Choosing Technologies

- Different technologies have different security properties
  - Operating systems
  - Languages
  - Object management systems
  - Libraries
- Important to choose wisely
  - Understand the implications of the choice

Choices and Practicalities

- You usually don't get to choose the OS
- The environment you're writing for dictates the choice
  - E.g., commercial software often must be written for Windows
  - Or Linux is the platform in your company
- Might not get choice in other areas, either
  - But exercise it when you can

Operating System Choices

- Rarely an option, and does it matter anyway?
- Probably not, any more
  - All major choices have poor security histories
    - No, Linux is not necessarily safer than Windows
  - All have exhibited lots of problems
  - In many cases, problems are in the apps, anyway
- Exception if you get to choose a really trusted platform
  - E.g., SE Linux or Trusted Solaris
    - Not perfect, but better
    - At a cost in various dimensions

Language Choices

- More likely to be possible
  - Though often hard to switch from what’s already being used
- If you do get the choice, what should it be?

C and C++

- Probably the worst security choice
- Far more susceptible to buffer overflows than other choices
- Also prone to other reliability problems
- Often chosen for efficiency
  – But is efficiency that important for <u>your</u> application?

Java

- Less susceptible to buffer overflows
- Also better error handling than C/C++
- Has special built-in security features
  - Which aren’t widely used
- But has its own set of problems
  - E.g., exception handling issues
  - And issues of inheritance
- 19 serious security flaws between 1996 and 2001
- Multiple serious security problems in recent years

Scripting Languages

- Depends on language
- Many are type safe (or non-typed), limiting buffer overflow possibilities
- Javascript and CGIbin have awful security reputations
- Perl offers some useful security features
- But there are some general issues

Scripting Language Security Issues

- Might be security flaws in their interpreters
  - More likely than in compilers
- Scripts often easily examined by attackers
  - Obscurity of binary is no guarantee, but it is an obstacle
- Scripting languages often used to make system calls
  - Inherently dangerous, esp. things like `eval()`
- If they call libraries, there can be overflows there
  - E.g., Python buffer overflow in 2014
- Many script programmers don’t think about security at all

Open Source vs. Closed Source

- Some argue open source software is inherently more secure
- The "many eyes" argument –
  - Since anyone can look at open source code,
  - More people will examine it
  - Finding more bugs
  - Increasing security

Is the "Many Eyes" Argument Correct?

- Probably not
- At least not in general
- Linux has security bug history similar to Windows
- Other open source projects even worse
  - In many cases, nobody really looks at the code
  - Which is no better than closed source

The Flip Side Argument

- "Hackers can examine open source software and find its flaws"
- Well, Windows' security history is not a recommendation for this view
- Most commonly exploited flaws can be found via black-box approach
  - E.g., typical buffer overflows

The Upshot?

- No solid evidence that open source or closed source produces better security
- Major exception is crypto
  - At least for crypto standards
  - Maybe widely used crypto packages
  - Criticality and limited scope means many eyeballs will really look at it

One More Consideration

- The Snowden leaks suggest some companies put trapdoors in software
  - Especially security-related software
- When it's closed source, nobody else can check that
- When it's open source, maybe they can
  - Emphasis on the "maybe," though