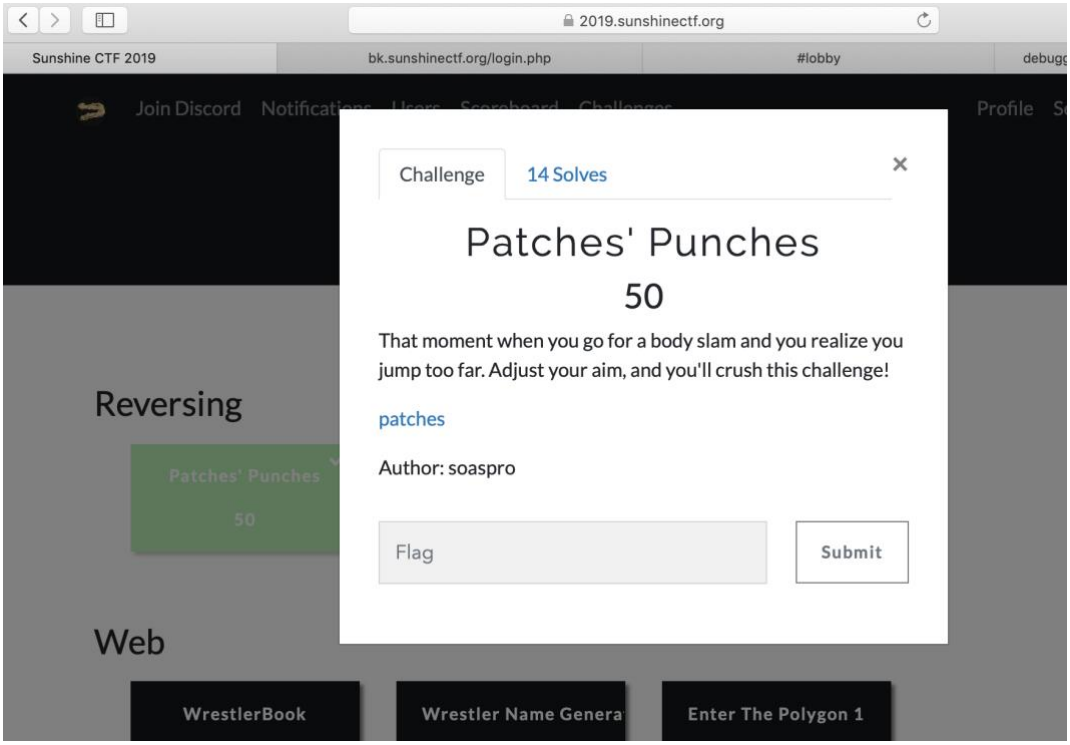
1. 다음과 같이 바이너리 파일이 제공된다.

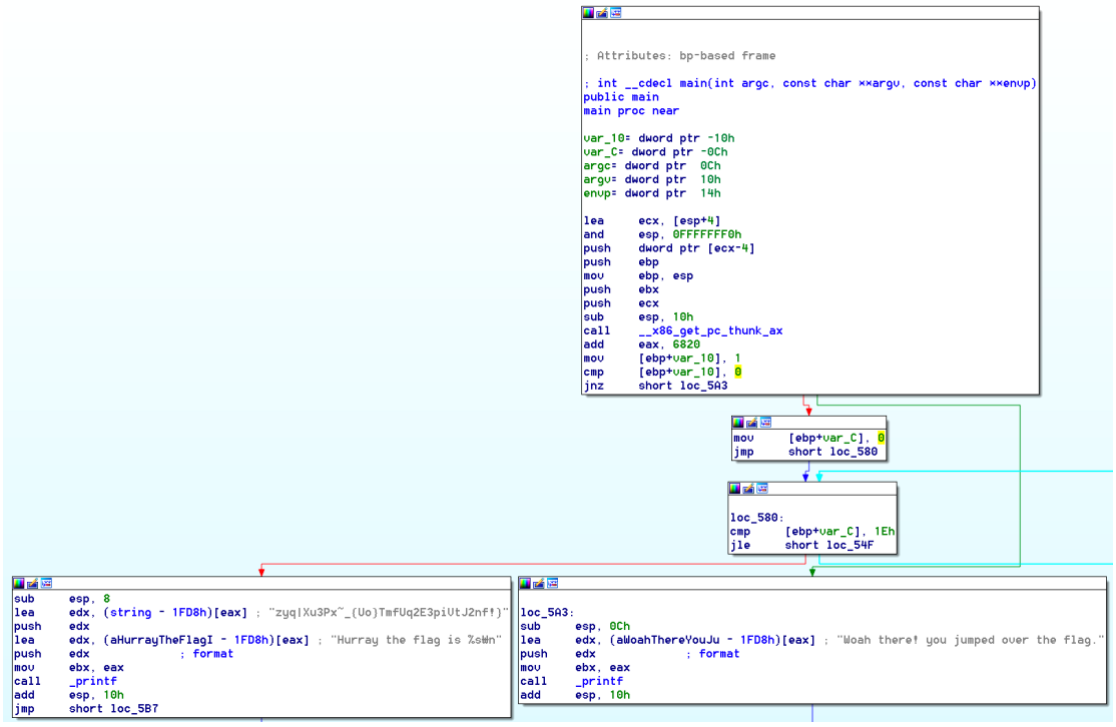



```
lhj@lhj-VirtualBox:~/바탕화면$ file patches.dms
patches.dms: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynami
cally linked, interpreter /lib/ld-, for GNU/Linux 3.2.0, BuildID[sha1]=4c73701f7
3d24817059a9878e13be44803352270, not stripped
```

2. ELF 32비트 파일인 것을 확인했으니까 분석을 해보자.

```
lhj@lhj-VirtualBox:~/바탕화면$ cat text
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
printf
__cxa_finalize
__libc_start_main
GLIBC_2.1.3
GLIBC_2.0
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
UWVS
[^_]
Hurray the flag is %s
Woah there! you jumped over the flag.
;*2$"
zyq|Xu3Px~_{Uo}TmfUq2E3piVtJ2nf!}
GCC: (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
```

- 플래그 같이 생겨서 뭔가 저걸 잘 이용하면 될 것 같았다.
- 혹시 모르니까 시저 암호를 돌렸는데, 매칭되는 부분을 찾을 수는 없었다.



- 조금 더 자세한 힌트를 찾아보기 위해 IDA를 이용하여 바이너리를 살펴보니 strings로 추출했던 스트링값과 동일한 문자열을 확인 가능했다.
- 분기문을 보면 mov [ebp+var_10], 1 로 인해 cmp가 틀린 값이 나와서 ZF가 0이 되어 jnz에서 종료되는 분기문으로 이동했다.
- 해결하기 위해 GDB를 사용하여 분기문을 통과시키면 flag값을 얻을 수 있을 것 같았다.

```
(gdb) b main
Breakpoint 1 at 0x52c
(gdb) r
Starting program: /home/lhj/바탕화면/patches.dms

Breakpoint 1, 0x5655552c in main ()
(gdb) set disas maQuit
(gdb) disas main
Dump of assembler code for function main:
   0x5655551d <+0>:     lea    ecx,[esp+0x4]
   0x56555521 <+4>:     and    esp,0xfffffff0
   0x56555524 <+7>:     push   DWORD PTR [ecx-0x4]
   0x56555527 <+10>:    push   ebp
   0x56555528 <+11>:    mov    ebp,esp
   0x5655552a <+13>:    push   ebx
   0x5655552b <+14>:    push   ecx
=> 0x5655552c <+15>:    sub    esp,0x10
   0x5655552f <+18>:    call   0x565555c6 <__x86.get_pc_thunk.ax>
   0x56555534 <+23>:    add    eax,0x1aa4
   0x56555539 <+28>:    mov    DWORD PTR [ebp-0x10],0x1
```

- main함수에 breakpoint를 걸어준다.

```
(gdb) disas main
Dump of assembler code for function main:
   0x5655551d <+0>:     lea    ecx,[esp+0x4]
   0x56555521 <+4>:     and    esp,0xfffffff0
   0x56555524 <+7>:     push   DWORD PTR [ecx-0x4]
   0x56555527 <+10>:    push   ebp
   0x56555528 <+11>:    mov    ebp,esp
   0x5655552a <+13>:    push   ebx
   0x5655552b <+14>:    push   ecx
   0x5655552c <+15>:    sub    esp,0x10
   0x5655552f <+18>:    call   0x565555c6 <__x86.get_pc_thunk.ax>
   0x56555534 <+23>:    add    eax,0x1aa4
   0x56555539 <+28>:    mov    DWORD PTR [ebp-0x10],0x1
   0x56555540 <+35>:    cmp    DWORD PTR [ebp-0x10],0x0
=> 0x56555544 <+39>:    jne    0x565555a3 <main+134>
   0x56555546 <+41>:    mov    DWORD PTR [ebp-0xc],0x0
   0x5655554d <+48>:    jmp    0x56555580 <main+99>
   0x5655554f <+50>:    lea    ecx,[eax+0xc8]
   0x56555555 <+56>:    mov    edx,DWORD PTR [ebp-0xc]
   0x56555558 <+59>:    add    edx,ecx
   0x5655555a <+61>:    movzx  edx,BYTE PTR [edx]
```

- IDA에서 보았던 jne에서 ZF를 1로 변환이 필요했다.

```
(gdb) set ($eflags)=0x243
(gdb) info reg
eax            0x56556fd8       1448439768
ecx            0xffffcf40       -12480
edx            0xffffcf64       -12444
ebx            0x0      0
esp            0xffffcf10       0xffffcf10
ebp            0xffffcf28       0xffffcf28
esi            0xf7fb1000       -134541312
edi            0x0      0
eip            0x56555544       0x56555544 <main+39>
eflags         0x243    [ CF ZF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
```

- set ($eflags)=0x243 으로 ZF를 1로 올려줄 수 있었다.

```
(gdb) c
Continuing.
Hurray the flag is sun{To0HotToHanDleTo0C0ldToH0ld!}
[Inferior 1 (process 21368) exited normally]
```

- 계속 실행하면 플래그가 확인된다.