Outline

- RacerX: Effective, Static detection of Race conditions and Deadlocks
  - Dawson Engler and Ken Ashcraft

  - Summary: Tim Faltemier
  - Protagonist: Brett Keck
  - Antagonist:Eric Albert

  - Standby: Christopher Boehnen

Challenges

- Finding errors in large programs (such as Operating Systems), written by a number of independent programmers (who may not fully understand the code) is a hard problem
  - For example, programmer may not understand locking primitives or scope

Problem: Detect deadlocks in programs

- Detecting errors are hard
  - Compiler options, debug print statements, processor speed etc. change execution order
  - Errors may not manifest immediately
  - Errors depend on execution path; depends on hardware configuration (driver path)
  - Language based analysis force coding in same language
  - Dynamic tool depends on program execution path (not all paths are taken)
    - Good thing is that we know which paths to analyze
  - Invasive instrumentation (can slow down)
  - Post mortem log file analysis - less intrusive
  - Model checking - formal code verification
  - Static analysis - offline analysis, bugs that may not occur

Problems

- Annotations help - too much annotations is cumbersome
- RacerX - only annotation needed is to specify locking and other primitives
- Static Interprocedural analysis
- Its fast
- Found some errors in Linux, FreeBSD and System “X”