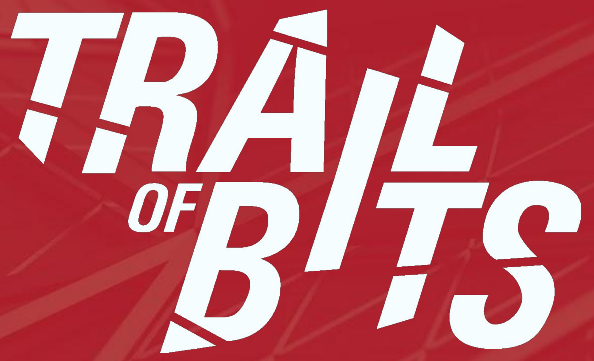


Getting started with osquery

Lauren Pearl & Andy Ying
Engineering team @ Trail of Bits

Agenda

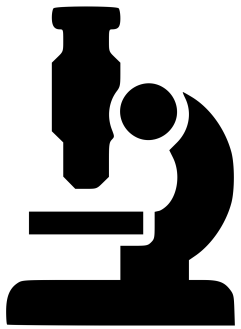
- Trail of Bits
- Intro to osquery
- How to contribute to osquery
- Q&A

Trail of Bits

Cyber security research company - High-end security research with a real--world attacker mentality to reduce risk and fortify code.

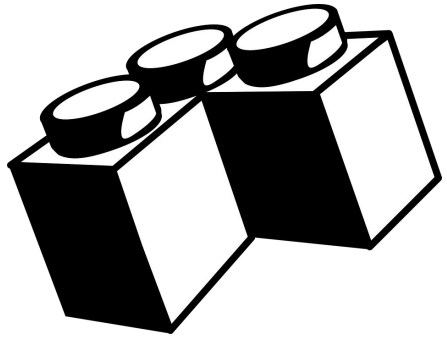
Security Research

- As a leading cybersecurity research provider to DARPA, the Army and the Navy – we create and release open source research tools



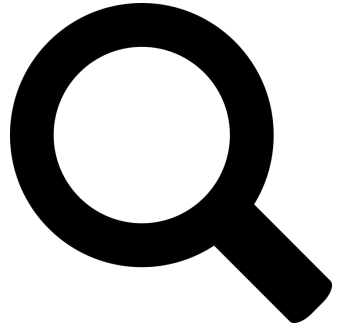
Security Engineering

- We offer custom engineering for every stage of software creation, from initial planning to enhancing the security of completed works



Security Assessments

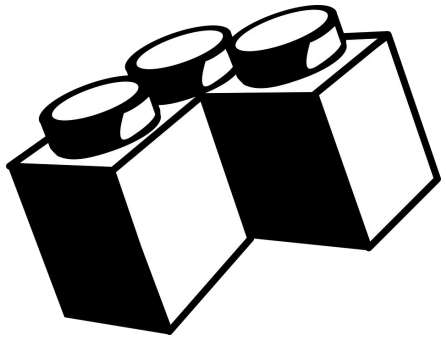
- We offer security auditing for code and systems requiring extreme robustness and niche system expertise



Trail of Bits - Engineering

Security Engineering

- We offer custom engineering for every stage of software creation, from initial planning to enhancing the security of completed works



Open source engineering examples:

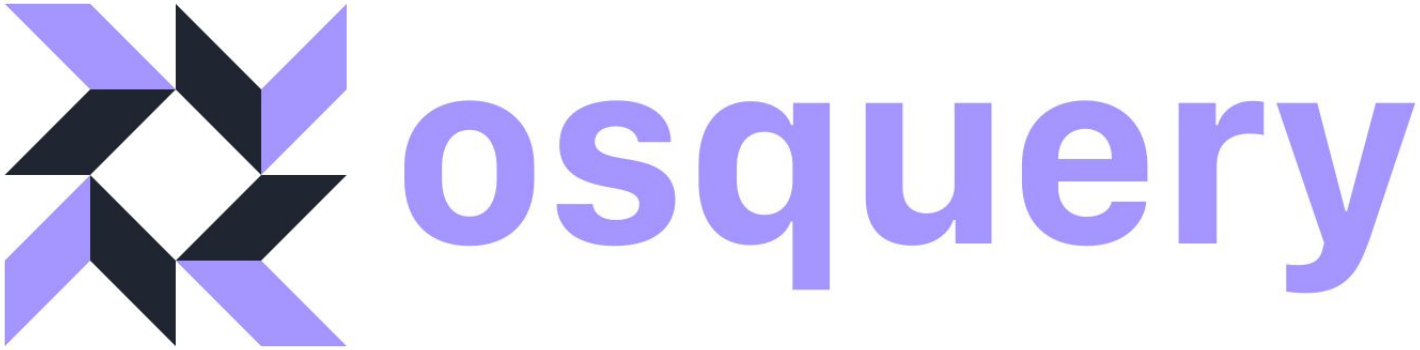
Google Santa

StreamAlert

osquery

osquery: The basics

What is osquery?



Open source endpoint detection tool, developed by **Facebook**, that allows users to query their fleets' system data like they would a relational database.

osquery - What can you ask it?

```
osquery> SELECT * FROM mounts m, disk_encryption d
WHERE m.device_alias = d.name
AND m.path = "/"
AND d.encrypted = 0;

           device = /dev/disk1
     device_alias = /dev/disk1
```

Primary disks that are unencrypted

Disk encryption is an organizational policy for many companies, this query returns hosts whose primary disk is currently unencrypted.

See Available Tables

Other query examples:

- Did anyone download a file called “XYZ” today?
- Tell me all the users that have accessed root
- What processes are running on disk without a binary?

How queries run:

- **Osqueryi:**
  - One-off
- **Osqueryd (daemon):**
  - Scheduled (query once per hour)
  - Event-triggered (FIM, Process Auditing)

How to write queries

Format:

```
> SELECT columns FROM table
  WHERE parameters;
```

Rules of the Road:

- Don't use SELECT * without parameters for a data-heavy table!
  - ~~SELECT * FROM processes;~~
- Read up on SQL constraints!
- Start from queries in QueryPacks, documentation, and blogs!
- Test queries on your local machine before running on your fleet!

Tables & schemas

Currently 226 tables to join!

- Each table has a different schema
- You can view all available tables and their schemas on the [osquery website]
- Or you can find either using in-line commands:

`.schema [TABLE]` **Show the CREATE statements**

`.tables [TABLE]` **List names of tables**

Some popular examples:

- acpi_tables
- arp_cache
- crontab
- file_events
- kernel_info
- listening_ports
- logged_in_users
- mounts
- pci_devices
- processes
- routes
- shell_history
- smbios_tables
- suid_bin
- system_controls
- usb_devices
- users
- groups
- rpm_packages
- apt_sources
- deb_packages
- homebrew_packages
- kernel_modules
- memory_map
- shared_memory
- browser_plugins
- startup_items

Live Example:

Hey osquery, is my machine running hot?

```
SELECT * FROM temperature_sensors;
```

Contributing

Extending osquery - Core or extensions?

Does my contribution belong in Core or in an Extension?

Belongs in Core:

- Observes guiding principles
- Has been shared with and approved by osquery project maintainers as a new feature in Core
- Meets Facebook's testing and quality standards

Belongs in an extension:

- Might not observe the osquery core guiding principles
- Has not been shared with or approved by Facebook as a new feature in Core
- Expands the scope of use for osquery beyond endpoint monitoring
- Integrates with a proprietary or esoteric tool that is not widely applicable

Contributing to Core

Guidelines for contributing features to osquery core

While there are occasional exceptions, contributions to core should abide by the following osquery guiding principles in order to be accepted:

1. osquery doesn't change the state of the system
2. osquery doesn't create network traffic to third parties
3. osquery's endpoint binaries have a light memory footprint
4. osquery minimizes system overhead & maximizes performance
5. The query schema for osquery seeks uniformity between operating systems

Freeze!

Osquery core development will be frozen until 11/19/2018

Contributing to Core

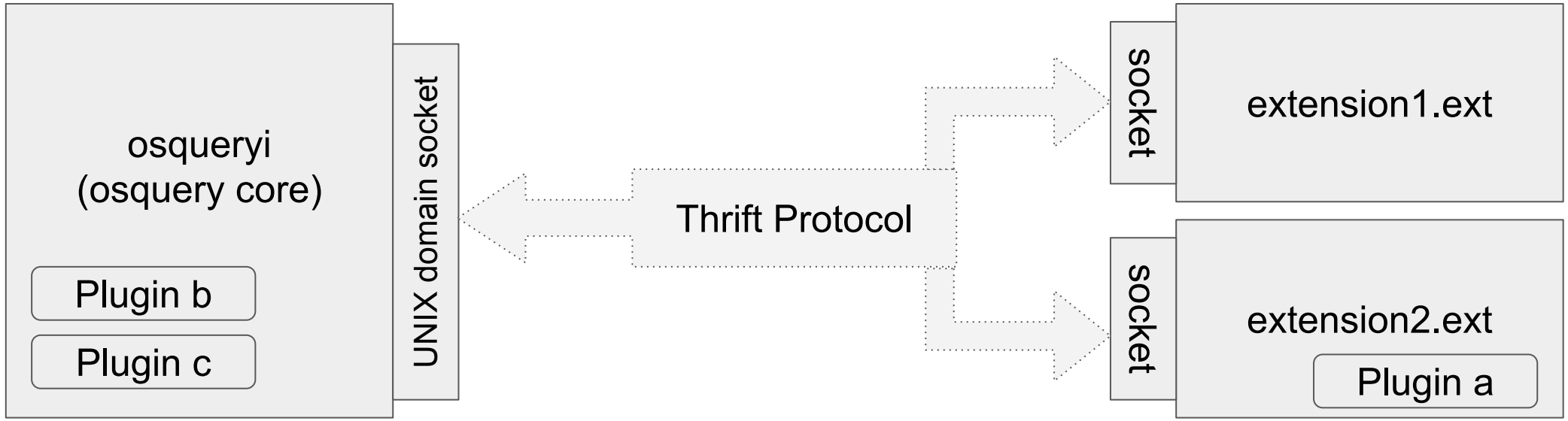
1. Read the CONTRIBUTING.md doc on osquery github repo!
2. Leverage the issue submission templates
3. For significant changes start by opening a Blueprint issue which describes your idea, the problem you're solving and how you plan to implement your solution
4. Single PRs should represent a single body of work - never several unrelated changes (this includes required bug fixes)
5. Start by developing on a feature branch, then submit a pull request against the osquery master branch
6. Tag any related issues in the PR description
7. During testing, submit updated code to comments rather than editing initial PR

Contributing via extensions

Extensions are the way to develop custom functionality in a separate component that *extends* (via virtual tables) or *overrides* (via plugins) the osquery core behavior.

Extensions compile and run as separate executables. They communicate with the osquery core process using the **Thrift** RPC protocol.

osquery extensions




```
osqueryi --extension /path/to/extension1.ext --extension /path/to/extension2.ext
```

Building an osquery extension (C++)

1. Statically link the core code
2. `#include <osquery/sdk.h>`
3. Inherit from `TablePlugin` and implement appropriate methods
4. Register the table or plugin
5. Initialize osquery worker threads
6. `startExtension()` to connect to osquery core
7. Symlink your build directory into osquery's
8. `make externals` or
   `cmake --build . --config Release --target external_EXTENSION_NAME`

Building an osquery extension (Python)

https://github.com/osquery/osquery-python

```
1. > pip install osquery
2. import osquery
3. @osquery.register_plugin
4. class MyTablePlugin(osquery.TablePlugin)
5. def name(), def columns(), def generate()
6. osquery.start_extension(name="my_ex", version="1.0.0")
```

```
osqueryi --extension path_to_my_table_plugin.py
```

Some useful osquery resources

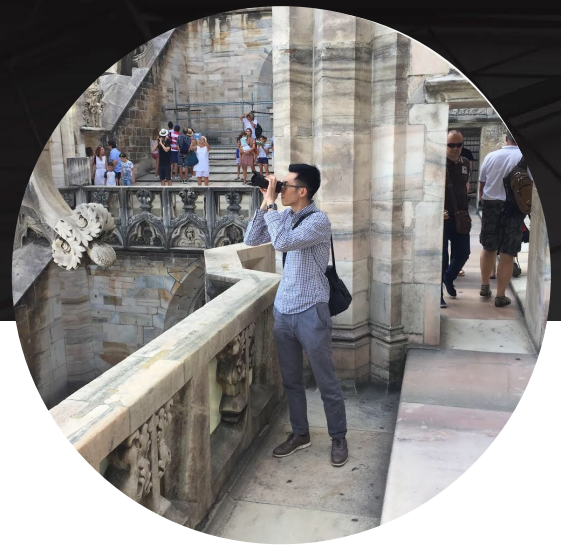
- **Super-simple guide for getting started:** Scott Roberts - *osquery 101 — Getting Started*
  https://medium.com/@sroberts/osquery-101-getting-started-78e063c4e2f7
- **Use cases for the security team:** Chris Long - *osquery For Security*
  https://medium.com/@clong/osquery-for-security-b66fffdf2daf
- **How to build extensions in C++ for Windows, straight from the source:** Nick Anderson - *Building and deploying osquery extensions on Windows*
  https://brewfault.io/blog/2018/1/29/building-extensions-for-osquery-on-windows
- **The future - what osquery users want next:** Trail of Bits - *What do you wish osquery could do?*
  https://blog.trailofbits.com/2018/04/10/what-do-you-wish-osquery-could-do/

Contact Us

Lauren Pearl

Head of Strategy and Ops

lauren@trailofbits.com



Andy Ying

Security Engineer

andy@trailofbits.com