Runtime Instrumentation of VxWorks

Jeff Hollingsworth
<hollings@cs.umd.edu>
Ray Chen rchen@cs.umd.edu

Dyninst

VxWorks OS Overview

- General properties
  - Designed for embedded systems
  - Wide range of deployed environments
    - Consumer hardware (Linksys routers)
    - Scientific research equipment
    - Telecommunications Systems
- Ported to many platforms
  - 68K/CPU32, ARM, ColdFire, i960, MIPS, PowerPC, SH, SPARC, x86/Pentium/IA-32, XScale
  - We target PowerPC for the port
- Highly configurable kernel
  - Interactive shell
  - File system

VxWorks Development

- Separate development from runtime
  - Cross compiler on "host" machine
  - Upload binary and execute on "target" machine
- Debugging must involve both systems
  - Functionality provided by Target Agent (target)
  - Physical link managed by Target Server (host)
    - Ethernet, serial, USB, etc.
    - Modular to provide for future

Target Agent

- Compile kernel with target agent
  - Akin to compiling with debug information
- Basic debugging features
  - Reading/writing task registers
  - Reading/writing process memory
  - Event callback system
    - Task creation/deletion
    - Breakpoints
    - Watchpoints
  - Cache flush/invalidate

Dyninst

Target Agent

- Advanced features
  - Loading/launching RTP/kernel tasks
  - Memory disassembler
  - Target function call
  - Symbol query system
    - Includes adding and removing symbols
    - In core memory only
  - Loading modules from host
    - Kernel or real-time process

WTX Protocol

- Protocol for debugging tools
  - Used to send requests to Target Server
- User friendly libraries for 3rd party use
  - C interface libraries provided
  - Integrate easily with modular design of Dyninst
- Allows any Dyninst platform to be a host
  - WTX libraries must exist on platform

Issues Solved

- Understanding use of WTX Protocol
  - WinRiver recently asked us questions about the API!
- Differing Endianness
  - Internal to Dyninst
    - Required some cleanup of internals
  - Public Functions
    - API supported reading un-typed memory
    - Added calls for readInt, writeInt etc.
      - Permits automatic conversion of byte ordering

Approach: Static Analysis

- Loadable Kernel Module Analysis
  - Handles incomplete address information
    - Similar to unlinked object file
- Enough for full SymtabAPI support
  - Function and variable information parsed
  - Execution environment unnecessary
    - No need for target hardware or simulator
  - Similar to opening shared library

Approach: Dynamic Analysis

- Full address information
  - Includes relocation phase of text section
- Control Flow Graph
  - Graph produced for each function found
  - Provides basic block and instruction information
- Force-load additional kernel modules
- Dynamic Instrumentation
  - From function entry/exit down to instruction-level

Dyner Command Line Tool

- Power of Dyninst without the C++ code
- Sample commands:
  - wtxConnect – connect to VxWorks target server
  - wtxPs – process list on VxWorks
  - show modules
  - show functions
  - insert at main entry { printf("Hello world!\n"); }
  - count fooFunc – counts calls to a function
  - run

API for Snippet Compiler

- Goals
  - Provide simple way to generate snippets
  - Refactoring of dyner tool
- Add methods to BPatch_AddressSpace
  - bool generateSnippet(const char *code, BPatch_snippet *&retval);
    - Snippet can be inserted at multiple points
    - Can refer to global variables
  - bool generateSnippet(const char *code, BPatch_point pt, BPatch_snippet *&retval);
    - Can refer to local variables

Dyninst

Current Status

- Dyninst port complete
  - Can parse programs
  - Analyse binaries
  - Insert new code into running programs
  - Attach to running process
  - Analyze binaries from memory image only
- Tools
  - Dyner command line tool running
  - Able to measure program performance

Feature Wish List

- Cross platform SymtabAPI support
  - in one library
- Enable x86 mutatee mode for simulation tests
- Workbench (GUI) integration

Future Work

- Kernel-level Analysis
  - Aggressive branching behavior is non-ABI
  - Affects the ability to manipulate libc functions
- Local variable information
- Extended cross-platform support
  - Goal to have more flexible mutators
- Investigate latency of WTX
  - Mutator effectively 2 indirections away
    - Worse if using a serial line to target
- Experiments using Dyninst

Dyninst