WriteUp – 2018年“华为杯”极客出发XMan冬令营线上CTF选拔赛

转载

weixin_34122810 于 2018-02-05 01:22:13 发布 298 收藏
文章标签： java 开发工具
原文链接：https://segmentfault.com/a/1190000013144470

Mobile

第二题：返老还童（1000pt）

- 拿到apk之后丢进JEB，发现只有.class，java代码，没有so。整个apk的class文件结构如下：

```
a.a.a
com.reverse.daydayup.a
com.reverse.daydayup.b
com.reverse.daydayup.MainActivity
```

- 阅读MainActivity，发现页面下方是一个WebView，用于展示结果的区域，MainActivity中有关于加载Google WebView的代码，文件中还有一个check_flag.html，打开看好多JS，一脸懵逼，不过目标还是转向java代码。
- 定位到a.a.a，发现两个函数a和b，分析之发现第一个函数a要求了flag格式为xxxx-xxxx-xxxx-xxxx，也就是以“-”进行split之后必须是四个字串，同时每个字串长度都为4，字串中所有字符都必须是字母和数字。经过我简单修改替换后的代码如下（修改函数返回值为boolean，便于后续破解）：

```
public static boolean a(String paraString) {
//        int v6 = 4;
        Check.a = 0;
        Check.b = 0;
//        MainActivity.b.getSettings().
//setUserAgentString(MainActivity.b.getSettings().
//getUserAgentString() + ";" + Check.a + ";" + Check.b);
//        以“-”分割字符串
        String[] subString = paraString.split("-");
//        分割后必须产生4个子串：即格式为xxxx-xxxx-xxxx-xxxx
        if(subString.length == 4) {
            int i = 0;
            while(i < subString.length) {
//                检查四个字串，要求长度也为4，并且只能为字母或数字
                if(subString[i].length() == 4) {
                    int k;
                    for(k = 0; k < subString[i].length(); ++k) {

                        if( !(subString[i].charAt(k) >= '0' && subString[i].charAt(k) <= '9')
                        && !(subString[i].charAt(k) >= 'A' && subString[i].charAt(k) <= 'Z')
                        && !(subString[i].charAt(k) >= 'a' && subString[i].charAt(k) <= 'z')) {
                            return false;
                        }
                    }
                    ++i;
                }
                else {
                    return false;
                }
            }
//            满足条件，进入b方法
            return Check.b(paraString);
        }
        return false;
    }
```

- 以上条件满足后进入b函数，b函数有很多递推关系，具体关系如代码注释所示，同样修改函数返回值为boolean：

```
public static boolean b(String paraString) {
//        int v8 = 3;
//        int v6 = 2;
        String[] subString = paraString.split("-");
        char[] subString0 = subString[0].toCharArray();
        char[] subString1 = subString[1].toCharArray();
        char[] subString2 = subString[2].toCharArray();
        char[] subString3 = subString[3].toCharArray();
//         (a, b) 意思是第a部分的第b个字符

//         (1, 1) == (4, 2) -3
//         (1, 4) == (4, 3) | 1（最低位变为1）
//         (4, 3) 的ASCII是奇数
        if(subString0[0] == subString3[1] - 3 && subString0[3] == (((char)(subString3[2] | 1))) && subStrin
//            a =  (4, 4) ^ 55
            Check.a = subString3[3] ^ 55;
//             (4, 3) == (1, 3) + 8
//             (4, 1) == (1, 1) - 2
//             (4, 2) == (1, 4) ^ 18
//             (1, 2) * 2 == (1, 3) -8
//             (4, 4) == (1, 3)
            if(subString3[2] == subString0[2] + subString0.length * 2
            && subString3[0] == subString0[0] - subString0.length / 2
            && subString3[1] == (((char)(subString0[3] ^ 18)))
            && subString0[1] * 2 == subString0[2] - 8
            && subString3[3] == subString0[2]) {

//                 (2, 2) +12== (2, 4)
//                 (3, 2) * 2 ==  (2, 4) - 11
                if(subString.length * 3 + subString1[1] == subString1[3]
                && subString2[1] * 2 == subString1[3] - 11) {
//                     b =  (2, 3) - 117
                    Check.b = subString1[2] - subString2.length ^ 113;

//                     (2, 1) + (3, 1) == 187
//                     (2, 1) + (3, 4) == 210
//                     (2, 4) ^ (2, 3) == 47
//                     (2, 1) ^ (2, 2) == 15
//                     (3, 3) ^ (2, 2) == 5
                    if(subString1[0] + subString2[0] == 187 &&
                       subString1[0] + subString2[3] == 210 &&
                       (subString1[3] ^ subString1[2]) == 47 &&
                       (subString1[0] ^ subString1[1]) == 15 &&
                       (subString2[2] ^ subString1[1]) == 5 &&
//                         (4, 4) ^ 55 > (2, 3) - 117
//                         (4, 4) ^ 55 < 100
                       Check.a > Check.b && Check.a < 100) {
                        return true;
//                         MainActivity.b.getSettings()
//.setUserAgentString(MainActivity.b.getSettings()
//.getUserAgentString() + ";" + Check.a + ";" + Check.b);
                    }
                }
            }
        }
        return false;
    }
```

- 分析所有限制条件发现，只需要遍历（2，1）和（4，3）的所有可能（ASCII码范围从字符0到字符z）再推算出其他字符，满足条件的就是答案，将以上代码放入Eclipse，编写解题代码如下：

```
/**
     * 主函数 ShellCode
     * @param args
     */
    public static void main(String[] args) {
        for(char c21='0';c21<='z';c21++) {
            for(char c43='0';c43<='z';c43++) {
                StringBuilder result = new StringBuilder("");
                char c13=(char) (c43-8);
                char c12=(char) ((c13-8)/2);
                char c14=(char) (c43 | 1);
                char c22=(char) (c21^15);
                char c24=(char) (c22+12);
                char c23=(char) (c24^47);
                char c31=(char) (187-c21);
                char c32=(char) ((c24-11)/2);
                char c33=(char) (c22^5);
                char c34=(char) (210-c21);
                char c42=(char) (c14^18);
                char c11=(char) (c42-3);
                char c41=(char) (c11-2);
                char c44=c13;
                result.append(c11);
                result.append(c12);
                result.append(c13);
                result.append(c14);
                result.append('-');
                result.append(c21);
                result.append(c22);
                result.append(c23);
                result.append(c24);
                result.append('-');
                result.append(c31);
                result.append(c32);
                result.append(c33);
                result.append(c34);
                result.append('-');
                result.append(c41);
                result.append(c42);
                result.append(c43);
                result.append(c44);
                if (a(result.toString())==true) {
                    System.out.println("OK:"+result);
                }
            }
        }
    }
```

运行程序，程序输出：

```
OK:d2lu-bmVy-Y7hp-bgtl
OK:h4py-bmVy-Y7hp-fkxp
OK:d2lu-dkXw-W6nn-bgtl
OK:h4py-dkXw-W6nn-fkxp
OK:d2lu-fiZu-U5ll-bgtl
OK:h4py-fiZu-U5ll-fkxp
OK:d2lu-naBm-M1dd-bgtl
```

按顺序提交测试，刚好第一个就是flag

```
XCTF{d2lu-bmVy-Y7hp-bgtl}
```

第三题 神秘的txt（1000pt）

题目是一个txt文档：

```
XCTF{(37，99)(19,99)(19,108)(28,99)(28,108)(37,108)(37,117)(28,117)(19,117)}
```

- 原本第一个37 99中间的逗号是乱码，猜测是中文逗号，编码问题。仔细观察发现，以上坐标可以对应坐标系中的一个“九宫格”。有点脑洞，把九个坐标编好数字，然后按照坐标给出的顺序写数字就是答案。
- 一开始尝试的是数学坐标系：

```
(19,117)(28,117)(37,117)
(19,108)(28,108)(37,108)
(19,99)(28,99)(37,99)
```

```
7  8  9
4  5  6
1  2  3
```

- 尝试答案：

```
XCTF{314256987}
```

- 答案错误，后来想，可能是按照二维数组的排列顺序，数字的顺序我们就坚信是按照电脑键盘小键盘的排列顺序了，于是编写程序：

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    char array[118][118];
    for(int i=0;i<118;i++) {
        for(int j=0;j<118;j++) {
            array[i][j]=' ';
        }
    }
    array[37][99]='1';
    array[19][99]='7';
    array[19][108]='8';
    array[28][99]='4';
    array[28][108]='5';
    array[37][108]='2';
    array[37][117]='3';
    array[28][117]='6';
    array[19][117]='9';
    for(int i=0;i<118;i++) {
        for(int j=0;j<118;j++) {
            printf("%c",array[i][j]);
        }
    }
    return 0;
}
```

- 程序输出：

```
7  8  9
4  5  6
1  2  3
```

- 将上述程序变量定义的顺序写入flag，即为答案：

```
XCTF{178452369}
```

- 提交，答案正确。