【HCTF】2015hctf单刷_writeup

原创

Angel枫丨灬红叶 于 2015-12-09 00:02:22 发布 3544 收藏

分类专栏： CTF 文章标签： ctf hctf

本文链接：https://blog.csdn.net/yuanyunfeng3/article/details/50226705

这是一份HCTF不完全writeup。。

0x00 misc1

拿到题目，binwalk分析下文件。发现zip文件结尾和png文件开头中间有一段数据，拿出该段数据，base64既视感，base32既视感，base16既视感。。

```
flag:hctf{nn1sc_ls_s0_34sy!}
```

0x01 web1

访问，用burp抓包，看头。。

```
hctf{w3lcome_t0_hc7f_f4f4f4}
```

0x02 personalblog

flag在源码里。

github敏感信息泄漏，google搜不到的，比赛前才上传的源码。

直接在github里面search of all 搜索博主。

```
hctf{H3xo_B1og_Is_Niu8i_B1og}
```

0x03 fuck ===

产生两个具有相同md5值的二进制文件，使用python对其进行urlencode，提交即可。

```
hctf{dd0g_fjdks4r3wrkq7jl}
```

0x04 andy

逆向apk jd-gui打开，简单分析一下，python脚本如下

```
import base64
a1 = "0 1 2 3 4 5 6 7 8 9 a b c d e f g h i j k l m n o p q r s t u v w x y z = A B C D E F G H I J K L
a2 = "W,p,X,4,5,B,q,A,6,a,V,3,r,b,U,s,E,d,C,c,D,O,t,T,Y,v,9,Q,2,e,8,P,f,h,J,N,g,u,K,k,H,x,L,w,R,I,j,i,y

a1 = a1.split(' ')
a2 = a2.split(',')
output = ''
input = 'SRlhb70YZHKvlTrNrt08F=DX3cdD3txmg'
for x in input:
    output+=a1[a2.index(x)]
print base64.b64decode(output)[::-1]
```

`hctf{and8n6yandr3w2i0dhdu1s8}`

0x05 友善的逆向（真友善。。

问题描述

果然很友善，check按钮都会乱跳，搞的我以为作者没写check事件的处理过程呢。。

使用OD载入文件。。 程序秒退，应该是加了dedug判断的函数，如果存在调试器就退出进程。

解决方案

首先使用IDA对`call isdebug`这个函数进行patch操作，对其机器码全部赋值为90(nop) 再使用OD载入时，程序不秒退了。

对于无法点击check按钮，直接进入该事件的处理过程即可，无视按钮（劳资根本不需要按钮。。）具体操作是：在判断事件的swich的任意case位置下断点，然后更改eip至想要执行的事件过程里。

静态分析

使用IDA进行静态分析，发现其第一个函数`sub_401DA0`进行了第一次字符串判断。分析后得到，字符串总长度为22
其中xxx还不确定

`HCTF{xxxx}`

然后继续静态分析。。我靠。。 算法没学好`sub_401BB0`这个函数，看得我眼睛都要瞎了，最后分析出其具体功能是返回当前字符的索引，若为字母则为字母表索引x。若为数字则为数字表索引。（这里用OD动态分析更快）。总共对12个数据进行操作，测试数据为从a-o。

大写为:0x00+x
小写为:0x64+x
数字为:0xC8+x

测试字串
`HCTF{abcdefghixyoXXXX}`

之后进行了替换操作
swap 6,0
swap 8,3
swap 5,2
swap 4,11

有点坑的是，除了主线程之外，进程还创建了两个分线程去对两个变量进行操作，每隔一段时间，更换一下变量。而这两个变量正好是产生最后异或key的关键值。。。（其实最后想到的是。。直接IDA倒着流程看就直接得到了。。我是煞笔。。在这里卡了很久，最终用OD解决的，不过也顺便补了补C++关于多线程的姿势）

获得最终flag如下。

`HCTF{UareS0cLeVerGg04}`

0x06 nes

无敌+4爆导弹，谁挡谁死，最后通过出flag，使用正确的模拟器，分离图层。

`ILOVENESFUCKYOUHCGORSA`

0x07 re250

逆向分析一下（使用tr，动态看汇编），

用tr动态看下汇编：

写出input2output的python代码

```
al = (x)&0xfc>>2:                              bx=1:  to switch    //入口点

bx == 1 : al = (x)&3 << 4 + (x+1)&0xf0 >>4 :   bx=2:  to switch
bx == 2 : al = (x+1) &0f <<2 + (x+2)&0xc0>>6 : bx=3:  to switch
bx == 3 : al = (x+2) &0x3f:                    bx=4:  to switch
bx == 4 : x = x+3;                  to begin
```

后面进行了switch,分几个case

```
al<0x19:       al+=0x41:          f+=al:  to bx switch
0x19<al<=0x33: al+=0x61-0x1a:     f+=al:  to bx switch
al==0x3e:      al = 2b:           f+=al:  to bx switch
al==0x3f:      al = 2f:           f+=al:  to bx switch
default:       al+=0x30-0x34:     f+=al:  to bx switch
```

3 -> 4的扩展

cx ==8以上循环 8次

循环结束后

循环16次 每次涉及到2个swap，总共32位

逆向出来的源码如下

```
#!/usr/bin/env python
# -*- coding: gbk -*-
# -*- coding: utf_8 -*-
#Date: 2015/12/6
#Create By WinterSun
#pip install simplejson requests
#注意优先级，>>大于&
#左移时注意溢出情况
#loop指令与cx有关，loop一次，cx自减少1
# 进行代码转换：

input = ''
####
output = '45,49,42,55,56,4E,6F,4A,66,49,5D,73,5D,45,4E,41,48,7B,66,6B,48,62,75,35,40,37,69,42,43,69,43,
output = output.split(',')
temp = ''
```

```
    for x in output:
        temp+=chr(int(x,16))
    output = temp
    ####

    i = 0 #下标
    bx = 0 #选择器
    al = 0
    def switch_calc(al):
        global output
        if(al<=0x19):
            al+=0x41
            output+=chr(al&0xff)
        elif(al>0x19 and al<=0x33):
            al+=0x61-0x1a
            output+=chr(al&0xff)
        elif(al==0x3e):
            al = 0x2b
            output+=chr(al&0xff)
        elif(al==0x3f):
            al = 0x2f
            output+=chr(al&0xff)
        else:
            al+=0x30-0x34
            output+=chr(al&0xff)

    while(i<len(input)):
        if(bx==0):
            al =(ord(input[i])&0xfc)  >> 2
            switch_calc(al)
            bx = 1
        elif(bx == 1):
            al = (ord(input[i])&0x3) << 4 &0xff
            ah = (ord(input[i+1])&0xf0) >>4
            al +=ah
            switch_calc(al)
            bx = 2
        elif(bx == 2):
            al = (ord(input[i+1])&0x0f) << 2 &0xff
            ah = (ord(input[i+2])&0xc0) >> 6
            al+=ah
            switch_calc(al)
            bx = 3
        elif(bx == 3):
            al =  ord(input[i+2])&0x3f
            switch_calc(al)
            bx = 4
        else:
            i += 3
            bx = 0
    i = 0
    while(i<len(output))
        al = output[i] ^0x7
        ah = output[i+0x10] ^0x0c
        output[i+0x10] = al
        output[i] = ah
        i+=1
    raw_input()
```

分析下

源程序对输入进行4个一组进行编码，然后换位。最后与内存中的32位字符串进行比对。
输入应该为32位字符串。
逆向思路》》先进行换位，然后用源程序的机制，进行字串爆破。

逆向的换位代码如下

```
    while(i<len(output))
        al = output[i] ^0xc
        ah = output[i+0x10] ^0x7
        output[i+0x10] = al
        output[i] = ah
        i+=1
```

然后进行字串爆破

```
def switch_calc(al):
    if(al<=0x19):
        al+=0x41
        return al&0xff
    elif(al>0x19 and al<=0x33):
        al+=0x61-0x1a
        return al&0xff
    elif(al==0x3e):
        al = 0x2b
        return al&0xff
    elif(al==0x3f):
        al = 0x2f
        return al&0xff
    else:
        al+=0x30-0x34
        return al&0xff

def crack(o):
    for i2 in range(0x20,0xff):
        for i3 in range(0x20,0xff):
            for i1 in range(0x20,0xff):
                if(switch_calc((i1&0xfc)  >> 2)==o[0] and switch_calc(i3&0x3f)==o[3] and switch_calc(((
                    print chr(i1)+chr(i2)+chr(i3)

####
output = '45,49,42,55,56,4E,6F,4A,66,49,5D,73,5D,45,4E,41,48,7B,66,6B,48,62,75,35,40,37,69,42,43,69,43,
output = output.split(',')
# temp = 'UJUJUJUJ^A^A^A^A[d[d[d[dPoPoPoPo'
# output = []
# for x in temp:
    # output.append(hex(ord(x)).replace('0x',''))
####
input = ''
i = 0
j = 0
new_output = [0]*32
while(i<len(output)/2):
    new_output[j] = output[i]
    new_output[j+1] = output[i+0x10]
    i+=1
    j+=2
i = 0
output = new_output
while(i<len(output)/2):
    al = int(output[i],16) ^0xc
    ah = int(output[i+0x10],16) ^0x7
    output[i+0x10] = al
    output[i] = ah
    i+=1
i =0
while(i<len(output)):
    crack(output[i:i+4])
    i+=4
```

得到flag

```
hctf{Dd0g 1s 1066 d0gs!}
```

解题的过程：

https://github.com/Winter3un/pic/raw/master/writeup/hctf/%E8%A7%A3%E9%A2%98%E8%BF%87%E7%A8%8B.zip

题目打包地址（直接点击即可下载）：

https://github.com/Winter3un/pic/raw/master/writeup/hctf/%E9%A2%98%E7%9B%AE%E6%89%93%E5%8C%85.zip