안드로이드 기반 모바일 단말 루팅 공격 검출 및 악성 앱 이벤트 모니터링☆

이 형 우*

◈ 목 차 ◈

1. 서 론

최근 안드로이드[1] 기반 모바일 단말 이용자가 급증하고 있으며 모바일 어플리케이션[2,3]은 안드로이드(Android) 마켓이나 블랙 마켓, 앱스토어 등을 통해 배포되고 있다. 하지만 모바일 단말 내 설치된 악성 어플리케이션을 통한 공격 또한 급증하고 있다[4]. 이는 안드로이드 마켓을 통해 공격자가 개발한 어플리케이션이 손쉽게 배포될 수 있기 때문이다. 특히 공격자가 배포한 어플리케이션에는 루팅(rooting) 공격[5,6,7,8]을 수행할 수 있는 악성코드가 삽입되어 있어서 이와 같은 악성 어플리케이션에 의해 단말내 개인정보에 대한 유출 및 시스템 정보에 대한 변경 공격 등이 발생하고 있다.

사용자가 정상 형태로 보이는 악성 어플리케이션을 모바일 단말에 설치 및 실행하면 adb 루트 권한 획득 후 단말 내 저장된 사용자 개인정보를 외부로 유출시킨다. 이는 모바일 단말에 대해 루팅 공격을 수행하였을 경우 공격자가 배포한 악성 어플리케이션이 관리자 권한을 획득하여 시스템 파일에 대한 공격이 가능해 진다. 루팅 공격은 다중 shell 생성을 통해 adb 루트 권한을 획득하는 것으로 모바일 단말에 대한 가장 큰 보안위협으로 대두되고 있다.

* 한신대학교 컴퓨터공학부 교수(hwlee@hs.ac.kr)
☆ 이 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2012R1A1A2004573)

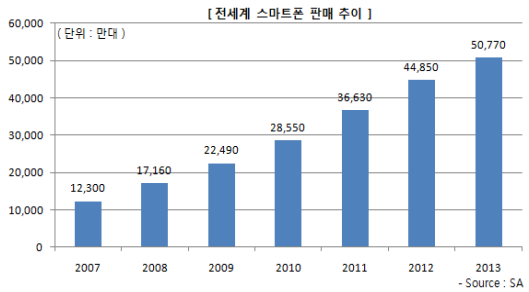
모바일 단말 시스템 파일의 취약점은 모바일 단말이 루팅 공격을 받았을 경우 관리자 권한으로 모바일 단말의 중요 데이터인 시스템 파일에 대한 공격이 가능하게 된다. 하지만 아직까지 이에 대한 대응 체계가 구축되지 못하고 있다. 따라서 모바일 단말에 대한 루팅 공격시, 가능한 공격에 대한 분석 및 대응 방안이 제시되어야 한다. 이에 모바일 단말에 대한 루팅 공격 기법에 대해 분석하고 이를 검출할 수 있는 방안에 대해 고찰하고자 한다. 또한 악성 어플리케이션에 의해서 발생하는 이벤트에 대한 보안 위험도 진단 기술에 대해 살펴보고자 한다.

2. 모바일 단말용 악성 앱 분석

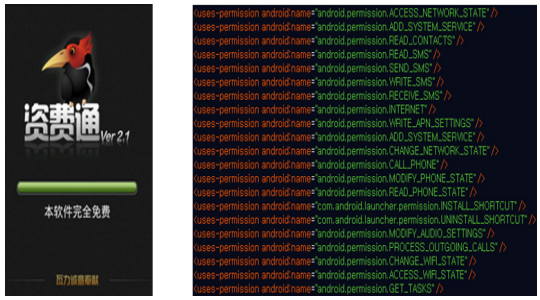
2.1 모바일 단말의 취약성

2012년 현재 전세계 IT 시장의 최대 화두는 모바일 단말이라고 해도 과언이 아니다. 전세계 스마트폰 판매 추이를 분석해 보면 2011년 3억 6천만대 정도가 판매되었으며 2013년에는 5억대 이상이 판매될 것으로 예상되고 있다.

대부분의 모바일 단말 사용자들의 이용 형태를 분석해 보면 무선 네트워크를 이용한 정보습득 및 위치정보 등을 이용한 서비스를 활용하고 있는 것으로 나타나고 있다. 하지만 모바일 단말을 사용하고 있는 사용자들인 경우 보안 문제에 대해 인식하고 있으나, 실



(그림 1) 전세계 스마트폰 판매 추이



(그림 2) Zft 앱 실행 및 내부 작동 방식

제로 보안 조치를 취하고 있는 경우는 전체 사용자의 57% 정도에 그치고 있다. 따라서 모바일 단말은 기존 PC 이용 환경 보다도 손쉽게 보안 위협에 노출되어 있는 등 심각한 문제로 대두되고 있다.

아래 표와 같이 모바일 단말은 PC의 기능을 대부분 포함하고 있으면서도 상대적으로 많은 개인정보를 포함하고 있다. 따라서 PC 보다도 보안위협이 상대적으로 높다는 특징이 있으며 기존 PC 환경에서 적용되었던 공격 방식을 약간의 수정만을 통해서 모바일 단말에 손쉽게 적용할 수 있기 때문에 보안 취약성이 많다고 할 수 있다.

(표 1) PC와 스마트폰 비교 분석

구 분	PC	스마트폰
운영체제	• Windows, Linux 등	• WinMobile, MacOS, Android 등
용도	• 인터넷(게임, Comm, SNS..), 문서 작성, 업무	• 전화통화, 문자전송(Comm,SNS..), 업무
사용시간	• 사용 후 Power Off	• 24시간 Power On
휴대성	• 낮음	• 높음
정보(데이터)	• **오피스 문서 위주**	• **전화번호, 통화내역 등 개인정보, GPS정보, 사진, 오피스 문서**
App 배포	• 유료/신뢰된 배포방식(PKG)	• 무료/OMP
중요 정보 소유자	• **전문가 중요 정보 보유**	• **비전문가도 중요 정보 보유**
성능/네트워크 속도	• **높음/높음**	• **낮음/낮음**
연결 수단/비용	• **제한/낮음**	• **다양/높음**
Property	• 우리/단체	• 개인
Trouble	• 불편/대체	• 치명적/신규
보안위협	• 악성코드 감염 • 정보유출 • 봇넷(botnet) • 접속경로(N/W, Server)	• 악성코드 감염 • 정보유출 • 봇넷(botnet) • 접속경로(N/W, Server, tethering) • **도난/분실** • **오과금**
보안수준	• 다양한 보안기능 구현 가능 • 다양한 보안제품 출시(성숙)	• 보안기능 구현 한계 • 보안제품 필요성 제기(초기)

2.2 모바일 단말용 악성 앱 현황

루팅 모듈이 포함된 악생 앱을 통해 모바일 단말내 개인 프라이버시 정보 등이 외부로 유출될 수 있는 문제점이 발생하고 있다. Android-Exploit/Zft 어플리케이션을 설치할 경우 안드로이드 기반 모바일 단말에 대해 루팅 과정을 수행하고 과다한 권한 변경 과정을 통해서 모바일 단말내 사용자 개인 정보 등을 공격자가 지정한 외부 서버로 전송하는 과정이 수행된다.

DroidKungFu 앱인 경우 마찬가지로 안드로이드폰에 대한 강제 루팅 과정을 수행하고, 루트 권한을 탈취하여 모바일 단말을 좀비폰으로 변경시키며, 결국에는 모바일 단말내 주요 정보를 공격자가 지정한 외부 서버로 사용자 모르게 전송하게 된다. 루팅 과정을 통해서 모바일 단말내 획득되는 정보로는 imei 정보, 단말 모델 정보, SD 카드 정보 및 내부 메모리 크기와 모바일 단말내 구동되는 서비스 정보 등에 해당한다.

GoldDream 앱인 경우 자동차 경주 형태의 게임 앱으로 위장하여 스마트단말내 통화 및 SMS 수신 내역을 감시하고 단말내 주요 정보를 공격자가 지정한 외부 서버로 사용자 모르게 전송하는 과정을 수행한다.

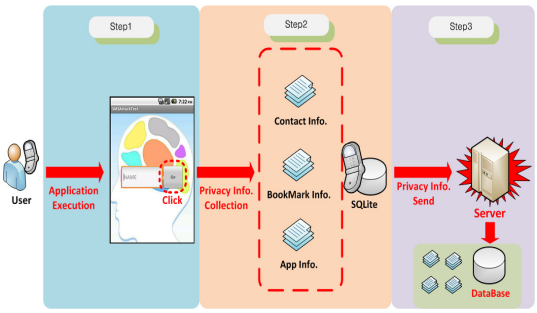
(그림 3) GoldDream 앱 실행화면

따라서 이와 같이 모바일 단말용 악성 앱의 공통점은 대부분 일반적인 형태의 서비스 또는 게임용 앱으로 위장하여 일반인들이 마켓을 통해 손쉽게 설치하

도록 유도하며 내부에는 루팅 모듈이 포함되어 있어서 결국에는 단말내 리소스에 대한 과도한 접근 및 변경, 정보에 대한 외부 유출 등의 과정이 수행된다는 것이다. 이밖에도 ZSONE, DroidDream 등 많은 종류의 악성앱이 지속적으로 발견되고 있어서 사회적인 문제도 대두되고 있다.

2.3 루팅 모듈과 연계된 악성 앱 작동 방식

루팅 모듈이 포함된 악성 앱의 작동 방식은 다음과 같다. 우선 아래 그림과 같이 일반적인 형태의 앱 인터페이스 형태로 구성되어 있으면서 사용자가 앱을 실행한 후 특정 부분에 대해 클릭 등의 과정을 수행하게 되면 내부에 은닉되어 있는 루팅 모듈이 자동적으로 실행된다. 즉 기존에 잘 알려진 앱 형태의 인터페이스로 구성되어 있으면서 기존 앱에 대한 역공학(reverse engineering) 과정을 통해 안드로이드 앱에 대한 자바 코드를 획득하고 이를 수정하여 악성 앱으로 재구성(repackaging)하는 과정을 수행하게 된다. 루팅 모듈에 의해 모바일 단말에 대한 루트 권한을 획득하게 되면 단말내 주요 정보에 대한 획득/변경 및 외부로의 유출 등의 과정을 수행할 수 있게 된다. 모바일 단말내 사용자 연락처 정보와 인터넷 접속 기록, 앱 설치 및 실행 정보 등을 획득하여 이를 외부 서버로 전송하는 과정을 수행하게 된다. 그리고 이와 같은 과정을 수행한 후에는 모바일 단말내에 설치된 루팅 관련 모듈 부분을 삭제한다. 따라서 모바일 단말에 대한 루팅 공격에 대해 살펴보면 다음과 같다.

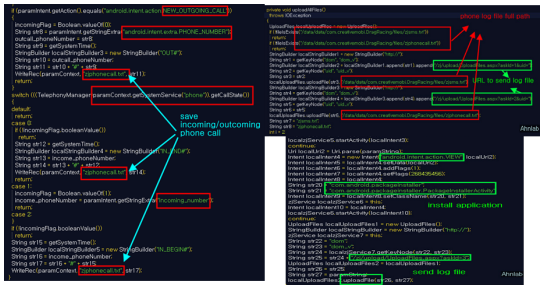


(그림 4) 루팅 모듈 포함 악성 앱 작동방식(예시)

3. 모바일 단말 루팅 공격

3.1 루팅 공격

루팅(rooting)이란 '안드로이드 기반 모바일 단말에서 운영체제를 해킹해 관리자 권한을 얻는 과정'으로 정의할 수 있다. 기존의 Exploit 기법을 안드로이드 기반 플랫폼에 적용하여 모바일 단말을 공격하는 기술이라고 할 수 있으며 루팅 공격으로 인해 단말기내 주요 정보가 사용자도 모르게 외부로 유출될 수 있다. 루팅은 사용자가 편의를 위해 기존 루팅 프로그램을 다운로드 받아 설치하는 능동적 루팅과 공격자가 악성 어플리케이션 내 루트 킷을 삽입하여 사용자도 모르게 강제적으로 모바일 단말의 관리자 권한을 획득하는 악성 루팅으로 나눌 수 있다.



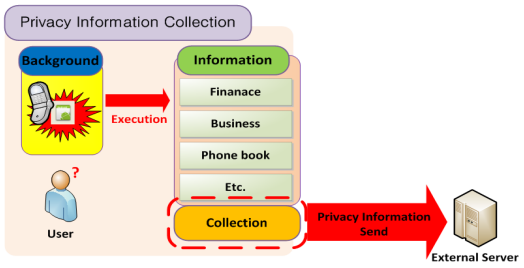
(그림 5) 루팅 공격 수행 단계

3.2 루팅 도구

모바일 단말에서 선의의 루팅 과정이 수행될 경우 모바일 단말내 CPU의 클럭을 조절하여 속도 개선 및 배터리의 효율적 사용, 폰트 수정을 통한 글꼴 변경, 불법 몰래 카메라 방지를 위한 카메라 셔터음 해제와 함께 원하는 키패드 위치값 수정, 불필요한 어플리케이션 삭제 및 구동 정지, 부팅 애니메이션 및 기타 환경 수정, 어플리케이션 외장 메모리카드 설치 등과 같이 스마트단말 소유자가 원하는 작업을 수행할 수 있다. 안드로이드 마켓을 통해 공개적으로 루팅 과정을 수행하는 앱은 SuperOneClick, Z4Root, Tegrak 및 Universal AndRoot과 같이 사용자가 직접 자신의 모바

일 단말에 대해 루팅 과정을 수행할 수 있다[9].

하지만 악의적인 루팅을 수행할 경우에는 모바일 단말 시스템 정보에 대해 수정 및 삭제, 추가, 그리고 단말에 저장된 개인정보를 외부 공격자 서버로 유출시키는 문제점이 발생할 수 있다. 결국 악의적인 루팅이 수행된 경우에 단말내 저장된 개인정보 등이 공격자가 지정한 외부 서버로 전송될 수 있다. 루팅 모듈을 포함한 악성 어플리케이션은 스마트단말내에서 백그라운드로 실행되면서 단말내 저장된 금융정보 및 업무정보, 전화번호부 및 SMS 송수신 내역 등을 외부 서버로 전송할 수 있다.



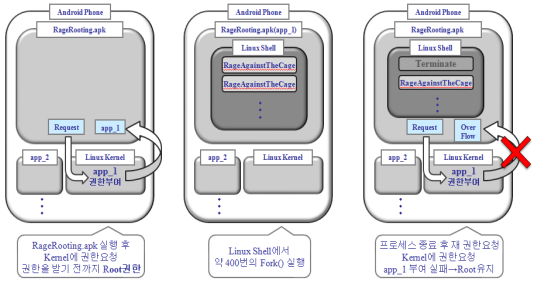
(그림 6) 루팅 공격으로 인한 개인정보 유출

3.3 루팅 프로그램 구조 분석

모바일 모바일 단말에 대한 루팅 공격이 수행되는 과정을 살펴보면 다음과 같다. 우선 루팅 프로그램은 모바일 모바일 단말 내 SD 카드를 이용하여 root shell 명령어를 수행함으로써 단말에 대한 관리자 권한을 획득한다. push 명령어를 이용하여 Exploit 코드 및 su, busybox 등을 단말내 SD 카드에 삽입한다. 그리고 adb shell 명령어를 실행한 후에 SD 카드에 삽입된 Exploit 권한을 변경하여 실행한다. 최종적으로는 root shell을 이용하여 SD 카드에 삽입된 파일을 스마트단말내 시스템 폴더로 복사하고 마지막으로 복사된 파일의 권한을 변경한다. 루팅 공격에 의한 루트 권한의 획득은 단말 내 사용자 개인정보 유출 등에 대한 공격을 유발시키며, 이는 정상 형태의 어플리케이션으로 위장하여 모바일 단말에 설치되어 악성 행위를 수행한다. 우선 RageAgainstTheCage 루팅 방식과 GingerBreak 루팅 방식에 대해 분석하면 다음과 같다.

3.4 RageAgainstTheCage[10] 루팅 구조 분석

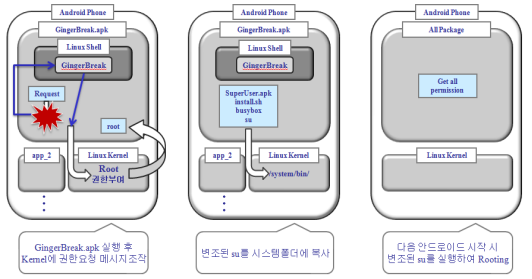
RageAgainstTheCage[10]는 C언어 기반으로 작성된 루트킷으로 사용자가 루트 킷을 내포한 악성 어플리케이션을 실행하게 되면, Cross Compile된 Binary 파일을 모바일 단말내에 복사한 후 chmod를 이용하여 권한을 변경하고 수백회 이상의 Fork() 명령어를 실행하여 프로세스를 메모리내에 지속적으로 생성하며 결국에는 버퍼 오버플로우를 발생시킨다. 이제 커널이 더 이상 프로세스를 생성할 수 없으면 리눅스 쉘이 강제로 종료되고 이후 쉘에 접속하는 과정에서 권한을 부여하는 프로세스를 실행하지 못하고 결국에는 시스템 내부의 관리자 루트 권한을 획득하는 방식으로 수행된다.



(그림 7) RageAgainstTheCage 루팅 방식

3.5 GingerBreak[11] 루팅 구조 분석

GingerBreak 루트킷을 실행하게 되면, 리눅스 쉘에서 해당 루트킷이 관리자 권한을 요청하게 되며, 요청 메시지에 대해 메시지 후킹 과정을 수행하여 관리자



(그림 8) GingerBreak 루팅 방식

권한을 획득하게 된다. GingerBreak는 변조된 su 파일을 포함하고 있으며, 이를 시스템 폴더에 복사한 후에 이를 실행하여 최종적으로 모바일 단말에 대한 루팅 과정을 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 unrooting 과정을 수행해야만 복구가 된다. GingerBreak에서는 영구적인 루팅을 제공하고 있으며, 임시적인 루팅과 영구적인 루팅은 su파일의 시스템 폴더 복사 유무로 구분된다.

3.6 GingerMaster[12] 루팅 구조 분석

GingerMaster 루트킷은 GingerBreak 방식과 유사하나 루팅후 수행되는 과정에서 기존의 GingerBreak 방식보다 많은 정보를 외부로 유출, 전송하는 기능을 제공한다고 알려져 있으며, 주요 작동 방식 및 구조는 GingerMaster와 유사한 것으로 알려져 있음. GingerMaster

```
private static void e()
  {
    ...
    String str3 = String.valueOf(b);
    String str4 = str3 + "/gb[redacted].sh  2>&1 ";
    //Execute the root exploit
    Process localProcess = Runtime.getRuntime().exec(str4);
    ...
    String str10 = String.valueOf(b);
    StringBuilder localStringBuilder3 = new StringBuilder(str10).append("/sh ");
    String str11 = b;
    StringBuilder localStringBuilder4 = localStringBuilder3.append(str11).append("/in[redacted].sh ");
    ...
    //Install the root shell
    int i9 = Runtime.getRuntime().exec(str12).waitFor();
    ...
  }

Content of /install.sh

...
cat /system/bin/sh > /data/data/com.[redacted].app[redacted]/files/sh.new
chown 0.0 /data/data/com.[redacted].app[redacted]/files/sh.new
chmod 4755 /data/data/com.[redacted].app[redacted]/files/sh.new
mount -o remount system /system

mkdir /system/xbin/app[redacted]
myuid=$1
...
chown ${myuid} /system/xbin/app[redacted]
chmod 700 /system/xbin/app[redacted]
cat /system/bin/sh > /system/xbin/app[redacted]/sh
chown 0.0 /system/xbin/app[redacted]/sh
chmod 4755 /system/xbin/app[redacted]/sh
sync
...
String str1 = ((TelephonyManager)localObject).getDeviceId();
this.f = str1;
String str2 = ((TelephonyManager)localObject).getSubscriberId();
this.g = str2;
String str3 = ((TelephonyManager)localObject).getSimSerialNumber();
this.h = str3;
String str4 = ((TelephonyManager)localObject).getLine1Number();
this.i = str4;
String str5 = String.valueOf(((TelephonyManager)localObject).getNetworkType());
this.j = str5;
FileReader localFileReader = new FileReader("/proc/cpuinfo");
BufferedReader localBufferedReader = new BufferedReader(localFileReader);
localObject = localBufferedReader.readLine();
...
localaq.a("http://client.[redacted].com/report/first_[redacted]");

#!/data/data/com.i[redacted].app[redacted]/files/sh

result=`/system/bin/pm install -r $1`
#if [ "$result" == "Success" ]; then
        if [ "$2" == "com.[redacted]" ]; then
#               sleep 2
                /data/data/com.[redacted]/files/runme.sh /system/bin/am start
                        -n com.[redacted]/com.[redacted] &
        fi
#fi
echo $result
```

(그림 9) GingerMaster 루팅 단계

는 일반적인 정상 앱에 은닉/포함되어 백그라운드 형태로 추가적인 기능을 구동시키고, 모바일 단말내 사용자 정보를 수집한 뒤에 특정 외부 서버로 정보를 전송하는 기능을 포함하고 있다.

GingerMaster는 gbfm.png라는 파일이 실행되어 안드로이드 단말에 대한 루트 권한을 획득한 이후에 C&C 서버에 접속하여 서버에 저장된 악성 코드를 단말 내부로 다운로드한 후에 사용자도 모르게 단말내에 악성코드를 설치하며 공격자에게 데이터를 전송한다.

4. 루팅 검출 기반 앱 이벤트 진단

4.1 루팅 모듈 검출 방식

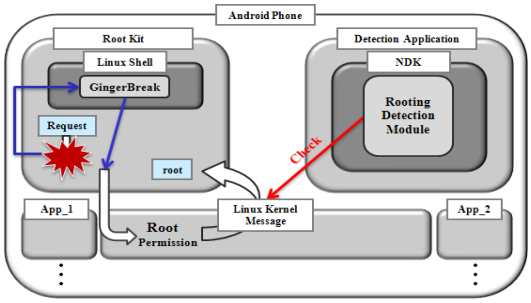
모바일 단말에 대한 효율적인 루팅 공격탐지 및 대응 메커니즘이 개발되어야 한다. 이를 위해서는 단말 내부 메모리, 프로세스 및 저장장소 등에 대한 변화 등을 실시간으로 모니터링하여 악성 앱에 의한 루팅 공격 실행 여부를 판단하고 악의적인 공격을 주기적으로 모니터링할 수 있는 메커니즘이 개발되어야 한다.

또한 모바일 단말내 설치된 활성 앱에 대해 주기적으로 모니터링하여 루팅 공격후 발생하는 악성 앱 이벤트를 모니터링하고 개인정보 유출 활동 등에 대해 대응할 수 있는 메커니즘이 개발되어야 한다[13].

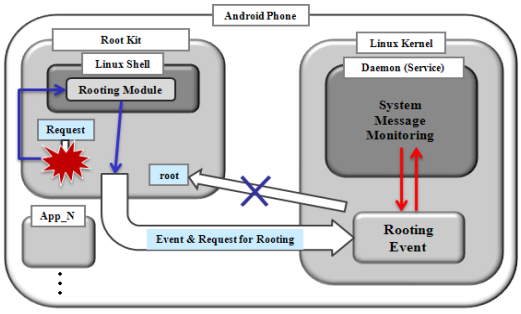
따라서 이와 같은 과정을 단계적으로 수행하기 위해서는 일차적으로 최근까지 알려진 루팅 코드에 대한 내부 구조 및 작동 방식에 대해 분석하고 주요 특징을 파악해야 한다. 또한 루팅 모듈이 포함된 악성 앱과의 연동에 의한 실행 과정을 분석하고 해당 시그니처의 특성을 파악해야 한다[14].

이에 대한 해결 방법으로는 모바일 단말내 커널을 중심으로 루팅 공격으로 인해 발생하는 이벤트 정보에 대해 모니터링 할 수 있는 방법이 필요하다. 기존 방식은 NDK 기반 루팅 공격을 탐지하거나 모니터링하는 방식으로 커널내 루팅 모듈이 실행될 경우 메시지를 검출하여 이를 분석하는 방식이다.

기존 NDK 기반 모니터링 방식의 문제점을 해결하고 보다 검출 및 대응 성능을 향상시키기 위해서는 아래 그림과 같이 루트킷이 실행되는 과정에서 발생하는 메시지에 대해서 커널내 데몬을 설치하여 GingerBreak



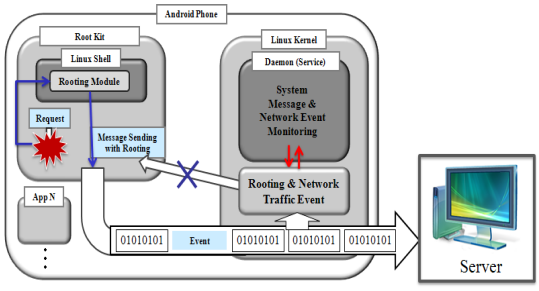
(그림 10) NDK 기반 루팅 공격 검출 방식



(그림 11) 커널내 시스템 메시지 모니터링 기반 검출 방식

와 같은 루팅 모듈에 의해 송수신되는 공격 메시지를 검출하는 방법이 개발되어야 한다[14]. 이를 통해서 루팅 모듈 수행후 단말기에서 외부로 송수신되는 정보에 대해 커널내 데몬 모듈을 개발하여 루팅 공격 시도를 실시간으로 모니터링하고, 기존 NDK 방식보다도 개선된 능동적 모니터링 방식을 제공할 수 있을 것으로 기대된다.

즉, IPC(Inter Process Communication) 메시지에 대한 모니터링을 통해서 모바일 단말에 대한 루팅 공격 시도를 판단할 필요가 있다. 루팅 공격 모듈인 경우 루트 권한을 획득하는 과정에서 Pipe 방식을 이용하여 메시지에 대한 후킹을 수행하게 된다. OpenBinder 기반의 안드로이드 IPC는 자바 코드 레벨에서 어플리케이션 간의 통신을 제공한다. 따라서 IPC 메시지 기반 루팅 공격 검출 기법은 어플리케이션에서 수행되는 Pipe 메시지에 대한 발생 회수 분석을 통해 루팅 공격을 탐지하는 방식이다. GingerBreak 방식인 경우 루팅 공격 시도와 관련된 IPC Pipe 메시지 회수 분석을 통



(그림 12) IPC 시스템 메시지 및 네트워크 이벤트 모니터링 기반 검출 방식

해 루팅 공격으로 의심되는 프로세스를 분류할 수 있다. 하지만 일반적인 프로세스 역시 Pipe 메시지가 생성되어 전송되기 때문에 IPC 기반 루팅 공격 방식에 오탐율이 증가한다는 문제점이 발생할 수 있다.

물론 가장 손쉽게 모바일 단말에 대한 루팅 공격 여부를 판단할 수 있는 방법으로는 주기적으로 su 명령어를 실행해 보는 방법이 있다. 기존의 루팅 공격 모듈인 경우에 대부분 루트 권한을 획득한 후에 수정된 su 명령어 모듈을 시스템 내부에 설치하고 이를 통해서 접근권한 등을 수정/변경하기 때문에 결국 주기적으로 모바일 단말에 대한 su 명령어 실행 여부를 체크하는 방법을 통해 감염 여부를 확인할 수 있다. 하지만 이는 너무 간단한 방법이며 궁극적인 해결책이 되지 않는다. 따라서 보다 근본적인 접근 방법으로는 루팅 공격으로 인해 발생하는 커널내 시스템 이벤트 정보에 대해 주기적으로 모니터링 하는 방법을 생각할 수 있다. 하지만 이 방법 역시 모바일 단말에 대한 부하를 증가시키는 단점이 될 수 있으며 이를 악용하여 또다른 루팅 공격 모듈로 변형이 가능하다는 문제점이 발생한다.

4.2 악성 앱 검출 방식

악성 앱에 대해 탐지하는 방법으로는 각 앱에 포함되어 있는 AndroidManifest.xml 파일 내에 정의되어 있는 리소스에 대한 접근권한 정보를 분석하여 과도한 권한 정보가 포함되어 있는지에 대해 정량적으로 분석하는 기법을 구성할 수 있다. 또한 APK 파일에 대한 역컴파일(decompile) 과정을 수행하여 기존에 알려

(그림 13) 앱 내부 루팅 모듈 탐지 결과

(그림 14) 악성 앱 모니터링 구조

진 악성코드에 관련된 시그니처의 포함 여부를 판별하는 기법을 구성할 수 있다.

따라서 이와 같은 권한 기반 악성 앱 검출 기법인 경우 모바일 단말 내 백그라운드로 실행되고 있는 앱에 설정된 서비스 권한에 대해 검사하여 악성 앱 여부를 판단하고 해당 어플리케이션을 차단하는 방식이다.

또한 루팅 모듈이 포함되어 있는 APK 파일인 경우 C 언어로 구성된 exploit를 포함하고 있으며 이를 NDK 과정을 통해 앱 내부에 삽입하였기 때문에 APK 파일에 대한 바이너리 코드 분석 과정을 수행하면 공통적으로 ELF 시그니처가 포함되어 있는 것을 확인할 수 있다. 하지만 이와 같은 시그니처 기반 악성 앱 탐지 방법은 우회 공격에 취약하다는 단점이 있다.

악성 앱을 판단하기 위해서 모바일 단말에 대한 시스템 작동 방식을 모니터링 하는 방법을 적용할 수 있다. 이 방식은 안드로이드 기반 모바일 단말의 CPU 소비율, Wi-Fi 및 3G 네트워크 등을 통해 단말 외부로 전송된 패킷의 수, 실행중인 프로세스의 수와 특징, 시스템에서 발생하는 이벤트를 주기적으로 모니터링하는 방식이다.

예를 들어 RageAgainstTheCage 루팅 방법은 무한횟수의 fork()를 실행하도록 하여 결국 단말 내 프로세스의 수를 급격히 증가 시키는 기법으로 루팅 공격을 수행하기 때문에 메모리 과다 사용 등의 악의적 행위를 탐지할 수 있다.

하지만 GingerBreak 방식은 프로그램에 의해 정상적인 형태로 보이는 이벤트를 생성하여 메시지 후킹 기반 루팅 공격을 수행하는 것이기 때문에 행위기반 대응 기법으로는 루팅 검출이 불가능하다는 단점이 있다.

5. 앱 보안 위험도 분석

5.1 앱 이벤트 상관관계 기반 분석

안드로이드 단말에서 구동되는 앱을 대상으로 앞에서 제시한 기법을 적용하여 실험한 결과는 다음과 같다. 본 연구에서 사용한 기법은 클라이언트/서버 모델로 구성되며 안드로이드 단말내에 악성 앱 진단용 도구를 설치한 후에 모바일 단말내에 설치/구동되고 있는 앱에 대한 접근권한 정보와 이벤트 정보를 서버로 전송한 후에 서버에서 해당 앱에 대한 보안 위협을 평가하는 과정으로 진행하였다.

안드로이드 단말인 경우 대략 140여개의 접근권한 정보 및 이벤트 정보가 발생한다. 따라서 각 이벤트 및 접근권한 정보에 대한 상관관계를 설정하기 위해 주요 이벤트에 대한 연관성을 가중치로 설정토록 하였다. 아래 표와 같이 안드로이드 단말에서 발생하는 이벤트는 크게 9가지 작동 방식과 형태로 나눌 수 있다. 단말 내부 리소스에 접근 및 메시지에 대해 수신하는 Access, Receive 이벤트와 리소스에 대한 정보 획득 및 설정 등에 해당하는 Get, Set 이벤트가 있다. 또한 내부 리소스에 대한 읽기, 쓰기 및 삭제 등에 해당하는 Read, Write, Delete 이벤트가 있으며 블루투스와 인터넷 관련 이벤트가 있다. 따라서 본 연구에서는 전체 이벤트간에 연관성을 분석하여 아래와 같이 상관

(표 2) 앱 이벤트 상관관계도

Event	ACCESS	RECEIVE	GET	SET	READ	WRITE	DELETE	BLUE TOOTH	INTERNET
ACCESS	1.0	0.6	0.9	0.6	0.9	0.8	0.2	0.7	0.8
RECEIVE	0.6	1.0	0.7	0.6	0.9	0.4	0.1	0.8	0.6
GET	0.9	0.7	1.0	0.5	0.8	0.5	0.2	0.7	0.4
SET	0.6	0.6	0.5	1.0	0.6	0.8	0.2	0.6	0.4
READ	0.9	0.9	0.8	0.6	1.0	0.1	0.1	0.6	0.5
WRITE	0.8	0.4	0.5	0.8	0.1	1.0	0.1	0.6	0.2
DELETE	0.2	0.1	0.2	0.2	0.1	0.1	1.0	0.2	0.1
BLUE TOOTH	0.7	0.8	0.7	0.6	0.6	0.6	0.2	1.0	0.1
INTERNET	0.8	0.6	0.4	0.4	0.5	0.2	0.1	0.1	1.0

관계표를 작성하였으며 이를 토대로 전체 이벤트에 대한 연관성을 분석하는 방법을 통해 앱에서 발생하는 이벤트에 대한 보안 위험을 정량적으로 측정하도록 설계하였다.

앞에서 제시한 내용을 토대로 앱에 대한 보안 위험도를 측정하는 단계는 다음과 같다.

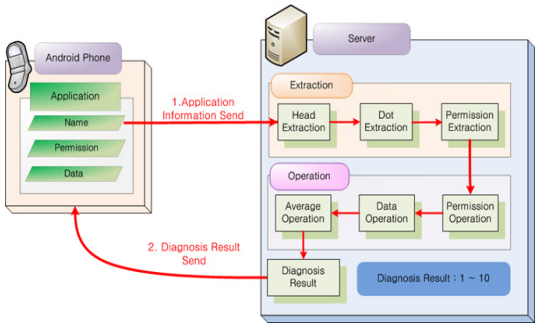
- 1단계 : 앱 실행 정보 전송
  - 실행중인 앱에 대한 정보 획득
  - 앱 이벤트 리스트 전송
- 2단계 : 앱 퍼미션 정보 분석
  - 앱에 대한 권한 설정 정보 전송
  - 권한간 상관관계 가중치 기반 분석
- 3단계 : 앱 보안 위험도 측정
  - 앱에 대한 정량적 보안 위험도 계산
  - 앱 이벤트와 연계한 위험도 측정
- 4단계 : 단람로 측정 결과 전송
  - 측정 결과를 단말로 전송
  - 단말내 실행 앱에 대한 실행 중지

5.2 보안 위험도 측정 및 실험 결과

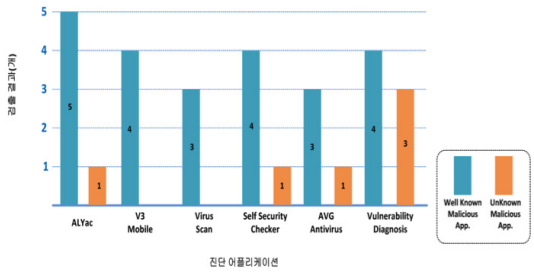
각 앱에 포함된 권한정보에 대해 단말에서 서버로 전송하면 서버에서는 이벤트 $E_i(0 \le i <= n)$를 토큰으로 분류한 후에 각 이벤트에 대해 위험도를 1부터 10의 범위 내에서 수치화한다. 만일 n개의 이벤트로부터 발생하는 m개의 이벤트 상관관계를 중심으로 추출된 가중치값 C_j를 기반으로 상대적인 위험도 $R_i = (\sum_{j=1}^{m}(1-C_j)*E_i)/m$를 계산하고, 이를 이용하여 n개의 이벤트 E_i에 대한 위험도 $T = (\sum_{i=1}^{n} Ei * Ri)/n$를 계산하여 최종적으로 앱에서 발생하는 전체 이벤트에 대한 위험도를 측정하게 된다.

기존에 알려진 악성 앱 5개와 본 연구에서 실험을 위해 별도로 작성한 5개의 악성 앱 등 모두 10개의 앱을 대상으로 탐지 성능을 평가하였다.

성능 비교를 위해 기존에 모바일 단말용 보안성 진단 및 백신 기능을 제공하는 대표적인 다섯 가지 앱 (AlYak, V3 Mobile, Virus Scan, Self Security Checker



(그림 14) 악성 앱 보안 위험도 분석 구조



(그림 15) 악성 앱 탐지 실험 결과

및 AVG Antivirus)에서의 악성 앱 검출 결과와 비교하도록 하였다. 실험 결과 본 연구에서 제시한 기법은 기존의 도구보다도 높은 탐지 성능을 보인 것을 확인할 수 있었다. 하지만 앞으로도 실험 대상을 확대하고 검출 기준과 알고리즘에 대한 보완 등의 과정을 통해서 검출 성능을 향상하기 위한 연구가 지속될 필요가 있다.

6. 결 론

본 연구에서는 최근 주요 이슈로 부각되고 있는 안드로이드 기반 모바일 단말에 보안 취약성에 대해 분석하고 루팅 공격 모듈의 특성과 작동 방식과 함께 현재까지 알려진 주요 앱에 대해 살펴보았다. 그리고 루팅 공격의 내부 구조와 작동 방식에 대한 분석을 통해서 모바일 단말내 악성 앱을 판별하는 방법을 고찰해 보았다. 모바일 단말에 설치된 앱에 포함되어 있는 내부 접근권한 정보와 앱 실행시 발생하는 이벤트 정보를 중심으로 보안 위험도를 측정할 수 있는 방법

과 성능 평가 결과에 대해 제시하였다. 앞으로도 지속적으로 모바일 단말내 공격에 능동적으로 대응할 수 있는 방법에 대한 연구가 필요하다고 판단된다.

참 고 문 헌

[1] Android.com. (2009b, December 16). What is android? Retrieved December 21, 2009, from http://developer.android.com/guide/basics/what-is-android.html

[2] Rick Rogers외 1인, “Android Application Development”, O’Reilly Media, 2009

[3] 안드로이드사이드, http://www.androidside.com/

[4] 안철수연구소, “국내 첫 모바일 단말 악성코드 피해 발생”, http://blog.ahnlab.com/ahnlab/836, 2010.

[5] Wikipedia, Rooting (Android OS), November 20, 2011, from http://en.wikipedia.org/wiki/Rooting_ (Android_OS)

[6] Haroon Q. Raja, How to Root Your Android Phone / Device, January 8, 2011, from http://www.addictivetips.com/mobile/how-to-root-your-android-phone-device/

[7] John A., What is Rooting on Android? The Advantages and Disadvantages, February 15, 2011, from http://droidlessons.com/what-is-rooting-on-android-the-advantages-and-disadvantages/

[8] Derek Scott, Rooting for Dummies: A Beginner’s Guide to Rooting your Android Device, March 22, 2011, from http://www.androidauthority.com/rooting-for-dummies-a-beginners-guide-to-root-your-android-phone-or-tablet-10915/

[9] Eric Geier, How and Why to Root your Android: 15 Worthwhile Apps, August 25, 2011, from http://www.tomsguide.com/us/Root-Your-Android-Phone,review-1688.html

[10] Thesnkchrmr, RageAgainstTheCage, March 24, 2011, from http://thesnkchrmr.wordpress.com/2011/03/24/rageagainstthecage/

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

[12] Xuxian Jiang, “GingerMaster: First Android Malware Utilizing a Root Exploit on Android 2.3 (Gingerbread),” http://www.cs.ncsu.edu/faculty/jiang/ GingerMaster/

[13] Asaf Shabtai Uri Kanonov, Yuval Elovici, Chanan Glezer, Yael Weiss, “’Andromaly’: a behavioral malware detection framewor for android devices,” Journal of Intelligent Information Systems, Vol.38, Issue 1, pp.161-190, 2012.

[14] W. Jang, S. Cho, H. Lee, H. Ju, J. Kim, “Rooting Attack Detection Method on the Android-based Smart Phone,” Proceeding of 2011 ICCSNT, IEEE, pp. 477-481, 2011.

◐ 저 자 소 개 ◑

이 형 우

1994년 고려대학교 컴퓨터학과(이학사)
1996년 고려대학교 컴퓨터학과(이학석사)
1999년 고려대학교 컴퓨터학과(이학박사)
1999년~2003년 백석대학교 정보통신학부 조교수
2003년~현재 한신대학교 컴퓨터공학부 교수
관심분야 : 정보보호, 네트워크보안, 스마트폰보안, 포렌식스 등