# The osquery Extensions Skunkworks Project

Unconventional Uses for Osquery

Mike Myers
Principal Security Engineer @ Trail of Bits

## osquery 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.

# Terminology: Plugins vs. Extensions

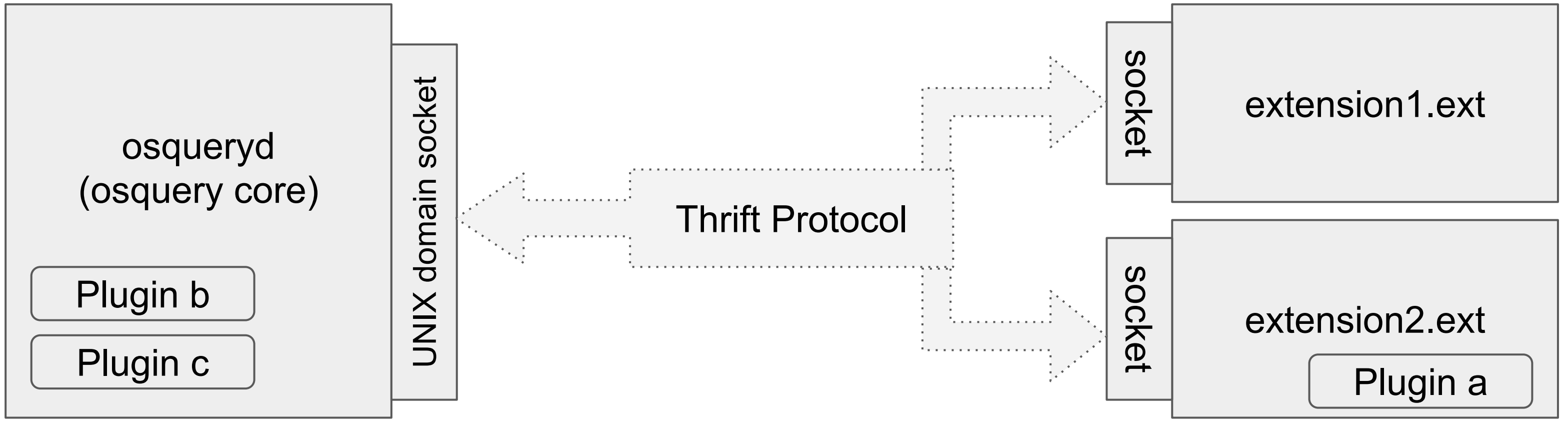
## Plugins

- User-overridable implementations of particular key features
- Query configuration (filesystem *or* TLS server)
- Logging (filesystem, TLS, syslog or windows_event_log, kinesis, firehose, kafka)

## Extensions

- An extension can *implement* a plugin
- More commonly, it implements custom (sometimes proprietary) virtual tables

## osquery extensions



```
cat --disable_extensions=false > /etc/osquery/osquery.flags
```

THEN

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

OR

```
osqueryd --extension_autoload /path/to/list_of_all_extensions.txt.load
```

## 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`

## 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
```

# Building an osquery extension (Go)

https://github.com/kolide/osquery-go

1. install the Kolide library in your GOPATH
2. `import("github.com/kolide/osquery-go")`
3. `server := osquery.NewExtensionManagerServer()`
4. `server.RegisterPlugin(table.NewPlugin(`
   `"foobar", FoobarColumns(), FoobarGenerate))`
5. define the foobar methods to return stuff.

```
go build -o my_table_plugin my_table_plugin.go
osqueryi --extension /path/to/my_table_plugin
```

## osquery extensions enable new horizons

## Why create extensions?

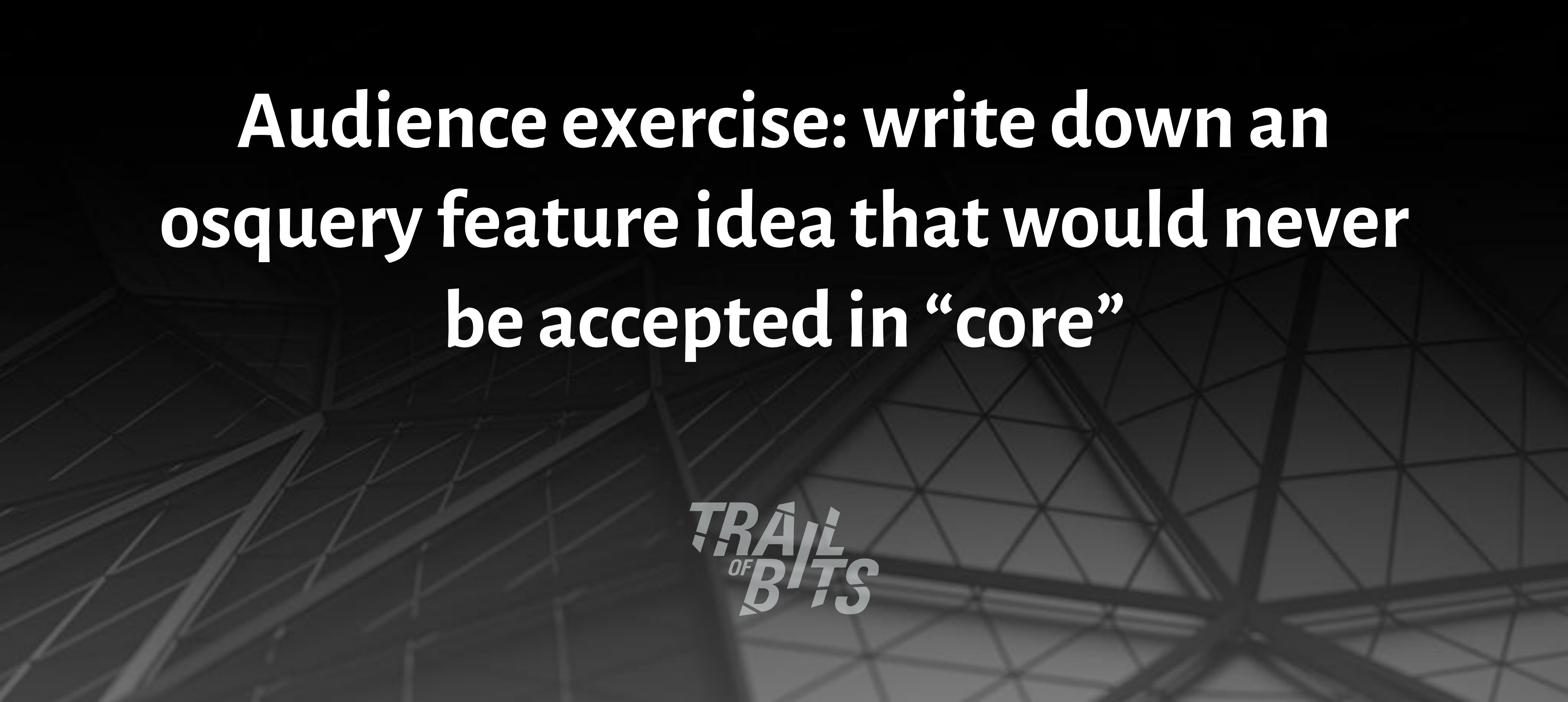
osquery extensions allow you to:

- try “dangerous” features
- ...without the risk of crashing osquery (worker/watchdogs)
- develop with autonomy, outside of core
- ...without maintaining your own fork
- ~~protect secret-sauce ideas~~

## What is not allowed in osquery core?

1. Don't pry into users' data
2. Don't create network traffic to third parties
3. Don't change the state of the system
4. Don't use undocumented APIs or private interfaces
5. Don't use external dependencies
6. Do not fork a new process

## Audience exercise: write down an osquery feature idea that would never be accepted in "core"



## Trail of Bits' extensions repository

https://github.com/trailofbits/osquery-extensions

Skunkworks Project: any small group working on radical ideas outside of standard procedures & constraints.

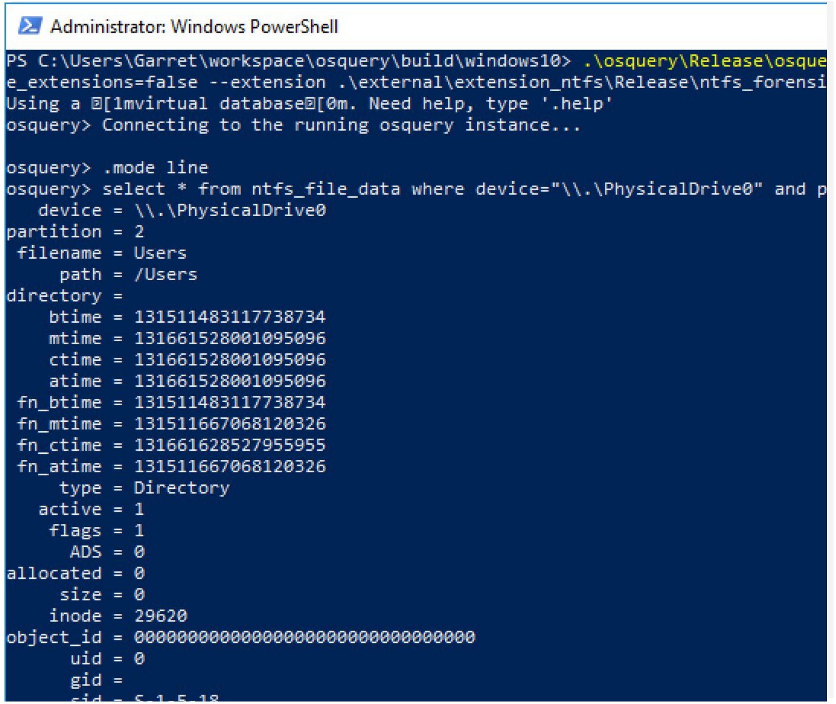
Contributing engineers at Trail of Bits:

- Alessandro Gario
- Garret Reece

## Extension highlight: NTFS forensics

### Remotely query NTFS metadata

- **Metadata valuable in incident response**
  - Additional timestamp entries, file security descriptors, whether a file has Alternate Data Streams (ADS).
- **NTFS filesystem forensics**
  - Index entries for directory indices, including entries that are deallocated
- **Accelerated response**
  - Remote investigations with the free osquery that you've already deployed



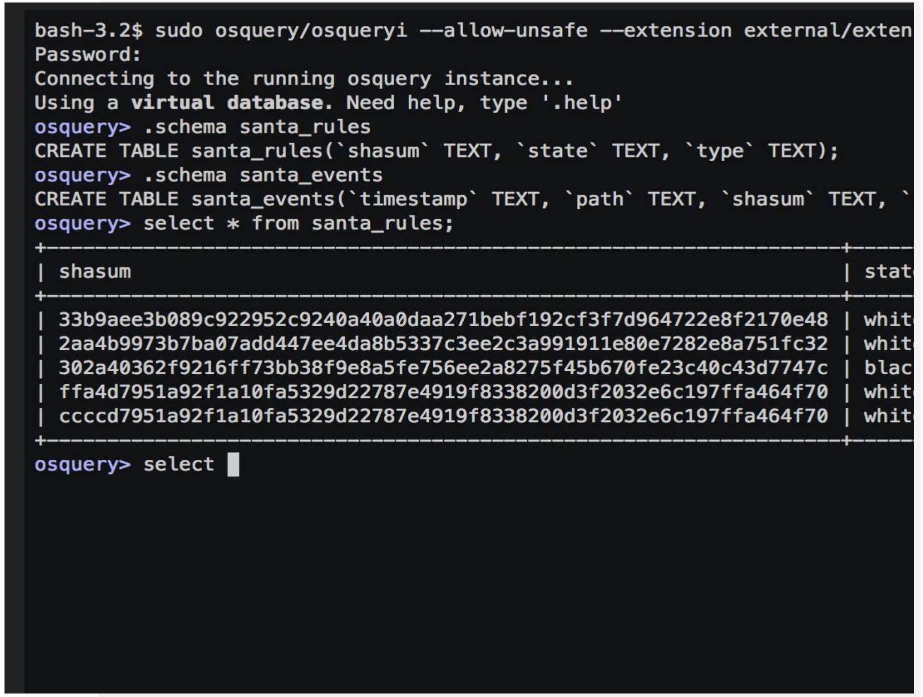
```
PS C:\Users\Garret\workspace\osquery\build\windows10> .\osquery\Release\osque
e_extensions=false --extension .\external\extension_ntfs\Release\ntfs_forensi
Using a ?[1mvirtual database?[0m. Need help, type '.help'
osquery> Connecting to the running osquery instance...

osquery> .mode line
osquery> select * from ntfs_file_data where device="\\.\PhysicalDrive0" and p
   device = \\.\PhysicalDrive0
partition = 2
 filename = Users
     path = /Users
directory =
    btime = 131511483117738734
    mtime = 131661528001095096
    ctime = 131661528001095096
    atime = 131661528001095096
 fn_btime = 131511483117738734
 fn_mtime = 131511667068120326
 fn_ctime = 131661628527955955
 fn_atime = 131511667068120326
     type = Directory
   active = 1
    flags = 1
      ADS = 0
allocated = 0
     size = 0
    inode = 29620
object_id = 00000000000000000000000000000000
      uid = 0
      gid =
      sid = S-1-5-18
```

CRYPSIS™

## Extension highlight: manage Santa whitelist

### Track and manage application whitelisting on macOS

- **See applications that tried to run**
  - See how Santa enforces its whitelist or blacklist of allowed executions.
- **Illustrates writeable tables PR #4094**
  - Not only reads Santa events log, but can also update Santa rules.
- **No longer need separate server**
  - Reduced operating overhead. Single interface: osquery.

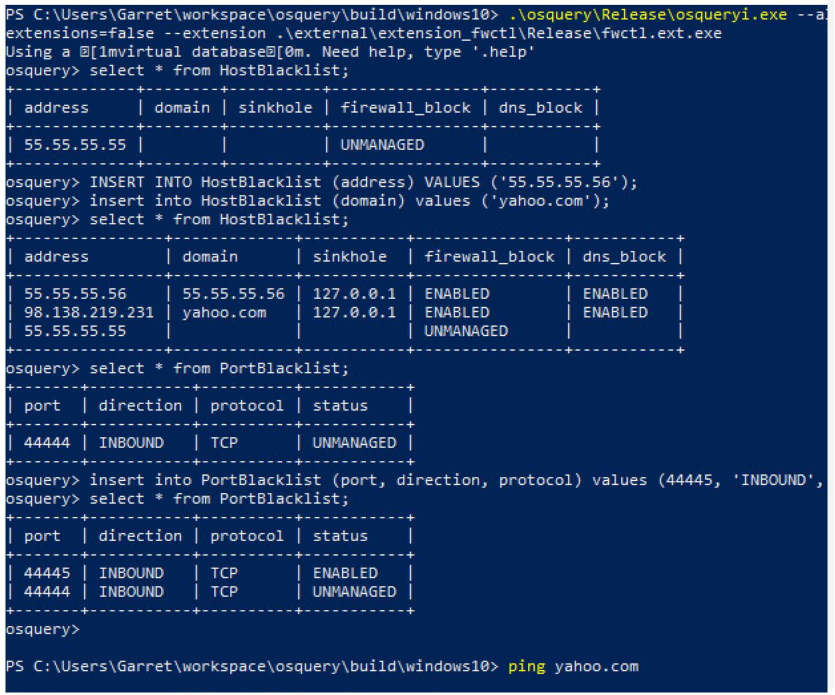


```
bash-3.2$ sudo osquery/osqueryi --allow-unsafe --extension external/exten
Password:
Connecting to the running osquery instance...
Using a virtual database. Need help, type '.help'
osquery> .schema santa_rules
CREATE TABLE santa_rules(`shasum` TEXT, `state` TEXT, `type` TEXT);
osquery> .schema santa_events
CREATE TABLE santa_events(`timestamp` TEXT, `path` TEXT, `shasum` TEXT, `
osquery> select * from santa_rules;
+------------------------------------------------------------------+-----
| shasum                                                           | stat
+------------------------------------------------------------------+-----
| 33b9aee3b089c922952c9240a40a0daa271bebf192cf3f7d964722e8f2170e48 | whit
| 2aa4b9973b7ba07add447ee4da8b5337c3ee2c3a991911e80e7282e8a751fc32 | whit
| 302a40362f9216ff73bb38f9e8a5fe756ee2a8275f45b670fe23c40c43d7747c | blac
| ffa4d7951a92f1a10fa5329d22787e4919f8338200d3f2032e6c197ffa464f70 | whit
| ccccd7951a92f1a10fa5329d22787e4919f8338200d3f2032e6c197ffa464f70 | whit
+------------------------------------------------------------------+-----
osquery> select
```

## Extension highlight: manage the OS firewall

Simple, cross-platform interface to endpoints' host-based firewalls

- **Block hosts or check blocked hosts**
  - Turns /etc/hosts into a virtual table.
  - Again, illustrates PR #4094
- **Block ports or check blocked ports**
  - These common firewall management tasks are now simple SQL syntax.
- **Single interface across macOS / Linux / Windows**
  - Accelerate incident response
  - Ensure policy compliance

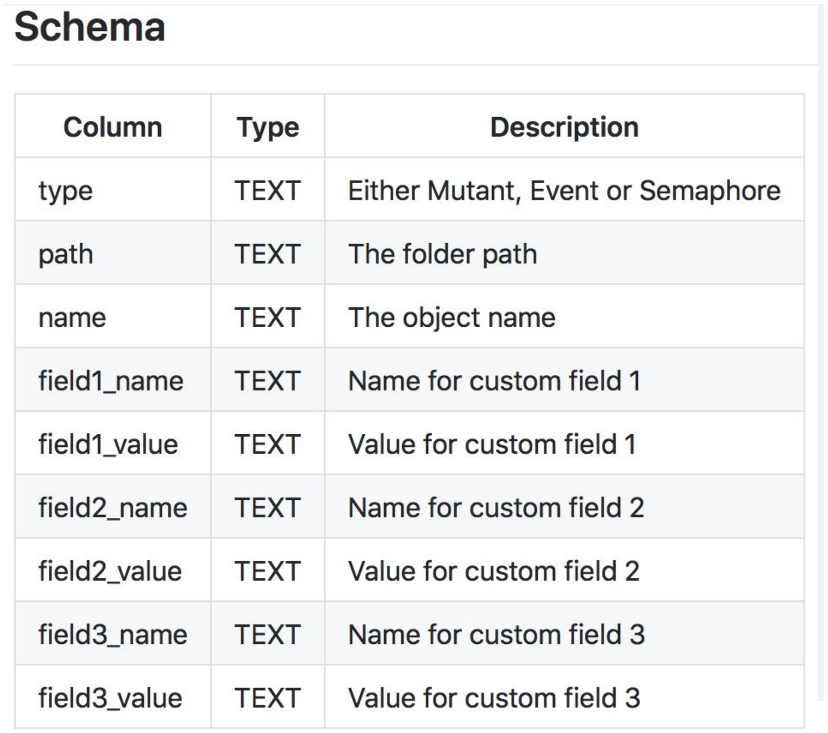


```
PS C:\Users\Garret\workspace\osquery\build\windows10> .\osquery\Release\osqueryi.exe --a
extensions=false --extension .\external\extension_fwctl\Release\fwctl.ext.exe
Using a [1mvirtual database[0m. Need help, type '.help'
osquery> select * from HostBlacklist;
+-------------+--------+----------+-----------------+-----------+
| address     | domain | sinkhole | firewall_block  | dns_block |
+-------------+--------+----------+-----------------+-----------+
| 55.55.55.55 |        |          | UNMANAGED       |           |
+-------------+--------+----------+-----------------+-----------+
osquery> INSERT INTO HostBlacklist (address) VALUES ('55.55.55.56');
osquery> insert into HostBlacklist (domain) values ('yahoo.com');
osquery> select * from HostBlacklist;
+----------------+-------------+-----------+-----------------+-----------+
| address        | domain      | sinkhole  | firewall_block  | dns_block |
+----------------+-------------+-----------+-----------------+-----------+
| 55.55.55.56    | 55.55.55.56 | 127.0.0.1 | ENABLED         | ENABLED   |
| 98.138.219.231 | yahoo.com   | 127.0.0.1 | ENABLED         | ENABLED   |
| 55.55.55.55    |             |           | UNMANAGED       |           |
+----------------+-------------+-----------+-----------------+-----------+
osquery> select * from PortBlacklist;
+-------+-----------+----------+-----------+
| port  | direction | protocol | status    |
+-------+-----------+----------+-----------+
| 44444 | INBOUND   | TCP      | UNMANAGED |
+-------+-----------+----------+-----------+
osquery> insert into PortBlacklist (port, direction, protocol) values (44445, 'INBOUND',
osquery> select * from PortBlacklist;
+-------+-----------+----------+-----------+
| port  | direction | protocol | status    |
+-------+-----------+----------+-----------+
| 44445 | INBOUND   | TCP      | ENABLED   |
| 44444 | INBOUND   | TCP      | UNMANAGED |
+-------+-----------+----------+-----------+
osquery>

PS C:\Users\Garret\workspace\osquery\build\windows10> ping yahoo.com
```

# Extension highlight: synchronization objects

## Windows objects: mutants, semaphores, and ‘events’

- **Useful in incident response**
  - Malware infection markers
- **Vaccination via mutant squatting**
  - Threat intelligence → vaccinate
- **Edge browser tabs**
  - Edge browser creates named mutants for every cross-site domain you’re currently browsing (find XSS and iFrames)



**Schema**

| Column | Type | Description |
|---|---|---|
| type | TEXT | Either Mutant, Event or Semaphore |
| path | TEXT | The folder path |
| name | TEXT | The object name |
| field1_name | TEXT | Name for custom field 1 |
| field1_value | TEXT | Value for custom field 1 |
| field2_name | TEXT | Name for custom field 2 |
| field2_value | TEXT | Value for custom field 2 |
| field3_name | TEXT | Name for custom field 3 |
| field3_value | TEXT | Value for custom field 3 |

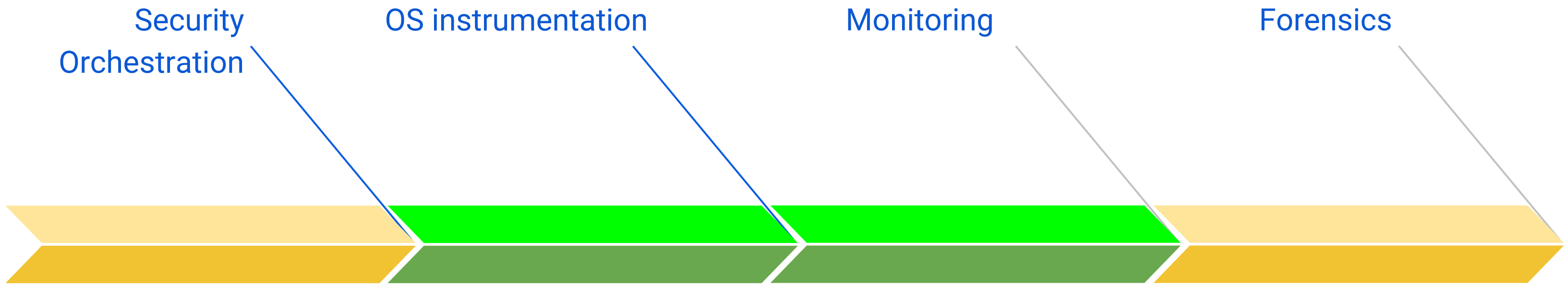
## Improving extension performance

- PR #4335 (tagged 3.3.0)
- Extension must call new `add_osquery_extension_ex()` function
- Reduced overhead from Thrift
- Fewer processes running
- Will enable bundling extensions with, *e.g.*, Kolide Launcher

# Who else is creating extensions?

| Organization | Extension functionality | Availability |
|---|---|---|
| Facebook | tables that implement proprietary detection strategies | Proprietary |
| Kolide | adds Go language support bridge for the osquery SDK; others bundled with Launcher | Open-source |
| PolyLogyx | traditional and event-driven tables; includes a kernel driver | Closed-source but free |
| SpellSecurity Labs | traditional and event-driven tables; includes a kernel driver. Also implements an osquery software update mechanism. | Commercial |
| **Trail of Bits** | **new category of management-capable tables; integration with other open-source endpoint security tools and libraries** | **Open-source** |

# Attack detection and response timelines



**Manage rules & settings**

Control Santa, firewall, etc.

**osquery core**

SELECT * FROM settings

**Event-driven**

Auditing, eventing, maybe eBPF (tracing)

**Leverage 3rd party code**

Memory capture/ forensics, filesystem metadata

## Q&A

## Challenges & Opportunities

TRAIL OF BITS

# Extension limitations & caveats

## Lack of pub/sub

No integration with event pub/sub in osquery core. This means a lack of the cache (backing store) provided by RocksDB, and no support for certain CLI controls. Your extension must implement its own database (easy in Go, not as easy in C++). Also, no expiry (event expiration).

## Thrift overhead

If transferring a large quantity of table data, the Thrift serialization/ deserialization causes performance degradation (poor event-driven table performance in extensions). You can mitigate this by throttling events.

## No result streaming

Extensions don't have a keepalive mechanism while busy generating table data, so osquery core is prone to complaining about timeouts on queries of tables implemented in extensions. Annoyance, not a stopper?

## Stretching the SQL table metaphor

Inherent Challenges:

- Translating complex rule data to tabular format
- Cannot always flatten hierarchical data
- Mutant table: each object has different properties

Writable Tables Challenges:

- Translating back from SQL format to update rule-driven configuration files
- Synchronizing state (no reusing row IDs)

## Call to action

### Develop your idea

Do you have an idea for an osquery extension? File an issue on our GitHub repo for it. Or, develop it yourself using the osquery SDK.

### Make a pull request

Get our feedback on your extension. If we think it's great and you want it featured, we can add it to our growing repo of extensions.

### You can hire us

Contact us to discuss an estimate for implementing your idea. We work in the open, engage with the community, and stay in regular contact with our customers.

# Contact Us

**Dan Guido**
CEO

dan@trailofbits.com
516.359.3208

**Mike Myers**
Principal Security Engineer

mike.myers@trailofbits.com
708.374.7853 on Signal