Recitation 4: The Stack & Lab3

Andrew Faulring

15213 Section A

30 September 2002

Andrew Faulring

- faulring@cs.cmu.edu
- Office hours:
  - NSH 2504 (lab) / 2507 (conference room)
  - Thursday 5-6

- Lab 3: due Monday (7 Oct), 11:59pm
- Exam 1: Tuesday (8 Oct), 6:00-7:30pm
  Doherty Hall 2315

Today's Plan

- Practical skills for Lab 3
  - Out of bounds array access
  - Mechanics of putting *your* code onto the stack

Local Variables

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

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```

%ebp - 24

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
movl   $0x2,0xfffffffc(%ebp)
movl   $0xdeadbeef,0xfffffff0(%ebp)
add    $0xfffffff8,%esp
push   $0x80484a8
lea    0xfffffff4(%ebp),%eax
push   %eax
call   0x8048308 <strcpy>
add    $0x10,%esp
movb   $0x6c,0xfffffffc(%ebp)
mov    $0xfffffffc,%eax
lea    0xfffffff4(%ebp),%edx
movb   $0xa8,(%eax,%edx,1)
mov    %ebp,%esp
pop    %ebp
ret
```

Local Variables

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

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```

%ebp - 4

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
movl   $0x2,0xfffffffc(%ebp)
movl   $0xdeadbeef,0xfffffff0(%ebp)
add    $0xfffffff8,%esp
push   $0x80484a8
lea    0xfffffff4(%ebp),%eax
push   %eax
call   0x8048308 <strcpy>
add    $0x10,%esp
movb   $0x6c,0xfffffffc(%ebp)
mov    $0xfffffffc,%eax
lea    0xfffffff4(%ebp),%edx
movb   $0xa8,(%eax,%edx,1)
mov    %ebp,%esp
pop    %ebp
ret
```

Local Variables

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

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```

%ebp - 16

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
movl   $0x2,0xfffffffc(%ebp)
movl   $0xdeadbeef,0xfffffff0(%ebp)
add    $0xfffffff8,%esp
push   $0x80484a8
lea    0xfffffff4(%ebp),%eax
push   %eax
call   0x8048308 <strcpy>
add    $0x10,%esp
movb   $0x6c,0xfffffffc(%ebp)
mov    $0xfffffffc,%eax
lea    0xfffffff4(%ebp),%edx
movb   $0xa8,(%eax,%edx,1)
mov    %ebp,%esp
pop    %ebp
ret
```

Local Variables

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

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```

%ebp – 32

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
movl   $0x2,0xfffffffc(%ebp)
movl   $0xdeadbeef,0xfffffff0(%ebp)
add    $0xfffffff8,%esp
push   $0x8048488
lea    0xfffffff4(%ebp),%eax
push   %eax
call   0x8048308 <strcpy>
add    $0x10,%esp
movb   $0x6c,0xfffffffc(%ebp)
mov    $0xfffffffc,%eax
lea    0xfffffff4(%ebp),%edx
movb   $0xa8,(%eax,%edx,1)
mov    %ebp,%esp
pop    %ebp
ret
```

Local Variables

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

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```

%ebp – 12, allocated for buf

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
movl   $0x2,0xfffffffc(%ebp)
movl   $0xdeadbeef,0xfffffff0(%ebp)
add    $0xfffffff8,%esp
push   $0x80484a8
lea    0xfffffff4(%ebp),%eax
push   %eax
call   0x8048308 <strcpy>
add    $0x10,%esp
movb   $0x6c,0xfffffffc(%ebp)
mov    $0xfffffffc,%eax
lea    0xfffffff4(%ebp),%edx
movb   $0xa8,(%eax,%edx,1)
mov    %ebp,%esp
pop    %ebp
ret
```

Local Variables

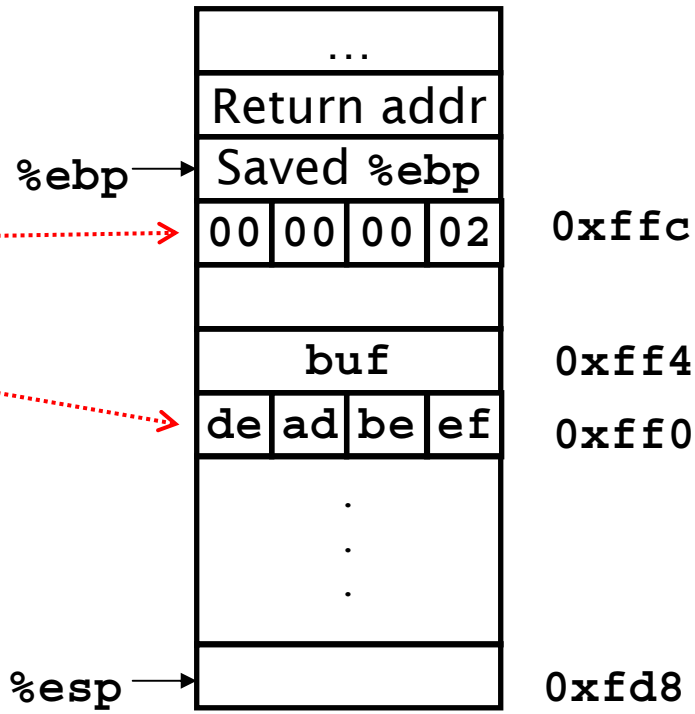
```
void localvars()
{
  volatile int n;
  char buf[8];
  volatile int x;

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```



So what's happening after strcpy?

Local Variables

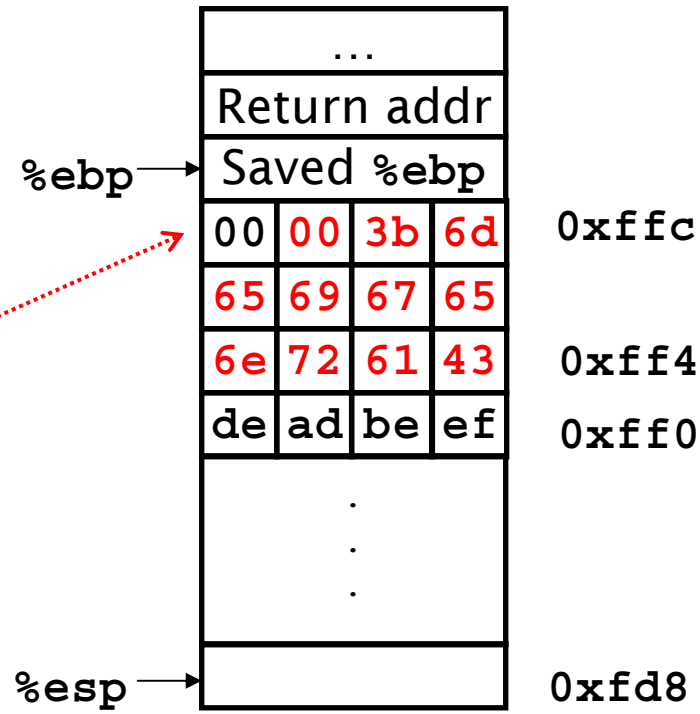
```
void localvars()
{
  volatile int n;
  char buf[8];
  volatile int x;

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```



Local Variables

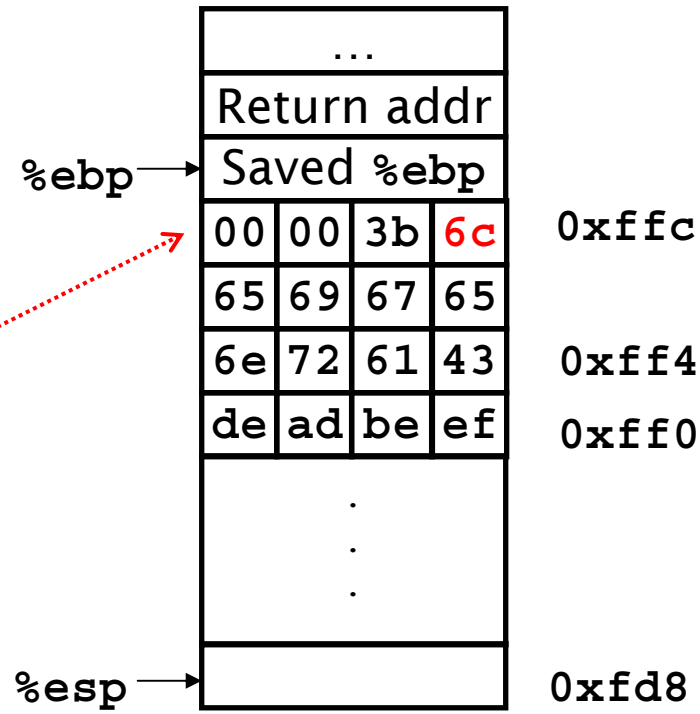
```
void localvars()
{
  volatile int n;
  char buf[8];
  volatile int x;

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```



Local Variables

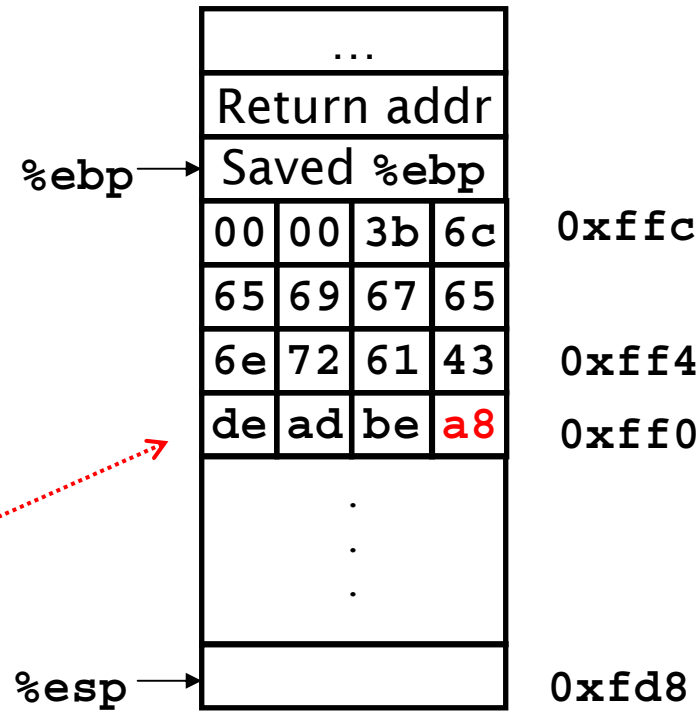
```
void localvars()
{
  volatile int n;
  char buf[8];
  volatile int x;

  n = 2;
  x = 0xdeadbeef;

  strcpy(buf, "Carnegiem;");
  // 'm' = 0x6d, ';' = 0x3b
  // n = 15213 (0x3b6d)

  buf[8] = 0x6c;
  // n = 15212

  buf[-4] = 0xa8;
  // x = 0xdeadbea8
}
```



Buffer Overflow

```
int bufoverflow(
   char* string, int n)
{
  char buf[8];
  strcpy(buf, string);
  return n;
}
```

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
mov    0x8(%ebp),%eax
add    $0xfffffff8,%esp
push   %eax
lea    0xfffffff0(%ebp),%eax
push   %eax
call   0x804833c <strcpy>
mov    0xc(%ebp),%eax
mov    %ebp,%esp
pop    %ebp
ret
```

Your Exploit Code

```
int abs_shift(int n) {
  return (n>=0 ? n : -n) << 2;
}
```

```
        movl 8(%ebp),%eax
        testl %eax,%eax
        jge .L1
        negl %eax
.L1:
        sall $2,%eax
        .long 0x00000000
```

Putting exploit code onto the stack (1)

```
unix> gcc -c exploit.s
unix> objdump -d exploit.o
00000000 <.text>:
   0:   8b 45 08                mov    0x8(%ebp),%eax
   3:   85 c0                   test   %eax,%eax
   5:   7d 02                   jge    0x9
   7:   f7 d8                   neg    %eax
   9:   c1 e0 02                shl    $0x2,%eax
   c:   00 00                   add    %al,(%eax)
unix> cat > exploit.txt
8b 45 08 85 c0 7d 02 f7 d8 c1 e0 02
unix> sendstring < exploit.txt > exploit.raw
unix> od -t x1 exploit.raw
0000000 8b 45 08 85 c0 7d 02 f7 d8 c1 e0 02 0a
```

Putting exploit code onto the stack (2)

```
unix> gdb bufoverflow
(gdb) break bufoverflow
(gdb) run < exploit.raw
(gdb) x/4w $ebp-16
(gdb) nexti 6
(gdb) x/4w $ebp-16
(gdb) disas 0xbffff7c8 0xbffff7d4
```