【安卓逆向】CTF实战分析（什么是CTF,我们一起来看看）

原创

BL11.11 于 2020-01-14 11:53:21 发布 921 收藏 1

文章标签： 安全 信息安全 android

本文链接：https://blog.csdn.net/BL9794/article/details/103970717

既然这篇文章提到了CTF 我就先来科普一下 What is CTF???

赛事介绍:

CTF是一种流行的信息安全竞赛形式，其英文名可直译为“夺得Flag”，也可意译为“夺旗赛”。其大致流程是，参赛团队之间通过进行攻防对抗、程序分析等形式，率先从主办方给出的比赛环境中得到一串具有一定格式的字符串或其他内容，并将其提交给主办方，从而夺得分数。为了方便称呼，我们把这样的内容称之为“Flag”。

赛事起源:

CTF起源于1996年DEFCON全球黑客大会，以代替之前黑客 们通过互相发起真实攻击进行技术比拼的方式。发展至今，已 经成为全球范围网络安全圈流行的竞赛形式。

竞赛模式:

CTF竞赛模式具体分为以下三类：

```
一、解题模式（Jeopardy）
```

在解题模式CTF赛制中，参赛队伍可以通过互联网或者现场网络参与，这种模式的CTF竞赛与ACM编程竞赛、信息学奥赛比较类似，以解决网络安全技术挑战题目的分值和时间来排名，通常用于在线选拔赛。题目主要包含逆向、漏洞挖掘与利用、Web渗透、密码、取证、隐写、安全编程等类别。

```
二、攻防模式（Attack-Defense）
```

在攻防模式CTF赛制中，参赛队伍在网络空间互相进行攻击和防守，挖掘网络服务漏洞并攻击对手服务来得分，修补自身服务漏洞进行防御来避免丢分。攻防模式CTF赛制可以实时通过得分反映出比赛情况，最终也以得分直接分出胜负，是一种竞争激烈，具有很强观赏性和高度透明性的网络安全赛制。在这种赛制中，不仅仅是比参赛队员的智力和技术，也比体力（因为比赛一般都会持续48小时及以上），同时也比团队之间的分工配合与合作。

```
三、混合模式（Mix）
```

结合了解题模式与攻防模式的CTF赛制，比如参赛队伍通过解题可以获取一些初始分数，然后通过攻防对抗进行得分增减的零和游戏，最终以得分高低分出胜负。采用混合模式CTF赛制的典型代表如iCTF国际CTF竞赛。

CTF的江湖:

我们一般习惯把ctf分成四类：

1.题目来源是渗透实战或安全研究的paper

- 国内：BCTF(blue lotus)、0CTF(0ops)、HITCON(hitcon\217)
- 国际：DEFCON、CODEGATE、PlaidCTF、Boston Key Party CTF

2.题目来源是各类有趣的漏洞，举办者一般是赛棍

- 国内：HCTF(HDUISA)、CCTF(0xFA)、XDCTF(XDSEC)、SCTF(三叶 草)、ALICTF(alibaba)、360

3.举办者一般是赛棍，但往往并不擅长于举办比赛

- 国内：SSCTF(四叶草)、RCTF(ROIS)、华山杯、ISG、信息安全国 赛、XCUNA(高校网安联赛)

4.举办者一般是技术从业者，并不懂CTF

- 国内：WHCTF、新XDCTF、首都安全日等...

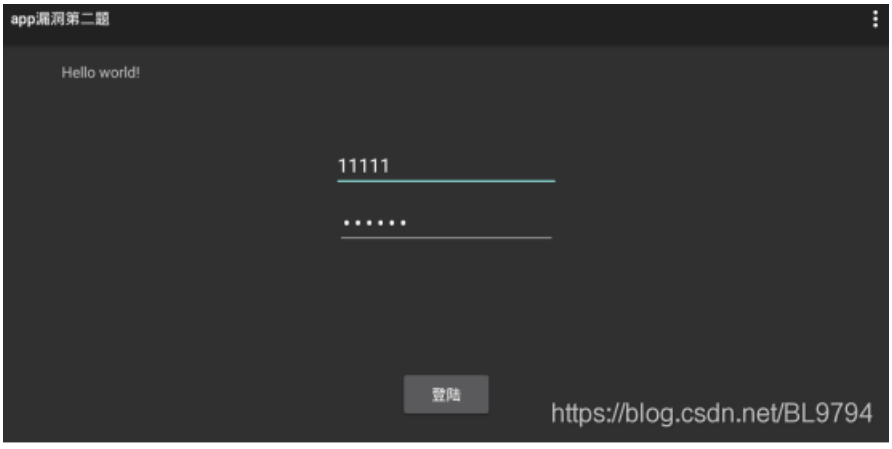
那么科普完毕 我们接下来就实战一波

我这里就拿国内一家安全公司出的一个入门级app的题目做分析

要求如下:

1.有壳就脱壳　2.拿到此题的Flag

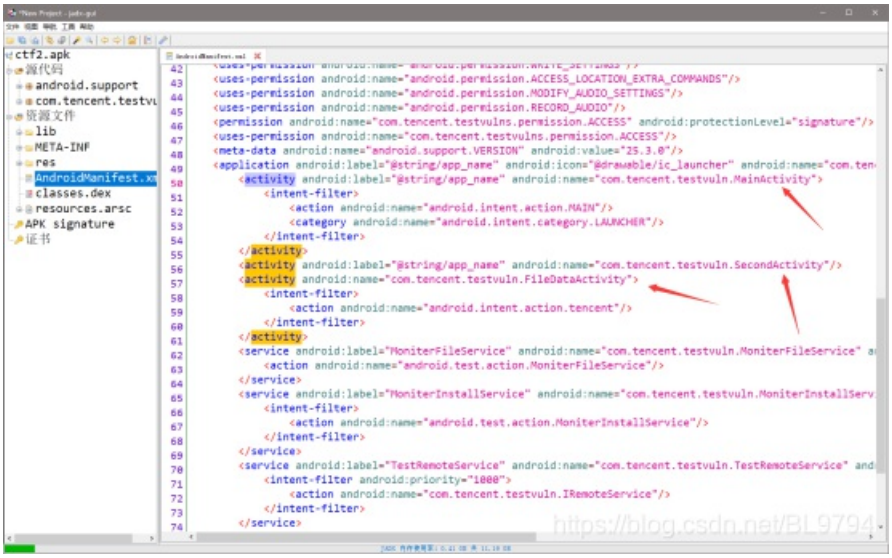
首先拿到一个app用工具查一下 这里提示没有壳 那么我们安装一下看一下软件页面



一打开就是如此 两个编辑框 然后点击登陆就跳转到另一个页面提示”Waiting fot you!”

既然如此 打开jadx-gui工具反编译一波 从java层入手

首先找到他的”AndroidManifest.xml” 里面有个”activity”标签里面就是他的活动页面的信息



可以看到这里有3个活动页面信息

有这两个属性的值上面的活动页面就是我们的入口页面(一开打软件所显示的页面信息)

既然如此我们之前看到过有个登陆的按钮在一启动的活动下

那么我们到此活动页面看一下按钮下的逻辑事件(“com.tencent.testvuln.MainActivity”)

来到这个类里面找到按钮点击事件 也就是onClick事件下

```
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.button1 /*2131165187*/:
            if (this.c.getText().length() == 0 || this.d.getText().length() == 0) {
                Toast.makeText(this, "不能为空", 1).show();
                return;
            }
            String obj = this.c.getText().toString();
            String obj2 = this.d.getText().toString();
            Log.e("test", obj + " test2 = " + obj2);
            Intent intent = new Intent(this, SecondActivity.class);
            intent.putExtra("ili", obj);
            intent.putExtra("lil", obj2);
            startActivity(intent);
            return;
        default:
            return;
    }
```


首先一个switch语句获取view.id 进入第一个分支也就是 case R.id.button1

然后接着判断我们输入的内容是不是为空 或者说输入的长度是否为0 如果条件成立就进行一个弹窗提示我们输入的内容”不能为空”

然后定义两个string类型的变量接收我们输入的内容

Obj是第一个编辑框的内容 obj2是第二个编辑框的内容

然后startActivity跳转到另一个活动页面也就是”SecondActivity”并将我们输入的内容也传递过来 然后在onCreate初始化函数进行判断

```
/* access modifiers changed from: protected */
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_main2);
    Intent intent = getIntent();
    String stringExtra = intent.getStringExtra("ili");
    String stringExtra2 = intent.getStringExtra("lil");
    if (Encryto.doRawData(this, stringExtra + stringExtra2).equals("VEIzd/V2UPYNdn/bxH3Xig==")) {
        intent.setAction("android.test.action.MoniterInstallService");
        intent.setClass(this, MoniterInstallService.class);
        intent.putExtra("company", "tencent");
        intent.putExtra("name", "hacker");
        intent.putExtra("age", 18);
        startActivity(intent);
        startService(intent);
    }
    Editor edit = getSharedPreferences("test", 0).edit();
    edit.putString("ilil", stringExtra);
    edit.putString("lili", stringExtra2);
    edit.commit();
}
```


这一块儿的逻辑就很清晰了

if (Encryto.doRawData(this, stringExtra + stringExtra2).equals(“VEIzd/V2UPYNdn/bxH3Xig==”))

所有的比较逻辑都是在这里了

调用一个Encryto类下的doRawData将this 以及stringExtra + stringExtra2).equals("VElzd/V2UPYNdn/bxH3Xig=="传递进去

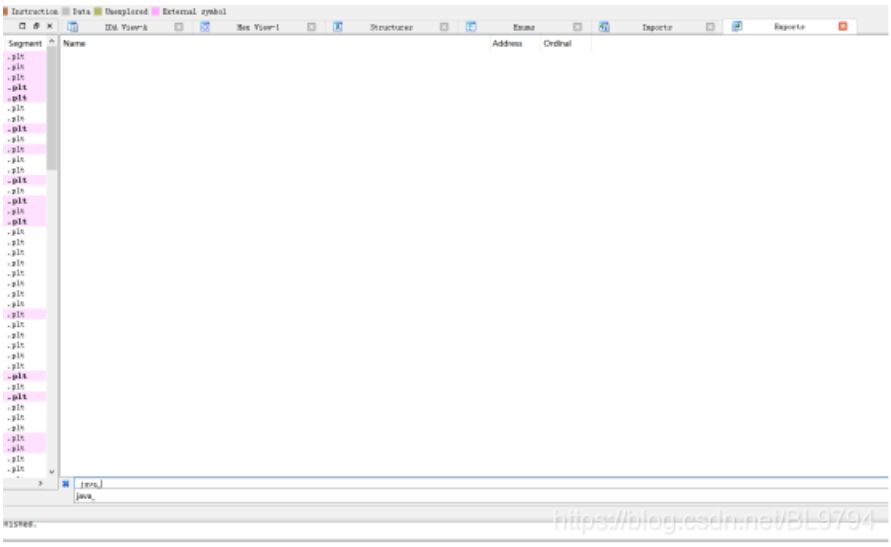
我们现在跟踪到doRawData方法

```java
package com.tencent.testvuln.c;

public class Encryto {
    public static native int checkSignature(Object obj);

    public static native String decode(Object obj, String str);

    public static native String doRawData(Object obj, String str);

    public static native String encode(Object obj, String str);

    public native String HelloLoad();

    static {
        System.loadLibrary("JNIEncrypt");
    }
}
```


发现System.loadLibrary(“JNIEncrypt”); 加载了这样一个so库 此方法也是由native关键词所修饰的

那么打开此app的lib文件夹发现也只有一个so文件库

拿出我们的ida工具反编译一波先



发现并没有java_ 静态注册的方法那么我们也只能找到JNI_OnLoad动态注册的位置

```
17  return result;
18}
```


来到了

```
1 unsigned int __fastcall register_ndk_load(JNIEnv *a1)
2 {
3   JNIEnv *v1; // r4
4   int v2; // r1
5   unsigned int result; // r0
6
7   v1 = a1;
8   v2 = ((*a1)->FindClass)(a1, "com/tencent/testvuln/c/Encryto");
9   if ( v2 )
10    result = (((*v1)->RegisterNatives)(v1, v2, off_6008, 4) >> 31) ^ 1;
11  else
12    result = 0;
13  return result;
14}
```


RegisterNatives就是注册的函数 他的第三个参数就是我们要找到的方法体 第四个参数就是他注册的个数

```
                                ; "UNSIGNATURE"
008     | DCD aChecksignature_0   ; DATA XREF: register_ndk_load+22to
                                ; .text:off_221C4o
                                ; "checkSignature"
        DCD aLjavaLangObjec     ; "(Ljava/lang/Object;)I"
        DCD check+1
        DCD aDecode_0           ; "decode"
        DCD aLjavaLangObjec_0   ; "(Ljava/lang/Object;Ljava/lang/String;)L"...
        DCD decode+1
        DCD aEncode_0           ; "encode"
        DCD aLjavaLangObjec_0   ; "(Ljava/lang/Object;Ljava/lang/String;)L"...
        DCD encode+1
        DCD aDorawdata_0        ; "doRawData"
        DCD aLjavaLangObjec_0   ; "(Ljava/lang/Object;Ljava/lang/String;)L"...
        DCD doRawData+1
ta      ends
```



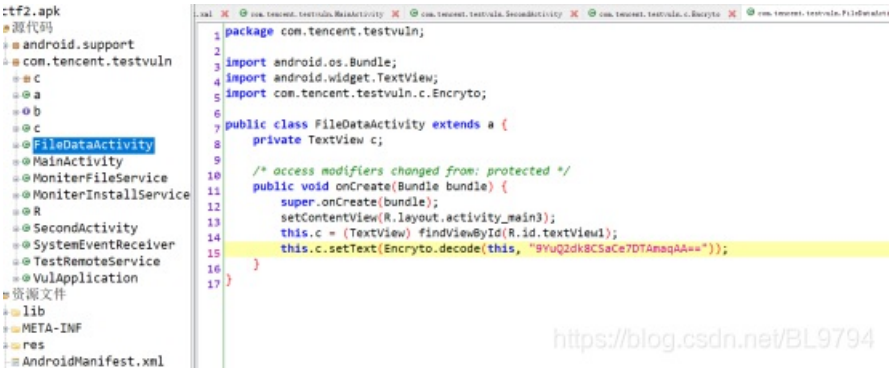
```
1 int __fastcall doRawData(JNIEnv *a1, int a2, int a3, int a4)
2 {
3   JNIEnv *v4; // r4
4   int v5; // r9
5   int v6; // r6
6   int v7; // r8
7   int result; // r0
8   jstring (*v9)(JNIEnv *, const jchar *, jsize); // r6
9   char *v10; // r5
10  size_t v11; // r2
11  int v12; // [sp+0h] [bp-28h]
12  int v13; // [sp+18h] [bp-10h]
13
14  v4 = a1;
15  v5 = a4;
16  if ( j_checkSignature(a1) != 1
17    || (strcpy(&v12, "thisisatestkey=="),
18        v6 = ((*v4)->GetStringUTFChars)(v4, v5, 0),
19        v7 = j_AES_128_ECB_PKCS5Padding_Encrypt(),
20        ((*v4)->ReleaseStringUTFChars)(v4, v5, v6),
21        result = ((*v4)->NewStringUTF)(v4, v7),
22        _stack_chk_guard != v13) )
23  {
24    do
25    {
26      v9 = (*v4)->NewString;
27      v10 = UNSIGNATURE[0];
28      v11 = strlen(UNSIGNATURE[0]);
29    }
30    while ( _stack_chk_guard != v13 );
31    result = (v9)(v4, v10, v11);
32  }
33  return result;
34}
```


然后我们一步步跟踪到了这里 发现采用了一个AES/ECB/PKCS5Padding的加密方式

Key勒就摆在面前的”thisisatestkey==”

然后我们来解密一波

将VEIzd/V2UPYNdn/bxH3Xig== 解密得出”aimagetencent”



第三个页面这里会解密一个9YuQ2dk8CSaCe7DTAmaqAA值

将9YuQ2dk8CSaCe7DTAmaqAA 解密得出”Cas3_0f_A_CAK3”

谢谢大家的支持，另外在给大家分享一个棋牌协议分析的教学视频
https://pan.baidu.com/s/1KPMFnysCPi_dVMFZiJyKKw