

ISC2016训练赛——phrackCTF CrazyAndroid Writeup

原创

Pyinal 于 2018-06-04 01:17:37 发布 2099 收藏 2

分类专栏: [CTF](#) 文章标签: [CTF Reverse](#) [CrazyAndroid](#) [ISC206](#)

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/Druke/article/details/80562202>

版权



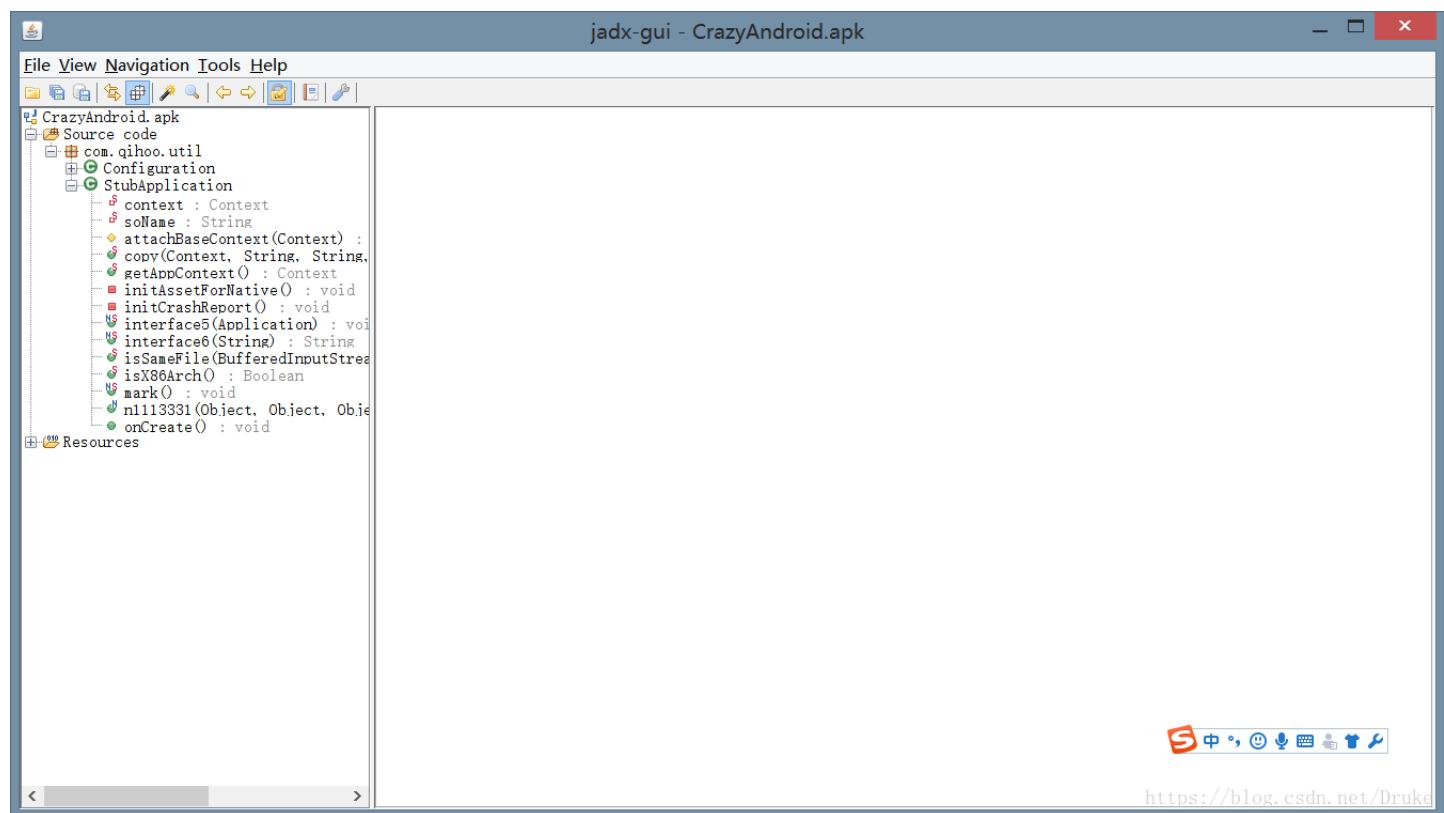
[CTF 专栏收录该内容](#)

2 篇文章 0 订阅

订阅专栏

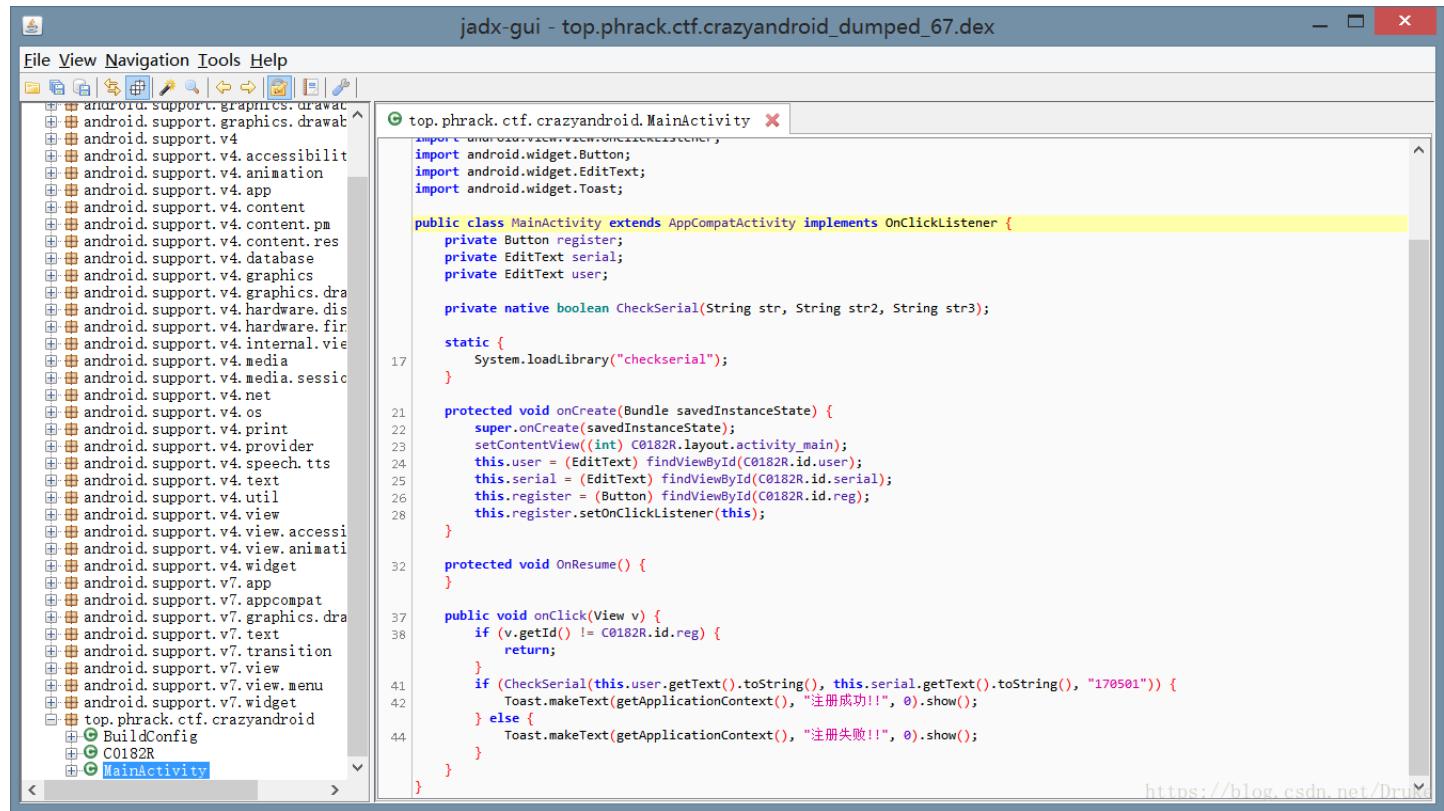
0x00 脱壳

jadx直接分析apk，发现是360的老壳。



脱壳工具推荐(drizzledump)[<https://github.com/DrizzleRisk/drizzleDumper>],(教程)[<http://www.freebuf.com/sectool/105147.html>]

脱壳后获得dex



The screenshot shows the jadx-gui interface with the file 'top.phrack.ctf.crazyandroid_dumped_67.dex' open. The left sidebar displays a tree view of the package structure, including 'BuildConfig', 'C0182R', and 'MainActivity'. The main window shows the Java code for 'MainActivity'.

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements OnClickListener {
    private Button register;
    private EditText serial;
    private EditText user;

    private native boolean CheckSerial(String str, String str2, String str3);

    static {
        System.loadLibrary("checkserial");
    }

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) C0182R.layout.activity_main);
        this.user = (EditText) findViewById(C0182R.id.user);
        this.serial = (EditText) findViewById(C0182R.id.serial);
        this.register = (Button) findViewById(C0182R.id.reg);
        this.register.setOnClickListener(this);
    }

    protected void OnResume() {
    }

    public void onClick(View v) {
        if (v.getId() != C0182R.id.reg) {
            return;
        }
        if (CheckSerial(this.user.getText().toString(), this.serial.getText().toString(), "170501")) {
            Toast.makeText(getApplicationContext(), "注册成功!!", 0).show();
        } else {
            Toast.makeText(getApplicationContext(), "注册失败!!", 0).show();
        }
    }
}
```

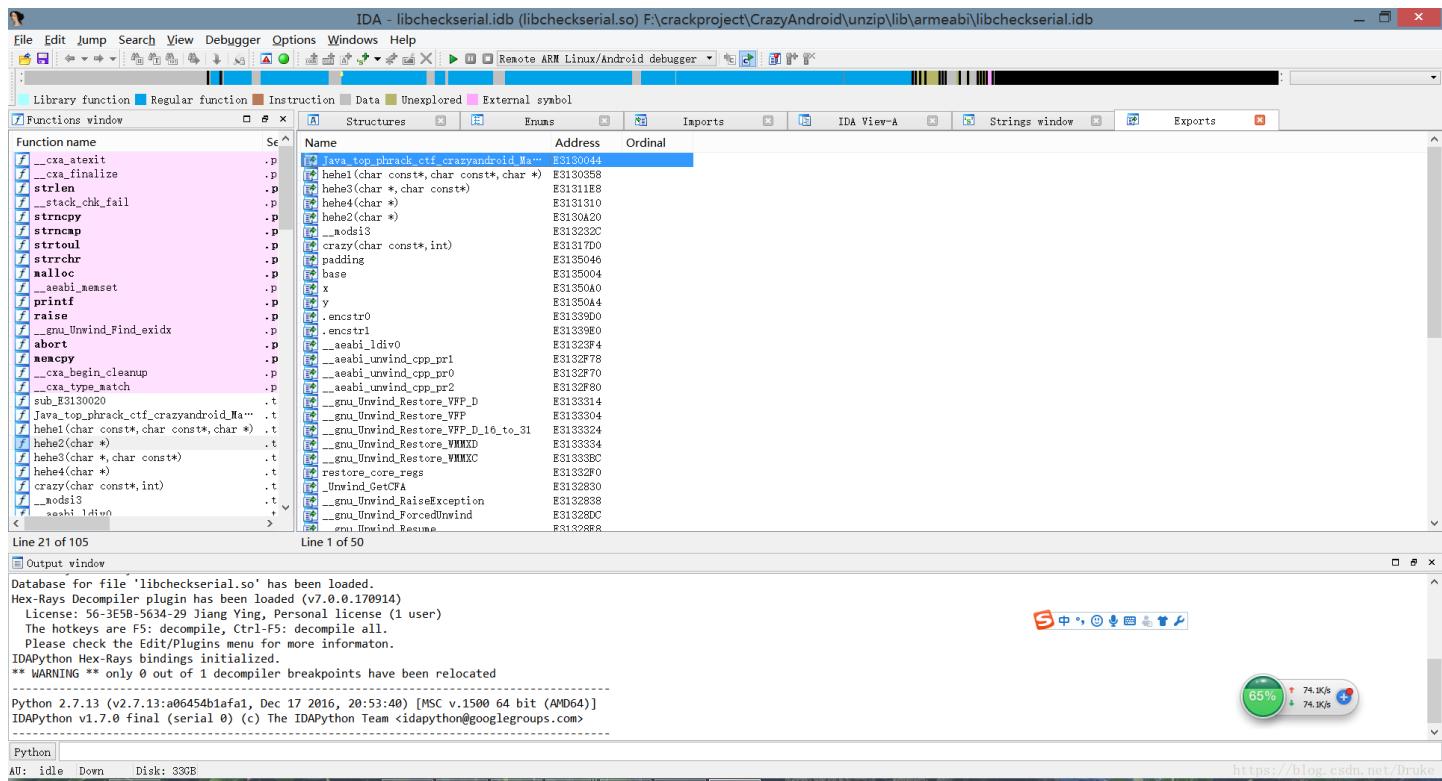
https://blog.csdn.net/Dru...

看到这里其实都是老套路了，解密的逻辑都写在native层，不会写在java层

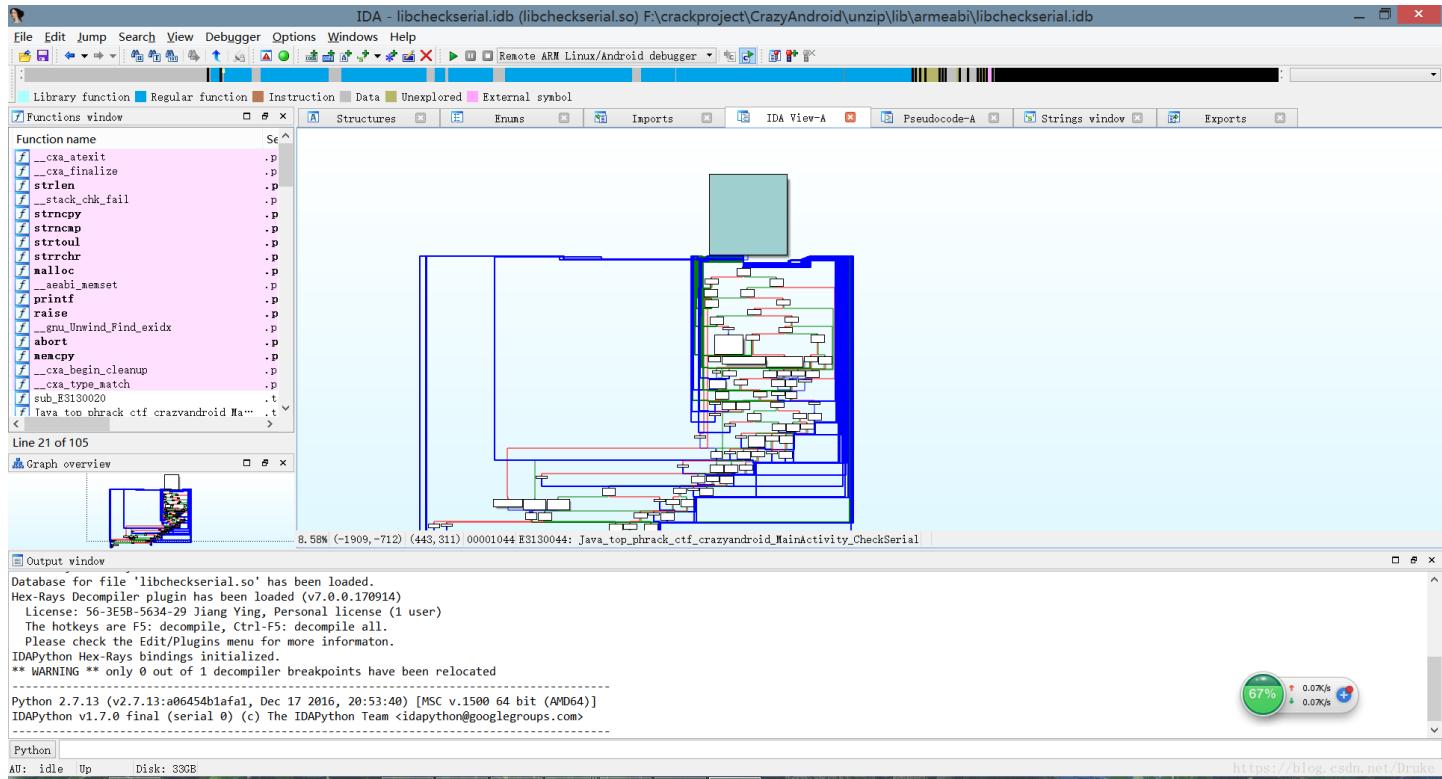
所以脱不脱壳不其实重要，直接分析so就好了（逃）。不过脱壳的话就有个好处，壳会在启动时如果检测到IDA的Android Service 是已经启动的话会直接结束进程，去掉壳可以方便调试。当然，也可以不脱壳，先启动完应用再用IDA附加也可以绕过检测。

0x01 libcheckserial 分析

IDA打开libcheckserial.so,主要注意导出表中的前五个函数。crazy在hehe2和hehe4中存在引用



先分析Java_top_phrack_ctf_crazyandroid_MainActivity_CheckSerial



看到一堆花里胡哨的东西不要方，先f5冷静分析一波

CheckSerial的跳转流程虽然十分的混乱，但是其实也就调用了4个hehe函数。为确定运行流程，可以在四个hehe函数调用各下一个断点后

随意输入一个注册码，IDA动态调试，会发现hehe1返回0后函数就结束运行，并不会调用后面三个函数，手动修改R0的值（返回值）为1，才会触发后续函数的调用，连续修改4个返回值后便会可以注册成功。

所以只需确保四个hehe函数的返回值为1便可注册成功。

0x02 hehe1

hehe1主要校验输入的长度是否为40。并对输入字符串进行转换（一下讨论均针对已转换后的字符串，暂且称为enstr）,输入输出对照如下

```
input = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-+'  
output = 'nopqrstuvwxyzabcdefghijklmnopqrstuvwxyzNOPQRSTUVWXYZABCDEFHIJKLMNOPQRSTUVWXYZ012345678901234-+'
```

最后校验estr前六位是否为pctfef（用户为pctf时）

0x03 hehe2

hehe2 流程略显复杂，但分析主要是体力活，主要校验enstr的第7位和第24位是否为'-'。至于7-24之间的16位先经过部分换位再经运算后生成字符串需等于'Pctf2016'（crazy函数生成，crazy其实是个骗子，里面虽然很复杂但是只要看结果就够了）,运算方法如下（以两位为一组,共八组,生成8个字符）:

```
#部分换位逆方法  
keys = list(keys)  
if len(keys) != 16:  
    print('[*]warning bad keys input')  
i = keys.pop(6)  
keys.insert(5, i)  
i = keys.pop(9)  
keys.insert(11, i)  
i = keys.pop(13)  
keys.insert(12, i)  
i = keys.pop(15)  
keys.insert(14, i)  
keys = ''.join(keys)
```

```
#运算  
result1 = ord(p1) << 4  
result2 = 0xD0 - (~ord(p2) + 1)  
result = (result2 ^ result1) | (result2 & result1)  
result = chr(hex(result & 0xFF))
```

0x04 hehe3

hehe3其实已经放弃抵抗了，仅是简单校验enstr的25-30是否为170501（也就是Java层输入的第三个参数）。

0x05 hehe4

最后一步主要校验enstr的最后10位，通过crazy(padding,88)生成一个64字符串与estr拼接，通过运算生成最后10位校验数,运算如下

```
padding = '075e191fe314c1e7917d9c71f7b6ed9842090f28f649bad384d0880d103b99a8'  
encpadding = enstr + padding  
initl = 0  
for i in encpadding:  
    result = ord(i) - (~(0x28 * initl) + 1)  
    initl = result & 0xFFFFFFFF  
print(initl)  
#initl为生成的最后十位
```

0x06 python代码

附赠一份生成注册码的python代码，变量命名有些随意大家将就看吧（逃）

```
from random import choice
import itertools

enc = 'nopqrstuvwxyzabcdefghijklmNOPQRSTUVWXYZABCDEFGHIJKLM5678901234-+'
raw = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-+'

source = '0123456789-+'
keyw = "Pctf2016"
keyhex = [hex(ord(i)) for i in keyw]
conta = []
key = ''
user = 'pctf'
a2 = '170501'

# 双字符转密钥
def getResult(p1, p2):
    result1 = ord(p1) << 4
    result2 = 0xD0 - (~ord(p2) + 1)
    result = (result2 ^ result1) | (result2 & result1)
    return hex(result & 0xFF)

#hehe2
def keyReverse(keys):
    keys = list(keys)
    if len(keys) != 16:
        print('[*]warning bad keys input')

    i = keys.pop(6)
    keys.insert(5, i)
    i = keys.pop(9)
    keys.insert(11, i)
    i = keys.pop(13)
    keys.insert(12, i)
    i = keys.pop(15)
    keys.insert(14, i)

    keys = ''.join(keys)
    return keys

def hehe4(enstr):
    if len(enstr) != 30:
        print('[*]warning bad input enc ')
    padding = '075e191fe314c1e7917d9c71f7b6ed9842090f28f649bad384d0880d103b99a8'
    encpadding = enstr + padding
    initl = 0
    for i in encpadding:
        result = ord(i) - (~(0x28 * initl) + 1)
        initl = result & 0xFFFFFFFF
    enstr += str(initl)
    return enstr

# 生成映射表
reflect = {}
```

```
for index, i in enumerate(enc):
    reflect[i] = raw[index]

# 生成元素组合
for index, k in enumerate(keyhex):
    temp = []
    # print(k)
    for i in source:
        for g in source:
            if getResult(i, g) == k:
                temp.append(i + g)
                # print(getResult(i, g))
                # print(i + g)
    conta.append(temp)
    # print('-----')

# 随机生成key
for i in conta:
    key += choice(i)

# key位置交换
key = keyReverse(key)

# key拼接
enc = user + 'ef' + '-' + key + '-' + a2

# 后10位
enc = hehe4(enc)

# key解密
dec = ''
for i in enc:
    dec += reflect[i]
print('congratulation! user {}'s key is {}'.format(user, dec))
```

writeup写的时候思维有些混乱，难免有什么地方出错，欢迎指出、py（大雾）.....