

【bugkuCTF】 LoopAndLoop(阿里CTF)writeup

原创

iqiqiya 于 2018-05-22 20:41:41 发布 2656 收藏 1
分类专栏: -----bugkuCTF 我的CTF之路 我的CTF进阶之路 文章标签: LoopAndLoop(阿里CTF)writeup LoopAndLoop bugkuCTF Reverse

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/xiangshangbashaonian/article/details/80411445>

版权



[-----bugkuCTF 同时被 3 个专栏收录](#)

9 篇文章 0 订阅

订阅专栏



[我的CTF之路](#)

92 篇文章 5 订阅

订阅专栏

[我的CTF进阶之路](#)

108 篇文章 18 订阅

订阅专栏

下载链接:

https://pan.baidu.com/s/1uvsl_xusNMMnnrjOzuoffg 密码: me4j

LoopAndLoop(阿里CTF) 100

flag格式
alictf{ }

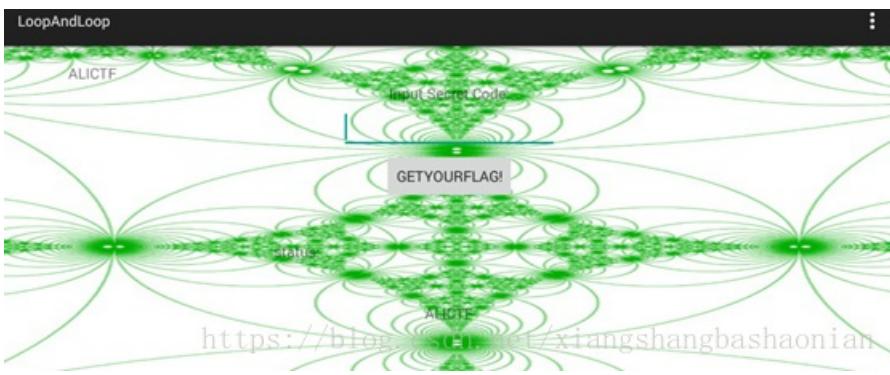
下载链接: https://pan.baidu.com/s/1uvsl_xusNMMnnrjOzuoffg 密码:
me4j

Flag

Submit

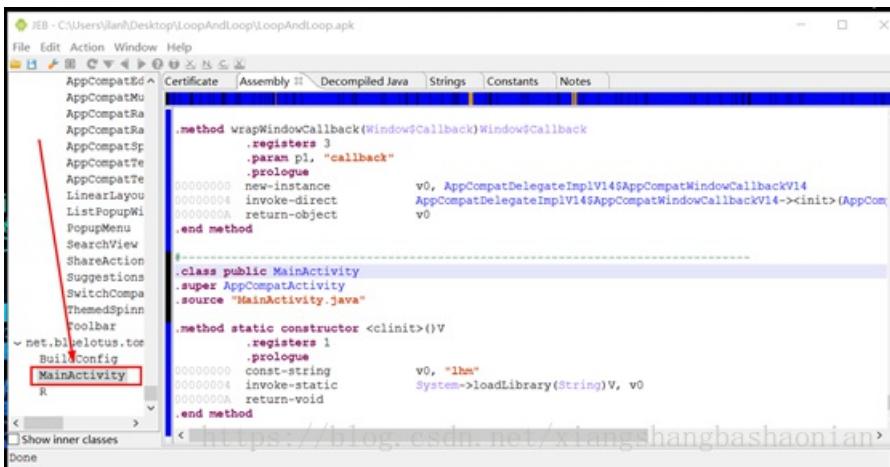
<https://blog.csdn.net/xiangshangbashaonian>

先解压安装下看看什么情况

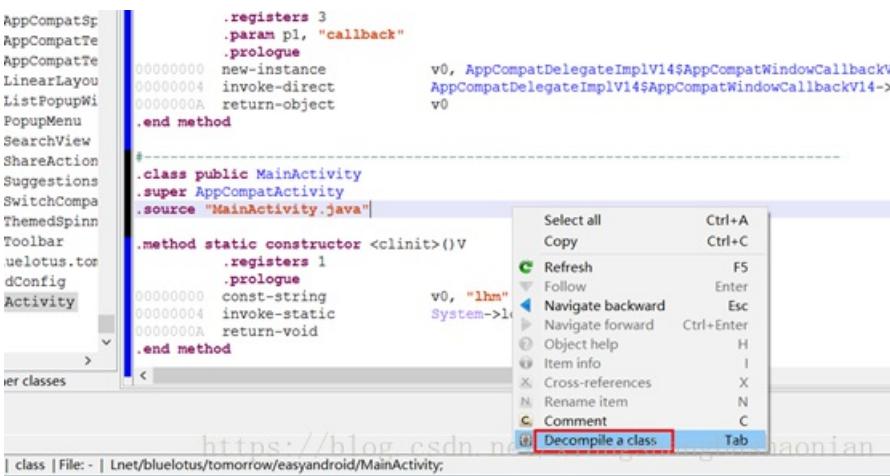


接着直接载入JEB

双击>MainActivity



右击空白处选择Decompile a class



```
package net.bluelotus.tomorrow.easyandroid;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
```

```
public class MainActivity extends AppCompatActivity { // System.loadLibrary()在我们使用Java的JNI机制时，会片
    static {
        System.loadLibrary("lhm"); // 它的作用即是把我们在Java code中声明的native方法的那个libraryload进来，或者
    }

    public MainActivity() {
        super();
    }

    public native int chec(int arg1, int arg2) { // native层的chec方法
    }

    public int check(int input, int s) { // check方法将我们的输入和一个int型变量s返回到chec
        return this.chec(input, s);
    }

    public int check1(int input, int s) { // check1定义v1为我们的输入，v0为循环变量1
        int v1 = input;
        int v0 = 1;
        label_2: // 进入到label_2
        if(v0 < 100) {
            v1 += v0; // 先判断v0是否小于100，如果成立，那么v1每次加上v0的值
            ++v0; // v0每次也要自增1
            goto label_2; // 直接走向label_2
        }

        return this.chec(v1, s); // 还是将得到的v1及s返回给chec方法
    }

    public int check2(int input, int s) { // 和上面的check与check1一样，都是需要两个参数，一个是我们的输入，一个是
        int v2; // v2还是作为chec方法的返回值
        int v3 = 1000;
        int v1 = input; // 定义三个int型变量，v1还是我们的输入，v3做为定值1000，而v2作为返回值(相当于一个标志)
        if(s % 2 == 0) {
            int v0; // 当s对2取余等于0，也就是说s能被2整除时，定义一个新的循环变量v0
            for(v0 = 1; v0 < v3; ++v0) {
                v1 += v0; // v0等于1，当小于v3的值也就是小于1000时，v1每次加上v0，v0++
            }

            v2 = this.chec(v1, s); // v2还是作为chec方法的返回值
        }
        else {
            for(v0 = 1; v0 < v3; ++v0) { // 如果取余不等于0，那么还是和上边一样，只不过v1每次减去v0
                v1 -= v0;
            }

            v2 = this.chec(v1, s); // chec方法的返回值
        }
    }

    return v2; // 返回v2
}

public int check3(int input, int s) { // check3将输入给v1，将循环变量v0初值设为1，当v0小于10000，v1每次加上v0，不
    int v1 = input;
    int v0;
    for(v0 = 1; v0 < 10000; ++v0) {
        v1 += v0;
    }

    return this.chec(v1, s);
}
```

```

}

public String messageMe(String text) { //messageMe方法是返回字符串"LoopOk"+text
    return "LoopOk" + text;
}

protected void onCreate(Bundle savedInstanceState) { //关键的方法onCreate
    super.onCreate(savedInstanceState);
    this.setContentView(2130968600); //这两行和布局有关，不用管
    this.findViewById(2131492946).setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) { //最最重要的，从名字就可以看出，当我们点击GETYOURFLAG！这个按钮时触发的onClick方法
            int v1;
            String v2 = this.findViewById(R.id.editText).getText().toString(); //v2获取我们的输入并转成字符串
            try { //下边这个try,catch用来捕获将类型异常
                v1 = Integer.parseInt(v2); //将v2这个String字符类型数据转换为Integer整型数据赋值给v1
            }
            catch(NumberFormatException v0) {
                this.findViewById(R.id.textView1).setText("Not a Valid Integer number"); //如果不可以转成整数，就在屏幕打印"不是一
                return;
            }

            if(MainActivity.this.check(v1, 99) == 1835996258) { //如果我们输入的v1和s也就是99传给check方法，返回值等于1835996258，就输出flag
                this.findViewById(R.id.textView1).setText("The flag is:"); //接着传向check方法得到的返回值等于1835996258，就输出flag
                this.findViewById(R.id.textView2).setText("alictf{" + MainActivity.this.stringFromJNI2(v1) + "}"); //括号内是native层的函数
            }
            else {
                this.findViewById(R.id.textView1).setText("Not Right!");
            }
        }
    });
}

public boolean onCreateOptionsMenu(Menu menu) {
    this.getMenuInflater().inflate(2131558400, menu);
    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    boolean v1 = item.getItemId() == 2131492961 ? true : super.onOptionsItemSelected(item);
    return v1;
}

public native String stringFromJNI2(int arg1);
}
}

```

经过分析可知 重要的chec和stringFromJNI2都在native层

那么就需要将liblhsm.so文件载入IDA进行分析啦

LoopAndLoop > LoopAndLoop > lib > armeabi				
名称	修改日期	类型	大小	
liblhsm.so	2016/5/17 17:48	SO 文件	1	

<https://blog.csdn.net/xiangshangbashaonian>

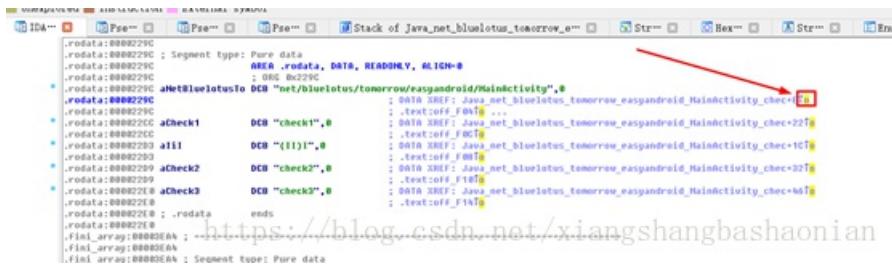
载入后直接shift+F12搜索字符串

双击MainActivity进入

Address	Length	Type	String
.rodata:000002...	00000030	C	net/bluelotus/tomorrow/easyandroid/MainActivity
.rodata:000002...	00000007	C	check3
.rodata:000002...	00000007	C	check2
.rodata:000002...	00000007	C	check1
.rodata:000002...	00000006	C	(ID)

<https://blog.csdn.net/xiangshangbashaonian>

接着双击“蓝色向上的小箭头” 找到引用



在按F5就可以将chec反汇编成伪代码了

```
1 int __fastcall Java_net_bluelotus_tomorrow_easyandroid_MainActivity_chec(int a1, int a2, int a3, int a4)
2 {
3     int v9; // r4@1
4     int v8; // r7@1
5     int result; // r0@2
6     int v7; // [sp+Ch] [bp-3h]@1
7     int v8; // [sp+18h] [bp-2h]@1
8     int v9; // [sp+1h] [bp-2Ch]@1
9     int v10; // [sp+1Ch] [bp-2h]@1
10    int v11; // [sp+2h] [bp-2h]@1
11    int v12; // [sp+24h] [bp-1Ch]@1
12
13    v9 = a2;
14    v8 = a4;
15    v9 = a1;
16    v7 = a3;
17    v5 = (*((int **)(void))(*(_DWORD *)v9 + 28))();
18    v10 = _JNINvoc::GetMethodID(v9, v5, "check1", "(I)I");
19    v11 = _JNINvoc::GetMethodID(v9, v5, "check2", "(I)I");
20    v12 = _JNINvoc::GetMethodID(v9, v5, "check3", "(I)I");
21    if ( v8 - 1 < 0 )
22        result = v7;
23    else
24        result = _JNINvoc::CallIntMethod(v9, v9, *(&v10 + 2 * v8 % 3));
25    return result;
26}
```

<https://blog.csdn.net/xiangshangbashaonian>

经过分析 可以知道chec方法根据第二个参数乘2对3取模的结果调用Java层的三个check函数对我们的输入进行处理

所以我们只需要写脚本将算法逆过来就好

```

#!/usr/bin/env python
#-*- coding: utf-8 -*-
def getinput():
    target = 1835996258
    for i in range(2,100):
        if 2 * i % 3 == 0:
            target = check1(target,i - 1)
        elif 2 * i % 3 == 1:
            target = check2(target,i - 1)
        else:
            target = check3(target,i - 1)
    print target
def check1(input,loopNum):
    t = input
    for i in range(1,100):
        t = t - i
    return t

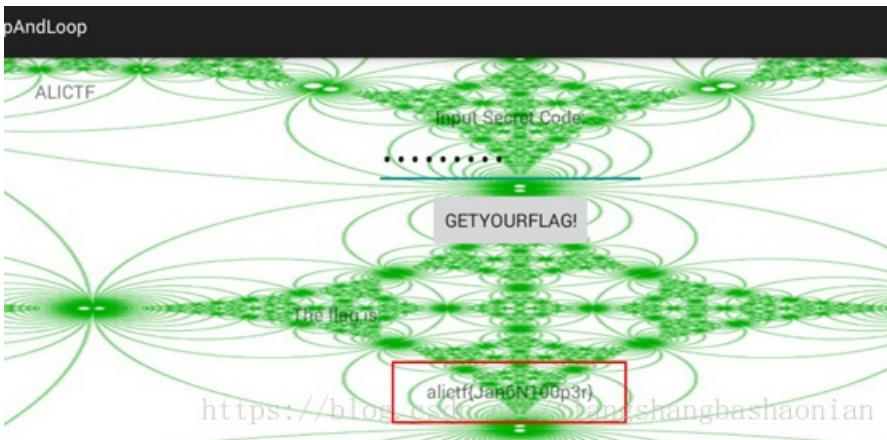
def check3(input,loopNum):
    t = input
    for i in range(1,10000):
        t = t - i
    return t

def check2(input, loopNum):
    t = input
    if loopNum % 2 == 0:
        for i in range(1,1000):
            t -= i
        return t
    for i in range(1,1000):
        t += i
    return t

if __name__ == '__main__':
    getinput()

```

成功截图



参考资料: https://blog.csdn.net/qq_29343201/article/details/51607527?locationNum=12&fps=1

<https://blog.csdn.net/jasalee/article/details/70342487>