Automated Analysis and Deobfuscation of Android Apps & Malware

Jurriaan Bremer
@skier_t

Freelance Security Researcher

June 10, 2013

Introduction

- Who am I?

Introduction

- Who am I?
  - Student (University of Amsterdam)
  - Freelance Security Researcher
  - Cuckoo Sandbox Developer (Malware Analysis System)

Introduction

Android?

Introduction

Android?

- Smartphones
- Runs custom Linux
- Millions of Devices
- Hundreds of thousands of applications
- etc..

Android Applications

Android Applications?

Android Applications

Android Applications?

- Application Package File (APK)
  - Download from Google Play
  - Zip file
  - Some Metadata (Manifest, Images, ..)
  - classes.dex

Android Applications

Android Applications?

- Application Package File (APK)
  - Download from Google Play
  - Zip file
  - Some Metadata (Manifest, Images, ..)
  - classes.dex
- All your code are belong to classes.dex
  - More on this later.

Android Applications

Android Applications?

- Application Package File (APK)
  - Download from Google Play
  - Zip file
  - Some Metadata (Manifest, Images, ..)
  - classes.dex
- All your code are belong to classes.dex
  - More on this later.
- Resources
  - Images
  - Data files
  - Native libraries

Running Code on Android

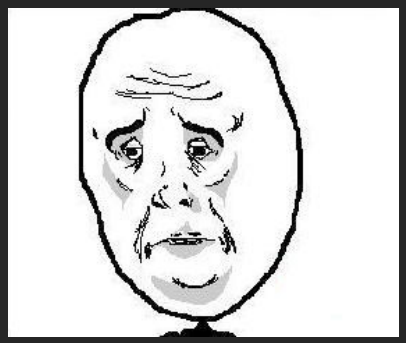
There are two ways.

- Running native libraries
  - Extremely awesome
  - This talk does not focus on native

Running Code on Android

There are two ways.

- Running native libraries
  - Extremely awesome
  - This talk does not focus on native



Running Code on Android

There are two ways.

- Running native libraries
  - Extremely awesome
  - This talk does not focus on native
- Running Dalvik Bytecode
  - Dalvik is Compiled Java
  - Dalvik != Java
  - classes.dex
  - (More on this later)

Dex File Format (I)

- Dalvik Executable Format
- classes.dex
  - Container format to store Dalvik Bytecode with Metadata

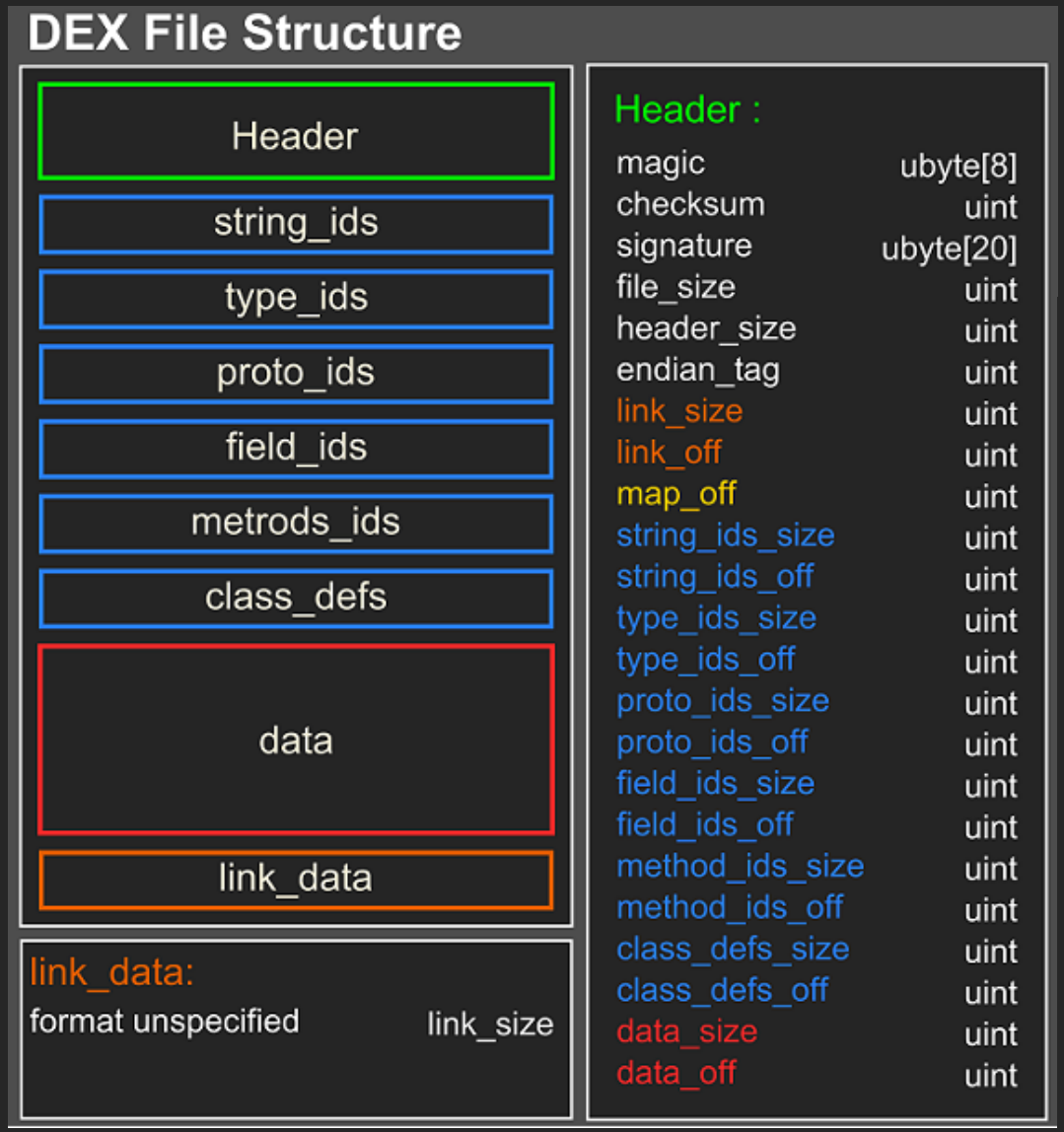
Dex File Format (I)

- Dalvik Executable Format
- classes.dex
  - Container format to store Dalvik Bytecode with Metadata
  - Various Data Pools
    - Strings **"Hello World"**
    - Classes **Ljava/lang/String;**
    - Fields **Ljava/lang/String;->value**
    - Prototypes **(I)Ljava/lang/String;**

Dex File Format (I)

- Dalvik Executable Format
- classes.dex
  - Container format to store Dalvik Bytecode with Metadata
  - Various Data Pools
    - Strings **"Hello World"**
    - Classes **Ljava/lang/String;**
    - Fields **Ljava/lang/String;->value**
    - Prototypes **(I)Ljava/lang/String;**
  - Lots of headers
    - Complex Cross-references between fields and headers
    - The Classname is a String
    - A Prototype has a String as return value
    - A method links to a Prototype, etc..

Dex File Format (II)



Strings

"Hello World"

Classes

Ljava/lang/String;

Fields

Ljava/lang/String;->value

Prototypes

(I)Ljava/lang/String;

By Rodrigo Chiossi

Dalvik Bytecode Example

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

->

Dalvik Bytecode Example

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

->

```
sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
```

Dalvik Bytecode Example

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

->

```
sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
const-string v1, "Hello AthCon"
```

Dalvik Bytecode Example

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

->

```
sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
const-string v1, "Hello AthCon"
invoke-virtual v0, v1,
    Ljava/io/PrintStream;->println(Ljava/lang/String;)V
```

Dalvik Bytecode Example

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

->

```
sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
const-string v1, "Hello AthCon"
invoke-virtual v0, v1,
    Ljava/io/PrintStream;->println(Ljava/lang/String;)V
return-void
```

What's your point?

- Decompiling is mostly trivial
- JEB - `http://android-decompiler.com/`

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

What's your point?

- Decompiling is mostly trivial
- JEB - `http://android-decompiler.com/`

```
public static void hello() {
    System.out.println("Hello AthCon");
}
```

- Smali/Baksmali allows you to quickly modify code
- Based on .smali files, a wrapper around Dalvik bytecode
- Free and Open Source
  `https://code.google.com/p/smali/`

Let's welcome Obfuscators

- Commercial solutions
- Make Reverse Engineering harder
- Make automated analysis harder (what to look at?)

Let's welcome Obfuscators

- Commercial solutions
- Make Reverse Engineering harder
- Make automated analysis harder (what to look at?)
- What can we do..?

Let's welcome Obfuscators

- Commercial solutions
- Make Reverse Engineering harder
- Make automated analysis harder (what to look at?)
- What can we do..?
- Deobfuscate the obfuscated code!

Let's welcome Obfuscators

- Commercial solutions
- Make Reverse Engineering harder
- Make automated analysis harder (what to look at?)
- What can we do..?
- Deobfuscate the obfuscated code!

But first..

Introduction to Our Tools

readdex(1)

- Custom utility to read .dex files
- Not very strict
- Works in cases where traditional tools fail
- E.g., dexdump, dex2jar, sometimes even JEB
- (Will report JEB bugs later)

Introduction to Our Tools

readdex(1)

- Custom utility to read .dex files
- Not very strict
- Works in cases where traditional tools fail
- E.g., dexdump, dex2jar, sometimes even JEB
- (Will report JEB bugs later)
- Handles the following cases correctly
  - Invalid checksum hashes (fails dexdump)
  - Unused opcodes (fails dex2jar/dexdump)
  - Invalid Data Pool Indices (dexdump/dex2jar)
  - Unicode function names (IDA Pro?!)
  - Etc..

Introduction to Our Libraries

- Dalvik Disassembler

Introduction to Our Libraries

- Dalvik Disassembler
- Basic Dalvik Emulator
  - Supports most Dalvik Instructions
  - Supports simple Java Classes (Strings, etc.)

Introduction to Our Libraries

- Dalvik Disassembler
- Basic Dalvik Emulator
  - Supports most Dalvik Instructions
  - Supports simple Java Classes (Strings, etc.)
- Dex File Parser
  - Dex File Creator is Work in Progress

Introduction to Our Libraries

- Dalvik Disassembler
- Basic Dalvik Emulator
  - Supports most Dalvik Instructions
  - Supports simple Java Classes (Strings, etc.)
- Dex File Parser
  - Dex File Creator is Work in Progress
- Totalling more than 5kloc C (including readdex)

Introduction to Our Libraries

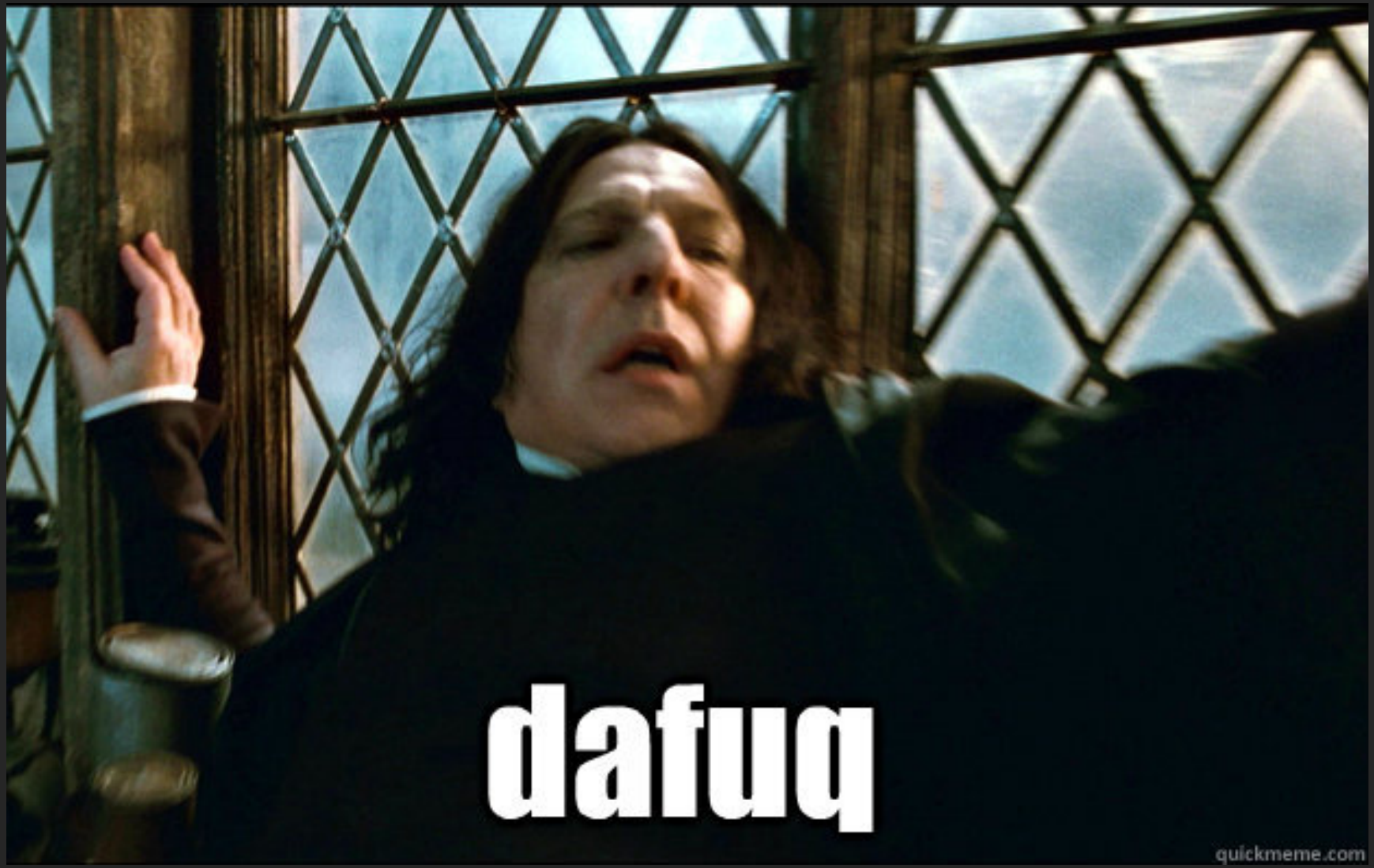
- Dalvik Disassembler
- Basic Dalvik Emulator
  - Supports most Dalvik Instructions
  - Supports simple Java Classes (Strings, etc.)
- Dex File Parser
  - Dex File Creator is Work in Progress
- Totalling more than 5kloc C (including readdex)
- Not to mention basic Python wrappers

Introduction to Our Libraries

- Dalvik Disassembler
- Basic Dalvik Emulator
  - Supports most Dalvik Instructions
  - Supports simple Java Classes (Strings, etc.)
- Dex File Parser
  - Dex File Creator is Work in Progress
- Totalling more than 5kloc C (including readdex)
- Not to mention basic Python wrappers
- All of it will be Open Source soon (TM)

What's next? This stuff is actually useful?

What's next? This stuff is actually useful?



Class & Function Name Obfuscation

Used by for example Dexguard & Freedom.apk..

Class & Function Name Obfuscation

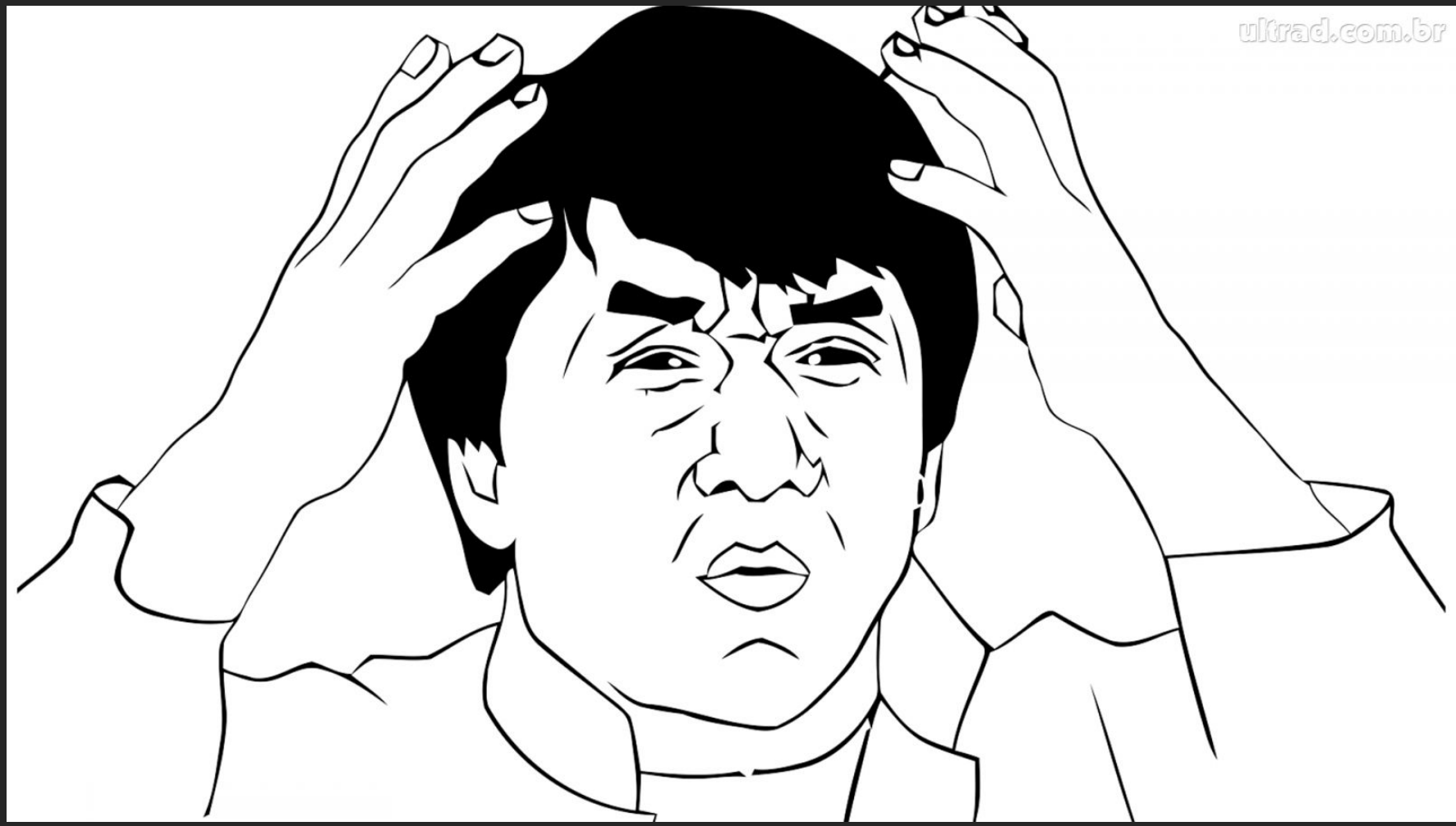
Used by for example Dexguard & Freedom.apk..
Welcome to China..

Class & Function Name Obfuscation

Used by for example Dexguard & Freedom.apk..
Welcome to China..

```
static  {
    int v1 = 0;
    瘴.蘑 = new 瘴("None", 0, 0, "no-error");
    瘴.草 = new 瘴("Generic", 1, 1, "generic-error");
    瘴.蚯 = new 瘴("NoClass", 2, 2, "no-such-class");
    瘴.雪 = new 瘴("NoField", 3, 3, "no-such-field");
    瘴.麦 = new 瘴("NoMethod", 4, 4, "no-such-method");
    瘴.演 = new 瘴("AccessClass", 5, 5, "illegal-class-access");
    瘴.艽 = new 瘴("AccessField", 6, 6, "illegal-field-access");
    瘴.鹅 = new 瘴("AccessMethod", 7, 7, "illegal-method-access");
    瘴.鏽 = new 瘴("ClassChange", 8, 8, "class-change-error");
    瘴.鈁 = new 瘴("Instantiation", 9, 9, "instantiation-error");
    瘴[] v0 = new 瘴[10];
    v0[0] = 瘴.蘑;
    v0[1] = 瘴.草;
    v0[2] = 瘴.蚯;
    v0[3] = 瘴.雪;
    v0[4] = 瘴.麦;
    v0[5] = 瘴.演;
```

Class & Function Name Obfuscation



China?

- Unreadable identifiers
- Problematic when Modifying Dalvik Code (.smali)

China?

- Unreadable identifiers
- Problematic when Modifying Dalvik Code (.smali)
- unchina.py to the rescue!

unchina.py

- Walks the Dex file
- Enumerates all classes and methods
- Renames Chinese names with something readable
- **"zmagic_" + number**
- (For now, can be changed of course..)

unchina.py

- Walks the Dex file
- Enumerates all classes and methods
- Renames Chinese names with something readable
- **"zmagic_" + number**
- (For now, can be changed of course..)
- Simple Python script using some hacky functionality
- Rewrites parts of the Dex file as needed
- Writes a new Dex file (still kind of experimental)
- Sounds easier than it is!

unchina.py Demo

Demo of Unchina.py..

Obfuscated Strings (I)

Used by for example Dexguard, Whatsapp.apk, Freedom.apk

Obfuscated Strings (I)

Used by for example Dexguard, Whatsapp.apk, Freedom.apk

- Instead of using Hardcoded Strings
- Build strings up at runtime

Obfuscated Strings (I)

Used by for example Dexguard, Whatsapp.apk, Freedom.apk

- Instead of using Hardcoded Strings
- Build strings up at runtime
- Makes it harder to analyze
  - Strings usually have meaningful information
  - (Function names, Debug information, URLs, etc.)

Obfuscated Strings (I)

Used by for example Dexguard, Whatsapp.apk, Freedom.apk

- Instead of using Hardcoded Strings
- Build strings up at runtime
- Makes it harder to analyze
  - Strings usually have meaningful information
  - (Function names, Debug information, URLs, etc.)
- More code in the binary
  - Normally one string
  - Now entire functions for decoding, function calls, etc..

Obfuscated Strings (II)

We want to reconstruct the obfuscated strings

Obfuscated Strings (II)

We want to reconstruct the obfuscated strings

- Use our Simple Dalvik Emulator
- Combined with some heuristics (in the future)
- For now a bit hardcoded..

Three different String Obfuscation examples

- Whatsapp.apk
- Freedom.apk
- A Dexguarded binary

Whatsapp (I)

#1 - Whatsapp.apk

- Defines <clinit>for lots of classes
  - Class Initialization function
  - Called when the class is being loaded

Whatsapp (II)

```
static  {
    int v0_1;
    char[] v0 = "G@\u0007?M".toCharArray();
    int v3 = 0;
    int v2 = v0.length;
    char[] v1 = v0;
    while(v2 > v3) {
        int v4 = v1[v3];
        switch(v3 % 5) {
            case 0: {
                v0_1 = 20;
                break;
            }
            case 1: {
                v0_1 = 8;
                break;
            }
            case 2: {
                v0_1 = 70;
                break;
            }
            case 3: {
                v0_1 = 18;
                break;
            }
            default: {
                v0_1 = 124;
                break;
            }
        }

        v1[v3] = ((char)(v0_1 ^ v4));
        ++v3;
    }

    mb.z = new String(v1).intern();
}
```

Whatsapp (III)

- We emulate the method
- Intercept the **sput-object** instruction
  - **sput-object v0, mb->z:Ljava/lang/String;**
- "Assign Static Class Variable"

Whatsapp (III)

- We emulate the method
- Intercept the **sput-object** instruction
  - **sput-object v0, mb->z:Ljava/lang/String;**
- "Assign Static Class Variable"
- We now have the deobfuscated string
- (or multiple strings, in some cases)

Whatsapp (III)

- We emulate the method
- Intercept the **sput-object** instruction
  - **sput-object v0, mb->z:Ljava/lang/String;**
- "Assign Static Class Variable"
- We now have the deobfuscated string
- (or multiple strings, in some cases)
- Roughly 5000 strings deobfuscated!

Freedom (I)

#2 - Freedom.apk

- Has xor decryption methods
- Calls functions with magic decoding value

Freedom (II)

```
if(this.嚎 != 0 || this.賽 != 0) {
    arg10.print(arg9);
    arg10.print(誰.弓(123));
    arg10.print(Integer.toHexString(this.嚎));
    arg10.print(誰.丌(122));
    arg10.println(Integer.toHexString(this.賽));
}
```

```
private static String 弓(byte arg7) {
    int v0 = 11;
    byte[] v1 = new byte[12];
    v1[v0] = 88;
    v1[1] = 62;
    v1[8] = 18;
    v1[5] = 9;
    v1[0] = 22;
    v1[6] = 58;
    v1[7] = 21;
    v1[4] = 30;
    v1[2] = 21;
    v1[9] = 22;
    v1[3] = 15;
    v1[10] = 70;
    do {
        v1[v0] = ((byte)(v1[v0] ^ arg7));
        --v0;
    }
    while(v0 >= 0);

    return new String(v1);
}
```

Freedom (III)

- The xor decryption methods have a specific signature
- Their prototype is always **(B)Ljava/lang/String;**
- (Accepts an 8bit integer, returns a String.)

Freedom (III)

- The xor decryption methods have a specific signature
- Their prototype is always **(B)Ljava/lang/String;**
- (Accepts an 8bit integer, returns a String.)
- We scan every method in the Dex file
- Function Call to Decryption Method ->Decrypt the String

Freedom (III)

- The xor decryption methods have a specific signature
- Their prototype is always **(B)Ljava/lang/String;**
- (Accepts an 8bit integer, returns a String.)
- We scan every method in the Dex file
- Function Call to Decryption Method ->Decrypt the String
- Roughly 600 strings deobfuscated!

Dexguard (I)

#3 - Dexguard is a Commercial Obfuscator
As example we use an obfuscated Cyanide.apk

- Root exploit for some Motorala device
- (Thanks to Justin Case for the sample)

Dexguard (II)

```
public class MainActivity extends Activity {
    private static final byte[] 鵟;
```

```
private static String 鵜(int arg6, int arg7, int arg8) {
    int v3;
    int v2;
    arg7 += 62;
    byte[] v5 = MainActivity.鵟;
    int v4 = 0;
    arg6 += 409;
    byte[] v1 = new byte[arg6];
    if(v5 == null) {
        v2 = arg6;
        v3 = arg8;
    }
    else {
    label_11:
        v1[v4] = ((byte)arg7);
        ++v4;
        if(v4 >= arg6) {
            return new String(v1, 0);
        }
        else {
            v2 = arg7;
            v3 = v5[arg8];
        }
    }

    ++arg8;
    arg7 = v2 + v3 - 8;
    goto label_11;
}
```

Dexguard (III)

- Dexguard initializes a lookup table on <clinit >
- Decrypts strings using this lookup table
- One dedicated decryption method
- Signature **(III)Ljava/lang/String;**

Dexguard (IV)

- Dexguard is a combination of Whatsapp and Freedom
- (With regards to techniques)
- First emulate <clinit >
- To obtain the lookup table

Dexguard (IV)

- Dexguard is a combination of Whatsapp and Freedom
- (With regards to techniques)
- First emulate <clinit >
- To obtain the lookup table
- Then scan every method in the Dex file
- Find function calls to the decryption method
- Decrypt strings!

Dexguard (IV)

Original Dexguarded Cyanide.apk

```
protected void onCreate(Bundle arg5) {
    super.onCreate(arg5);
    this.setContentView(2130903040);
    if(new File(MainActivity.鶸(-387, -15, 608)).exists()) {
        MainActivity.鶸(MainActivity.鶸(-389, 52, 159));
        MainActivity.鶸(MainActivity.鶸(-333, 37, 17));
        MainActivity.鶸(MainActivity.鶸(-407, 53, 629), MainActivity.鶸(-395, -15, 0), this);
        MainActivity.鶸(MainActivity.鶸(-398, 53, 92), MainActivity.鶸(-386, -15, 586), this);
        MainActivity.鶸(MainActivity.鶸(-402, 52, 102), MainActivity.鶸(-378, -15, 665), this);
        MainActivity.鶸(MainActivity.鶸(-368, 37, 119));
        MainActivity.鶸(this);
        return;
    }
```

Rewriting the Dex file (I)

Rewriting Whatsapp, Freedom and Dexguarded Cyanide.apk

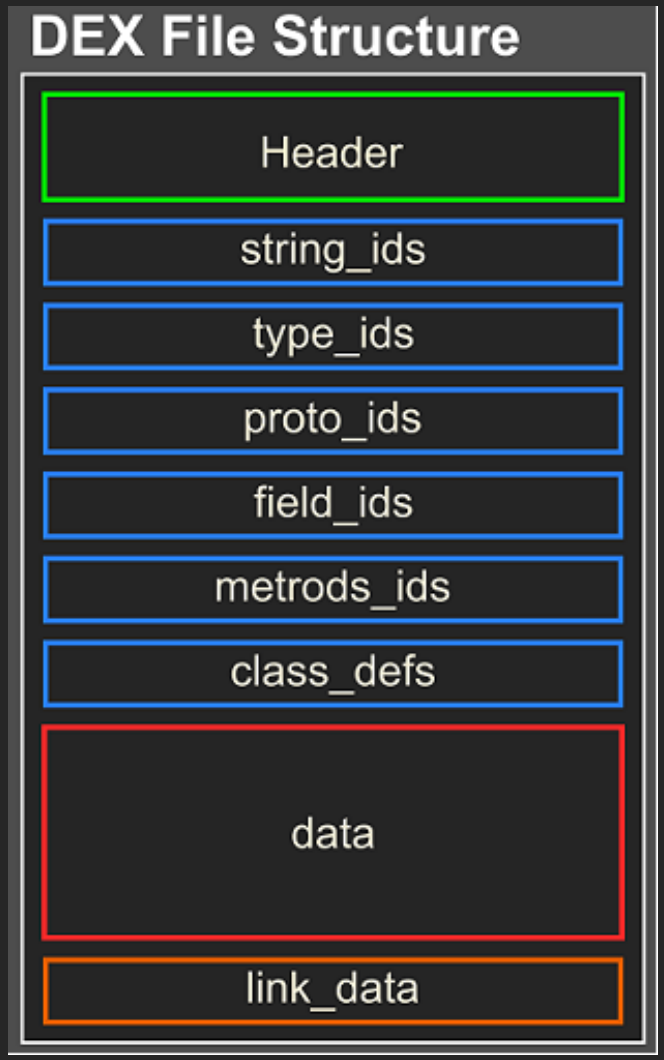
- We have the decrypted strings
- Obfuscated code always takes **more** instructions than deobfuscated code
- Patching time..!

Rewriting the Dex file (II)

Some problems..

- We have to introduce new strings
  - Extend the String Data Pool
  - Shuffle around half the Dex..

Rewriting the Dex file (II)



Rewriting the Dex file (III)

Some problems..

- We have to introduce new strings
  - Extend the String Data Pool
  - Shuffle around half the Dex..

Rewriting the Dex file (III)

Some problems..

- We have to introduce new strings
  - Extend the String Data Pool
  - Shuffle around half the Dex..
- Patch Dalvik instructions (straightforward)
- Remove obsolete functions
  - String Decryption Methods are now unused
  - Quite painful.. Dex file-wise
  - *Work in Progress*

Rewriting the Dex file (IV)

- We move all strings to EOF
- We fixup other data structures
- Demo time

Rewriting the Dex file (V)

Demo of reconstructing Dexguarded Cyanide.apk

How do we go from here?

- Generic Deobfuscation
  - Based on Heuristics with Prototypes etc

How do we go from here?

- Generic Deobfuscation
  - Based on Heuristics with Prototypes etc
- Classification based on stripped down binaries
  - One binary can have many obfuscated representations
  - Deobfuscate to something like the original binary
  - Allows more accurate classification

How do we go from here?

- Generic Deobfuscation
  - Based on Heuristics with Prototypes etc
- Classification based on stripped down binaries
  - One binary can have many obfuscated representations
  - Deobfuscate to something like the original binary
  - Allows more accurate classification
- Did I mention plaintext strings?

How do we go from here?

- Generic Deobfuscation
  - Based on Heuristics with Prototypes etc
- Classification based on stripped down binaries
  - One binary can have many obfuscated representations
  - Deobfuscate to something like the original binary
  - Allows more accurate classification
- Did I mention plaintext strings?
- Plaintext Strings!

Automated Malware Analysis!

Yesterday a new malware was found in the wild..

`http://www.securelist.com/en/blog/8106/The_most_sophisticated_Android_Trojan`

High Expectations Asian Dad strikes again!

Backdoor.AndroidOS.Obad.a

- Seems like a pretty advanced android malware
- Multiple obfuscation layers (for strings)
  - Got a start, but far from complete..
  - *Quick Demo*
- Some Plaintext Strings..
  - Tries to enable Bluetooth
  - getSimSerialNumber
  - ..
  - (I need some more time)

Questions?

Any questions?

Cheers to..
p1ra, nex‘, rep, blasty, thuxnder, diff-, jcase, George, jduck, ..

Interested in Android Security?
Join **#droidsec** on irc.freenode.org (thanks jduck!)