Buffer Overflow, Worms, Viruses

Instructor: Sanjeev Setia

Internet Worm and IM War

- **November, 1988**
  - Internet Worm attacks thousands of Internet hosts.
  - How did it happen?
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

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.
String Library Code

- Implementation of Unix function gets()

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

- No way to specify limit on number of characters to read

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

Vulnerable Buffer Code

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

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

```console
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   0xffffffff8(%ebp),%ebx
  80484f7:      83 ec 14  sub    $0x14,%esp
  80484fa:      89 1c 24  mov    %ebx,(%esp)
  80484fd:      e8 ae ff ff ff  call   80484b0 <gets>
  8048502:      89 1c 24  mov    %ebx,(%esp)
  8048505:      e8 8a fe ff ff  call   8048394 <puts@plt>
  804850a:      83 c4 14  add    $0x14,%esp
  804850d:      5b   pop    %ebx
  804850e:      c9   leave
  804850f:      c3   ret
  80485f2:      e8 f9 fe ff ff  call   80484f0 <echo>
  80485f7:      8b 5d fc  mov 0xfffffffc(%ebp),%ebx
  80485fa:      c9   leave
  80485fb:      c3   ret
```

Buffer Overflow Stack

### Before call to gets

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

```
Stack Frame
for main

Return Address
Saved %ebp

[3][2][1][0]

buf

%ebp

Stack Frame
for echo
```

```
  echo:
    pushl %ebp    # Save %ebp on stack
    movl %esp, %ebp
    pushl %ebx    # Save %ebx
    leal -8(%ebp),%ebx # Compute buf as %ebp-8
    subl $20, %esp # Allocate stack space
    movl %ebx, (%esp) # Push buf on stack
    call gets # Call gets
...
```
Buffer Overflow Example

Before call to gets

Stack Frame for main

Return Address

Saved %ebp

buf

[3][2][1][0]

Stack Frame for echo

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

Before call to gets

Stack Frame for main

08 04 85 f7

0xffffffff

Stack Frame for echo

xx xx xx xx

buf

Stack Frame for main

08 04 85 f7

0xffffffff

Stack Frame for echo

00 37 36 35

34 33 32 31

buf

Overflow buf, but no problem

Input 1234567

0xffffffff
Buffer Overflow Example #2

Before call to `gets`

```
Stack Frame for main
08 04 85 f7
ff ff c6 58
xx xx xx xx
```

Input 12345678

```
Stack Frame for main
08 04 85 f7
ff ff c6 00
```

Base pointer corrupted

```
0xffffc638
0xffffc658
```

Before call to `echo`

```
Stack Frame for echo
```

```
Stack Frame for echo
```

```
Before call to `gets`

```
Stack Frame for main
08 04 85 f7
ff ff c6 58
xx xx xx xx
```

Input 12345678

```
Stack Frame for main
08 04 85 00
```

Return address corrupted

```
0xffffc638
0xffffc658
```

```
Stack Frame for echo
```

```
Stack Frame for echo
```

```
Before call to `echo`

```
Stack Frame for echo
```

```
Stack Frame for echo
```

```
```

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
Malicious Use of Buffer Overflow

- Input string contains byte representation of executable code
- Overwrite return address with address of buffer
- When `bar()` executes `ret`, will jump to exploit code

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.
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.

---

Date: Wed, 11 Aug 1999 11:30:57 -0700 (PDT)
From: Phil Bucking <philbucking@yahoo.com>
Subject: AOL exploiting buffer overrun bug in their own software!
To: rms@pharlap.com

Mr. Smith,

I am writing you because I have discovered something that I think you might find interesting because you are an Internet security expert with experience in this area. I have also tried to contact AOL but received no response.

I am a developer who has been working on a revolutionary new instant messaging client that should be released later this year.

... It appears that the AIM client has a buffer overrun bug. By itself this might not be the end of the world, as MS surely has had its share. But AOL is now "exploiting their own buffer overrun bug" to help in its efforts to block MS Instant Messenger.

... Since you have significant credibility with the press I hope that you can use this information to help inform people that behind AOL's friendly exterior they are nefariously compromising peoples' security.

Sincerely,
Phil Bucking
Founder, Bucking Consulting
philbucking@yahoo.com

*It was later determined that this email originated from within Microsoft!*
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

- **When We Set Up CS:APP Web Site**
  - Received strings of form
    
    ```
    GET /default.ida?
    NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
    %u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%03%u8b00%u531b%u53ff%u0078%u0000%u00=a
    HTTP/1.0" 400 325 "-" "-
    ```

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
Code Red Effects

- Later Version Even More Malicious
  - Code Red II
  - As of April, 2002, over 18,000 machines infected
  - Still spreading
- Paved Way for NIMDA
  - Variety of propagation methods
  - One was to exploit vulnerabilities left behind by Code Red II
- ASIDE (security flaws start at home)
  - .rhosts used by Internet Worm
  - Attachments used by MyDoom (1 in 6 emails Monday morning!)

Avoiding Overflow Vulnerability

```c
/* 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
    - 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**
  - In traditional x86, can mark region of memory as either “read-only” or “writeable”
    - Can execute anything readable
    - Add explicit “execute” permission

---

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**
  - Add itself to other programs
  - Cannot run independently

- **Both are (usually) designed to spread among computers and to wreck havoc**