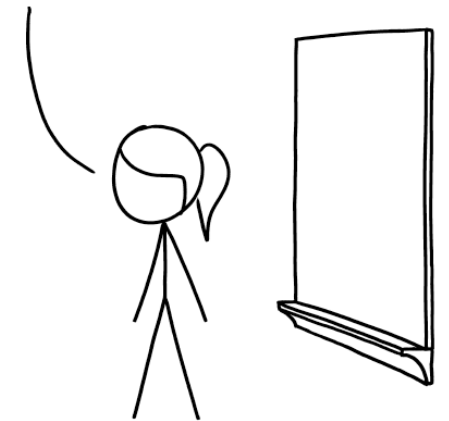# COS 217: Introduction to Programming Systems

Buffer Overrun Vulnerabilities and Assignment 6 (The 'B' Attack)

WELCOME TO YOUR FINAL EXAM.

THE EXAM IS NOW OVER.

I'M AFRAID ALL OF YOU FAILED.

YOUR GRADES HAVE BEEN STORED ON OUR DEPARTMENT SERVER AND WILL BE SUBMITTED TOMORROW.

CLASS DISMISSED.

CYBERSECURITY FINAL EXAMS

xkcd.com/2385

PRINCETON UNIVERSITY

```c
#include <stdio.h>
int main(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
            "and everything is %d\n", magic);
    return 0;
}
```

```
$ ./a.out
What is your name?
John Smith
Thank you, John Smith.
The answer to life, the universe, and everything is 42
```
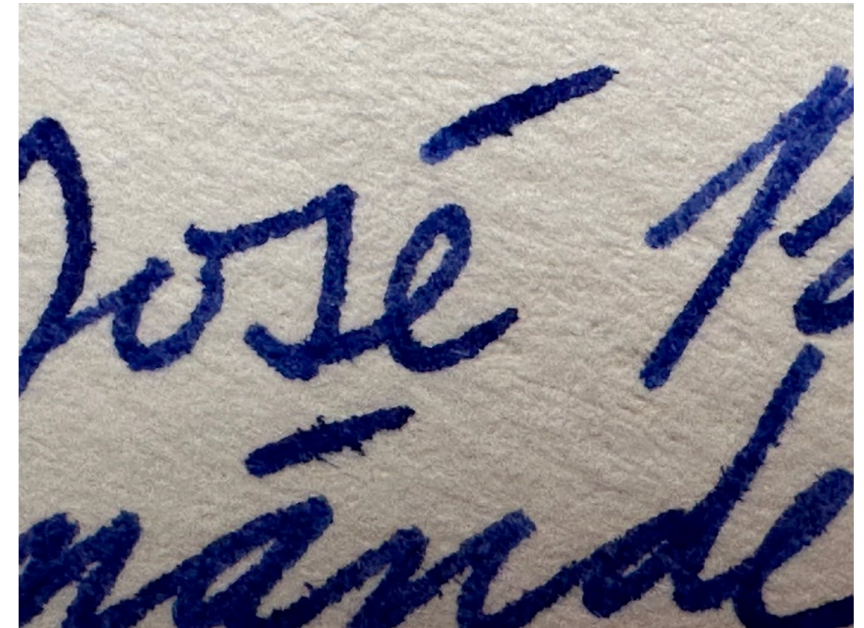
2

@grakozy

```c
#include <stdio.h>
int main(void)
{
   char name[12], c;
   int i = 0, magic = 42;
   printf("What is your name?\n");
   while ((c = getchar()) != '\n')
       name[i++] = c;
   name[i] = '\0';
   printf("Thank you, %s.\n", name);
   printf("The answer to life, the universe, "
          "and everything is %d\n", magic);
   return 0;
}
```

THE PROSPECT

Hello, my name is…

José Pablo Fernández García / The Daily Princetonian

José Pablo Fernández García
November 28, 2022 | 11:39pm EST

??? (!)

?

(Note: this is just the number that's actually printed when you run the code. It's not an attempt to Easter egg a phone number or anything like that. Please don't try to call it. Doing so almost certainly won't give you the answer to life, the universe, and everything.)

```
$ ./a.out
  What is your name?
  Christopher Moretti
  Thank you, Christopher Mor
  tti.
  The answer to life, the universe, and everything is 6911092
```

3

When there are too many characters, program carelessly writes beyond space "belonging" to name.

- Overwrites other variables
- This is a *buffer overrun*, or *stack smash*
- The program has a security bug!

```c
#include <stdio.h>
int main(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
           "and everything is %d\n", magic);
    return 0;
}
```

0

SP →

Return addr
name
⋮
c
magic
i

Old SP →

5

Example Trace

```c
#include <stdio.h>
int main(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
           "and everything is %d\n", magic);
    return 0;
}
```

0

SP →

Return addr
name
:
:
c
magic
i

Old SP →
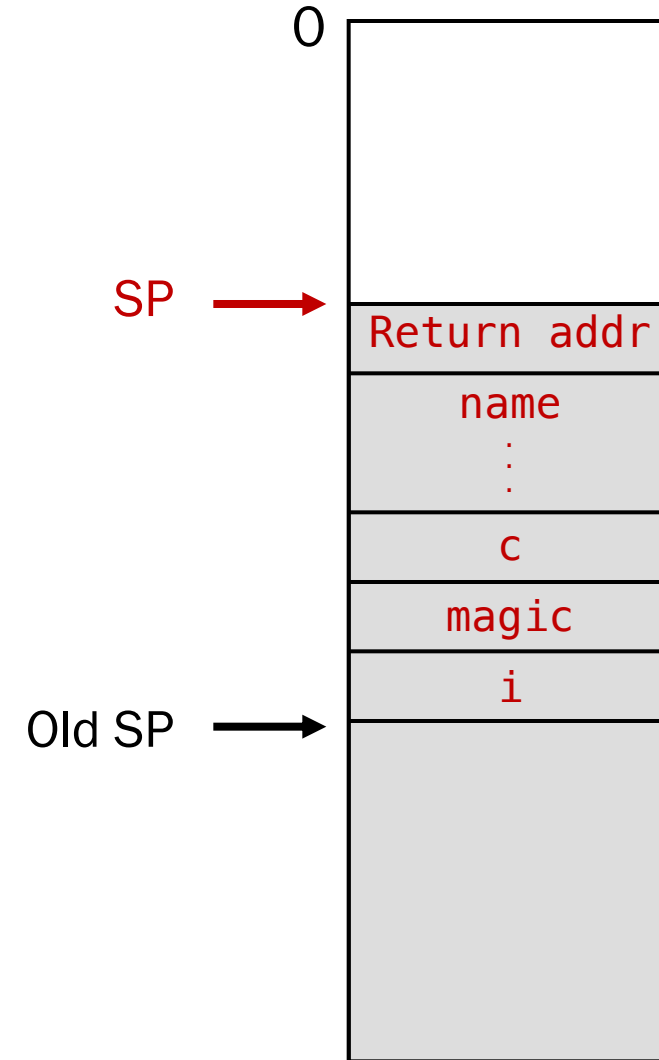
Christopher$_s$ (not \0 terminated) in name[0]–name[11]

Mor in 3 padding bytes before c, effectively: name[12]–name[14]

Each letter from getchar updates c , until c becomes '\n'.

(It is also overwritten once by name[i++] = c,

when i is 15 and c is 'e' because &c==&(name[15]))

First 't' overwrites 42 with 0x74 ('t') (2 high-order bytes still 0)

Second 't' makes magic 29812 (2 high-order bytes still 0)

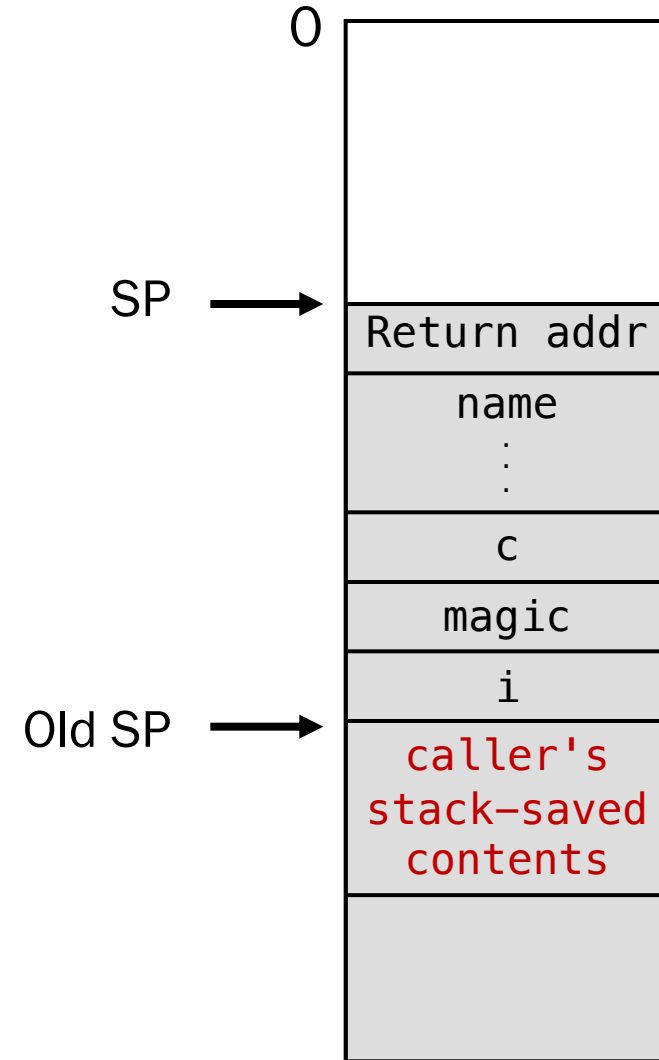Final 'i' makes magic 6911092 (1 high-order byte still 0)
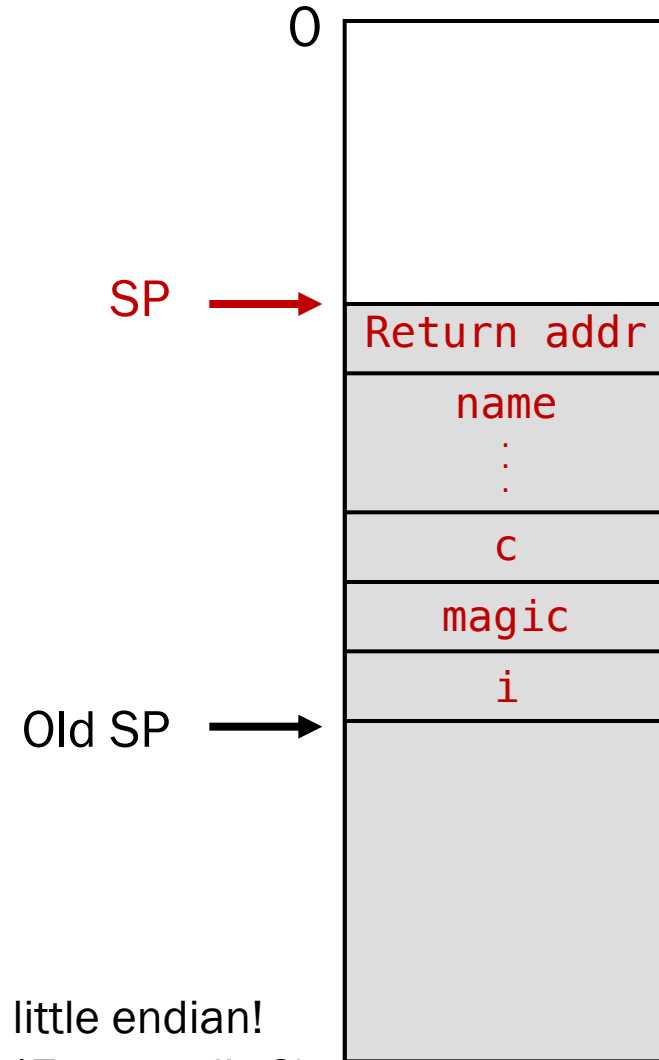
little endian!
(L17 appendix 2)

6

Buffer overrun can overwrite onto its
caller function's stack frame!


```c
#include <stdio.h>
int callee(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
           "and everything is %d\n", magic);
    return 0;
}
```
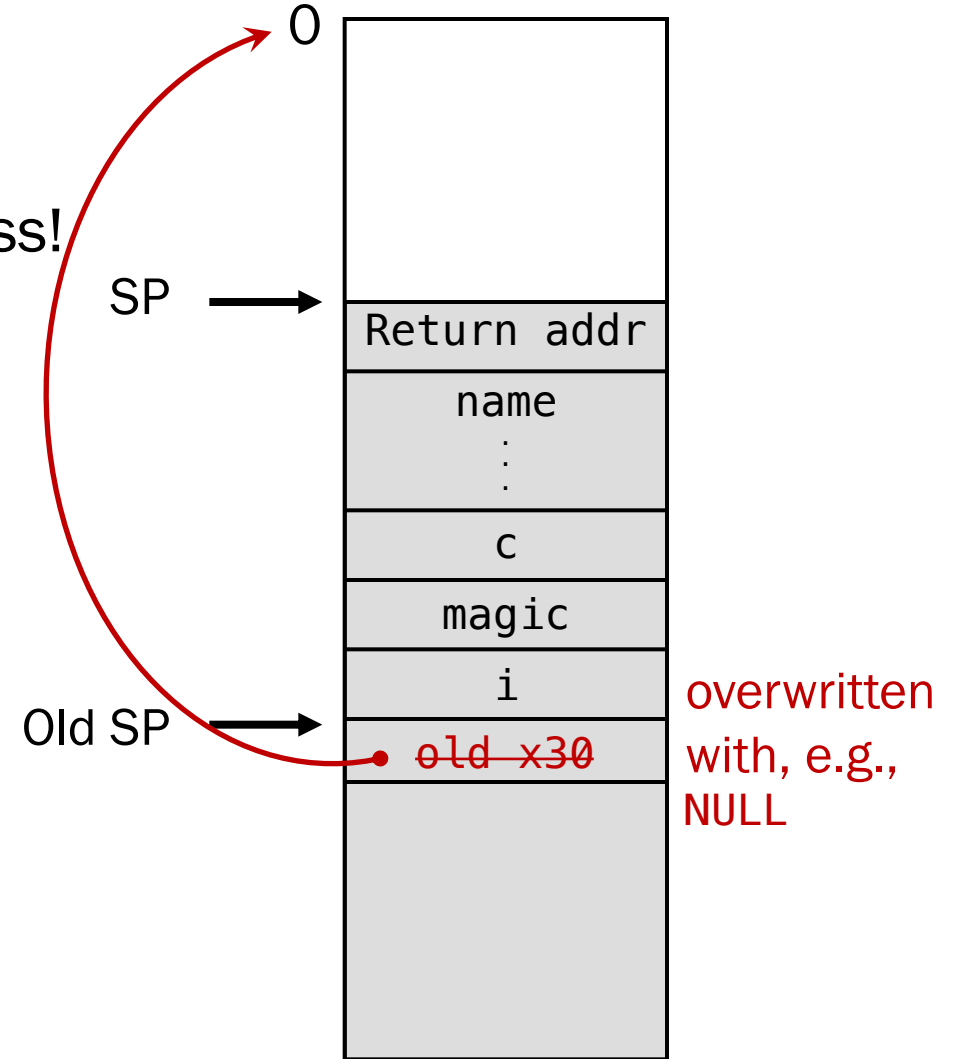
7

And somewhere on caller's stack frame is
the saved return address for that function ...

Buffer overrun can overwrite caller's return address!

- Replacement value can be an invalid address,
  leading to a segfault, or ...

```c
#include <stdio.h>
int callee(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
           "and everything is %d\n", magic);
    return 0;
}
```
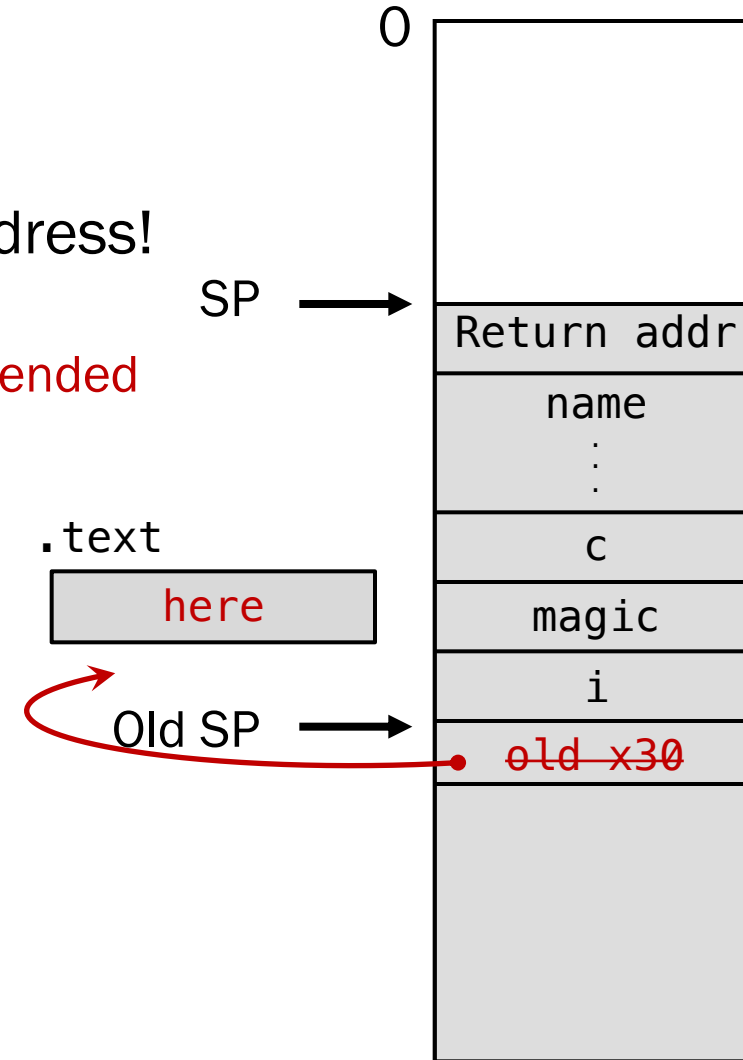
8

0

SP →

Return addr

name
:
:

c

magic

i

Old SP →

~~old x30~~

overwritten
with, e.g.,
NULL

It Gets Much Worse...

And somewhere on caller's stack frame is
the saved return address for that function ...

Buffer overrun can overwrite caller's return address!
- Replacement value can be an invalid address,
  leading to a segfault, or it can cleverly cause unintended
  control flow!

```c
#include <stdio.h>
int callee(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
           "and everything is %d\n", magic);
    return 0;
}
```

0

SP →

| Return addr |
| name |
| . |
| . |
| c |
| magic |
| i |
| old x30 |

.text

here

Old SP →

It Gets Much, Much Worse...

And somewhere on caller's stack frame is
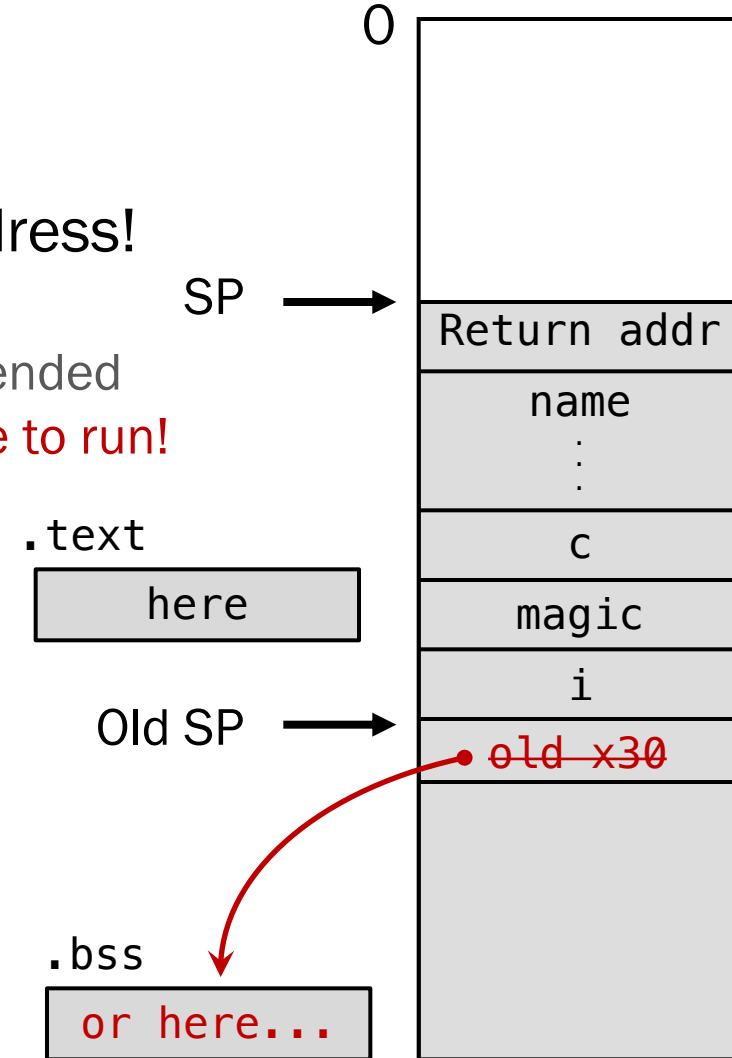the saved return address for that function ...

Buffer overrun can overwrite caller's return address!

- Replacement value can be an invalid address,
  leading to a segfault, or it can cleverly cause unintended
  control flow, or even cause arbitrary malicious code to run!

```c
#include <stdio.h>
int callee(void)
{
    char name[12], c;
    int i = 0, magic = 42;
    printf("What is your name?\n");
    while ((c = getchar()) != '\n')
        name[i++] = c;
    name[i] = '\0';
    printf("Thank you, %s.\n", name);
    printf("The answer to life, the universe, "
           "and everything is %d\n", magic);
    return 0;
}
```
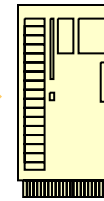
0

SP →

| Return addr |
| name |
| . |
| . |
| . |
| c |
| magic |
| i |

Old SP → old x30

.text

| here |

.bss

| or here... |

Attacking a Web Server

URLs

Input in web forms

Crypto keys for SSL

etc.

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

Client PC

Web Server

www.cs.princeton.edu
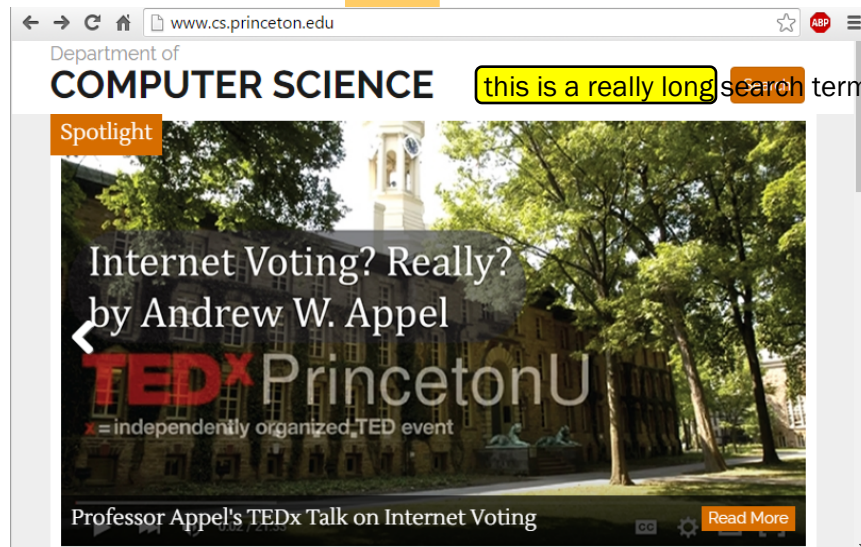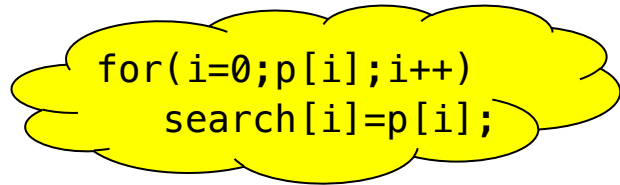
Department of
COMPUTER SCIENCE   this is a really long search term that overflows a buffer

Spotlight

Internet Voting? Really?
by Andrew W. Appel

TEDxPrincetonU
x = independently organized TED event

Professor Appel's TEDx Talk on Internet Voting   Read More

Attacking Everything in Sight

C/C++ MP4 video library (4/2023)

OpenSSL crypto library (11/2022)

Smart UPS devices (3/2022)

Zoom (11/2021)

VLC media player (1/2019)

Nintendo Switch (4/2018)

...

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

Client PC

The Internet
@ badguy.com

E-mail clients

PDF viewers

Operating-system kernels

TCP/IP Stack
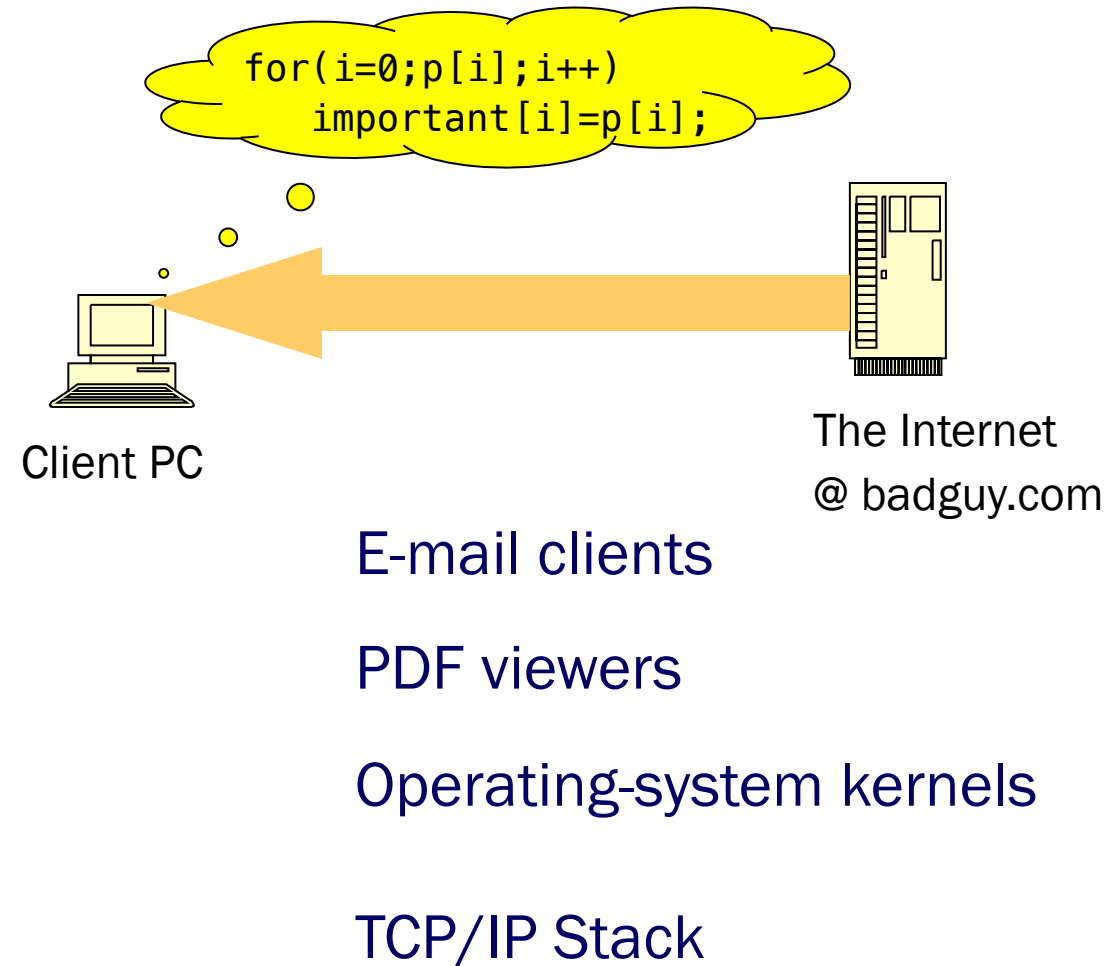
Any application that ever sees input directly from the outside!

Defenses Against This Attack

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

But if you need to use C…

Defenses Against This Attack

In C: use discipline and software analysis tools to check bounds of array subscripts

```
DESCRIPTION
        The strcpy() function copies the string pointed to by src, including
        the terminating null byte ('\0'), to the buffer pointed to by dest.
        The strings may not overlap, and the destination string dest must be
        large enough to receive the copy. Beware of buffer overruns! (See
        BUGS.)

BUGS
        Never use gets(). Because it is impossible to tell without knowing the data in advance how many characters gets() will read, and
        because gets() will continue to store characters past the end of the buffer, it is extremely dangerous to use. It has been used to
        break computer security. Use fgets() instead.
```

Augmented by OS- or compiler-level mitigations:

- Randomize initial stack pointer

- "No-execute" memory permission for sections other than .text
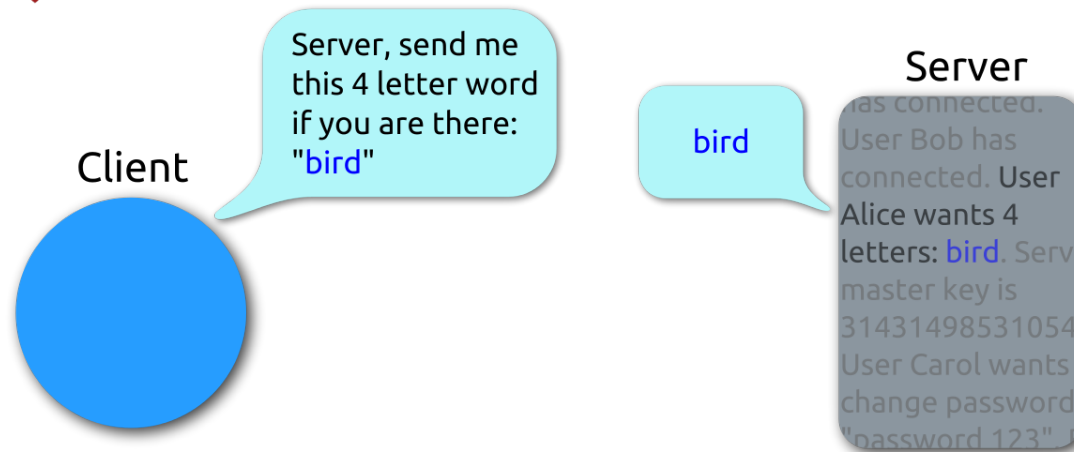
- "Canaries" at end of stack frames
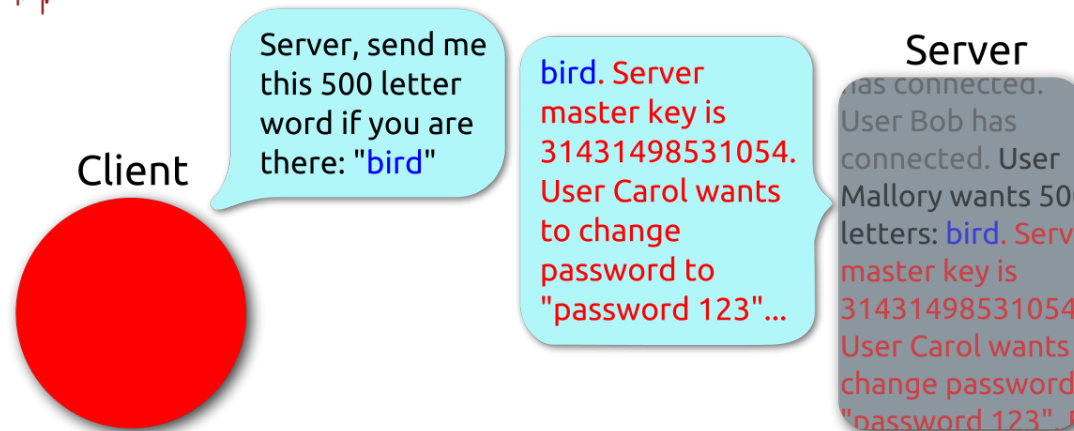
None of these would have prevented the "Heartbleed" attack

15

```
enum {BUFSIZE = 48};
char grade = 'D';
char name[BUFSIZE];
...
int main(void)
{
   mprotect(...);
   getname();
   if (strcmp(name, "Andrew Appel") == 0)
      grade = 'B';
   printf("%c is your grade.\n", grade);
   printf("Thank you, %s.\n", name);
   return 0;
}
```

```
$ ./grader
What is your name?
Joe Student
D is your grade.
Thank you, Joe Student.
$ ./grader
What is your name?
Andrew Appel
B is your grade.
Thank you, Andrew Appel.
```

17

```c
/* Prompt for name and read it */
void getName() {
  printf("What is your name?\n");
  readString();
}
```

```c
/* Read a string into name */
void readString() {
  char buf[BUFSIZE];
  int i = 0;
  int c;

  /* Read string into buf[] */
  for (;;) {
    c = fgetc(stdin);
    if (c == EOF || c == '\n')
      break;
    buf[i] = c;
    i++;
  }
  buf[i] = '\0';

  /* Copy buf[] to name[] */
  for (i = 0; i < BUFSIZE; i++)
    name[i] = buf[i];
}
```

Unchecked write to buffer!

```
enum {BUFSIZE = 48};
char grade = 'D';
char name[BUFSIZE];
...
int main(void)
{
    mprotect(...);
    getname();
    if (strcmp(name, "Andrew Appel") == 0)
        grade = 'B';
    printf("%c is your grade.\n", grade);
    printf("Thank you, %s.\n", name);
    return 0;
}
```
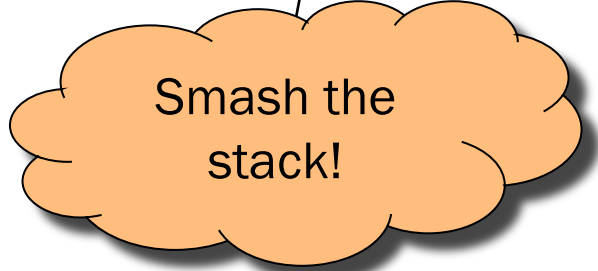
```
$ ./grader
What is your name?
Joe Student\0(#@&$%*#&(*^!@%*!(&$
B is your grade.
Thank you, Joe Student.
```
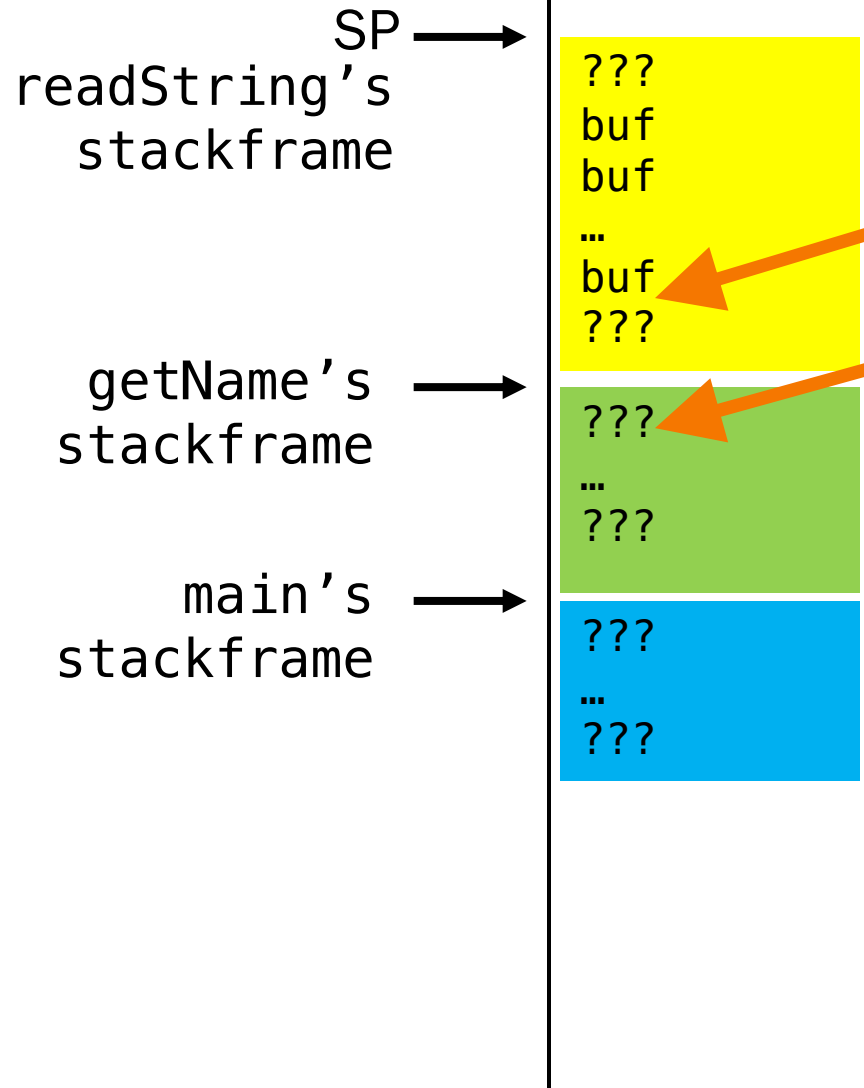
Smash the stack!

19

SP →

readString's
stackframe

??? buf buf … buf ???

Keep writing past end of buf

Get to getName's stackframe

getName's
stackframe →

??? … ???

What's there?

getName's saved x30!
(somewhere on stack)

main's
stackframe →

??? … ???

Overwrite it!

With what?

20
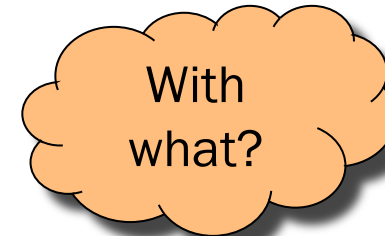
Assignment 6: Attack the "Grader" Program

```
enum {BUFSIZE = 48};
char grade = 'D';
char name[BUFSIZE];
...
int main(void)
{
   mprotect(...);
   getname();
   if (strcmp(name, "Andrew Appel") == 0)
      grade = 'B';
   printf("%c is your grade.\n", grade);
   printf("Thank you, %s.\n", name);
   return 0;
}
```

```
$ ./grader
What is your name?
Joe Student\0(#@&$%*#&(*^!@%*!(&$
B is your grade.
Thank you, Joe Student.
```

Memory Map of TEXT Section

readString →

```
rS prolog
rS instrs…
rS instrs…
…
rS epilog
rS return
```
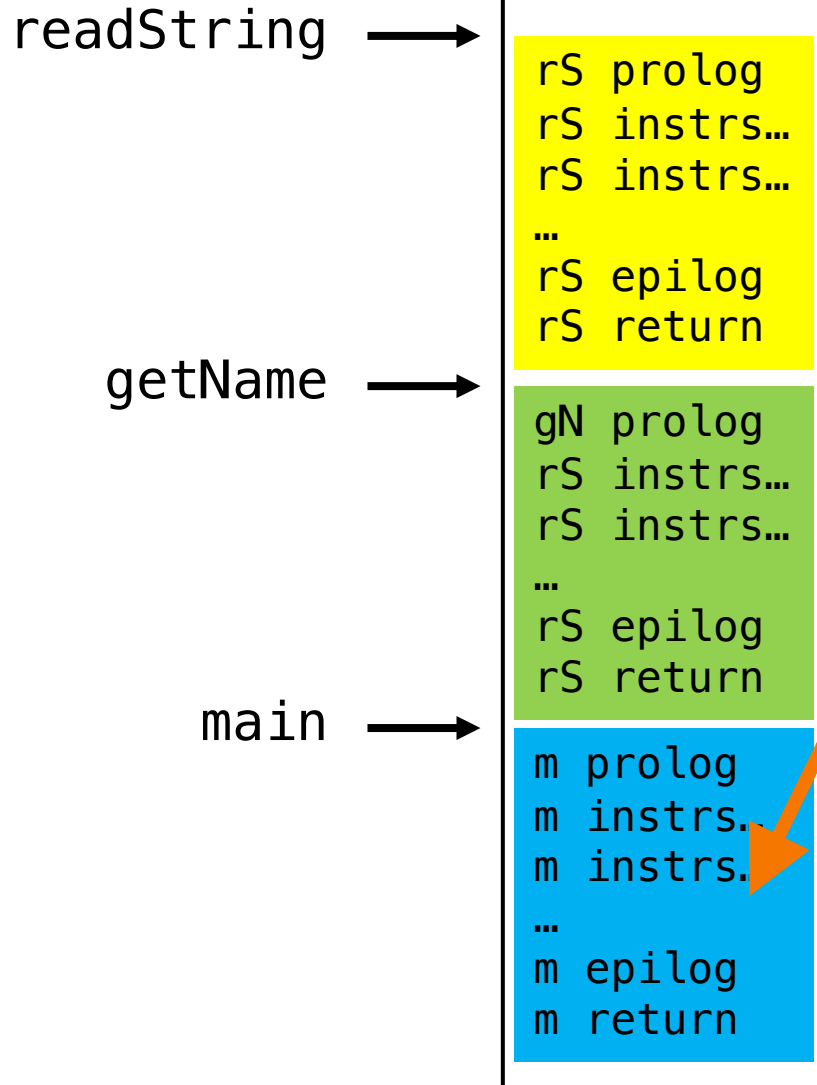
getName →

```
gN prolog
rS instrs…
rS instrs…
…
rS epilog
rS return
```

main →

```
m prolog
m instrs…
m instrs…
…
m epilog
m return
```

(All of these instructions are actually machine code, not flattened C, of course!)
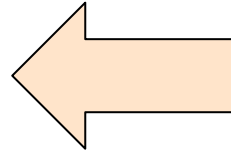
```
...
checkappel:
    if (strcmp(name, "Andrew Appel") != 0)
        goto afterb
    grade = 'B'  ← HERE!
afterb:
    print ...
...
```

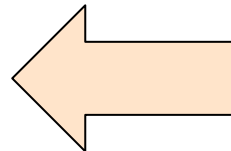# Construct Your Exploit String (`createdataB.c`)

1. **Your name.**
   - After all, the `grader` program's last line of output must be:
     "`Thank you, [your name].`"

   > `fopen` the file `"dataB"` and write your name into that file (e.g. with `fprintf`)

2. **A null byte.**
   - Otherwise, the `grader` program's last line of output will be corrupted.
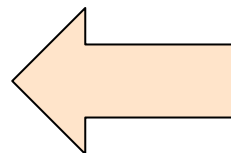
   > See "Writing Binary Data" precept handout. `'\0'` is just a single byte of binary data.

3. **Filler to overrun until** `x30`.
   - Presumably more null bytes are easiest, but easter eggs are fine.

4. **The address of the target**
   - The statement `grade = 'B'`.

   > Address is a 64-bit (little-endian) unsigned integer: C `unsigned long`

Let's Not Get Thrown in Jail, Please

Legal Information Institute [LII]
OPEN ACCESS TO LAW SINCE 1992

ABOUT LII ▸   GET THE LAW ▸   LAWYER DIRECTORY   LEGAL ENCYCLOPEDIA ▸   HELP OUT ▸

LII > U.S. Code > Title 18 > PART I > CHAPTER 47 > § 1030

18 U.S. Code § 1030 - Fraud and related activity in connection with computers

U.S. Code   Notes   State Regulations

prev | next

(a) Whoever—

(1) having knowingly accessed a computer without authorization or exceeding authorized access, and by means of such conduct having obtained information that has been determined by the United States Government pursuant to an Executive order or statute to require protection against unauthorized disclosure for reasons of national defense or foreign relations, or any restricted data, as defined in paragraph y. of section 11 of the Atomic Energy Act of 1954, with reason to believe that such information so obtained could be used to the injury of the United States, or to the advantage of any foreign nation willfully communicates, delivers, transmits, or causes to be communicated, delivered, or transmitted, or attempts to communicate,

https://www.law.cornell.edu/

Summary

- This lecture:
  - Buffer overrun attacks in general
  - Assignment 6 "B Attack" principles of operation

- Next precept:
  - Assignment 6 "B Attack" recap
  - Memory map using gdb
  - Writing binary data

- Final 2 lectures:
  - Assignment 6 "A Attack" overview
  - Machine language details needed for "A Attack"
  - *Finally* finishing the 4-stage build process: the Linker!

- Final precept:
  - MiniAssembler and "A Attack" details

25

"A 5"-second check-in

No judgement, we're not looking at your individual responses ...

We want to get a sense of where folks are on A5 (due next Thursday)

A. Haven't gotten a chance to start

B. Working on Part 1

C. Working on Part 2 (d)

D. Working on Part 2 (e)

E. *Must ... get ... below ...*
   1.5 ... seconds