

CSE 127 Discussion 1

1/11/22

Introductions

Zijie Zhao

- BS/MS: 2nd year Master
- Do research on fuzzing and wasm security

Cameron Trando

- Software Engineer at Confluent since 2019
- Started UCSD in 2017, doing BS/MS for fun now
- More of an industry person

Chris Liu

- Resting right now from working hard on the PA :)
- He'll be here next week



Demo

GDB Commands

- `b _main` → add breakpoint
- `info frame` → print info about the current stack frame
- `x/10x $ebp+4` → show as hex
- `x/10i $eip` → show as instructions
- `x/5c name` → show as char
- `x/10xw, x/10xh, x/10xb` → unit size word(4 bytes)/half(2 bytes)/byte
- `disass _main` → disassemble a function
- `tui enable` → enable text user interface
- `layout src/asm` → show source code/assembly
- `tui reg general` → show registers
- `set $ebp = 123` → set value for a register
- `set {int}0xfff12345 = 123` → set value for a memory region
- `run "$(python3 solX.py)"` → run with args generated by solX.py

And there are a lot more...

Stack Management

Instead of thinking of the stack as a tall rectangle, we can think of it as one big array

Memory is read from low to high - consider an int pointer

That's why all the examine gdb commands (x/10x address) reads memory low to high

Stack Management

Once we visualize with that, it becomes apparent why these overflows happen, let's draw a picture

main:

Some code...

call 0x8040...

Vulnerable:

push ebp onto stack

buf[20]

Read into buf...

return

Stack Questions

Why does the stack grow down?

Can we get buffer overflows if the stack grows up?

In the previous example, I used `0x8040...` as an instruction address, is that a typical instruction address?