Finding Bugs in Operating Systems

Kai Shen

Dept. of Computer Science, University of Rochester

Finding Bug-related Patterns

- Compiler can check language syntax errors:
  - mismatched "{" and "}"
  - accessing an undefined variable
- Similarly we can check bugs with known patterns:
  - mismatched "disable_interrupt()" and "enable_interrupt()"
  - dereferencing a pointer that was possibly assigned to "NULL"
- But these are a lot more difficult:
  - not expressible in a context-free grammar; requiring flow-sensitive analysis.

Formal Verification – Model Checking

- Formal verification (e.g., model checking) proves the absence of certain bugs by exhaustively explore the whole state space.
- Weakness #1: typically works on toy languages with restricted syntax/semantics
  - e.g., doesn't like loop;
  - not practical for systems written in high-level languages like C.
- Weakness #2: potentially large search space
  - imagine a program with a series of 10 if-then-else statements;
  - not practical for systems with millions of lines of source code like operating system.

Scalable Model Checking

- Weaknesses of model checking:
  - typically works on toy languages with restricted syntax/semantics;
  - potentially large search space.
- Optimizations [Engler et al OSDI'00]:
  - state caching;
  - end loop checking when all reachable states are checked;
  - reduce search scope – function-local analysis only.

Deriving Bug-related Patterns

- How to derive bug patterns?
- Derive correct behaviors:
  - matched "disable_interrupt()" and "enable_interrupt()"
  - never dereferencing a pointer that was possibly assigned to "NULL"
  - files must be opened before use; closed after use; no double close
  - a variable "V" is protected by lock "L"
    - V is only accessed after lock(L) before unlock(L)
- A rumor repeated a thousand times becomes a fact.

Deriving Bug-related Patterns

- Bugs as deviant behavior [Engler et al SOSP'01]
  - find common patterns and call them beliefs
  - but there must be counter examples to be interesting
- For each suspected belief:
  - n total checks, e successes, c=n-e counter examples
- Rank the beliefs:
  - just rank e/n?
  - also consider the number of samples: z-statistic or t-statistic on how far e/n is from a given distribution [Engler et al SOSP'01]

Empirical Results

- A study of bugs in Linux and FreeBSD [Chou et al SOSP'01]
- Where are the errors?
  - driver code has error rates three to seven times higher than code in the rest of the kernel.
- How are bugs distributed?
  - skewed error distribution across files (a logarithmic series distribution).
- How long do bugs live?
  - average bug lifetime is about 1.8 years.
- How do operating system kernels compare?
  - OpenBSD has higher error rates than Linux (1.2 to six times higher error rates for error types examined).