DyninstAPI
A Binary Instrumentation and Analysis Tool

**Challenges**

- **Mutator** (User Application)
  - **Mutatee Control**
    - Create/attach to a process
  - **Binary Analysis**
    - Present a model of a binary
  - **Instrumentation**
    - Specify what to insert where

- **Binary Analysis**
  - Stripped binaries lack symbols, debug information, or linker relocations.
  - Complex binaries produced by modern compilers:
    - Share code between functions
    - Interleave code and data
    - Frequently use indirect control transfers
    - Overlap instruction sequences
    - Non-returning function calls

- **Platform Independent Interface**
  - Complexity of binaries hidden from users.
  - Platform independent interface.

**Technologies**

- **Process Control**
  - Call Graph
  - Control Flow Graph
  - Instrumentation Points

- **Binary Analysis**
  - Platform Independent Abstractions:
    - Call Graph
    - Control Flow Graph
    - Instrumentation Points
      - Memory instructions
      - Function entry/exit
      - Call sites
      - Loops
      - Arbitrary Instructions

- **Binary Modification**
  - Cross-platform support across architectures and OSs.
  - Static instrumentation on binaries or dynamic instrumentation on processes.
  - Low instrumentation overhead and perturbation.

- **Mutatee** (App Being Instrumented)
  - 1000: mov 1,eax
  - 1005: jmp 101e
  - 1014: mov 3,eax
  - 1019: jmp 101e
  - 101e: push eax
  - 1025: je 1030
  - 102a: jmp *ebx
  - 1030: call exit
  - 1035: nop

- **Trampolines** have overhead of only a single jump and do not perturb uninstrumented parts of application.

- **Create an instrumented binary.**
  - or
  - Write instrumentation to a running process.

Available on Linux-x86/x86_64/ppc32/ppc64, Windows-x86, FreeBSD, Cray, BlueGene

http://www.paradyn.org