15213 Recitation Section C

Shimin Chen

Sept. 30, 2002

Outline

- Buffer overflow
- Putting code onto stack

Example 1: Buffer Overflow

Please draw the stack frame of “example1”. What are the values of n and x at the marked points?

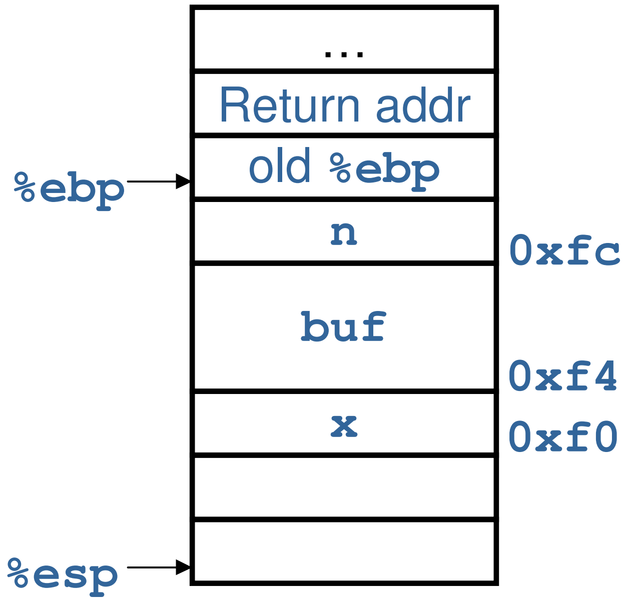
```
void example1()
{
  volatile int n;
  char buf[8];
  volatile int x;

  n = 0x12345678;  x = 0xdeadbeef;          1. n=? x=?
  strcpy(buf, "abcdefghijk");               2. n=? x=?
  // a=0x61 b=0x62 ...
  buf[8] = 0xab;
  buf[-4] = 0xcd;                           3. n=? x=?
}
```

ASM of example1

```
0x80483f0    push   %ebp
0x80483f1    mov    %esp,%ebp
0x80483f3    sub    $0x18,%esp
0x80483f6    movl   $0x12345678,0xfffffffc(%ebp)
0x80483fd    movl   $0xdeadbeef,0xfffffff0(%ebp)
0x8048404    add    $0xfffffff8,%esp
0x8048407    push   $0x80484a8
0x804840c    lea    0xfffffff4(%ebp),%eax
0x804840f    push   %eax
0x8048410    call   0x8048308 <strcpy>
0x8048415    add    $0x10,%esp
0x8048418    movb   $0xab,0xfffffffc(%ebp)
0x804841c    mov    $0xfffffffc,%eax
0x8048421    lea    0xfffffff4(%ebp),%edx
0x8048424    movb   $0xcd,(%eax,%edx,1)
0x8048428    mov    %ebp,%esp
0x804842a    pop    %ebp
0x804842b    ret
```

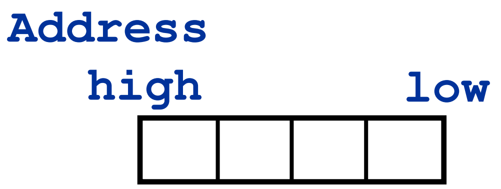
Stack Frame

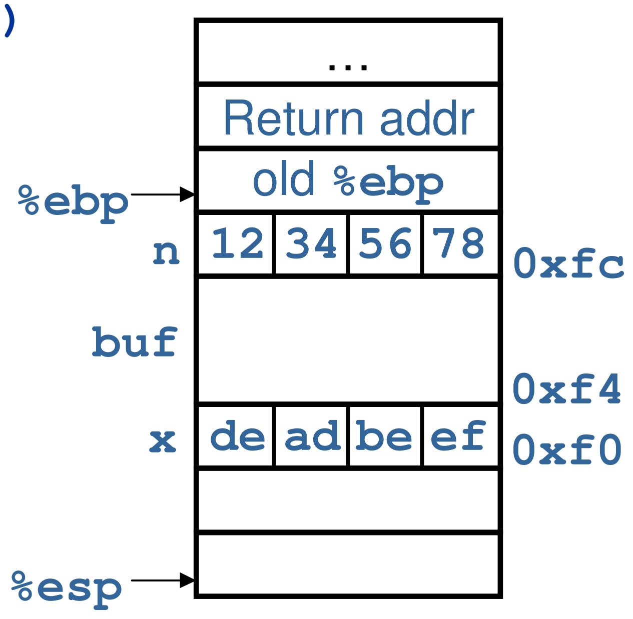
```
push    %ebp
mov     %esp,%ebp
sub     $0x18,%esp
movl    $0x12345678,0xfffffffc(%ebp)
movl    $0xdeadbeef,0xfffffff0(%ebp)
add     $0xfffffff8,%esp
push    $0x80484a8
lea     0xfffffff4(%ebp),%eax
push    %eax
call    0x8048308 <strcpy>
add     $0x10,%esp
movb    $0xab,0xfffffffc(%ebp)
mov     $0xfffffffc,%eax
lea     0xfffffff4(%ebp),%edx
movb    $0xcd,(%eax,%edx,1)
mov     %ebp,%esp
pop     %ebp
ret
```



Before Calling strcpy()

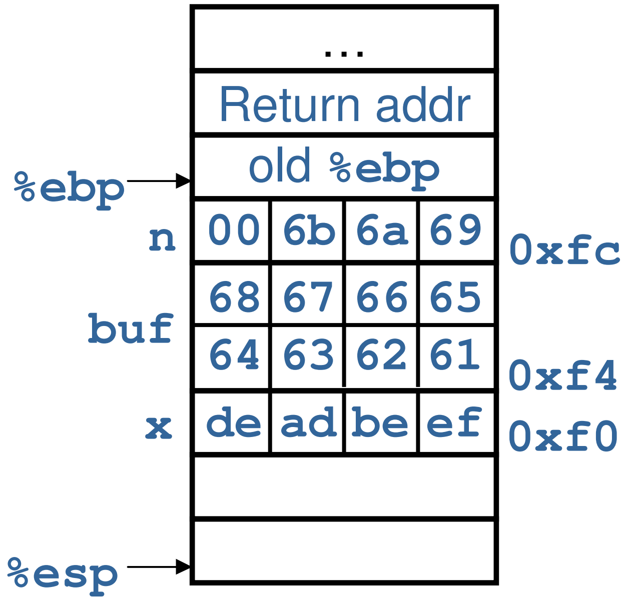
```
push    %ebp
mov     %esp,%ebp
sub     $0x18,%esp
movl    $0x12345678,0xfffffffc(%ebp)
movl    $0xdeadbeef,0xfffffff0(%ebp)
```





After Calling strcpy()

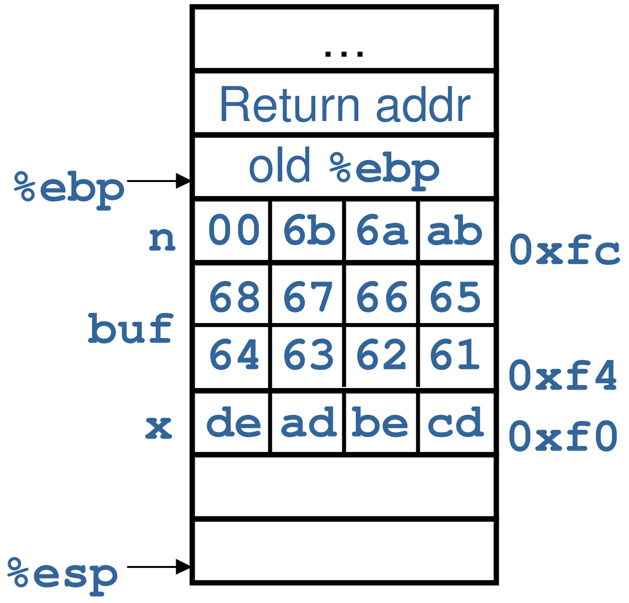
```
push    %ebp
mov     %esp,%ebp
sub     $0x18,%esp
movl    $0x12345678,0xfffffffc(%ebp)
movl    $0xdeadbeef,0xfffffff0(%ebp)
add     $0xfffffff8,%esp
push    $0x80484a8
lea     0xfffffff4(%ebp),%eax
push    %eax
call    0x8048308 <strcpy>
add     $0x10,%esp
```



```
Strcpy (buf, "abcdefghijk”);
```

Before Return

```
push    %ebp
mov     %esp,%ebp
sub     $0x18,%esp
movl    $0x12345678,0xfffffffc(%ebp)
movl    $0xdeadbeef,0xfffffff0(%ebp)
add     $0xfffffff8,%esp
push    $0x80484a8
lea     0xfffffff4(%ebp),%eax
push    %eax
call    0x8048308 <strcpy>
add     $0x10,%esp
movb    $0xab,0xfffffffc(%ebp)
mov     $0xfffffffc,%eax
lea     0xfffffff4(%ebp),%edx
movb    $0xcd,(%eax,%edx,1)
```



What If …

- What if we instead

  $$\texttt{strcpy(buf, "}\underbrace{\texttt{abcdefghijklmn"}}_{\textit{14+1 chars}}\texttt{);}$$

  - Old ebp is overwritten

- What if we instead

  $$\texttt{strcpy(buf, "}\underbrace{\texttt{abcdefghijklmnopq"}}_{\textit{17+1 chars}}\texttt{);}$$

  - Return addr is overwritten

Example 2: How to Put Code onto Stack?

```
int example2 ()
{
   char buf[8];
   gets (buf);
   return 0;
}
```

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
add    $0xfffffff4,%esp
lea    0xfffffff8(%ebp),%eax
push   %eax
call   0x80482e8 <gets>
xor    %eax,%eax
mov    %ebp,%esp
pop    %ebp
ret
```

Steps

1. Write assembly code
2. Get binary representation of the code
3. Generate ASCII for the binary code
4. Run the program with the input

Write assembly code

- Use your favorite text editor
- For example,

```
movl $0, -8(%ebp)
addl $0x12345678, %eax
```

- Save as `*.s, e.g. input.s`

Get binary representation of the code

- Compile the assembly with gcc

```
gcc -c input.s
```

- Display binary representation with objdump:

```
objdump -d input.o
```

- Copy the byte code into a text file

Generate ASCII for the binary code

- Use sendstring to generate ASCII string:

```
sendstring < input.txt > input.raw
```

Run the program with the input

- Run at the command line:

```
example2 < input.raw
```

- Run in gdb:

```
gdb example2
run < input.raw
```

Show Code on the Stack

```
(gdb) break example2
(gdb) break *0x80483f6
(gdb) run < input.raw
(gdb) p/x $ebp - 8
(gdb) p/x $ebp + 3
(gdb) continue
(gdb) disas 0xbffffa40 0xbffffa4b
```

Important Dates

- Lab 3: due Monday (Oct. 7), 11:59pm

- Exam 1: Tuesday (Oct. 8), 6:00–7:30pm
  Doherty Hall 2315