RE-CTF_100

原创

iriris 于 2019-10-11 20:51:45 发布 358 收藏

分类专栏： 逆向 文章标签： CTF 安卓逆向 IDA调试

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/weixin_42011443/article/details/102385650

版权

逆向 专栏收录该内容

18 篇文章 0 订阅

订阅专栏

安装并打开CTF_100.apk之后，发现题目为爬到指定的楼层然后才能显示FLAG：

先使用JEB2反编译apk查看逻辑：

```
public void Btn_up_onclick(View arg6) {
    ++this.has_gone_int;
    this.findViewById(2131492948).setText("" + this.has_gone_int);
    if(this.to_reach_int <= this.has_gone_int) {
        this.findViewById(2131492950).setClickable(true);
    }
}
```

爬一层楼的点击事件：首先已爬楼层数+1并更新已爬楼层数，紧接着判断是否达到要爬楼层数，如果达到，就设置爬到了，看FLAG的点击结果为true。

```
public void btn2_onclick(View arg4) {
```

```
    }

    public native String get_flag(int arg1) {
    }
```

最终看到FLAG按钮的点击事件，可以看到最终的返回FLAG方法在native层实现。我们可以使用IDA进行分析。

```
    protected void onCreate(Bundle arg7) {
        super.onCreate(arg7);
        this.setContentView(2130968601);
        this.findViewById(2131492950).setClickable(false);
        this.has_gone_int = 0;
        Random v1 = new Random();
        this.to_reach_int = v1.nextInt();
        while(true) {
            if(this.to_reach_int < 0) {
                this.to_reach_int *= -1;
            }

            if(5 < this.to_reach_int) {
                this.to_reach_int %= 32;
                this.to_reach_int *= 16384;
                this.findViewById(2131492947).setText("" + this.to_reach_int);
                this.findViewById(2131492951).setText("");
                return;
            }

            this.to_reach_int = v1.nextInt();
        }
    }
}
```

https://blog.csdn.net/weixin_42011443

onCreate方法的实现，主要包含以下几步：看FLAG按钮的点击事件设置为flase，只有当达到要爬楼层数才设置点击事件可见；要爬楼层数设置为随机数；while循环里面是对随机数的一些操作。如果是负数或者大于5时先模32求余再乘以16384。

分析完成之后进行解题：

（1）最简单的是对java层进行修改并重新打包

首先想到的就是在onCreate方法启动时修改要看Flag的点击事件结果为true

```
.local v0, "bt":Landroid/widget/Button;
invoke-virtual {v0, v5}, Landroid/widget/Button;->setClickable(Z)V
```

```
protected void onCreate(Bundle arg7) {
    super.onCreate(arg7);
    this.setContentView(2130968601);
    this.findViewById(2131492950).setClickable(false);
```

对比smali和java，可见控制参数的寄存器为v5,修改寄存器初始值，重新编译，提示：

```
> .class输出目录: F:\环境安装\ApkIDE最新3.5.0少月增强版20170808\ApkIDE最新3.5.0少月增强版\Worksrc\com.ctf.test.ctf_100
> 正在编译Apk...  -
  - 失败:
W: libpng error: Not a PNG file
W: ERROR: Failure processing PNG image F:\环境安装\ApkIDE最新3.5.0少月增强版20170808\ApkIDE最新3.5.0少月增强版\Work\com.ctf.test.ctf_100\res\mipmap-x
Exception in thread "main" brut.androlib.AndrolibException: brut.androlib.AndrolibException: brut.common.BrutException: could not exec (exit code = 1
        at brut.androlib.Androlib.buildResourcesFull(Androlib.java:496)
        at brut.androlib.Androlib.buildResources(Androlib.java:430)
        at brut.androlib.Androlib.build(Androlib.java:329)
        at brut.androlib.Androlib.build(Androlib.java:267)
        at brut.apktool.Main.cmdBuild(Main.java:230)
        at brut.apktool.Main.main(Main.java:83)
```

百度下知道了原来是：

你用apktool b回编译的时候会报错，看第一个错就可以了，不知道怎么插图~~第一个错是

[!]libpng error: Not a PNG file

ERROR: Failure processing PNG image /home/mindmac/Repackage/CqmamsMobile1.1.6/res/drawable-hdpi/station.png[/!]

其实就是res/drawable-hdpi/station.png不是一个PNG格式的文件，用file命令可以看到，其实是个windows的icon文件，所以把这个文件重新命名为station.icon就可以重新打包了！

这个主要是Apktool在回编译时会检查是否是png后缀的文件，如果是，会按png文件进行重打包处理，这样对于不是png文件来说，肯定出错了！这个技巧是今年xkungfoo深圳会议上，玩命提出来的，想不到竟然今天遇到了！

https://blog.csdn.net/weixin_42011443

用010editor查看对用位置的png文件发现其实文件为jpeg格式，修改图片格式后重新编译：

https://blog.csdn.net/weixin_42011443

（2）由于前面分析时提到返回FALG的方法是在native层实现，使用IDA调试so文件：
静态分析so文件发现方法为动态注册，并且搜索不到get_flag方法（自己注册的实现方法，因此函数名称可能会发生变化）。
使用IDA进行动态调试,由于是动态注册，所以定位到JNI_OnLoad方法：

```
1  signed int __fastcall JNI_OnLoad(int a1)
2  {
3    int v2; // [sp+0h] [bp-18h]
4    const char *v3; // [sp+4h] [bp-14h]
5    const char *v4; // [sp+8h] [bp-10h]
6    int (__fastcall *v5)(JNIEnv *); // [sp+Ch] [bp-Ch]
7
8    v2 = 0;
9    v3 = "get_flag";
10   v4 = "(I)Ljava/lang/String;";
11   v5 = sub_74F86F90;
12   if ( (*(int (**)(void))(*(_DWORD *)a1 + 24))() || !(*(int (**)(void))(*(_DWORD *)v2 + 24))() )
13     return -1;
14   (*(void (**)(void))(*(_DWORD *)v2 + 860))();
15   return 65540;
16 }
```

https://blog.csdn.net/weixin_42011443

继续跟进，发现返回flag字符串的方法比较复杂，不过最后终究会返回一个字符串：

```
}
while ( v11 != 8 );
v25 = 0;
result = ((int (__fastcall *)(JNIEnv *, char *))(*v1)->NewStringUTF)(v1, v24);
if ( v26 != _stack_chk_guard )
  result = ((int (__fastcall *)(int))unk_74F86E18)(result);
return result;
}
```

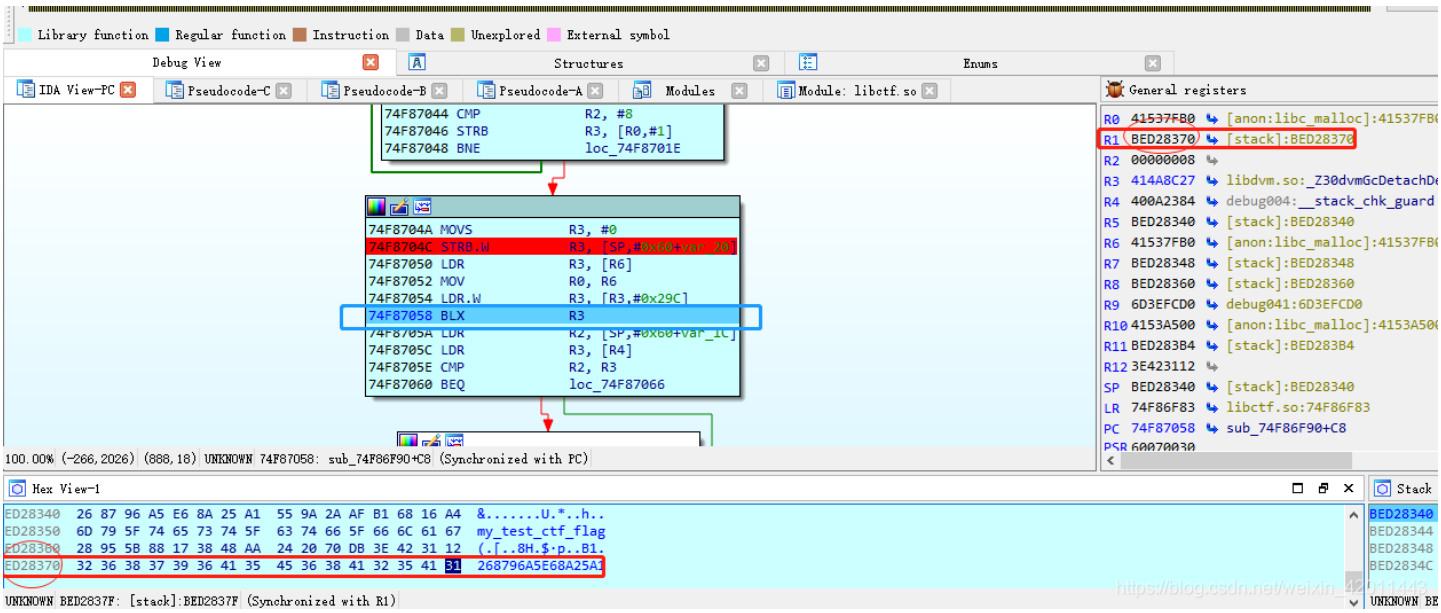
在这里进行下断点，点击“爬到了，看FLAG”按钮进行触发逻辑，并F8进行调式直到BLX为止（一般跳转指令都用来调用函数）：

```
74F8704A MOVS            R3, #0
74F8704C STRB.W          R3, [SP,#0x60+var_20]
74F87050 LDR             R3, [R6]
```

```
74F87052 MOV             R0, R6
74F87054 LDR.W           R3, [R3,#0x29C]
74F87058 BLX             R3
74F8705A LDR             R2, [SP,#0x60+var_1C]
74F8705C LDR             R3, [R4]
74F8705E CMP             R2, R3
74F87060 BEQ             loc_74F87066
```

BLX调用NewStringUTF函数返回result，查看此时R1寄存器的值（R1寄存器保存了NewStringUTF函数第二个参数的值，刚好是result，第一个参数为env）：

synchronize with R1看一下，发现：



返回的result值为26879...，刚好为FLAG的值。

(3)HOOK

这里使用Frida来hook，可以想到的方法有：

1）修改已爬楼层数为一个很大的值

```
import frida, sys

def on_message(message, data):
        if message['type'] == 'send':
                print("[*] {0}".format(message['payload']))
        else:
                print(message)


jscode = """
Java.perform(function () {
    var MainActivity = Java.use('com.ctf.test.ctf_100.MainActivity');
    //hook onClick方法，此处要注意的是onClick方法是传递了一个View参数v
    MainActivity.Btn_up_onclick.implementation = function (v) {
        send("Hook Start...");
        //调用onClick,模拟点击事件
        this.Btn_up_onclick(v);
        //修改参数
        this.has_gone_int.value = 1000000;
        send("Success!")
    }
});
"""

process = frida.get_usb_device().attach('com.ctf.test.ctf_100') #获取通信设备信息
script = process.create_script(jscode) #创建script实例，参数未定
script.on('message', on_message) #设置自定义回调函数
script.load() #在服务端启动js脚本
sys.stdin.read()
```

https://blog.csdn.net/weixin_42011443

2）修改要爬楼层数为很小的值

```
import frida, sys
```

```
def on_message(message, data):
        if message['type'] == 'send':
                print("[*] {0}".format(message['payload']))
        else:
                print(message)


jscode = """
Java.perform(function () {
    var MainActivity = Java.use('com.ctf.test.ctf_100.MainActivity');
    //hook onClick方法，此处要注意的是onClick方法是传递了一个View参数v
    MainActivity.Btn_up_onclick.implementation = function (v) {
        send("Hook Start...");
        //调用onClick,模拟点击事件
        this.Btn_up_onclick(v);
        //修改参数
        this.to_reach_int.value = 1;
        send("Success!")
    }
});
"""

process = frida.get_usb_device().attach('com.ctf.test.ctf_100') #获取通信设备信息
script = process.create_script(jscode) #创建script实例，参数未定
script.on('message', on_message) #设置自定义回调函数
script.load() #在服务端启动js脚本
sys.stdin.read()
```

https://blog.csdn.net/weixin_42011443

3）在点击事件中调用get_flag方法

```
import frida, sys

def on_message(message, data):
        if message['type'] == 'send':
                print("[*] {0}".format(message['payload']))
        else:
                print(message)


jscode = """
Java.perform(function () {
    var MainActivity = Java.use('com.ctf.test.ctf_100.MainActivity');
    //hook onClick方法，此处要注意的是onClick方法是传递了一个View参数v
    MainActivity.Btn_up_onclick.implementation = function (v) {
        send("Hook Start...");
        //调用onClick,模拟点击事件
        this.Btn_up_onclick(v);
        //修改参数
        this.to_reach_int.value = 1;
        var flag = this.get_flag(5);
        send("flag is " + flag);
    }
});
"""

process = frida.get_usb_device().attach('com.ctf.test.ctf_100') #获取通信设备信息
script = process.create_script(jscode) #创建script实例，参数未定
script.on('message', on_message) #设置自定义回调函数
script.load() #在服务端启动js脚本
sys.stdin.read()
```

https://blog.csdn.net/weixin_42011443

```
[*] Hook Start...KeyboardInterrupt

[*] flag is 268796A5E68A25A1
>>>
```