

# OS X Security at Scale

*mike arpaia / facebook*

*ted reed / facebook*

# OS X security at Facebook

production hardening



client engineering



intrusion detection



# catch attackers

“detection” and “response”

- insider threats
  - espionage
- external threats
  - APT
  - hacktivists
  - mass malware
  - the list is endless

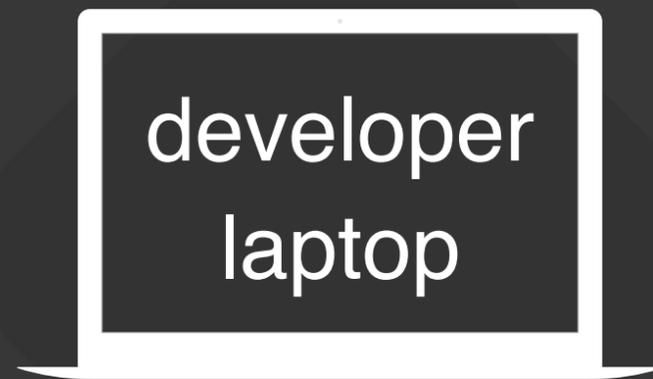
# single intrusion detection team

defend enterprise and production infra

- extract as much signal as possible
  - make high confidence decisions
  - harder for the more variable OS X client fleet
- avoid duplication in production
  - ease burden for humans
  - apply the same intelligence feeds
  - reuse storage

# focus on client machines

mac and linux laptops



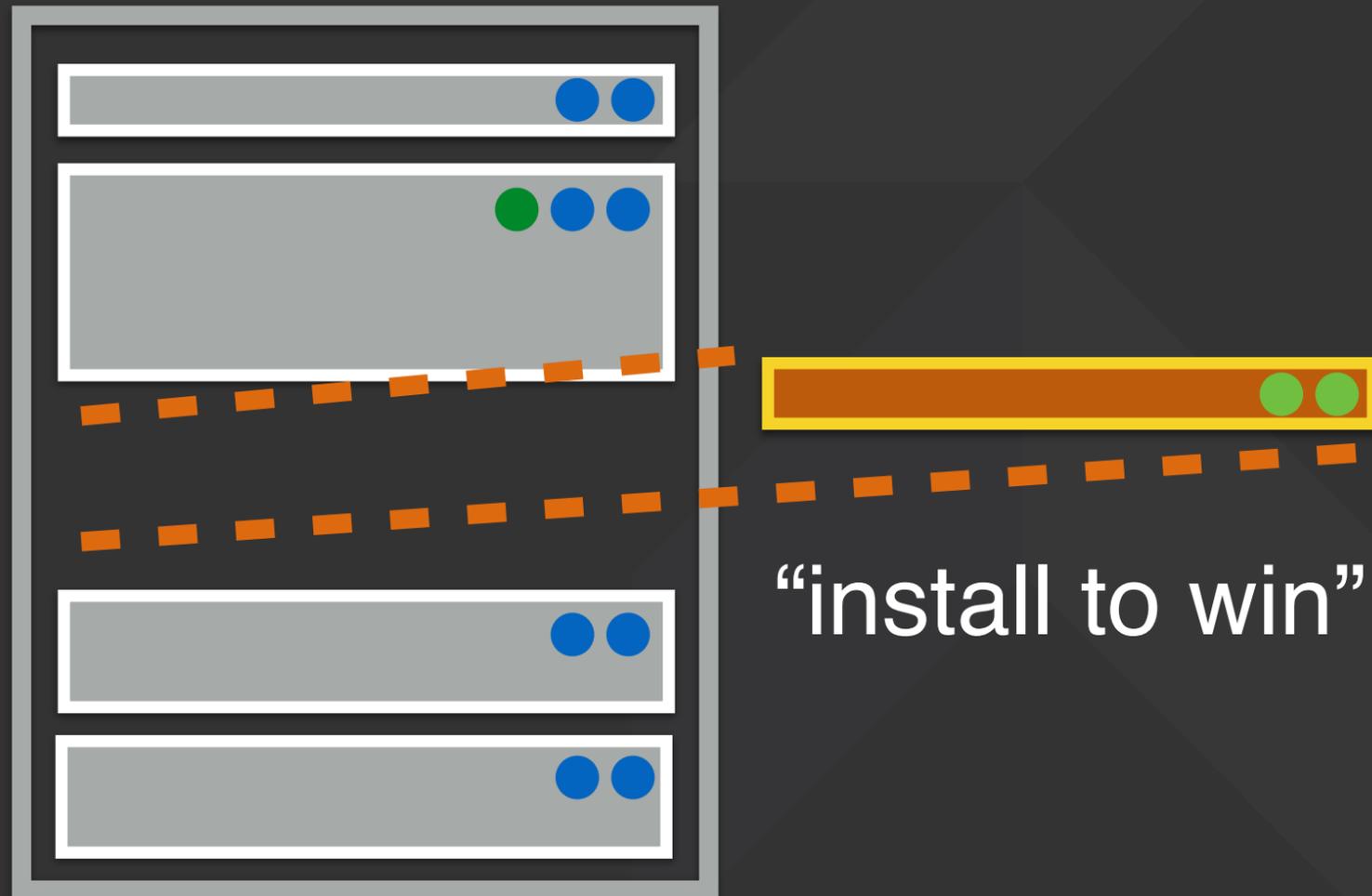
Most variable

*'Highest'* risk

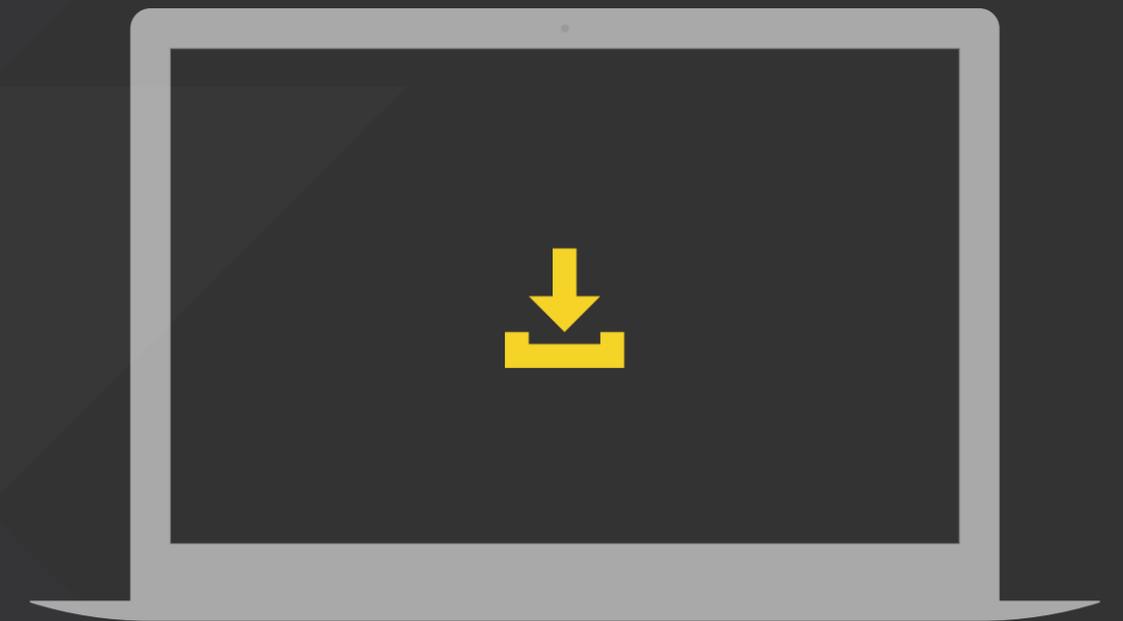
Largest attack surface

# but it's a hard problem

network-based IDS

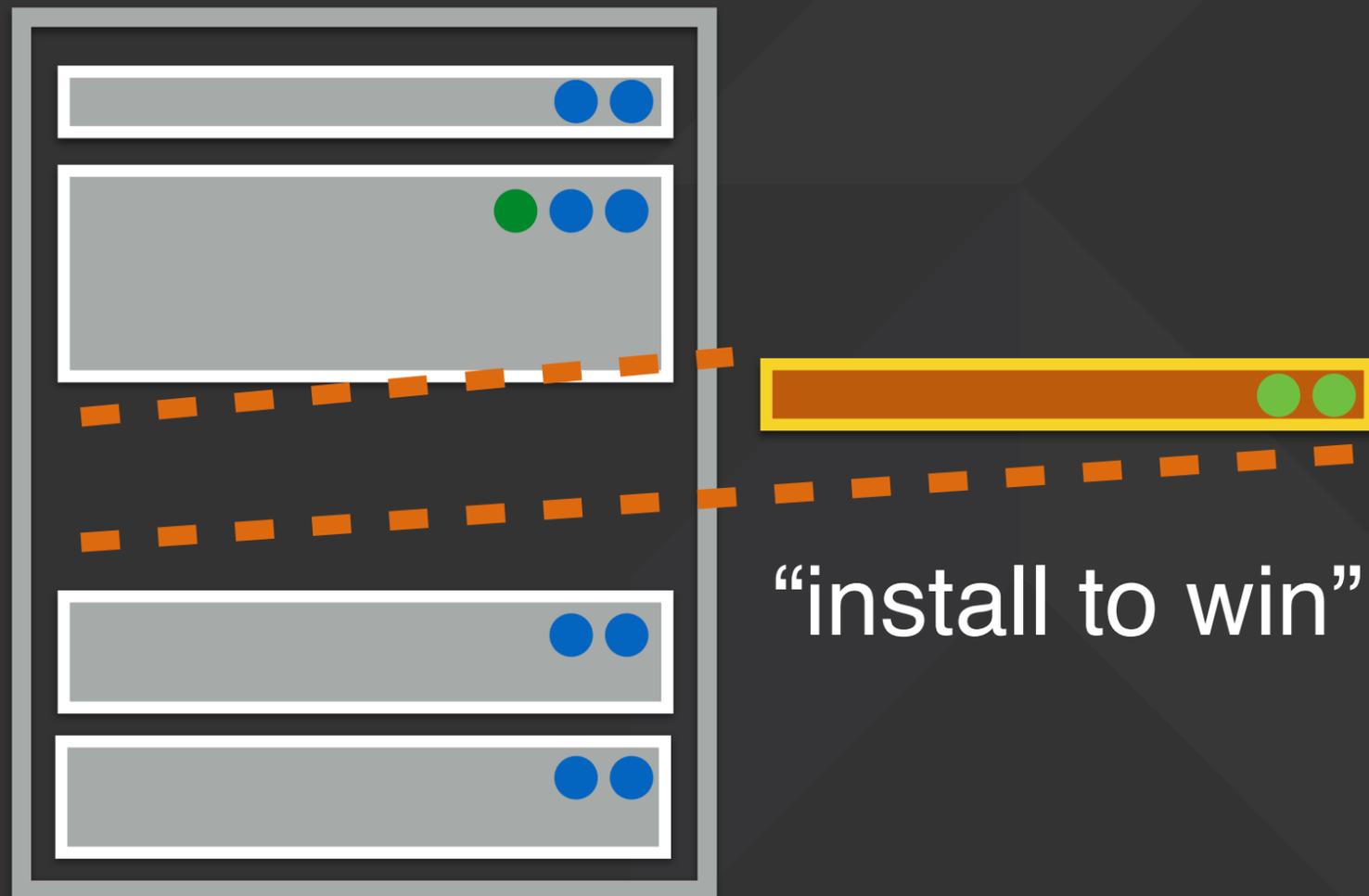


host-based IDS

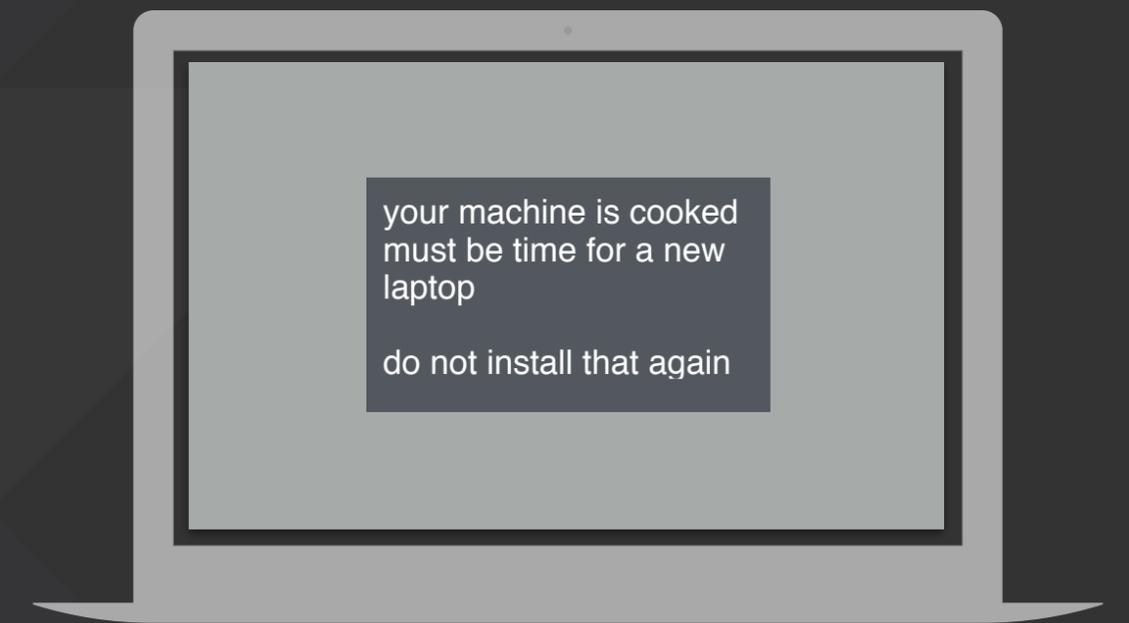


# but it's a hard problem

network-based IDS



host-based IDS



"install and pray"

# we live in a windows centric world

but, times are changing

- more OS X laptops
- most production infrastructure runs on Linux
- few are instrumenting their OS X and Linux hosts
  - affordably
  - tailored to medium enterprises or large infrastructures
  - how would we solve that problem?



**desired  
properties**

upgrades

**simple**

deployable

configurable

development

**easy**

low maintenance

integrations

long uptime

automation

metrics

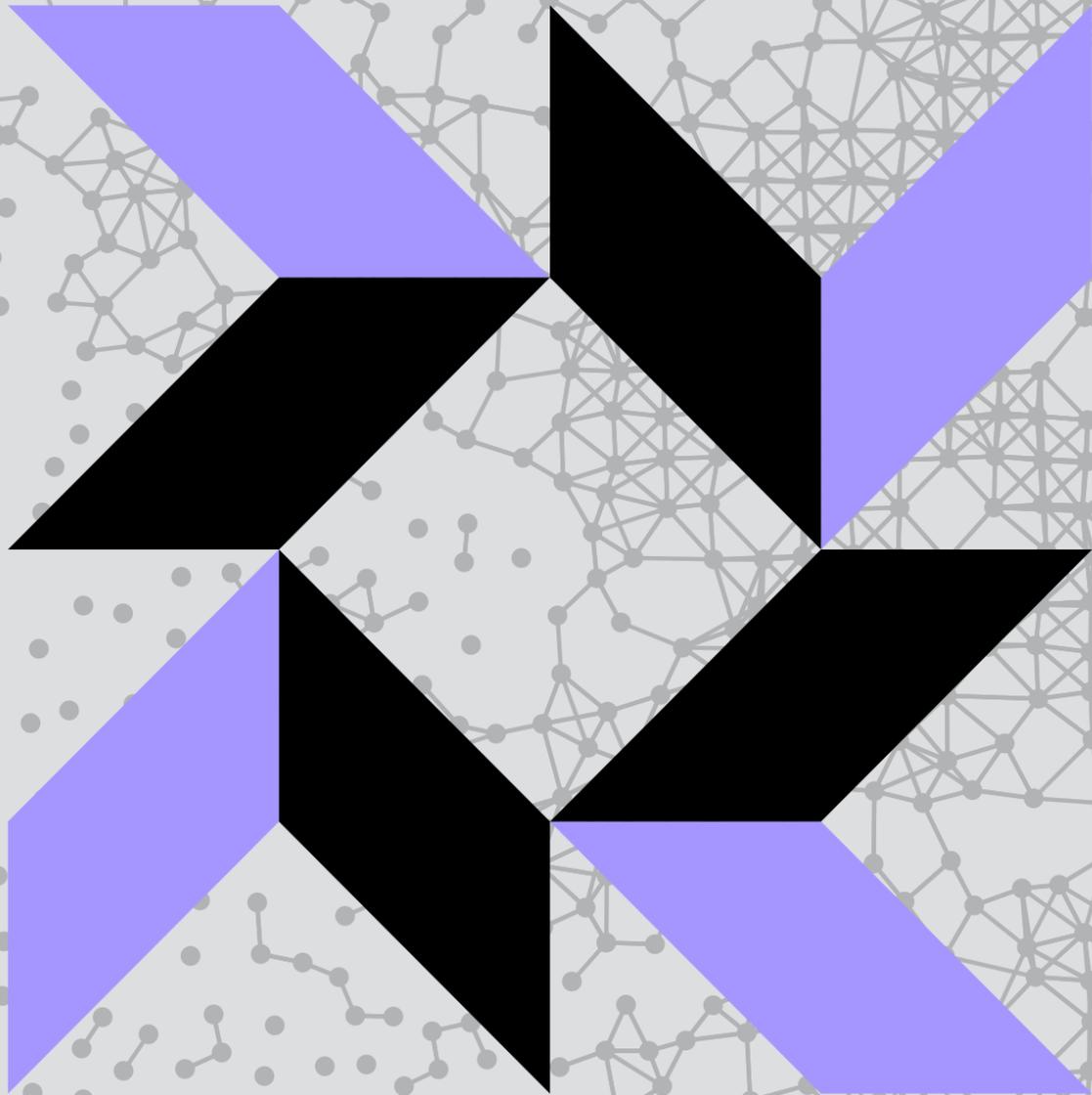
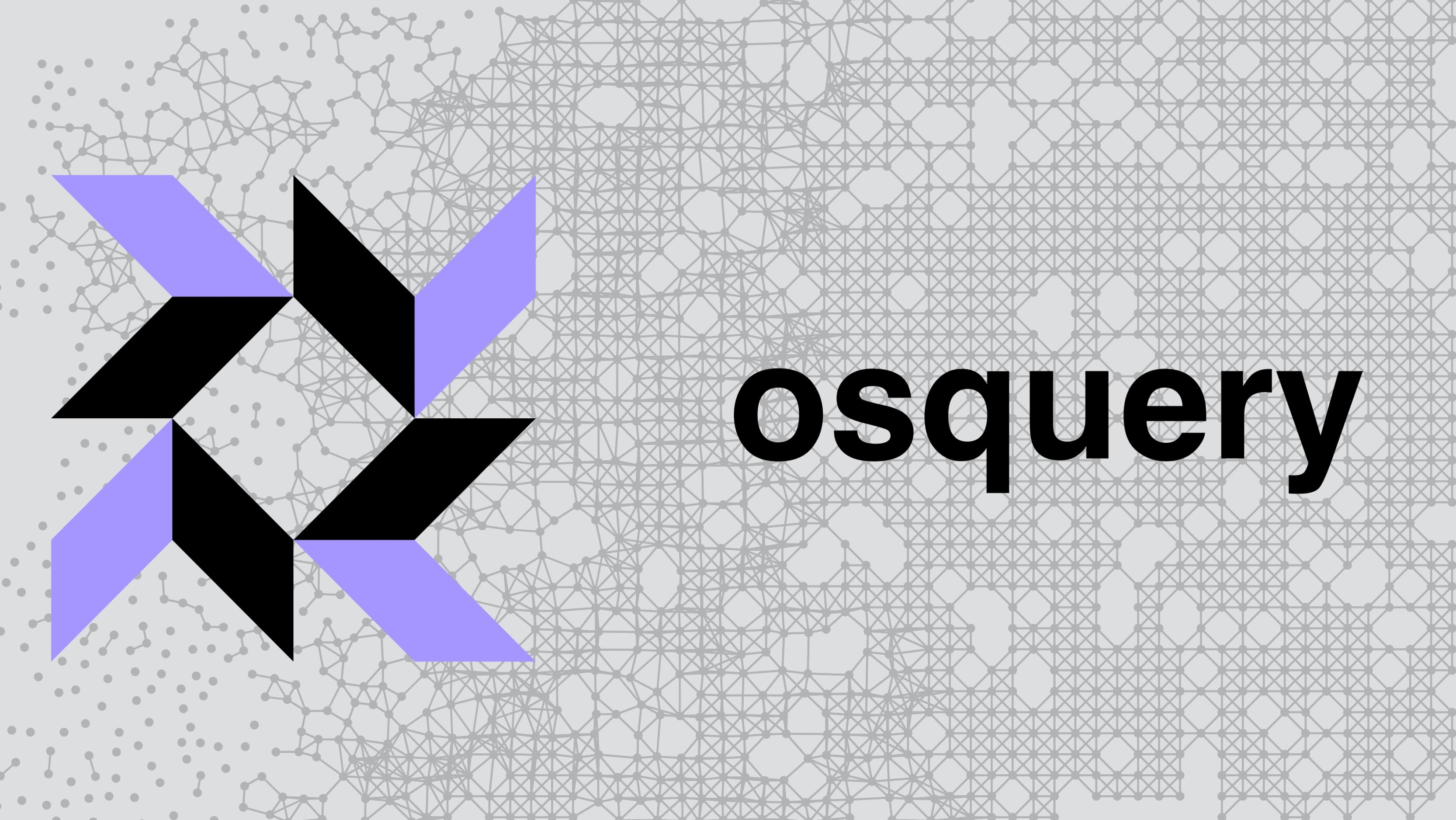
**flexible**

**performant**

vulnerability  
management

compliance

user impact



**osquery**

# osquery

SQL for your infrastructure

use SQL queries to explore OS state

- running processes
- loaded kernel modules
- active network connections
- route table
- firewall settings
- installed software
- file modifications

# why SQL?

```
SELECT pid, name, uid FROM processes
```

OS concepts are **shared** on Mac, Linux, and Windows

the “concepts” have **attributes**:  
*user ids, process ids, descriptors, ports, paths*

most developers and administrators know SQL

# why SQL?

[concept]

```
SELECT pid, name, uid FROM processes
```

# why SQL?

[attributes]

[concept]

```
SELECT pid, name, uid FROM processes
```

# why SQL?

```
SELECT pid, name, uid FROM processes
```

```
WHERE uid != 0
```

```
[constraints]
```

# why SQL?

[attribute]

```
SELECT pid, name, username FROM processes
```

```
JOIN users ON processes.uid=users.uid
```

[join]

```
WHERE uid != 0
```

# many tables are available

more tables are being written every day

- 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

# use simple tables, together

osquery enables complex analysis by allowing users to join and aggregate across several simple tables

- simple tables have many advantages
  - easier to write
  - easier to maintain
  - can be used in many contexts

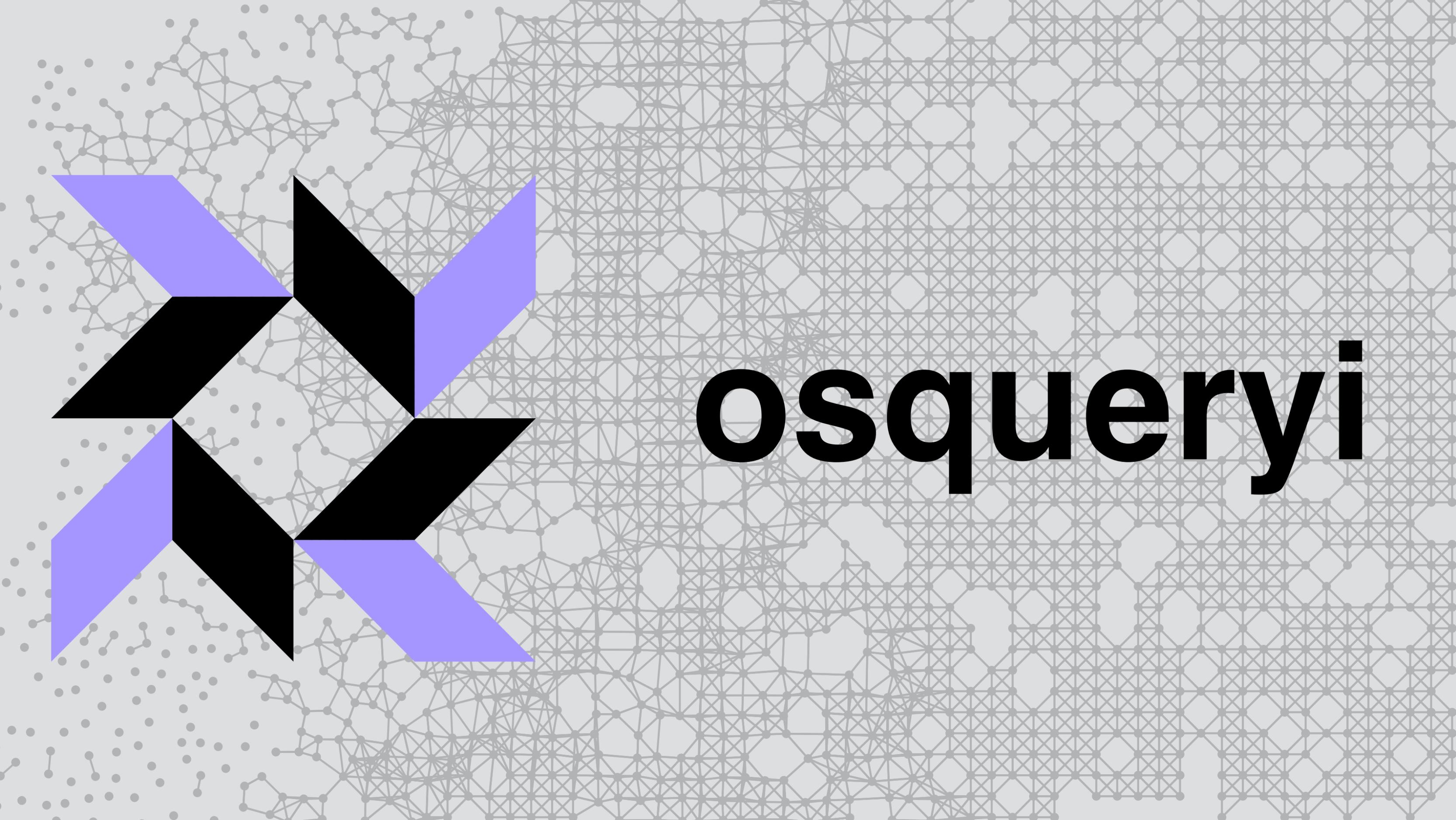


osquery is much more  
than a security tool

osquery is much more  
than a security tool

actually, literally...

it is a family of tools



**osqueryi**

```
[marpaia-mbp] ~ echo "SELECT * FROM time;" | osqueryi
```

```
+-----+-----+-----+
| hour | minutes | seconds |
+-----+-----+-----+
| 0    | 58     | 30     |
+-----+-----+-----+
```

```
[marpaia-mbp] ~ █
```

```
1. osqueryi (osqueryi)
osqueryi (osqueryi)  ⌘1
[marpaia-mbp] ~ osqueryi
~~~~~
osquery - being built, with love, at Facebook
~~~~~
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
osquery> SELECT name, program || program_arguments AS executable
...> FROM launchd
...> WHERE (run_at_load = 'true' AND keep_alive = 'true')
...> AND (program != '' OR program_arguments != '')
...> AND executable LIKE '/System/%';

+-----+-----+
| name                | executable                                                                 |
+-----+-----+
| com.apple.backupd-auto.plist | /System/Library/CoreServices/backupd.bundle/Contents/Resources/backupd-helper -launchd |
| com.apple.logind.plist      | /System/Library/CoreServices/logind                                         |
| com.apple.mtmtd.plist       | /System/Library/CoreServices/backupd.bundle/Contents/Resources/mtmd         |
| com.apple.mtmfs.plist       | /System/Library/CoreServices/backupd.bundle/Contents/Resources/mtmfs --tcp --resvport --listen localhost --oneshot --noportmap --nobrowse |
| com.apple.revisiond.plist    | /System/Library/PrivateFrameworks/GenerationalStorage.framework/Versions/A/Support/revisiond |
| com.apple.usbmuxd.plist     | /System/Library/PrivateFrameworks/MobileDevice.framework/Versions/A/Resources/usbmuxd -launchd |
| com.apple.diagnostics_agent.plist | /System/Library/CoreServices/diagnostics_agent                             |
+-----+-----+
osquery> █
```

LaunchDaemons which run a binary at boot

```
1. marpaia@marpaia-mbp: ~ (zsh)
~ (zsh)  ⌘1
[marpaia-mbp] ~ echo "SELECT path, pid FROM processes LIMIT 10;" | osqueryi
+-----+-----+
| path                                     | pid  |
+-----+-----+
| /usr/local/bin/osqueryi                 | 9850  |
| /System/Library/PrivateFrameworks/Heimdall.framework/Helpers/kcm                | 9825  |
| /System/Library/Frameworks/Python.framework/Versions/2.7/Resources/Python.app/Contents/MacOS/Python | 9774  |
| /bin/sleep                               | 9541  |
| /bin/zsh                                  | 9348  |
| /System/Library/Frameworks/OpenGL.framework/Versions/A/Libraries/CVMCompiler    | 9344  |
| /System/Library/Frameworks/OpenGL.framework/Versions/A/Libraries/CVMCompiler    | 9343  |
| /usr/libexec/opendirectoryd              | 8968  |
| /Applications/Google Chrome.app/Contents/Versions/38.0.2125.104/Google Chrome Helper.app/Contents/MacOS/Google Chrome Helper | 8615  |
| /usr/sbin/spindump                       | 8557  |
+-----+-----+
[marpaia-mbp] ~ █
```

running processes

[marpaia-mbp] ~ osqueryi

osquery - being built, with love, at Facebook

Connected to a transient in-memory database.

Use ".open FILENAME" to reopen on a persistent database.

osquery&gt; SELECT DISTINCT

```

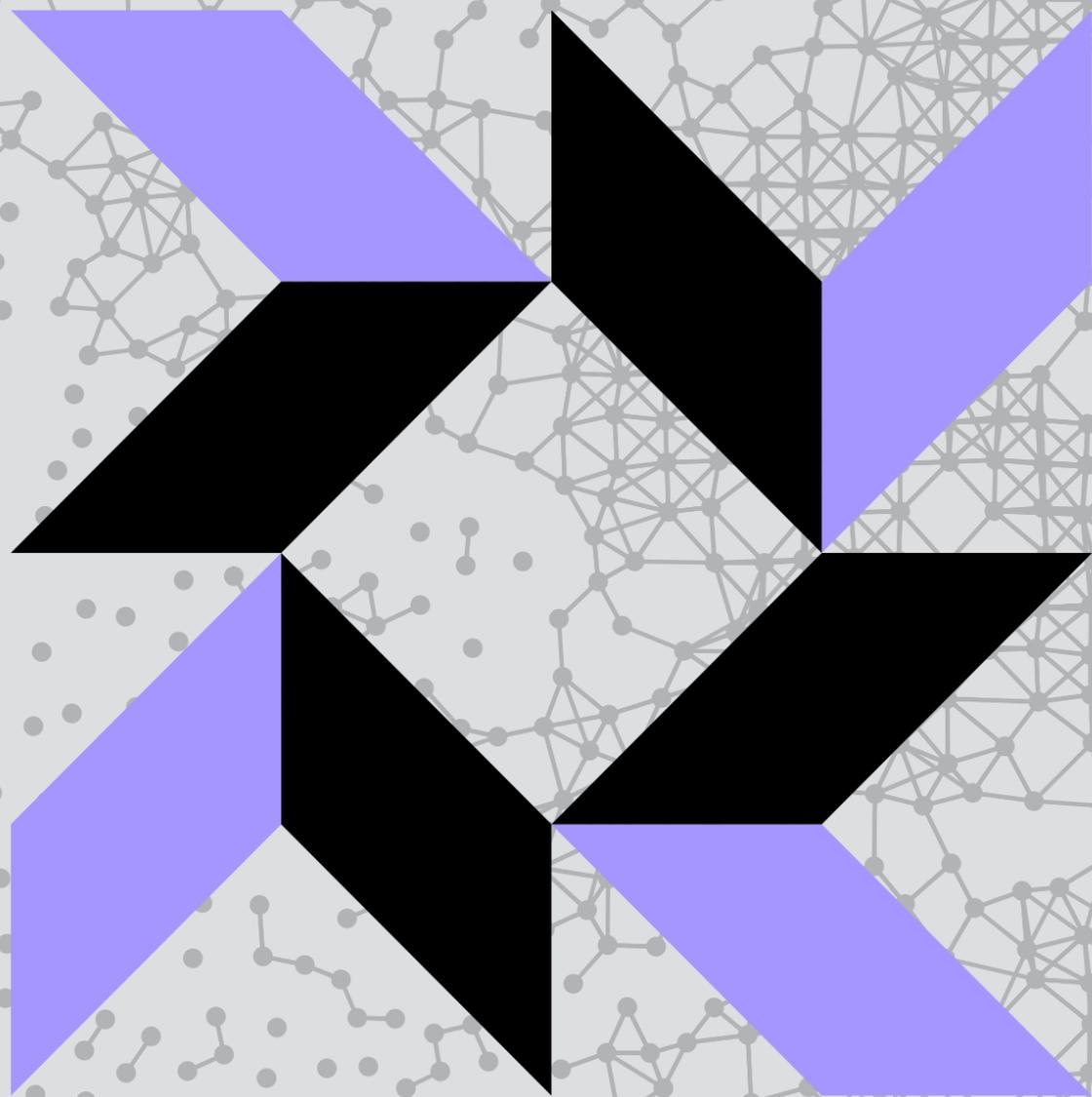
...> process.name,
...> process.pid,
...> process.parent,
...> process.on_disk,
...> process.phys_footprint,
...> listening.port,
...> listening.protocol,
...> listening.address
...> FROM processes AS process
...> JOIN listening_ports AS listening
...> ON process.pid = listening.pid;

```

name	pid	parent	on_disk	phys_footprint	port	protocol	address
TextMate	8133	1	1	54149120	52698	6	0000:0000:0000:0000:0000:0000:0000:0001:
SpotifyWebHelpe	569	1	1	5505024	4370	6	127.0.0.1
SpotifyWebHelpe	569	1	1	5505024	4380	6	127.0.0.1
2BUA8C4S2C.com.	557	1	1	25423872	6258	6	127.0.0.1
2BUA8C4S2C.com.	557	1	1	25423872	6258	6	0000:0000:0000:0000:0000:0000:0000:0001:
2BUA8C4S2C.com.	557	1	1	25423872	6263	6	127.0.0.1
2BUA8C4S2C.com.	557	1	1	25423872	6263	6	0000:0000:0000:0000:0000:0000:0000:0001:
ARDAgent	471	1	1	7933952	3283	17	0000:0000:0000:0000:0000:0000:0000:0000:
ARDAgent	471	1	1	7933952	3283	17	0.0.0.0
SystemUIServer	458	1	1	23773184	49366	17	0.0.0.0
Spotify	450	1	1	234070016	4381	6	127.0.0.1
Spotify	450	1	1	234070016	4371	6	127.0.0.1
Spotify	450	1	1	234070016	8099	6	127.0.0.1
Spotify	450	1	1	234070016	57621	17	0.0.0.0
Spotify	450	1	1	234070016	57621	6	0.0.0.0

osquery&gt; █

# processes listening on ports



**osqueryd**

# osqueryd

daemon for low-level host monitoring

know how the results of a query change over time

- schedule a query on your hosts via a config
- the daemon takes care of periodically executing your queries
  - buffers results to disk and generates a log of state changes
  - logs results for aggregation and analytics

# host eventing stream

event-based operating system introspection

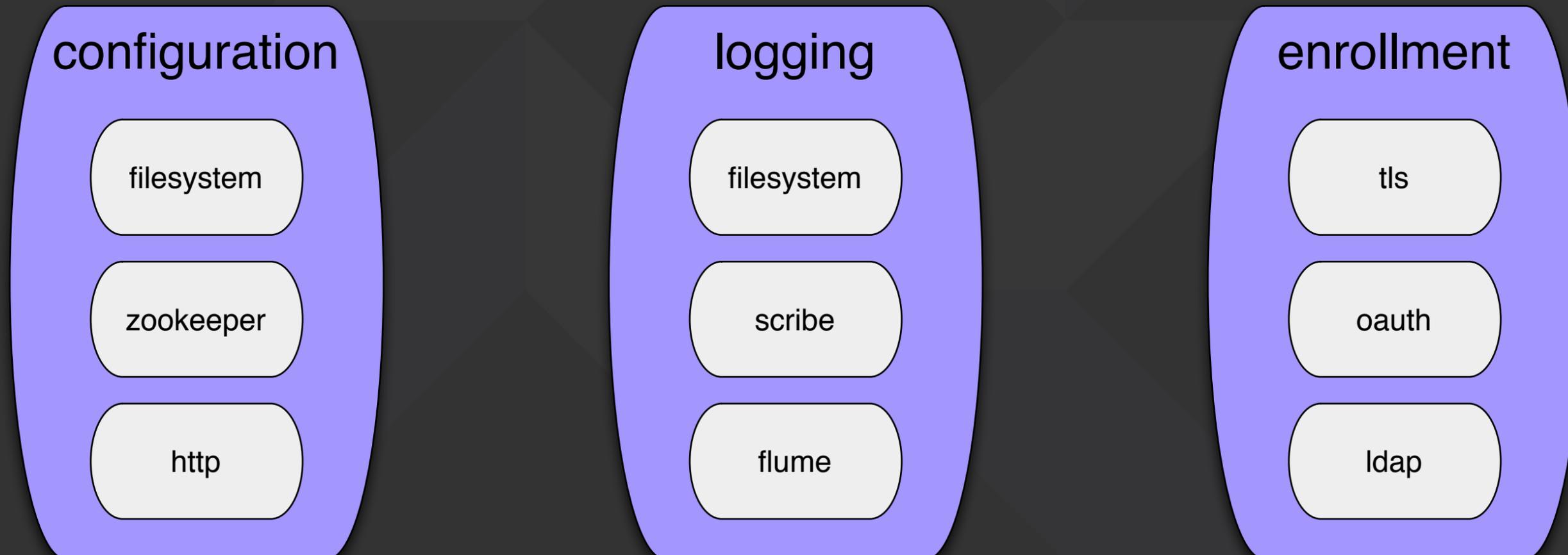
subscribe to key OS events to create dynamically growing tables

- subscribe to “publishers”
  - filesystem changes (inotify, FSEvents)
  - network setting changes (SCNetwork)
  - application usages (NSNotificationCenter)
- query the history of your host, as it evolves

# plugin system

for config distribution, data infrastructure and more

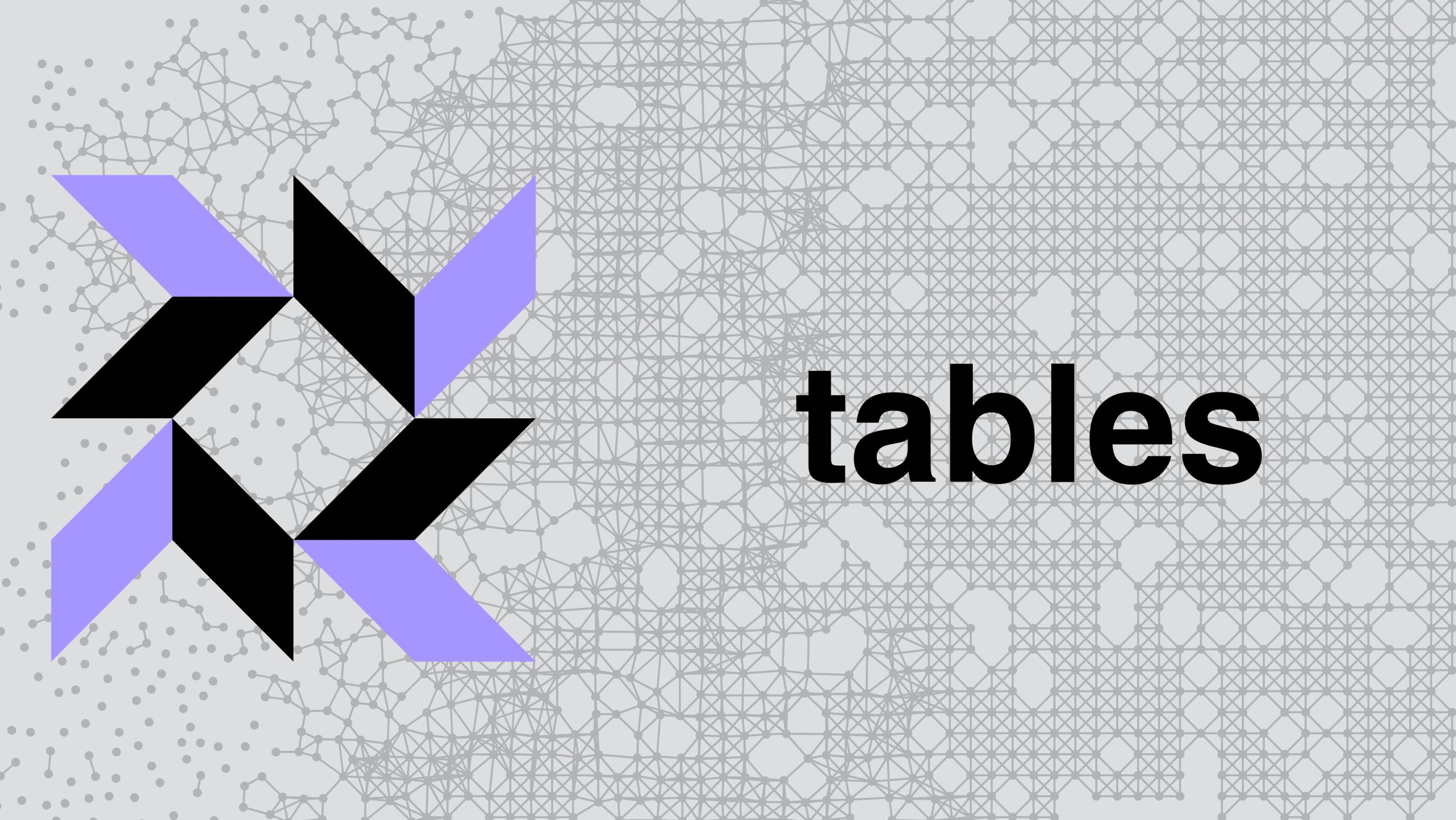
- simple plugin API
- specify your plugins at runtime with a command-line flag



# facebook workflow

how we config and log results

1. osquery.pkg published automatically to <https://osquery.io>
2. download weekly and update chef cookbook
3. chef writes configuration and installs pkg
  1. newsyslog.d rotation file
  2. list of scheduled queries
4. results written to `/var/log/osqueryd.results.log`
5. splunk lightweight forwarder
6. backend analytics



**tables**

# creating tables is easy

easily define what your tables “look like” in Python and use C++ to implement what a full-table scan would return

- the Python is used to generate faster C++ code transparently
- you write a single C++ function which implements a full-table scan

```
table_name("time")
schema([
    Column("hour", INTEGER),
    Column("minutes", INTEGER),
    Column("seconds", INTEGER),
])
implementation("time@genTime")
```

```
namespace osquery {
namespace tables {

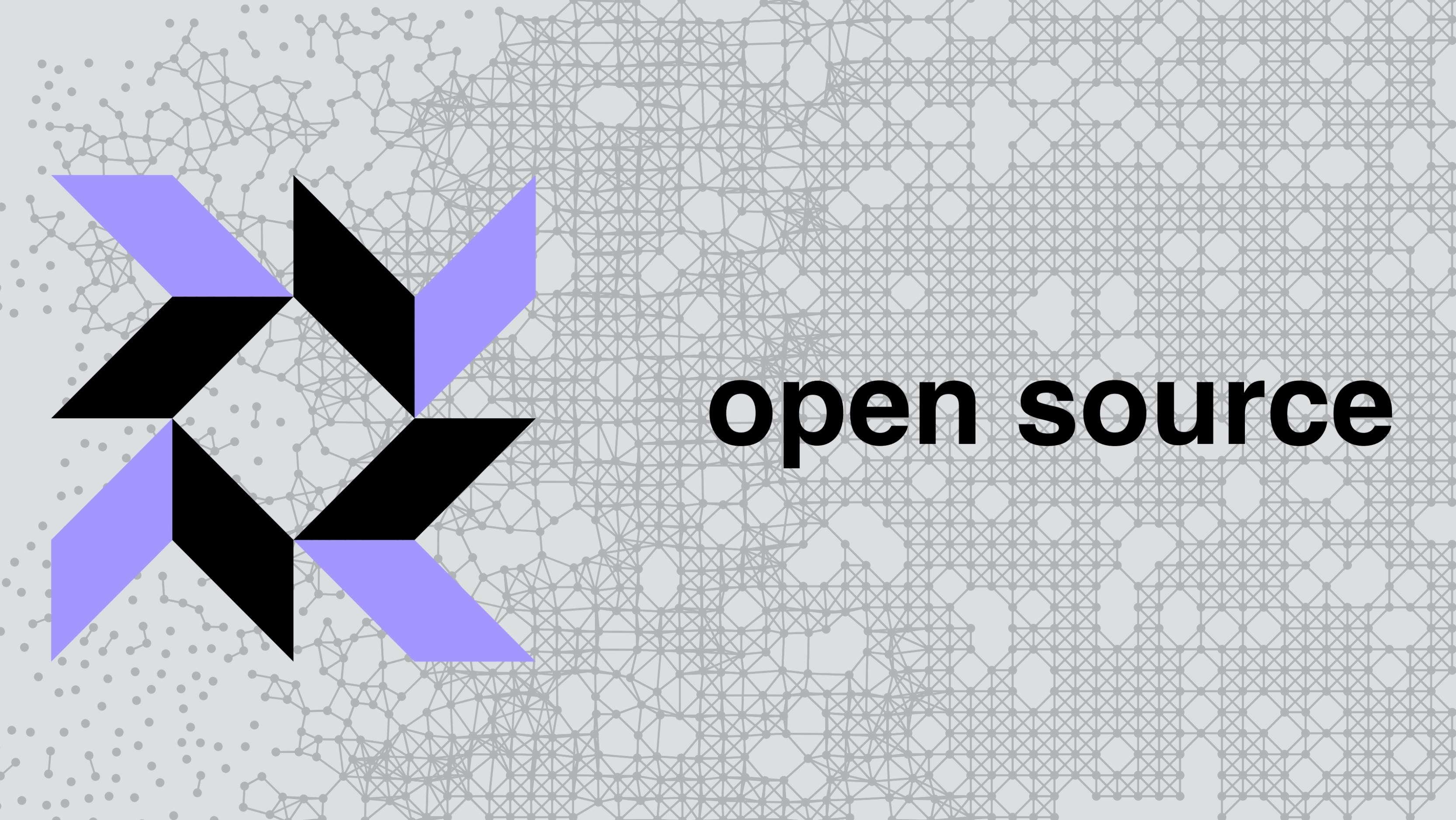
QueryData genTime(QueryContext& ctx) {
    QueryData results;
    struct tm* now = localtime(time(0));

    Row r;
    r["hour"] = INTEGER(now->tm_hour);
    r["minutes"] = INTEGER(now->tm_min);
    r["seconds"] = INTEGER(now->tm_sec);
    results.push_back(r);

    return results;
}
}
}
```

<https://osquery.io/tables>

browse all tables, columns, descriptions, and example queries



**open source**

# work on osquery with us

all development happens in the open, on GitHub

the problem that osquery solves isn't unique to facebook

- <https://github.com/facebook/osquery>
- <https://osquery.io>
- <https://osquery.readthedocs.org>

this journey is 1% finished: get involved

- we're excited to take on future challenges in the open
- let's build together

Detecting unauthorized cross-app resource access on OS X

June 22, 2015 at 1:48pm

Last week, Luyi Xing, a Ph.D. student from Indiana University Bloomington and his colleagues released a [research paper](#) on several vulnerabilities that exist in OS X and iOS. To help other organizations keep their infrastructure secure, we're sharing a brief overview of how these vulnerabilities work --- as well as new detection capabilities just added to [osquery](#), a host instrumentation product that we released as open source last October.

At this time, we're not aware of defensive security products, other than osquery, that can detect the exploitation of all of the vulnerabilities outlined in Xing's paper. We hope the information shared below will also help other developers build detection capabilities into their tools.

### Password Stealing

When an application stores an item in the OS X Keychain, the application has the option of attaching zero or more access control list (ACL) entries to that item. These ACL entries restrict the applications that are allowed to access a given keychain item, as well as restricting what the application can do with the keychain item.

Unfortunately, due to a deficiency in the underlying implementation, an arbitrary application can delete a keychain item without having the permissions to read that keychain item. The malicious application can then re-create that keychain item and add itself as well as the original application to the ACL entries for that keychain item. When this happens:

TTI:2.4s E2E:6.2s

Protect the Graph

Notes by Protect the Graph

All Notes

TAGGED

Get Notes via RSS

Embed Post

Report

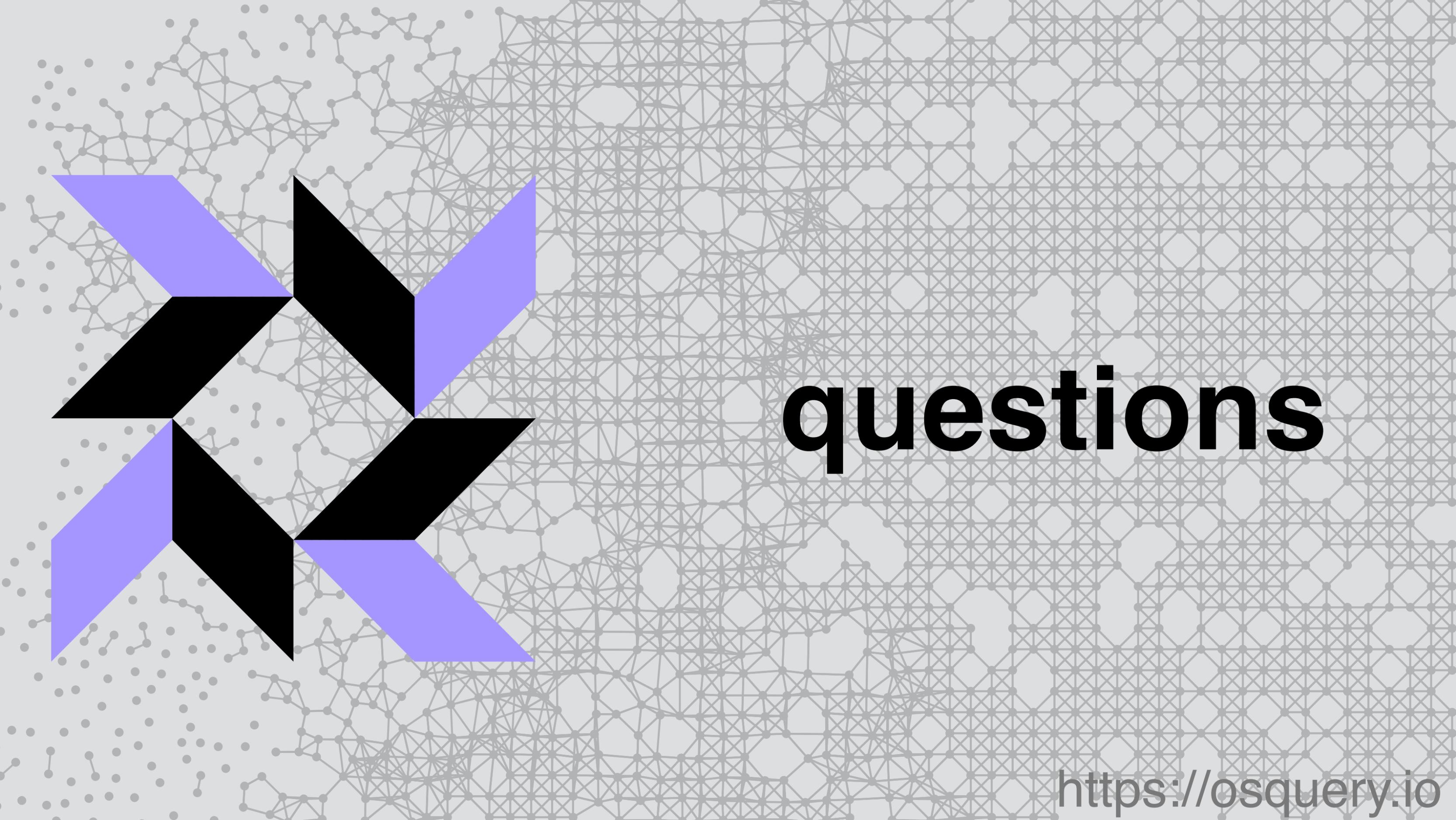
FRIEND REQUESTS See All

Drasko Budac

Confirm Friend

SPONSORED Create Ad

Chat (30)



**questions**

<https://osquery.io>