Model Checking

- Control flow analysis
- Specification: CQUAL - Type vs. Metal - State Machine
- Program = state machine

Specification state machine

- Machine for accepting a language
  - Try to stop: system calls, function calls
- We walk the graph and try to see if it’s possible to get the bad state

Example:

A: set uid(s);

control: flow automations:

B: while (1){

C: Systemexec( );

D: .....more code

E: set uid(s);

Steps:

1. Convert program into state machine
2. Machines: program & state machine
   a. \( L(P) \cap L(S) = \emptyset \)
   b. \( L(P) = \text{final states} \)
3. If no path from state to accept state, then program is good

- Metal ignore data, which can lead to false positives and false negatives
  Example:
  If (flag)
  Setuid(x);
  If(!flag)
  System();
  If(flag)
  Setuid(x);

Flag cant be zero and not zero, but CQUAL doesn’t keep track of data so false positives occur
We can solve the problem by:

**Variable-based state:**
- For each variable in the program store state $S$
  - Issue: copying state from one variable to another
  - Can’t do control and dataflow easily

**Applications:**
- Successful at finding bugs
- Not sound: no interaction between data and control

**Static/Dynamic Defenses**

**Static: CQUAL & Metal:**

**Static Dynamic:**

- Less work for the programmer
- Run time overhead
  - Compile adds a lot of code
- Potential DoS
  - Re-writers are stupid, so they crash programs when a bug is discovered
- Lower False positives
  - Late false positive discovery
- False negative: can go either way depending on implementation