

Language Based Security

Summer Semester 2006

1. Homework

03 May 2006

Exercise 1:

Consider the following C program. `strtol (s, NULL, 10)` returns the integer written as string `s` in decimal notation. Call the program with appropriate inputs to print `x=10`.

```
#include <stdio.h>
void f (int i, int j) {
    int a[50];
    a[i] += j;
}
int main (int argc, char *argv[]) {
    int x = 10, y, z;
    if (argc > 1) {
        y = strtol (argv[1], NULL, 10);
        z = strtol (argv[2], NULL, 10);
        f(y, z);
        x = 20;
        printf ("x=%d\n", x);
    }
    return 0;
}
```

Here is the output of `gdb` on the executable produced after compilation. (Or work with your own compiled version of the program.)

Dump of assembler code for function `main`:

```
0x8048396 <main>:      push   %ebp
0x8048397 <main+1>:     mov    %esp,%ebp
0x8048399 <main+3>:      sub   $0x18,%esp
0x804839c <main+6>:     and   $0xffffffff0,%esp
0x804839f <main+9>:     mov   $0x0,%eax
0x80483a4 <main+14>:    sub   %eax,%esp
0x80483a6 <main+16>:    movl  $0xa,0xffffffffc(%ebp)
0x80483ad <main+23>:   cmpl  $0x1,0x8(%ebp)
0x80483b1 <main+27>:   jle   0x8048412 <main+124>
```

```

0x80483b3 <main+29>:   sub    $0x4,%esp
0x80483b6 <main+32>:   push  $0xa
0x80483b8 <main+34>:   push  $0x0
0x80483ba <main+36>:   mov   0xc(%ebp),%eax
0x80483bd <main+39>:   add   $0x4,%eax
0x80483c0 <main+42>:   pushl (%eax)
0x80483c2 <main+44>:   call  0x804827c <strtol>
0x80483c7 <main+49>:   add   $0x10,%esp
0x80483ca <main+52>:   mov   %eax,0xffffffff8(%ebp)
0x80483cd <main+55>:   sub   $0x4,%esp
0x80483d0 <main+58>:   push  $0xa
0x80483d2 <main+60>:   push  $0x0
0x80483d4 <main+62>:   mov   0xc(%ebp),%eax
0x80483d7 <main+65>:   add   $0x8,%eax
0x80483da <main+68>:   pushl (%eax)
0x80483dc <main+70>:   call  0x804827c <strtol>
0x80483e1 <main+75>:   add   $0x10,%esp
0x80483e4 <main+78>:   mov   %eax,0xffffffff4(%ebp)
0x80483e7 <main+81>:   sub   $0x8,%esp
0x80483ea <main+84>:   pushl 0xffffffff4(%ebp)
0x80483ed <main+87>:   pushl 0xffffffff8(%ebp)
0x80483f0 <main+90>:   call  0x8048374 <f>
0x80483f5 <main+95>:   add   $0x10,%esp
0x80483f8 <main+98>:   movl  $0x14,0xffffffc(%ebp)
0x80483ff <main+105>:  sub   $0x8,%esp
0x8048402 <main+108>: pushl 0xffffffc(%ebp)
0x8048405 <main+111>: push  $0x80484e4
0x804840a <main+116>: call  0x804829c <printf>
0x804840f <main+121>: add   $0x10,%esp
0x8048412 <main+124>: mov   $0x0,%eax
0x8048417 <main+129>: leave
0x8048418 <main+130>: ret

```

End of assembler dump.

Dump of assembler code for function f:

```

0x8048374 <f>:      push  %ebp
0x8048375 <f+1>:     mov   %esp,%ebp
0x8048377 <f+3>:     sub   $0xd8,%esp
0x804837d <f+9>:     mov   0x8(%ebp),%ecx
0x8048380 <f+12>:    mov   0x8(%ebp),%edx
0x8048383 <f+15>:    mov   0xc(%ebp),%eax
0x8048386 <f+18>:    add   0xffffffff28(%ebp,%edx,4),%eax
0x804838d <f+25>:    mov   %eax,0xffffffff28(%ebp,%ecx,4)
0x8048394 <f+32>:    leave
0x8048395 <f+33>:    ret

```

End of assembler dump.