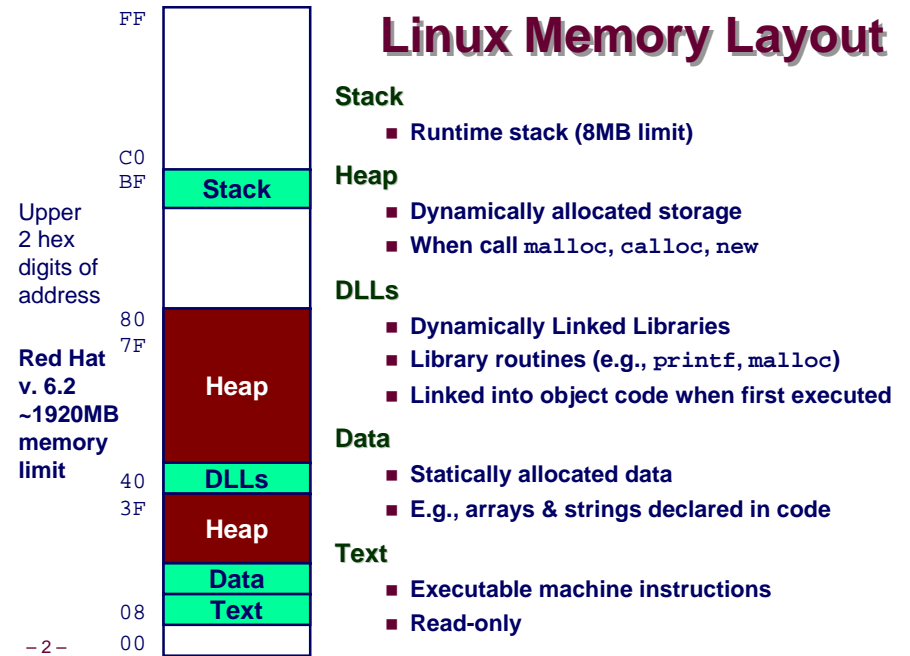


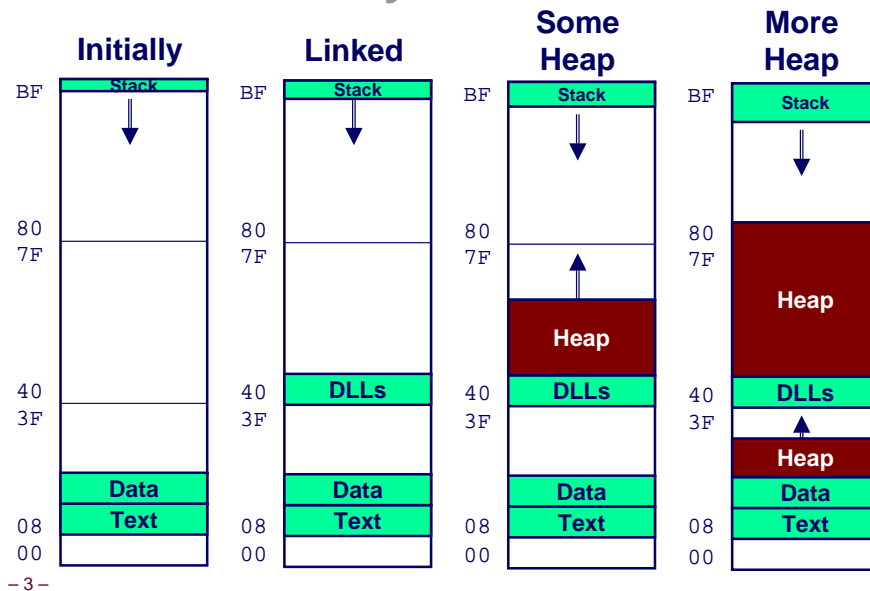
# Machine-Level Programming: Miscellaneous Topics From CS:APP by Bryant & O'Hallaron

## Topics

- Linux Memory Layout
- Buffer Overflow



## Linux Memory Allocation



## Text & Stack Example

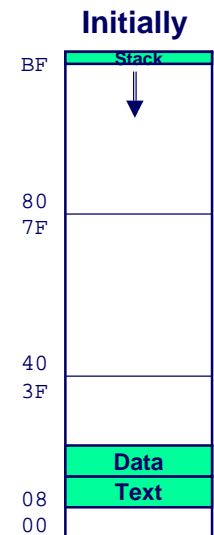
```
(gdb) break main
(gdb) run
Breakpoint 1, 0x804856f in main ()
(gdb) print $esp
$3 = (void *) 0xbffffc78
```

### Main

- Address `0x804856f` should be read `0x0804856f`

### Stack

- Address `0xbffffc78`



## Dynamic Linking Example

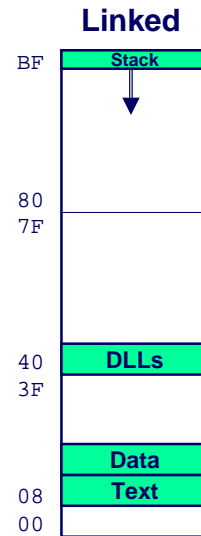
```
(gdb) print malloc
$1 = {<text variable, no debug info>
0x8048454 <malloc>}
(gdb) run
Program exited normally.
(gdb) print malloc
$2 = {void *(unsigned int)}
0x40006240 <malloc>}
```

### Initially

- Code in text segment that invokes dynamic linker
- Address 0x8048454 should be read 0x8048454

### Final

- Code in DLL region



## Memory Allocation Example

```
char big_array[1<<24]; /* 16 MB */
char huge_array[1<<28]; /* 256 MB */

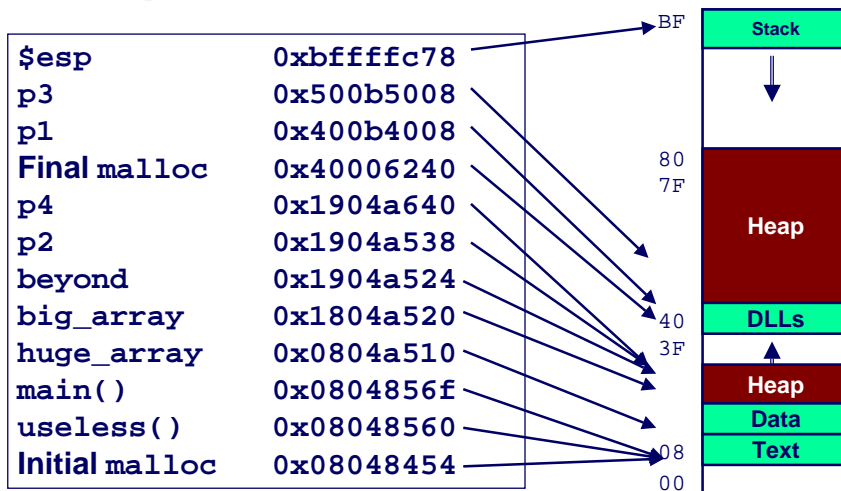
int beyond;
char *p1, *p2, *p3, *p4;

int useless() { return 0; }

int main()
{
    p1 = malloc(1 <<28); /* 256 MB */
    p2 = malloc(1 << 8); /* 256 B */
    p3 = malloc(1 <<28); /* 256 MB */
    p4 = malloc(1 << 8); /* 256 B */
    /* Some print statements ... */
}
```

-6-

## Example Addresses



-7-

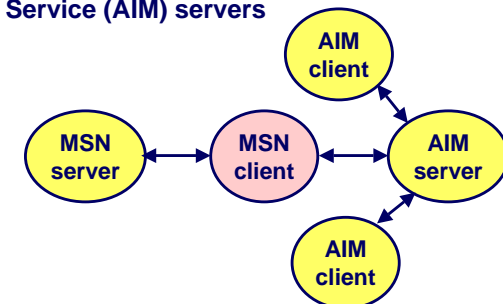
## Internet Worm and IM War

### November, 1988

- Internet Worm attacks thousands of Internet hosts.
- How did it happen?

### July, 1999

- Microsoft launches MSN Messenger (instant messaging system).
- Messenger clients can access popular AOL Instant Messaging Service (AIM) servers



-8-

## Internet Worm and IM War (cont.)

### August 1999

- Mysteriously, Messenger clients can no longer access AIM servers.
- Microsoft and AOL begin the IM war:
  - AOL changes server to disallow Messenger clients
  - Microsoft makes changes to clients to defeat AOL changes.
  - At least 13 such skirmishes.
- How did it happen?

### The Internet Worm and AOL/Microsoft War were both based on *stack buffer overflow* exploits!

- many Unix functions do not check argument sizes.
- allows target buffers to overflow.

- 9 -

## 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;
}
```

- Similar problems with other Unix functions
  - strcpy: Copies string of arbitrary length
  - scanf, fscanf, sscanf, when given %s conversion specification

- 10 -

## Vulnerable Buffer Code

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

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

- 11 -

## Buffer Overflow Executions

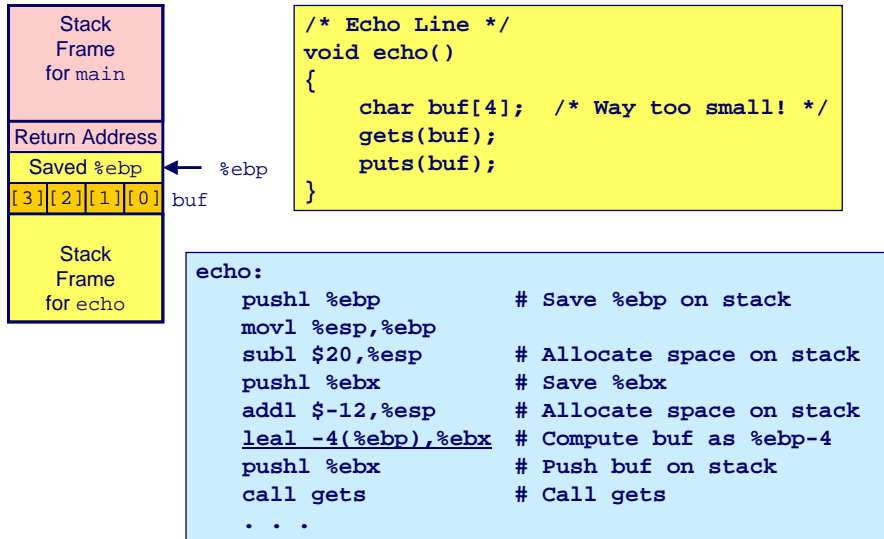
```
unix> ./bufdemo
Type a string:123
123
```

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

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

- 12 -

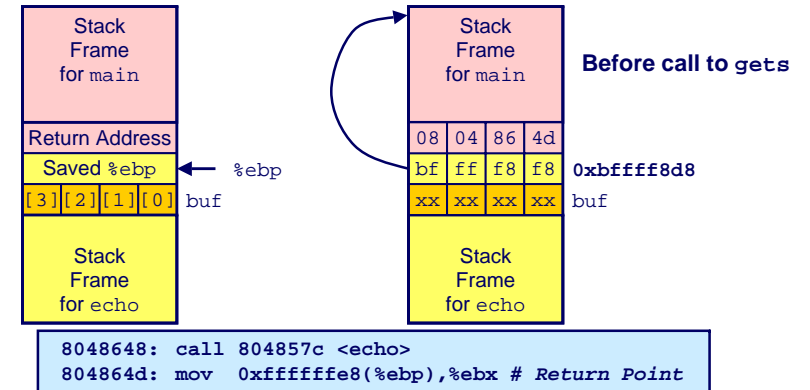
# Buffer Overflow Stack



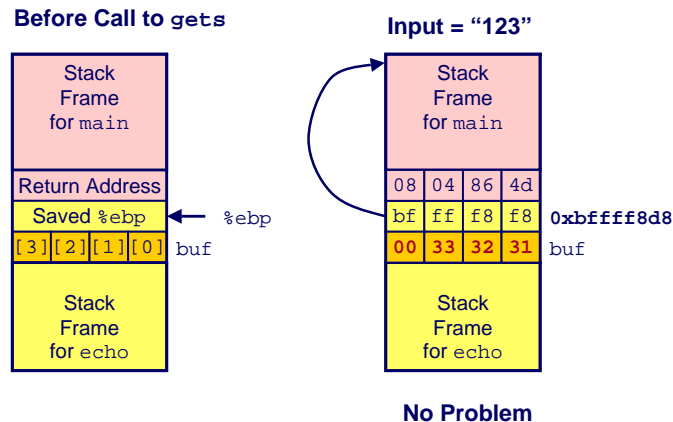
# Buffer Overflow Stack Example

```

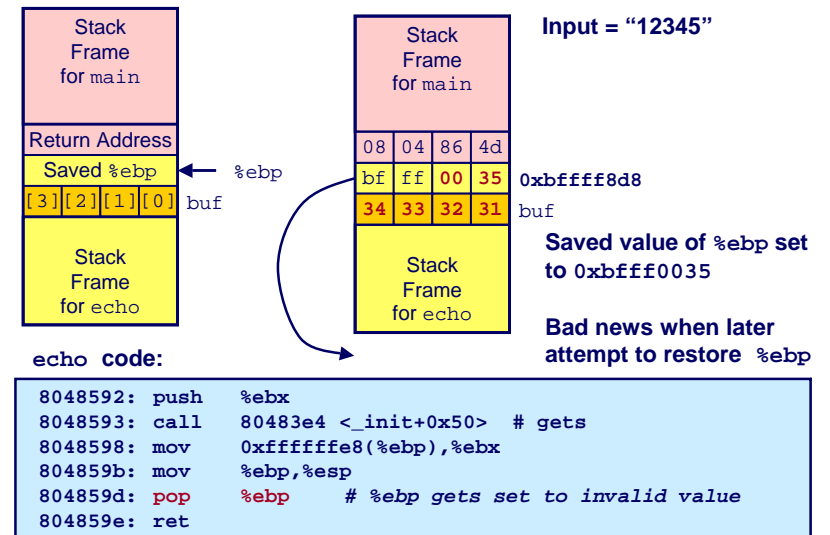
unix> gdb bufdemo
(gdb) break echo
Breakpoint 1 at 0x8048583
(gdb) run
Breakpoint 1, 0x8048583 in echo ()
(gdb) print /x *(unsigned *)$ebp
$1 = 0xbffff8f8
(gdb) print /x *((unsigned *)$ebp + 1)
$3 = 0x804864d
    
```



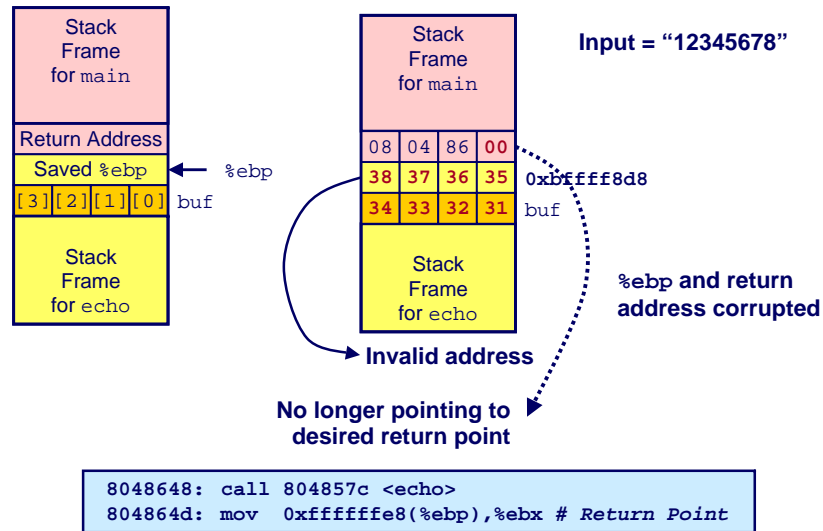
# Buffer Overflow Example #1



# Buffer Overflow Stack Example #2

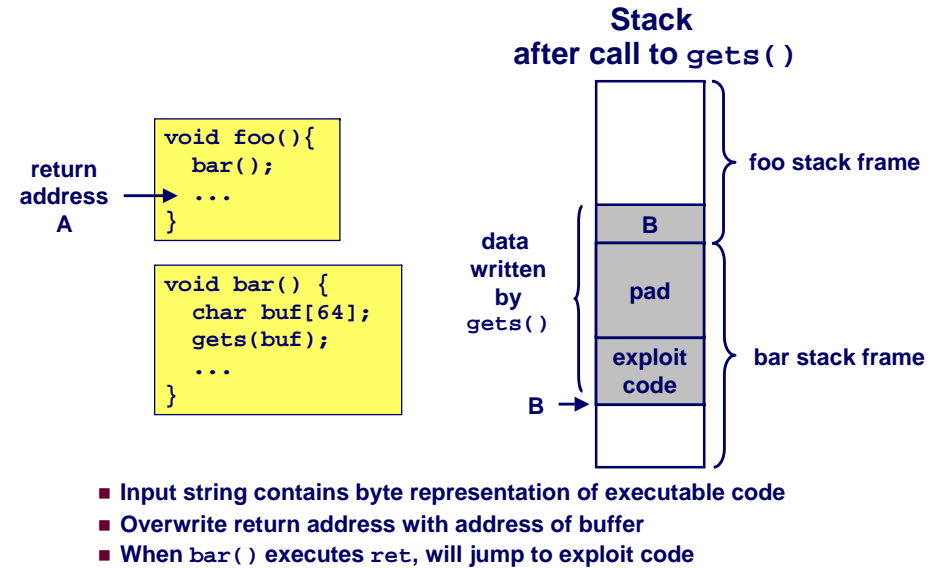


## Buffer Overflow Stack Example #3



- 17 -

## Malicious Use of Buffer Overflow



- 18 -

## Exploits Based on Buffer Overflows

**Buffer overflow bugs allow remote machines to execute arbitrary code on victim machines.**

### Internet worm

- Early versions of the finger server (fingerd) used `gets()` to read the argument sent by the client:
  - `finger droh@cs.cmu.edu`
- Worm attacked fingerd server by sending phony argument:
  - `finger "exploit-code padding new-return-address"`
  - exploit code: executed a root shell on the victim machine with a direct TCP connection to the attacker.

- 19 -

## Exploits Based on Buffer Overflows

**Buffer overflow bugs allow remote machines to execute arbitrary code on victim machines.**

### IM War

- AOL exploited existing buffer overflow bug in AIM clients
- exploit code: returned 4-byte signature (the bytes at some location in the AIM client) to server.
- When Microsoft changed code to match signature, AOL changed signature location.

- 20 -



## Avoiding Overflow Vulnerability

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

### Use Library Routines that Limit String Lengths

- `fgets` instead of `gets`
- `strncpy` instead of `strcpy`
- Don't use `scanf` with `%s` conversion specification
  - Use `fgets` to read the string

## Final Observations

### Memory Layout

- OS/machine dependent (including kernel version)
- Basic partitioning: stack/data/text/heap/DLL found in most machines

### Working with Strange Code

- Important to analyze nonstandard cases
  - E.g., what happens when stack corrupted due to buffer overflow
- Helps to step through with GDB