DDCTF2018安卓部分WriteUp

原创

Magic1an 于 2018-04-23 11:28:47 发布 730 收藏
分类专栏： android

0x0 前言

感觉自己还是太菜了，还静不下心去学习，可能这就是自己这么菜的原因吧。七天的时间零零散散的做了三个安卓题，简单写一下思路。题目及部分解题脚本https://pan.baidu.com/s/1MpnW_st_VnRRUJx6rHqZOA

0x1 RSA

刚看到题目还以为程序实现了RSA加密，因此找RSA加密算法找了好久。后来才发现程序经过运算生成一个大数，然后对输入进行取模，如果能整除则正确。

本题目的JAVA代码比较简单，将输入的字符串传入jni层的stringFromJNI函数，如果stringFromJNI函数的返回值为true则显示Correct.

```
static {
    System.loadLibrary("hello-libs");
}

public MainActivity() {
    super();
}

public void onClickTest(View arg3) {
    this.n.setText("Empty Input");
    if(this.stringFromJNI(this.m.getText().toString())) {
        this.n.setText("Correct");
    }
    else {
        this.n.setText("Wrong");
    }
}
```

将so文件拖入IDA，发现主要处理逻辑为__aeabi_wind_cpp_prj（int a1）函数，因此我们将研究重点放到__aeabi_wind_cpp_prj函数中。

该函数首先判断输入长度是否为31位，然后将输入的每一位与key的每一位进行异或处理。

```
if ( *(_DWORD *)(v1 - 12) == 31 )             // cmp length 31
{
  v3 = 0;
  do
  {
    v4 = *((_BYTE *)&key - v3);
    if ( *(_DWORD *)(v1 - 4) >= 0 )
    {
      v5 = *((_BYTE *)&key - v3);
      sub_2FF8C(v29);
      v4 = v5;
      v1 = *v29;
    }
    v42[-v3] = *(_BYTE *)(v1 - v3) ^ v4;       // 将输入的每一位与key的每一位进行异或
    --v3;
  }
  while ( v3 != -31 );
```

```
while ( 1 )
{
  if ( v10 >= 1 && v9 < length
  {
    v2 = 0;
    if ( v7[10] != *v7 )
      break;
  }
  ++v9;
  ++v7;
  if ( ++v8 >= 10 )
  {
    v9 = v27 + 10;
    v7 = v30 + 10;
    ++v10;
    v8 = 0;
    if ( v10 < 5 )
      goto LABEL_7;
```

这一块主要是判断输入异或后的结果前十位与十一到二十位、二十一到三十位、三十一位是否相等，所以输入的字符只看前十位即可，后面的与前面的相等。下面又将前十位与后面分隔开，将前十位转成long long 型。

```
    sub_31364(&v32, (int)v42, (int)&v38);
    sub_2F50C(v29, &v32);                   // 置0
    sub_308E4((int *)(v32 - 12));
    sub_30A08(&v39, v29);
    j_str2vec((int)&v40, (int)&v39);
    sub_308E4((int *)(v39 - 12));
    v12 = j_atoll(*v29);
```

程序下面做了一系列的运算，最后得到一个大数5889412424631952987

```
  }
  v28 = v31;
  v18 = j_atoll(v31);                       // 5889412424631952987
  j_j___aeabi_uldivmod(v18, v12);
  if ( v19 )
    goto LABEL_47;
  v20 = j_j___aeabi_uldivmod(v18, v12);
```

然后将大数和上面被转成long long的输入做取模运算，判断模是否为0，既5889412424631952987%s==0？这地方有点问题的是ida看到的ulddivmod函数里面的逻辑是返回a1/a2的结果。

因此我们只需将5889412424631952987做大数分解，求他的两个因子，然后做异或即可。我的脚本写的比较乱，是去爆破输入，只可做参考。

0x2 Hello Baby Dex

```
else {
    EditText v0_2 = this.val$input_text;
    Editable v0_3 = v0_2.getText();
    v0_1 = TextUtils.isEmpty(((CharSequence)v0_3));
    if(!v0_1) {
        v0_2 = this.val$input_text;
        v0_3 = v0_2.getText();
        String v0_4 = v0_3.toString();
        StringBuilder v1_2 = new StringBuilder();
        String v2_1 = "DDCTF{";
        v1_2 = v1_2.append(v2_1);
        v2_1 = this.val$result;
        v1_2 = v1_2.append(v2_1);
        v2_1 = "}";
        v1_2 = v1_2.append(v2_1);
        v1_3 = v1_2.toString();
        v0_1 = v0_4.equals(v1_3);
        if(v0_1) {
            v0_5 = MainActivity.this;
            v1_3 = "恭喜大佬！密码正确！";
            v0_6 = Toast.makeText(((Context)v0_5), ((CharSequence)v1_3), 0);
            v0_6.show();
            return;
        }
    }
```

程序关于flag的逻辑是相当简单了...可能这就是叫baby dex的原因吧..只是对输入与程序的拼接的字符串做了个equals比较

```
SignCheck v10 = new SignCheck(this, ((Context)this), v0_3);
v10.check();
Debug.isDebuggerConnected();
```


不过app做了签名的校验与debug的检测，所以如果我们动态调试的话可能会有些麻烦。但是仔细看一下前面flag比较的代码，我们可以发现equals函数是我们的输入调用的，把程序计算好的flag作为参数做对比，所以我们可以使用xposed去hook一下equals函数获取参数来获取flag。

```
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable {
        if (!loadPackageParam.packageName.equals("cn.chaitin.geektan.crackme")){
            return;
        }
        XposedHelpers.findAndHookMethod("java.lang.String", loadPackageParam.classLoader, "equals", Object.
            @Override
            protected void beforeHookedMethod(MethodHookParam param) throws Throwable {
                if(param.args[0].toString().contains("DDCTF")){
                    XposedBridge.log("before"+param.args[0]);
                }

            }

            @Override
            protected void afterHookedMethod(MethodHookParam param) throws Throwable {

            }
        });
    }
```

0x3 Diffie-Hellman

这个题感觉是在第一题上面改的，java层没有变化，改动了一些jni中的代码，如果看懂了第一题第三题也就没什么问题了。

```
  j_j___aeabi_ldivmod(2LL, p);
  if ( v16 ^ 1 | v6 )
  {
    do
    {
      v12 = 1;
      if ( v9 >= v16 )
        v12 = 0;
      v13 = 1;
      if ( v9 >> 31 >= v6 )
        v13 = 0;
      if ( v9 >> 31 != v6 )
        v12 = v13;
      HIDWORD(v14) = v11 >> 31;
      LODWORD(v14) = 2 * v11;                 // v14=2^input
      j_j___aeabi_ldivmod(v14, v10);          // v10=0x1E75BF5AB47900
      ++v9;
    }
    while ( v12 );
  }
  v1 = 1;
  if ( mod_residual != v11 )
    v1 = 0;
}
```


大致流程是将你的输入作为2的幂数，与v10进行取模，将得到的模与mod_residual进行比较 既(2^input%v10)==mod_residual ?

因为我对于密码学没有过深入的了解，所以只能采用了爆破的方法，希望有好办法的大佬能给予指导~

```
mod_res=0x2BF5B02AC9500
mod=0x1E75BF5AB47900
for i in range(150000,550000):
    if (2**i) %mod==mod_res:
        print(i)
        break
print("stop..")
```