공격은 승인되지 않은 인텐트 전달(이하, Unauthorized Intent Receipt)[5, 6, 7]의 일종으로 악의적인 어플리케이션의 액티비티(혹은 서비스)가 인텐트를 가로챈다는 뜻에서 액티비티(혹은 서비스) 하이제킹(Activity Hijacking or Service Hijacking)이라고도 한다[5]. Unauthorized Intent Receipt는 악의적인 어플리케이션이 인텐트를 가로채는 모든 종류의 공격을 포함한다. 암시적 인텐트를 가로채는 액티비티 하이제킹과 서비스 하이제킹, 브로드캐스트 인텐트를 가로채는 브로드캐스트 도둑(Broadcast Thief)[1] 이 그러한 예들이다[5-7].

인텐트 인터셉트는 액션의 종류에 따라서 그 형태와 대응방안에 차이가 있다. 다음에서 각각의 공격의 구체적인 예를 제시하고, 그 대응방안을 논의한다.

2.2.1 개발자 정의 액션(developer-created action)을 사용한 인텐트의 인터셉트

Paypal의 인텐트 스푸핑 공격사례와 같이 공격자가 개발자가 정의한 액션 값을 이용하여 인텐트 인터셉트 공격을 시도할 수 있다. 스푸핑과 다른 점은 스푸핑에서는 인텐트를 위조하는 것이고, 인터셉트에서는 인텐트를 악성 어플리케이션 전달받도록 유도하는 것이다. 개발자가 정의한 액션 값을 공격자가 알게 되면 이를 사용하여 인텐트 필터를 등록할 수 있고, 인텐트를 가로채어 악용할 수 있다.

1) 우리 논문에서는 암시적 인텐트에 관련된 내용에 초점을 맞추어, 브로드캐스트 인텐트에 관련된 설명은 생략한다. 자세한 내용은 [5,7,11-16]을 참고한다.

2.2.1.1 공격

공격의 예시를 위해 간단한 로그인 어플리케이션을 만들었다. Fig 3의 왼쪽은 로그인 어플리케이션의 첫 화면이다. 이 화면에서 사용자가 아이디와 비밀번호를 입력하면 Fig 3의 오른쪽과 같이 정상적으로 로그인이 된다. 이 로그인 어플리케이션에서 로그인 화면 컴포넌트의 암시적 인텐트에 사용되는 액션 값이 "kr.ac.sju.security"로 설정되어 있다(Fig 4). 액션 값을 공격자가 디컴파일 등을 통해 알게 되어 인텐트 필터에 해당하는 액션을 등록하는 악성 어플리케이션을 만들 수 있다(Fig 5).

그러한 악성 어플리케이션이 존재하면 사용자가 아이디와 비밀번호를 입력하여 로그인을 하려고 할 때, 인텐트의 수신 어플리케이션이 2개이므로 Fig 6-(b)처럼 사용자가 인텐트를 보낼 어플리케이션을 선택하게 한다. 이 때 두 개의 어플리케이션의 이름이 똑같이 "Login"으로 보인다. 하지만 왼쪽은 악성 어플리케이션이고 오른쪽은 정상적은 어플리케이션이다.

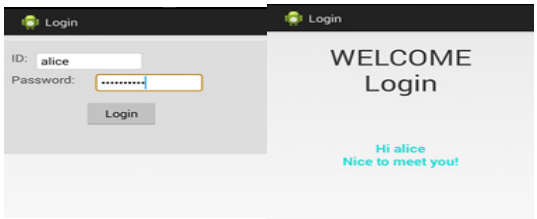


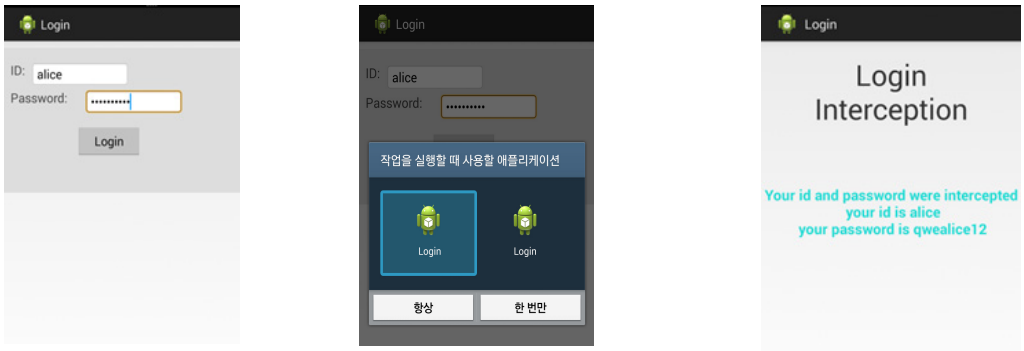
Fig. 3. Screen Captures of Normal Login Process Using the Login Application

```
Intent intent = new Intent();
intent.putExtra("id", id);
intent.putExtra("pw", pw);
intent.setAction("kr.ac.sju.security");
intent.addCategory(Intent.CATEGORY_DEFAULT);
startActivity(intent);
```

Fig. 4. Source code of an Implicit Intent with a Developer-Created Action called "kr.ac.sju.security"

```
<activity android:name = ".CatcherImplicitIntent" >
    <intent-filter >
        <action android:name="kr.ac.sju.security"/>
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Fig. 5. Source Code of CatcherImplicitIntent Activity, registering an Intent-Filter in AndroidManifest.xml file to receive the Intent with an Action called "kr.ac.sju.security"



(a) Login Screen (b) Appliction Selection Screen for Choosing Login Application (c) ID and password intercepted by a Fake Login Application

Fig. 6. Intent Interception when an Implicit Intent with a Developer-Created Action is used

실제로 악성 어플리케이션의 이름은 "Login"이 아니라 문자 앞에 공백문자가 넣어진 " Login"이다. 어플리케이션의 리스트는 특수문자, 한글, 영어 순서로 정렬되므로 악성 어플리케이션이 기존 어플리케이션보다 먼저 왼쪽에 배치되게 된다(Fig 6-(b)). 이와 같은 방법으로 사용자가 어플리케이션 선택하는데 혼란을 줄 수 있다. 이 때, 사용자가 악성 어플리케이션을 선택하면 사용자의 아이디와 비밀번호가 악성 어플리케이션에 의해 가로채어져 공격자에게 전달되게 된다(Fig 6-(c)).

위와 같은 공격은 사용자가의 두 개의 어플리케이션 중 하나를 선택해야 한다는 제약이 있다. 이때 사용자가 이상한 점을 발견하고 로그인을 중단하게 되면 위의 공격은 실패하게 된다. 하지만 이러한 진행과정에 익숙하지 않은 사용자는 의심 없이 주어진 선택을 진행할 수 있고, 이때, 악성 앱 로그인을 선택하게 되면 위의 공격이 가능하게 된다.

2.2.1.2 대응방안

개발자가 내부 어플리케이션에서만 사용하고자 하는 의도로 액션을 정의하여 암시적 인텐트를 사용하는 경우에 개발자 정의 액션이 공격자에게 노출되면 위와 같이 인텐트 인터셉트가 가능 할 수 있다. 이때 민감한 정보를 인텐트를 통해서 전달하면 민감한 정보가 유출되는 위험이 발생할 수 있다. 따라서 개발자가 내부 어플리케이션 사이의 통신에 인텐트를 사용한다면 암시적 인텐트를 사용하기보다는 명시적 인텐트를 사용하여 이를 피할 수 있다[3,4,5]. 또한, 인텐트 스푸핑에서와 마찬가지로 안드로이드 시스템에서 암시적 인텐트의 사용을 내부 어플리케이션 간의 통신을 제약하는 것도 해결방안이 된다[7]. 하지만 인텐트 스푸핑의 경우와 달리 메니페스트 파일의 exported의 값을 false로 설정하는 것은 해결방안이 되지 않는다. exported의 값이 false라는 것은 해당 컴포넌트에서 인텐트는 내부 어플리케이션에서만 받게 다는 것이다. 인텐트 스푸핑은 외부 어플리케이션에서 인텐트를 보내는 것이기 때문에 해당 방안으로 대응이 가능하지만 인텐트 인터셉트는 내부에서 보내는 인터셉트를 가로채는 것이기 때문에 이 방안은 인터셉트 공격에 적용되지 않는다.

2.2.2 표준 액션(Android Standard Action)을 사용한 인텐트의 인터셉트

2.2.2.1 공격

여기서는 안드로이드에서 제공하는 표준 액션을 사용하는 인텐트를 가로채는 공격을 살펴본다. 1.2장에 소개한 웹 페이지 링크를 공유하는 예에서 공유할 링크 정보를 전달하기 위하여 암시적 인텐트에 'ACTION_SEND'라는 표준 액션을 사용한다. 이 예에서 사용자가 카카오톡(kakao talk)을 이용해서 링크를 공유한다고 가정하자. 공격자는 카카오톡과 혼동할 수 있는 이름의 악성 어플리케이션을 구성하여 사용자가 진짜 카카오톡 대신에 가짜 카카오톡 어플리케이션(즉, 악성 앱)을 선택하도록 유도할 수 있다(Fig 7). 이 공격예시에서는 악성 어플리케이션의 이름을 'kakao'앞에 공백 문자를 넣어 ' kakao'로 구성하여, Fig 8에서와 같이 공유목록의 가장 첫

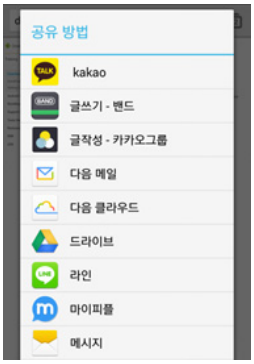


Fig. 7. A Selection list of Applications to share a link.: The kakao application in the list is fake.

번째에 오도록 만들었다. 이 때, 사용자가 " kakao"를 선택하게 되면 악성 어플리케이션이 선택되어 실행되게 된다.

위의 공격 예시에서 공격자는 악성 어플리케이션을 만들어 크롬 어플리케이션이 보내는 ACTION_SEND 액션 인텐트에 대한 인텐트 필터를 등록하여 악성 어플리케이션이 '공유'시에 공유 목록에 보이도록 하고, 사용자가 진짜 어플리케이션과 이름을 혼동하여 악성 어플리케이션의 이름을 선택하게 되면, 가짜 어플리케이션이 인텐트를 받아 동작하게 된다.

가짜 어플리케이션이 인텐트를 받게 되면 주소록, 메시지, 위치 정보 등을 포함하는 사용자의 개인 정보를 탈취하여 악성코드가 실행되어 공격자에게 전송하도록 하는 행동을 하거나 사용자 모르게 과금을 결제하여 공격자가 금전적인 이득을 얻게 할 수 있다.

혹은 가짜 어플리케이션이 사용자의 스마트폰을 공격하는 대신 사용자의 지인들에게 피싱 공격을 할 수 있다. 사용자(A)가 지인(B)에게 웹 페이지를 공유할 때 가짜 어플리케이션이 사용자가 공유하고자 하는 웹 페이지가 아닌 공격자가 수정한 웹 페이지로 사용자의 지인에게 전달되게 된다(Fig 8). 이때 지인(B)이 가짜 웹페이지 링크를 사용자(A)가 보낸 것으로 믿고 클릭하게 되면 사용자의 지인에게 피싱 공격이 이루어진다.

위의 피싱 공격 예에서 공격자의 어플리케이션이 카카오톡 어플리케이션을 호출하여 링크를 공유하기 위해서는 카카오가 제공하는 SDK를 사용하여야 한다. 카카오 SDK는 카카오 개발자[10]에서 다운받아 사용할 수 있는데, 이때 SDK를 적용할 어플리케이션을 개발자 사이트에 등록해야만 어플리케이션을 사용할 수 있다.

Fig. 8. Captured image when the Fake Application sends a shared link

또한, 카카오 SDK를 이용한 어플리케이션을 통해 링크를 공유하기 위해서는 공유할 페이지의 주소를 개발자 사이트에 미리 등록해야한다. 따라서 위의 피싱 공격의 예에서도 공격자가 유도할 가짜 웹페이지의 주소를 개발자 사이트에 등록해야 피싱 공격이 가능하다. 하지만, 개발자 사이트는 이메일 주소만으로 가입할 수 있고, 공유할 웹페이지 주소를 등록하는 데는 특별한 제약사항이 없다.

한편, 카카오 SDK를 이용한 어플리케이션을 통해 링크를 공유할 때는 Fig 8에서 볼 수 있듯이 화면에 공유하는 링크의 주소는 나타나지 않고 링크의 이름만 나타난다[2]. 이러한 링크의 이름은 개발자가 임의로 지정할 수 있다. 따라서 피싱 사이트 링크가 신뢰할 만한 지인으로부터 전달되었을 경우에 그 사이트로 유도될 가능성이 더욱 커져 피싱 공격에 더욱 취약할 수 있다.

2.2.2.2 대응방안

인텐트 인터셉트의 개념과 공격 형태는 액티비티 하이제킹으로 소개되었지만[5,7], 위와 같은 구체적인 공격 예시는 제시되지 않았다. 특히, 안드로이드 표준 액션을 이용한 인텐트를 인터셉트하는 공격에 대한 소개나 명확한 해결책은 아직 제시되지 않았다. 2.1와 2.2.1 장에서 소개한 안드로이드 플랫폼을 수정하여 어플리케이션간의 통신을 컨트롤하는 방안[7]은 개발자 정의 액션을 이용한 암시적 인텐트 인터셉트(그리고 스푸핑)에만 적용이 된다: 이 방안에서는 안드로이드 표준 액션을 이용한 인텐트의 경우에는 Unauthorized Intent Receipt의 해결책으

2) 기존의 버전에서는 링크 공유 시 링크의 이름과 함께 주소가 보였으나, 최신 버전에서는 링크의 이름만 보이고 주소는 보이지 않는 것으로 변경되었다.

로 안드로이드 표준 액션이 아닌 경우에만 외부 어플리케이션과 통신을 통제하고, 안드로이드 표준 액션을 이용한 인텐트의 경우 제약 없이 허용하고 있다.

안드로이드 표준 액션을 이용한 인텐트 인터셉트를 직접적으로 막는 해결책은 쉽게 발견하기 어렵다. 또한, 개발자 정의 액션의 경우에 제시된 해결방안들을 표준 액션에 적용하여 내부 어플리케이션 간의 통신만을 허용한다면 표준 액션의 사용이 유명무실해질 수 있으며 외부 어플리케이션 간의 통신을 쉽게 구현하기 어려워진다.

간접적이지만 궁극적인 해결책으로 다음과 같은 방안들의 활성화가 요구된다:

- 어플리케이션의 검증을 제공하여 악성 어플리케이션의 다운로드 및 설치를 막는다. 어플리케이션 설치 시 어플리케이션의 검증을 통해 해당 어플리케이션이 악성 어플리케이션인지 아닌지를 분류하여 악성 어플리케이션의 설치를 막는다. 또한, apk 파일을 다운 받을 때 악성 코드의 검증을 통해서 사용자가 다운 여부를 올바르게 판단할 수 있게 도와주어야한다.
- 어플리케이션 인증을 활성화하여 어플리케이션 다운로드 및 설치 시 뿐만이 아니라 특정 이벤트에 따른 어플리케이션 선택 시에도 사용자에게 어플리케이션의 인증 정보를 보여 준다.
- 외부 어플리케이션 간의 통신에 대하여 수신 어플리케이션의 목록을 사용자가 직접 등록 및 설정할 수 있는 환경을 제공한다. 이를 위해, 어플리케이션의 권한을 세분화하고, 사용자가 쉽게 인식할 수 있는 권한 설정 환경을 제공하여 가짜 어플리케이션이 임의로 수신 어플리케이션의 자격이 되는 것을 막는다.

또한, 어플리케이션 개발자들은 어플리케이션간의 통신을 암시적으로 사용하기보다는 검증된 API를 제공하는 어플리케이션에 대해서만 송신 어플리케이션이 데이터를 전송하도록 개발하는 것이 바람직하다.

III. 결론

본 논문에서는 암시적 인텐트의 취약점에 대한 연구로 이에 대한 공격방법들과 대응방안을 소개하였다. 먼저, 앞서 연구된 내용들을 바탕으로 암시적 인텐트의 취약점을 이용한 공격의 두 가지 종류인 인텐트 스푸핑과 인텐트 인터셉트의 개념을 소개하고, 인텐트 스푸핑 공격의 예와 대응방안을 소개하였다.

인텐트 인터셉트 공격에 대해서는 개발자 정의 액션과 안드로이드 표준 액션의 경우로 나누어, 개발자 정의 액션을 사용하는 인텐트의 취약점은 간단한 로그인 어플리케이션의 예를 통해 공격을 보여주고, 이에 대한 대응방안으로 시큐어 코딩[3-5]과 내부 어플리케이션간의 통신을 제약하는 방법[7]을 대응방안으로 소개하였다.

다음에, 이 논문의 가장 핵심이 되는 내용인, 안드로이드 표준 액션을 이용한 인텐트 인터셉트 공격을 구체적인 공격 예시를 통해 보여주었다. 이에 대한 대응방안으로 어플리케이션의 검증과 인증의 활성화에 대해 논의하고, 어플리케이션 간의 통신을 위한 API의 적극적인 활용을 제안하였다.

References

[1] Android API Reference, http://developer.android.com/reference/packages.html

[2] PALOMINO LABS, http://blog.palominolabs.com/2013/05/13/android-security/

[3] CERT Android Secure Coding, https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=111509535

[4] Android Appliction Secure Design/ Secure Coding Guidebook, http://www.jssec.org/dl/android_securecoding_en.pdf

[5] E. Chin, A.P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-app lication communication in Android," In Proceedings of the 9th international conference on Mobile systems, applications, and services, ACM, pp. 239-252 ,2011

[6] Cozzette. A, Lingel. K, Matsumoto. S, Ortlieb. O, Alexander. J, Betser. J, Florer. L, Kuenning. G, Nilles. J and Reiher. P, "Improving the security of Android inter-component communi cation," In Integrated Network Man agement, IFIP/IEEE International Symposium on pp. 808-811, 2013.

〔7〕 D. Kantola, E. Chin, W. He, and D. Wagner, "Reducing attack surfaces for intra-application communication in android," In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices, pp. 69, 2012.
〔8〕 Zdnet News, http://www.zdnet.co.kr/ news/news_view.asp?artice_id=20140731154840
〔9〕 Fortinet, http://www.fortinet.com/ sites/default/files/whitepapers/Threat-Landscape-2014.pdf
〔10〕 Kakao Developer Site , https://developers.kakao.com/
〔11〕 Young-dong Kim, Ikhwan Kim and Taehyoun Kim, "Analysis of Usage Patterns and Security Vulnerabilities in Android Permissions and Broadcast Intent Mechanism," Journal of the Korea Institute of Information Secu rity and Cryptology, 22(5), pp. 1145-1157, Oct. 2012.
〔12〕 Jae-wan Lim, Hwang-bin Ryu and Chang-Pyo Yoon, "Response Techn ique for the Vulerability of Broadcast Intent Security in Android," Korea Convergence security journal, 12(6), pp. 61-67, Dec. 2012.
〔13〕 K. Hamandi, I.H. Elhajj, A. Chehab, and A. Kayssi, "Android SMS botnet: A new perspective," In Proceedings of the 10th ACM international symposium on Mobility management and wireless access, pp. 125 - 129, 2012.
〔14〕 Yang. C, Yegneswaran. V, Porras. P, and Gu. G, "Detecting Money-Stealing Apps in Alternative Android Mark ets," In Proceedings of the 2012 ACM Conference on Computer and Comm unications Security, pp. 1034-1036, Oct. 2012.
〔15〕 Oh. J. S, Park. M. W and Chung. T. M. "The solution of denial of service attack on ordered broadcast Intent," IEEE 16th International Conference In Advanced Communication Tech nology (ICACT), pp. 397-400, 2014.
〔16〕 Kim, Dongwoo, and Jaecheol Ryou. "SecureSMS: prevention of SMS interception on Android platform," In Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, ACM, 2014.
〔17〕 KISA, "Analysis of Android Mobile Platform Security Model," Aug. 2010.
〔18〕 McAfee Report, http://www.mcafee. com/sg/resources/reports/rp-quarterly-threat-q1-2014.pdf

〈저 자 소 개〉

조 민 재 (Min Jae Jo) 학생회원
2013년 2월: 세종대학교 컴퓨터공학과 졸업
2014년 3월~현재: 세종대학교 컴퓨터공학과 석사과정
〈관심분야〉 모바일 보안, 네트워크 보안

사 진

신 지 선 (Ji Sun Shin) 정회원
2001년 2월: 서울대학교 컴퓨터공학과 졸업
2009년 5월: 메릴랜드 주립대학 (University of Maryland at College Park) 컴퓨터과학과 박사
2009년 9월~2012년 2월: 삼성 SDS 책임연구원
2012년 3월~현재: 세종대학교 조교수
〈관심분야〉 정보보호, 암호학, 컴퓨터 보안