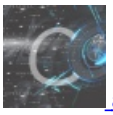
看雪ctf2017 |WP Ericky.apk

原创

坚强的女程序员 于 2018-04-07 18:33:15 发布 520 收藏

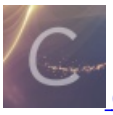
分类专栏： android CTF 文章标签： 算法 wp

本文链接：https://blog.csdn.net/qq_33438733/article/details/79843768


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

参考文章

https://blog.csdn.net/wmh_100200/article/details/73368859

https://bbs.pediy.com/thread-218455.htm

前言

看雪的ctf还是很有水平的，我这菜鸡表示只能看看

分析

用jeb打开，发现做了很恶心的混淆，汗颜。使用n键对变量重命名。

```
    protected void onCreate(Bundle arg3) {
        super.onCreate(arg3);
        this.setContentView(2130968603);
        this.button = this.findViewById(2131427415);
        this.edit_text = this.findViewById(2131427416);
        this.button.setOnClickListener(new View$OnClickListener() {
            public void onClick(View arg2) {
                this.[illegible].cilcked();
            }
        });
    }

    public void cilcked() {
        String v0 = this.edit_text.getText().toString().trim();
        StringBuilder v1 = new StringBuilder();
        v1.append(v0);
        if(utils.check(v1.toString().trim())) {
            Toast.makeText(((Context)this), MainActivity$1$utils.dbcb("       [illegible]"), 0).show();
        }
        else {
            Toast.makeText(((Context)this), MainActivity$1$utils.dbcb("       [illegible]"), 0).show();
        }
    }
}
```


java层的分析我就不写了，这题的考点在于so层算法的分析以及指令的混淆。

ps:我几乎用了一天的时间，去看这个check函数

去除指令混淆

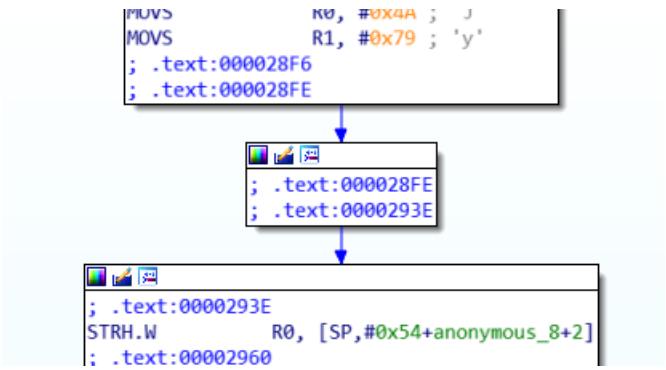
要么手动的patch要么写个脚本去除，我看还是写个脚本吧，我刚开始手动patch，实在是太累了！！！

```
#include <idc.idc>

static main()
{
    auto i,pos,size,JMP_SIZE,FLOWER1_SIZE,FLOWER2_SIZE;
    pos=0x286C; //START
    size=0x1A000;//SIZE
    JMP_SIZE = 0x40;
    FLOWER1_SIZE = 0x1e;
    FLOWER2_SIZE = 0x8;


    for ( i=0; i < size;i++ ) {
        //PATCH JMPS
        if (
            (Byte(pos)==0x13)&&(Byte(pos+1)==0xe0)&&(Byte(pos+2)==0xbd)&&
            (Byte(pos+3)==0xe8)&& (Byte(pos+4)==0xf0)&&(Byte(pos+5)==0x47))
        {
            for(i=0;i<JMP_SIZE;i++)
            {
                PatchByte(pos+i,0x0);//change
            }
            HideArea(pos,pos+JMP_SIZE,atoa(pos),atoa(pos),atoa(pos+JMP_SIZE),-1);
            continue;
        }

        // PATCH FLOWER1
        //.text:00002A80 B1 B5                    PUSH            {R0,R4,R5,R7,LR}
        //.text:00002A82 82 B0                    SUB             SP, SP, #8
```

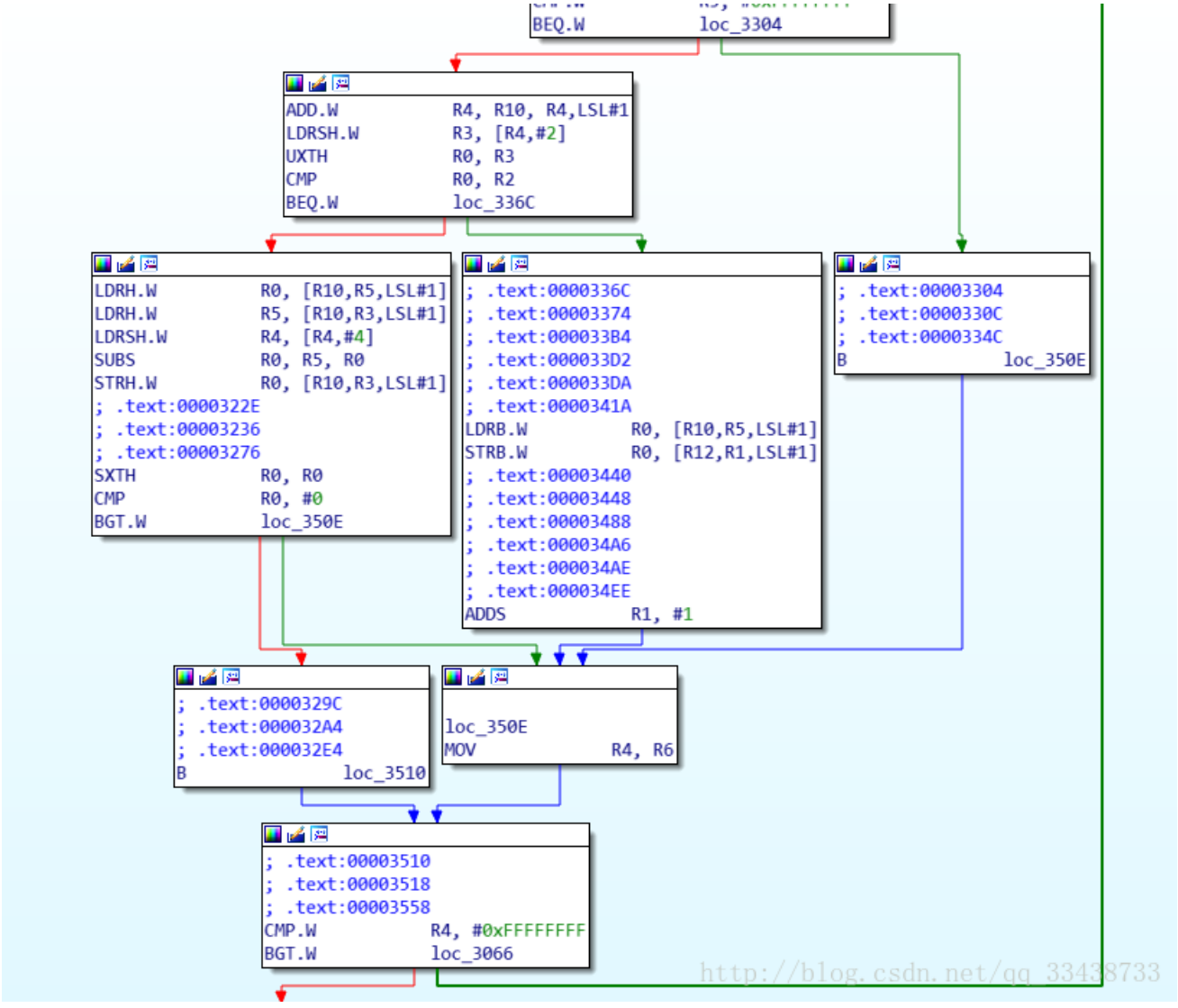
```
//.text:00002A84 12 46                    MOV             R2, R2
//.text:00002A86 02 B0                    ADD             SP, SP, #8
//.text:00002A88 00 F1 01 00              ADD.W           R0, R0, #1
//.text:00002A8C A0 F1 01 00              SUB.W           R0, R0, #1
//.text:00002A90 1B 46                    MOV             R3, R3
//.text:00002A92 BD E8 B1 40              POP.W           {R0,R4,R5,R7,LR}
//.text:00002A96 01 F1 01 01              ADD.W           R1, R1, #1
//.text:00002A9A A1 F1 01 01              SUB.W           R1, R1, #1

    if (
        (Byte(pos)==0xb1)&&(Byte(pos+1)==0xb5)&&(Byte(pos+2)==0x82)&&(Byte(pos+3)==0xb0)&&
        (Byte(pos+0x1a)==0xa1)&&(Byte(pos+0x1b)==0xf1)&&(Byte(pos+0x1c)==0x01)&&(Byte(pos+0x1d)==0x
    {
        for(i=0;i<FLOWER1_SIZE;i++)
        {
            PatchByte(pos+i,0x0);
        }
        HideArea(pos,pos+FLOWER1_SIZE,atoa(pos),atoa(pos),atoa(pos+FLOWER1_SIZE),-1);
        continue;
    }

    //PATCH FLOWER2
    // "PUSH.W {R4-R10,LR}"
    // "POP.W {R4-R10,LR}"

    if (
        (Byte(pos)==0x2d)&&(Byte(pos+1)==0xe9)&&(Byte(pos+2)==0xf0)&&(Byte(pos+3)==0x47)&&
        (Byte(pos+4)==0xbd)&&(Byte(pos+5)==0xe8)&&(Byte(pos+6)==0xf0)&&(Byte(pos+7)==0x47))
    {
        for(i=0;i<FLOWER2_SIZE;i++)
        {
            PatchByte(pos +i,0x0);
        }
        HideArea(pos,pos+FLOWER2_SIZE,atoa(pos),atoa(pos),atoa(pos+FLOWER2_SIZE),-1);
        continue;
    }
    pos++;
}

Message("\n" + "DE-FLOWERS FINISH BY Ericky\n");
}
```

重新create function

手动的删除check和jni_load中间的函数，然后重新create_function，之后便可以使用F5反汇编了。虽然仍存在很多混淆指令，但这样已经可以看到check的整个代码了。
可以使用流程图，看整个函数的执行过程



```
; .text:00002968
; .text:000029A8
STRH.W          R1, [SP,#0x54+var_30]
MOVS            R1, #0x75 ; 'u'
; .text:000029CC
; .text:000029D4
; .text:00002A14
STRH.W          R1, [SP,#0x54+var_30+2]
MOVS            R1, #0x33 ; '3'
; .text:00002A38
; .text:00002A40
; .text:00002A80
STRH.W          R1, [SP,#0x54+var_2C]
MOVS            R1, #0x43 ; 'C'
; .text:00002AA4
; .text:00002AAC
; .text:00002AEC
STRH.W          R1, [SP,#0x54+var_2C+2]
; .text:00002B0E
; .text:00002B16
; .text:00002B56
MOVS            R1, #0
STRH.W          R0, [SP,#0x54+var_28]
MOVS            R0, #0x6C ; 'l'
; .text:00002B7C
; .text:00002B84
; .text:00002BC4
STRH.W          R0, [SP,#0x54+var_28+2]
MOVS            R0, #0x56 ; 'V'
; .text:00002BE8
```


```
; .text:00003586
```

```
; .text:000035C6
; .text:000035E4
; .text:000035EC
; .text:0000362C
BL              sub_19FC
; .text:0000364E
; .text:00003656
; .text:00003696
LDR.W           R0, [R9] ; 到此循环结束
MOV             R1, R8
MOVS            R2, #0
MOVS            R4, #0
LDR.W           R3, [R0,#0x2A4]
MOV             R0, R9
BLX             R3
; .text:000036C6
; .text:000036CE
; .text:0000370E
; .text:0000372C
; .text:00003734
; .text:00003774
BL              sub_19DA8 ; 这里应该是一个重要的加密函数
MOV             R1, R0
LDR.W           R0, =0x1C74E
; .text:0000379C
; .text:000037A4
; .text:000037E4
; .text:00003802
; .text:0000380A
; .text:0000384A
; .text:00003868
; .text:00003870
```



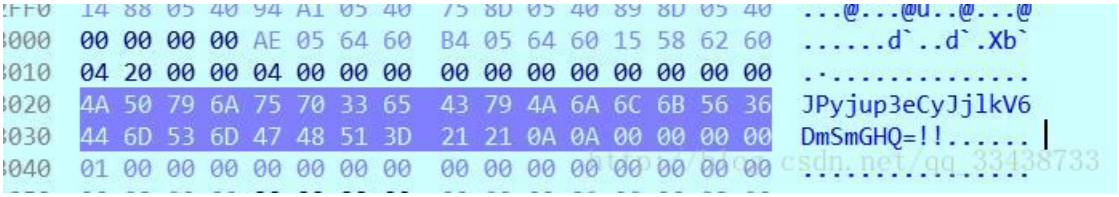
```
1  }
2  while ( v6 > -1 );
3  sub_19FC();
4  v13 = 0;
5  ((void (__fastcall *)(JNIEnv *, int, _DWORD))(*v3)->GetStringUTFChars)(v3, v4, 0);
6  sub_19DA8();
7  while ( *(unsigned __int8 *)(v13 + 131104) == *(unsigned __int8 *)(v14 + v13) )
8  {
9    if ( ++v13 == 24 )
0      return 1;
1  }
2  sub_27C8(byte_20020);
3  return 0;
4}
```

比较函数

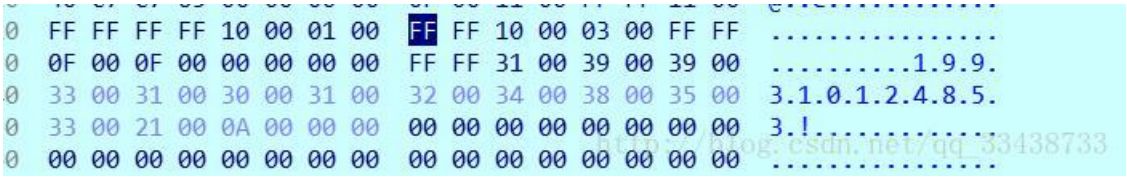
接下来就是需要动态调试去跟了。

动态调试

由于我们修改了so文件，因此在加载的时候要注意。使ida加载本地的so，而不是手机中的so（只需要默认选择就好）



经过初始化后可以看到该值，这一串字符就是最终我们需要匹配的。也就是最后的密文。




```
19931012应该就是加密的密匙
```

最后分析算法(可能也是猜测吧，分析是不可能的，永远也不可能的！)，可以知道为rc4算法，找一个在线rc4加解密的网站。最后可以解得密码`madebyericky94528`

总结

可以说这题折腾了很久，但是最终还是没有整出来，一方面是对指令混淆不熟，说白了也就是对arm指令不熟，另一方面就是对各种算法的反汇编不熟悉，真的很难。而且这两天我心情也不好，很难静下心来仔细的分析。这篇文章匆匆写下，其实那个rc4算法我还是没有看懂，而且密匙为什么是199931012而不是199310124853呢？问题还有很多，我不想折腾了。算法看重的不是加密的过程，而是注意输入和输出，抓住这两点，那么即使这个算法你识别不出来，也是可以看懂的。
这两天心情真的是糟透了，每天都活的很累，好好的假期，每天却如同末日一般。我的心悬在高处，只能用枯燥的指令占据我的逻辑。
程序员是孤独的，因为很少有人能懂他在干什么。不奢求别人能懂，只希望在放松的时候能有人嘘寒问暖 –2018.4.7