# 안드로이드 운영체제의 구조와 보안성

학번: 2016113566, 이름: 김남영

## I. 개요

안드로이드 운영체제는 각종 모바일 기기의 동작을 위해 구글에서 개발한 운영체제이며, 리눅스 커널을 기반으로 동작한다. 모든 소스가 오픈 소스로 공개되어 누구나 안드로이드의 내부 구조를 볼 수 있으며, 앱의 수정과 재배포가 가능하다. 이러한 개방성을 강조한 OS 정책은 모바일 OS 시장의 후발 주자인 안드로이드를 급속도로 성장시켰다. 현재 안드로이드는 전 세계 모바일 OS 시장에서 약 75%의 점유율을 가진다 [1]. 하지만 사용자의 자유도 보장이라는 정책의 근본적인 목적과 달리 동시에 여러 보안 관련 이슈를 만들어 내기도 했다. 본 보고서에서는 안드로이드 운영체제의 계층적 구조와 각 계층의 역할을 설명하고, 여러 보안적 이슈를 타 OS 와 비교하여 소개한다.

## II. 안드로이드 운영체제의 구조

안드로이드 운영체제는 리눅스 커널을 기반으로 만들어졌으며, Table 2 에서 볼 수 있듯 Applications, Application Framework, Libraries, Android Runtime, Linux Kernel 의 총 5 개의 계층으로 나누어져 있다. 해당 섹션에서는 안드로이드 운영체제의 계층 구조와 각 계층의 역할을 소개한다 [2].

### *A. Linux Kernel*

안드로이드 운영체제에서 사용되는 리눅스 커널은 일반적으로 PC 에서 사용되는 리눅스 커널에 각종 드라이버, 전력 관리 등 모바일 플랫폼에 필수적이고 중요한 몇 가지의 기능을 추가로 포함하고 불필요한 기능을 삭제한 형태로 제공된다. 즉, 안드로이드 운영체제는 리눅스의 배포판 중 하나라고 볼 수 있다. 리눅스 커널의 다양한 역할을 다음과 같이 소개한다 [2].

- 하드웨어 추상화

하드웨어 추상화란 소프트웨어가 CPU 나 Memory 와 같은 하드웨어 자원을 사용할 때, 다르게 제작된 하드웨어라 할지라도 같은 종류의 하드웨어라면 일관적인 서비스를 제공하는 기능을 말한다 [7].

Table 1. Smartphone operating system market share worldwide, 2019 [1]

| S/N | Mobile OS | Market Share(%) |
|---|---|---|
| 1 | Android OS | 75.33 |
| 2 | Ios | 22.4 |
| 3 | KaiOS | 0.84 |
| 4 | Windows OS | 0.61 |
| 5 | Unknown | 0.36 |
| 6 | Windows | 0.28 |

- 선점형 멀티태스킹

  강제적으로 CPU 의 제어권을 가져올 수 없는 비선점형 멀티태스킹과 달리, 스케쥴러의 필요에 따라 CPU 의 제어권을 강제로 가져와서 더 높은 우선순위를 가진 Task 부터 처리가 가능하도록 만드는 기능이 선점형 멀티태스킹이다.

- 자원의 효율적 관리

  Memory, Process and Power Management 등을 통해 한정된 자원을 효율적으로 관리하고, 하드웨어에 공급되는 전력을 관리한다.

- 하드웨어 드라이버

  카메라, 블루투스, WIFI 등 각종 하드웨어를 제어하기 위한 드라이버를 포함한다.

*B. Android Runtime [8]*

런타임 환경은 프로그래밍 언어가 구동되는 환경을 의미한다. 안드로이드의 경우 프로그래밍 언어로 JAVA 를 사용하고, JAVA 는 가상 머신을 통해 컴파일 되므로 어떤 가상 머신을 사용하는지는 운영체제 관점에서 중요한 이슈가 될 수 있다. 크게 Dalvik 가상 머신(DVM)과 Android Runtime(ART)로 나누어 진다.

보통 JAVA 언어를 사용하는 경우 JAVA 에서 제공하는 자바 가상 머신(JVM)을 이용하지만, 라이센스와 메모리 효율성의 문제로 구글에서는 자체적으로 Dalvik 이라는 이름의 가상 머신을 만들었다. 실행 속도가 느린 JAVA 의 특성을 보완하기 위해 DVM 은 자주 사용되는 바이트 코드를 미리 컴파일 해놓는 JIT(Just-In-Time)방식을 도입했고, 이러한 컴파일 방식때문에 기존의 컴파일 과정보다 속도를 크게 향상 시킬 수 있었다. 하지만 JIT 컴파일러는 하드웨어에 상당한 부하를 걸어 배터리 소모량이 크다는 치명적인 단점이 있어, 안드로이드 5.0 버전 이후에 DVM 은 완전히 폐지되었다.

DVM 의 한계를 극복하기 위해 개발된 것이 Android Runtime(ART)이다. ART 는 앱을 설치할 때 기계어로 해석을 완료하는 컴파일 방식인 AOT(Ahead-Of-Time) 방식을 사용한다. 따라서 설치하는 시간은 느릴 수 있으나, 일단 설치가 끝나면 실행 속도가 빠르다는 장점이 있다. 엄밀히 말하면 ART 는 가상머신이 아니라, 컴파일이 완료된 앱의 실행 시 사용되는 라이브러리이다. 설치가 끝나는 시점에 이미 컴파일이 완료되기 때문에 가상 머신이 필요가 없으며, 기계어와 동급의 실행 속도를 가질 수 있다. 이러한 이유로 안드로이드 5.0 버전 이후에 기본 런타임으로 적용되었다.

안드로이드 7.0 버전 이후로는 최초 설치시에는 JIT 컴파일 방식을 사용하고, 차후에 자주 사용되는 앱에 대해서는 AOT 컴파일 방식으로 전환되는 방식을 사용하여 기존 AOT 방식만 사용할 때 보다 더욱 더 속도를 향상시켰다.

Table 2. Android OS 의 계층적 구조 [2]

<table>
<tr><td>Application</td><td colspan="3">Home, Contacts, Phone, Browser,</td></tr>
<tr><td>Application Framework</td><td colspan="3">Manager(Activity, Window, Notification, Package, Telephony, Resource, Location),<br>Content Providers, View System, XMPP Service</td></tr>
<tr><td>Libraries</td><td>Surface Manager, Media Framework, SQLite, OpenGL/ES, FreeType, WebKit, SGL, SSL libc</td><td>Android Runtime</td><td>Core Libraries,<br>Android Runtime(ART)</td></tr>
<tr><td>Linux Kernel</td><td colspan="3">Drivers (Display, Camera, Bluetooth, Flash Memory, Binder, USB,<br>Keypad, WIFI, Audio), Power Management</td></tr>
</table>

### *C. Libraries*

안드로이드 운영체제는 어플리케이션 개발의 용이함을 위해 다양한 라이브러리를 제공한다. 몇 가지 예시를 다음과 같이 소개한다 [2].

- Surface Manager – 디스플레이의 다양한 그래픽 효과를 제공하는 라이브러리
- Media Framework – 오디오, 비디오 또는 정적 이미지 파일에 대한 라이브러리
- SQLite – 데이터베이스 프로그래밍을 위한 라이브러리
- OpenGL/ES – 그래픽 처리 하드웨어에 대한 표준 인터페이스를 제공하는 라이브러리
- FreeType – 비트맵과 벡터 폰트 렌더링을 담당하는 라이브러리
- WebKit – 웹 브라우저를 만드는 데 기반이 되는 라이브러리
- SGL(Scalable Graphic Library) – 텍스트, 이미지 등 2D 그래픽을 담당하는 라이브러리
- SSL(Secure Sockets Layer) – 클라이언트와 서버의 안전한 통신을 담당하는 라이브러리
- Libc – 표준 C 시스템 라이브러리

### *D. Application Framework*

어플리케이션의 실행과 관리를 위한 서비스의 집합을 어플리케이션 프레임워크라고 칭한다. 안드로이드 어플리케이션 프레임워크에서는 다음과 같은 핵심 서비스를 제공한다 [2].

- Activity Manager – 어플리케이션의 실행과 종료를 담당
- Window Manger – 어플리케이션과 관련된 윈도우 관리
- Notification Manager – 메시지, 어플리케이션의 알림 등을 관리
- Package Manger – 실행 중인 어플과 관련된 정보를 모두 수집하여 관리
- Telephony Manager – 전화 서비스에 대한 정보를 어플리케이션에 제공
- Resource Manager – 코드에 포함되지 않는 모든 부분에 대한 접근 권한 제공
- Location Manager – 어플리케이션이 위치 정보를 수집할 수 있도록 권한 제공
- Content Provider – 다른 어플리케이션에 대한 접근과 데이터 공유를 가능하게 함
- View System – 어플리케이션 UI 에 사용되는 뷰의 집합
- XMPP Service – 유저 간 양방향 채팅을 할 때 사용되는 통신 프로토콜

### *E. Application*

이메일, SMS, 캘린더 등의 플랫폼에 기본적으로 포함되어 있는 어플리케이션과 사용자가 별도로 설치한 어플리케이션을 통칭한다. 즉, 사용자와 만나는 최상위 계층이라고 볼 수 있다.

## III. 안드로이드 운영체제의 보안 취약성

수 많은 개인 정보를 담고 있는 모바일 기기의 사용이 현대 사회의 필수적인 요소가 되면서, 모바일 기기의 보안 관련 이슈는 늘 사용자의 주 관심이 되어 왔다. 일반적으로 모바일 기기의 사용자들은 그들이 사용하는

어플리케이션이나 컨텐츠에 악의적인 공격 요소가 있는지 없는지 관심이 없다. 따라서 운영체제 차원에서 악의적으로 보안을 위협하는 각종 행위를 차단할 수 있어야 한다. 본 섹션에서는 안드로이드 운영체제와 iOS 운영체제의 보안 기법에 대해 소개하고, 안드로이드 운영체제가 iOS 에 비해 보안에 취약한 이유를 소개한다.

### *A.* 안드로이드의 보안 기법

안드로이드 운영체제는 리눅스 커널 기반으로 개방성을 강조한 오픈 소스 정책을 펼치기 때문에 어플리케이션의 검수 절차가 없고 수정과 재 배포 등이 자유로운 편이다. 이러한 오픈 소스 정책은 모바일 OS 시장의 후발 주자인 안드로이드를 압도적인 선두 주자로 만들었지만, 동시에 누구나 운영체제의 취약함을 들여다볼 수 있으므로 누구나 악의적인 마음을 가졌다면 보안의 결점을 파고들 수 있다. 해당 섹션에서는 다양한 보안 기법 중 커널 차원에서의 보안 기법들을 소개한다.

- 어플리케이션 샌드박스

  안드로이드 운영체제는 리눅스 커널에 기반하므로 리눅스의 사용자 기반 보호 기능을 그대로 활용할 수 있다. 어플리케이션의 설치 시에 고유한 사용자 ID(UID)를 지정하고, 자체 프로세스에서 이를 실행한다 [5]. 이 UID 를 통해 커널 수준의 샌드박스가 설정되고 해당 UID 을 가진 어플리케이션이 아니라면 읽기/쓰기 등 어떠한 권한도 갖지 않으므로, 악의적인 작업이 허용되지 않는다.

- 파일 시스템 권한

  리눅스는 파일에 대한 접근 권한을 부여할 수 있으므로, 명시적으로 파일을 공유하지 않으면 다른 어플리케이션에서 해당 파일을 변경할 수 없다 [5].

- 자체 검사 부팅

  안드로이드 7.0 버전 이후에서는 부팅하는 동안 소프트웨어의 무결성을 암호화 방식으로 확인하므로, 보안이 침해된 기기는 부팅되지 않는다 [5].

- 보안이 강화된 리눅스(SELinux)

  루트 및 수퍼 유저 권한으로 실행되는 작업들은 해당 작업의 데이터에 대한 모든 접근 권한을 가지므로, 보안에 매우 치명적이다. 하지만 안드로이드 개발자에게는 어플리케이션이나 시스템을 디버깅하거나 수정할 수 있는 권한이 필수적이므로, 권한을 어느 지점까지 허용 시킬 것인지는 매우 중요한 문제가 된다. 보안이 강화된 리눅스 모델인 SELinux 는 루트 및 수퍼 유저 권한으로 실행되는 모든 작업에 강제적으로 접근을 통제(MAC) 할 수 있으므로 악성 어플리케이션의 영향과 잠재적 코드 결함을 방지할 수 있다 [5].

### *B.* iOS 의 보안 기법

애플에서 개발한 모바일 OS 인 iOS 는 유닉스를 기반으로 하여 안드로이드 운영체제와 달리 폐쇄성을 강조한 정책을 펼친다. 사용자의 자유가 제한되기는 하지만 개방적인 정책의 여러 리스크를 근본적으로 차단할 수 있어, 안드로이드 운영체제 보다 보안적 이슈가 훨씬 적게 발생한다. 해당 섹션에서는 iOS 의 다양한 보안 기법을 소개한다.

- 어플리케이션 검수과정

  iOS 에 설치되는 모든 어플리케이션은 앱스토어를 통해서만 설치가 가능하고, 애플 사의 전담 팀에 의해 충분한 검수 과정 이후에 앱스토어에 등록되므로, 악의적 어플리케이션이 운영체제에 침범할 여지가 적다.

- 접근 권한

  타 사의 어플리케이션이 애플 사의 기본 어플리케이션에 접근할 권한을 요구할 수는 있지만, 민감한 개인 정보에 대해서는 권한을 부여하지 않으므로 원천적으로 접근이 불가능하다. 이러한 엄격한 폐쇄성은 사용자에게 불편함을 주기도 하지만 악의적 어플리케이션의 접근을 근본적으로 봉쇄하므로 안드로이드 운영체제에 비하여 보안적 이슈가 상당히 적다.

### *C.* 안드로이드 운영체제와 타 OS 의 보안 취약성 비교

안드로이드 운영체제를 제외한 타 OS(iOS, Windows Phone, BlackBerry, Symbian 등)는 모두 폐쇄적인 운영체제 정책을 가진다. 안드로이드가 개방성을 유지하면서도 보안성을 높이기 위해 커널 차원에서 여러 기법들을 제공하고 있지만, 그럼에도 불구하고 반드시 소스 코드의 결점은 존재하고 누구나 그 결점을 들여다볼 수 있으므로 보안의 근본적 문제를 해결하기는 힘들다. 그리고 데이터 공유, 메모리 관리 등 상당한 부분의 접근 권한을 자유롭게 제공한다는 점은 여러 보안 이슈에 대한 방임이라고 표현되기도 한다. 또한, 전 세계 모바일 OS 시장에서 약 75%의 점유율[1]을 가지므로, 악의적인 목적을 가진 해커들의 주요한 타겟이 될 수 밖에 없다.

아래의 Fig. 1 은 F-secure 과 Kaspersky 사에서 조사한 운영체제별 Malware attack 의 비율을 비교한 차트이다. 폐쇄적인 정책을 가지는 타 운영체제와는 달리 압도적으로 주요한 타겟이 되고 있음을 확인할 수 있다.[3]

## IV. 토의 및 결론

앞서 말했듯, 스마트폰을 비롯한 다양한 모바일 기기는 현대 사회에서 필수적인 요소가 되었다. Kaspersky 사의 통계에 의하면 2020 년도 1 분기 모바일 기기에 대한 1,152,662 건의 악의적인 공격을 감지했다 [6]. 수 많은 개인 정보를 포함하고 있고, 압도적인 사용자 수를 보유하고 있는 만큼 보안 취약성의 극복은 현재 안드로이드의 최우선 과제이다. 안드로이드가 커널 차원에서 다양한 보안 기법을 제공하지만, 취약성의 상당 부분이 사용자와 개발자들에게 많은 권한을 부여하고 소스가 공개되어 있다는 점에 기인한다. 사용자는 새로운 버전으로 운영체제를 업데이트 하는 것에 관심이 없고, 검수가 충분히 이루어지지 않은 어플리케이션은 다양한 방식으로 사용자의 운영체제에 침범할 수 있다. 커널 차원의 보안 기법에는 분명히 한계가 존재하며, 어쩌면 이러한 개방적인 정책은 방임으로 보여 질 수 있다. 따라서, 타 운영체제의 폐쇄성을 일부 수용하여 모바일 OS 시장의 대표 주자로서 책임감을 보여야만 할 것이다.

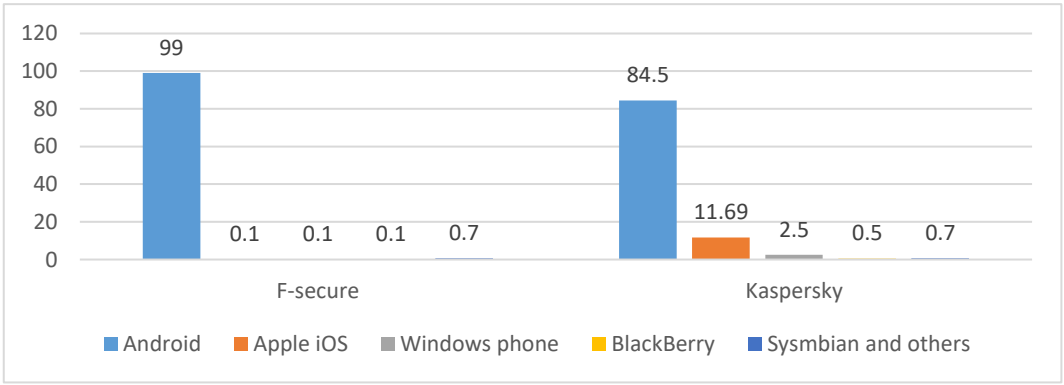


Fig. 1. Malware attacks on smartphone OSes [3].

## REFERENCES


[1] Sikder, Ratul, et al. "A survey on android security: development and deployment hindrance and best practices." *Telkomnika* 18.1 (2020): 485-499.

[2] Nell Smyth, Android Studio 3.3 Development Essentials, Android 9 Edition, February 5, 2019, pp.65-68.

[3] Ahvanooey, Milad Taleby, et al. "A survey on smartphones security: Software vulnerabilities, malware, and attacks." *arXiv preprint arXiv:2001.09406* (2020).

[4] 박상호, and 권태경. "안드로이드 앱을 위한 플랫폼 보안 분석." *한국인터넷정보학회 학술발표대회 논문집* (2013): 79-80.

[5] Android, https://source.android.com/security/overview/kernel-security

[6] Kaspersky, https://securelist.com/it-threat-evolution-q1-2020-statistics/96959/

[7] Microchip Technology Inc, Hardware Abstraction Layer, http://ww1.microchip.com/downloads/en/DeviceDoc/hardware-abstraction-layer.pdf, pp. 2.

[8] Ndwaru, Laban. “Introduction to Android ART; The next generation of Android Runtime” (2014)