ez_android

发表于 2021-01-15 分类于 Challenge， 2020， 安洵杯， Reverse
Challenge | 2020 | 安洵杯 | Reverse | ez_android

点击此处获得更好的阅读体验

WriteUp来源

https://xz.aliyun.com/t/8582

题目考点

- *APK逆向*

解题思路

拖进gda看一下，发现入口是一个NativeActivity

看了一下好像还有一些Java层的东西，看一下

发现按钮会调用getRes方法，跟踪发现getRes是native方法，还找到了提示flag是否正确的方法。

分析librun.so，找了一下没发现JNI_OnLoad，直接分析getRes方法

从参数中获取输入的flag，可以分析出flag的长度为20，同时将flag复制到内存中

FUN_00013490函数传递了含flag的内存地址，跟进后发现无法解析。推测可能会运行时解密

继续向下分析

发现是一个散列算法，将flag计算后的结果进行散列后与数组中的数据进行比较，如果不符合则弹出提示。

显然关键在于如何找到FUN_00013490的解密函数，检查文件发现entry的数据不正常，可能是存储了加密所需的信息。

如果是一个段加密，那么可能会在so加载时进行解密。同时解密过程需要寻找so基地址和使用mprotect的方式修改内存。不妨从寻找mprotect的调用处着手

发现7处调用mprotecrt的地方，先在第一个地方看看

发现fscanf的输入格式是%p-%p，应该是从maps中寻找地址用的。同时发现这个函数是init_array函数调用的一环。

分析一下这个函数，发现在两个mprotect之间夹着一个循环

可以看到这个循环里会对base_addr加一个偏移的位置与某个数组的某个元素进行异或操作。再将异或后的结果减去当前计数器计数。我太菜了看不懂ghidra这段的伪代码，换个工具

舒服多了，很清楚看到是对下标为计数器余50的元素进行异或，同时可以发现FUN_00013490这个函数位于.anti段中。根据段的偏移和大小来编写程序解密这个段

解密后可以分析整个流程

首先去第一个字符，如果余6的结果是0和3则到FUN_00012508，1和4到FUN_00013644，2和5到FUN_00013788

跟进这个三个函数，发现他们之间的区别在于是否有对异或后的结果加一个值

以FUN_00012508为例，这里换用ida

可见传入的flag的前四个字符会先复制到tmp_arr中，然后与v10指向的数组的每个元素进行异或。这里v10的元素可知为AA,BB,CC,DD。将异或后的所有数相加余3，当为0则进入sub_38CC，为1则进入sub_3964，为2则进入sub_39FC。

先进sub_38CC看看

可以看到flag的前14个字符会与v3中的相关元素进行异或，第5个元素+5，第19个元素与第5个元素异或。其余的两个函数处理过程一样，不过v3的值不同

如何判断到底执行了哪个函数，不妨做一波猜测。猜测前四个字符是flag头，可能是D0g3,d0g3,flag等等。写一个小程序将这些头加密一遍与程序中加密后的数据进行比较，可以得出D0g3的可能性高。

根据猜测可知调用FUN_00013788和sub_39FC。如果没有进行散列，则确定加密流程后便能写出解密流程得到结果。但函数处理完成后会经过散列处理，散列导致的不可逆是本题最大的败笔。出题人再次深感抱歉（出题人太菜了）

如果硬要解，提供两个思路：

1.利用已知信息和产生冲突元素时所确定的先后顺序关系来进行爆破

2.猜flag：根据加密流程可知flag的后6个字符没有经过最后一步的处理，分析散列后flag的明文信息可以确定后6个字符的位置，再根据猜测的flag头进一步减少需要猜测的信息。最后剩下来的元素一一与v3中的值异或，得到产生的明文来猜测

Flag

- **本文作者：** CTFHub
- **本文链接：** https://writeup.ctfhub.com/Challenge/2020/安洵杯/Reverse/ijGAaK1zFwJgL4f2pi2YqA.html
- **版权声明：** 本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！

Challenge # 2020 # Reverse # 安洵杯
deebugging
easy_web