Buffer Overflows

Many of the following slides are based on those from

Complete Powerpoint Lecture Notes for
Computer Systems: A Programmer's Perspective (CS:APP)

Randal E. Bryant and David R. O'Hallaron

http://csapp.cs.cmu.edu/public/lectures.html

The book is used explicitly in CS 2505 and CS 3214 and as a reference in CS 2506.

©2005-2013 McQuain

Buffer Overflows

What is a buffer overflow?

How can it be exploited?

How can it be avoided?

- Through programmer measures
- Through system measures (and how effective are they?)

©2005-2013 McQuain

String Library Code

Implementation of Unix function **gets**

No way to specify limit on number of characters to read

```
/* Get string from stdin */
char *gets(char *dest)
{
    int c = getc();
    char *p = dest;
    while (c != EOF && c != '\n') {
        *p++ = c;
        c = getc();
    }
    *p = '\0';
    return dest;
}
```


©2005-2013 McQuain

Vulnerable Buffer Code


```
int main()
{
  printf("Type a string:");
  echo();
  return 0;
}
```

```
/* Echo Line */
void echo()
{
    char buf[4];  /* Way too small! */
    gets(buf);
    puts(buf);
}
```


©2005-2013 McQuain

Buffer Overflow Executions

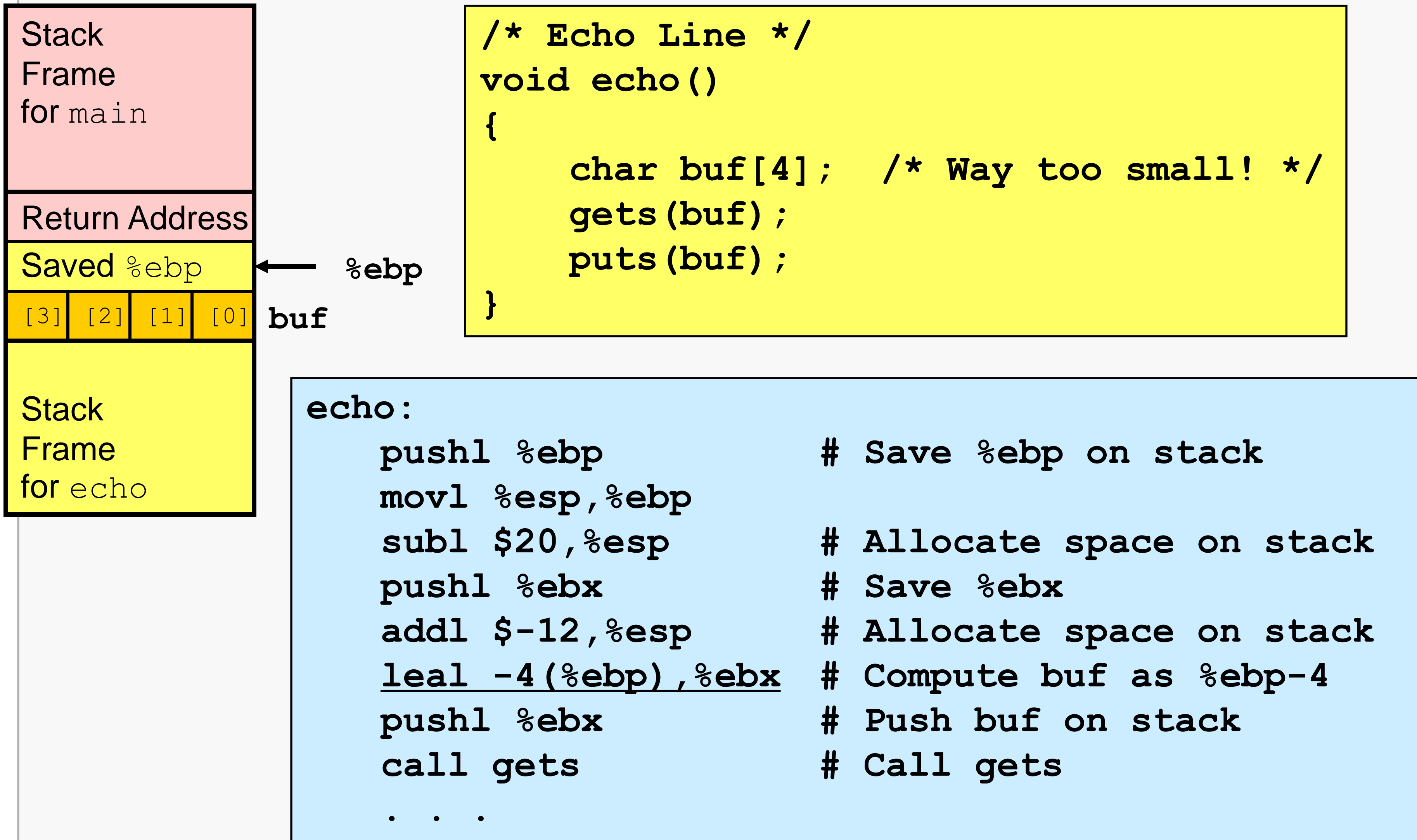
```
unix>./bufdemo
Type a string:123
123
```

```
unix>./bufdemo
Type a string:12345
Segmentation Fault
```

```
unix>./bufdemo
Type a string:12345678
Segmentation Fault
```


©2005-2013 McQuain

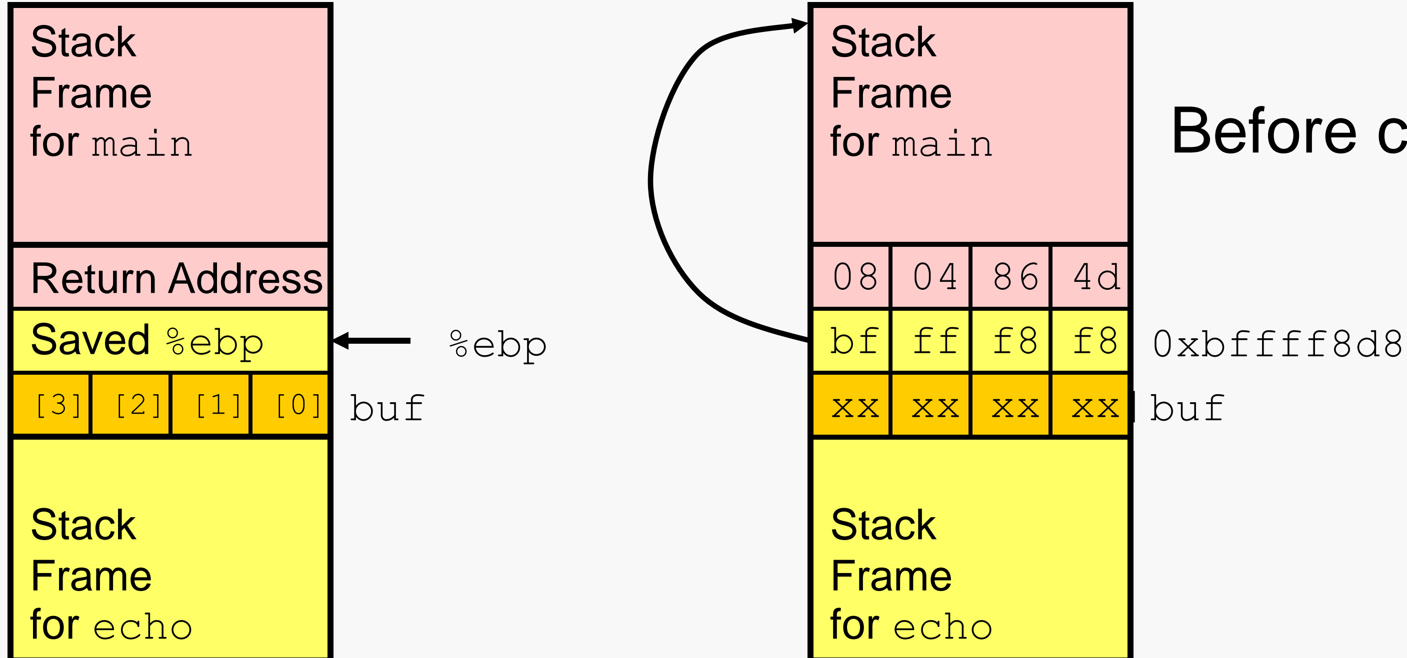
Buffer Overflow Stack



©2005-2013 McQuain

Buffer Overflow Stack Example

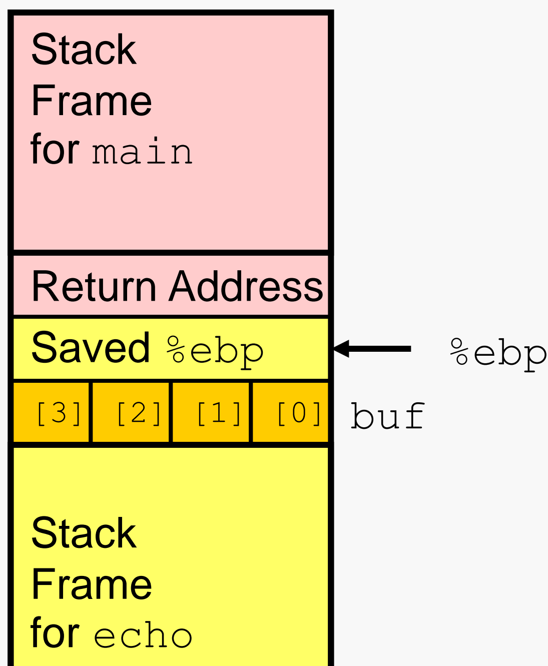
```
unix> gdb bufdemo
(gdb) break echo
Breakpoint 1 at 0x8048583
(gdb) run
Breakpoint 1, 0x8048583 in echo ()
(gdb) print /x *(unsigned *)$ebp
$1 = 0xbffff8f8
(gdb) print /x *((unsigned *)$ebp + 1)
$3 = 0x804864d
```

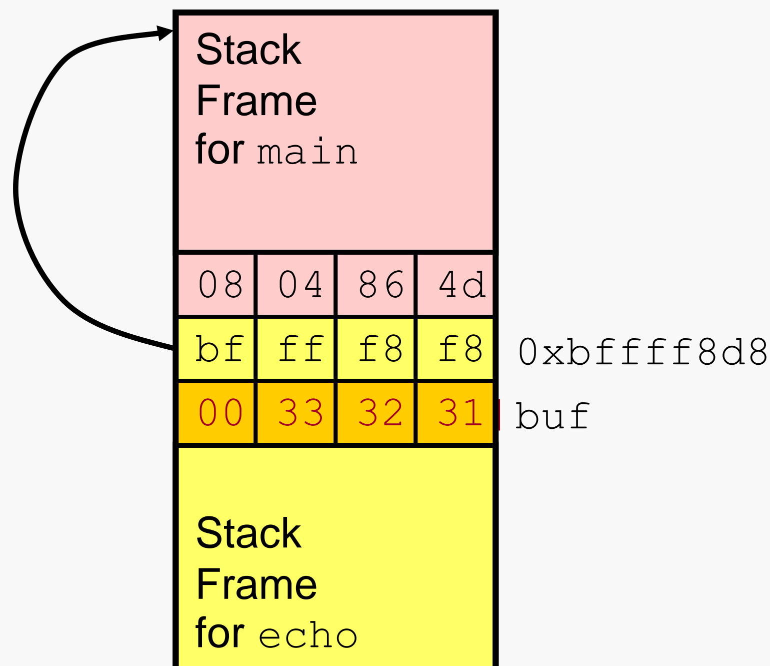


Before call to gets

```
8048648:  call  804857c <echo>
804864d:  mov   0xffffffe8(%ebp),%ebx # Return Point
```

Buffer Overflow Example #1

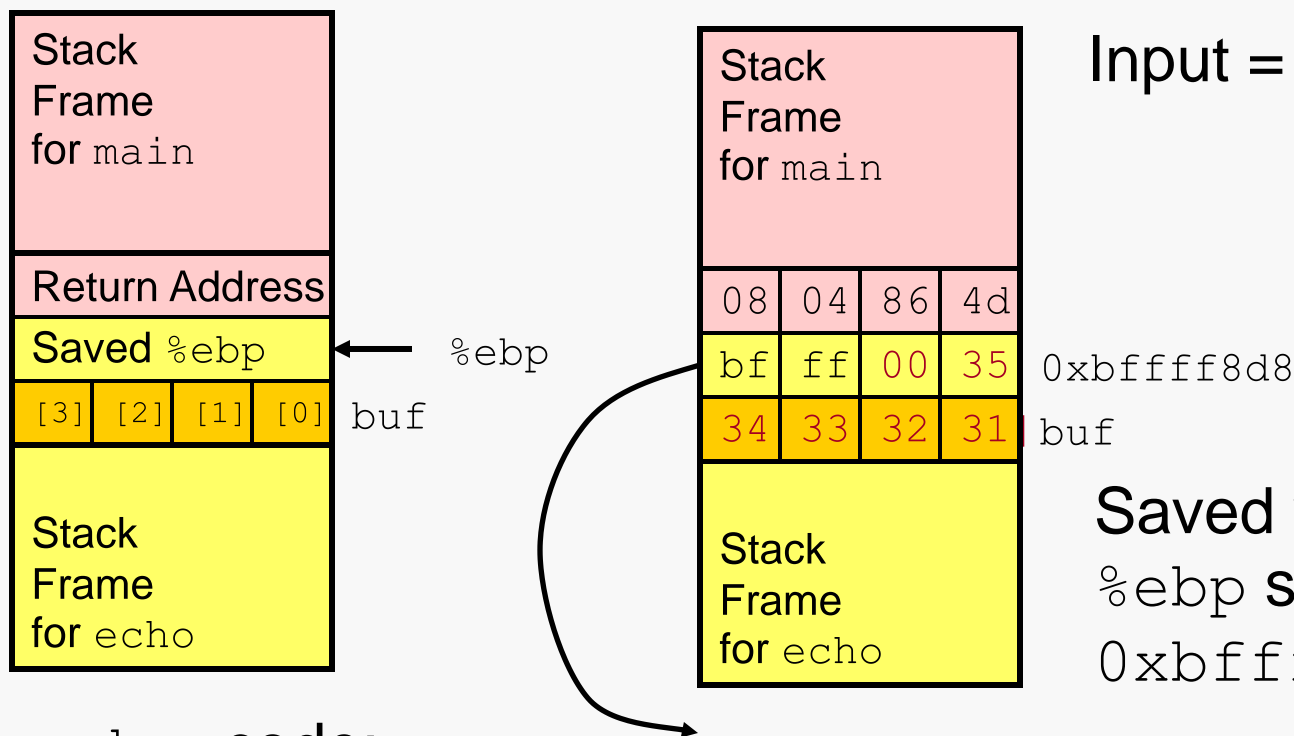




No Problem

©2005-2013 McQuain

Buffer Overflow Stack Example #2



Input = “12345”

Saved value of %ebp set to 0xbfff0035

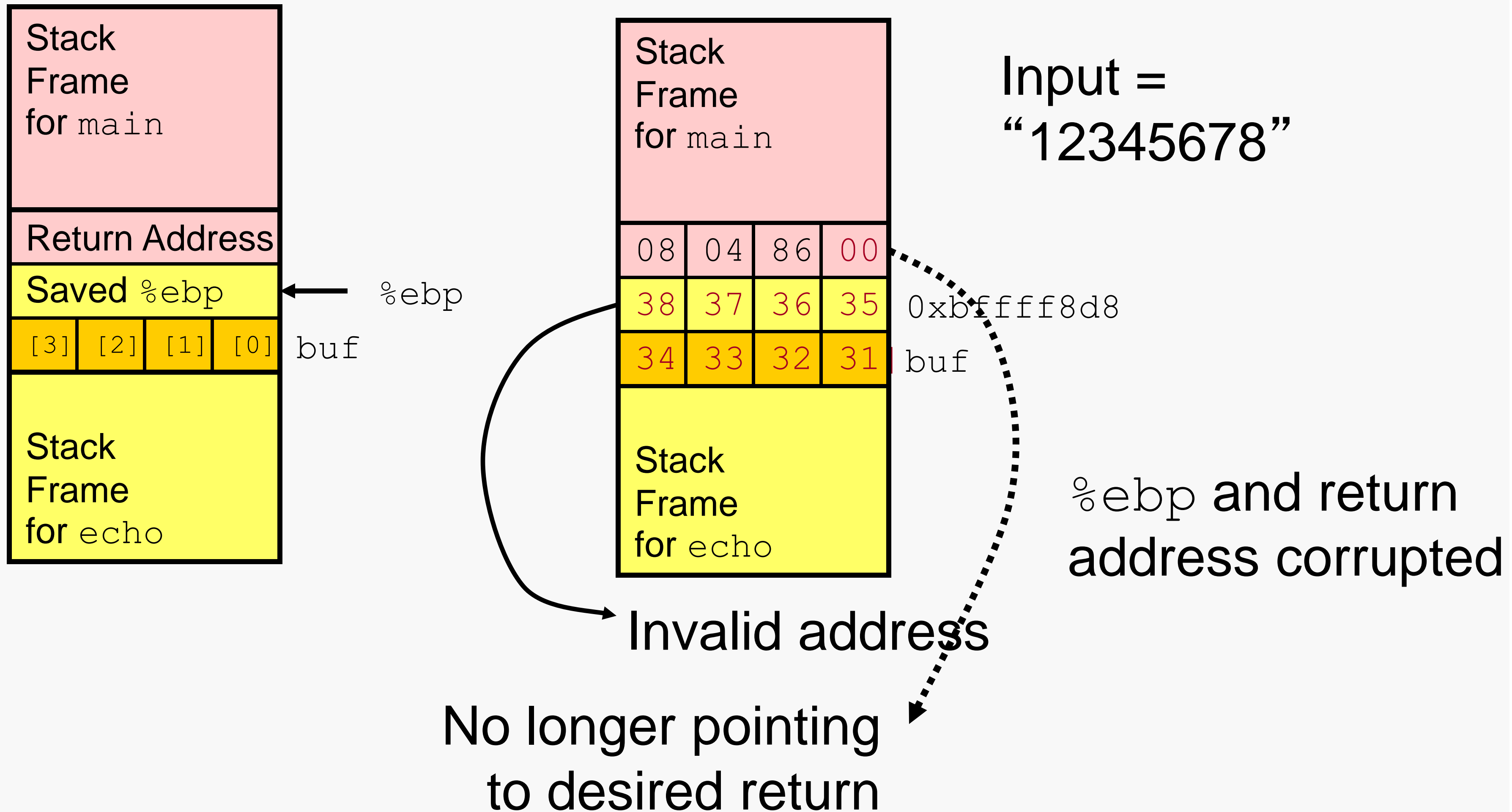
echo code:

```
8048592:  push   %ebx
8048593:  call   80483e4 <_init+0x50>  # gets
8048598:  mov    0xffffffe8(%ebp),%ebx
804859b:  mov    %ebp,%esp
804859d:  pop    %ebp      # %ebp gets set to invalid value
804859e:  ret
```

Bad news when later attempt to restore %ebp

©2005-2013 McQuain

Buffer Overflow Stack Example #3




```
8048648:  call  804857c <echo>
804864d:  mov   0xffffffe8(%ebp),%ebx # Return Point
```

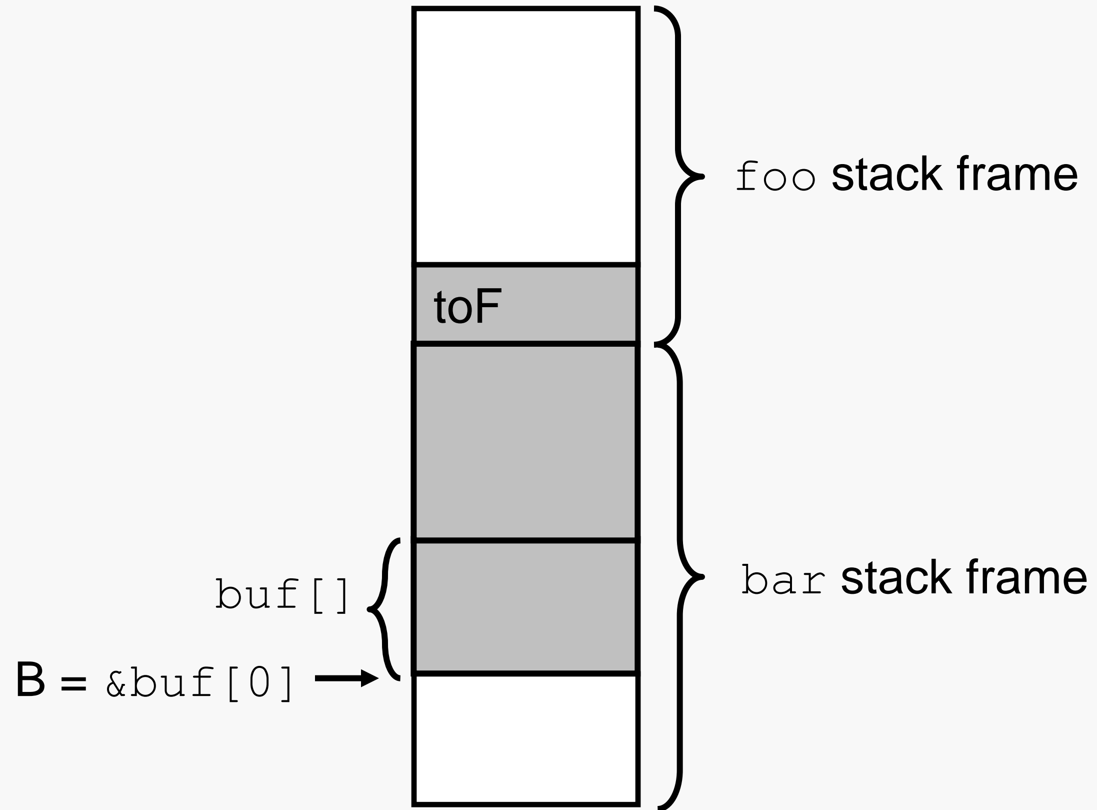

©2005-2013 McQuain

Malicious Use of Buffer Overflow

return address toF

```
void foo(){
   bar();
   ...
}
```

```
void bar() {
   char buf[64];
     ...
}
```



©2005-2013 McQuain

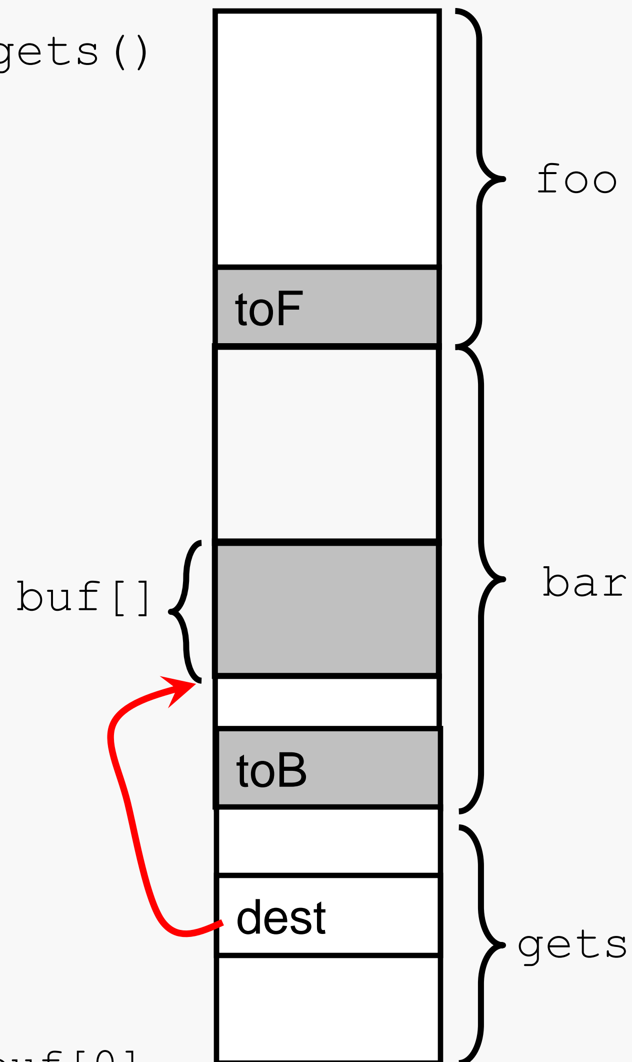
Malicious Use of Buffer Overflow


```
void foo(){
  bar();
  ...
}
```

return address toB →

```
void bar() {
  char buf[64];
  gets(buf);
  ...
}
```

Stack right after call to `gets()`



`gets()` can now write whatever it wants, beginning at `buf[0]`

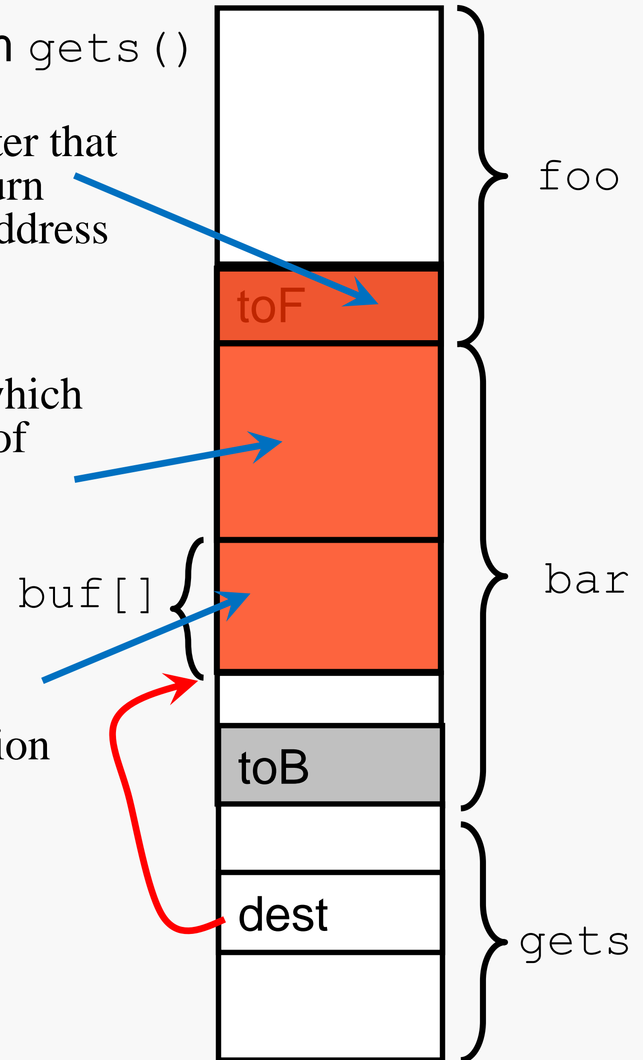
Malicious Use of Buffer Overflow

```
void foo(){
   bar();
   ...
}
```

```
void bar() {
   char buf[64];
   gets(buf);
   ...
}
```

return address toB

Stack right before return from `gets()`



And new pointer that overwrites return address with address of buffer

And padding (which overwrites rest of bar()'s frame

Input string to `gets()` contains byte representation of executable code

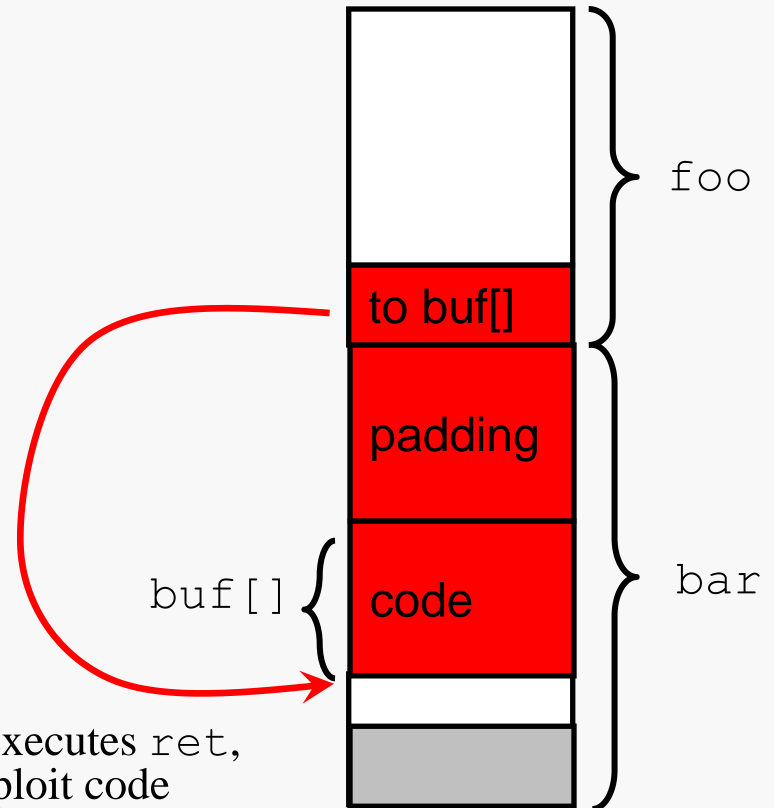
©2005-2013 McQuain

Malicious Use of Buffer Overflow

```
void foo(){
   bar();
   ...
}
```

```
void bar() {
   char buf[64];
   gets(buf);
   ...
}
```

Stack right after return to `bar()` from `gets()`



When `bar()` executes `ret`, will jump to exploit code

©2005-2013 McQuain

Avoiding Overflow Vulnerability

```
/* Echo Line */
void echo()
{
    char buf[4];  /* Way too small! */
    fgets(buf, 4, stdin);
    puts(buf);
}
```

Use Library Routines that check/limit string lengths

- **fgets** instead of **gets**
- **strncpy/strlcpy** instead of **strcpy**
- **snprintf** instead of **sprintf**
- Don't use **scanf** with **%s** conversion specification
  - Use **fgets** to read the string

http://lwn.net/Articles/507319/

©2005-2013 McQuain