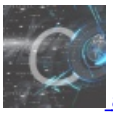
强网杯WriteUp | pictureLock

原创

坚强的女程序员 于 2018-03-30 19:33:54 发布 587 收藏

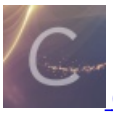
分类专栏： android CTF 文章标签： android 解密

本文链接：https://blog.csdn.net/qq_33438733/article/details/79733233


android 同时被 2 个专栏收录

34 篇文章 0 订阅

订阅专栏


CTF

61 篇文章 4 订阅

订阅专栏

前言

通往结果的路途是最美好的！珍惜在这过程中你所遇到的人和经历。不负遇见！

分析

这道题我感觉只是套了个apk的外壳，一点都没有做Android题目的感觉。本质上是一个aes和一个异或的解密。难受。

```
private String j()
{
  int j = 0;
  try
  {
    Object localObject1 = getPackageManager().getPackageInfo("com.a.sample.picturelock", 64).signatures;
    Object localObject2 = MessageDigest.getInstance("MD5");
    int k = localObject1.length;
    int i = 0;
    while (i < k)
    {
      ((MessageDigest)localObject2).update(localObject1[i].toByteArray());
      i += 1;
    }
```


j()用于获取apk签名的MD5值

```
  ((Cursor)localObject).moveToFirst();
  paramIntent = ((Cursor)localObject).getString(((Cursor)localObject).getColumnIndex(paramIntent[0]));
  ((Cursor)localObject).close();
  enc(paramIntent, getFilesDir().getAbsolutePath() + paramIntent.substring(paramIntent.lastIndexOf("/")) + ".lock", j());
  i();
  Toast.makeText(this, String.format("%s encrypting", new Object[] { paramIntent }), 1).show();
}
```


关键是这个enc()是一个native方法。动态调。

动态调试可以参考这篇文章

http://blog.csdn.net/qq_33438733/article/details/79516372

主要就是这个sub_5fa37a48这个函数

```
int __fastcall Java_com_a_sample_picturelock_MainActivity_enc(int a1, int a2, int a3, int a4, int a5)
{
  int v5; // r5@1
  int v6; // r8@1
  int v7; // r9@1
  char *v8; // r4@1
  int v9; // r6@1
  int v10; // r11@2

  v5 = a1;
  v6 = a4;
  v7 = a3;
  v8 = (char *)(*(int (**)(void))(*(_DWORD *)a1 + 676))();
  v9 = (*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v5 + 676))(v5, v6, 0);
  if ( !dword_5FA3C00C )
  {
    dword_5FA3C00C = (int)malloc(0x20u);
    v10 = (*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v5 + 676))(v5, a5, 0);
    _aeabi_memcpy(dword_5FA3C00C, v10, 32);
    (*(void (__fastcall **)(int, int, int))(*(_DWORD *)v5 + 680))(v5, a5, v10);
  }
  sub_5FA37A48(v8);
  (*(void (__fastcall **)(int, int, char *))(*(_DWORD *)v5 + 680))(v5, v7, v8);
  return (*(int (__fastcall **)(int, int, int))(*(_DWORD *)v5 + 680))(v5, v6, v9);
```


进去之后就是一大堆加密算法。看了下特征，发现一个类似S盒的数据块，猜测是aes加密。

```
for ( i = 0; ; ++i )
{
  v28 = *(_BYTE *)(dword_5FA3C00C + (i & 0x1F));
  v29 = fread(v24, 1u, *(_BYTE *)(dword_5FA3C00C + (i & 0x1F)), v25);
  v30 = v29;
  if ( !v29 )
    break;
```

0x60A4E4B0

fread在读取数据时会按着指定序列读取，这个序列也就是签名的MD5值。这个值可以通过动态调smali的方法拿到。刚开始我用smali插桩的方法，导致我在尝试aes解密的时候，密匙出错，所以卡了很久。因为如果修改了smali，重新打包后签名会变化，所以得到的密匙也就不一样。

```
196     add-int/lit8 v0, v0, 0x1
197
198     goto :goto_1
199
200     :cond_2
201     invoke-virtual {v2}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
202     :try_end_0
203     .catch Landroid/content/pm/PackageManager$NameNotFoundException; {:try_start_0 .. :try_end_0} :catch_1
204     .catch Ljava/security/NoSuchAlgorithmException; {:try_start_0 .. :try_end_0} :catch_0
205
206     move-result-object v0
207     invoke-static {v0}, LMyCrack;->LogStr(Ljava/lang/String;)V
208     :goto_2
209     return-object v0
210
```

Variables

this = {MainActivity@830036466944}

Watches

v0 = "e89b158e4bcf988ebd09eb83f5378e87"

这是获取二次打包的密钥的截图。正确的密钥其实可以字节从内存中获取。

```
  }
  v32 = &dword_5FA3C008;
  if ( !(v28 & 1) )
    v32 = &dword_5FA3C004;
  v33 = *v32;
```


比较关键地方，v32是用于AES的密匙，在这里他做了判断

```
v28 = *(_BYTE *)(dword_5FA3C00C + (i & 0x1F));
```

if (!(v28 & 1))
也就是每次的密匙都会发生变化，查看&dword_5FA3C008;和&dword_5FA3C004;的内容可以看到他这里对原本32位的MD5值分别取了前半段和后半段。

```
v54 = *((_BYTE *)v24 + 15);
sub_5FA3732C(&v39, v33);
sub_5FA373B0(&v39);
sub_5FA375C8(&v39);
sub_5FA3762C(&v39);
sub_5FA3732C(&v39, v33 + 16);
sub_5FA373B0(&v39);
sub_5FA375C8(&v39);
sub_5FA3762C(&v39);
sub_5FA3732C(&v39, v33 + 32);
sub_5FA373B0(&v39);
sub_5FA375C8(&v39);
sub_5FA3762C(&v39);
sub_5FA3732C(&v39, v33 + 48);
sub_5FA373B0(&v39);
sub_5FA375C8(&v39);
sub_5FA3762C(&v39);
sub_5FA3732C(&v39, v33 + 64);
sub_5FA373B0(&v39);
```

这一大段就是AES，不用管。(估计应该是用标准的AES进行加密的，如果不是那就GG)

```
if ( v30 >= 0x11 )
{
  v34 = 16;
  v35 = dword_5FA3C00C;
  do
  {
    v26[v34] = *((_BYTE *)v24 + v34) ^ *(_BYTE *)(v35 + v34 % 32);
    ++v34;
  }
  while ( v34 < v30 );
}
```

这里就是进行异或加密了，这里只对数据的前16个字节进行了aes加密，而后对剩余的字节进行异或加密。根据加密的过程写出解密代码。

```
# -*- coding: UTF-8 -*-
from Crypto.Cipher import AES
from binascii import b2a_hex
from binascii import a2b_hex
from aes_util import *
def dec_xor_data(cipher):
        plain=[]
        for i in range(16,len(cipher)):
                temp=chr(ord(cipher[i]) ^ ord(akey[i%32]))
                plain.append(temp)
        return "".join(plain)
akey="f8c49056e4ccf9a11e090eaf471f418d"
dword_5FA3C00C=[0x66,0x38,0x63,0x34,0x39,0x30,0x35,0x36,0x65,0x34,0x63,0x63,0x66,0x39,0x61,0x31,0x31,0x
def main():
        fp_enc=open("flag.jpg.lock","rb")
        fp_dec=open("de.jpg","wb")
        i=0
        while 1:
                v24 = ord(akey[i&0x1f])
                cipher=fp_enc.read(v24)
                cipher_aes=b2a_hex(cipher[:16])
                if  (v24&0x1)==0:
                        hkey=akey[:16]
                        plain_aes=decrypt_ECB(cipher_aes,hkey)
                else:
                        hkey=akey[16:]
                        plain_aes=decrypt_ECB(cipher_aes,hkey)
                if len(plain_aes) < 16:
                        plain_aes+=chr(0)*(16-len(plain_aes))
                fp_dec.write(plain_aes)
                plain_xor=dec_xor_data(cipher)
                fp_dec.write(plain_xor)
                i=i+1
                if len(cipher)<1:
                        break
        fp_dec.close()
if __name__=="__main__":
        main()
```

总结

这题最难的地方是解aes加密，刚开始也不确定，所以就自己加密了一张图片，然后测试前16个字节进行比对。拿github上的aes尝试了一下，不对，然后找了一个python的解密脚本，成了！
最后解密图片，拿到flag