Porting DTrace to NetBSD/arm

Ryota Ozaki
ozaki-r@{iij.ad.jp,NetBSD.org}

AsiaBSDCon 2014
March 16, 2014

Who’s am I?

- Ryota Ozaki
- working at IIJ
  - ISP company in Japan
- a NetBSD developer
  - since the last month :)
- an OSS developer
  - find me (ozaki-r) at github or Ohloh

Motivation

- We (IIJ) want DTrace on ARM for our productions
- Not supported yet on NetBSD :-/
  - even on FreeBSD
- Do it by ourselves!
-
- For fun :)

Standing on the shoulders of giants

- Solaris DTrace
- FreeBSD DTrace
- NetBSD DTrace
- DTrace for ARM
  - Some source codes for ARM were already imported into NetBSD
  - written by gonzo@FreeBSD
  - imported by christos@NetBSD

What I need to do

- Fix existing code to make it buildable
  - Support SDT provider
- Support of FBT provider
  - Probable instruction explore
  - Exception handling
  - Instruction emulations
- Others
  - Optimizations
  - Support THUMB instructions in the kernel
  - Support other providers
  - etc.

Supporting FBT provider

Background

- FBT: Function Boundary Tracing
- How it works
  - Preparation
    - Explore probable instructions
    - Replace instructions at the entry and return points of a target function with a breakpoint instruction
      - Preserve original instructions
  - Probing
    - Handle an exception of the breakpoint and probe the runtime context
    - Emulate the replaced instruction
    - Return to the original context
  - Cleanup
    - Restore original instructions to probe points

Supporting FBT provider

Current implementation

- Breakpoint
  - Use *undefined* instructions
- Trap handler
  - added in `undefinedinstruction` of `sys/arch/arm/arm/undefined.c`
- Instruction emulations
  - Written in C
  - Need to optimize in the future

Supporting FBT provider

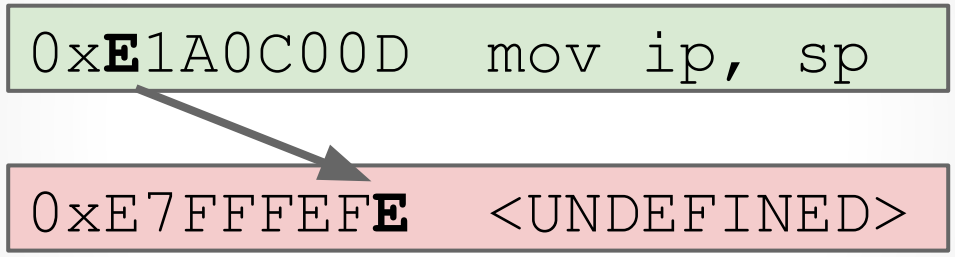
Instruction emulations

- The instruction deconding on ARM is easier than i386/amd64
  - Thanks to constant size of instructions
- ARM allows many instructions to be entry points and return points
- 11 instruction emulations cover ~80% of probe points
  - On amd64 `push` and `retq` emulations can cover most probe points :-/

Supporting FBT provider

Conditional executions

- Encode a *condition specifier* into a breakpoint



- Get a condition specifier from a breakpoint instruction on exception handling
- Run the DTrace probe function only if the condition passes

Current status

- Done
  - Half of my patch (trivial parts) were committed already
  - FBT patches have been commited during the conference :)
- Tested environments
  - -m evbarm (-a earm) kernel=BEAGLEBONE
- Acknowledgment
  - I have to say thank you to `matt` and `christos` for great helps!

Any questions?