

# RE Android

Základy reverzného inžinierstva

# How to reverse Android malware ?



1 First glance matters

- Are they trying to hide something?
- What's the name of the package?
- What does the certificate say?
- Where did I find it?

2 Disassemble it

```
const-v  
get i0
```

The code says it all!  
Don't be lazy and read it in <sup>AAAAH</sup> depth.

3 Still don't understand?

Run it in an emulator, display logs and capture network traffic.



*THE CODE DOES NOT MAKE SENSE ?*

Maybe it's heavily obfuscated or packed.

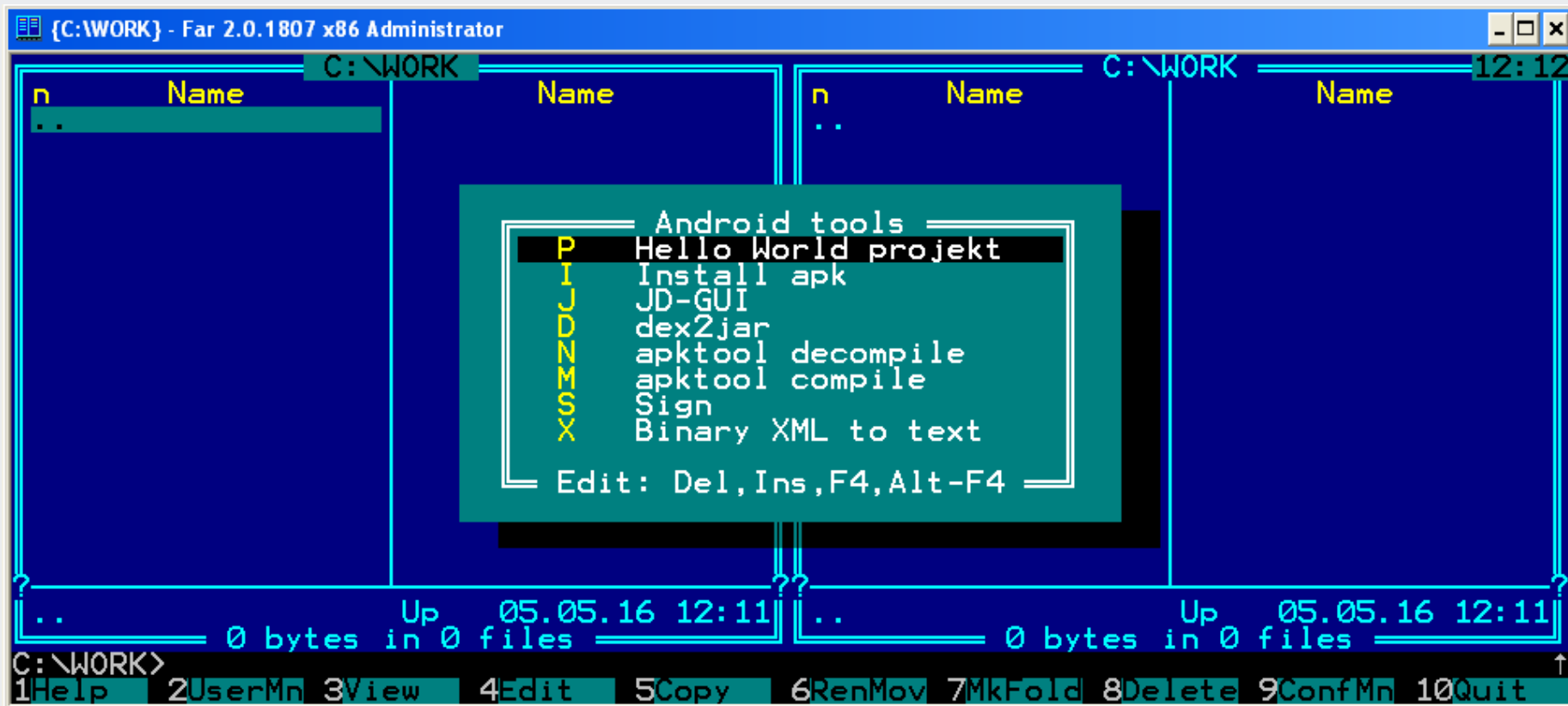
*THERE'S NOTHING SUSPICIOUS ?!*

Good 😊 Check the assets and resources directory for Javascript or ARM executables.



Never use your own phone.  
Do not provide any personal data (name, IMEI, phone number...)

# RE – nástroje



# RE – nástroje

## Android SDK

- adb  
základný nástroj pre ladenie Android zariadenia
- android  
Android SDK Manager
- monitor  
Android Device Monitor
- zipalign  
Optimalizačný nástroj na „zarovnanie“ archívneho súboru .apk
- Aapt  
Android Asset Packaging Tool – výpis relevantných info z .apk
- logcat

# RE – nástroje

## Java JDK

- keytool

Generovanie / spravovanie certifikátov, verejného / privátneho kľúča.

- jarsigner

Podpisovanie a overovanie podpisu.

# RE – nástroje

## Nástroje tretích strán

- Apktool

<https://ibotpeaches.github.io/Apktool/>

- BinXML to Text

Konverzia

- backsmali / smali

Vytiahnutie smali kódu z dex súboru a naopak

- DexToJar

Pomocný prepis do pseudo java zdrojového kódu

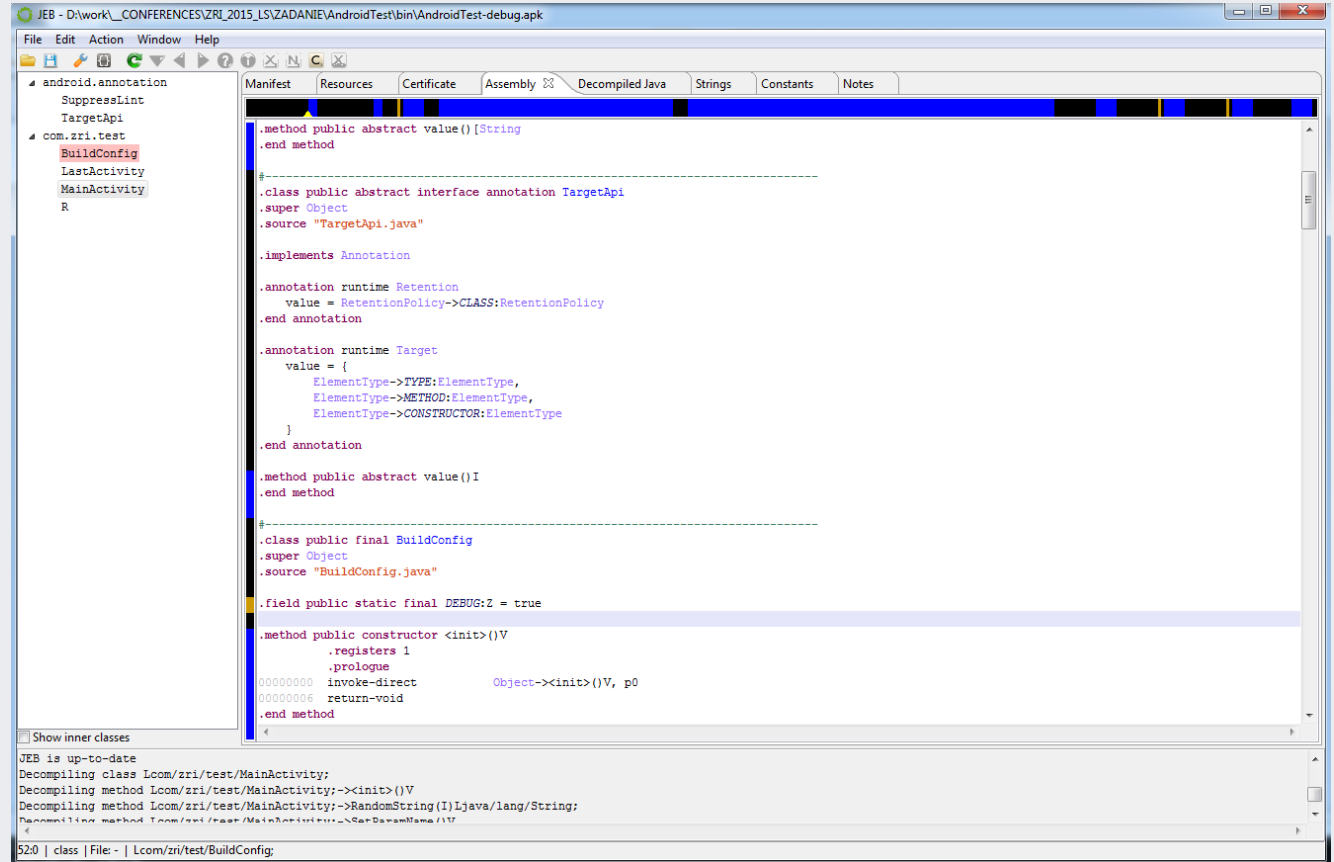
- JD-GUI

Java Decompiler

- Unzip

# RE – advanced tools

- IDA
- JEB
- VTS



The screenshot shows the JEB decompiler interface. The left sidebar displays a project tree with the following structure:

- android.annotation
  - SuppressLint
  - TargetApi
- com.zri.test
  - BuildConfig
  - LastActivity
  - MainActivity
  - R

The main window displays the decompiled Java code for the BuildConfig class. The code is as follows:

```
.method public abstract value() [String]
.end method

-----
.class public abstract interface annotation TargetApi
.super Object
.source "TargetApi.java"

.implements Annotation

.annotation runtime Retention
    value = RetentionPolicy->CLASS:RetentionPolicy
.end annotation

.annotation runtime Target
    value = {
        ElementType->TYPE:ElementType,
        ElementType->METHOD:ElementType,
        ElementType->CONSTRUCTOR:ElementType
    }
.end annotation

.method public abstract value() I
.end method

-----
.class public final BuildConfig
.super Object
.source "BuildConfig.java"

.field public static final DEBUG:Z = true

.method public constructor <init>() V
    .registers 1
    .prologue
    00000000    invoke-direct        Object-><init>() V, p0
    00000006    return-void
.end method
```

The status bar at the bottom indicates the current class being viewed: 520 | class | File: - | Lcom/zri/test/BuildConfig;

# Android zadanie – Vypíš dekryptovaný text!

## Android zadanie

- Vyhľadanie / stiahnutie apk do pc; na základe názvu package **2b**
- Rozbalenie, dekompilácia, zisti entry-point (Activity) **2b**
- Spustenie aplikácie cez adb, logcat s nájdeným tagom **2b**
- Spustenie aplikácie so správnym názvom parametru + hodnota parametru = krsné meno **2b (3b)**
- Splniť podmienku na dekryptovanie textu (prebalenie apk) **2b (3b)**
- Vložiť výpis dekryptovaného textu (prebalenie apk) **3b**

*Zadanie slúži aj ako praktická ukážka reverzovania Android aplikácií*



# Android zadanie – Vypíš dekryptovaný text!

1. adb shell dumpsys package
2. adb pull
3. FAR; Shift+F2 (unzip)
4. FAR; F2+A+X(BinXML2Text)
5. Nájdem “entry-point” Activity
6. FAR; F2+A+D (dexToJar)
7. FAR; F2+A+J (JD-GUI)
8. Statická analýza + vyhledanie Log tagu
9. adb shell am start -n
10. adb logcat -s

# Android zadanie – Vypíš dekryptovaný text!

11. adb shell am start -n ... -e
12. apktool d XXX.apk
13. Upraviť LastActivity.smali
14. apktool b XXX
15. jarsigner (heslo:android)
16. adb uninstall com.zri.test
17. FAR; F2+A+I (install apk)
18. adb shell am start -n
19. adb shell am start -n ...-e

# Android zadanie – Vypíš dekryptovaný text!

**Živá ukážka**

# Modularita

## Hlavná časť

- Obsahuje všetky potrebné permissions
- Má prístup na internet
- Dokáže spúšťať program z externého zdroja (DexClassLoader) / modul

## Prídavné moduly

- Vopred neznámy zdroj
- Obsahuje škodlivý kód (bez permissions)

# Modularita

**Živá ukážka**

# Obfuskácia

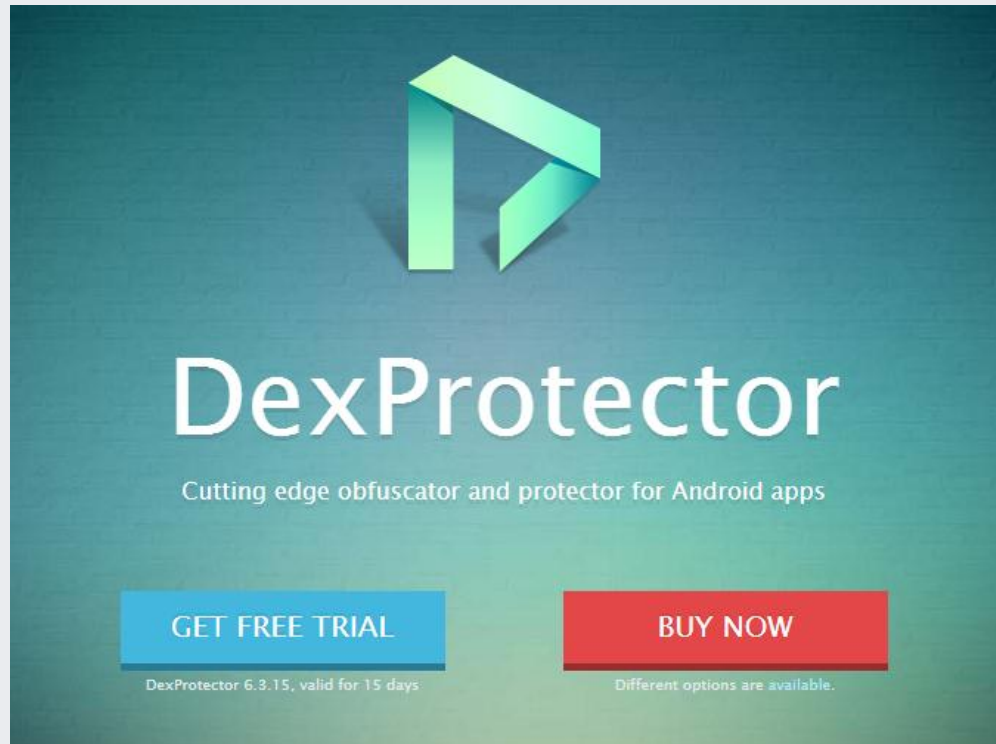
- Zmena názvu tried a metód
  - Vo väčšine prípadov sú názvy tvorené jedným písmenom
- Nadbytočné prázdne triedy a metódy
  - Zreťazenie vzájomného volania „prázdnych“ metód.
- Parametrické volanie tied
  - Metóda invoke prijíma ako parameter názov volanej metódy a jej parametre

# Obfuskácia

## Živá ukážka

# DexProtector

DexProtector 672,60 €



The advertisement features a dark teal background with a 3D play button icon in the upper center. Below the icon, the product name 'DexProtector' is written in a large, white, sans-serif font. Underneath the name, the tagline 'Cutting edge obfuscator and protector for Android apps' is displayed in a smaller white font. At the bottom of the banner, there are two prominent buttons: a blue one on the left labeled 'GET FREE TRIAL' and a red one on the right labeled 'BUY NOW'. Below the blue button, the text 'DexProtector 6.3.15, valid for 15 days' is visible. Below the red button, the text 'Different options are available.' is visible.

**DexProtector**  
Cutting edge obfuscator and protector for Android apps

**GET FREE TRIAL**  
DexProtector 6.3.15, valid for 15 days

**BUY NOW**  
Different options are available.



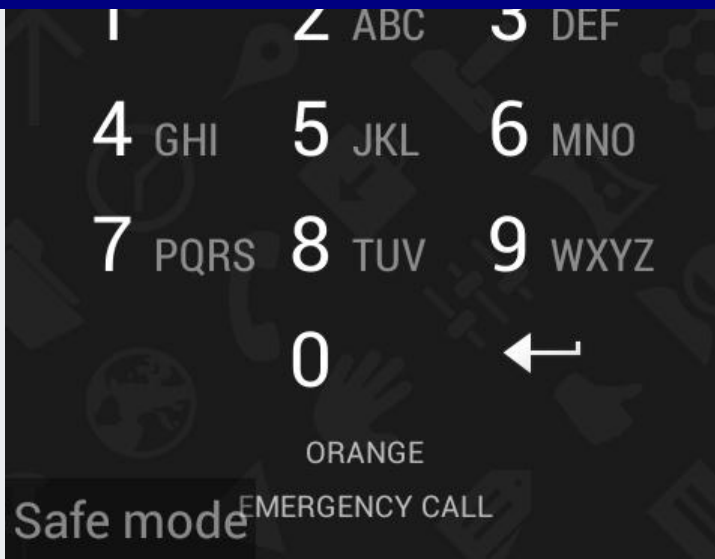
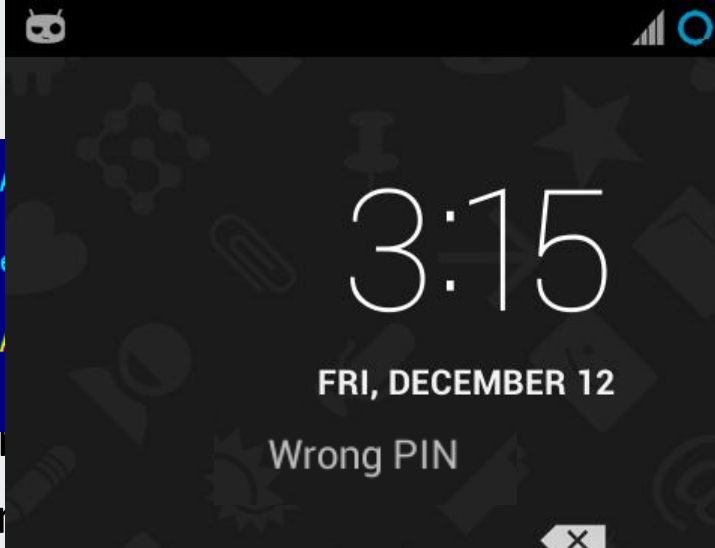
# Hard reset

1. A
2. N
3. Nastavit PIN kod
4. Vypnutý debuging

```
public class DevicePolicyManager {  
    public CharSequence getResetPasswordText() {  
        return "WHA";  
    }  
}
```

Intent intent)

```
((DevicePolicyManager)this.getSystemService("device_policy")).resetPassword("1234", 0);
```



EOF