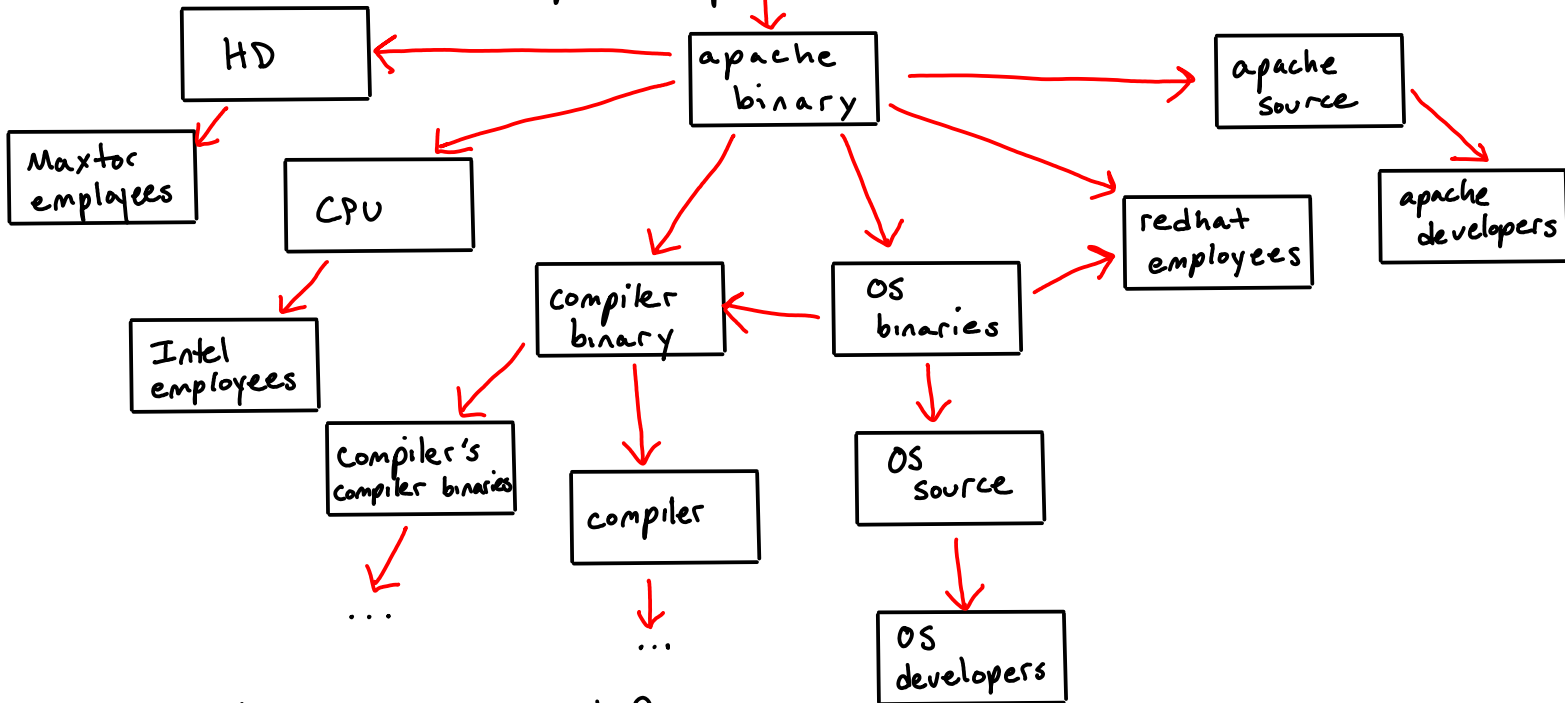


Full disclosure - treatise on locks

Trust == dependence == bad problem: *transitive trust*

example: Apache enforces .htraces



What can we do?

① brute force: do it yourself

② Policies can reduce how many you need to trust

- solves trusting every piece of code ever run on CPU
- still has problem of trusting every Maxtor employee

③ Prioritize

④ Redundancy: buy Maxtor, WD, seagate; run and compare

- before: vulnerable to single failure
- after: vulnerable only to multiple failure

Rule: Keep the TCB small
"Trusted Computing Base" (TCB)

Rule: Economy of mechanism (keep it simple)
100 lines of code more secure than 10,000 lines
short code can be proven correct
eg. passwords in flat file

Rule: Failsafe defaults

default deny vs default allow

- default allow fails silently - you'll never know
 - requires list of all bad things
- default deny fails loudly - people complain

Rule: Least privilege

give each process/user least amount of power to do job

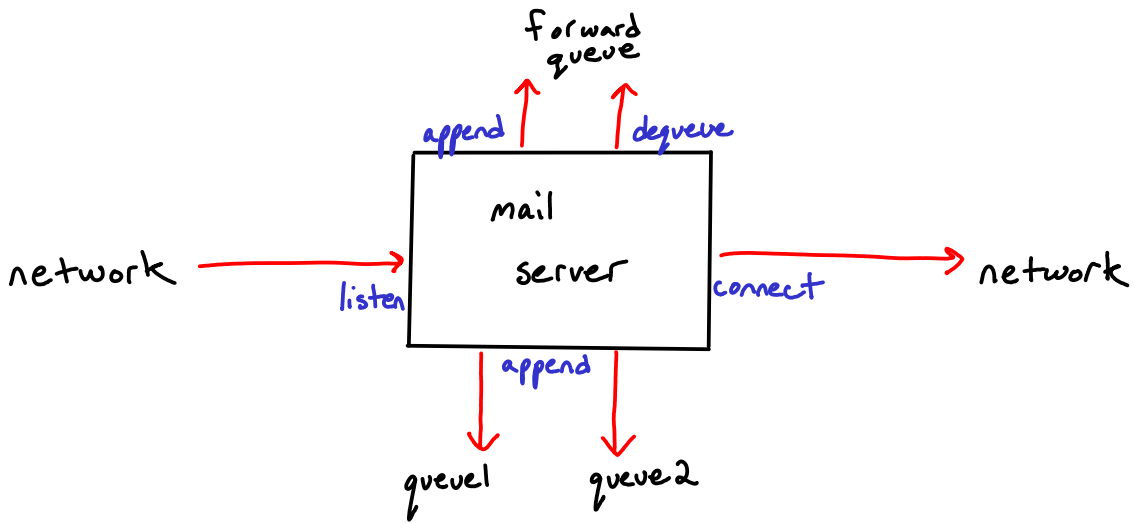
Rule: Complete mediation

if you want to restrict access to an object,
you must restrict accessing that object
Unix: open/read? broke the rule for efficiency

Rule: Separation of privilege

two people in different rooms must turn key at same
time to launch nuclear bomb

mail server



Preventing a worm - separation of privilege

