15-213 Recitation: Attack Lab

Your TAs
September 23rd, 2019

Agenda

- Reminders
- Stacks
- Attack Lab Activities

Reminders

- **Bomb Lab is due tomorrow!**
- **Attack Lab is due Oct 1st, 2019!**
  - “But if you wait until the last minute, it only takes a minute!” – ***NOT!***
  - Don’t waste your grace days on this assignment!

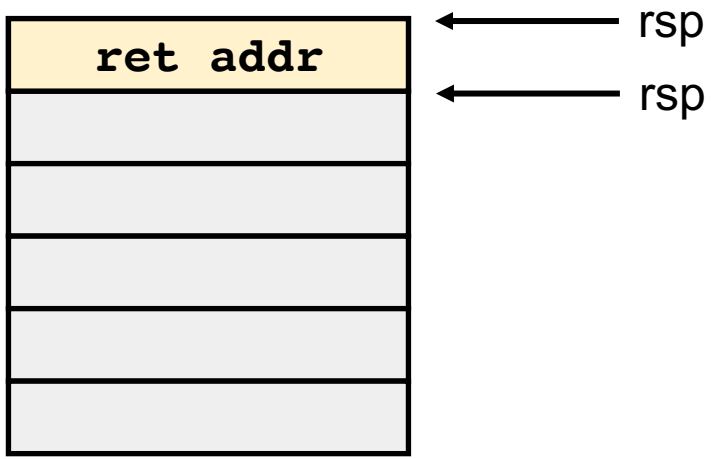
Attack Lab

- We’re letting you hijack programs by running buffer overflow attacks on them.
  - Is that not justification enough?
- Helps you understand stack discipline and stack frames
  - Most difficult part of the midterm exam
- Also let you defeat relatively secure programs with return oriented programming

Stack Overview

Let's say you have the following stack diagram. What happens when you call a function?

What information always goes on the stack?



Attack Lab Activities

Don’t be afraid if these concepts are unfamiliar! You will be learning them this week in lecture.

Attack Lab Activities

- **Three activities**
  - Each relies on a specially crafted assembly sequence to purposefully overwrite the stack

- **Activity 1 – Overwrites the return addresses**
- **Activity 2 – Writes an assembly sequence onto the stack**
- **Activity 3 – Uses byte sequences in libc as the instructions**

Attack Lab Activities

- **One student needs a laptop**
- **Login to a shark machine**

$ wget http://www.cs.cmu.edu/~213/activities/attacklab_activity.tar
$ tar xvf attacklab_activity.tar
$ cd attacklab_activity
$ make
$ gdb act1

Activity 1

(gdb) break clobber

(gdb) run

(gdb) x $rsp

(gdb) backtrace

Q. Does the value at the top of the stack match any frame?

Activity 1 Continued

(gdb) x /2gx $rdi // Here are the two key values

(gdb) stepi // Keep doing this until

```
(gdb)
clobber () at support.s:16
16              ret
```

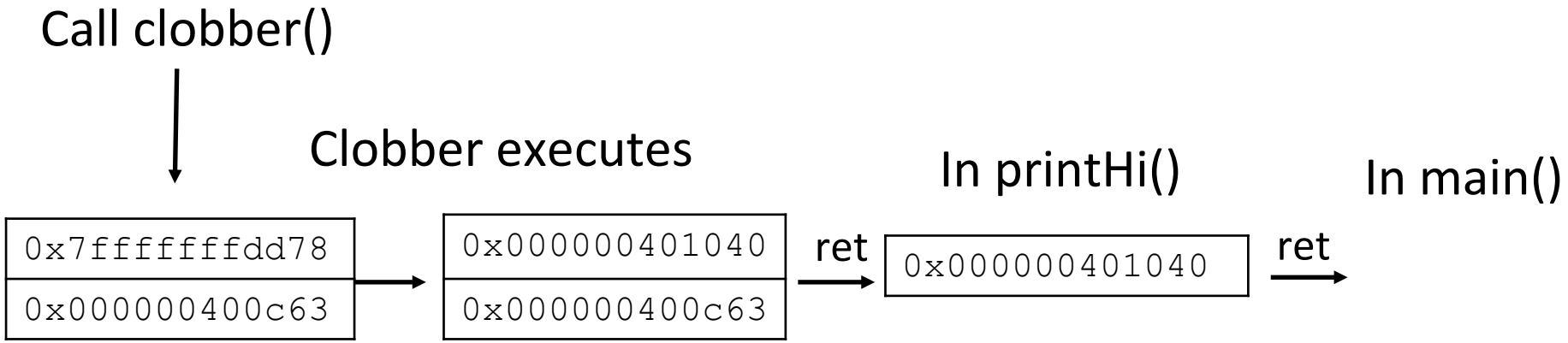
(gdb) x/gx $rsp

Q. Has the return address changed?

(gdb) finish // Should exit and print out “Hi!”

Activity 1 Post

- **Clobber overwrites part of the stack with memory at $rdi, including the all-important return address**
- **In act1 it writes two new return addresses:**
  - 0x401040: address of printHi()
  - 0x400c63: address in main



Activity 2

$gdb act2

(gdb) break clobber

(gdb) run

(gdb) x $rsp

Q. What is the address of the stack and the return address?

(gdb) x /4gx $rdi

Q. What will the new return address be?

(i.e., what is the first value?)

Activity 2 Continued

(gdb) x/5i $rdi + 8 // Display as instructions

Q. Why rdi + 8?

Q. What are the three addresses?

(gdb) break puts

(gdb) break exit

Q. Do these addresses look familiar?

Activity 2 Post

- **Normally programs cannot execute instructions on the stack**
  - Main used mprotect to disable the memory protection for this activity

- **Clobber wrote an address that's on the stack as a return address**
  - Followed by a sequence of instructions
  - Three addresses show up in the exploit:
    - 0x49b259 → "Hi\n" string
    - 0x4023b0 → puts() function
    - 0x401fe0 → exit() function

Activity 3

```
$gdb act3
(gdb) break clobber
(gdb) run
(gdb) x /5gx $rdi
```

Q. Which value will be first on the stack?

Q. At the end of clobber, where will the function return to?

Activity 3 Continued

(gdb) x /2i <return address>

Q. What does this sequence do?

Q. Do the same for the other addresses. Note that some are return addresses and some are for data. When you continue, what will the code now do?

Activity 3 Post

- It's harder to stop programs from running existing pieces of code in the executable.

- Clobber wrote multiple return addresses (aka gadgets) that each performed a small task, along with data that will get popped off the stack while running the gadgets.

  - 0x401a6e: pop %rdi; retq
  - 0x4941f0: Pointer to the string "Hi\n"
  - 0x476397: pop %rax; retq
  - 0x401060: Address of a printing function
  - 0x44ad15: callq *%rax

Activity 3 Post

- Note that some of the return addresses actually cut off bytes from existing instructions

```
465b54:    5b          pop    %rbx
465b55:    5d          pop    %rbp
465b56:    41 5c       pop    %r12
465b58:    41 5d       pop    %r13
465b5a:    41 5e       pop    %r14
465b5c:    41 5f       pop    %r15
465b5e:    c3          retq
465b5f:    90          nop
```

0x465b5c …0c …0d

---

pop %r15 retq
41 5f c3

pop %rdi retq
5f c3

Operation	Register R							
	%rax	%rcx	%rdx	%rbx	%rsp	%rbp	%rsi	%rdi
popq R	58	59	5a	5b	5c	5d	5e	5f

Attack Lab Tools

- gcc –c test.s; objdump –d test.o > test.asm
  - Compiles the assembly code in test.s and shows the actual bytes for the instructions

- ./hex2raw < exploit.txt > converted.txt
  - Convert hex codes in exploit.txt into raw ASCII strings to pass to targets
  - See the writeup for more details on how to use this

- (gdb) display /12gx $rsp (gdb) display /2i $rip
  - Displays 12 elements on the stack and the next 2 instructions to run

GDB is also useful to for tracing to see if an exploit is working

If you get stuck

- **Please read the writeup.** ***Please read the writeup.*** ***Please read the writeup.*** ***Please read the writeup!***
- CS:APP Chapter 3
- View lecture notes and course FAQ at http://www.cs.cmu.edu/~213
- Office hours Sunday through Friday 5:30-9:30pm in GHC 5207
- Post a **private** question on Piazza
- man gdb, gdb's help command