안드로이드 모바일 단말에서의 이벤트 수집을 통한 악성 앱 탐지 시스템 설계 및 구현[☆]

Design and Implementation of Malicious Application Detection System Using Event Aggregation on Android based Mobile Devices

함 유 정¹ 이 형 우^{1*}
You Joung Ham Hyung-Woo Lee

요 약

모바일 단말 환경이 활성화되면서 안드로이드 플랫폼을 탑재한 상용 모바일 단말이 널리 보급되고 있다. 최근 안드로이드 기반 모바일 단말에서 보안 취약성이 발견되면서 악성 어플리케이션을 통한 공격이 급증하고 있다. 대부분의 악성 어플리케이션은 오픈마켓 또는 인터넷을 통해 배포되며 어플리케이션 내에 악성코드가 삽입되어 있어 단말 사용자의 SMS, 전화번호부, 공인인증서 등 개인정보와 금융정보 등을 외부 서버로 유출시키는 공격을 시도한다. 이에 따라 상용 모바일 단말에 대한 보안 취약점 분석과 그에 따른 능동적인 대응 방안이 필요하다. 이에 본 연구에서는 최근 급증하는 악성 앱에 의한 피해를 최소화하기 위해 다수의 모바일 단말서 발생하는 이벤트 수집을 통해 모바일 단말 내에서 실행되는 악성 어플리케이션에 의한 공격을 탐지하는 시스템을 설계 및 구현하였다.

주제어 : 안드로이드, 악성 어플리케이션, 이벤트 수집, 모바일 디바이스, 악성 앱 탐지

ABSTRACT

As mobile terminal environment gets matured, the use of Android platform based mobile terminals has been growing high. Recently, the number of attacks by malicious application is also increasing as Android platform is vulnerable to private information leakage in nature. Most of these malicious applications are easily distributed to general users through open market or internet and an attacker inserts malicious code into malicious app which could be harmful tool to steal private data and banking data such as SMS, contacts list, and public key certificate to a remote server. To cope with these security threats more actively, it is necessary to develop countermeasure system that enables to detect security vulnerability existing in mobile device and take an appropriate action to protect the system against malicious attacks. In this sense, this paper aggregates diverse system events from multiple mobile devices and also implements a system to detect attacks by malicious application.

☞ keyword : Android, Malicious Application, Event Aggregation, Mobile Device, Malicious App. Detection

1. 서 론

최근 모바일 단말 사용자가 증가하고 있으며 특히 안드로이드 플랫폼(Android platform)을 탑재한 상용 모바일 단말이 널리 보급되고 있다. 하지만 안드로이드 플랫폼의 보안 취약성이 계속적으로 발견되면서 악성 어플리케이션을 통한 공격이 급증하고 있다. 상용 모바일 단말을 대

상으로 공격자는 어플리케이션 내에 악성코드를 삽입한 후 이를 오픈마켓 또는 인터넷을 통해 정상 어플리케이션인 것처럼 위장해 일반 사용자에게 배포하게 된다. 만일 사용자가 이를 자신의 단말에 설치한 후에 실행하게 되면 상용 모바일 단말 내에 저장된 SMS 송수신 정보, 전화번호부, 인터넷 접속 기록 등의 개인정보 뿐만 아니라 모바일 뱅킹 등에서 사용되는 모바일 공인인증서 등을 외부 서버로 유출시키는 공격을 시도할 수 있다는 문제점이 발견되고 있다. 이와 같은 보안위협에 능동적으로 대응하기 위해서는 상용 모바일 단말에 대한 보안 취약성을 분석하고 이를 능동적으로 탐지할 수 있는 시스템 설계 및 구현이 필요하다.

모바일 단말을 대상으로 한 보안 위협에 대응하기 위해

¹ School of Computer Engineering, Hanshin University, Gyeonggi, 447-791, Korea.

* Corresponding author (hwlee@hs.ac.kr)

[Received 12 April 2013, Reviewed 24 April 2013, Accepted 15 May 2013]

☆ 본 연구는 2012년도 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2012R1A1A2004573)

여러 가지 형태의 시스템과 기법이 제시되었다. 우선 안드로이드 플랫폼 기반 어플리케이션은 모바일 단말 내 리소스(resource)에 대한 권한(permission) 정보를 포함하고 있으며, 이를 통해 어플리케이션은 단말 내에 있는 각종 디바이스 및 서비스 등을 접근하게 된다. 따라서 어플리케이션에 포함된 권한 정보에 대한 검사를 통해 악성 여부를 판별하는 기법이 제시되었다. 하지만 기존 기법은 어플리케이션이 단말에서 직접 실행되는 과정에서 발생하는 이벤트 정보에 대한 악성 여부 판별 및 공격 검출 기능이 미약하다는 단점이 있다. 이와 같은 문제점을 해결하기 위해서 시스템 구동 과정에서의 이상 현상을 검출(anomaly detection)하거나 IPC(Inter process communication) 메시지 정보에 대한 모니터링을 통해 악성 어플리케이션의 구동 여부를 판단하는 방법도 고려해 볼 수 있다. 하지만 이 경우 공격 탐지율이 상대적으로 낮을 수 있으며 오탐율(false detection ratio)이 높아진다는 문제점이 발생하게 된다.

기존에 상용 모바일 단말 내 어플리케이션에 대한 무결성(integrity) 검증을 통해 정상적인 형태의 어플리케이션과 악성 코드 등이 포함된 어플리케이션을 판별하는 기법[1]이 제시되었다. 하지만 이 기법 역시 정적 형태의 탐지 방식(static detection)으로 상용 모바일 단말 내에서 구동되는 어플리케이션에 대한 이상 현상을 실시간으로 탐지하지 못한다는 문제가 있다. 이러한 문제점을 해결하기 위해 제시된 정적/동적 분석 기법인 경우 어플리케이션에 대한 무결성과 위험성을 정적으로 분석한 후에 실행 과정에서 발생하는 이벤트에 대한 위험도를 토대로 악성 어플리케이션을 판별하는 기법[2]이다. 하지만 정적/동적 분석 기법을 구동하기 위해서는 안드로이드 플랫폼 내에 달빅 가상기계(Dalvik Virtual Machine)에 대한 수정을 통해 어플리케이션에서 생성되는 이벤트 정보를 수집해야 한다는 문제점으로 인해 기존 상용 모바일 단말 내 커널을 수정해야 한다는 한계점이 존재한다. 이 밖에도 블랙리스트(black list)[3] 및 기본 블록(basic block)[4]을 토대로 악성 어플리케이션을 탐지하는 방법이 제시되었으나 각각의 방법 역시 상용 모바일 단말에 적용하는 과정에서 제약 사항이 있으며 문제점이 발견되고 있어 새로운 접근 방법이 필요한 시점이다.

이에 본 연구에서는 다수의 상용 모바일 단말 내에서 생성되는 이벤트 정보에 대한 추출 및 수집 과정을 통해 이를 서버에 저장한 후에 악성 어플리케이션에 의해 수행되는 공격을 탐지하는 시스템을 설계 및 구현하였다. 본 연구에서 구현한 시스템인 경우 여러 사용자의 상용 모바일 단말에서 어플리케이션을 실행하는 과정에서 발생하는

다양한 형태의 이벤트 정보를 수집하여 이를 DB에 수집 및 저장하는 과정을 수행한다. 상용 단말에서 어플리케이션이 구동되거나 단말 내 시스템 설정 등이 변경될 경우 실시간으로 발생하는 이벤트 정보를 수집하여 다수의 상용 단말에서 발생하는 정보를 비교 분석할 수 있도록 시스템을 구축하였다. 또한 수집된 정보를 기반으로 각 사용자의 단말에 대한 행위를 분석하고 악성 어플리케이션의 실행 유무 등을 확인 할 수 있도록 하였다. 이를 위해 우선 상용 모바일 단말에 대한 보안 취약점에 대한 분석 과정을 수행하여 다양한 형태의 악성 어플리케이션에 대해 효율적으로 대응할 수 있는 방식을 제시하고, 이를 기반으로 상용 모바일 단말 내 악성 어플리케이션에 의한 공격을 탐지할 수 있는 시스템을 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서 모바일 단말 보안 취약성에 대한 내용을 살펴보고, 이러한 연구 내용들의 특징을 분석한다. 3장에서는 제안하는 시스템을 제시하고, 4장에서는 설계 및 구현결과를 제시하였으며 기존 연구와의 성능 비교 분석 및 평가에 대해 서술하였다. 5장에서는 본 연구의 결론을 제시하였다.

2. 모바일 단말 보안 취약성 및 대응 시스템

본 장에서는 기존에 제시된 모바일 단말 보안 취약성에 대해 고찰하고 이를 해결하기 위해 제시되었던 기존 대응 시스템에 대해 분석하였다.

2.1 기존 모바일 단말 보안 취약성 대응 기법

안드로이드 기반 기존 모바일 단말에서 발생하는 보안 취약점을 분석하기 위한 방법을 살펴보면 다음과 같이 분류할 수 있다.

2.1.1 권한 기반 대응 기법

권한 기반 대응 기법은 사용자가 설치한 어플리케이션이 과도한 권한을 요구할 경우 모바일 단말 내 백그라운드로 실행되고 있는 서비스가 권한에 대해 검사하여 해당 어플리케이션을 차단하는 방식이다. 각각의 어플리케이션 안에는 AndroidManifest.xml 파일이 있어서 각 어플리케이션에게 허용된 권한 정보를 포함하고 있다. 만일 공격자가 악의적인 목적으로 단말 내 주요 정보를 외부로 유출하고자 하는 경우 과도한 권한 정보가 어플리케이션에 포함되어 있을 수 있다. 따라서 상용 단말에 대한 보안 취약점에 대응하기 위해 각각의 어플리케이션에 부여된 권한 정보

를 검색하여 기존에 알려진 악성 어플리케이션과의 연관성을 분석하는 방법이 제시[5,6]되었다. 하지만 이 방법은 어플리케이션 실행 과정에서 발생하는 이벤트에 대한 실시간 모니터링 기능을 포함하도록 개선되어야 하며 악성 어플리케이션 탐지 과정에 오답율이 높다는 문제점이 있다.

2.1.2 행위 기반 대응 기법

시스템이 구동되는 과정에서 발생하는 다양한 형태의 이벤트에 대한 모니터링을 통해 공격 취약점을 분석하는 기법이다. 이 방식은 안드로이드 기반 모바일 단말의 CPU 소비율, Wi-Fi 및 3G 네트워크 등에 대한 모니터링을 통해 단말에서 외부로 전송되는 패킷의 수, 실행중인 프로세스의 수와 특징 및 시스템에서 발생하는 이벤트 등을 대상으로 주기적으로 모니터링 하는 방식[7,8]이다. 이 방식은 루팅(routing) 등의 과정으로 인해 단말에 대한 오작동 등을 일부 파악할 수 있다는 장점을 제공하고 있으나, 단말 사용자가 정상적으로 수행한 작업에 대해서도 공격이라고 판단하는 경우가 있어 이에 대한 보완 등이 필요하다.

2.1.3 IPC 모니터링 기반 대응 기법

IPC 모니터링 기반 대응 기법은 IPC(Inter Process Communication)을 통해 발생하는 프로세스간의 통신 메시지의 특징 및 내용을 중심으로 모니터링 하여 모바일 단말에 대한 공격을 탐지하는 방법이다. 루팅 공격 모듈인 경우 루트 권한을 획득하는 과정에서 Pipe 방식을 이용하여 메시지 후킹(message hooking)을 수행하게 된다. OpenBinder 기반의 안드로이드 IPC는 자바 코드 레벨에서 어플리케이션 간의 통신을 제공한다. 따라서 IPC 메시지 기반 공격 검출 기법은 어플리케이션에서 수행되는 Pipe 메시지에 대한 발생 회수를 분석하여 악성 어플리케이션에 대한 실행 여부를 탐지하는 방식[9]이다. 이 방법은 시스템 작동 방식 기반 대응 방식보다도 하위 레벨의 이벤트에 대한 분석을 수행하는 것으로 일반적 시스템 이벤트와 루팅 공격으로 인해 발생하는 이벤트를 분류할 수 있는 메커니즘이 체계적으로 설계 및 구현되어야 한다.

2.2 기본 모바일 단말 보안 취약성 대응 시스템

안드로이드 기반 상용 모바일 단말에서 발생하는 보안 취약점에 대응하기 위한 기존 대응 시스템의 구조와 특징 및 개선점에 대해 고찰하면 다음과 같다.

2.2.1 어플리케이션에 대한 무결성 검증 시스템(1)

우선 어플리케이션에 대한 무결성 검증 시스템은 (그림 1)과 같이 개발자가 어플리케이션을 개발하고 마켓에 등록을 하면 마켓 내 어플리케이션을 관리하는 DB에서는 무결성 검증(integrity verification) 모듈이 실행된다. 각 어플리케이션에 대한 해시 값(hash value)을 생성해 이전에 등록되어 있는 각 어플리케이션의 해시 값과 비교하는 방식이다.

(그림 1) 어플리케이션에 대한 무결성 검증 구조도
(Figure 1) Overview of Application's Integrity Verification

어플리케이션을 사용하려는 구매자의 구매요청이 있을 때 우선 어플리케이션을 다운로드 하려는 구매자는 어플리케이션 마켓에 구매를 요청하면 구매자의 구매요청이 최초인지 아닌지의 여부를 판단한다. 만일 구매자의 구매요청이 처음이라면 검증 관리 어플리케이션을 다운로드 받아 해당 상용 모바일 단말에 설치하게 된다. 구매자의 구매요청이 처음이 아니라면 구매자가 요청한 어플리케이션과 어플리케이션 코드 검증 값 DB에 있던 어플리케이션 코드 검증 값을 모바일 단말에 다운로드 받은 후 구매자는 요청한 어플리케이션을 설치하기 전 검증관리 어플리케이션을 이용하여 다운로드 받은 요청 어플리케이션의 코드 검증 값과 비교하는 과정을 수행하게 된다. 만일 두 개의 코드 검증 값이 같다면 다운로드한 구매 어플리케이션을 상용 모바일 단말에 설치하여 사용하지만, 두 개의 코드 검증 값이 같지 않으면 구매 어플리케이션을 삭제하고 오픈 마켓(open market)에 검증 결과를 통보하는 방식이다. 전체적인 구조는 (그림 2)와 같다.

하지만 이와 같은 어플리케이션 무결성 검증 시스템인 경우 사용자가 직접 어플리케이션을 설치하고 업데이트를 수행해야 하기 때문에 어플리케이션에 대한 실시간 관리가 불가능하다. 그리고 무결성 기능만을 제공하며 보안 취약성 및 안전성에 대한 검증이 불가능하다. 이를 보완하기 위해서는 서비스 및 프로세스 검사를 수행하는 모듈 및 방

(그림 2) 어플리케이션에 대한 무결성 검증 기법 순서도
(Figure 2) Flowchart of Application's Integrity Verification

(그림 3) 정적/동적 분석 기반 악성행위 탐지 기법
(Figure 3) Static/Dynamic Analysis Method for detecting App's Malicious Behavior

식이 필요하며, 악성 어플리케이션에 대한 실시간 감지가 가능하도록 하는 기능을 갖는 시스템이 필요하다.

2.2.2 앱 악성행위 시그니처 분석 시스템(2)

앱 시그니처 분석(application signature analysis) 시스템은 아래 (그림 3)과 같이 크게 정적 분석(static analysis)과 동적 분석(dynamic analysis)의 두 가지 방식으로 수행된다. APK 파일을 실행하면 DB에 MD5, SHA-1 등의 해시 알고리즘을 이용하여 APK 파일에 대한 시그니처 분석 값을 획득한다. 정적 분석 과정에서는 어플리케이션에 대한 역어

(그림 4) 앱 시그니처 분석 구현 및 작동방식
(Figure 4) App Signature Implementation and Working Mechanism

셈블링(reassembling) 과정을 수행하며, 동적 분석 과정에서는 어플리케이션 실행 시 발생하는 시스템 내부 로그 정보를 분석하는 과정을 수행한 후 획득된 정보를 토대로 앱에 대한 위험도를 판별하게 된다.

정적 분석은 dex 파일을 역어셈블링 또는 역컴파일(recompiling) 하여 얻어진 코드 상에서 어플리케이션의 행위를 분석하는 방법으로 정적 분석을 수행하기 위해 dex 파일을 역컴파일 하는 방법과 dex 파일을 역어셈블링하는 방법을 수행한다. 먼저, dex 역컴파일을 이용한 분석 방법은 apk 파일의 압축을 해제하여 dex 파일을 추출한 후, dex 파일을 JAVA 클래스파일(.jar)로 변환해주는 역컴파일 도구를 이용하여 JAR 파일을 생성한 후에 이를 역어셈블링 및 역컴파일 하는 과정을 통해 어셈블리 코드 또는 JAVA 소스코드를 확보하여 이를 분석하는 과정을 수행한다. 두 번째, dex 역어셈블링을 이용한 분석 방법은 apk 파일로부터 추출된 dex 파일에 대하여 직접 분석하는 과정을 수행한다.

동적 분석 기법은 (그림 4)와 같이 apk 파일을 설치 및 실행하는 과정에서 안드로이드 내에 호출되는 API의 순서 및 횟수를 파악한 후에 악성행위 패턴과 비교 및 분석을 통해 해당 어플리케이션의 위험성 정도를 판단하는 방식이다. 안드로이드 플랫폼에 포함된 Dalvik 가상머신에 대한 수정을 통해 시스템 내부에서 발생하는 API System Call 관련 로그 정보를 실시간 수집 및 분석하는 과정을 수행한다. Python 언어에서 제공하는 정규표현(Regular Expression) 도구를 이용하여 악성 어플리케이션에 대한 실행 여부를 판단하게 된다.

하지만 이 시스템은 시그니처를 일부 수정했을 때 악성 어플리케이션의 탐지율이 다소 저하될 수 있고 그로 인해 새로운 공격이 발생했을 때 검출이 일부 불가능할 수 있다.

(그림 5) 상용 모바일 단말 시스템 점검

(Figure 5) System Checking for Currently Used Mobile Terminal

(그림 6) 블랙리스트 기반 악성 앱 탐지

(Figure 6) Black list based Malicious App Detection

또한 정적 분석시 패턴 매칭 중심으로 검출 과정을 수행하기 때문에 오탐율 문제가 높을 수 있으며 동적 분석 과정에 위험 API에 대한 구분 및 판단이 어렵다는 점이 있다.

2.2.3 단말 내부 점검 및 블랙리스트 기반 보안 취약성 분석 시스템(3)

블랙리스트 기반 모바일 보안 진단 시스템은 스마트폰 시스템 점검, 블랙리스트 비교 및 경고, 악성코드 채증 등 총 세 가지 기능을 제공한다. 상용 모바일 단말 내 악성코드들은 악성 프로세스를 실행하거나 주요 파일들의 권한을 변경하는 등의 행위를 하기 때문에 모바일 시스템 점검은 (그림 5)와 같이 단말 내 기초 설정 정보를 수집 및 분석하는 과정을 수행한다. 먼저, 서버가 상용 모바일 단말 내 시스템 점검 결과를 수집한다. 만일 악성코드에 감염

```

1: hA=OpenFile("a.txt",...);
2: if (hA == -1){
3:   hA = hB;
4:   flag = false;
5: }else{
6:   flag = true;
7: }
8: if (!flag) {
9:   buf = pCode;
10:  return;
11: }
12: hFd = hA;
13: nSize = GetSize(hFd);
14: ReadFile(hFd,buf,nSize,...);
    
```

(그림 7) 앱 실행시 행위 그래프 분석

(Figure 7) Execution Graph Analysis on App.

되었을 경우에는 모바일에 사용 경고 및 점검 권고 메시지를 보내고 승인을 했을 경우 사후 처리과정으로 모바일 악성코드 검사, 악성코드 채증, 블랙리스트 등록 등의 과정을 수행하며 관련 정보를 블랙리스트 DB에 저장한다.

수집된 정보를 이용하여 블랙리스트 기반 악성 앱 탐지 과정을 수행하게 된다. 구체적으로 살펴보면 (그림 6)과 같이 서버에 저장된 블랙리스트 정보와 비교하여 현재 실행되고 있는 앱에 대한 악성 여부를 판별하는 과정을 수행한다. 이미 악성코드로 식별된 앱과 사용자의 모바일에 설치된 앱을 비교해서 일치여부를 서버로 전달하게 된다. 만일 악성 앱이 설치된 경우 사용자 단말에 경고 메시지를 보내는 방식이다.

하지만 이 방식은 블랙리스트 DB 내에는 악성 앱 이력 관리 정보가 들어가는데 이 정보만으로는 악성 앱에 대한 판단이 어려울 수 있으며 또한 변형된 어플리케이션에 대한 검사가 일부 불가하다. 이를 보완하기 위해서는 예를 들어 앱에서 발생하는 네트워크 트래픽 정보에 대한 분석 과정을 함께 적용할 수 있다. 또한 변형된 어플리케이션에 대한 효율적인 판단 모듈이 추가될 필요가 있다.

2.2.4 기본 블록의 실행 추적을 통한 악성 앱 탐색 시스템(4)

기본 블록의 실행 추적을 통한 악성 앱 탐색 시스템은 BAB(Behavior Analysis based on Basic Block)를 기반으로 특정 변수에 영향을 미치는 명령어들을 추출하는 기법이다. 어플리케이션 실행시 발생하는 행위 정보를 토대로 이를 그래프로 구성하여 악성 여부를 판별하는 기능을 제공한다. 예를 들어 (그림 7)과 같이 시스템에서 발생한 OpenFile 함수와 ReadFile 함수와의 연관성을 분석하여 두 개의 API간 의존성 여부를 판단한다. BAB 분석 도구를 통해 어플리케이션 구동에 의해 실행되는 API 의존성 또는 연관성을 파

(그림 8) BAB 분석 툴
(Figure 8) BAB Analysis Tool

악하고 이를 통해 만일 행위 그래프에 오류가 있을 경우 이를 공격으로 판별하는 기법이다.

행위 그래프 분석 과정을 이용하여 (그림 8)과 같이 분석 대상 어플리케이션에 의해 활성화되는 API 간의 의존성 여부를 중심으로 악성 여부를 판별하는 방식이다.

하지만, 기본 블록의 실행 추적을 통한 악성 앱 탐색 시스템은 한정된 리소스와 계산능력을 갖는 모바일 단말에 적용할 경우 전체 시스템의 성능 저하를 유발할 가능성이 있다.

2.2.5 기존 시스템에 대한 고찰 및 개선방안

앞에서 분석한 네 가지 시스템 등에 대한 장단점 분석 과정을 통해 상용 모바일 단말에 대한 보안 취약성 과정에서 필요한 기능을 고찰하고 개선방안을 제시하고자 한다. 우선, 어플리케이션에 대한 해시 검사 기능을 제공해야 한다. MD5 또는 SHA-256 등의 해시 함수를 이용하여 단말 내 설치된 어플리케이션에 대한 무결성을 검증할 수 있는 구조 및 기능이 제공되어야 한다. 또한 단말 내 주요 시스템 파일에 대한 검사 및 모니터링 기능이 제공되어야 한다. 루팅 모듈이 포함된 어플리케이션이 설치 및 구동될 경우 내부 시스템 파일에 대한 권한 변경 여부를 수시로 확인할 필요가 있다. /system/bin 파일 리스트에 대한 주기적인 검사 과정이 수행되어야만 악성 어플리케이션에 의한 감염 여부를 확인할 수 있으며 루팅 여부도 확인할 수 있게 된다.

그리고 네트워크 트래픽 또는 네트워크 이벤트에 대한 탐지 기능을 포함하고 있어야 한다. 단말 내 프로세스에 의해 접속된 네트워크 정보를 파악하고 이를 계속적으로 모니터링 할 수 있어야 한다. 또한 단말 내 프로세스 및 서비스에 대한 분석 기능이 제공되어야 한다. 결국 상용 단말을 사용하는 과정에서 발생하는 각종 이벤트 정보를 실시간으로 수집하고 이를 분석할 수 있는 시스템을 통해 단

(그림 9) 시스템 내부 구성도 설계
(Figure 9) Design of Internal System Module

말 내 악성 앱을 탐지하는 과정에 적용할 수 있을 것으로 예상되어, 본 논문에서는 다음과 같이 상용 모바일 단말 내에서 발생하는 이벤트 정보를 수집하여 악성 앱을 탐지할 수 있는 시스템을 설계 및 구현하였다.

3. 시스템 구조 설계

기존 시스템은 단일 상용 모바일 단말에 대해서 발생하는 악성 앱의 유포 및 위험성을 탐지하는 과정을 수행하였다. 이에 본 연구에서는 다수의 상용 모바일 단말에서 수집된 시스템, 서비스, 프로세스, 및 네트워크 등의 이벤트 정보를 각각 수집하고 이에 대한 분석 및 공격 탐지 기능을 제공하는 시스템을 다음과 같이 설계하였다.

3.1 이벤트 기반 악성 앱 탐지 시스템 구조 설계

악성 앱 탐지를 위한 모듈 구성도는 (그림 9)와 같다. 보안 취약성을 점검하는 기능을 제공하도록 하였으며 서버 내에는 보안성 점검 모듈을 통해 수집된 정보를 저장 및 분석할 수 있도록 하였다. 클라이언트에는 단말 내에서 발생한 이벤트 정보를 서버로 전송할 수 있도록 하였으며 다수의 상용 단말에서 발생한 정보를 서버로 전송하도록 하였다.

전체 시스템 구성도를 살펴보면 (그림 10)과 같다. 모바일 단말 내 어플리케이션과 엔진 서버, DB 서버로 구성되어 있으며 어플리케이션은 앱 정보, 프로세스, 서비스, 루팅, 네트워크와 관련된 정보들을 서버로 전송하게 된다. 엔진 서버는 어플리케이션을 실행하기 전에 사용자에게 대한 인증 과정을 수행한 후에 최신 보안 취약성 정보를 각 사용자의 단말에 유지할 수 있도록 업데이트 기능을 제공한

(그림 10) 전체 시스템 구조도 설계
(Figure 10) Overall System Structure Design

(그림 11) 상세 시스템 구조도 설계
(Figure 11) Detailed System Structure Design

다. DB 서버는 User App, Log DB, App DB, User DB 등 네 가지의 테이블로 구성되어 사용자로부터 해당 이벤트 정보를 수집 및 저장하게 된다.

각 사용자 단말에서는 (그림 11)과 같이 실행중인 어플리케이션의 정보를 일정한 주기로 각 사용자 단말 내 SQLite DB에 저장하도록 하였으며 또한 단말 내 루팅 공격과 같은 악성 이벤트가 발생하였을 경우에는 이를 서버로 전송하는 기능을 제공하였다. 이와 함께 해당 파일에 대한 실행 위치 또는 경로(path)를 확인할 수 있도록 하였으며 명령어 실행 정보를 서버로 전송하도록 하였다.

3.2 이벤트 기반 악성 앱 탐지 시스템 작동 방식

3.2.1 엔진 업데이트, 사용자 인증 및 단말 정보 전송 과정

모바일 단말과 엔진 서버, 데이터베이스 서버 간의 작

(그림 12) 시스템 네트워크 흐름도
(Figure 12) System Network Diagram

동방식을 (그림 12)와 같이 설계하였다. 우선 모바일 단말에서 본 연구에서 개발한 악성 앱 탐지 어플리케이션을 설치하도록 하였다. 각 사용자 단말에서 본 연구에서 개발한 앱을 구동시키면 앱 내부에 포함된 정보를 토대로 엔진 서버에 연결되어 업데이트 여부를 확인하도록 하여 항상 최신의 탐지 엔진을 단말에 탑재하도록 설계하였다. 단말 내 최신 업데이트 과정을 수행한 후 사용자는 서버와의 인증 과정을 수행하도록 하였다. 본 연구에서 개발한 클라이언트용 어플리케이션을 통해 서버에 로그인 과정을 수행한 후에 각 사용자의 모바일 단말 내부 시스템 정보, 안드로이드 버전 정보, 클라이언트용 앱 버전 정보 및 엔진 업데이트 정보 등을 서버로 전송하도록 하였다.

또한 접속 과정 초기 단계에서는 각 사용자 단말에 설치된 어플리케이션 정보, 현재 상태에서 실행중인 프로세스 정보, 제공되는 서비스 정보를 서버로 전송하도록 하였으며 단말에 대한 루팅 상태 정보를 서버로 전송하도록 하여 각 사용자 단말에 대한 초기 상태를 저장/확인할 수 있도록 하였다. 이와 함께 현재 사용 가능한 네트워크 디바이스 정보도 서버로 전송하도록 하였고 전체 정보는 각 사용자 단말 내 SQLite DB에도 저장토록 하였다. 서버로 전송된 정보는 서버내 App DB에 저장되며 사용자 접속시 이전 상태와 비교할 수 있는 기능을 제공하였다.

3.2.2 다중 단말로 부터의 앱 실행 정보 전송 과정

앞에서 제시한 초기 연결 과정을 수행한 후에는 (그림 13)과 같이 각 사용자 단말에서 실행되는 일반 앱에 대한 실행 정보를 서버로 전송 및 수집하는 과정을 수행하게 된

(그림 13) 모바일 단말과 DB 서버 작동방식

(Figure 13) Diagram between Mobile Device and DB Server

(그림 14) 다수 모바일 단말에 대한 이벤트 수집 및 분석
(Figure 14) Event Aggregation and Analysis from Multiple Mobile Devices

다. 각 사용자 단말 내에는 앱이 실행될 때마다 단말 내 SQLite DB에 해당 앱의 실행 시간 정보 등을 저장하고 또한 현재 시점에서 활성화된 실시간 프로세스 정보, 서비스 구동 정보 등도 서버에 저장하게 된다. 빈번한 전송으로 인해 발생할 수 있는 모바일 단말에 대한 부하를 최소화하기 위해서 서버로의 정보 전송 주기를 설정할 수 있도록 하였다. 기본 설정은 1분 단위로 수집된 단말 내부 앱 관련 정보를 전송하도록 하였다.

이와 같은 과정을 통해 기존 기법에서 제공하였던 앱에 대한 무결성 검증을 제공할 수 있으며 strace 등의 시스템 이벤트 모니터링 도구를 통해 서버에 수집된 정보를 대상으로 시그니처 기반 분석 기법을 적용하게 된다. (그림 14)와 같이 다중 사용자 단말로부터 수집된 정보를 대상으로 동적 분석 과정을 제공하기 위해 단말 내에서 수행되는

PID 및 PPID 기반 프로세스 정보를 수집하고 구동되는 서비스로부터 활성화되는 strace 기반 시스템 이벤트 정보를 대상으로 비교 분석하는 과정을 수행하게 된다. 또한 서버에는 악성 어플리케이션이 실행되는 과정에서 발생한 이벤트 패턴 집합을 저장하고 있어서 사용자 단말로부터 전송된 이벤트와의 유사도 비교 과정을 수행하게 된다.

4. 시스템 구현 및 성능 비교

본 연구에서 설계한 시스템에 대해 Java 언어를 이용하여 Eclipse 환경에서 Apache Tomcat과MySQL DB를 이용하여 구현하였다.

4.1 구현 결과

사용자 단말에 앱을 설치한 후 실행하면 엔진 서버와 연동하여 (그림 15)와 같은 업데이트 과정을 수행하게 된다.

(그림 15) 시스템 체크 과정

(Figure 15) System Check Procedure

시스템 초기 업데이트 과정을 수행하면 서버로 로그인하게 되며 (그림 16)과 같이 사용자 접속 정보가 서버 DB에 저장된다.

```
mysql> select * from log_db;
+-----+-----+-----+-----+-----+-----+-----+-----+
| log_code | user_code | user_id | user_name | log_date | android_dev | android_ver | engine_ver | app_ver | app_info |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | root | 2012-9-16 20:01:35 | sdb | 2.3.3 | 1.0.1 | 1.0.0 | Login Success |
+-----+-----+-----+-----+-----+-----+-----+-----+
mysql> select * from log_db;
+-----+-----+-----+-----+-----+-----+-----+-----+
| log_code | user_code | user_id | user_name | log_date | android_dev | android_ver | engine_ver | app_ver | app_info |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | root | 2012-9-16 20:01:35 | sdb | 2.3.3 | 1.0.1 | 1.0.0 | Login Success |
| 2 | 1 | 1 | root | 2012-9-17 01:54:28 | sdb | 2.3.3 | 1.0.1 | 1.0.0 | Login Success |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

(그림 16) 서버 로그인 기록

(Figure 16) Server Login Information

(그림 17) 실행 중인 어플리케이션 정보 확인 결과
(Figure 17) Activated Application Checking

(그림 18) 사용자 단말 내 어플리케이션 관련 정보 송신 결과
(Figure 18) Transmission of User's Application-related Information

4.1.1 실행 중인 어플리케이션 이벤트 정보 확인

(그림 17)과 같이 실행 중인 어플리케이션 정보를 확인할 수 있다. 각 사용자 단말에서 백그라운드 모드로 실행되고 있는 어플리케이션의 아이콘, 이름, 패키지명 등이 나타나고 실행중인 총 어플리케이션의 수, 구동중인 프로세스와 서비스의 수를 확인할 수 있다. 패키지 이름과 어플리케이션 UID 및 PID, 버전 정보와 해당 경로 정보 등을 확인할 수 있으며 이는 서버로도 전송되는 것을 확인할 수 있었다.

또한 각 사용자 단말에서 실행한 어플리케이션에 포함된 권한 정보 등이 서버로 전송되는 것을 확인할 수 있었다. 효율적인 전송을 위해 (그림 18)과 같이 사용된 패키지 정보와 앱 이름, 설치 경로 정보 및 프로세스 관련 정보 등

(그림 19) Logcat을 통한 네트워크 정보 확인
(Figure 19) Network Info Check using Logcat

(그림 20) cmd를 통한 네트워크 정보 확인
(Figure 20) Network Info Check using cmd

을 패킷 형태로 재구성하여 서버로 전송하였다. 이러한 기능을 통해 모바일 단말에서의 네트워크 트래픽을 줄일 수 있도록 하였으며 서버에 일정한 주기로 각 단말에서 실행 중인 어플리케이션의 목록을 유지 및 확인하도록 하여 만일 사용자가 악성 어플리케이션을 잘못 설치하여 이용하였을 경우 서버와의 체크 과정을 통해 이를 바로 확인할 수 있는 기능을 제공하였다.

4.1.2 네트워크 이벤트 정보 및 악성 앱 실행 여부 확인

(그림 19) 및 (그림 20)과 같이 실행중인 어플리케이션과 연계된 네트워크 트래픽 정보가 수집되는 것을 확인하였다. 이는 Eclipse에서도 특정 어플리케이션을 실행할 경우 확인할 수 있으며 해당 정보가 서버에 저장되도록 하였다. 또한 단말에 adb shell을 이용하여 네트워크 접속 정보를 확인할 수 있었으며 해당 정보를 일정한 주기로 서버에 저장하도록 하였다.

서버에서는 수집된 네트워크 정보를 대상으로 만일 특정 사용자 단말에서 구동되는 어플리케이션에 의해 상대적으로 많은 양의 트래픽이 발생할 경우 악성 앱 실행 여부를 판단하는 과정에 적용되도록 하였다.

(그림 21) 악성 앱 실행 여부 탐지
(Figure 21) Malicious App Execution Detection

각 사용자 단말 내에서 구동되는 악성 어플리케이션에 의해서 사용자도 모르게 진행되는 루팅 여부를 확인하는 기능을 제공하였다. (그림 21)과 같이 사용자 단말에서 악성 앱이 실행되면서 루팅 상태로 전환되면 이를 사용자에게 알려주는 기능이 활성화 되는 것을 확인하였다.

4.2 성능 비교 및 평가

기존에 개발되었던 시스템들과 본 연구에서 설계 및 구현한 시스템의 성능 비교 결과를 제시하면 다음 (표 1)과 같다. 무결성 검증 기법인 경우 MD5, SHA-1 등의 해시 함수를 이용하여 정상 앱과 비정상 앱을 구별하는 기능을 제공하지만, 어플리케이션에 대한 실시간 검사 기능이 부족한 것으로 나타났다. 또한 정적/동적 분석 기법인 경우 기존 스마트워크 단말에 대한 가상머신 변형 과정이 필요한 방법이어서 실제 상용 단말에 적용하기 위해서는 커널에 대한 변형이 필요하였다. 또한 블랙리스트 기법인 경우에도 블랙리스트 DB 구축 과정의 어려움으로 인해 오탐율이 높아질 수 있었다. 이에 반해 본 연구에서 제시한 시스템인 경우 단말 내 프로세스 및 서비스에 대한 모니터링 과정을 수행하여 악성 앱에 의해 수행되는 루팅 공격에 대한 탐지 기능을 제공하며 다수의 사용자 단말을 대상으로 한 악성 앱 탐지 과정에 적용 가능하다는 특징을 제공한다.

(표 1) 기존기법들과 제안 시스템의 성능비교
(Table 1) Performance Comparison

성능비교	기법	무결성검증 기법[4]	정적/동적분석 기법[2]	블랙리스트 기법[4]	기분분석기법 기법[4]	제안한 DroidVulMon
시그니처 분석		MD5, SHA-1 등 해시함수 이용	MD5, SHA-1 등 해시함수 이용	MD5, SHA-1 등 해시함수 이용		MD5, SHA-1 등 해시함수 이용
정적 분석			dedexer 이용		기본 블록에 기반을 둔 행동 분석 (BAD : Behavior Analysis based on Basic Block)	
동적 분석		탈빅 머신에서 안드로이드 API 루팅 및 분석 후 악성 코드 탐지 가능 제공		시스템 파일 분석을 통한 악성코드 감염의심 여부 가능 제공	동적 프로그램 실행이상을 통해 특정 변수에 영향을 미치는 명령어들을 추출	단말내 프로세스 및 서비스에 대한 모니터링 가능 및 루팅 탐지 가능 제공
분석 엔진		무결성 검증 모듈 엔진	디스어셈블러 및 리눅스 커널 내 변형된 탈빅 가상머신	악성 앱을 수집 및 관리하는 블랙리스트 엔진	블록 단위 CFG 생성기	루팅 공격 및 루팅에 대한 업데이트 엔진 서버 연동
분석 환경		단말기 및 에뮬레이터	에뮬레이터	단말기 및 에뮬레이터	BAB 분석 모듈	단말기 및 에뮬레이터
분석 결과		어플리케이션에 대한 무결성 검증	어플리케이션에 대한 정적/동적 분석을 통해 악성 코드 탐지 가능	어플리케이션의 악성 코드 여부를 블랙리스트 DB에 저장하여 악성 앱이 발견될 때 해당 앱 삭제	수반변의 반복을 통해 악성 앱의 행동 실행 시간을 측정 및 변형된 파일의 과의 차이와 실행 부하 확인	다수의 상용 모바일 단말 내 정보를 수집해서 루팅 공격 탐지 및 보안 취약성 분석
한계점		어플리케이션에 대한 실시간 관리 및 안전성에 대한 체계적인 검증이 불가능	정적 및 동적 분석에 대한 오탐율 존재	무결성 검사 중심의 앱 관점에서 보안에 취약	BAB 분석 모듈 상용 모바일 단말 내 실행 로그 분석	시스템 내부에서 발생하는 저수준의 이벤트 정보의 수집 및 분석 여부

하지만, 본 연구에서 설계 및 구현한 시스템 역시 다음과 같은 부분에 대한 개선이 필요하다. 우선 현재 시스템인 경우 다수의 모바일 사용자를 대상으로 기초적인 이벤트 정보를 수집 및 분석할 수 있다는 장점을 제공하지만, 해당 기능을 사용하기 위해서는 모바일 단말에 본 연구에서 개발한 클라이언트 앱을 설치하여야 하며 서버와의 연동을 필요로 한다는 단점이 존재한다. 일반적으로 기존의 시스템인 경우 대부분 별도의 서버 연동 없이 단말 내 자체 모듈에 의해서 악성 앱에 대한 판단 과정을 수행하고 있다. 하지만 이 경우 단일 단말을 대상으로 모니터링 과정을 수행하기 때문에 최근 급증하고 있는 Zero-day 공격 등에는 효율적으로 대응할 수 없을 것으로 판단된다. 이밖에도 악성 앱을 대상으로 실험이 많이 상대적으로 적게 수행되어 오탐율을 낮출 수 있는 방안이 더욱 보완되어야 한다. 하지만 본 연구에서 제시한 기법을 사용할 경우 다수의 사용자로부터 발생한 단말 내 이벤트 정보를 수집할 수 있다는 장점이 있으며 이를 토대로 신규 공격이나 이상 패턴에 대한 1차적인 수집 및 분석 과정을 수행할 수 있다는 점에서 의미가 있다고 판단된다.

5. 결 론

본 논문에서는 상용 단말에 대한 공격에 대응하기 위해서 다수 사용자 모바일 단말에 대한 악성 앱 탐지 시스템을 설계 및 구현하였다. 우선 단말 내 이벤트 및 시스템 영역에서 실행되는 프로세스 정보 등을 획득하고 이를 서버

로 전송할 수 있도록 클라이언트 및 서버 시스템을 개발하여 다수의 사용자로부터 발생한 이벤트 정보를 토대로 악성 앱에 대한 실행 여부를 판단할 수 있는 기초적인 기능을 제공하였으며 이를 토대로 상용 단말에서 생성되는 각종 이벤트 정보를 수집 및 분석할 수 있는 시스템을 제공할 수 있었다. 향후 연구과제로는 시스템 내부에서 발생하는 저수준의 이벤트 정보까지도 획득/분석할 수 있는 체계 구축하고 다수의 사용자 단말에서 수집된 이벤트 정보에 대한 그룹핑 방법과 실시간으로 수신되는 정보에 대한 효율적인 유사도 판별 메커니즘 및 악성 여부 판단의 성능과 오답율을 줄일 수 있는 방법에 대해 연구할 필요하다.

참 고 문 헌(Reference)

- [1] Jaeyoung Lee, Doeun Kim, Jiyoung Lee, "An Integrity Verification Method for Secure Application on the Smartphone", *Journal of Advanced Information Technology and Convergence*, Vol. 9, No. 10, pp.223-228, 2011. 10.
- [2] Wontae Sim, Jongmyoung Kim, Jaechul Ryu, Bongnam Noh, "Android Application Analysis Method for Malicious Activity Detection", *Journal of KIISC(Korea Institution of Information Security and Cryptology)* Vol. 21, No.1, pp.213-219, 2011. 2.
- [3] Woobong Chun, Junghee Lee, Wonhyung Park, Taemyoung Jung, "The Mobile Security Diagnostic System against Smartphone Threat", *Journal of KIISC*, Vol.22, No.3, pp.537-544, 2012. 6.
- [4] Woohyung Ahn, Jungmin Lee, Seungho Lim, Jaewon Oh, "Detection Malware on Smartphone by Tracking of the Execution of Basic Blocks", *Journal of KIISE(Korean Institute of Information Scientists and Engineers): System and Theory*, Vol.39, No.3, pp.179-187, 2012. 6
- [5] 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
- [6] Kyoungmin Kim, "A Study on Client Information Protection in Domestic Banking As Introducing Smartphone Banking", *Journal of KIISC*, Vol.20, No.6, pp.51-62, 2010. 12.
- [7] Youngdong Kim, Ikwhan Kim, Taehyun Kim, "Analysis of Usage Patterns and Security Vulnerabilities in Android Permissions and Broadcast Intent Mechanism", *Journal of KIISC*, Vol.22, No.5, pp.1145-1157, 2012. 10.
- [8] Moonyoung Whang, Woong Goh, Dongbum Lee, Jin Gwak, "Certificate Management Plan in Smartphone-banking using the Mobile-Cloud Computing", *Proceedings of IEEK(Institute of Electronics Engineers of Korea) Conference*, pp. 1873-1876, 2010. 6.
- [9] "Andromaly: a behavioral Malware detection framework for android devices", *Proceedings of IEEK Conference*, pp.1-30, 2010.6

● 저 자 소 개 ●

함 유 정

2012년 한신대학교 정보통신학과 졸업(학사)

2012년~현재 한신대학교 대학원 컴퓨터공학과 재학중(석사)

관심분야 : 스마트폰 보안, Smart Fuzzing.

E-mail : you86400@hanmail.net

이 형 우

1994년 고려대학교 컴퓨터학과 졸업(학사)

1996년 고려대학교 대학원 컴퓨터학과 졸업(석사)

1999년 고려대학교 대학원 컴퓨터학과 졸업(박사)

1999년~2003년 백석대학교 정보통신학부 조교수

2003년~현재 한신대학교 컴퓨터공학부 부교수, 정교수

관심분야 : 정보보호, 스마트폰 보안, 컴퓨터 포렌식스, Fuzzing.

E-mail : hwlee@hs.ac.kr