The Assignment 6 ‘B’ Attack
A Program

```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
A Reason Why People With Long Names Can’t Have Nice Things

What is your name?
Christopher Moretti
Thank you, Christopher Moretti.
The answer to life, the universe, and everything is 6911092

```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?
Christopher Moretti
Thank you, Christopher Moretti.
The answer to life, the universe, and everything is 6911092
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;
}
```
```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;
}
```

Christopher's (not \0 terminated) in name[0]–name[11]
Mor in 3 padding bytes before c

Each letter from getchar overwrites c (it is also
overwritten once by name[i++] = c, when i is 15 and c is
‘e’) until c becomes ‘\n’ and the loop ends.

First t overwrites 42 with 0x74 (‘t’) – little endian!
Second t makes magic 29812 (2 high-order bytes still 0)
Final i makes magic 6911092 (1 high-order byte still 0)
Buffer overrun can overwrite return address of a previous stack frame!

```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;
}
```
Buffer overrun can overwrite return address of a previous stack frame!

- Value can be an invalid address, leading to a segfault,

```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;
}
```
It Gets **Much Worse**...

Buffer overrun can overwrite return address of a previous stack frame!

- Value can be an invalid address, leading to a segfault, or it can cleverly cause unintended control flow!

```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;
}
```
Buffer overflow can overwrite return address of a previous stack frame!

- 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 execute!
Defenses Against This Attack

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

If you must program in C: use discipline and software analysis tools to check bounds of array subscripts

Otherwise, stopgap security patches:
- Operating system randomizes initial stack pointer
- “No-execute” memory permission
- “Canaries” at end of stack frames
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;
}
Assignment 6: Attack the “Grader” Program

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];
}

void getName()
{
    printf("What is your name?\n");
    readString();
}

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

Unchecked write to buffer!
Assignment 6: Attack the “Grader” Program

```c
int main(void) {
    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?

Bob\0(#@$%**&(*^!@%*!)(&%$(@*

B is your grade.
Thank you, Bob.
Memory Map of STACK Section

-(SP) readString’s stackframe
  - ???
  - buf
  - buf
  - ...
  - buf
  - ???

- getName’s stackframe
  - ???
  - ...
  - ???

Keep writing past end of buf
Get to getName’s stackframe
What’s there?
getName’s saved x30! (somewhere on stack)
Overwrite it!

With what?
Assignment 6: Attack the “Grader” Program

```c
int main(void) {
    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?
Bob
B is your grade.
Thank you, Bob.
Memory Map of TEXT Section

readString →

prolog
instrs...
epilog
return

getName →

gN prolog
instrs...
epilog
return

main →

m prolog
instrs.
epilog
return

... 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].”

2. A null byte.
   • Otherwise the grader program’s last line of output will be corrupted.

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

4. The address of grade = ‘B’.

1. Open the file dataB and write your name into that file (e.g. with fprintf)

2. See “Writing Binary Data” precept handout. ‘\0’ is just a single byte of binary data.

4. The address is a little-endian two’s complement unsigned long.
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