The Realities of DTrace on FreeBSD

Jonathan Anderson, **Brian Kidney**, George Neville-Neil,
Arun Thomas, Robert Watson

brian.kidney@mun.ca

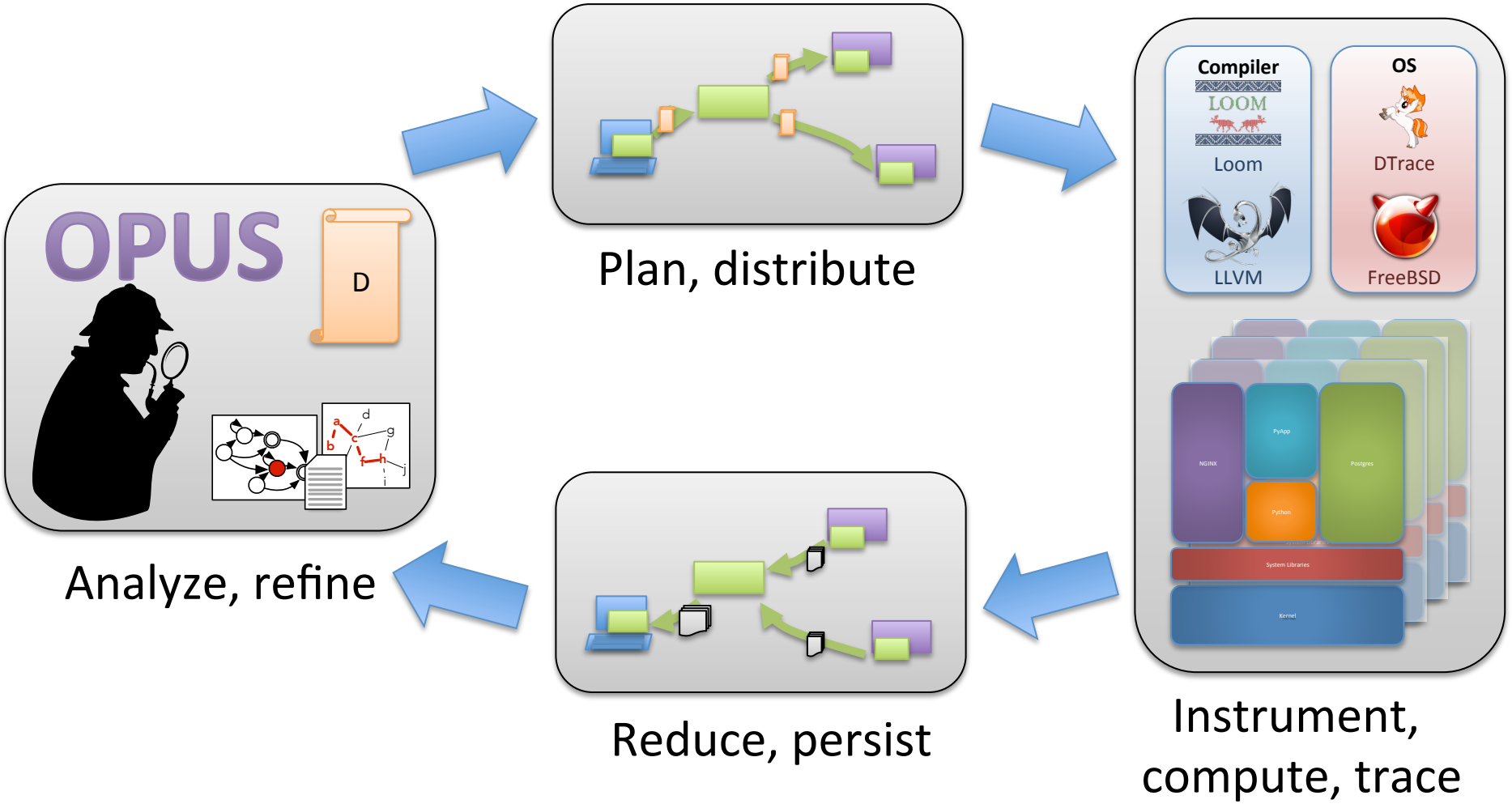
BSDCan
June 9, 2017
Ottawa, ON

Overview

- CADETS Project
- DTrace Improvements
- How CADETS uses DTrace
- Future Improvements

THE CADETS PROJECT

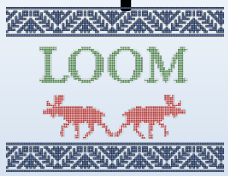
CADETS: Causal, Adaptive, Distributed and Efficient Tracing System



Ground-Up Local Instrumentation

Loom specification-driven program instrumentation

Compiler



Loom

LLVM

LLVM IR fat binaries support JIT (re-) instrumentation

OS

DTrace

FreeBSD

DTrace scriptable full-system dynamic tracing framework

FreeBSD open-source OS extended for transparency

How We Use DTrace

- Leverage DTrace for Security Instrumentation
- Meaning?
  - DTrace is now always on
  - DTrace protections can used against us
  - Improvements need to being made

https://github.com/cadets/freebsd

DTRACE IMPROVEMENTS

DTrace Improvements

- Machine-Readable Output
- D Language Improvements
- New `audit(4)` Provider

Machine-Readable Output

```
# dtrace –n 'syscall::write:entry'

dtrace: description 'syscall::write:entry' matched 2 probes
CPU     ID                    FUNCTION:NAME

  0  59780                       write:entry

  0  59780                       write:entry
```

```
# dtrace -O json -n 'syscall::write:entry'
dtrace: description 'syscall::write:entry' matched 2 probes
CPU     ID                    FUNCTION:NAME
{
  "probe": {
    "timestamp": 3594774042481656,
    "cpu": 1,
    "id": 59780,
    "func": "write",
    "name": "entry"
  }
```

D Language Improvements

- DTrace is scriptable
- D Language
  - C-like language that supports all C operators
  - Structured like awk
  - Supports tread and clause local variables
  - Subroutines to handle common tasks

copyoutmbuf

- Allows for reading chained mbufs in D
- Important for BSD derived network stacks

```
dtrace -n ':::tcp_input:entry { tracemem(copyoutmbuf(*args[0], 64), 64); }'
dtrace: description ':::tcp_input:entry ' matched 1 probe

CPU     ID                    FUNCTION:NAME
  3  34345                 tcp_input:entry
             0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  0123456789abcdef
         0: 45 10 00 34 46 f4 40 00 40 06 70 0a c0 a8 01 01  E..4F.@.@.p.....
        10: c0 a8 01 64 3d 3c 00 16 70 7d fa 57 c2 cc f1 4b  ...d=<..p}.W...K
        20: 80 10 04 10 b5 a5 00 00 01 01 08 0a 3a c8 04 ae  ............:...
        30: 94 3c 09 5f 00 00 00 00 00 00 00 00 00 00 00 00  .<._............
```

D Language Improvements (3)

- if statements
  - D has ternary operator

```
hexval = (c >= '0' && c <= '9') ? c - '0' : (c >= 'a' && c <=
                'z') ? c + 10 - 'a' : c + 10 - 'A';
```

  - if statement improves readability
  - Syntactic sugar imported from Solaris

D Language Improvements (3)

```
vdev_queue_pending_remove:entry {
  if (stringof(args[1]->io_spa->spa_name) == $$1){
    if (args[1]->io_type == ZIO_TYPE_READ) {
      @bytes_read = sum(args[1]->io_size);
    }
    else if (args[1]->io_type == ZIO_TYPE_WRITE
            && args[1]->io_bookmark.zb_level != 2) {
      @bytes_written = sum(args[1]->io_size);
    }
  }
}
```

* Example by Matthew Ahrens

D Language Improvements (4)

```
dtrace:::ERROR{ self->_XD_error = 0x1; }

::vdev_queue_pending_remove:entry{ self->_XD_error = 0x0; }

::vdev_queue_pending_remove:entry /!self->_XD_error/
{ this->_XD_condition1 = 0x1 && stringof(args[1]->io_spa->spa_name) == $$1; }

::vdev_queue_pending_remove:entry /!self->_XD_error/
{ this->_XD_condition2 = this->_XD_condition1 && args[1]->io_type == ZIO_TYPE_READ; }

::vdev_queue_pending_remove:entry /(!self->_XD_error) && this->_XD_condition2/
{ @bytes_read = sum(args[1]->io_size); }

::vdev_queue_pending_remove:entry /!self->_XD_error/
{ this->_XD_condition3 = this->_XD_condition1 && !this->_XD_condition2; }

::vdev_queue_pending_remove:entry /!self->_XD_error/
{ this->_XD_condition4 = this->_XD_condition3 && args[1]->io_type == ZIO_TYPE_WRITE
  && args[1]->io_bookmark.zb_level != 2; }

::vdev_queue_pending_remove:entry /!self->_XD_error && this->_XD_condition4/
{ @bytes_written = sum(args[1]->io_size); }
```

* Example by Matthew Ahrens

Audit Provider

- What is Audit?
  - Subsystem for logging security related events
  - Government Common Criteria security standards
  - Optional component of FreeBSD since 2004

Audit Provider

- What is a provider?
  - DTrace code that provides access to a set of trace points
- What does the provider get us?
  - Access to audit framework data in DTrace…
  - … with filtering and statistics through D.

CADETS AND DTRACE

DTrace Performance Pitfalls?

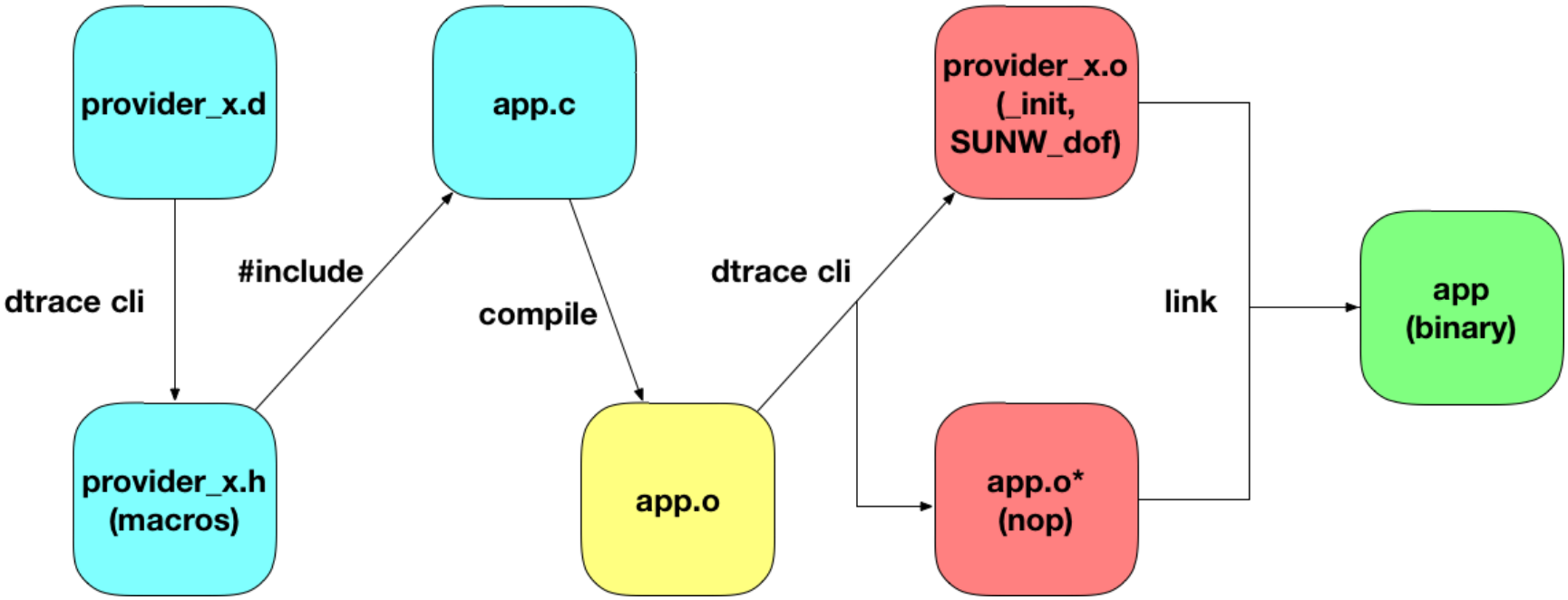
- DTrace shouldn't degrade performance
  - Drops Records
  - Kernel can killed tracing under high load
- Advantage attacker!
  - Flood the system first, then attack
- Solutions
  - Our monitoring cycle
  - Buffer Sizes now configurable with sysctl

FUTURE IMPROVEMENTS

Aside – Loom

- Loom is a instrumentation framework
  - Based on LLVM toolchain
  - Weaves instrumentation into LLVM IR
  - Instrumentation defined in Policy files
  - Instrumentation can be done at any time
    - As long as LLVM IR is available
- We want to use Loom for DTrace probes in Userspace

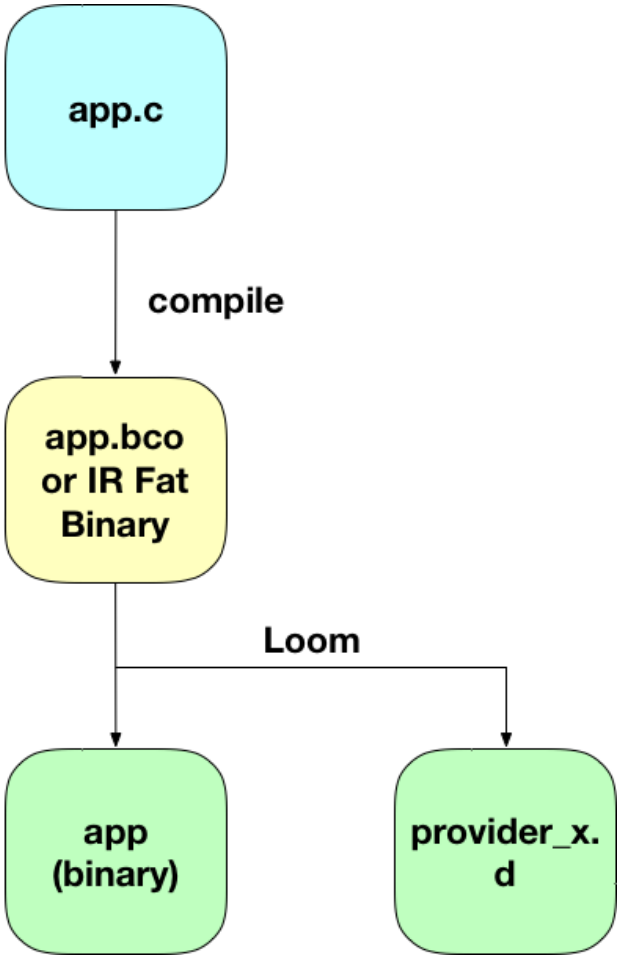
Userland Statically Defined Tracing (USDT)



USDT Performance

- Probes disabled when not tracing
  - Probe site replaced with NOP/function pointer
  - Near zero overhead – theoretically
- Problems
  - DTrace tool modifies binaries
  - Doesn't play well with Make
  - Makes heavy use of relocations

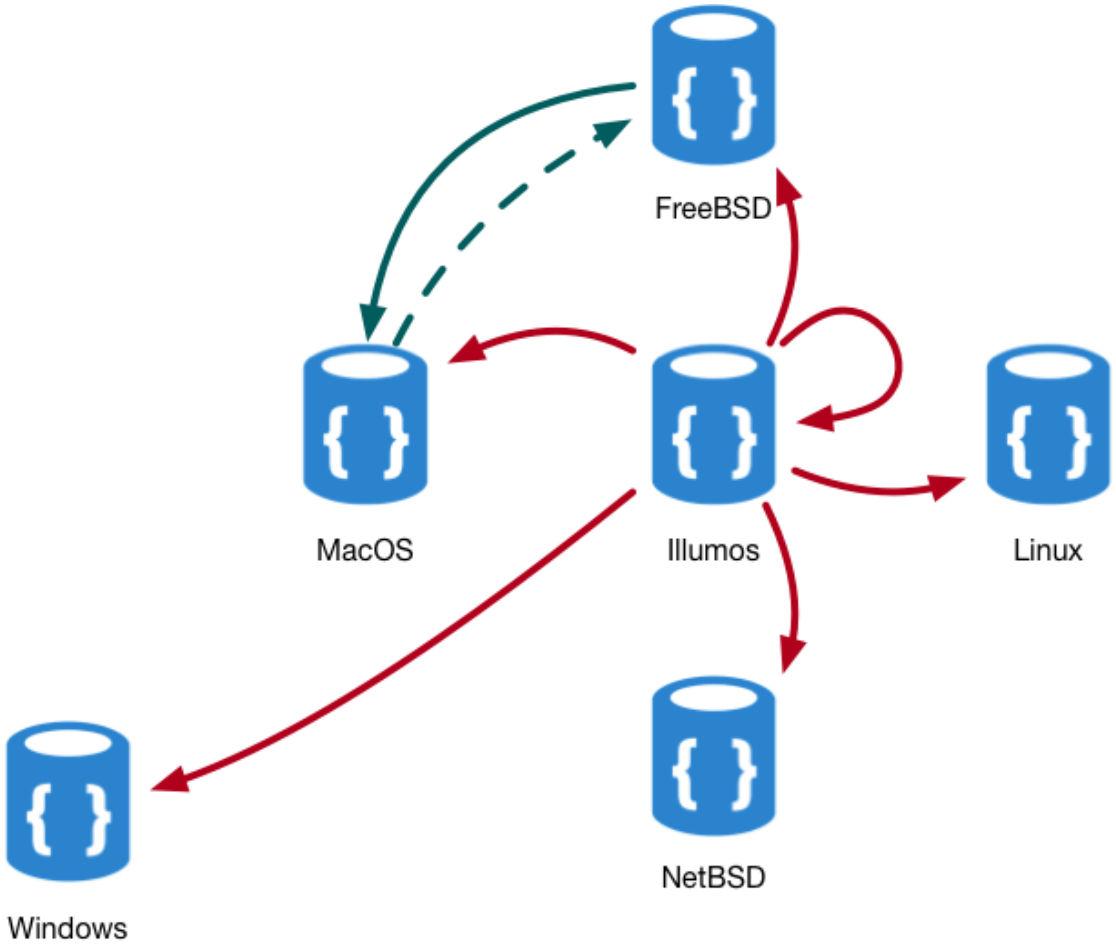
Loom Base Userland Tracing



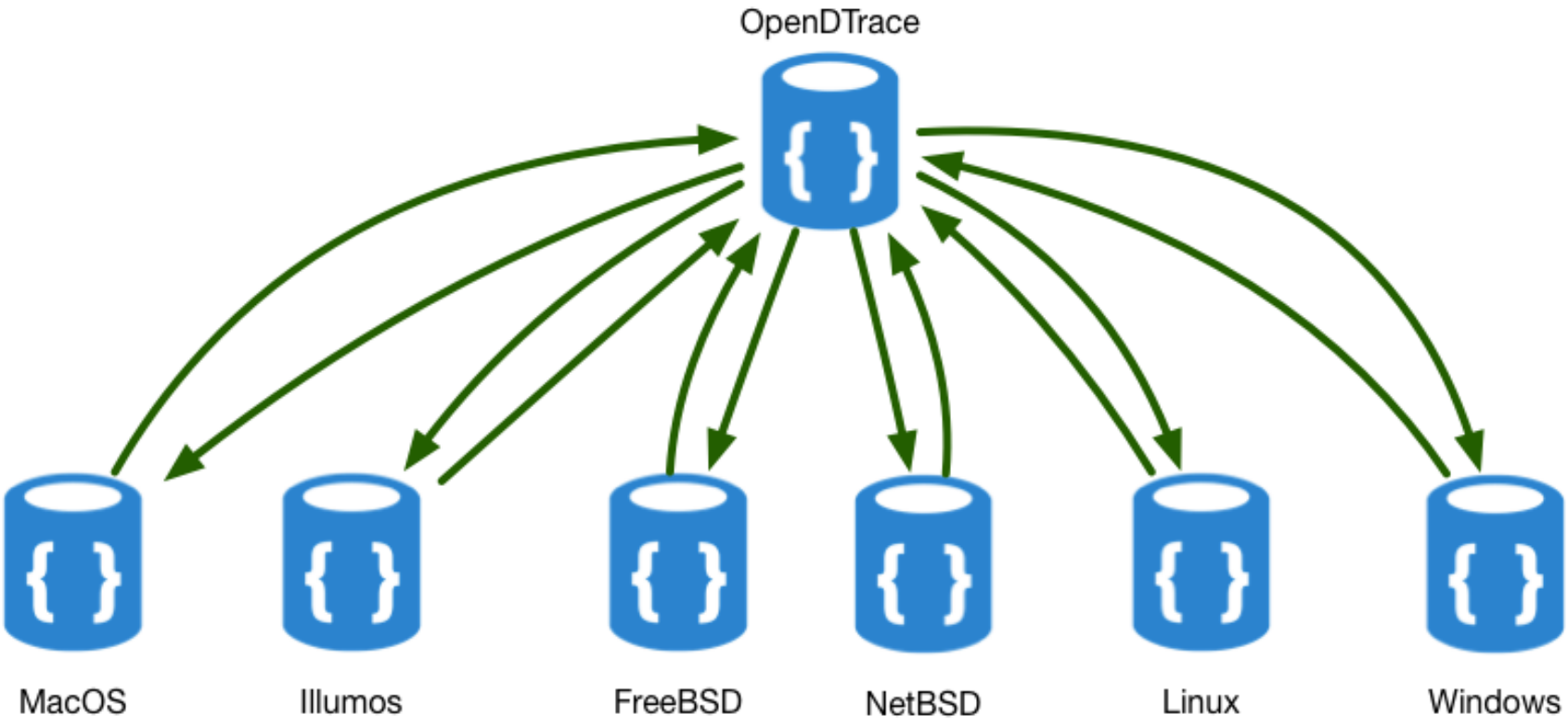
Dynamic Userland Tracing

- Very Early Stages of Development!
  - Prototype system call (dt_probe)
  - Instrumentation via Loom
  - No change to binary when no instrumentation
- To be complete
  - Performance/Overhead testing
  - Provider Generation

DTrace Ecosystem



OpenDTrace Ecosystem



OpenDTrace

- Current Status
  - Code repositories for most distros in place
  - Work being done to create common upstream
  - RFD process for adding new features
  - Now maintains the DTrace Toolkit
  - DTrace Specification of DIF and DOF in progress
    - Better testing of Framework
    - Support new execution substrates (JIT)
    - Make it easier to make future extension

Thank You!

Questions?