

第二次笔试 自动化 46 何宜晖

1.

$$\begin{aligned} a_0 &= 1 & a_{n+1} &= 24 \left\lfloor \frac{a_n}{8} \right\rfloor + 728 & a_n &= 24 \left\lfloor \frac{a_{n-1}}{8} \right\rfloor + 728 \\ a_n &= 24 \left\lfloor \frac{1}{8} (24 \left\lfloor \frac{a_{n-2}}{8} \right\rfloor + 728) \right\rfloor + 728 \\ a_n &= 24 \left\lfloor 3 \left\lfloor \frac{a_{n-2}}{8} \right\rfloor \right\rfloor + 3 \times 728 + 728 \\ &= 24 \left\lfloor 3 \left\lfloor 3 \left\lfloor \frac{a_{n-3}}{8} \right\rfloor \right\rfloor \right\rfloor + (3^2 + 3 + 1) \cdot 728 & (n \geq 3) \\ &= 24 \left\lfloor 3 \cdots \left\lfloor \frac{a_0}{8} \right\rfloor \cdots \right\rfloor + (3^0 + 3 + \cdots + 3^{n-1}) \cdot 728 \\ \therefore \frac{a_n}{24} &= \left\lfloor \frac{a_0}{8} \right\rfloor = 0 & \therefore a_n &= (3^0 + \cdots + 3^{n-1}) \cdot 728 \\ a_n &= (3^n - 1) \cdot 364 \end{aligned}$$

2015-04-11 23:03

2.

第二步: 使用 apktool decompile apk

Apktool 是一个 decompile 的工具, 使用它就可以将 apk 变为 source file

与 imissTest.apk 相同的文件夹下 shift+right click, w 启动 cmd 按照下图依次键入两个命令进行 decompile

```
C:\Windows\system32\cmd.exe

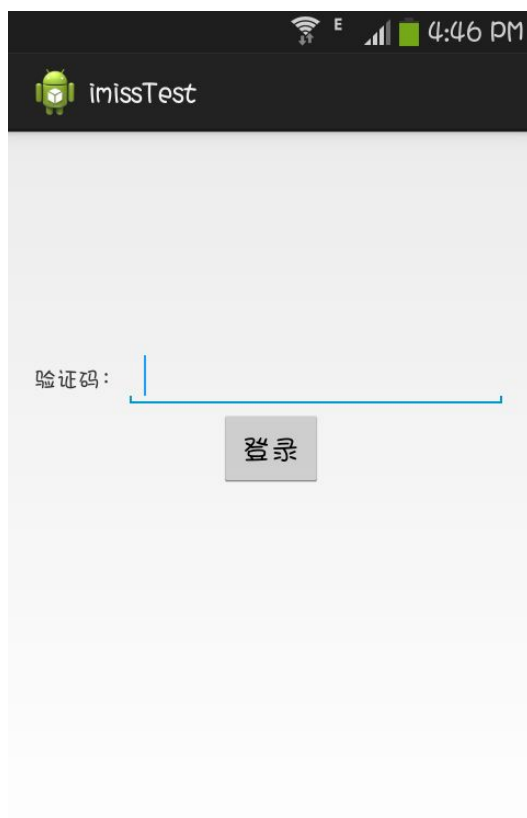
D:\Program Files (x86)\apktool-files>apktool if imissTest.apk
I: Framework installed to: C:\Users\Pomodori\apktool\framework\127.apk

D:\Program Files (x86)\apktool-files>apktool d imissTest.apk
I: Using Apktool 2.0.0-RC2 on imissTest.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Pomodori\apktool\framework\1.apk
I: Regular manifest package...
Cleaning up unclosed ZipFile for archive C:\Users\Pomodori\apktool\framework\1.apk
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

D:\Program Files (x86)\apktool-files>_
```

第二步: 观察源码,安装 apk 到真机

安装这个 apk 后发现, 随便输入几个是进不去的, 所以先看看源码吧..



浏览一下 decompile 后的文件夹

build	4/12/2015 5:17 PM	File folder	
dist	4/13/2015 10:30 AM	File folder	
lib	4/13/2015 9:31 AM	File folder	
original	4/12/2015 12:09 AM	File folder	
res	4/12/2015 12:09 AM	File folder	
smali	4/12/2015 12:09 AM	File folder	
AndroidManifest	4/12/2015 12:09 AM	XML Document	1 KB
apktool.yml	4/12/2015 12:09 AM	YML File	1 KB

Build 是系统自动生成的,lib 是库, original 里面是 manifest, res 一般是用到的图片按钮之类的, smali 应该就是源码了.

注意到 decompile 后的源码是 smali 语言, 了解到 java 转换为 smali 的时候会丢失一些信息, 尤其 class 的信息, 所以 smali 转换为 java 可能会有些问题, 所以就学习一下 smali 语言, 修改这个 project.

android	4/12/2015 12:09 AM	File folder
com	4/12/2015 12:09 AM	File folder

打开 small, android 中是 support -v4 library, com 里面就是我们要找的源码!

BuildConfig.smali	4/12/2015 12:09 AM	SMALI File
HelloAty.smali	4/13/2015 10:09 AM	SMALI File
MainActivity\$1.smali	4/13/2015 10:15 AM	SMALI File
MainActivity.smali	4/13/2015 10:09 AM	SMALI File
R\$attr.smali	4/12/2015 12:09 AM	SMALI File
R\$dimen.smali	4/12/2015 12:09 AM	SMALI File
R\$drawable.smali	4/12/2015 12:09 AM	SMALI File
R\$id.smali	4/12/2015 12:09 AM	SMALI File
R\$layout.smali	4/13/2015 10:09 AM	SMALI File
R\$menu.smali	4/12/2015 12:09 AM	SMALI File
R\$string.smali	4/12/2015 12:09 AM	SMALI File
R\$style.smali	4/12/2015 12:09 AM	SMALI File
R.smali	4/12/2015 12:09 AM	SMALI File

现在大概分析一下这个 app 的结构

目测它只有两个 Activity

R 开头的在 android 中都是 resource, 这个 MainActivity 一定就包含我们想找的 onCreate 入口了, \$1 是他的 sub-class.

Dalvik opcodes

Author: [Gabor Paller](#)

Vx values in the table denote a Dalvik register. Depending on the instruction, 16, 256 or 64k registers can be accessed. Operations on long and double values use two registers, e.g. a double value addressed in the V0 register occupies the V0 and V1 registers.

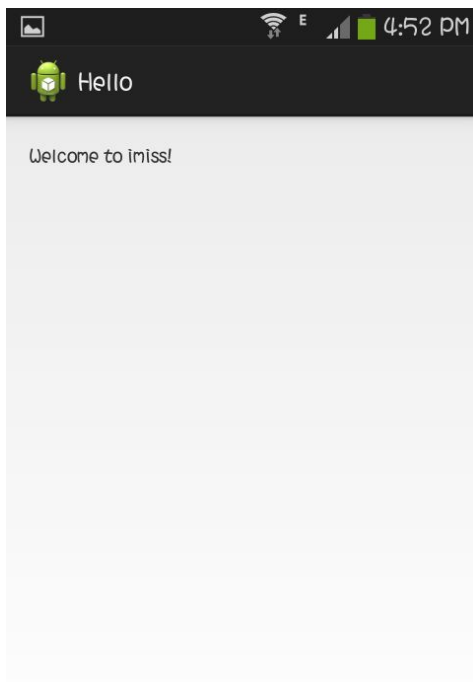
Boolean values are stored as 1 for true and 0 for false. Operations on booleans are translated into integer operations.

All the examples are in hig-endian format, e.g. 0F00 0A00 is coded as 0F, 00, 0A, 00 sequence.

Note there are no explanation/example at some instructions. This means that I have not seen that instruction "in the wild" and its presence/name is only known from [Android opcode constant list](#).

Opcode (hex)	Opcode name	Explanation	Example
00	nop	No operation	0000 - nop
01	move vx,vy	Moves the content of vy into vx. Both registers must be in the first 256 register range.	0110 - move v0, v1 Moves v1 into v0.
02	move/from16 vx,vy	Moves the content of vy into vx. vy may be in the 64k register range while vx is one of the first 256 registers.	0200 1900 - move/from16 v0, v25 Moves v25 into v0.
03	move/16		
04	move-wide		
05	move-wide/from16 vx,vy	Moves a long/double value from vy to vx. vy may be in the 64k register range while vx is one of the first 256 registers.	0516 0000 - move-wide/from16 v22, v0 Moves v0 into v22.
06	move-wide/16		
07	move-object vx,vy	Moves the object reference from vy to vx.	0781 - move-object v1, v8 Moves the object reference in v8 to v1.
08	move-object/from16 vx,vy	Moves the object reference from vy to vx. vy can address 64k registers and vx can address 256 registers.	0801 1500 - move-object/from16 v1, v21 Move the object reference in v21 to v1.
09	move-object/16		
0A	move-result vx	Move the result value of the previous method invocation into vx.	0A00 - move-result v0 Move the return value of a previous method invocation into v0.
0B	move-result-wide vx	Move the long/double result value of the previous method invocation into vx,vx+1.	0B02 - move-result-wide v2 Move the long/double result value of the previous method invocation into v2,v3.
0C	move-result-object vx	Move the result object reference of the previous method invocation into vx.	0C00 - move-result-object v0
0D	move-exception vx	Move the exception object reference thrown during a method invocation into vx.	0D19 - move-exception v25
0E	move-exception/16		

查阅 smali 的指令后发现， 声明一个 string 的语句中含有”string”，
所以我用 sublime text 打开这个 folder. 在整个 project 中搜索 string，找到几处，其中
const-string v2, "imiss111111"这个看起来最像密码了，输入之后果然登录成功.



现在修改源码变得就很简单了

我找到两处 imiss111111 的地方

它们下面的语句里面有个 equals，这一定比较了，那我们把函数的实参改成一样的就 ok 了

```
invoke-virtual {v1, v1}, Ljava/lang/String;.>equals(Ljava/lang/Object;)Z
```

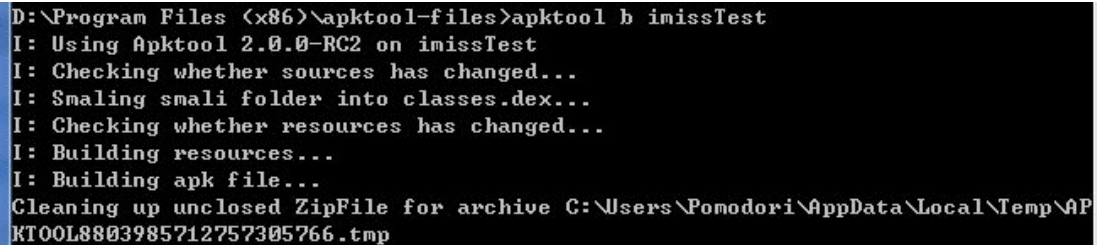
```
invoke-virtual {v2, v2}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
```

```
move-result v3
```

```
if-eqz v3, :cond_0
```

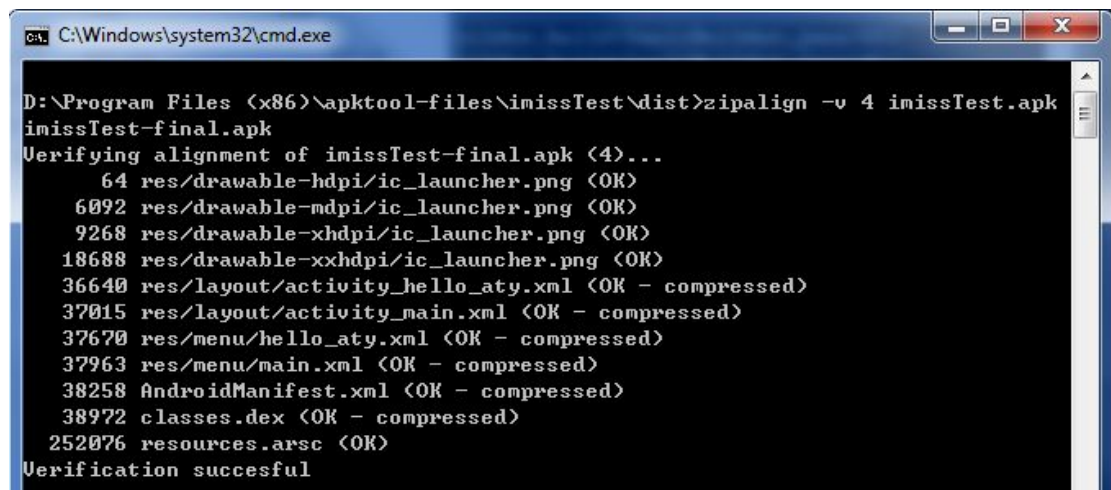
第三步: recompile:

imissTest 文件夹同一文件夹下 Shift+right click ,w 按照下图进行 recompile 得到 imissTest.apk



```
D:\Program Files (x86)\apktool-files>apktool b imissTest
I: Using Apktool 2.0.0-RC2 on imissTest
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
Cleaning up unclosed ZipFile for archive C:\Users\Pomodori\AppData\Local\Temp\APKTOOL8803985712757305766.tmp
```

对 recompile 后的 apk 进行压缩, 得到 imissTest-final.apk, 如果不进行压缩, app 运行时会对系统资源占用较大.



```
C:\Windows\system32\cmd.exe
D:\Program Files (x86)\apktool-files\imissTest\dist>zipalign -v 4 imissTest.apk
imissTest-final.apk
Verifying alignment of imissTest-final.apk (4)...
 64 res/drawable-hdpi/ic_launcher.png (OK)
6092 res/drawable-mdpi/ic_launcher.png (OK)
 9268 res/drawable-xhdpi/ic_launcher.png (OK)
18688 res/drawable-xxhdpi/ic_launcher.png (OK)
36640 res/layout/activity_hello_aty.xml (OK - compressed)
37015 res/layout/activity_main.xml (OK - compressed)
37670 res/menu/hello_aty.xml (OK - compressed)
37963 res/menu/main.xml (OK - compressed)
38258 AndroidManifest.xml (OK - compressed)
38972 classes.dex (OK - compressed)
252076 resources.arsc (OK)
Verification succesful
```

第四步: signature

将这个 apk 放到真机中, 发现未安装, 卸载原来的 apk 发现还是无法安装. 想到用 eclipse 或者 android studio 开发的时候, 安装 apk 时 最后的 signature 都是自动进行的, 在这里并没有.

于是下载 keytool 生成.keystore, 参照下图:


```
C:\Windows\system32\cmd.exe

D:\Program Files (x86)\apktool-files\imissTest\dist>keytool -genkey -v -keystore
my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 1
0000
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: RP
What is the name of your organizational unit?
[Unknown]: Itech
What is the name of your organization?
[Unknown]: Ite
What is the name of your City or Locality?
[Unknown]: Xi
What is the name of your State or Province?
[Unknown]: an
What is the two-letter country code for this unit?
[Unknown]: CH
Is CN=RP, OU=Itech, O=Ite, L=Xi, ST=an, C=CH correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) wi
th a validity of 10,000 days
    for: CN=RP, OU=Itech, O=Ite, L=Xi, ST=an, C=CH
Enter key password for <alias_name>
    <RETURN if same as keystore password>:
Re-enter new password:
[Storing my-release-key.keystore]
```

再使用 jarsigner 进行 signature 时 ,发现它需要 JDK1.7, 只好将 1.8 先卸载, 去 Oracle 安装 1.7

```
C:\Users\Pomodori>java -version
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
```

之后再使用 jarsigner signature 发现还出现问题, 它说未能找到 sun.....类,
在 stackoverflow 上查了一下, 把 jdk7 下面的 tools 复制过来即可
最后再次使用 jarsigner, 就成功签名了.

安装到真机, 实验成功, 任意输入一个 string 都能成功登入
最后我的破解 apk 是 imissTest.apk

```
C:\Windows\system32\cmd.exe

D:\Program Files (x86)\apktool-files\imissTest\dist>jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore imissTest-final.apk
alias_name
Enter Passphrase for keystore:
  adding: META-INF/MANIFEST.MF
  adding: META-INF/ALIAS_NA.SF
  adding: META-INF/ALIAS_NA.RSA
signing: res/drawable-hdpi/ic_launcher.png
signing: res/drawable-mdpi/ic_launcher.png
signing: res/drawable-xhdpi/ic_launcher.png
signing: res/drawable-xxhdpi/ic_launcher.png
signing: res/layout/activity_hello_aty.xml
signing: res/layout/activity_main.xml
signing: res/menu/hello_aty.xml
signing: res/menu/main.xml
signing: AndroidManifest.xml
signing: classes.dex
signing: resources.arsc
jar signed.

Warning:
No -tsa or -tsacert is provided and this jar is not timestamped. Without a times
tamp, users may not be able to validate this jar after the signer certificate's
expiration date (2042-08-29) or after any future revocation date.

D:\Program Files (x86)\apktool-files\imissTest\dist>
```

3. 我的总体思想是用递归实现的，将等待匹配的字符串分割成 只含一种字符的字符串，如:aabcbddd--aa b c ddd. 再用模式字符串进行匹配，匹配时，模式字符串也分割成份 如: a*.bb+ 有三种分法: 1: a*. bb+ 2: a* .bb+ 3:a* . bb+ 模式字符串有多种分法时，利用递归进行分叉。程序运行时，按照顺序依次判断分割后的字符串。

```
#include <stdio.h>
#include <iostream>
#include <string>
#include <list>
#include <fstream>
#define INF 10000

using namespace std;

bool match(string a, string b){

    list<char> pattern;
    int size = 0;
    if (b.empty() && a.empty()) // 如果两个字符串已空，说明可匹配
```

```

{
    return true;
}
else if (!b.empty() && a.empty()) // 待匹配字符串未空，模式字符串已空，不可匹配
{
    return false;
}
else if (b.empty() && !a.empty()) // 待匹配字符串已空，模式字符串未空，有可能能
匹配
{
    while (!a.empty())
    {
        pattern.push_back(a.back());
        a.pop_back();
    }
    goto end;
}
char chab = b.back();
while (!b.empty() && chab == b.back()) //取出待匹配字符串尾部的只含一种字符的字
符串
{
    b.pop_back();
    size++;
}
//取出模式字符串直到遇见不同的字母
while (!a.empty() && (a.back() == chab || a.back() == '.' ||
a.back() == '+' || a.back() == '*'
|| a.back() == '?'))
{
    pattern.push_back(a.back());
    a.pop_back();
}
if (pattern.back() == '+' || pattern.back() == '*' || pattern.back() == '?')
//这几个符号属于前一个字符，归还
{
    a.push_back(pattern.back());
    pattern.pop_back();
}
end:
int dots = 0;
while (!pattern.empty() && pattern.back() == '.') //将'.'分离出，它是可
变的参数
{
    pattern.pop_back();

```



```

        dots++;
    }
    int min = 0, max = 0;
    while (!pattern.empty()) //计算取出的模式字符串的可变长度
    {
        //? 0 1 . single * 0~INF + 1~INF
        if (pattern.front() == '?')
        {
            pattern.pop_front();
            if (max != INF) max++;
        }
        else if (pattern.front() == '*')
        {
            pattern.pop_front();
            if (max != INF)
            {
                max = INF;
            }
        }
        else if (pattern.front() == '+')
        {
            pattern.pop_front();
            if (max != INF)
            {
                max = INF;
            }
            min++;
        }
        else if (pattern.front() == '.')
        {
            min++;
            if (max != INF)
            {
                max++;
            }
        }
        else// if (pattern.front() == chab)
        {
            min++;
            if (max != INF)
            {
                max++;
            }
        }
    }
}

```

```

        pattern.pop_front();
    }

    bool result = false;

    for (int i = 0; i < dots; i++)
    {
        a.push_back('.');
    }

    for (int i = 0; i <= dots; i++) //根据加'.'的多少, 进行分叉
    {
        if ((max + i) < size || (min + i) > size)
        {
            if (!a.empty()) a.pop_back();
            continue;
        }
        result |= match(a, b);
        if (!a.empty()) a.pop_back();
    }
    return result;
}

```

```

int main(){

    string pattern, str;
    bool result;
    fstream f("sample.txt");

    while (true)
    {
        f >> pattern;

        if (!pattern.compare("END"))
        {
            break;
        }
        f >> str;
        /*cin >> pattern;
        cin >> str;*/
        result = match(pattern, str);
        if (result){

```

```
        cout << "true" << endl;
    }
    else cout << "false" << endl;
}
return 0;

}.
```