Android: Game of Obfuscation

Jurriaan Bremer & Rodrigo Chiossi

Who?

- Jurriaan Bremer
  - Freelance Security Researcher
  - Student (University of Amsterdam)
  - Self-proclaimed expert at Mobile Security & Low-level
    - Core Developer of Cuckoo Sandbox (http://cuckoosandbox.org/)
    - Author of Open Source ARMv7 Disassembler (http://darm.re/)
  - Eindbazen CTF Team

Who?

- Rodrigo Chiossi
  - Security Researcher @ Samsung.
  - Founder/Maintainer of AndroidXRef.
    - www.androidxref.com
  - Member of SmashTheStack Network.
    - www.smashthestack.org
  - BinaryBandits CTF Team.

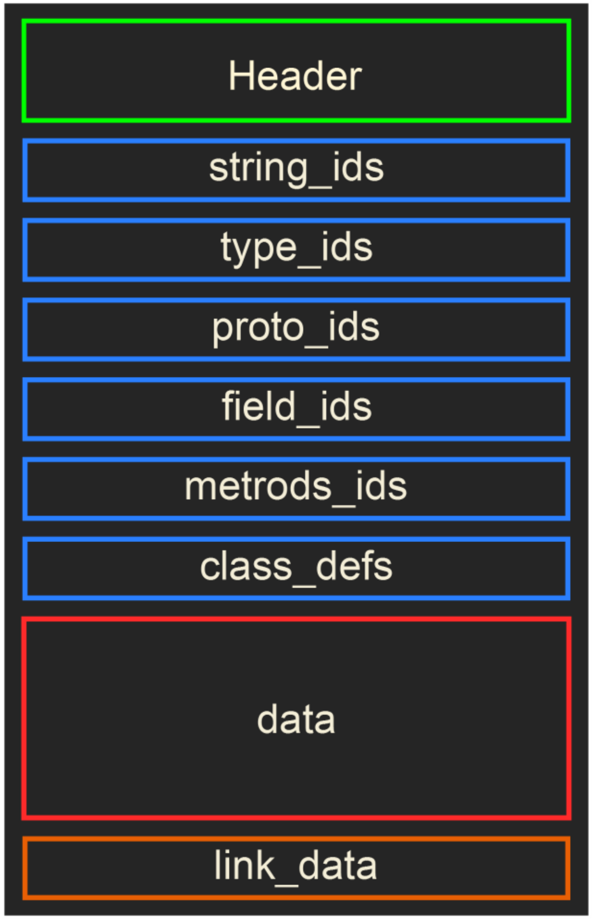
What?

- Instrumentation at DEX level.
- Dalvik emulation
- Obfuscation techniques...
- ..and how to break them.

Why?

- DEX instrumentation is hard!
- Lack of existing tools to operate at DEX level.
- Independency from other tools.
- Obfuscation techniques try to break reversing at higher levels.

DEX Structure



Header

string_ids

type_ids

proto_ids

field_ids

metrods_ids

class_defs

data

link_data

Header :

magic	ubyte[8]
checksum	uint
signature	ubyte[20]
file_size	uint
header_size	uint
endian_tag	uint
link_size	uint
link_off	uint
map_off	uint
string_ids_size	uint
string_ids_off	uint
type_ids_size	uint
type_ids_off	uint
proto_ids_size	uint
proto_ids_off	uint
field_ids_size	uint
field_ids_off	uint
method_ids_size	uint
method_ids_off	uint
class_defs_size	uint
class_defs_off	uint
data_size	uint
data_off	uint

map_item:

type	ushort
unused	ushort
size	uint
offset	uint

Type Codes:

HEADER_ITEM	0x0000
STRING_ID_ITEM	0x0001
TYPE_ID_ITEM	0x0002
PROTO_ID_ITEM	0x0003
FIELD_ID_ITEM	0x0004
METHOD_ID_ITEM	0x0005
CLASS_DEF_ITEM	0x0006
MAP_LIST	0x1000
TYPE_LIST	0x1001
ANNOTATION_SET_REF_LIST	0x1002
ANNOTATION_SET_ITEM	0x1003
CLASS_DATA_ITEM	0x2000
CODE_ITEM	0x2001
STRING_DATA_ITEM	0x2002
DEBUG_INFO_ITEM	0x2003
ANNOTATION_ITEM	0x2004
ENCODED_ARRAY_ITEM	0x2005
ANNOTATIONS_DIRECTORY_ITEM	0x2006

LEB 128

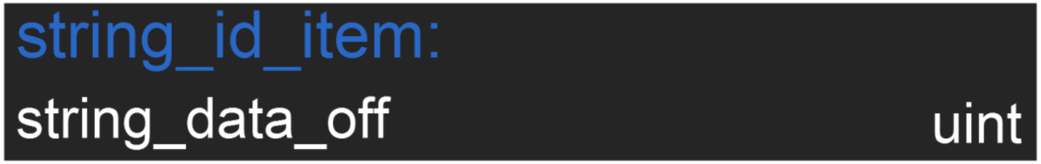
- Encoding format from DWARF3.
- Used to encode signed (SLEB128) and unsigned (ULEB128) numbers.
- Used in DEX for encoding 32-bit numbers.
- **Numbers are encoded using 1 to 5 bytes.**
  - Depending on the highest **'1'**-bit

Case Study: Adding a String

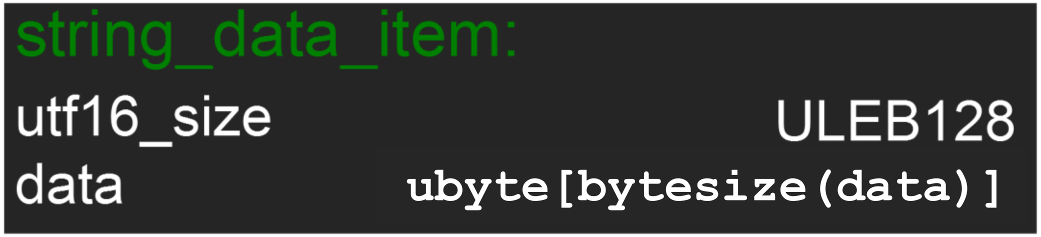
- Most obfuscation techniques involve string manipulation.
- The process of adding strings can be extended to replace/remove strings.
- Objective:
  - Keep the DEX file valid after instrumentation.
  - Pass DexOpt checking (strict verification of DEX files)

String Structure

- Represented by the pair (**string_id_item**, **string_data_item**)
- **string_id_item** list must be sorted
  - Sorted by the utf16 code points of the string
- Strings are referenced by its index position in the **string_id_item** list.



string_id_item:	
string_data_off	uint



string_data_item:	
utf16_size	ULEB128
data	ubyte[bytesize(data)]

Adding a string_id_item

- Must be added in the position of the list that will keep the list sorted.
- Header adjustments:
  - Data offset.
  - File size.
- Maps adjustments:
  - **string_id_item** map size.
- Entire file adjustments:
  - Offsets references in data area must be shifted 4 bytes.
  - String references equal or bigger than the added string must be increased by 1.

LEB128 Expansion

- Some offsets are encoded as ULEB128.
  - E.g. **code_off** inside **encoded_method** object.
- Some string_id_item references are encoded as ULEB128.
  - E.g. **name_idx** inside **annotation_element** object.
- After shifting offsets or increasing **string_id_item** references, the size of the LEB128 in bytes may increase.
- If the expansion occurs, further shifting of offsets is needed in the file.
- Maps size and offset must be updated.

Alignment

- Some structures in the DEX file must be 4-byte aligned.
  - E.g., **code_item**.
- **string_id_item** is 4-byte in size, so adding a new object will not misalign the DEX.
- LEB128 expansion will often add 1 byte shifting, which will break alignment.
- If realignment is required, offset references must be updated.
- Maps size and offset must be updated.

Adding a string_data_item

- Must be inside the data area.
- Header adjustments:
  - Data size.
  - File size.
- Maps adjustments:
  - **string_data_item** map size.
- Entire file adjustments:
  - Offsets references after the offset of the new **string_data_item** must be shifted by the size of the added object.
  - String references equal or bigger than the added string must be increased by 1.
- Check for LEB128 expansion and apply shifting.
- Check for alignment and apply shifting.

Demo: String Addition

Dalvik 101

```
public static void hello() {
    System.out.println(“Hello H2HC”);
}
```

```
sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
const-string v1, ”Hello H2HC”
invoke-virtual v0, v1, Ljava/io/PrintStream;->println(Ljava/lang/String;)V
return-void
```

Dalvik 102

- Register-based Instruction Set
  - Allocates a fixed-size amount of registers for a function
- Various General Purpose Instructions
  - Move, add, subtract, multiply, etc
- Fixed branches
  - No “jump register”, only “goto $+30” and alike
- Class, Static and Array get/put instructions
  - To read/write class members & array indices
- Special: Switch/case, array-length, const-string, ..

Dalvik Analysis

- Dalvik is a fairly limited instruction set
  - Small but powerful instructions
- Easy to analyze all possible code execution paths
  - All possible code paths known through static analysis
  - (Just follow all branches)
  - Recursive Traversal Algorithm for discovering code
    - Get all cross references to instructions for free

Analyzing Cross References

```
$ readdex -D srTools.dex --xref

 index   offset   instruction                   cross-references
 [..]
 #6      @11      new-array v1, v6, type@34     (xref @10)
 #7      @13      if-nez v4, +7                 (xref @11)
 #8      @15      move v2, v7                   (xref @13)
 #9      @16      move v3, v8                   (xref @15)
 #10     @17      add-int/2addr v2, v2, v3      (xref @16, @37)
 #11     @18      add-int/lit8 v8, v2, #+84     (xref @17)
 #12     @20      move v2, v5                   (xref @13, @18)
 #13     @21      add-int/lit8 v7, v7, #+1      (xref @20)
 #14     @23      add-int/lit8 v5, v5, #+1      (xref @21)
 [..]
```

Basic Block Analysis

- Based on Cross-reference Information
  - Assume new basic block when xref count > 1
  - Basic Block ends at **throw**, **return**, **goto**
  - Special case: try/catch handler
- Required for backtracing.. (skip catch handlers, ..)

```
#6     @11     new-array v1, v6, type@34    (xref @10)
#7     @13     if-nez v4, +7                (xref @11)
#8     @15     move v2, v7                  (xref @13)
#9     @16     move v3, v8                  (xref @15)
```

```
#10    @17     add-int/2addr v2, v2, v3     (xref @16, @37)
#11    @18     add-int/lit8 v8, v2, #+84    (xref @17)
```

```
#12    @20     move v2, v5                  (xref @13, @18)
#13    @21     add-int/lit8 v7, v7, #+1     (xref @20)
#14    @23     add-int/lit8 v5, v5, #+1     (xref @21)
```

Walking the Samples

- 1000+ malware samples
- 50+ malware families
- AndroMalShare + Contagio + Private Sources

Unknown (Proprietary)

- Sample: fdad65[..].apk
- Package renaming
  - Compressed to a 6 digits lowercase alpha name
  - E.g., com.h2hc.exemple-> kkkkkk
- Class renaming
  - 6 alpha digits, all lowercase.
  - E.g., com.h2hc.exemple.Main -> kkkkkk.aaaaaj

Unknown (Proprietary)

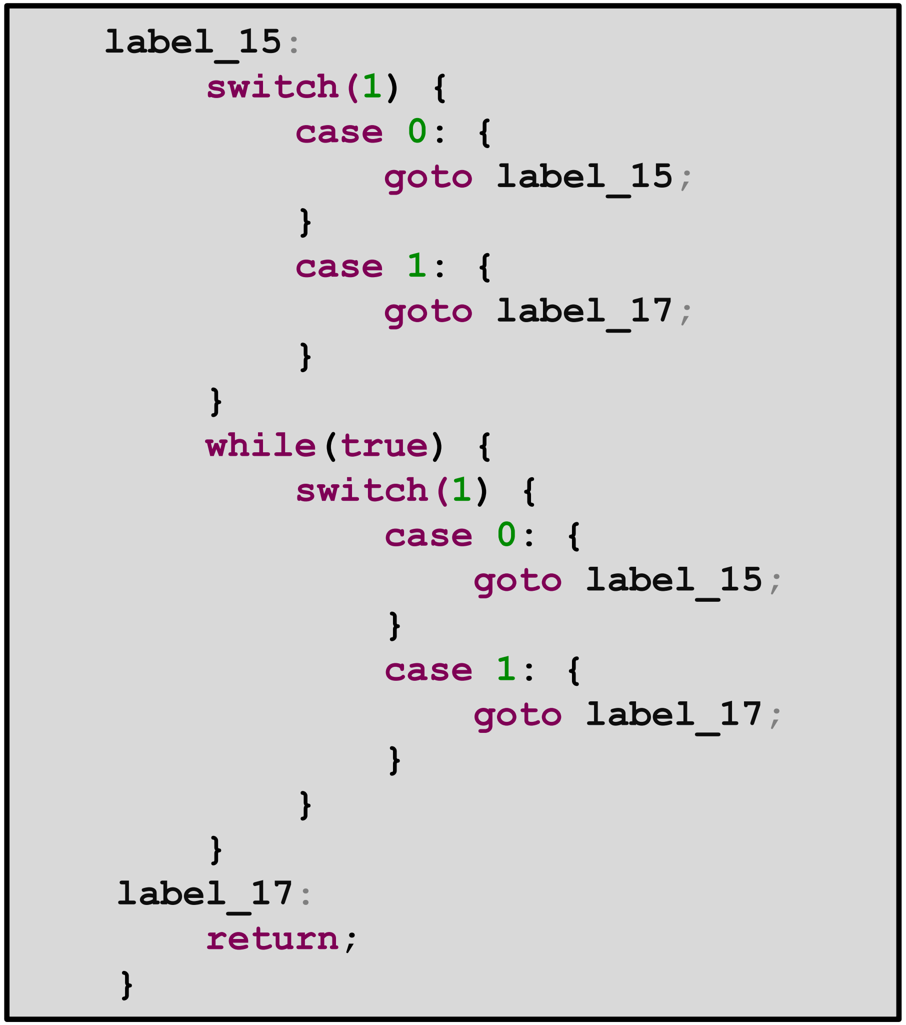
- Methods and variables renaming

```
public class ddpppd {
    private long bй0439й0439йй;

    public long b044A044Aъъ044Aъ() {
        ...
    }
}
```

Unknown (Proprietary)

- Empty switch cases
- Confuses JAD



```
label_15:
    switch(1) {
        case 0: {
            goto label_15;
        }
        case 1: {
            goto label_17;
        }
    }
    while(true) {
        switch(1) {
            case 0: {
                goto label_15;
            }
            case 1: {
                goto label_17;
            }
        }
    }
label_17:
    return;
}
```

Unknown (Proprietary)

- String Encryption
  - Single byte XOR encryption

```
llllqq.b0425XX0425XX("«\u0081\u0096\u009B¿\u008A\u0097\u0
008D\u0088±\u009C", 'ø');
```

```
public static String b0425XX0425XX(String arg5, char arg6)
{
    char[] v1 = arg5.toCharArray();
    char[] v2 = new char[v1.length];
    for (int v0 = 0; v0 < v1.length; ++v0) {
        v2[v0] = (char)(v1[v0] ^ arg6);
    }
    return new String(v2);
}
```

Extracting Function Arguments

```
llllqq.b0425XX0425XX("«\u0081\u0096\u009B¿\u008A\u0097\u
008D\u0088±\u009C", 'ø');
```

```
0000004C  const-string              v4,
"«\u0081\u0096\u009B¿\u008A\u0097\u008D\u0088±\u009C"
00000050  const/16                  v6, 0xF8
00000054  invoke-static llllqq->b0425XX0425XX(String, C)String, v4, v6
0000005A  move-result-object        v4
```

- Backtrace arguments to the function call:
  - Cross-reference information
  - Basic Block Analysis (to ensure correctness)
    - If argument comes from another basic block it may be arbitrary

Emulating the Deobfuscation Function

- Generic deobfuscation with a custom emulator
  - Implements all relevant instructions & Java functions
- Setup the arguments & call the function
  - Arguments extracted statically:

```
0000004C  const-string                v4,
"«\u0081\u0096\u009B¿\u008A\u0097\u008D\u0088±\u009C"
00000050  const/16                    v6, 0xF8
00000054  invoke-static llllqq->b0425XX0425XX(String, C)String, v4, v6
0000005A  move-result-object          v4
          halt execution
```

- Halt execution
- Read string from **v4**
- **"SyncGroupId"**

Replacing the Code

Original Code:

```
0000004C  const-string            v4,
"«\u0081\u0096\u009B¿\u008A\u0097\u008D\u0088±\u009C"
00000050  const/16                v6, 0xF8
00000054  invoke-static llllqq->b0425XX0425XX(String, C)String, v4, v6
0000005A  move-result-object      v4
```

New Code:

```
0000004C  const-string            v4, "SyncGroupId"
```

- Reduced code to a single instruction
- Optimizes the Dex file as well
  - Original string can be removed
  - String deobfuscation function can be removed
  - (But only if the strings and functions are not used elsewhere anymore)

Extracting Function Arguments Part 2

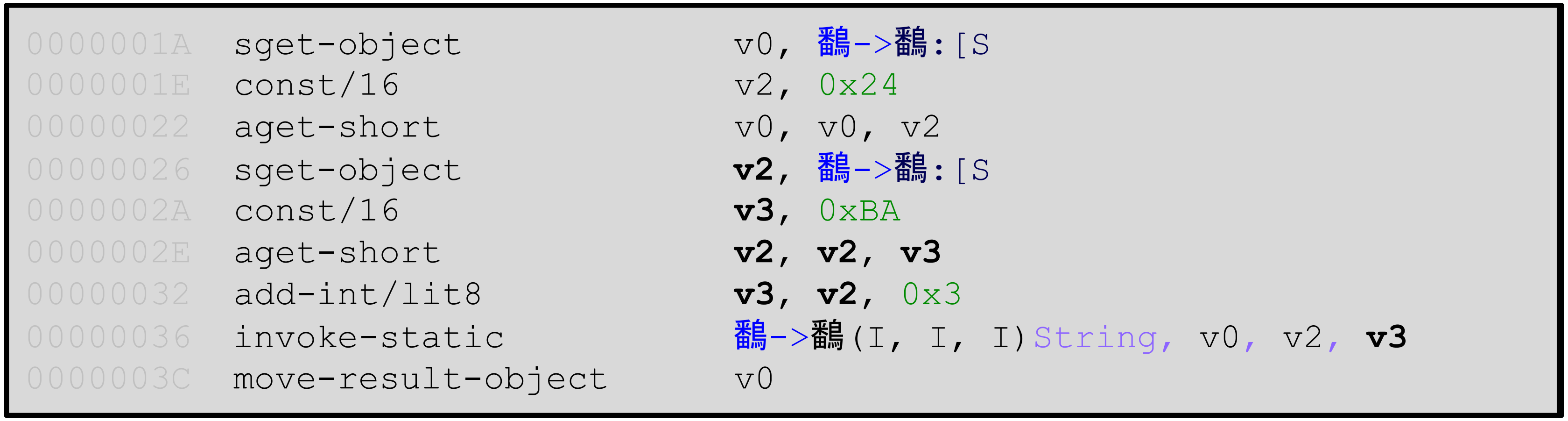
- Not always as straightforward as **const-string** and **const** immediate, i.e., **deobf("string", value)**.

```
v0_1 = 鷭.鷭(鷭.鷭[36], 鷭.鷭[186], 鷭.鷭[186] + 3);
```

- Takes three integers as arguments, returns String
- Integer magic with lookup tables

Extracting Function Arguments Part 2

- Dalvik representation of the function call



```
0000001A  sget-object           v0, 鸐->鸐:[S
0000001E  const/16              v2, 0x24
00000022  aget-short            v0, v0, v2
00000026  sget-object           v2, 鸐->鸐:[S
0000002A  const/16              v3, 0xBA
0000002E  aget-short            v2, v2, v3
00000032  add-int/lit8          v3, v2, 0x3
00000036  invoke-static         鸐->鸐(I, I, I)String, v0, v2, v3
0000003C  move-result-object    v0
```

- Backtrace instructions for each argument
  - Only get what’s interesting for us
  - Allows extraction in the middle of a large function, etc.
- Requires correct initialization of static values

ProGuard

- Most common type of obfuscation used
  - Free to use
- Renames classes and packages.
  - E.g., com.h2hc.test.Main becomes **com.h2hc.test.a**
  - Or, **com.h2hc.a.a**, **com.h2hc.a**, **com.a**, or just **a**
  - Various variations..

DexGuard

- Used by “Most Sophisticated Android Trojan”
  - Android.OBad.A
- Renames classes and identifiers
  - Just like ProGuard
  - Not lowercase ascii, but Chinese characters
    - (Sometimes also other weird characters)
- unchina.py
  - Renames identifier with Chinese names
  - Format: “china_” + counter

Unchina.py

- Our earlier example, with Chinese identifiers

```
v0_1 = 鷭.鷭(鷭.鷭[36], 鷭.鷭[186], 鷭.鷭[186] + 3);
```

- Becomes the following

```
v0_1 = china_0.china_1(china_0.china_2[36],
                       china_0.china_2[186],
                       china_0.china_2[186] + 3);
```

- Not perfect, but easier on the eye
  - Could also use different formats..

mzhengDS

- One of many custom obfuscations
- Uses rot16 to “encrypt” strings

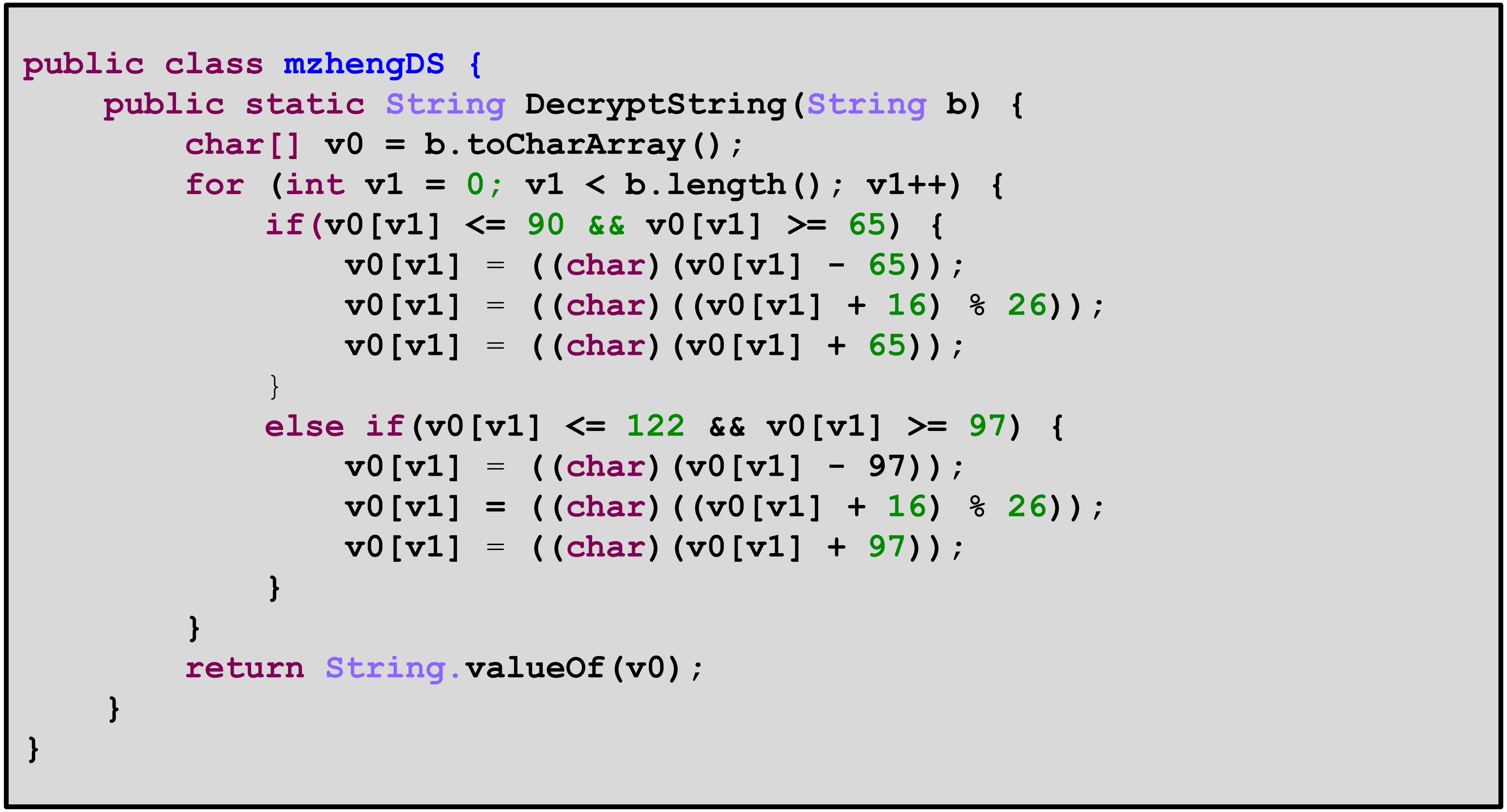
```
if(url.startsWith(mzhengDS.DecryptString("rddz://ggg.iy
edelo.myw/"))) {
```

```
>>> rot16_decode(‘rddz://ggg.iyedelo.myw/’)
‘http://www.youtube.com/'
```

- Encryption is hard!

```
new InetSocketAddress(mzhengDS.DecryptString("10.0.0.172"), 80));
```

mzhengDS



```java
public class mzhengDS {
    public static String DecryptString(String b) {
        char[] v0 = b.toCharArray();
        for (int v1 = 0; v1 < b.length(); v1++) {
            if(v0[v1] <= 90 && v0[v1] >= 65) {
                v0[v1] = ((char)(v0[v1] - 65));
                v0[v1] = ((char)((v0[v1] + 16) % 26));
                v0[v1] = ((char)(v0[v1] + 65));
            }
            else if(v0[v1] <= 122 && v0[v1] >= 97) {
                v0[v1] = ((char)(v0[v1] - 97));
                v0[v1] = ((char)((v0[v1] + 16) % 26));
                v0[v1] = ((char)(v0[v1] + 97));
            }
        }
        return String.valueOf(v0);
    }
}
```

Source Code

- **dexterity** - DEX manipulation library.
  - BSD 3-clause license.
  - Still in early development.
  - https://github.com/rchiossi/dexterity
  - **#dexterity** on Freenode

Jurriaan Bremer
me@jbremer.org
@skier_t

Rodrigo Chiossi
r.chiossi@androidxref.com
@rchiossi

Special thanks to:

Stephan Chenette, jcase, Luanderock, Glauco Junquera,
Daan Raman, Peter Geissler, Rodrigo Branco, r0b, zoidberg