

# Today

- Buffer overflow example
- Memory hierarchy organization

## But first... Midterm info! ☺

- Midterm to be held next Wednesday, May 4<sup>th</sup>, in class
- Tomorrow's section will be a review for the midterm
- And the structure is...



# 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;
}
    
```

```

unix> ./bufdemo
Type a string: 1234567
1234567

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

unix> ./bufdemo
Type a string: 123456789ABC
Segmentation Fault
    
```

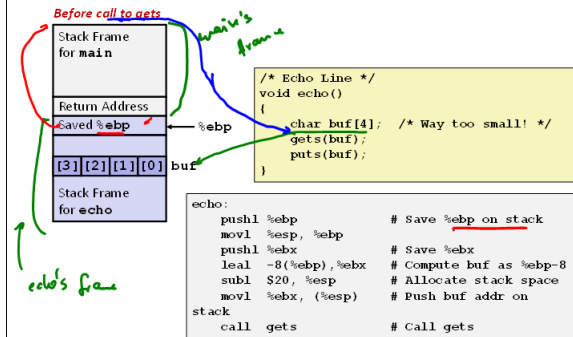
# Buffer Overflow Disassembly

```

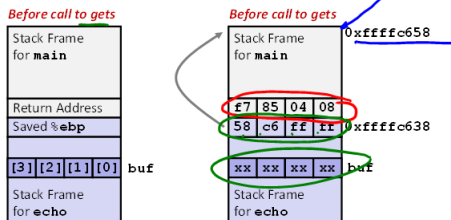
080484f0 <echo>:
80484f0: 55          push  %ebp
80484f1: 89 e5      mov   %esp,%ebp
80484f3: 53          push  %ebx
80484f4: 8d 5d f8   lea  0xfffffff8(%ebp),%ebx
80484f7: 83 ec 14   sub  $0x14,%esp
80484fa: 89 1c 24   mov  %ebx,(%esp)
80484fd: e8 ae ff ff call  80484b0 <gets>
8048502: 89 1c 24   mov  %ebx,(%esp)
8048505: e8 8a fe ff call  8048394 <puts@plt>
804850a: 83 c4 14   add  $0x14,%esp
804850d: 5b        pop   %ebx
804850e: c9        leave
804850f: c3        ret

80485f2: a8 f9 fe ff call  80484f0 <echo>
80485f7: 8b 5d fc   mov  0xfffffff8(%ebp),%ebx
80485fa: c9        leave
80485fb: 31 c0     xor  %eax,%eax
80485fd: c3        ret
    
```

# Buffer Overflow Stack



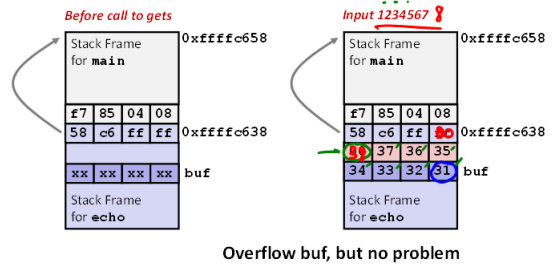
# Buffer Overflow Stack Example



```

80485f2: call 80484f0 <echo>
80485f7: mov 0xfffffff8(%ebp), %ebx # Return Point
    
```

# Buffer Overflow Example #1



Overflow buf, but no problem

### Buffer Overflow Example #2

*main Who has my esp?*

Before call to gets

Stack Frame for main				0xffffc658
7	85	04	08	0xffffc638
58	c6	ff	ff	
xx	xx	xx	xx	buf
Stack Frame for echo				

Input 123456789ABC-esp

Stack Frame for main				0xffffc658
7	85	04	08	0xffffc638
58	c6	ff	ff	
38	37	36	35	buf
34	33	32	31	buf
Stack Frame for echo				

**Base pointer corrupted**

```

804850a: 83 c4 14 add $0x14,%esp # deallocate space
804850d: 5b pop %ebx # restore %ebx
804850e: c9 leave # movl %ebp, %esp; popl %ebp
804850f: c3 ret # Return
  
```

### Buffer Overflow Example #3

Before call to gets

Stack Frame for main				0xffffc658
7	85	04	08	0xffffc638
58	c6	ff	ff	
xx	xx	xx	xx	buf
Stack Frame for echo				

Input 123456789ABC

Stack Frame for main				0xffffc658
7	85	04	00	0xffffc638
43	42	41	39	
38	37	36	35	buf
34	33	32	31	buf
Stack Frame for echo				

**Return address corrupted**

```

80485f2: call 80484f0 <echo>
80485f7: mov 0xffffffff(%ebp),%ebx # Return Point
  
```

### Malicious Use of Buffer Overflow

```

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

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

Stack after call to gets ()

- ➡ Input string contains byte representation of executable code
- ➡ Stack frame must be big enough to hold exploit code
- ➡ Overwrite return address with address of buffer (need to know B)
- ➡ When bar () executes ret, will jump to exploit code (instead of A)

### 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.

### Code Red Worm

- History
  - June 18, 2001. Microsoft announces buffer overflow vulnerability in IIS Internet server
  - July 19, 2001. over 250,000 machines infected by new virus in 9 hours
  - White house must change its IP address. Pentagon shut down public WWW servers for day

### Code Red Exploit Code

- Starts 100 threads running
- Spread self
  - Generate random IP addresses & send attack string
  - Between 1st & 19th of month
- Attack www.whitehouse.gov
  - Send 98,304 packets; sleep for 4-1/2 hours; repeat
    - Denial of service attack
  - Between 21st & 27th of month
- Deface server's home page
  - After waiting 2 hours
- Later versions even more aggressive
- And it goes on still...

## How could we avoid buffer overflows vulnerabilities?

- randomize
- fix your code (limit to buf size)

## Avoiding Overflow Vulnerability

```

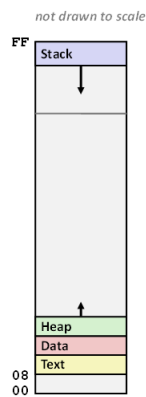
/* Echo Line */
void echo()
{
    char buf[4]; /* Way too small! */
}
*/
fgets(buf, 4, stdin);
puts(buf);

```

- Use library routines that limit string lengths
  - fgets instead of gets (second argument to fgets sets limit)
  - strncpy instead of strcpy
  - Don't use scanf with %s conversion specification
    - Use fgets to read the string
    - Or use %ns where n is a suitable integer

## System-Level Protections

- Randomized stack offsets
  - At start of program, allocate random amount of space on stack
  - Makes it difficult for hacker to predict beginning of inserted code
- Nonexecutable code segments
  - Only allow code to execute from "text" sections of memory
  - Do NOT execute code in stack, data, or heap regions
  - Hardware support



## Worms and Viruses

- Worm: A program that
  - Can run by itself
  - Can propagate a fully working version of itself to other computers
- Virus: Code that
  - Adds itself to other programs
  - Cannot run independently
- Both are (usually) designed to spread among computers and to wreak havoc (and, these days, profit\$\$\$)