

# REVERSING SLIDELOCK 1.1

## Disclaimers

All code included with this tutorial is free to use and modify; we only ask that you mention where you found it. This tutorial is also free to distribute in its current unaltered form, with all the included supplements.

All the commercial programs used within this document have been used only for the purpose of demonstrating the theories and methods described. No distribution of patched applications has been done under any media or host. The applications used were most of the times already been patched, and cracked versions were available since a lot of time. ARTeam or the authors of the paper cannot be considered responsible for damages to the companies holding rights on those programs. The scope of this tutorial as well as any other ARTeam tutorial is of sharing knowledge and teaching how to patch applications, how to bypass protections and generally speaking how to improve the RCE art. We are not releasing any cracked application.

## Verification

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: <http://releases.accessroot.com>

## Table of Contents

1. Introduction to SlideLock 1.1 .....	2
1.1. What is it, and why do I care? .....	2
1.2. Target: .....	2
1.3. Tools used.....	2
1.4. References.....	2
2. Investigating Slide Lock .....	3
2.1. Finding the SlideLOCK Classes.....	3
2.2. Analyzing the Classes.....	3
2.2.1. RightsInfo .....	3
2.2.2. RightsManager.....	4
2.2.3. RightsActivity .....	5
2.2.4. RightsActivity\$3.....	6
2.2.5. Finding the Developers Class .....	7
3. Patching/Removing the DRM .....	8
3.1. Patching the Switch Statement .....	8
3.2. Changing the Startup Class .....	8
4. What about the Application KEY???	9
5. Conclusions .....	10
6. Greetings.....	10

## 1. Introduction to SlideLock 1.1

### 1.1. What is it, and why do I care?

SlideLOCK is a DRM system for AndroidOS programs that aims to prevent the sharing of purchased APKs amongst users. The protection lies in special classes that the programmer must implement into his/her own code that does server-side checking with device-specific information to ensure the user is authorized to access the application. Each application is assigned a 'key' that is unique to the application and no longer than 32 characters. It is important for this key to stay unknown to users, we'll find out why later.

While DRM protection is OK in some instances it doesn't seem good in this one. Let's say you bought the game AntiBody2 for \$2, and successfully installed it on your phone, and then your phone breaks/dies/gets upgraded, shouldn't you still be entitled to play the game without purchasing it again? I think so.

So with the motive above I dug deeper into SlideLock 1.1 and will detail my findings here, and the end of the tutorial you should be able to properly remove SlideLock 1.1 protection from any application.

It is important to note that since SlideLOCK 1.1 is NOT an out-of-the-box solution, no two scenarios will be exactly the same, so you must learn from this tutorial WHY I did what I did, and not just try to mimic the steps! Here We Go!

### 1.2. Target:

- AntiBody2 (Game)
  - Available for 1.99 USD on SlideME Market
  - Also here: <<LINK>>

### 1.3. Tools used

- APK Manager
  - For Decompiling/Recompiling/Signing APKs
- ADB
  - For pulling and installing APKs
- Notepad
  - For reading the smali/xml files
- Brain







### 1.4. References

- [http://pallergabor.uw.hu/androidblog/dalvik\\_opcodes.html](http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html)

## 2. Investigating Slide Lock

### 2.1. Finding the SlideLOCK Classes

Upon decompiling the application with APKManager we can find the SlideLOCK specific classes in the /smali/org/slideme/android/drm folder, and looks like this:

 RightsActivity\$1.smali	11/29/2010 11:11 ...	Text Document	9 KB
 RightsActivity\$2.smali	11/29/2010 11:11 ...	Text Document	2 KB
 RightsActivity\$3.smali	11/29/2010 11:29 ...	Text Document	6 KB
 RightsActivity.smali	11/29/2010 11:16 ...	Text Document	10 KB
 RightsInfo.smali	11/29/2010 11:11 ...	Text Document	2 KB
 RightsManager.smali	11/29/2010 11:11 ...	Text Document	12 KB

While there appears to be 6 classes, there are really only 3

- RightsActivity
- RightsInfo
- RightsManager

### 2.2. Analyzing the Classes

#### 2.2.1. RightsInfo

This is the smallest of the 3 classes in the DRM system, so it seems natural to look at this one first. Upon opening the file in notepad, the first thing we'll see is that it uses an explicit constructor (one taking arguments):

```
# direct methods
.method public constructor <init>(Ljava/lang/String;Ljava/lang/Class;)V
    .locals 2
    .parameter "key"
    .parameter "mainClass"

    .prologue|
    .line 9
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
```

We see that the constructor takes a String named 'key' and a Class named 'main' this information will be important later. For now we just need to know that any RightsInfo object requires the key and the mainClass of the application to be passed in upon creation.