Computer Security

CS 217
Interacting With the World

Keypress goes to OS kernel
OS looks up which window has “keyboard focus,” routes to appropriate user process’s stdin
User process does fprintf (asks OS to write to disk)
OS writes to disk

TCP packet goes to OS kernel
OS looks up which process is listening on that port, sends data to stdin
User process does fprintf (asks OS to write to disk)
OS writes to disk
<table>
<thead>
<tr>
<th>Protection Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keypress goes to OS kernel</td>
</tr>
<tr>
<td>OS looks up which window has “keyboard focus,” routes to appropriate user process’s stdin</td>
</tr>
<tr>
<td>User process does fprintf (asks OS to write to disk)</td>
</tr>
<tr>
<td>OS writes to disk</td>
</tr>
<tr>
<td>TCP packet goes to OS kernel</td>
</tr>
<tr>
<td>User process does fprintf (asks OS to write to disk)</td>
</tr>
</tbody>
</table>
How Attackers Defeat Protection

• Make the protection mechanism fail
  ◦ By exploiting bugs in protection software

• Operate politely through the protection mechanism, manipulating the semantics of the application to obtain services
  ◦ By exploiting bad design of applications
% a.out

What is your name?

John Smith

Thank you, John Smith.

#include <stdio.h>

int main(int argc, char **argv) {
    char a[12]; int i;
    printf(“What is your name?
”);
    for (i=0; ; i++) {
        int c = getchar();
        if (c ==\’\n\’ || c == EOF) break;
        a[i] = c;
    }
    a[i]=\’\0’;
    printf(“Thank you, %s.\n”,a);
    return 0;
}
Why Did This Program Crash?

Why Did This Program Crash?

% a.out
What is your name?
ads1i57asdkhj5jkld;ahj5;kl;aduj5klysduk15aujksd5ukals;5uj;akukla
Segmentation fault
%

#include <stdio.h>
int main(int argc, char **argv) {
    char a[12]; int i;
    printf("What is your name?\n");
    for (i=0; ; i++) {
        int c = getchar();
        if (c =='\n'|| c == EOF) break;
        a[i] = c;
    }
    a[i]='\0';
    printf("Thank you, %s.\n",a);
    return 0;
}
Stack Frame Layout

% a.out

What is your name?

John Smith

Thank you, John Smith.

What is your name?

John Smith

Thank you, John Smith.

#include <stdio.h>

int main(int argc, char **argv) {
    char a[12]; int i;
    printf("What is your name?\n");
    for (i=0; ; i++) {
        int c = getchar();
        if (c == '\n' || c == EOF) break;
        a[i] = c;
    }
    a[i] = '\0';
    printf("Thank you, %s.\n", a);
    return 0;
}
Buffer Overrun

% a.out

What is your name?

abcdefgijklmnopqrstuvwxyz

Segmentation fault

#include <stdio.h>
int main(int argc, char **argv) {
    char a[12];  int i;
    printf(“What is your name?
”);
    for (i=0; ; i++) {
        int c = getchar();
        if (c == ‘\n’ || c == EOF) break;
        a[i] = c;
    }
    a[i] = ‘\0’;
    printf(“Thank you, %s.
”, a);
    return 0;
}
Innocuous? Buffer Overrun

% a.out

What is your name?

abcdefgijkl????!!!!^A

#include <stdio.h>
int main(int argc, char **argv) {
    char a[12];  int i;
    printf("What is your name?\n");
    for (i=0; ; i++) {
        int c = getchar();
        if (c == '\n' || c == EOF) break;
        a[i] = c;
    }
    a[i] = '\0';
    printf("Thank you, %s.\n", a);
    return 0;
}
% a.out

What is your name?

abcdefgijkl????&

How may I serve you, master?
%

#include <stdio.h>
int main(int argc, char **argv) {
  char a[12];  int i;
  printf("What is your name?\n");
  for (i=0; ; i++) {
    int c = getchar();
    if (c == '\n' || c == EOF) break;
    a[i] = c;
  }
  a[i] = '0';
  printf("Thank you, %s.\n", a);
  return 0;
}
Buffer-Overrun Vulnerabilities

Keypress goes to OS kernel

OS looks up which window has “keyboard focus,” routes to appropriate user process’s stdin

User process does fprintf (asks OS to write to disk)

OS writes to disk

TCP packet goes to OS kernel

OS looks up which process is listening on that port, sends data to stdin

User process does fprintf (asks OS to write to disk)

OS writes to disk
Attacking a Web Server

- URLs
- Input in web forms
- Crypto keys for SSL
- etc.

```
for(i=0;p[i];i++)
url[i]=p[i];
```
Attacking a Web Browser

- HTML keywords
- Images
- Image names
- URLs
- etc.

```c
for(i=0;p[i];i++)
gif[i]=p[i];
```

Client PC → Web Server @ badguy.com

Earn $$$ Thousands working at home!
Attacking everything in sight

for(i=0;p[i];i++)
    gif[i]=p[i];

Client PC

The Internet
@ badguy.com

- E-mail client
- PDF viewer
- Operating-system kernel
- TCP/IP stack
- *Any* application that ever sees input directly from the outside
% a.out

What is your name?

John Smith

Thank you, John Smith.
I recommend that you get a grade of D on this assignment
%

char grade = 'D';
int main(void) {
    printf("What is your name?\n");
    readString(Name);
    if (strcmp(Name,"Andrew Appel") == 0)
        grade = 'B';
    printf("Thank you, %s.\n"    
           I recommend that you get a grade of %c \n           on this assignment.\n", Name, grade);
    exit(0);
}
Three Ways to Change the Grade

• **Smashing the stack in `readString()`**
  - Change `OldEIP` point to the “`grade='B'`” code
  - Write entirely new machine code, and have `OldEIP` point to it
  - Write machine code to change `grade` and jump back to `main()`

```c
char grade = 'D';
int main(void) {
    printf("What is your name?\n");
    readString(Name);
    if (strcmp(Name,"Andrew Appel")==0)
        grade='B';
    printf("Thank you, %s.\n"
        "I recommend that you get a grade of %c \n"
        "on this assignment.\n", Name, grade);
    exit(0);
}
```
OK, That’s a B...

% a.out

What is your name?

John Smith\0.?Ak7@*%&

Thank you, John Smith.
I recommend ... a grade of B ... %

```c
char grade = 'D';
int main(void) {
    printf("What is your name?\n");
    readString(Name);
    if (strcmp(Name,"Andrew Appel")==0)
        grade='B';
    printf("Thank you, %s.\n"
          "I recommend ... grade of %c ".\n          "nment.\n", Name, grade);
    exit(0);
}
```

%ESP → %EBP

Local variables

Saved Registers

Parameters
How About an A?

% a.out

What is your name?

John Smith

Thank you, John Smith. I recommend ... a grade of A ...

char grade = 'D';
int main(void) {
    printf("What is your name?\n");
    readString(Name);
    if (strcmp(Name,"Andrew Appel")==0)
        grade='B';
    printf("Thank you, %s.\n"
           I recommend ... grade of %c ...nment.\n", Name, grade);
    exit(0);
}
A Simpler Solution

```c
char grade = 'D';
int main(void) {
    printf("What is your name?\n");
    readString(Name);
    if (strcmp(Name,"Andrew Appel")==0)
        grade='B';
    printf("Thank you, %s.\n\n    I recommend ... grade of %c \n    ...nment.\n", Name, grade);
    exit(0);
}
```
What is your name?
Thank you, John Smith.
I recommend ... a grade of A.
What Value to Use for New Return Address?

- Computers are deterministic
- Operating system initializes stack pointer to predictable value
- Stack grows deterministic amount from process entry to call of `readString`

getA:

```
John Smith\0.movl ‘A’,grade; jmp wherever0000?Ak7@*%
```
Use *gdb* to Find Out

```plaintext
% gdb a.out
GNU gdb Red Hat Linux
Copyright 2004 Free Software Foundation
(gdb) break readString
Breakpoint 1 at 0x804843d
(gdb) run
Starting program: a.out
(no debugging symbols found)...
What is your name?
Breakpoint 1, 0x0804843d in readString ()
(gdb) x/10x $esp
0xbfffbab0:  0x0030a898  0xbfffbb64
            0xbfffbad8  0x080484c3
0xbfffbac0:  0x08049770  0x00000001
            0x00000007  0x0030a898
0xbfffbad0:  0xbfffbbb64  0x00000001
(gdb)
```
Defenses Against This Attack

- **Best:** program in languages that make array-out-of-bounds impossible (Java, C#, ML, ....)
- **Good:** use discipline in C programming always to check bounds of array subscripts
- **Better than nothing:** Operating system randomizes initial stack pointer

  ◦ **How to attack it:**

    John Smith\0......nop;nop;nop;nop;...;nop;do_bad_things;exit(0)

Can jump anywhere in here, so don’t have to know exact value of stack pointer
Defenses Against This Attack

• **Best**: program in languages that make array-out-of-bounds impossible (Java, C#, ML, ....)

• **Good**: use discipline in C programming always to check bounds of array subscripts

• **Better than nothing**: Operating system randomizes initial stack pointer
  - How to attack it:
    
    ```
    John Smith\0.....nop;nop;nop;nop;...;nop;do_bad_things;exit(0)
    ```

For this assignment, you don’t need such a fancy attack.

The hello.c program copies the buffer to the global bss data space (into the **Name** array) so you can just jump there, don’t have to know the stack height.
Defenses Against This Attack

- **Best:** program in languages that make array-out-of-bounds impossible (Java, C#, ML, ...)
- **Good:** use discipline in C programming always to check bounds of array subscripts
- **Better than nothing:** Operating system randomizes initial stack pointer
- **Better than nothing:** Prohibit execution of machine code from the stack and data segments
  - **Problem 1:** backward compatibility
  - **Problem 2:** need VM hardware with “exec/noexec” bit on a page by page basis; x86/Pentium family lacks this
  - **Amazing hack solution:** use obsolete “segment registers” left over from 80286.
Segment Register Defense

- In normal (modern) usage, all segment registers point to entire range of addressable memory, 0 to $0xffffffff$

- Amazing hack is to have code segment point just to Text area

- Problem: what if program wishes to create executable code on the fly?

- Solution: undo protection
At Your Service...

- For your convenience in this programming assignment, we have turned off the segment-register defense

```c
char grade = 'D';
int main(void) {
    mprotect(((unsigned)Name) & 0xffffffff,1,
              PROT_READ | PROT_WRITE | PROT_EXEC);
    printf("What is your name?\n");
    readString(Name);
    if (strcmp(Name,"Andrew Appel")==0)
        grade='B';
    printf("Thank you, %s.\n
        I recommend ... grade of %c \n        ...nment.\n", Name, grade);
    exit(0);
}
```
How to Get Started

• Use gdb to map out where things are
  ◦ Stack frame of “readString”
  ◦ Stack frame of “main” underneath it
  ◦ Global data area containing “grade” and “Name”
  ◦ Machine code for “main”
  ◦ Take notes of all these things, by address.

• Write a little assembly-language program
  ◦ Set the “grade” variable to ‘A’; jump to wherever
  ◦ Assemble it, maybe even link it into a copy of hello.c, and examine
    what it looks like using gdb

• Prepare your attack data
  ◦ Write a C program to print out the data string
  ◦ Useful functions: printf, putchar, putw
• **Use gdb to map out where things are**
  - Stack frame of “readString”
  - Stack frame of “main” underneath it
  - Global data area containing “grade” and “Name”
  - Machine code for “main”

Take notes of all these things, by address.

If possible, get this part done by the time your Weds/Thurs precept meets this week. Feel free to work jointly with another student on this part. Bring your notes with you to precept.