CSE 484
GDB Introduction Review

Nicholas Shahan

Thanks to:
Sunjay Cauligi (GDB Notes)
Gaetano Borriello (351 Slides)
Cliff Zou (Buffer Overflow Notes)

References

How values are stored in memory:

http://courses.cs.washington.edu/courses/cse351/14sp/lectures/02-memory.pdf

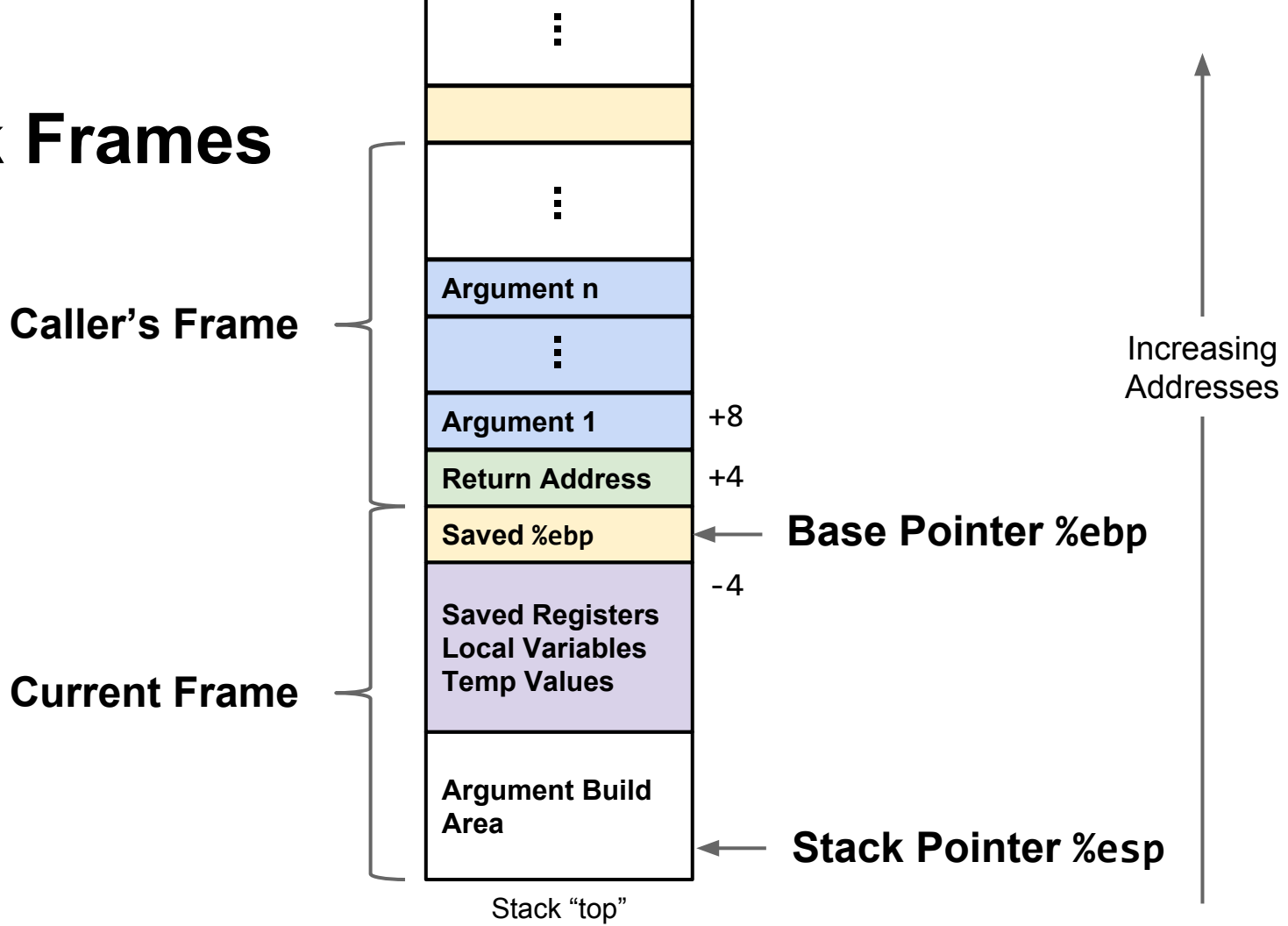
How the call stack works:

http://courses.cs.washington.edu/courses/cse351/14sp/lectures/06-procedures.pdf

GDB Cheat Sheet:

http://courses.cs.washington.edu/courses/cse351/14sp/sections/1/gdbnotes-x86-64.pdf

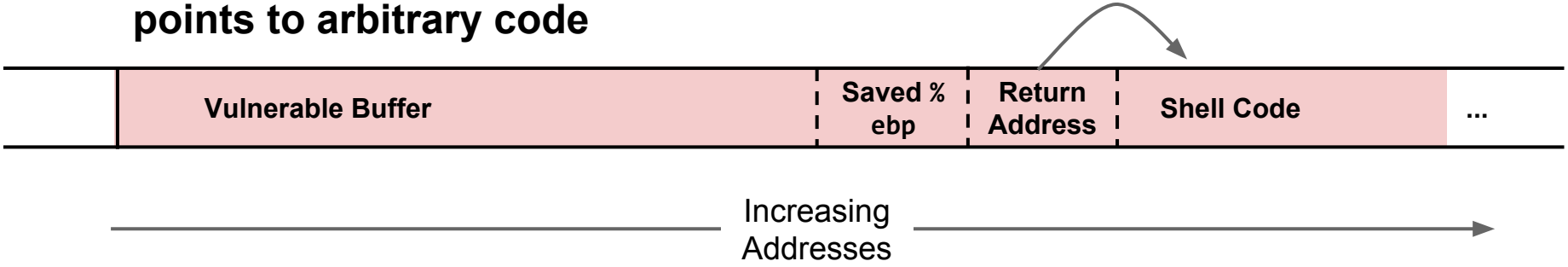
Stack Frames



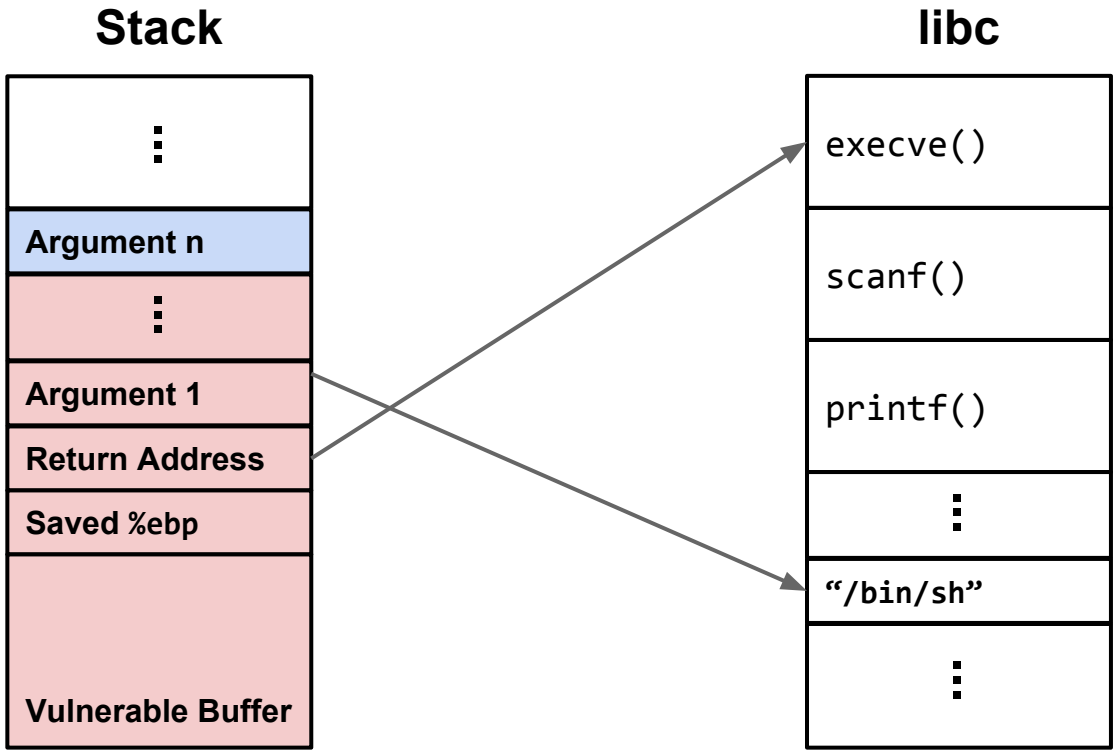
Basic Buffer Overflow

Original state of the stack memory

After overflowing the buffer the Return Address points to arbitrary code



Return to libc Attack



Format String Attacks

printf

%d decimal (int) value

%u unsigned decimal (unsigned int) value

%x hexadecimal (unsigned int) value

%s string ((const) (unsigned) char*) reference

%n number of bytes written so far, (*int) reference <- actually writes to a variable

Numbers directly after % force a width

example: printf(“%20s”, “hello”);

What does this do?

```
int num;
printf(“12345%5s%n”, “hi”, &num);
printf(“%d”, num);


12345   hi
10
```

Useful GDB Commands

step or s: executes the next line of code
next or n: executes the next line of code (does not enter functions)
break or b: sets a break-point (can give it line numbers or function names)
print or p: prints the value of a variable or register etc.
continue or c: runs the program until the next break-point
backtrace or bt: displays a stack trace from your current point of execution
x: examines the data at the given address. (can give it addresses, registers, variables etc.)
info register or i r: prints the values of all the registers
info frame or i f: prints important information about the current stack frame
layout src: splits window to show code (CTRL-x A to close the source view)

Lab1 Tips

1. Reconnaissance!
   a. Map out all the functions - WRITE THESE DOWN!
   b. Note the location of %esp and %ebp
   c. Note the location of the return address
   d. Note are the offsets of the local variables
2. Run GDB

   example: `$gdb ./sploit1`
3. In GDB:

   `catch exec`

   `run`

   `break main` (don't try to set breakpoints before you run)

GDB Example

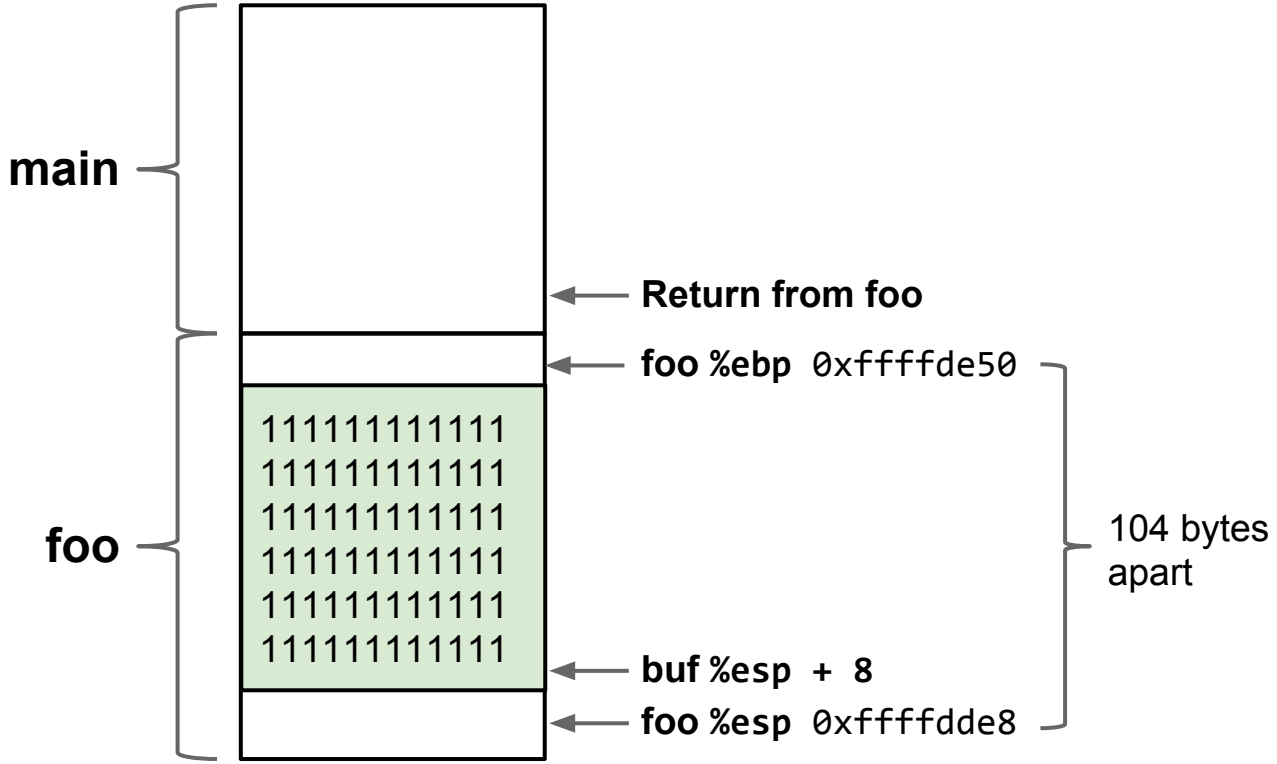
Demo Time

GDB Example - sploit1

Break in each function and examine the registers and variables.

Populate arguments with values that are easy to see on the stack.

Write down offsets!



Some x86 Assembly Review

LEAVE

copies %ebp to %esp, then POP %ebp

RET

POP %eip

POP <reg>

pops the top of the stack into the given register and increments %esp

Remember Byte Ordering?

Big-Endian (PowerPC, SPARC, The Internet)

Least significant byte has highest address

Little-Endian (x86 32 & 64 bit)

Least significant byte has lowest address

Example

Variable has 4-byte representation `0xA1B2C3D4`

Address of variable is `0x100`