2015阿里&看雪移动安全挑战赛-第二题

原创

scoronepion 于 2016-05-07 21:26:26 发布 2808 收藏 2

分类专栏： 安卓逆向 文章标签： 安卓逆向 移动安全 移动安全挑战赛 writeup

2015阿里&看雪移动安全挑战赛-第二题

题目传送门：AliCrackme

网上已经有很多writeup,我也是按照乌云上的2015移动安全挑战赛（阿里&看雪主办）全程回顾的基本思路来想的。但作为一个新手,就算照着教程来做也会踩到很多坑。所以我想把自己解题过程中遇到的一些细节问题跟大家分享一下。

文章中提到的环境配置是按照看雪论坛非虫的《Android软件安全与逆向分析》配置的，这本书很棒，讲的很详细。在这里安利一波。

0x01

app安装完后长这样：

输入错误的密码会提示校验码错误。我们先看看这次日志会不会有什么输出。

```
com.yaotong.crackme     yaotong          SecurityCheck Started...
com.yaotong.crackme     yaotong          SecurityCheck Started...
com.yaotong.crackme     yaotong          SecurityCheck Started...
com.yaotong.crackme     yaotong          SecurityCheck Started...
com.yaotong.crackme     yaotong          SecurityCheck Started...
com.yaotong.crackme     yaotong          SecurityCheck Started...
com.yaotong.crackme     Choreographer    Skipped 186 frames!  The application may be doing too much work on its main t □
                                         hread.
```

可以发现，每次输入密码提交后，日志总会输出 `SecurityCheck Started...`，结合上一题的经历，我们能够从代码中发现日志输出语句。
那么我们就来看看从jar文件中能够发现什么。

0x02

```
public void onClick(View paramAnonymousView)
{
  String str = MainActivity.this.inputCode.getText().toString();
  if (MainActivity.this.securityCheck(str))
  {
    Intent localIntent = new Intent(MainActivity.this, ResultActivity.class);
    MainActivity.this.startActivity(localIntent);
    return;
  }
  Toast.makeText(MainActivity.this.getApplicationContext(), "验证码校验失败", 0).show();
}
```

从代码中可以看到，校验成功与否是由`securityCheck()`这一方法的返回值决定的。而在它的Java代码中，我没有找到这一方法，也没有找到日志输出的语句。后来看乌云上的资料，在反编译后的lib文件夹下找到了`libcrackme.so`文件。根据提示，用IDA打开了这个文件。

(注：某些版本的IDA不支持调试so文件，这里我用的是6.6版本)

```
text:000011A8 ; =============== S U B R O U T I N E =======================================
text:000011A8
text:000011A8
text:000011A8                 EXPORT Java_com_yaotong_crackme_MainActivity_securityCheck
text:000011A8 Java_com_yaotong_crackme_MainActivity_securityCheck
text:000011A8
text:000011A8 var_20          = -0x20
text:000011A8 var_1C          = -0x1C
text:000011A8
text:000011A8                 STMFD   SP!, {R4-R7,R11,LR}
text:000011AC                 SUB     SP, SP, #8
text:000011B0                 MOV     R5, R0
text:000011B4                 LDR     R0, =(_GLOBAL_OFFSET_TABLE_ - 0x11C8)
text:000011B8                 LDR     R6, =(unk_6290 - 0x5FBC)
text:000011BC                 MOV     R4, R2
text:000011C0                 ADD     R0, PC, R0 ; _GLOBAL_OFFSET_TABLE_
text:000011C4                 ADD     R0, R6, R0 ; unk_6290
```

发现了`securityCheck()`方法，继续往下找，我们在0x1284处发现了日志输出函数`android_log_print`

```
.text:00001284                BL      __android_log_print
.text:00001288                LDR     R0, [R5]
.text:0000128C                MOV     R1, R4
.text:00001290                MOV     R2, #0
.text:00001294                LDR     R3, [R0,#0x2A4]
.text:00001298                MOV     R0, R5
.text:0000129C                BLX     R3
.text:000012A0                LDR     R1, =(off_628C - 0x5FBC)
.text:000012A4                LDR     R2, [R1,R7] ; off_628C
.text:000012A8
.text:000012A8 loc_12A8                               ; CODE XREF: Java_com_yaotong_crackme_MainActivity_securityCheck+120↓
.text:000012A8                LDRB    R3, [R2]
```

0x03

那么，`securityCheck`的执行流程是怎样的？从乌云上的文章解释得很详细：

> 在securityCheck这个方法调用前，在init_array段和JNI_Onload函数里程序都做了些处理，而在securityCheck方法的最后有一个判断，将用户输入和wojiushidaan做比较。尝试直接输入wojiushidaan，发现密码错误，因此可以猜测前面一大段逻辑的作用就是会把这个最终的字符串改掉。此时的思路是只需知道最终判断时候这个wojiushidaan地址上的变换后的值就行了。尝试使用IDA调试发现一旦attach上去，整个程序就退出，想必一定是在之前的代码中有反调试的代码。

既然我们要得到变换后的值，我们就可以借助日志输出函数将我们想要得到的值输出出来就好了，因此，我们需要对so文件进行一下修改。

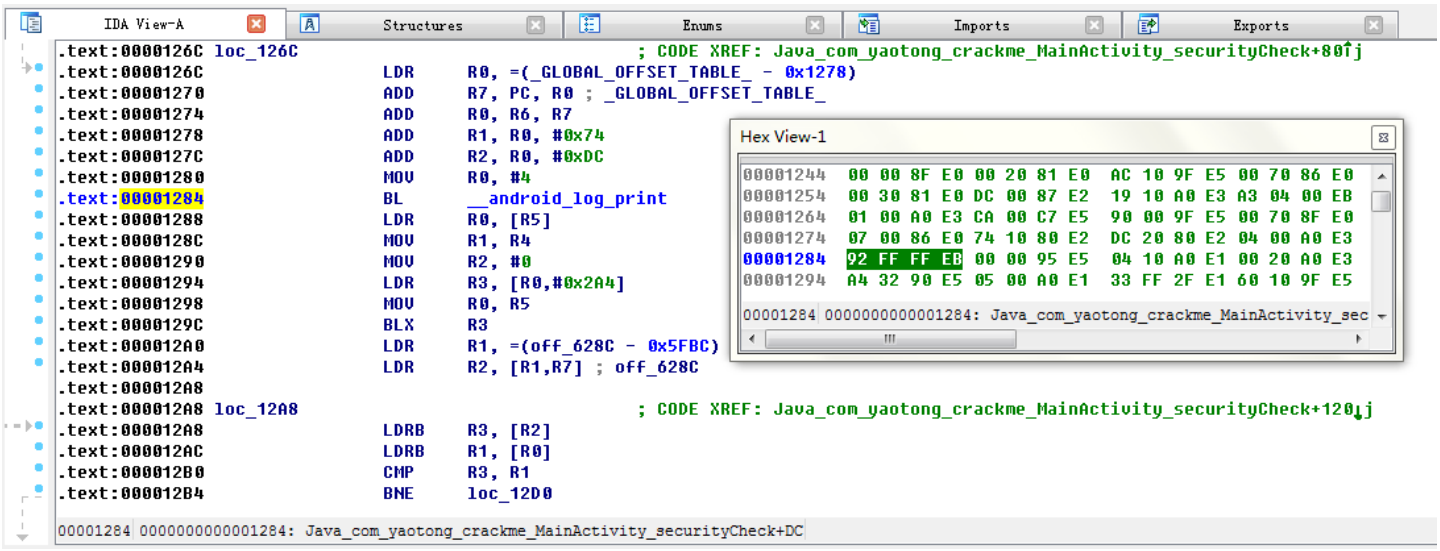
- 将从0x1284到0x129C处都用NOP改写（NOP为空操作）
- 在0x12AC处调用`android_log_print`函数
- 为了不影响R1的值，把0x12A0处的R1改成R3
- 将0x12A4处的R1改成R3
- 将0x12A8处指令改成`MOV R0,#4`

因为so为二进制文件，所以我们修改的也是二进制。

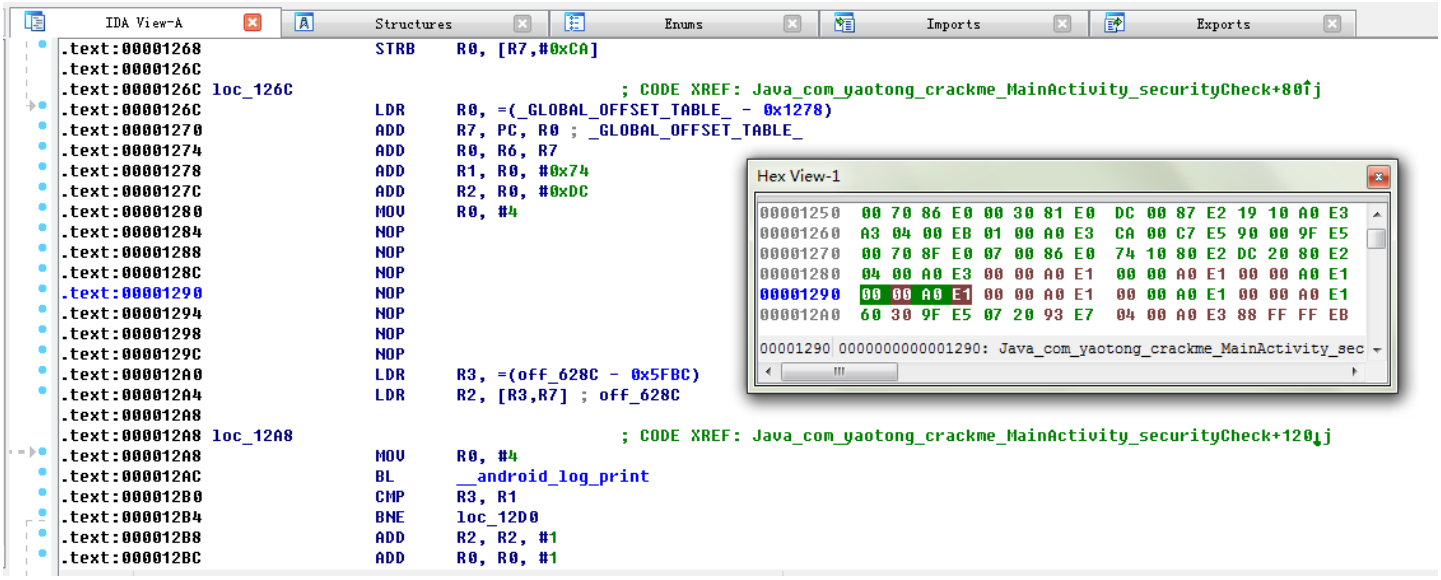
- NOP对应的二进制值为00 00 A0 E1
- `android_log_print`对应的二进制值为88 FF FF EB
- 0x12A0处的值应改为60 30 9F E5
- 0x12A4处的值应改为07 20 93 E7
- `MOV R0,#4`的值为04 00 A0 E3

在IDA中，选中某一行，进入Hex View选项卡就可以看到该行的二进制数值。按下F2键可以进行编辑，编辑好后再按下F2键可以进行保存。此时回到IDA View界面就可以看到程序已经改好了。但注意，IDA中的修改只是保存在了它自己的数据库里，并没有保存到so文件里。所以为了能够直接编辑so文件，我们可以使用UltraEdit来进行编辑。在IDA中确认无误后可将修改内容通过UE保存。

改之前代码：



改之后代码：



0x04

so文件修改完保存好后，将它与原来的反编译文件一起重新打包签名，生成一个新apk。然后将apk重新安装到虚拟机上。运行后观察输出日志。

```
com.android.phone       dalvikvm        GC_CONCURRENT freed 387K, 5% free 11369K/11847K, paused 105ms+22ms, total 342 □
                                        ms
com.yaotong.crackme     yaotong         aiyou,bucuoo
system_process          dalvikvm        Jit: resizing JitTable from 4096 to 8192
```

可以发现，日志输出了一个字符串`aiyou,bucuoo`，经验证，该字符串为正确的密码。

自毁程序密码

Congratulations!!!You Win!!