The story so far

~~Introduction~~

Software security

Host security

Network security

Web security

Today

What is a computer?

Software abstractions

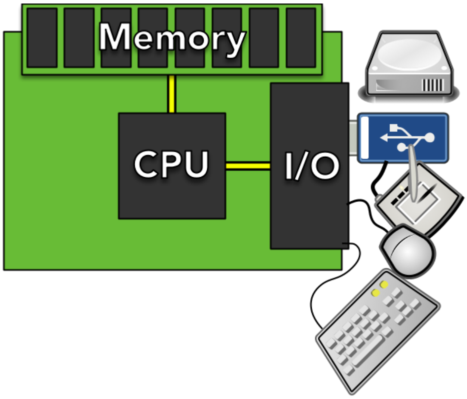
- memory
- execution
- resources

What is a computer?

Model from first year

- CPU to execute instructions
- memory to store information
- external resources

This model isn't *wrong*, just very abstract



A more realistic computer

Complex CPU:

Pipelining, instruction reordering, speculative execution...

Virtual memory:

Address != physical address, page faults, segmantation faults...

Let's look at some demos!

For those playing along at home:

- pointer.cpp
- vm.c
- vm.cpp
- vm.go
- vm.py
- Makefile

Look for:

- Impossibly-large addresses
- Various address ranges
- Program counter
- Space between memory regions
- Arbitrary pointer arithmetic

What did we just see?

Impossibly-large addresses

Various address ranges

Space between memory regions

Arbitrary pointer arithmetic

Memory's not just an array of bytes

A more realistic computer

Complex CPU:

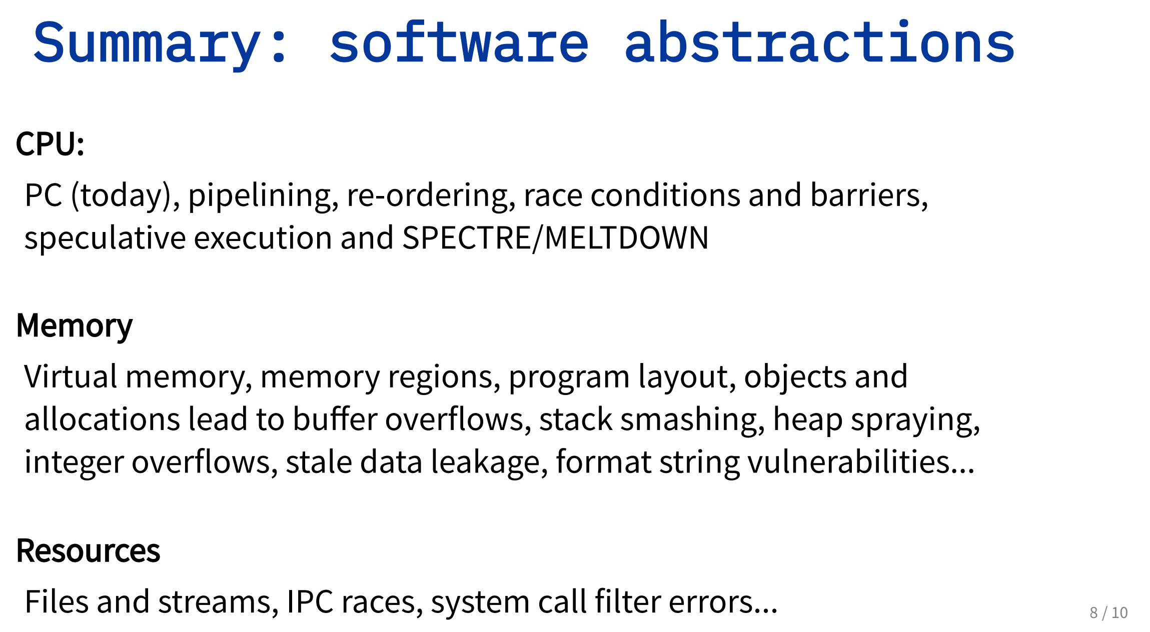
Pipelining, instruction reordering, speculative execution...

Virtual memory:

Address != physical address, page faults, segmantation faults...

External resources:

Files, streams, descriptors... (more detail in ECE 8400 / ENGI 9875)



Summary: software abstractions

CPU:

PC (today), pipelining, re-ordering, race conditions and barriers, speculative execution and SPECTRE/MELTDOWN

Memory

Virtual memory, memory regions, program layout, objects and allocations lead to buffer overflows, stack smashing, heap spraying, integer overflows, stale data leakage, format string vulnerabilities...

Resources

Files and streams, IPC races, system call filter errors...

If we fail to think about software execution in all of its glorious complexity, we run the risk of glossing over critical details.

So pay attention in ______________________________!

Next time

Memory unsafety

- buffer overflows
- stack smashing
- heap spraying