

学习练手——XCTF黑客精神

原创

[WuYu_AS](#) 于 2021-07-17 16:37:12 发布 64 收藏

分类专栏: [CTF学习](#) 文章标签: [java](#) [android](#) [算法](#)

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/WuYu_AS/article/details/118857117

版权



[CTF学习](#) 专栏收录该内容

4 篇文章 0 订阅

订阅专栏

一 环境

手机: Pixel 1

系统: Android 8.1

软件: IDA 7.5、JADX

难度: 简单

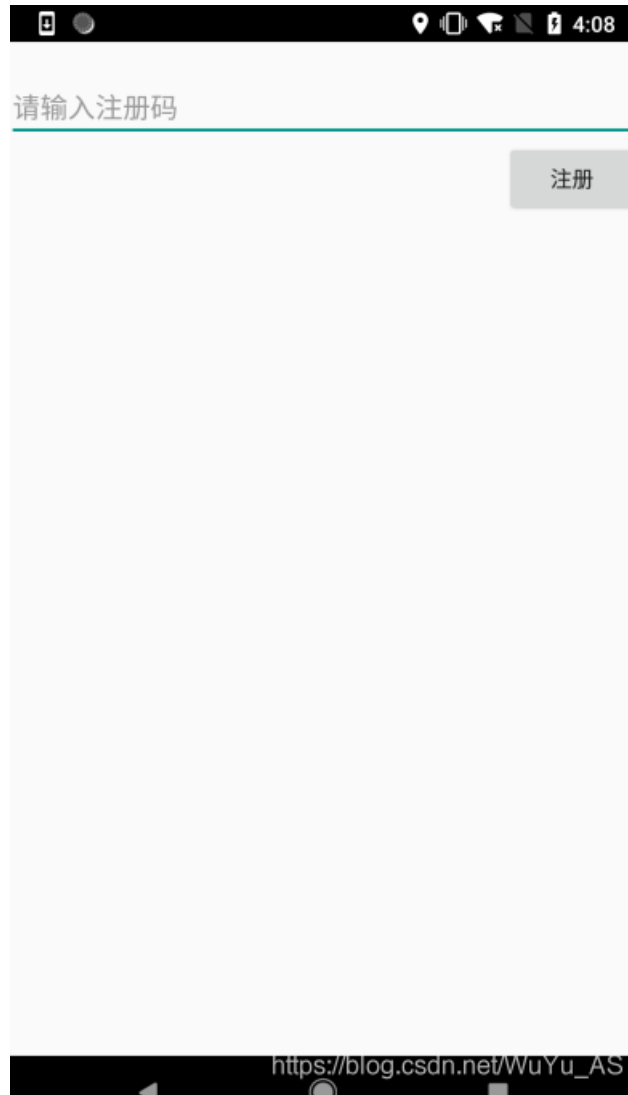
apk资源

链接: https://pan.baidu.com/s/1iEBK__qewKQAg9KFVraskA

提取码: jn3p

二 分析流程

1. 打开点击自由正义分享，再点击注册，会进入了注册界面



2. 输入11223344556677，再点击注册，点击好吧就会退出APP(点击对话框外的任意地方，不会退出APP)



3.打开JADX，搜索"您的注册码已保存"，然后看到了关键的saveSN

```
public class RegActivity extends Activity {
    private Button btn_reg;
    private EditText edit_sn;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reg);
        this.btn_reg = (Button) findViewById(R.id.button1);
        this.edit_sn = (EditText) findViewById(R.id.editText1);
        this.btn_reg.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                String sn = RegActivity.this.edit_sn.getText().toString().trim();
                if (sn == null || sn.length() == 0) {
                    Toast.makeText(RegActivity.this, "您的输入为空", 0).show();
                    return;
                }
                ((MyApp) RegActivity.this.getApplication()).saveSN(sn);
                new Builder(RegActivity.this).setTitle("回复").setMessage("您的注册码已保存").setPositiveButton("好吧", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        Process.killProcess(Process.myPid());
                    }
                }).show();
            }
        });
    }
}
```

https://blog.csdn.net/WuYu_AS

4.查看到了三个native方法

```
package com.gdufs.xman;

import android.app.Application;
import android.util.Log;

public class MyApp extends Application {
    public static int m = 0;

    public native void initSN();

    public native void saveSN(String str);

    public native void work();

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

    public void onCreate() {
        initSN();
        Log.d("com.gdufs.xman m=", String.valueOf(m));
        super.onCreate();
    }
}
```

https://blog.csdn.net/WuYu_AS

5.先查看 initSN方法只有在 本类的onCreate初始化，在JADX中搜索MyApp的类初始化的地方

节点	代码
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	MyApp myApp = (MyApp) getApplication();
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	MyApp myApp = (MyApp) MainActivity.this.getApplication();
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	int m = MyApp.m;

com.gdufs.xman.MainActivity.onCreate(Bundle) : void	MyApp myApp = (MyApp) getApplication();
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	if (MyApp.m == 0) {
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	MyApp myApp = (MyApp) MainActivity.this.getApplication();
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	((MyApp) MainActivity.this.getApplication()).work();
com.gdufs.xman.MyApp	public class MyApp extends Application {
com.gdufs.xman.RegActivity.onCreate(Bundle) : void	((MyApp) RegActivity.this.getApplication()).saveSN(sn);

6.发现关键的判断值 MyApp.m(图1), 搜索引用发现没有赋值的地方(图2), 那么说明有可能是SO里面, 加上APP初始化就进行判断, 说明MyApp.m赋值的initSN方法

```

3 public void onCreate(Bundle savedInstanceState) {
4     String str2;
5     super.onCreate(savedInstanceState);
6     setContentView(R.layout.activity_main);
7     String str1 = "Xman";
8     Log.d("com.gdufs.xman m=", str1);
9     MyApp myApp = (MyApp) getApplication();
10    int m = MyApp.m;
11    if (m == 0) {
12        str2 = "未注册";
13    } else if (m == 1) {
14        str2 = "已注册";
15    } else {
16        str2 = "已混乱";
17    }
18    setTitle(str1 + str2);
19    this.btn1 = (Button) findViewById(R.id.button1);
20    this.btn1.setOnClickListener(new OnClickListener() {
21        public void onClick(View v) {
22            MyApp myApp = (MyApp) MainActivity.this.getApplication();
23            if (MyApp.m == 0) {
24                MainActivity.this.doRegister();
25                return;
26            }
27            ((MyApp) MainActivity.this.getApplication()).work();
28            Toast.makeText(MainActivity.this.getApplicationContext(), MainActivity.workString, 0).show();
29        }
30    });
31 }

```

https://blog.csdn.net/WuYu_AS

图1

查找用例: com.gdufs.xman.MyApp.m : int

节点	代码
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	int m = MyApp.m;
com.gdufs.xman.MainActivity.onCreate(Bundle) : void	if (MyApp.m == 0) {
com.gdufs.xman.MyApp	public static int m = 0;
com.gdufs.xman.MyApp.onCreate() : void	Log.d("com.gdufs.xman m=", String.valueOf(m));

图2

7.打开IDA，导入SO，打开导出表搜索 initSN 发现并没有，说明是在JNI_OnLoad里进行动态注册

Name	Address	Ord
jni_tSN		

https://blog.csdn.net/WuYu_AS

8.查看JNI_OnLoad，并且导入jni.h文件，查看off_5004

```

jint JNI_OnLoad(JavaVM *vm, void *reserved)
{
    if ( !(*vm)->GetEnv(vm, &g_env, 65542) )
    {
        _android_log_print(2, "com.gdufs.xman", "JNI_OnLoad()");
        native_class = (*g_env)->FindClass(g_env, "com/gdufs/xman/MyApp");
        if ( !(*g_env)->RegisterNatives(g_env, native_class, off_5004, 3) )
        {
            _android_log_print(2, "com.gdufs.xman", "RegisterNatives() --> nativeMethod() ok");
            return 65542;
        }
        _android_log_print(6, "com.gdufs.xman", "RegisterNatives() --> nativeMethod() failed");
    }
    return -1;
}

```

https://blog.csdn.net/WuYu_AS

Address	Disassembly	Comment
.data:00005005	ULB 0	
.data:00005004	DCD aInitsn	; DATA XREF: JNI_OnLoad+32fo
.data:00005004		; JNI_OnLoad+38fo ...
.data:00005004		; "initSN"
.data:00005008	DCD aV	; "()"V"
.data:0000500C	DCD n1+1	
.data:00005010	DCD aSavesn	; "saveSN"
.data:00005014	DCD aLjavaLangStrin	; "(Ljava/lang/String;)V"
.data:00005018	DCD n2+1	
.data:0000501C	DCD aWork	; "work"
.data:00005020	DCD aV	; "()"V"
.data:00005024	DCD n3+1	

9.先查看n1,简单读取"/sdcard/reg.dat"文件(由于文件是放在SD卡里所以需要给APP存储权限就可以了),读取文件内的字符串,然后和"EoPAoY62@EIRD"进行比较

```

1 int __fastcall n1(int a1)
2 {
3     FILE *fpointer; // r0
4     FILE *fpointer_1; // r4
5     int v4; // r0
6     int fileSize; // r7
7     void *v6; // r5
8     int v8; // r0
9     int v9; // r1
10
11     fpointer = fopen("/sdcard/reg.dat", "r+");
12     fpointer_1 = fpointer;
13     if ( !fpointer )
14     {
15         v4 = a1;
16         return setValue(v4, 0);
17     }
18     fseek(fpointer, 0, 2);
19     fileSize = ftell(fpointer_1);
20     v6 = malloc(fileSize + 1);
21     if ( !v6 )
22     {
23         fclose(fpointer_1);
24         v4 = a1;
25         return setValue(v4, 0);
26     }
27     fseek(fpointer_1, 0, 0);
28     fread(v6, fileSize, 1u, fpointer_1);
29     *(v6 + fileSize) = 0;
30     if ( !strcmp(v6, "EoPAoY62@EIRD") )
31     {
32         v8 = a1;
33         v9 = 1;
34     }
35     else
36     {
37         v8 = a1;
38         v9 = 0;
39     }
40     setValue(v8, v9);
41     return j_fclose(fpointer_1);
42 }

```

https://blog.csdn.net/WuYu_AS

10. 会将结果通过setValue设置到 MyApp.m，那么就找到了关键的key"EoPAoY62@EIRD"

```

int __fastcall setValue(JNIEnv *a1, int a2)
{
    jclass v4; // r5
    jfieldID v5; // r0

    v4 = (*a1)->FindClass(a1, "com/gdufs/xman/MyApp");
    v5 = (*a1)->GetStaticFieldID(a1, v4, "m", "I");
    return ((*a1)->SetStaticIntField)(a1, v4, v5, a2);
}

```

11.查看动态注册里的n2(JAVA层saveSN)，分为三部分：

- 1.初始化 加密用的字符串；
- 2.将明文和从table中取出的字符进行异或；
- 3.写入/sdcard/reg.dat文件

```
1 int __fastcall n2(JNIEnv *env, jobject obj, jstring content)
2 {
3     FILE *fpointer; // r7
4     _DWORD *table_1; // r4
5     const char *v8; // r3
6     int v9; // r0
7     int v10; // r1
8     _WORD *v11; // r5
9     JNIEnv *v12; // r0
10    int table_index; // r4
11    JNIEnv v14; // r3
12    signed int index; // r6
13    const char *content_char; // r9
14    char *content_char_1; // r5
15    signed int content_char_len; // r10
16    char table_item; // r2
17    char content_char_item; // r3
18    _BYTE table[56]; // [sp+0h] [bp-38h] BYREF
19
20    fpointer = fopen("/sdcard/reg.dat", "w+");
21    if ( !fpointer )
22        return j__android_log_print(3, "com.gdufs.xman", byte_2DCA);
23    table_1 = table;
24    v8 = "W3_arE_wh0_we_ ARE";
25    do
26    {
27        v9 = *v8;
28        v8 += 8;
29        v10 = *(v8 - 1);
30        *table_1 = v9;
31        table_1[1] = v10;
32        v11 = table_1 + 2;
33        table_1 += 2;
34    }
35    while ( v8 != "E" );
36    v12 = env;
37    table_index = 2016;
38    *v11 = *v8;
39    v14 = *env;
40    index = 0;
41    content_char = v14->GetStringUTFChars(v12, content, 0);
42    content_char_1 = content_char;
43    content_char_len = strlen(content_char);
44    while ( index < content_char_len )
45    {
46        if ( index % 3 == 1 )
47        {
48            table_index = (table_index + 5) % 16;
49            table_item = table[table_index + 1];
50        }
51        else if ( index % 3 == 2 )
52        {
53            table_index = (table_index + 7) % 15;
54            table_item = table[table_index + 2];
55        }
56        else
57        {
58            table_index = (table_index + 3) % 13;
59            table_item = table[table_index + 3];
60        }
61        content_char_item = *content_char_1;
62        ++index;
63        *content_char_1++ = content_char_item ^ table_item;
64    }
65    fputs(content_char, fpointer);
66    return j_fclose(fpointer);
67 }
```

初始化 加密用的字符串

加密算法，最后使用的是异或

写入文件

12第一部分：.初始化 加密用的字符串，从伪代码中明显是固定的，加上也没有反调试，所以直接IDA，动态调试(下断点的地方 如图1)

```

.text:00001244 loc_1244          ; CODE XREF: n2+5C1j
.text:00001244      LDR     R0, [R3] ; "W3_arE_who_we_ARE"
.text:00001246      ADDS   R3, #8
.text:00001248      LDR.W  R1, [R3,#-4]
.text:0000124C      CMP    R3, R2
.text:0000124E      MOV    R5, R4
.text:00001250      STM    R5!, [R0,R1]
.text:00001252      MOV    R4, R5
.text:00001254      BNE    loc_1244
.text:00001256      LDRH   R3, [R3]
.text:00001258      MOV    R1, R9
.text:0000125A      MOV    R0, R6
.text:0000125C      MOVS   R2, #0
.text:0000125E      MOV.W  R4, #0x7E0
.text:00001262      STRH   R3, [R5]
.text:00001264      LDR    R3, [R6]
.text:00001266      MOVS   R6, #0
.text:00001268      LDR.W  R3, [R3,#0x2A4]
    
```

循环两次

每次设置到R5中(记住第一次的地址)

在这里下断点，查看第一次记录的地址，就能获取到结果

https://blog.csdn.net/WuYu_AS

图1

```

FFC3C5C0  20 C9 C5 FF B1 10 00 F1 00 50 44 FF 30 C0 C5 FF  (.....0...
FFC3C5D0  42 02 00 00 E2 93 FC 10 B8 71 0A F1 02 00 00 00  B.....q.....
FFC3C5E0  A0 02 65 EF 1B A2 C8 D4 57 33 5F 61 72 45 5F 77  ..e...w3_arE_w
FFC3C5F0  68 4F 5F 77 65 5F 41 52 45 00 FC 10 E2 93 FC 10  h0_we_ARE.....
FFC3C600  9C F0 F0 EF 02 00 00 00 00 00 00 28 C9 C3 FF  (.....(.....
FFC3C610  00 00 00 00 00 80 67 EF 20 C7 C3 FF 07 D1 CD D4  .....g.....
    
```

需要的字符串

图2

13.第二部分：加密算法很简单，实现的JAVA代码

```

public static String myEncript_CTF(String input){
    char[]table={0x57,0x33,0x5F,0x61,0x72,0x45,0x5F,0x77,0x68,0x4F,0x5F,0x77,0x65,0x5F,0x41,0x52,0x45,0x00};
    char[]result=new char[input.length()];
    int table_index=2016;
    char table_item=0;
    for (int i = 0; i <input.length() ; i++) {
        if(i%3==1){
            table_index=(table_index+5)%16;
            table_item=table[table_index+1];
        }else if(i%3==2){
            table_index=(table_index+7)%15;
            table_item=table[table_index+2];
        }else{
            table_index=(table_index+3)%13;
            table_item=table[table_index+3];
        }
        result[i]= (char) (input.charAt(i)^table_item);
    }

    return new String(result);
}
    
```

14.因为table_item的生成和明文没有任何关系，加上最后是异或，说明了传入密文，返回的就是明文

```

System.out.println(myEncript_CTF("EoPAoY62@EIRD"));
201608Am!2333
    
```


15.输入正确的flag,重新进入app,按照指定的格式xman{201608Am!2333}! 提交就好了, 不过这是很久以前的比赛APP, 所以当成功的标志就好了

