Journal of The Korea Institute of Information Security & Cryptology(JKIISC) ISSN 1598-3986(Print)
VOL.23, NO.3, June 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.3.479

안드로이드 기반 모바일 단말 루팅 공격에 대한 이벤트 추출 기반 대응 기법*

이 형 우 † ‡
한신대학교 컴퓨터공학부

Android based Mobile Device Rooting Attack Detection and Response Mechanism using Events Extracted from Daemon Processes*

Hyung-Woo Lee † ‡
School of Computer Engineering, Hanshin Universityy

요 약

최근 삼성 갤럭시 노트 및 갤럭시 탭 10.1 등 안드로이드 기반 상용 모바일 단말을 대상으로 정상 어플리케이션인 것처럼 오픈 마켓에 배포된 악성 어플리케이션에 의한 공격들이 급증하고 있다. 공격자는 정상적인 어플리케이션에 악성코드를 삽입하여 상용 모바일 단말에 대한 루팅(Rooting) 공격을 수행한 후, 단말 내 저장된 사용자의 SMS, 전화번호부 등 개인정보와 공인인증서 등과 같은 금융정보를 외부 서버로 유출시키는 공격을 수행하게 된다. 따라서 상용 모바일 단말에 대한 악성코드 감염 여부를 판별하고 루팅 공격을 탐지 및 대응하기 위한 기법이 필요하다. 이에 본 논문에서는 안드로이드 기반 상용 모바일 단말에 대한 루팅 공격 기법에 대해 분석하고 이를 토대로 모니터링 데몬(Daemon)을 이용하여 상용 단말 내 공격 이벤트를 추출 및 수집하여 악성 어플리케이션으로 인한 공격에 능동적으로 대응하는 기법을 제시하였다.

ABSTRACT

Recently, the number of attacks by malicious application has significantly increased, targeting Android-platform mobile terminal such as Samsung Galaxy Note and Galaxy Tab 10.1. The malicious application can be distributed to currently used mobile devices through open market masquerading as an normal application. An attacker inserts malicious code into an application, which might threaten privacy by rooting attack. Once the rooting attack is successful, malicious code can collect and steal private data stored in mobile terminal, for example, SMS messages, contacts list, and public key certificate for banking. To protect the private information from the malicious attack, malicious code detection, rooting attack detection and countermeasure method are required. To meet this end, this paper investigates rooting attack mechanism for Android-platform mobile terminal. Based on that, this paper proposes countermeasure system that enables to extract and collect events related to attacks occurring from mobile terminal, which contributes to active protection from malicious attacks.

Keywords: Android, Mobile Device, Rooting Attack, Rootkit, Event Extraction, Attack Detection

접수일(2013년 2월 22일), 수정일(2013년 4월 11일), 게재확정일(2013년 5월 2일)
* 본 연구는 2012년도 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2012R1A1A2004573)

† 주저자, hwlee@hs.ac.kr
‡ 교신저자, hwlee@hs.ac.kr(Corresponding author)

I. 서 론

최근 삼성 갤럭시 노트 I(Samsung Galaxy Note I) 및 갤럭시 탭 10.1(Samsung Galaxy Tab 10.1) 등 상용 모바일 단말에서 악성 코드를 내포해서 정상 어플리케이션인 것처럼 배포된 악성 어플리케이션을 이용한 다양한 공격들이 증가하고 있다. 특히, 공격자가 악성 어플리케이션에 악성코드를 삽입하여 루팅(Rooting) 공격을 수행한 후 상용 모바일 단말 내 저장된 사용자의 SMS, 전화번호부, 공인인증서 등과 같은 개인정보 및 금융정보를 외부 서버로 유출시키는 공격으로 인해 사회적으로 큰 문제점으로 대두되고 있다[1,2,3]. 이러한 악성 어플리케이션의 확산을 막기 위해서 안드로이드 기반 모바일 단말 내 루팅 프로그램을 분석하고 악성 어플리케이션들의 특성을 분석할 필요가 있다. 이를 통해 상용 모바일 단말을 대상으로 악성 어플리케이션에 대응할 수 있는 환경을 제공할 수 있을 것으로 기대된다.

본 연구에서는 우선 상용 모바일 단말에서 구동되는 악성 어플리케이션에 대해 살펴보고 내부 작동 방식을 분석하였다. RageAgainstTheCage 루트킷[4]을 이용하여 모바일 단말에 대한 관리자 권한을 불법적으로 획득하는 모듈이 처음 공개된 이후 GingerBread API를 기반으로 발전한 GingerBreak 루트킷[5] 구조에 대해 분석하고 이를 변형한 GingerMaster[6,7] 등의 악성 루팅 모듈이 최근 발견되었다. 루팅 모듈이 포함된 악성 어플리케이션이 단말에 설치되어 실행될 경우 사용자도 모르게 단말 내에 저장된 개인정보 등이 외부로 유출될 수 있다.

이에 실험용으로 루트킷(Rootkit)을 개발하여 상용 모바일 단말에 대한 보안 취약성을 분석하였으며 직접 상용 단말을 대상으로 악성 어플리케이션을 개발하여 단말 내부에 포함된 인증서를 외부로 유출하는 것이 가능하다는 것을 알게 되었다. 이와 같은 보안 취약성에 대응하기 위해서 리눅스 커널 기반 안드로이드 플랫폼에서 발생하는 이벤트를 수집하기 위한 데몬 프로세스를 생성하였다. 개발한 데몬 프로세스를 이용하여 단말에서 발생하는 각종 이벤트 정보를 백그라운드 형태로 획득 및 수집할 수 있었다. 수집된 이벤트 정보에는 단말 내에 정상적인 어플리케이션이 구동하면서 발생하는 정상적인 이벤트도 있으나, 악성 어플리케이션에 의해서 발생하는 공격 이벤트 역시 존재하게 된다. 따라서 본 연구에서는 정상 어플리케이션으로 위장한 악성 어플리케이션에 의해서 시도되는 루팅 공격이 수행되는 과정에서 발생하는 이벤트에 대한 추출 및 수집 과정을 통해 상용 모바일 단말에 대한 보안 취약성에 능동적으로 대응할 수 있는 방법을 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서 안드로이드 기반 모바일 단말에 대한 루팅 과정을 분석하였으며 루팅을 통한 모바일 단말 공격 어플리케이션을 살펴보았다. 3장에서는 상용 모바일 단말에 대한 보안 취약성을 분석하기 위해 실험용 악성 어플리케이션을 개발하여 단말 내 저장된 개인정보 및 금융정보에 대한 유출 문제점을 분석하였다. 4장에서는 이와 같은 공격에 대한 대응 방안으로 데몬 프로세스(Daemon Process)를 기반으로 단말 내에서 발생하는 이벤트에 대한 추출 시스템에 대한 구현 및 실험 결과를 제시하였다.

II. 모바일 단말 루팅 과정 분석

루팅이란 안드로이드 플랫폼을 탑재한 모바일 단말에서 관리자 권한을 획득하는 것으로 리눅스 환경에서 안드로이드 플랫폼의 최고 권한 계정인 루트 계정을 획득해 모든 파일과 프로그램에 접근할 수 있는 과정을 의미한다. 상용 모바일 단말에 대한 공격 기법을 분석하기 위해 기존에 발견된 루트킷 기반 루팅 공격에 대해서 분석하고 최근에 발견된 안드로이드 기반 악성 어플리케이션의 내부 구조를 분석해서 상용 모바일 단말의 보안의 취약점을 제시하고자 한다. 우선 모바일 단말 사용자 자신이 직접 단말에 대한 루팅 과정을 수행하는 과정에 대해 살펴보면 다음과 같다.

2.1 사용자에 의한 모바일 단말 루팅 방식

2.1.1 공개된 루팅 어플리케이션

안드로이드는 리눅스 운영체제를 기반으로 하기 때문에 일반적으로 일반 사용자는 모바일 단말에 대한 관리자 권한을 획득할 수 없다. 하지만, 안드로이드 기반 모바일 단말에 대해 사용자 자신이 직접 루팅 과정을 수행할 경우 자신의 모바일 단말에 대해 신규 폰트 추가 및 수정, 프로세스 성능 향상 및 모바일 단말 사용자 인터페이스에 대한 변경 등의 작업을 수행할 수 있다.

이와 같이 단말 사용자에 의해 능동적으로 모바일 단말에 대한 관리자 권한을 획득할 수 있는 루팅 프로

그램으로는 SuperOneClick, Universal Androot 및 Z4root 등이 있다. 이들 어플리케이션은 모바일 단말 사용자가 직접 단말에 대한 관리자 권한을 획득하기 위해 사용하는 선의의 루팅 어플리케이션에 해당한다. 예를 들어 SuperOne Click인 경우 "Microsoft.NET Frame work 2.0" 이상의 환경에서 작동되며, 거의 모든 모바일 단말에 대한 루팅 과정 및 언루팅(un-rooting) 과정을 지원하지만 반드시 PC와 연동되어 구동된 상태에서 모바일 단말에 대한 관리자 권한을 획득하게 된다[8,9,10].

실행 과정은 다음과 같다. 우선 USB를 이용하여 모바일 단말과 PC를 연결한 후에 SuperOneClick 프로그램을 실행시켜 'Root' 버튼을 클릭한다. 이제 모바일 단말을 재부팅 한 후 모바일 단말 내에 추가로 설치된 'Superuser' 어플리케이션을 실행하여 관리자 권한을 획득하게 된다. 다른 루팅 어플리케이션인 경우 SuperOneClick과 유사한 방식으로 진행된다.

2.1.2 갤럭시 탭 10.1에 대한 루팅 과정

삼성 갤럭시 탭 10.1에 대한 루팅 과정은 단말 내부 시스템 구조 등의 변경으로 인해 수동적인 형태로 수행할 수 있다.

[그림 1] 갤럭시 탭 10.1에 대한 루팅 실행 및 확인

단말 장치에 있는 Volume Down 버튼과 Power Button을 동시에 눌러 모바일 단말에 대한 복원 모드(Recovery mode)로 진입한 후에 상용 단말에 대한 루트킷 모듈을 실행한다. 이후 루팅 과정은 [그림 1]과 같이 기존 루팅 프로그램과 동일하게 모바일 단말 내 Superuser 어플리케이션이 설치되며 adb shell을 통해 루팅 여부를 확인할 수 있다.

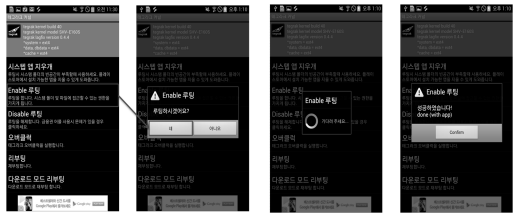
2.1.3 갤럭시 노트 I에 대한 루팅 과정

갤럭시 노트 I 상용 모바일 단말에 대한 루팅 과정을 수행하기 위해서는 'Tegrak Kernel'을 설치해야 한다. 사용자는 Tegrak Kernel을 기존의 상용 모바일 단말에 추가적으로 설치하여 정식 펌웨어의 시스템 파일과 기능은 그대로를 유지하면서도 단말 시스템 내부에 대한 변경 및 추가적인 모듈에 대한 설치가 가능한 방식이다.

Tegrak Kernel을 설치하기 위해서는 우선 삼성 통합 USB 드라이버를 설치한 후에 Odin3 소프트웨어를 설치한다. Odin3는 시스템 펌웨어를 사용자가 직접 바꿀 수 있는 기능을 제공하는 프로그램이다. 이제 상용 단말에 대한 버전 및 빌드 번호를 확인한 후에 Odin3 소프트웨어를 이용하여 각각의 모바일 단말에 해당하는 Tegrak Kernel을 설치하게 된다. 아래 그림과 같이 Odin3를 통해서 갤럭시 노트 I에 대한 커널 업그레이드 과정을 수행하게 된다.

구체적인 과정을 살펴보면 다음과 같다. 상용 모바일 단말의 sdcard 부분에 tegrak/update/ 폴더를 생성한 후 폴더 내 Tegrak-Kernel-Build*.zip 파일을 복사한 후 이를 실행하고 [그림 2]와 같이 Odin3를 구동하여 PDA 버튼 부분에서 Tegrak-Kernel-Build*.recovery.tar 파일을 실행하면 갤럭시 노트 I에 대한 복원 모드로 진입하게 된다. 복원 모드 진입 후 아래 그림과 같이 sdcard 부분에 설치된 Tegrak Kernel 모듈을 실행하게 된다.

Tegrak Kernel 프로그램은 루팅 및 언루팅 기능



[그림 2] 상용 모바일 단말 내 Tegrak Kernel 실행 과정

을 제공하며 시스템 성능 향상을 위한 오버 클럭(Over clock) 기능도 포함하고 있어 상용 단말에 대한 루팅 공격에 사용될 수 있다.

2.2 공격자에 의한 모바일 단말 루팅 방식

2.2.1 RageAgainstTheCage 및 GingerBreak 기법

앞에서 제시한 내용과 같이 모바일 단말 소유자에 의해 직접 실행되는 루팅 모듈은 악의적인 모바일 어플리케이션에 포함되어 단말 내 중요 개인정보 등을 외부로 유출하는 과정에 악용될 수 있다. 안드로이드 기반 모바일 단말에 대해 루팅 과정을 통해 관리자 권한을 획득하기 위해 원시적 코드로 널리 알려진 것이 'RageAgainstTheCage' 방식[4]이다. 이 방식은 C 언어 기반 NDK 방식으로 모바일 악성 어플리케이션에 포함되어 모바일 단말에 대한 관리자 권한을 획득하는 과정을 수행한다.

사용자가 'RageAgainstTheCage' 루트킷(Rootkit)을 내포한 악성 어플리케이션을 설치 및 실행하면, Cross Compile된 Binary 형태의 악성 코드 파일을 안드로이드 기기에 복사한 후 'chmod' 명령어를 이용하여 단말 내부 특정 폴더에 대한 접근권한을 변경하고 리눅스 쉘(Linux Shell)을 통해 400회 이상의 fork()명령어를 실행하여 프로세스를 생성한다. 이후 리눅스 커널내 버퍼 오버플로우(Buffer overflow)에 의해 더 이상 프로세스를 실행할 수 없으면 리눅스 쉘이 강제로 종료되고 이후 쉘에 접속할 때 단말 커널에 대한 관리자 권한을 획득하는 방식이다. 하지만 이와 같이 Fork() 함수를 이용하여 프로세스 복제를 수행한 후에 관리자 권한을 획득하는 기존의 'RageAgainstTheCage' 방식은 안드로이드 운영체제 2.3버전인 GingerBread로 업데이트 되면서 취약점이 수정/보완되었으나, GingerBreak라는 새로운 방식의 루트킷이 등장하게 된다.

GingerBreak 루트킷[5]은 리눅스 쉘에서 'init' 프로세스에 의해 실행되는 메시지에 대해 후킹(Message hooking) 방식을 통해 관리자 권한을 획득하게 된다. GingerBreak는 기존 변조된 su파일을 내포하고 있으며, 이를 시스템 폴더에 복사함으로써 영구적인 루팅을 수행하게 된다. 루팅 기법은 임시적인 루팅 및 영구적인 루팅으로 분류될 수 있다. 임시적인 루팅은 언루팅 과정 없이 루팅 후 시스템 재부팅을 할 경우 원상복구가 되며, 영구적인 루팅은 사용자가 언루팅 과정을 수행해야만 복구가 된다. GingerBreak에서는 영구적인 루팅을 제공하고 있으며, 임시적인 루팅과 영구적인 루팅은 su 파일의 시스템 폴더 복사 유무로 구분된다.

2.2.2 GingerMaster 기법

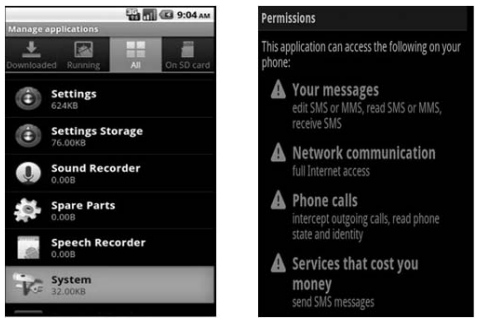
GingerMaster 루트킷 방식[5]은 GingerBreak 방식과 유사하나 루팅후 수행되는 과정에서 기존의 GingerBreak 방식보다 많은 정보를 외부로 유출, 전송하는 기능을 제공한다고 알려져 있다. 주요 작동 방식 및 구조는 GingerMaster[6,7]와 유사한 것으로 알려져 있다. 안드로이드 2.3을 대상으로 하는 루트킷으로 기존 바이러스 스캔 도구들에서 검출되지 않는 강력한 악성 소프트웨어라고 할 수 있다. GingerMaster는 일반적인 앱에 은닉/포함되어 백그라운드 형태로 추가적인 기능을 구동시키고, 안드로이드 단말 내 사용자 정보를 수집한 뒤에 특정 외부 서버로 정보를 전송하는 기능을 포함하고 있다. GingerMaster 루트킷 방식은 gbfm.png 라는 파일 이름으로 포함되어 안드로이드 단말에 대한 루트 권한을 획득하게 된다. GingerMaster는 루트 권한을 획득한 이후에 공격자가 지정한 서버에 저장된 악성 코드를 안드로이드 단말 내부로 다운로드하도록 유도한 후 단말내 주요 개인정보 등을 외부로 전송하는 기능에 적용될 수 있다.

앞에서 제시한 루트킷 기반 루팅 과정을 수행하게 되면 안드로이드 사용자는 단말 내 기본 시스템 구조에 대한 변경 등을 수행할 수 있다. 또한 안드로이드 단말 내 키패드 위치 등을 수정하거나, 단말 내 설치된 앱 등에 대해 자유롭게 수정/삭제 또는 변경 등의 과정을 수행할 수 있다.

또한 앞에서 제시한 바와 같이 더욱더 문제가 되는 것은 루팅 공격이 발생하였을 경우 단말 내에 저장된 개인정보 등이 사용자도 모르게 외부로 유출될 수 있다는 단점이 있다. 단말 내 연락처 정보, SMS 송수신 내역 그리고 웹페이지 접속 기록 등이 외부로 유출될 수 있다는 문제점 등으로 인해 루팅 공격은 매우 심각한 문제를 야기할 수 있다. 이와 같이 모바일 단말을 대상으로 한 루트킷을 포함한 악성 어플리케이션에 대해 살펴보면 다음과 같다.

2.3 안드로이드 기반 모바일 단말 악성 앱 분석

안드로이드 플랫폼을 대상으로 알려진 악성 앱 중에서 대표적인 것을 중심으로 살펴보면 다음과 같다. 첫번째로, DroidOS/Spitmo 트로이목마(SpyEye)는 〔그림 3〕과 같이 은행 어플리케이션에 악성코드를 삽입하여 구매자에게 다운로드를 유도한 다음에 상용 모바일 단말에 설치한 후 특정 숫자로 전화 걸기를 유도하여 과금을 발생시키는 방식이다. 활성화된 악성코드로 인해 트로이목마가 설치되면 단말 내 SMS 기록을 탈취해서 공격자가 지정한 C&C 서버로 전송하는 기능을 포함하고 있다〔11,12,13〕.



〔그림 3〕 DroidOS/Spitmo 트로이목마(SpyEye)

두 번째로, 〔그림 4〕와 같은 DroidDeluxe 앱인 경우 루트권한을 획득하는 악성코드가 포함된 악성 어플리케이션으로, 실행 시에 제조사, 모델명, 각 디바이스 정보 등 상용 모바일 단말에 관련된 내부 정보를 수집해서 특정 구글 계정(UA-19670793-1)으로 전송한다. 악성 앱이 실행되면 상용 모바일 단말 내 개인정보 관련 파일에 대해 읽기 및 쓰기가 가능하도록 접근모드를 변경하여 중요 개인정보가 외부로 유출된다〔14〕.

세 번째로, BaseBridge 앱인 경우 안드로이드 2.3 이전의 버전에 존재하는 취약점을 이용한 방식으로, 앱이 실행될 경우 원격 서버에 접속해서 상용 모바일 단말의 IMSI(International Mobile Subscriber Identity), OS 내부 설정 정보 등을 전송한다. 또한, SMS 관련 송수신 정보를 외부로 유출시키며 특정 SMS를 삭제하는 기능을 포함하고 있다.

마지막으로는, CruiseWind 앱인 경우 SMS를 재전송하는 악성코드가 내재되어 있으며 사용자도 모르게 FlashService가 자동으로 설치되어 원격 서버와 통신하는 기능을 포함하고 있다. 원격 서버로부터 XML로 설정된 파일을 다운로드한 후에 파일 내 인코딩된 특정 번호로 메시지를 지속적으로 전송해서 과도한 요금이 발생하도록 유도한다. 하지만 이런 과정에 대해 단말 사용자가 인지하지 못하도록 하기 위해 악성 앱에 의해 송신된 메시지는 자동으로 삭제하는 기능도 포함되어 있다.

〔그림 4〕 DroidDeluxe 악성 어플리케이션

이밖에도 SimpleEpo, Hexbot, BullMoose 등 다양한 형태의 악성 앱이 존재한다. SimpleEpo는 트로이 목마 형태의 악성 어플리케이션이며, Hexbot 악성 앱인 경우 정상적인 웹 서버에 내에 제공되는 HTML 파일에 공격 코드가 포함된 자바 스크립트를 자동으로 삽입하는 기능을 포함하고 있다. BullMoose는 Hexbot과 유사한 공격을 수행하는 악성 어플리케이션으로, Hexbot의 변종된 악성 어플리케이션이라고 볼 수 있다. 악성 어플리케이션의 세부 기능 및 작동 방식을 보다 구체적으로 살펴보면 다음과 같다.

III. 상용 모바일 단말 루팅 공격 앱 분석

상용 모바일 단말에 대한 보안 취약점을 분석하기 위해 단말에 대한 루팅 과정 수행 후 개인정보 및 금융정보를 외부로 유출시키는 실험용 악성 어플리케이션을 개발 및 구현하였다. 실험용으로 개발한 악성 어플리케이션은 대부분의 안드로이드 플랫폼 기반 상용 모바일 단말을 대상으로 구현하였다.

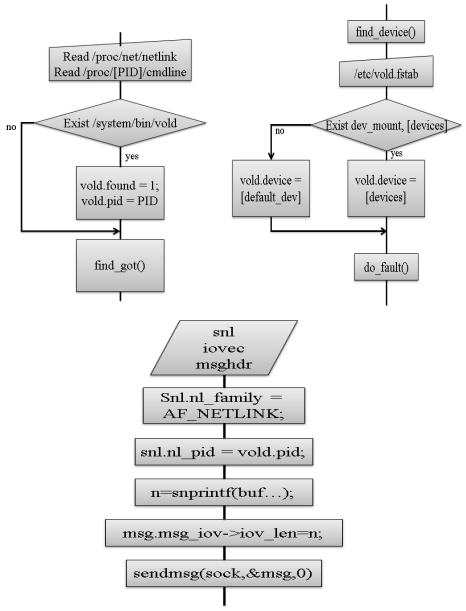
3.1 실험용 악성 앱 Andoku

본 연구에서는 실험용으로 BinBreak라는 루트킷을 개발하여 단말 내 저장된 개인정보 등을 외부로 유출하는 과정을 수행토록 하였다. 실험용으로 개발된

악성 어플리케이션을 이용하여 상용 모바일 단말에서의 보안 취약성에 대해 분석하는 과정을 수행하였다.

본 연구에서 개발한 BinBreak 루트킷은 이를 포함한 악성 어플리케이션이 시작될 경우 단말에 대한 관리자 권한을 획득하고, 이를 이용하여 단말 내에 저장된 공인 인증서 저장 폴더를 압축 하여 서버로 전송하는 기능을 제공한다. 또한 단말내 전화번호부와 웹 접속기록 등 SQLite DB 내에 저장된 정보를 외부 서버로 전송하는 기능을 포함하고 있도록 실험용으로 개발하였다.

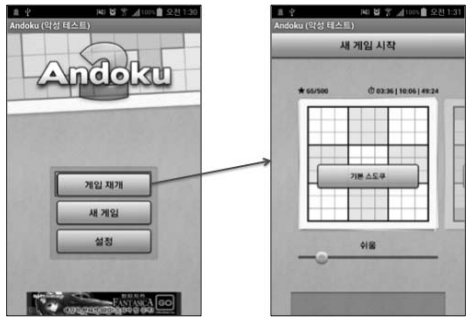
BinBreak에서의 루팅 과정은 〔그림 5〕와 같이 모바일 단말에 대해 관리자 권한을 획득하기 위해 사용되는 모듈로써 안드로이드 내의 리눅스 커널에서 수행되는 Volume Daemon을 통하여 NetLink-Message를 Spoofing 하는 방식으로 구동된다.



〔그림 5〕 BinBreak 루팅 흐름도

BinBreak를 통한 루팅 과정을 살펴보면 처음 루팅 프로세스가 시작되면 /proc/net/netlink라는 파일을 통하여 현재 실행중인 프로세스들의 ID를 받아와 /proc/〈PID〉/cmdline 파일을 통하여 Volume-Daemon의 PID를 찾는다. 그 다음 /etc/vold.fstable (Volume Daemon의 File System Table)을 통하여 Device 정보를 찾은 후 Volume Daemon의 PID를 이용하여 소켓을 연결시킨 후 특정 메시지를 보내 루트 권한을 획득한다.

본 연구에서는 상용 단말에 대한 보안 취약성을 분석하기 위해 BinBreak 루트킷을 포함한 악성 어플리케이션인 악성 Andoku 앱(〔그림 6〕)을 개발하였다. 악성 Andoku 앱인 경우 Tegrak Kernel을 기반으로 상용 모바일 단말에 대한 루팅 기능을 제공하며 상용 모바일 단말 내 저장된 정보에 대한 외부 유출 여부를 확인하는 목적으로 사용하였다. 아래 그림과 같이 최근 일반 사용자가 많이 사용하는 스도쿠(Sudoku) 응용 어플리케이션의 일종으로 Andoku라는 이름의 어플리케이션을 개발하였으며, '게임 재개' 부분을 누르게 되면 응용 프로그램에 포함되어 있는 루트킷 BinBreak가 자동으로 구동되는 방식이다.



〔그림 6〕 BinBreak 루트킷을 포함한 악성 Andoku 앱

안드로이드 기반 상용 모바일 단말에 사용자가 악성 Andoku 앱을 실행하게 되면 내부에 포함된 BinBreak 모듈이 사용자도 모르게 실행되면서 단말에 대한 관리자 권한을 획득하게 된다. 루팅 과정을 수행한 후에는 전화번호부, SMS 목록 및 인터넷 접속 기록 등 상용 모바일 단말 내에 저장된 개인정보를 외부로 유출하는 기능을 포함하고 있다. 또한 사용자

〔그림 7〕 악성 Andoku 앱에 포함된 단말 내 개인정보 압축 및 전송 모듈

가 모바일 단말을 통해서 금융 서비스를 이용할 경우 단말 내에 저장되어 있는 공인인증서를 공격자가 지정한 외부 서버로 압축하여 전송하는 기능을 포함하고 있다. 다음 〔그림 7〕과 같이 악성 Andoku 앱을 실행하였을 경우 단말 내 저장된 개인정보를 수집하여 이를 압축하고 외부로 전송하는 과정이 수행되는 것을 확인할 수 있다.

안드로이드 기반 상용 모바일 단말 내에 저장된 개인정보를 압축한 후에 외부로 전송하기 위해 아래 〔그림 8〕과 같은 ZipCompress 클래스를 추가로 개발하여 보다 효율적인 압축 전송 기능을 제공할 수 있도록 구현하였다.

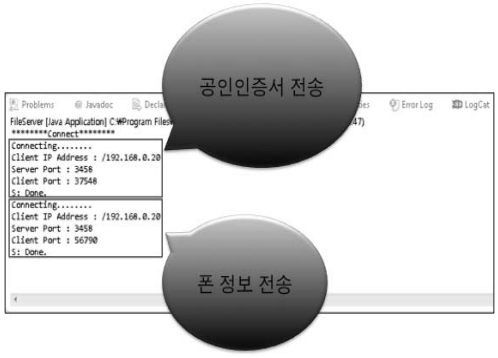
```
private static void zipEntry(File sourceFile, String sourcePath, ZipOutputStream zos) throws Exception {

    // sourceFile 이 디렉토리인 경우 하위 파일 리스트 가져와 재귀호출
    if (sourceFile.isDirectory()) {
        if (sourceFile.getName().equalsIgnoreCase(".metadata")) { // .metadata 디렉토리 return
            return;
        }
        File[] fileArray = sourceFile.listFiles(); // sourceFile 의 하위 파일 리스트
        for (int i = 0; i < fileArray.length; i++) {
            zipEntry(fileArray[i], sourcePath, zos); // 재귀 호출
        }
    } else { // sourceFile 이 디렉토리가 아닌 경우
        BufferedInputStream bis = null;
        try {
            String sFilePath = sourceFile.getPath();
            String zipEntryName = sFilePath.substring(sourcePath.length() + 1, sFilePath.length());
            bis = new BufferedInputStream(new FileInputStream(sourceFile));
            ZipEntry zentry = new ZipEntry(zipEntryName);
            zentry.setTime(sourceFile.lastModified());
            zos.putNextEntry(zentry);
            byte[] buffer = new byte[8];
            int cnt = 0;
            while ((cnt = bis.read(buffer, 0, 8)) != -1) {
                zos.write(buffer, 0, cnt);
            }
            zos.closeEntry();
        } finally {
            if (bis != null) {
                bis.close();
            }
        }
    }
}
```

〔그림 8〕 악성 Andoku 앱에 포함된 ZipCompress 클래스내 zipEntry 메소드

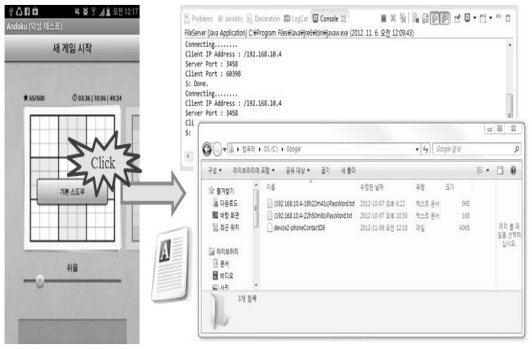
실험용 악성 Andoku 앱을 통한 개인정보 유출

악성 Andoku 앱을 실행하였을 경우 〔그림 9〕와 같이 어플리케이션에 포함된 ZipCompress 클래스를 이용하여 단말 내 개인정보가 외부 서버로 전송되는 것을 확인할 수 있다. 또한 단말 내 SQLite에 저장된 개인정보도 외부 서버로 송신되는 것을 확인할 수 있었다.

상용 모바일 단말 내에 생성 및 보관되고 있는 DB 내부 테이블 구조를 살펴보면 〔그림 10〕과 같이 이름과 전화번호 및 웹 접속 정보 등을 포함하고 있다. 따라서 악성 Andoku 앱을 실행하면 사용자도 모르게 내부 인터넷 접속 기록과 폰 내부에 저장된 전화번호부 목록 등을 외부 서버로 전송하게 된다. 또한 상용 모바일 단말 내에서 구동되고 있는 패키지 정보 등도



〔그림 9〕 사용자 단말로부터 서버로 전송된 정보 확인

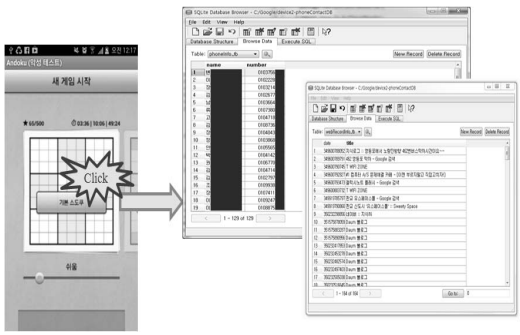


〔그림 10〕 사용자 단말로부터 서버로 전송된 전화번호부 목록 확인

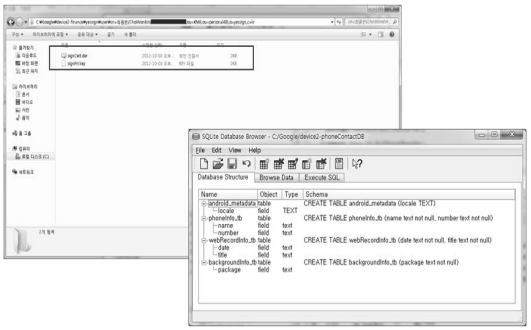
외부 서버에 전송할 수 있는 기능이 포함되어 있다.

또한, 상용 모바일 단말 내에 실행되고 있는 어플리케이션을 클릭하면 〔그림 11〕과 같이 사용자도 모르게 전화번호부 목록과 웹 서버 접속 기록 등이 외부 서버로 전송되는 것을 확인할 수 있었다.

마지막으로 〔그림 12〕와 같이 단말 내에 저장된 사용자의 공인인증서를 외부 서버로 압축하여 전송되는



〔그림 11〕 사용자 단말로부터 서버로 전송된 전화번호부 목록 및 웹 정보 확인



〔그림 12〕 공인인증서 유출 및 SQLite DB 정보 확인

것을 확인할 수 있었으며 단말 내 SQLite DB 내에 저장된 정보를 외부 서버에서 확인할 수 있었다.

이와 같이 상용 모바일 단말에 악성 어플리케이션이 설치되어 실행될 경우 내부에 포함된 루트킷이 실행되어 관리자 권한을 불법적으로 획득한 후에 단말 내 저장된 개인정보 등이 외부로 손쉽게 송신될 수 있다는 취약점이 발견되었으며 이에 대한 능동적 대응 기술에 대한 연구가 필요하다는 것을 확인할 수 있었다.

IV. 이벤트 추출 기반 악성 앱 루팅 공격 탐지 및 대응 기법

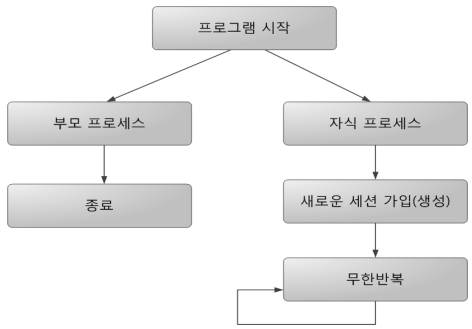
본 연구에서는 기존 기법〔16,17,18〕과 달리 루트킷을 이용한 악성 어플리케이션 기반 공격에 능동적으로 대응하기 위해서 리눅스 기반의 안드로이드 커널에 데몬 프로세스(Daemon Process)를 생성하여 악성 어플리케이션에 의해서 단말 내에서 생성되는 서비스와 프로세스 정보를 모니터링하고 이벤트를 추출 및 수집하여 이를 분석할 수 있는 시스템을 구현하였다.

구체적으로 이벤트를 추출하는 데몬 프로세스의 구조를 살펴보고 단말 내 프로세스와 서비스 정보를 수집해서 악성 이벤트를 추출하는 시스템에 대한 구현 결과를 제시하고자 한다. 본 연구에서 설계 및 구현한 어플리케이션을 'PrintDaemon'이라고 이름 붙이고 단말 내에서 구동되는 악성 어플리케이션에서 발생하는 이벤트를 효율적으로 모니터링하고 이에 능동적으로 대응할 수 있는 방법을 제시하고자 하였다.

4.1 단말 내 이벤트 추출을 위한 데몬 생성

리눅스 기반 운영체제인 안드로이드 플랫폼에서 데몬 프로세스는 해당 커널 시스템 내에서 백그라운드 형태로 특정 목적의 기능을 수행하는 프로세스로 운영체제가 종료될 때까지 실행 상태를 유지한다. 대표적인 데몬 프로세스로는 USENET에 뉴스를 전달하기 위한 데몬인 nntpd, 시스템에 로그인한 사용자의 정보를 제공하는 fingerd, 웹 서버 기능을 제공하는 httpd 및 부트 서버 기능을 하기 위한 데몬인 bootpd 등이 있다.

이처럼 다양한 기능을 제공하는 데몬 프로세스는 실행과정에서 데몬 관련 부모 프로세스로부터 자식 프로세스가 생성된 후에 부모 프로세스는 그대로 종료되고 자식 프로세스는 새로운 세션을 생성해서 자신이 리더가 된 후 무한 반복 및 실행 과정을 계속하면서 해당 서비스를 제공하게 된다. 〔그림 13〕은 데몬 프로세스의 생성 및 실행 구조를 보인다.



〔그림 13〕 데몬 프로세스 구조

데몬 프로세스를 생성하기 위해서는 크로스 컴파일(Cross Compile) 과정을 수행하여 생성할 수 있다. 크로스 컴파일 과정은 호스트 시스템과 타겟 시스템의 환경이 서로 호환되지 않을 때 서로의 환경에 맞도록 컴파일 하는 것을 의미한다. 즉, 안드로이드 플랫폼에서 해당 데몬을 적용하기 위해 NDK 기반 코딩 및 크로스 컴파일 과정을 통해 ARM 프로세스 환경에서 구동될 수 있는 데몬 프로세스를 생성하게 된다.

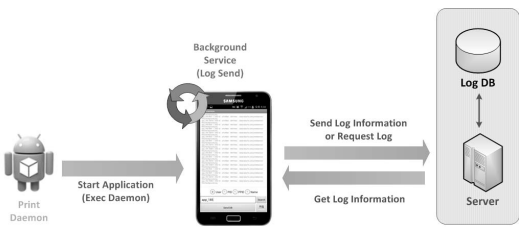
Sourcery G++ Lite을 이용하여 빌드 플랫폼 타겟이 GNU/Linux인 크로스 컴파일러를 설치한 후에 단말 내 이벤트에 대한 수집 기능을 제공하는 daemon.c 파일을 생성하고 이를 크로스 컴파일하게 된다. 컴파일 과정을 통해 생성된 루트킷 모듈(background.png)을 안드로이드 단말 내 폴더로 삽입한다. 이제 파일이 저장된 폴더에 대한 접근권한을 변경한 후에 이를 실행하면 안드로이드 단말에 새로운 데몬 프로세스가 커널내에 추가적으로 삽입되어 실행된다. 이와 같이 리눅스 커널 내에서 삽입된 이벤

트 분석 데몬 프로세스를 이용하여 단말 내에서 생성되는 서비스 및 프로세스 정보를 획득할 수 있으며, 이를 기반으로 악성 어플리케이션에 의해 발생하는 루팅 이벤트에 대한 분석이 가능하다.

4.2 데몬 기반 이벤트 추출 및 로깅 시스템

본 연구에서 설계 및 구현한 데몬 프로세스를 이용할 경우 안드로이드 기반 상용 모바일 단말 내 커널 내부에서 발생하는 프로세스 및 서비스에 대한 확인 및 검색 기능을 이용할 수 있다. 기존에는 악성 앱을 판단 하거나 단말 내 이상 현상을 모니터링 하는 기능을 제공하는 것이었으나, 본 연구에서 제시한 Print-Daemon 어플리케이션을 실행할 경우 〔그림 14〕와 같이 커널 내에 이벤트 모니터링 데몬이 백그라운드 서비스로 구동되면서 커널 내부에서 발생하는 이벤트 정보〔15〕에 대해 로그 형태로 DB 서버에 전송하는 구조이다.

기존 방식〔18〕에 대한 수정을 통해 DB 서버에는 다수의 단말에서 발생한 이벤트 정보에 대해서 수집 및 저장하는 기능을 제공하도록 하였다. 또한 수집된 이벤트 정보에 대한 분석을 통해 각 단말에 대한 루팅 공격으로 인해 발생하는 이벤트를 검출할 수 있도록 하였다.



〔그림 14〕 PrintDaemon 시스템 구조

PrintDaemon의 개발 환경으로는 〔그림 15〕와 같이 Eclipse를 이용하였으며 Java 소스가 있는 디렉토리인 src 폴더와 리소스 파일 및 메모리 번지로 mapping되는 R.java가 있는 gen 폴더, Android SDK를 포함한 라이브러리 파일, 바이너리를 그대로 보존하여 사용할 리소스가 있는 assets 폴더, Image/layout/String을 저장하는 res 폴더, Activity/Service/Permission 등에 대한 정의를 하는 AndroidManifest.xml 파일로 구성되어 있다.

〔그림 15〕 PrintDaemon 프로젝트 구조 및 실행 화면

PrintDaemon 프로그램을 구동하면 〔그림 16〕에서 제시된 형태와 같이 데몬 프로세스에 의해서 단말 내에서 발생하는 이벤트 정보를 수집하며 User, PID, PPID, Name 등으로 검색하는 기능을 구현하였고, 서버와 클라이언트는 모두 Thread로 구성해서 상위버전에서도 작동할 수 있도록 개발하였다. PrintDaemon 실행화면 내에서 Send DB 버튼을 클릭하면 현재 시점의 프로세스 정보를 DB로 전송하고 이를 확인할 수 있으며 각 사용자 단말별로 생성된 이벤트를 확인할 수 있다. 또한 각 단말에서 발생한 이벤트에 대한 PID, PPID 등에 대한 검색 기능도 제공하도록 구현하였다.

〔그림 16〕 프로세스 정보 전송

Toggle 버튼을 클릭했을 때에는 DB 전송이 활성화가 되면서 어플리케이션을 종료하여도 프로세스 정보를 계속 서버 DB로 전송하도록 구현하였으며, 어플리케이션을 다시 실행했을 경우 실행 여부에 따라서 버튼 상태가 켜짐이나 꺼짐으로 바뀌게 된다.

구현한 이벤트 모니터링 어플리케이션에서는 User, PID, PPID, Name 순으로 메인 화면에 출력이 되고 MySQL DB 내에는 User, PID, PPID, VSIZE, RSS, WCHAN, PC, NAME 순으로 저장하도록 하였다. 따라서 User, PID, PPID 및

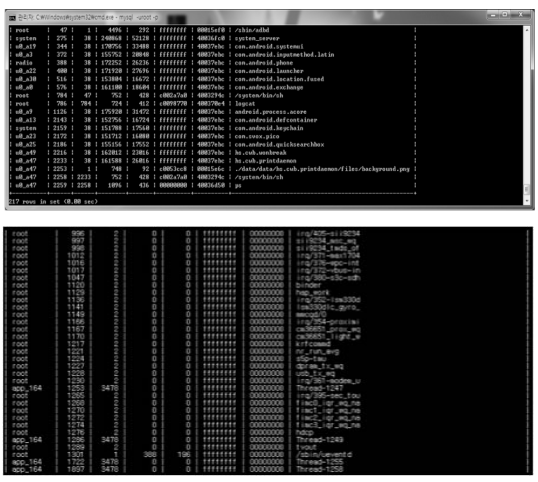
Name을 검색 조건으로 만일 악성 어플리케이션에서 발생한 이벤트 정보를 추출할 수 있도록 구현하였다.

4.3 데몬 기반 루팅 공격 이벤트 검출

이처럼 안드로이드를 기반으로 한 상용 모바일 단말의 커널 내부에서 실행을 할 수 있도록 데몬 방식으로 구동되는 모니터링 어플리케이션을 개발하여 단말 내 이벤트 발생 정보를 확인할 수 있었다. 본 연구에서 개발한 PrintDaemon 어플리케이션을 실행할 경우 안드로이드 기반의 리눅스 커널내 데몬을 실행하여 주요 프로세스의 로그 정보를 확인할 수 있었다. 또한 User와 PID, PPID, Name 중에 선택을 해서 해당 정보에 관한 로그들을 정렬할 수도 있었다.

이러한 기능들은 앞에서 살펴보았던 BinBreak 기반 Andoku와 같이 루팅 과정을 수행한 후에 단말 내 금융정보를 불법적으로 유출하는 앱 뿐만 아니라 정상 앱으로 위장한 수많은 악성 어플리케이션을 사용자가 다운받아 설치하였을 경우에 발생하는 공격 이벤트에 대한 추출 및 수집 기능을 제공하게 된다. 따라서 본 연구에서 개발한 데몬 기반 루팅 공격 이벤트 모니터링 방식을 이용할 경우 백그라운드를 통해 단말 내 서비스 및 프로세스 정보들을 모니터링하기 때문에 루팅 공격 모듈에 의해 발생하는 이벤트를 탐지할 수 있는 기능을 제공할 수 있었다.

루팅 공격이 수행되면 정상적인 단말에서의 이벤트 발생 형태와 다른 패턴으로 프로세스가 실행되는 것을 확인할 수 있었다. 본 연구에서 제시한 기법을 이용하여 악성 앱과 정상 앱에서 발생하는 PID와 PPID 이벤트 특성을 분석하였다.



〔그림 17〕 상용 단말에서의 이벤트 획득 결과

주요 특징으로는 〔그림 17〕과 같이 악성 앱인 경우 루팅 과정이 수행되면서 정상 앱 실행 과정보다도 많은 루트 권한 관련 이벤트가 발생하는 것을 확인할 수 있었으며 일부 악성 앱인 경우에는 flush-179:0 프로세스와 같이 상용 단말내 CPU 이용을 과도하게 증가시키는 이벤트가 활성화 되는 것을 확인할 수 있었다.

아래 〔그림 18〕은 서버에 수집된 이벤트 정보를 대상으로 루팅 공격을 탐지한 결과를 나타낸다. 실험용 악성 앱을 실행하여 수집된 PID와 PPID의 활성화 형태 정보를 모집단으로 하여 사용자가 일반 단말에서 앱을 실행할 경우 발생하는 PID/PPID 활성화 유사도를 측정하여 상용 단말에 대한 루팅 수행 여부를 판별토록 하였다.

본 연구에서 개발한 기법을 적용하여 모바일 단말에 직접 실험한 결과는 다음과 같이 루트킷이 포함된 악성 앱이 수행되었을 경우 이를 탐지하는 것을 확인할 수 있었다.

〔그림 18〕 상용 단말에서의 루팅 공격 탐지 결과

V. 결 론

안드로이드 기반 상용 모바일 단말에 대한 공격 형태와 악성 어플리케이션에 대해 살펴보고 이에 대한 대응 방안을 제시하였다. 먼저, 알려진 악성 어플리케이션과 불법적으로 단말에 대한 관리자 권한을 획득하기 위한 루팅 공격 에 대해 살펴보았다. 상용 모바일 단말에 대한 보안 취약성을 분석하기 위해 루트킷을 삽입한 악성 어플리케이션을 실험용으로 개발하여 단

말 내 저장된 개인정보와 인증서 등의 금융정보를 외부로 유출하는 앱을 실험용으로 개발하여 상용 단말에 대한 보안 취약성이 존재하는 것을 확인할 수 있었다.

이에 대한 대응방안으로 리눅스 커널 기반의 안드로이드 플랫폼에 이벤트 모니터링 기능을 제공하는 데몬을 생성하고 이와 연동하는 PrintDaemon 프로그램을 이용하여 단말 내에서 발생하는 프로세스 및 서비스 정보를 수집하여 백그라운드에서 단말 내 악성 어플리케이션의 실행 여부를 판단할 수 있는 어플리케이션을 구현하였다. 이와 같은 데몬 기반 이벤트 추출 방식을 통해 악성코드가 삽입된 악성 어플리케이션을 다운받은 후 실행하였을 경우 사용자 단말에 대한 악성코드 및 루팅 공격 여부를 탐지할 수 있도록 하였다.

향후 과제로는 상용 단말에서 수집된 커널내 이벤트 정보에 대한 오탐율을 최소화할 수 있는 방법과 함께 공격 탐지 성능을 보다 높일 수 있는 방법에 대한 연구가 필요하여 PID와 PPID 이외에 strace 등을 이용하여 앱 실행시 발생하는 커널내 시스템 이벤트에 대한 모니터링을 통해 루팅 공격을 탐지하는 방법에 대한 연구가 필요하다. 또한 다수의 단말에서 발생하는 이벤트 정보에 대한 상관관계를 분석하여 보다 빠르게 상용 모바일 단말에 대한 공격 여부를 판단할 수 있는 기법에 대한 연구가 필요할 것으로 판단된다.

참고문헌

〔1〕 Android rooting, http://en.wikipedia.org/wiki/Android_rooting

〔2〕 It's The Roots, get it?, http://www.androidcentral.com/root

〔3〕 Rooting – is it for me? Some Q&A, http://www.androidcentral.com/rooting-it-me-some-qa

〔4〕 Android Root Source Code: Looking at the C-Skills, http://intrepidusgroup.com/insight/2010/09/android-root-source-code-looking-at-the-c-skills/

〔5〕 Egzthunder1, Root your Gingerbread Device With Gingerbreak, April 21, 2011, from http://www.xda-developers.com/android/ root-your-gingerbread-device-with-gingerbreak/

〔6〕 Android Rooting for Programmers, http://www.apriorit.com/our-company/dev-blog/255-android-rooting

〔7〕 Android Developer Web Site, "Android.com. (2009b, December 16). What is android ?," Android Developer(http://developer.android.com/guide/basics/what-is-android.html), 2009. 12

〔8〕 Jill Duffy, "A Concise Guide to Android Rooting," pcmag (http://www.pcmag.com/article2/0,2817,2393273,00.asp), 2011. 9

〔9〕 Harron Q. Raja, "How to Root Your Android Phone/Device?,"(http://www.addictivetips.com/mobile/how-to-root-your-android-phone-device/), 2011. 1

〔10〕 Derek Scott, "Rooting for Dummies : A Begineer's Guide to Rooting Your Android Device,"(http://www.androidauthority.com/rooting-for-dummies-a-beginners-guide-to-root-your-android-phone-or-tablet-10915/), 2011. 3

〔11〕 "How and What to Root your Android : 15 Worthwhile Apps,"(http://www.tomsguide.com/us/Root-Your-Android-Phone,review-1688.html), 2011.

〔12〕 Malicious apps hosted in Google store turn Android phones into zombies http://arstechnica.com/gadgets/2012/05/malicious-apps-hosted-in-google-market-turn-android-phones-into-zombies/

〔13〕 Google now scanning Android apps for malware, http://news.cnet.com/8301-27080_3-57370650-245/google-now-scanning-android-apps-for-malware/

〔14〕 Felt, Adrienne Porte; Chin, Erika; Hanna, Steve; Song, Dawn; Wagner, David. Android Permissions Demystified.2012,http://www.cs.berkeley.edu/~afelt/android_permissions.pdf

〔15〕 Event monitoring, Wikipedia, http://en.wikipedia.org/wiki/Event_monitoring

〔16〕 Liang Xie, Xinwen Zhang, Jean-Pierre Selfert, Sencun Zhu, "pBMDS: a behavior-based malware detection system for cellphone devices," Proceeding

of the third ACM conference on Wireless network security, pp. 37-48, ACM, 2010.
〔17〕 Alexandre Bartel, Jacques Klein, Yves Le Traon, Martin Monperrus, "Automatically Securing Permission-Based Software by Reducing the Attack Surface: An Application to Android," Technical Report, University of Luxembourg, SNT, 2011.
〔18〕 Won-Jun Jang, Sik-Whan Cho, Hyung-Woo Lee, Hong-il Ju, Jeong-Nyeo Kim, "Rooting attack detection method on the Android-based smart phone," Proceeding of the International Conference on Computer Science and Network Technology, vol.1, no.1, pp. 477-481, IEEE, 2011.
〔19〕 Martin Szydlowski, Manuel Egele, Christopjer Kruegel, Giovanni Vigna, "Challenges for Dynamic Analysis of iOS Applications," iNetSec 2011, LNCS 7039, pp. 65-77, 2012

〈저자소개〉

이 형 우 (Hyung-Woo Lee) 종신회원
1994년 2월: 고려대학교 컴퓨터학과 졸업
1996년 2월: 고려대학교 컴퓨터학과 석사
1999년 2월: 고려대학교 컴퓨터학과 박사
1999년 3월~2003년 2월: 백석대학교 정보통신학부 조교수
2003년 3월~현재: 한신대학교 컴퓨터공학부 부교수, 정교수
〈관심분야〉 정보보호, 디지털 포렌식스, 스마트폰 보안