Center for Scalable Application Development Software

Understanding Executables Working Group

Working Group Members

- Drew Bernat
- Robert Cohn
- Mike Fagan
- Jeff Hollingsworth
- Mark Krentel
- Matt Legendre
- John Mellor-Crummey
- Nathan Tallent
- Bill Williams

Instruction Decoding/Encoding

- Discussion topics
  - Intel's Xed
  - Wisconsin's emerging Instruction API

Xed

- Versions
  - version 1 information available at http://rogue.colorado.edu/Pin/docs/13211/Doc/Xed/html
  - version 2 to be released soon
    - Robert Cohn to provide early access to manual for Xed-2 to the Rice and Wisconsin teams
- Capabilities
  - instruction decoding
    - point it at a memory location and ask it to try decoding an instruction at that location
    - Xed fills out a data structures based on the data at the specified memory location, length
  - instruction re-encoding
    - abstract representation simplifies instruction synthesis
    - Xed finds smallest encoding of the desired instruction
- Abstract representation
    - abstract opcode
      - FP, ALU, Branch, …
    - [base, index, displacement, scale]
    - not generic enough to descibe Itanium

Xed Properties

- Appropriate for first-party and 3rd party use
- No global state: it is reentrant, can be used in MT environment
- Using Xed
  - works one instruction at a time
  - the client then is responsible for managing the decoding of sequences Xed
- Itanium decoder/encoder is completely different

Instruction API

- Planned to be more general than Xed in several respects
  - cross platform support
  - client-focused interface
  - abstraction facilities
    - register read/write set
    - frame relevant?
    - etc.
  - aim: extensible representation
    - annotations framework that applies to instructions?
- A plan forward
  - Xed is a promising implementation technology for the x86* flavor of the Instruction API
  - Pin uses a similar abstraction on top of Xed to simplify its work

Analyzing Control Flow in Binaries

- Candidate shared infrastructure: OpenAnalysis
  - Rice/Colorado collaboration on representation-independent program analysis tools
  - currently used for analysis of loops in binaries by HPCToolkit project
  - available at http://developer.berlios.de/projects/openanalysis
- Interests from Intel and Wisconsin
  - iterative refinement of CFGs
    - support use of flow analysis to iteratively refine the CFG representation
    - requirements:
      - CFG representation must support “unknown” targets
        - e.g. destination of edge is node “HELL”
- Potential inputs to switch statement analysis
  - Cristina Cifuentes - de-compiler, binary translation work
    - she has been working switch statement decoding
    - worked on static analysis to support binary translation
  - Gabriel Marin
    - worked on understanding switch statements in EEL

Symtab API

- Capabilities
  - supports incremental addition of information
  - plan to support fast load of precomputed information from files
  - reads out of dwarf to understand
    - where local variables are in frame
    - variables in registers
- UW has draft for symbol table API
- Action items
  - UW will supply draft of the API to Nathan Tallent
  - Rice will provide feedback
    - Rice will look at Alpha ABI procedure descriptors
    - define candidate interface for run-time access to procedure descriptors for stack unwinding
  - possible interest from Pin to use API; Robert will review API spec
- Requirements for unwinding support procedure descriptors
  - for microkernel OS, if this info is computed ahead of time, it needs to be provided in the binary
    - link it in as static data
  - design the API that is agnostic to whether the implementation is eager or lazy
    - on disk, or computed on the fly