逆向BSides SF CTF之flagstore.apk

原创

caiqiiqi 于 2017-08-21 21:20:51 发布 2340 收藏

分类专栏： Android逆向

本文链接：https://blog.csdn.net/caiqiiqi/article/details/77460651

看有一个writeup写了个装drozer, 然后尝试了一下并没有成功。感觉虽然功能不错，但是有点麻烦。然后

参考：

https://ctf.rip/bsides-sf-ctf-2017-flag-receiver-mobile-reverse-engineering/

https://russtone.io/2017/02/14/bssidessf-2017-flagreceiver/

在`MainActivity`中发现启动之后就向`Send_to_Activity`这个广播接收器注册了一个action为`com.flagstore.ctf.INCOMING_INTENT`的广播，

```
public class MainActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(getApplicationContext());
        tv.setText("To-do: UI pending");
        setContentView(tv);
        IntentFilter filter = new IntentFilter();
        filter.addAction("com.flagstore.ctf.INCOMING_INTENT");
        registerReceiver(new Send_to_Activity(), filter, permission._MSG, null);
    }
}
```


而且界面只有一个`to-do: UI pending`，在`Send_to_Activity`的`onReceive()`方法中可以发现，需要发送一个满足特定条件的广播才能激活进入下一个Activity。

```
public class Send_to_Activity extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        if (intent.getStringExtra("msg").equalsIgnoreCase("OpenSesame")) {
            Log.d("Here", "Intent");
            context.startActivity(new Intent(context, CTFReceiver.class));
            return;
        }
        Toast.makeText(context, "Ah, ah, ah, you didn't say the magic word!", 1).show();
    }
}
```


这个特定条件是：

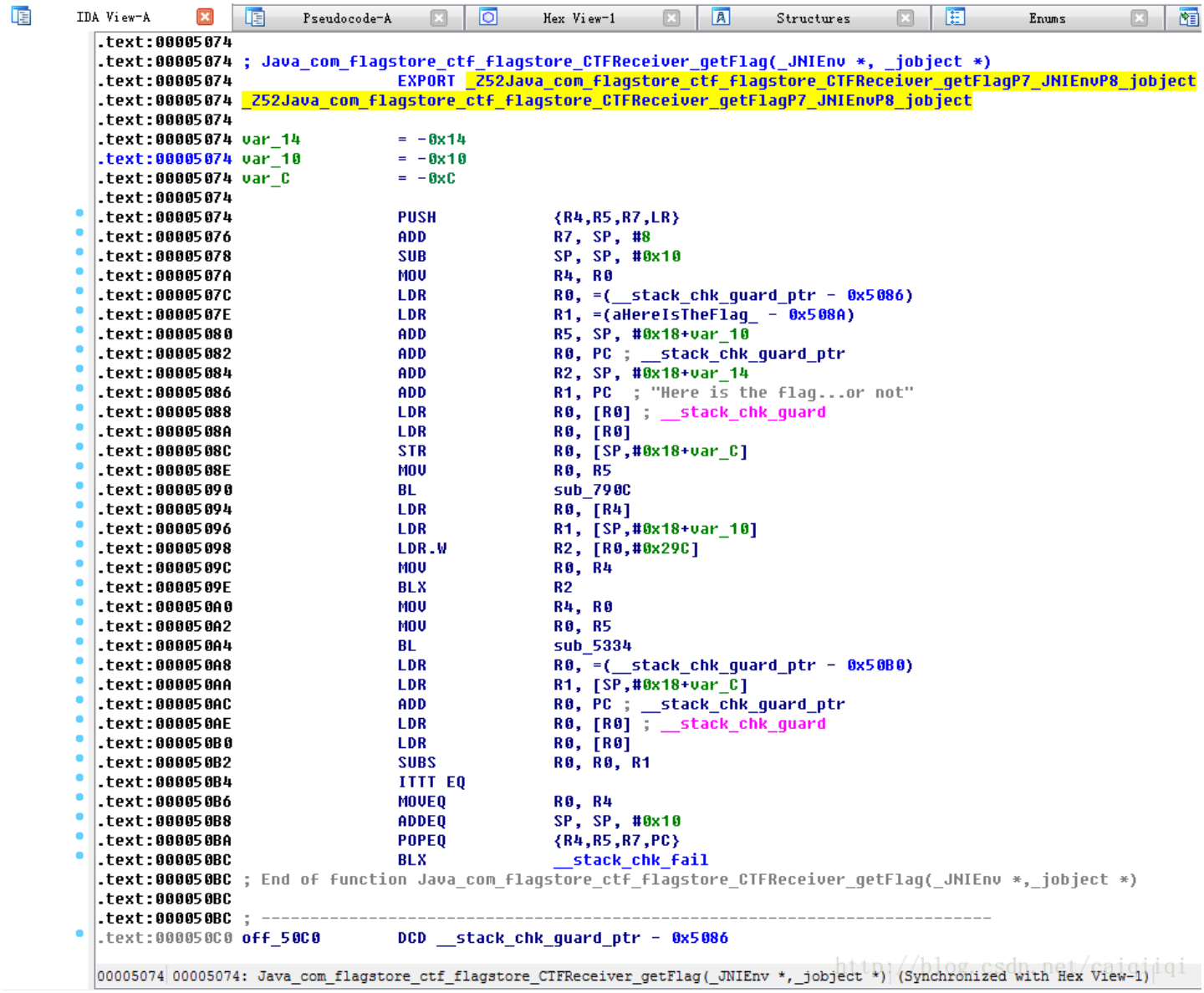
extra中的msg值为：`OpenSesame`。然后调用android的`am`工具。不过记住要用root权限，仅仅在adb shell am... 是不能触发的。

```
adb shell
su
am broadcast -a "com.flagstore.ctf.INCOMING_INTENT" --es msg "OpenSesame"
```

然后就可以看到界面跳转了。然后出现一个很大的按钮可以点击，然而点击之后，应用就崩溃了

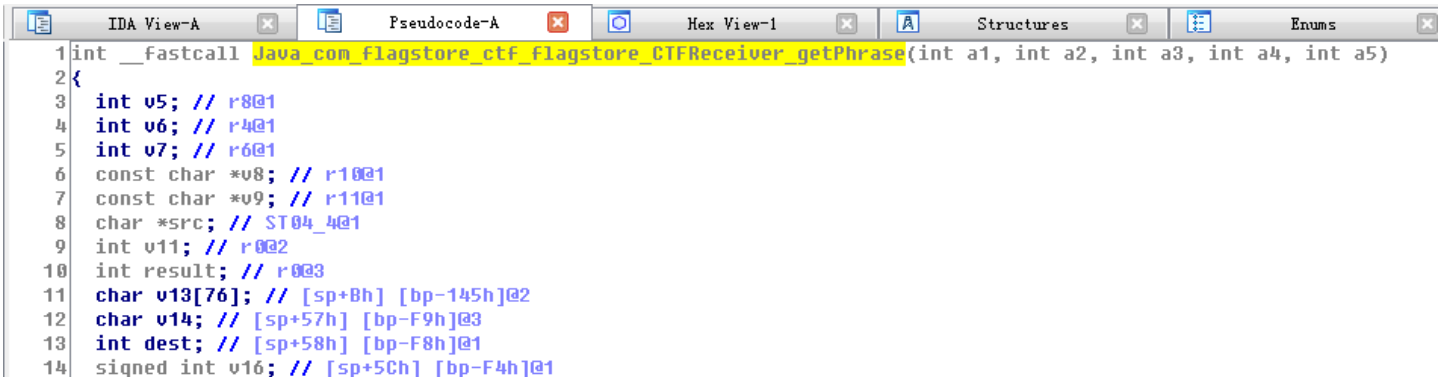
于是静态分析源码。

把这个`lib\armeabi-v7a\libnative-lib.so`文件拖到IDA中去



找到`Exports`就可以找到一个函数`getPhrase()`

然后双击，进入该函数，然后按F5是不行的，搜了一下才知道这时候需要点击右键，然后`Create Function`。这样再按F5就可以得到C代码了。



```
15  signed int v17; // [sp+60h] [bp-F0h]@1
16  signed int v18; // [sp+64h] [bp-ECh]@1
17  signed int v19; // [sp+68h] [bp-E8h]@1
18  signed int v20; // [sp+6Ch] [bp-E4h]@1
19  char v21; // [sp+70h] [bp-E0h]@1
20  char v22[77]; // [sp+A6h] [bp-AAh]@1
21  char v23[77]; // [sp+F3h] [bp-5Dh]@1
22  int v24; // [sp+140h] [bp-10h]@1
23
24  v5 = a1;
25  v6 = a4;
26  v7 = 0;
27  v24 = _stack_chk_guard;
28  v8 = (const char *)(*(int (**)(void))(*(_DWORD *)a1 + 676))();
29  v9 = (const char *)(*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v5 + 676))(v5, v6, 0);
30  src = (char *)(*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v5 + 676))(v5, a5, 0);
31  dest = 1216233792;
32  v16 = 1313163366;
33  v17 = 2003059012;
34  v18 = 1700753218;
35  v19 = 1349085773;
36  v20 = 1987932780;
37  v21 = aAHfhendadwbo_e[24];
38  strncat((char *)&dest, v8, 0x33u);
39  strncpy(v23, v9, 0x4Cu);
40  strncpy(v22, src, 0x4Cu);
41  do
42  {
43    v11 = (unsigned __int8)v23[v7] ^ (unsigned __int8)v22[v7] ^ *((_BYTE *)&dest + v7);
44    v13[v7] = v11;
45    printf("%c\n", (unsigned __int8)v11);
46    ++v7;
47  }
```

00004F6C Java_com_flagstore_ctf_flagstore_CTFReceiver_getPhrase:1


```
23
24  v5 = a1;
25  v6 = a4;
26  v7 = 0;
27  v24 = _stack_chk_guard;
28  v8 = (const char *)(*(int (**)(void))(*(_DWORD *)a1 + 676))();
29  v9 = (const char *)(*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v5 + 676))(v5, v6, 0);
30  src = (char *)(*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v5 + 676))(v5, a5, 0);
31  dest = 1216233792;
32  v16 = 1313163366;
33  v17 = 2003059012;
34  v18 = 1700753218;
35  v19 = 1349085773;
36  v20 = 1987932780;
37  v21 = aAHfhendadwbo_e[24];
38  strncat((char *)&dest, v8, 0x33u);
39  strncpy(v23, v9, 0x4Cu);
40  strncpy(v22, src, 0x4Cu);
41  do
42  {
43    v11 = (unsigned __int8)v23[v7] ^ (unsigned __int8)v22[v7] ^ *((_BYTE *)&dest + v7);
44    v13[v7] = v11;
45    printf("%c\n", (unsigned __int8)v11);
46    ++v7;
47  }
48  while ( v7 != 76 );
49  v14 = 0;
50  printf("Here is your Reply: %s", v13);
51  result = (*(int (__fastcall **)(int, char *))(*(_DWORD *)v5 + 668))(v5, v13);
52  if ( _stack_chk_guard != v24 )
53    _stack_chk_fail(result, _stack_chk_guard - v24);
54  return result;
55}
```

00004F6C Java_com_flagstore_ctf_flagstore_CTFReceiver_getPhrase:9

其中`__stack_chk_guard`和`__stack_chk_fail()`是GCC的堆栈保护机制，其中`__stack_chk_guard`叫作CANARY(金丝雀)值。

> 我们发现这一随机值是放在了函数的局部变量和保存的指令指针（译注：此处指返回地址和EBP）之间。这个值被称作金丝雀(“canary”)值，指的是矿工曾利用金丝雀来确认是否有气体泄漏，如果金丝雀因为气体泄漏而中毒死亡，可以给矿工预警。
> http://www.freebuf.com/articles/system/24177.html
> 在#27行中，将 `_stack_chk_guard` 保存在v24中，然后再在函数返回之前，#52，#53行将之前保存的值与现在的 `_stack_chk_guard` 比较，若两者不一致，金丝雀（canary）的值被修改了，栈溢出发生了，保存的指令指针可能也被修改了，因此不能安全返回，则执行 `_stack_chk_fail()` ，然后会丢出一个错误，退出进程。

IDA小技巧

注意这里的dest, v16, v17, v18, v19, v20等是十进制，可以在IDA里选择将其改成16进制或者字符串。

```
31  dest = 'H~A@';
32  v16 = 'NEHf';
33  v17 = 'wdAD';
34  v18 = 'e_oB';
35  v19 = 'PijM';
36  v20 = 0x767D726C;
37  v21 = aAHfhendadwbo_e[24];
38  strncat((char *)&dest, v8, 0x33u);
39  strncpy(v23, v9, 0x4Cu);
40  strncpy(v22, src, 0x4Cu);
41  do
```


关于 `strncat`

头文件：`#inclue <string.h>`
strncat()用于**将n个字符追加到字符串的结尾**，其原型为：

```
char * strncat(char *dest, const char *src, size_t n);
```

strncat()将会从字符串src的开头拷贝n 个字符到dest字符串尾部，dest要有足够的空间来容纳要拷贝的字符串。如果n大于字符串src的长度，那么仅将src全部追加到dest的尾部。
strncat()会将dest字符串最后的'\0'覆盖掉，字符追加完成后，再追加'\0'。
返回字符串dest。

关于 `strncpy`

头文件：`#include <string.h>`
strncpy()用来**复制字符串的前n个字符**，其原型为：

```
char * strncpy(char *dest, const char *src, size_t n);
```

【参数说明】dest 为目标字符串指针，src 为源字符串指针。
【返回值】返回字符串dest。

关于跨平台的移植性

> int类型比较特殊，具体的字节数同机器字长和编译器有关。如果要保证移植性，尽量用__int16 __int32 __int64吧。__int16、__int32这种数据类型在所有平台下都分配相同的字节。所以在移植上不存在问题。

建议：在代码中尽量避免使用int类型，根据不同的需要可以用short,long,unsigned int 等代替。
http://www.cppblog.com/xyjzsh/archive/2010/10/20/130554.html
怪不得很多IDA出来的C有很多__int8, __int16, __int32等等。

然后顺便反编译了`getFlag`

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1

```
1  int __fastcall Java_com_flagstore_ctf_flagstore_CTFReceiver_getFlag(int a1)
2  {
3    int v1; // r4@1
4    int v2; // r4@1
5    char v4; // [sp+4h] [bp-14h]@1
6    int v5; // [sp+8h] [bp-10h]@1
7    int v6; // [sp+Ch] [bp-Ch]@1
8
9    v1 = a1;
10   v6 = _stack_chk_guard;
11   sub_790C(&v5, "Here is the flag...or not", &v4);
12   v2 = (*(int (__fastcall **)(int, int))(*(_DWORD *)v1 + 668))(v1, v5);
13   sub_5334(&v5);
14   if ( _stack_chk_guard != v6 )
15     _stack_chk_fail(_stack_chk_guard - v6, v6);
16   return v2;
17 }
```