使用Ubuntu 20.04，編譯時使用-fno-stack-protector 以及-z execstack，並關閉ASLR。

將main function 做 disassembly :

```
(gdb) disass main
Dump of assembler code for function main:
   0x0000555555555189 <+0>:     endbr64
   0x000055555555518d <+4>:     push   %rbp
   0x000055555555518e <+5>:     mov    %rsp,%rbp
   0x0000555555555191 <+8>:     sub    $0x10,%rsp
   0x0000555555555195 <+12>:    lea    0xe6c(%rip),%rdi        # 0x555555556008
   0x000055555555519c <+19>:    callq  0x555555555070 <puts@plt>
   0x00005555555551a1 <+24>:    callq  0x5555555551da <ValidatePassword>
   0x00005555555551a6 <+29>:    mov    %al,-0x1(%rbp)
   0x00005555555551a9 <+32>:    movzbl -0x1(%rbp),%eax
   0x00005555555551ad <+36>:    xor    $0x1,%eax
   0x00005555555551b0 <+39>:    test   %al,%al
   0x00005555555551b2 <+41>:    je     0x5555555551c7 <main+62>
   0x00005555555551b4 <+43>:    lea    0xe65(%rip),%rdi        # 0x555555556020
   0x00005555555551bb <+50>:    callq  0x555555555070 <puts@plt>
   0x00005555555551c0 <+55>:    mov    $0xffffffff,%eax
   0x00005555555551c5 <+60>:    jmp    0x5555555551d8 <main+79>
   0x00005555555551c7 <+62>:    lea    0xe71(%rip),%rdi        # 0x55555555603f
   0x00005555555551ce <+69>:    callq  0x555555555070 <puts@plt>
   0x00005555555551d3 <+74>:    mov    $0x0,%eax
   0x00005555555551d8 <+79>:    leaveq
--Type <RET> for more, q to quit, c to continue without paging--c
```

將ValidatePassword做 disassembly :

```
(gdb) disass ValidatePassword
Dump of assembler code for function ValidatePassword:
   0x00000000000011da <+0>:     endbr64
   0x00000000000011de <+4>:     push   %rbp
   0x00000000000011df <+5>:     mov    %rsp,%rbp
   0x00000000000011e2 <+8>:     sub    $0x20,%rsp
   0x00000000000011e6 <+12>:    lea    -0x20(%rbp),%rax
   0x00000000000011ea <+16>:    mov    %rax,%rdi
   0x00000000000011ed <+19>:    mov    $0x0,%eax
   0x00000000000011f2 <+24>:    callq  0x1090 <gets@plt>
   0x00000000000011f7 <+29>:    lea    -0x20(%rbp),%rax
   0x00000000000011fb <+33>:    lea    0xe58(%rip),%rsi        # 0x205a
   0x0000000000001202 <+40>:    mov    %rax,%rdi
   0x0000000000001205 <+43>:    callq  0x1080 <strcmp@plt>
   0x000000000000120a <+48>:    test   %eax,%eax
   0x000000000000120c <+50>:    jne    0x1215 <ValidatePassword+59>
   0x000000000000120e <+52>:    mov    $0x1,%eax
   0x0000000000001213 <+57>:    jmp    0x121a <ValidatePassword+64>
   0x0000000000001215 <+59>:    mov    $0x0,%eax
   0x000000000000121a <+64>:    leaveq
   0x000000000000121b <+65>:    retq
End of assembler dump.
(gdb)
```

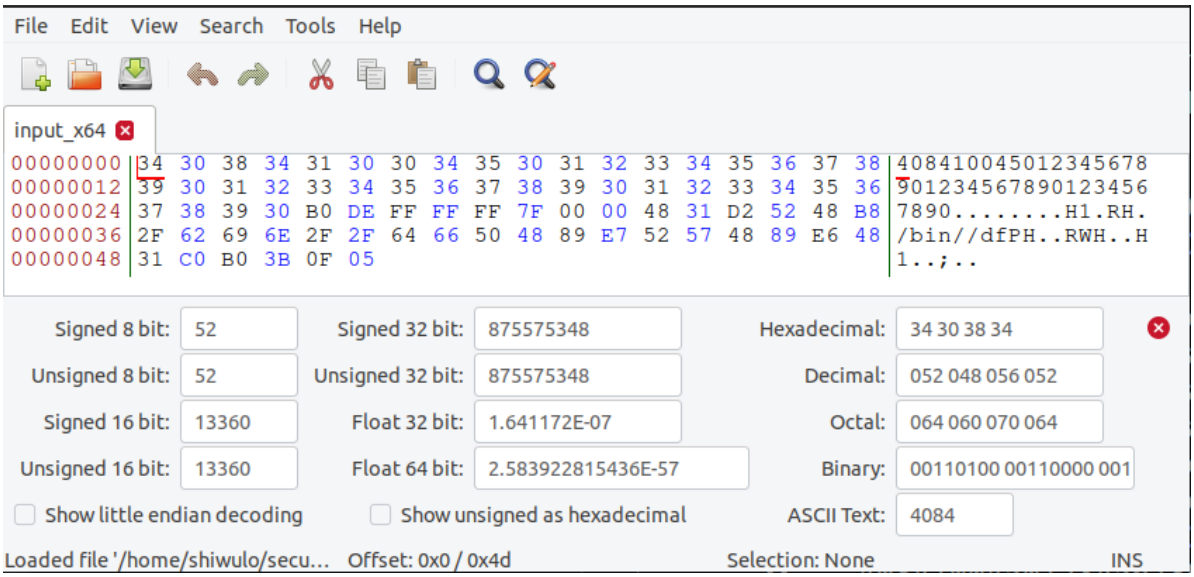
ValidatePassword這個function在執行階段的stack frame中的記憶體內容如下:

紅框為return address在記憶體中的所在地

```
(gdb) x/32wx $rsp
0x7fffffffde80: 0x34333231      0x38373635      0x32313039      0x36353433
0x7fffffffde90: 0x30393837      0x00005500      0xffffdfb0      0x00007fff
0x7fffffffdea0: 0xffffdec0      0x00007fff      0x555551a6      0x00005555
0x7fffffffdeb0: 0xffffdfb0      0x00007fff      0x00000000      0x00000000
0x7fffffffdec0: 0x00000000      0x00000000      0xf7de60b3      0x00007fff
0x7fffffffded0: 0x00000050      0x00000000      0xffffdfb8      0x00007fff
0x7fffffffdee0: 0xf7faa7a0      0x00000001      0x55555189      0x00005555
0x7fffffffdef0: 0x55555220      0x00005555      0xdacabc2e      0xd8bddca4
(gdb)
```

製作造成buffer overflow的input檔案的畫面：

新的return address 改成 0x7fffffffdeb0，shellcode從0x7fffffffdeb0這一個位址開始放



在gdb中執行shellcode: