

Latihan Soal Bahasa Level Mesin – Prosedur, Buffer overflow

1. Diberikan kode sebagai berikut:

```
#include <stdio.h>

int overflow(void);
int one = 1;

/* main - NukeJr's main routine */
int main() {
    int val = overflow();

    val += one;
    if (val != 15213)
        printf("Boom!\n");
    else
        printf("Curses! You've defused NukeJr!\n");
    _exit(0); /* syscall version of exit that doesn't need %ebp */
}

/* overflow - writes to stack buffer and returns 15213 */
int overflow() {
    char buf[4];
    int val, i=0;

    while(scanf("%x", &val) != EOF)
        buf[i++] = (char)val;
    return 15213;
}
```

Hasil kompilasinya pada Linux/x86 adalah sbb:

```
08048560 <main>:
08048560:    55                pushl   %ebp
08048561:    89 e5             movl    %esp,%ebp
08048563:    83 ec 08          subl    $0x8,%esp
08048566:    e8 31 00 00 00    call   804859c <overflow>
0804856b:    03 05 90 96 04    addl    0x8049690,%eax        # val += one;
08048570:    08
08048571:    3d 6d 3b 00 00    cmpl    $0x3b6d,%eax        # val == 15213?
08048576:    74 0a             je      8048582 <main+0x22>
08048578:    83 c4 f4          addl    $0xffffffff4,%esp
0804857b:    68 40 86 04 08    pushl   $0x8048640
08048580:    eb 08            jmp     804858a <main+0x2a>
08048582:    83 c4 f4          addl    $0xffffffff4,%esp
08048585:    68 60 86 04 08    pushl   $0x8048660
0804858a:    e8 75 fe ff ff    call   8048404 <_init+0x44>    # call printf
0804858f:    83 c4 10          addl    $0x10,%esp
08048592:    83 c4 f4          addl    $0xffffffff4,%esp
08048595:    6a 00            pushl   $0x0
08048597:    e8 b8 fe ff ff    call   8048454 <_init+0x94>    # call _exit

0804859c <overflow>:
0804859c:    55                pushl   %ebp
0804859d:    89 e5             movl    %esp,%ebp
0804859f:    83 ec 10          subl    $0x10,%esp
080485a2:    56                pushl   %esi
080485a3:    53                pushl   %ebx
080485a4:    31 f6             xorl    %esi,%esi
080485a6:    8d 5d f8          leal    0xffffffff8(%ebp),%ebx
```

```

80485a9:    eb 0d          jmp     80485b8 <overflow+0x1c>
80485ab:    90             nop
80485ac:    8d 74 26 00     leal    0x0(%esi,1),%esi
80485b0:    8a 45 f8        movb    0xffffffff8(%ebp),%al # L1: loop start
80485b3:    88 44 2e fc     movb    %al,0xffffffffc(%esi,%ebp,1)
80485b7:    46             incl    %esi
80485b8:    83 c4 f8        addl    $0xffffffff8,%esp
80485bb:    53             pushl   %ebx
80485bc:    68 80 86 04 08  pushl   $0x8048680
80485c1:    e8 6e fe ff ff  call    8048434 <_init+0x74> # call scanf
80485c6:    83 c4 10        addl    $0x10,%esp
80485c9:    83 f8 ff        cmpl    $0xffffffff,%eax
80485cc:    75 e2          jne     80485b0 <overflow+0x14> # goto L1
80485ce:    b8 6d 3b 00 00  movl    $0x3b6d,%eax
80485d3:    8d 65 e8        leal    0xffffffffe8(%ebp),%esp
80485d6:    5b             popl    %ebx
80485d7:    5e             popl    %esi
80485d8:    89 ec          movl    %ebp,%esp
80485da:    5d             popl    %ebp
80485db:    c3            ret

```

Berikut informasi tambahan yang diberikan:

- Mesin menggunakan Little Endian (least significant bytes has lowest address)
- Fungsi `scanf("%x", &val)` membaca sekuens karakter yang merepresentasikan hexa integer dari stdin, mengkonversi menjadi 32-bit int, dan menyimpan hasilnya pada val. Scnf mengembalikan 1 jika berhasil membaca nilai, dan EOF jika tidak ada nilai yang dibaca. Contoh: pemanggilan 4 kali `scanf` terhadap input string `"0 a ff"` akan menghasilkan berikut:
 - Pemanggilan `scanf` ke-1: `val = 0x0`, `scanf` mengembalikan 1
 - Pemanggilan `scanf` ke-2: `val = 0xa`, `scanf` mengembalikan 1
 - Pemanggilan `scanf` ke-3: `val = 0xff`, `scanf` mengembalikan 1
 - Pemanggilan `scanf` ke-4: `val = ?`, `scanf` mengembalikan EOF

A. Tuliskanlah alamat beberapa objek yang terdapat pada stack saat instruksi `subl` pada fungsi `overflow` (alamat `0x804859f`) selesai di-eksekusi sebagai offset dari `buf[0]`, dengan mengisi tabel berikut.

Stack object	Address of stack object
return address	&buf[0] + <u>8</u>
old %ebp	&buf[0] + <u>4</u>
buf[3]	&buf[0] + <u>3</u>
buf[2]	&buf[0] + <u>2</u>
buf[1]	&buf[0] + 1
buf[0]	&buf[0] + 0

- Tuliskan input string yang dapat mengakibatkan pemanggilan `overflow` kembali ke alamat `0x8048571` (bukan ke `0x804856b`)

`0 0 0 0 0 0 0 71 85 04 08`