Reverse engineering and hacking Android apps

Eric Lafortune
Background

Creator of ProGuard

CTO at GuardSquare (Belgium / USA)

Reporter of the Janus vulnerability
Topics

- Structure of Android apps
- Static analysis
  - Decompilers
  - Disassemblers
- Dynamic analysis
  - Hooking frameworks
- Modifying apps
Structure of Android apps

Source code

- Manifest (AndroidManifest.xml)
- Asset files (assets/*)
- Resource files (res/*/*)
- Java source (*.java)
- C/C++ source (*.c)
Structure of Android apps

**Source code**
- Manifest
  - (AndroidManifest.xml)
- Asset files
  - (assets/*)
- Resource files
  - (res/*/*)
- Java source
  - (*.java)
- C/C++ source
  - (*.c)

**Apk file**
- Manifest
  - (AndroidManifest.xml)
- Asset files
  - (assets/*)
- Resource constants
  - (resources.arsc)
- Resource files
  - (res/*/*)
- Dalvik bytecode
  - (classes.dex)
- Native libraries
  - (lib/*/*.so)
Static analysis

- jar, unzip
- aapt (Android SDK)
- apkanalyzer (Android SDK)
Decompilers

- jadx
- dare
- dextra
- Androguard DAD
- dex2jar + jad, JDGui, fernflower, procyon, ...

Java source (*.java)

Dalvik bytecode (classes.dex)
Wrong code, try again!
public class MyComputationClass {
    private MySettings settings;
    private MyAlgorithm algorithm;
    private int answer;

    public int computeAnswer(int input) {
        ...
        return answer;
    }
}
public class MyComputationClass {
    private MySettings settings;
    private MyAlgorithm algorithm;
    private int answer;

    public int computeAnswer(int input) {
        ...
        return answer;
    }
}

public class a {
    private b a;
    private c b;
    private int c;

    public int a(int a) {
        ...
        return c;
    }
}
Disassemblers

- dexdump (Android SDK)
- apkanalyzer (Android SDK)
- baksmali
- apktool

Readable bytecode

Dalvik bytecode (classes.dex)
Protect with string encryption?

String code = “1478”;
String code = “1478”;

byte[] data = 
    new byte[] { 0x4d, 0x54, 0x51, 0x33, 0x4f, 0x41, 0x3d, 0x3d };

String code = 
    new String(Base64.getDecoder().decode(data));
Dynamic analysis

Android software layers

Application
Android framework
Libraries
Virtual machine
Linux kernel
Dynamic analysis

Hooking frameworks

Application

VirtualApp

Android framework

Substrate

Xposed

Libraries

Frida

Virtual machine

YAHFA

Linux kernel

Magisk + modules
Frida

frida client

Python script

frida-server

Javascript script

App
Modifying apps

- apktool
- apksigner (Android SDK)
Demo

You cracked the safe!
More protection?

- Root detection
- Signing certificate checking
- Code encryption
- White box cryptography
- ...

Mobile Application Protection
Conclusions

Basic reverse engineering is easy!

Avoid putting secrets inside your application
Conclusions

Basic reverse engineering is easy!

Avoid putting secrets inside your application

Application protection raises the bar

Protection and attacks are in an arms race
Thank you