Observability for security.
Deep dive into Osquery.

Artem Mishchenko
15.12.2023
name = Artem Mishchenko
position = Infrastructure Security TechLead, inDrive
certifications = OSCP
interests = SOC, Linux, K8S and Cloud Security
Agenda

1. Observability and Security
2. Osquery Basics
3. Advanced Osquery
4. Osquery Management
5. Examples and experience
6. Conclusions
1. Observability And Security
Classic Observability

Source: Peter Bourgon, Metrics, tracing, and logging
Security Team Problems

- Basic logging is not enough
- No access to production with SSH
- Incident response process is slow
- Need to ask Linux admin to take file from production system
- No opportunity to make simple and fast vulnerability checks
- How to check remote system settings, line config lines?
What do we want?

- Get current remote machine state!
- Identify security misconfigurations!
- Do lightweight Threat Hunting!
- Collect artifacts for investigations!
- Make simple vulnerability checks!
- More security alerts for SOC team!
- Save money!
Can we get all of this with Osquery?

Let’s see at the end!
2. Osquery Basics
Osquery

- Initially Facebook project, now part of Linux Foundation
- Open Source, 20k+ stars on Github
- Cross platform (Windows, MacOS, Linux, Chrome OS)
- Exposes an operating system as a high-performance relational database
- Tries to follow the concept of read-only tool without OS changes and RCE
- `osqueryi` is a standalone console shell for local queries
- `osqueryd` is a monitoring daemon that allow you schedule queries
  - `osquery.conf` to store queries and packs
  - `osqueryflags` for daemon configuration options

https://osquery.io
https://www.uptycs.com/blog/osquery-what-it-is-how-it-works-and-how-to-use-it
Just a couple of Osquery examples

```sql
SELECT pid, name, path, cmdline FROM processes;
```

This might return:

```text
<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>path</th>
<th>cmdline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>systemd</td>
<td>/usr/lib/systemd</td>
<td>/usr/lib/systemd</td>
</tr>
<tr>
<td>2</td>
<td>kthreadd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>kworker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

[https://osquery.io/](https://osquery.io/)
Just a couple of Osquery examples

```sql
SELECT name, version, install_date FROM programs;
```

This might return:

```
+--------------------------+
| name | version | install_date |
+--------------------------+
| Google Chrome | 89.0 | 20210302 |
| VLC Media Player | 3.0.11 | 20200601 |
| ...                     |
```

https://osquery.io/
Why SQL is cool?

- SQL: Structured Query Language
- Many developers and admins are familiar with SQL
- Core concepts of SQL are platform agnostic
- Core concepts have attributes

```sql
SELECT pid, name, username FROM processes
JOIN users ON processes.uid=users.uid
WHERE uid != 0
```
### account_policy_data

Additional macOS user account data from the AccountPolicy section of OpenDirectory.

**Improve this Description on Github**

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>BIGINT</td>
<td>User ID</td>
</tr>
<tr>
<td>creation_time</td>
<td>DOUBLE</td>
<td>When the account was first created</td>
</tr>
<tr>
<td>failed_login_count</td>
<td>BIGINT</td>
<td>The number of failed login attempts using an incorrect password. Count resets after a correct password is entered.</td>
</tr>
<tr>
<td>failed_login_timestamp</td>
<td>DOUBLE</td>
<td>The time of the last failed login attempt. Resets after a correct password is entered.</td>
</tr>
<tr>
<td>password_last_set_time</td>
<td>DOUBLE</td>
<td>The time the password was last changed</td>
</tr>
</tbody>
</table>
Complex query for Osquery

```sql
SELECT p.*, pos.*
FROM process_open.Sockets AS pos
INNER JOIN processes AS p ON p.pid = pos.pid
WHERE remote_address <> "" AND remote_port != 0 AND pos.pid > 0
LIMIT 5;
```

<table>
<thead>
<tr>
<th>pid</th>
<th>laddr</th>
<th>lport</th>
<th>raddr</th>
<th>rport</th>
<th>family</th>
<th>proto</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1135</td>
<td>192..</td>
<td>56493</td>
<td>140..</td>
<td>443</td>
<td>2</td>
<td>6</td>
<td>..firefox</td>
</tr>
<tr>
<td>1135</td>
<td>192..</td>
<td>55620</td>
<td>35..</td>
<td>443</td>
<td>2</td>
<td>6</td>
<td>..firefox</td>
</tr>
<tr>
<td>1135</td>
<td>192..</td>
<td>56536</td>
<td>104..</td>
<td>443</td>
<td>2</td>
<td>6</td>
<td>..firefox</td>
</tr>
<tr>
<td>1135</td>
<td>192..</td>
<td>55527</td>
<td>34..</td>
<td>443</td>
<td>2</td>
<td>6</td>
<td>..firefox</td>
</tr>
<tr>
<td>1135</td>
<td>192..</td>
<td>56531</td>
<td>216..</td>
<td>443</td>
<td>2</td>
<td>6</td>
<td>..firefox</td>
</tr>
</tbody>
</table>

Complex query for Osquery

SELECT p.*, lp.*
FROM listening_ports AS lp
INNER JOIN processes AS p ON p.pid = lp.pid
WHERE address <> "" AND port != 0 AND lp.pid > 0
LIMIT 5;

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>port</th>
<th>family</th>
<th>proto</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemUIServer</td>
<td>0.0.0.0</td>
<td>57645</td>
<td>2</td>
<td>17</td>
<td>/System..</td>
</tr>
<tr>
<td>postgres</td>
<td>::1</td>
<td>5432</td>
<td>10</td>
<td>6</td>
<td>/Applic..</td>
</tr>
<tr>
<td>postgres</td>
<td>127.0..</td>
<td>5432</td>
<td>2</td>
<td>6</td>
<td>/Applic..</td>
</tr>
<tr>
<td>trezord</td>
<td>127.0..</td>
<td>21325</td>
<td>2</td>
<td>6</td>
<td>/Applic..</td>
</tr>
<tr>
<td>Chrome Helper</td>
<td>0.0.0.0</td>
<td>5353</td>
<td>2</td>
<td>17</td>
<td>/Applic..</td>
</tr>
</tbody>
</table>

Osquery: Under the Hood

https://www.kolide.com/blog/osquery-under-the-hood
Osquery: Under the Hood

https://www.kolide.com/blog/osquery-under-the-hood
Osquery Watchdog

- When we start osqueryd, we get two processes:
  - Parent process - The “watchdog”
  - Child process - The “worker”

Osquery Watchdog

- When we start osqueryd, we get two processes:
  - Parent process - The “watchdog”
  - Child process - The “worker”
- Potentially resource-intensive operations are performed in the worker process.
  - Run queries, output logs, etc.

Osquery Watchdog

- When we start osqueryd, we get two processes:
  - Parent process - The "watchdog"
  - Child process - The "worker"
- Potentially resource-intensive operations are performed in the worker process.
  - Run queries, output logs, etc.
- The watchdog process checks the utilization stats for the worker on an interval.
  - Resource utilization limits exceeded -> Watchdog kills/respawns worker
  - Multiple watchdog kills put the query to blacklist (denylist) for 24 hours

Osquery Watchdog

SELECT * FROM osquery_schedule WHERE blacklisted = 1

Configuration options:
--watchdog_level
--watchdog_utilization_limit
--watchdog_memory_limit
Osquery Watchdog

What should we do if we are not sure about the reliability of Watchdog?

Let's turn on Cgroups! (for Linux)

```python
osquery_memory_limit: "{{ (ansible_memtotal_mb | int >= 16384) | ternary('1G', '512M') }}"
osquery_cpu_quota: "{{ (ansible_processor_vcpus | int >= 20) | ternary('100%', '50%') }}"
```
Osquery Possible Use Cases

- **Server Infrastructure**
  - Works good in Linux and Windows servers
  - Observability tool for security regression tests
  - Check something for compliance requirements
  - Lightweight threat hunting and HIDS
Osquery Possible Use Cases

- **Corporate Laptops/PCs**
  - Connections can be unstable
  - Security posturing and device control
  - Lightweight threat hunting & HIDS tool
  - You have to think about employees privacy
  - You have to be ready to properly explain osquery purposes to your colleagues
Osquery from the perspective of some employees - probe
3. Advanced Osquery
Osquery Event Tables

- Helps to collect data **continuously**
- Uses OS features to generate events, like **Linux Audit Framework**
- The data is cached in internal **RocksDB** in Osquery
- The feature creates **additional system load** (not always predictable)
- Osquery Watchdog **often doesn’t work as expected**

https://fleetdm.com/guides/osquery-evented-tables-overview

---

```
--disable_events=false
--enable_file_events=true
--disable_audit=false
```

**IS IT AUDITD?**
Osquery Augeas

SELECT * FROM augeas WHERE path = '<path>'

- Use concept of lenses
- Can parse different configuration formats
- Can help us to organise security checks for configs

https://medium.com/swlh/parsing-configuration-files-with-augeas-on-osquery-ec8260a9a50b
Real life scenario for Osquery: SSH Agent Hijacking

Server
/etc/ssh/sshd_config
AllowAgentForwarding yes

Client
~/.ssh/ssh_config
ForwardAgent yes

https://hx015.medium.com/ssh-session-hijack-analytic-a2c684ba410f
Real life scenario for Osquery: SSH Agent Hijacking

Server
/etc/ssh/sshd_config
AllowAgentForwarding yes

Client
~/.ssh/ssh_config
ForwardAgent yes

SSH_AUTH_SOCK=/tmp/ssh-haqzR16816/agent.16816

https://hx015.medium.com/ssh-session-hijack-analytic-a2c684ba410f
Real life scenario for Osquery: SSH Agent Hijacking

Server
/etc/ssh/sshd_config
AllowAgentForwarding yes

Client
~/.ssh/ssh_config
ForwardAgent yes

SSH_AUTH_SOCK=/tmp/ssh-haqzR16816/agent.16816

SSH_AUTH_SOCK=/tmp/ssh-haqzR16816/agent.16816 ssh bob@10.10.0.1

https://hx015.medium.com/ssh-session-hijack-analytic-a2c684ba410f
Real life scenario for Osquery: SSH Agent Hijacking

What can we do here with Osquery?

1) Find hosts with allowed AllowAgentForwarding and disable it where possible

```
SELECT label, value FROM augeas WHERE path = '/etc/ssh/sshd_config' and label = "AllowAgentForwarding" and value = "yes"
```

2) Find users who currently use ssh agent and wean them off forwarding agents everytime, and also help fix their ssh configs

```
SELECT * FROM file JOIN users using (uid) WHERE file.path LIKE '/tmp/ssh%'
```

3) Made better network isolation to reduce blast radius

4) Decided to use alternative tunneling methods like limited AllowTcpForwarding
File Carving

SELECT * FROM `carves` WHERE `path` LIKE '/etc/osquery/%' and `carve`=1

- Take files from remote device with this feature
- Better to use it with osquery manager
- Carefully use this feature with employees’ devices or disable

https://fleetdm.com/docs/using-fleet/fleetctl-cli#file-carving
Yara Rules and Osquery

```
SELECT * FROM yara WHERE path like '/root/%%' AND sigrule IN ('rule eicar {
    strings:
$s1="X5O!P%@AP[4\PZX54(P^)7CC)7}EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*"
  fullword ascii
  condition:
    all of them
}
) AND matches='eicar'
```

- Swiss knife to identify malware
- Whole system scan can become expensive quickly
- Can work with osquery process and FIM events
- Target smaller set of files (like current processes or specific directory)

https://www.eicar.org/download-anti-malware-testfile/
https://github.com/InQuest/awesome-yara
Osquery Extensions

- Can be written on C++, Python or Go
- For example, with help of osquery-go
  - [https://github.com/osquery/osquery-go](https://github.com/osquery/osquery-go)
- Acts as a separate binary
- Some interesting extensions examples
  - [https://github.com/trailofbits/osquery-extensions](https://github.com/trailofbits/osquery-extensions)

Osquery Concerns and Lacks

- Lack of tables for container engines beyond Docker
- Lack of tables for Cloud and Kubernetes
  - Extensions from Uptycs company are deprecated and no more supported
- Works better in host operating systems
Osquery Concerns and Lacks

- We can try deploy Osquery in K8S, but implementation can be tricky (especially in dynamic/managed environment)
  - You can look a couple of examples in research of Alexander Ivanov from Wrike (https://www.youtube.com/watch?v=FvEMwV6bBI)
- You can’t use osquery to find arbitrary file on filesystem (https://www.kolide.com/blog/the-file-table-osquery-s-secret-weapon)
4. Osquery Management
How to manage Osquery for a whole infrastructure?

- Distribute specific queries through config
  - Can use Ansible, Puppet, Chef, SaltStack, etc
  - If you need to make changes - you need to apply them explicitly
  - Gather logs from local files on endpoints with your favorite log shipper

https://github.com/osquery/osquery?tab=readme-ov-file#osquery-fleet-managers
How to manage Osquery for a whole infrastructure?

- **Use manager tools for Osquery**
  - Fleetdm (formerly known as Kolide Fleet)
  - Kolide
  - Osctrl
  - Zentral
  - Zercurity
  - Elastic Stack Osquery Manager
  - etc

https://github.com/osquery/osquery?tab=readme-ov-file#osquery-fleet-managers
Fleetdm (formerly known as Kolide Fleet)

- Core is **open and free**
- **Live queries** across all Osquery fleet
- Can **schedule queries** and log results
- Can save queries as **policies** and notify about violations
- Supports **labels and packs** (yet)
- Other features like MDM in paid version

https://fleetdm.com
Fleetdm Architecture

https://fleetdm.com/docs/deploy/introduction#infrastructure-dependencies
Fleetdm Security Hardening

● SAML SSO
  ○ Group membership based access + implicit 2FA
  ○ Google Workspace, Okta, etc

● Minimal RBAC
  ○ Only three roles in free version (Admin, Maintainer, Observer)
  ○ Add user to “Observer” role and allow only specific queries
  ○ Unfortunately we can’t assign users to restricted scopes in core

https://fleetdm.com/docs/deploy/single-sign-on-sso
Fleetdm Security Hardening

- Separate settings for Fleetdm and Osquery handlers in LB
  - Stronger requirements for admin panels (2FA, etc)
  - Different ACL for osquery and users
- Gather audit events from MySQL DB to SIEM
  - MySQL “activities” table

https://fleetdm.com/docs/deploy/single-sign-on-sso
Separate settings for Fleetdm on Nginx

```nginx
location ~ /api/(v1/)?osquery {
    proxy_pass https://fleet;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_buffering off;
}
```

```nginx
location / {
    proxy_pass https://fleet;
    proxy_read_timeout 90;
    proxy_connect_timeout 90;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Proxy "";
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
}
```

https://mysteryincorporated.medium.com/nginx-configuration-for-fleetdm-setups-that-want-to-catch-3m-flatties-willem-powerfish-be-proud-7f99f97f9ede
Fleetdm Labels and Osquery Packs

- Packs - just a group of queries
- Labels - feature to create subgroups of hosts
- We can’t schedule query to labeled group of hosts, only pack
- There is a default global pack in Fleetdm for all hosts

Popular public packs:

https://github.com/osquery/osquery/tree/experimental/packs
https://github.com/palantir/osquery-configuration
https://github.com/teoseller/osquery-attck

Osquery MITRE ATT&CK Pack

```json
{
  "platform": "windows",
  "description": "ATT&CK: T1107,T1158,T1191,T1118,T1216,T1059,T1170,T1086,T1117,T1053,T1035,T1197,T1128,T1134,T1126,T1087,T1201,T1069,T1057
  "queries": {
    "attrib.exe": {
      "query": "select * from file WHERE directory = 'C:\\Windows\\Prefetch\\' and filename like '%attrib\';",
      "interval": 600,
      "description": "Attrib Execute, usaully used to modify file attributes - ATT&CK T1158",
      "platform": "windows"
    },
    "schtasks.exe": {
      "query": "select * from file WHERE directory = 'C:\\Windows\\Prefetch\\' and filename like '%schtasks\';",
      "interval": 600,
      "description": "Schtasks Execute, usaully used to create a scheduled task - ATT&CK T1053,S0111",
      "platform": "windows"
    },
    "taskeng.exe": {
      "query": "select * from file WHERE directory = 'C:\\Windows\\Prefetch\\' and filename like '%taskeng\';",
      "interval": 600,
      "description": "Taskeng Execute, usaully used to create a scheduled task - ATT&CK T1053",
      "platform": "windows"
    }
  }
}
```

Osquery Data Pulling Model

Available intervals options:

- **distributed_interval**: 60
  - Can be changed directly in Osquery flags and in Fleetdm

- **logger_tls_period**: 10
  - Can be changed directly in Osquery flags and in Fleetdm

- **config_refresh**: 60
  - Only in Osquery flags

5. Examples and experience
Osquery for Security: sshd config

**config-auth-ssh | allowed PermitRootLogin**

We should have only PermitRootLogin=no in production

Query:

```
1 SELECT label, value FROM augeas WHERE path = '/etc/ssh/sshd_config' and label = " PermitRootLogin" and value != 'no';
```

Compatible with:  
- macOS  
- Windows  
- Linux

Observers can run

Users with the Observer role will be able to run this query on hosts where they have access.

**config-auth-ssh | allowed PasswordAuthentication**

We should have only PasswordAuthentication=no in production

Query:

```
1 SELECT label, value, path FROM augeas WHERE path = '/etc/ssh/sshd_config' and label = "PasswordAuthentication" and value = "yes";
```

Compatible with:  
- macOS  
- Windows  
- Linux

Observers can run

Users with the Observer role will be able to run this query on hosts where they have access.
Osquery for Security: SSH Private Keys

```sql
SELECT path, mode, username, datetime(mtime, 'unixepoch', 'localtime') AS mtime, datetime(atime, 'unixepoch', 'localtime') AS atime, inode, file.uid, file.gid
FROM users
JOIN user_ssh_keys USING (uid), file USING (path);
```

Compatible with: ✓ macOS ✓ Windows ✓ Linux ✓ ChromeOS

☐ Observers can run

Users with the observer role will be able to run this query on hosts where they have access.
Osquery for Security: Privileged Docker Containers

Find Containers Running As Privileged

https://community.carbonblack.com/t5/Query-Exchange/Find-Containers-Running-As-Privileged/idi-p/75266

Query

```sql
SELECT id, name, image, state, started_at
FROM docker_containers
WHERE privileged=1;
```

Compatible with: ✓ macOS  × Windows  ✓ Linux  × ChromeOS

Observers can run

Users with the observer role will be able to run this query on hosts where they have access.
Osquery for Security: Software Packages

vulners | rpm packages

Get software packages from RHEL based servers for Vulners API requests

Query:

```sql
SELECT (SELECT REPLACE(value, '"', '') FROM augeas where path='/etc/os-release' and label='ID') as osname, (SELECT REPLACE(value, '"', '') FROM augeas where path='/etc/os-release' and label='VERSION_ID') as osversion, name || '-' || version || '-' || release || '-' || arch as package FROM rpm_packages
```

Compatible with: X macOS  X Windows  ✔ Linux

Users with the Observer role will be able to run this query on hosts where they have access.

vulners | deb packages

Get software packages from Debian based servers for Vulners API requests

Query:

```sql
SELECT (SELECT REPLACE(value, '"', '') FROM augeas where path='/etc/os-release' and label='ID') as osname, (SELECT REPLACE(value, '"', '') FROM augeas where path='/etc/os-release' and label='VERSION_ID') as osversion, name || '-' || version || '-' || arch as package FROM deb_packages
```

Compatible with: X macOS  X Windows  ✔ Linux

Users with the Observer role will be able to run this query on hosts where they have access.
# Osquery for Security: Software Packages

<table>
<thead>
<tr>
<th>Host</th>
<th>osname</th>
<th>osversion</th>
<th>package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>zstd 1.4.8+dfsg-3build1 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>zlib1g 1:1.2.11.dfsg-2ubuntu9.2 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>zabbix-agent2 1:6.2.9-1ubuntu22.04 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>xz-utils 5.2.5-2ubuntu1 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>xxd 2:8.2.3995-1ubuntu2.12 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>xkb-data 2.33-1 all</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>xf斯progs 5:13.0-1ubuntu2 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>xdg-user-dirs 0.17-2ubuntu4 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>wireless-regdb 2022.06.06-0ubuntu1~22.04.1 all</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>whiptail 0.52.21-5ubuntu2 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>wget 1.21.2-2ubuntu1 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>vim-tiny 2:8.2.3995-1ubuntu2.12 amd64</td>
</tr>
<tr>
<td></td>
<td>ubuntu</td>
<td>22.04</td>
<td>vim-runtime 2:8.2.3995-1ubuntu2.12 all</td>
</tr>
</tbody>
</table>

[https://vulners.com/](https://vulners.com/)
<table>
<thead>
<tr>
<th>Hosts</th>
<th>Controls</th>
<th>Software</th>
<th>Queries</th>
<th>Policies</th>
</tr>
</thead>
</table>

- There is no Packs button in Fleetdm anymore
- This page is available only by direct link `/packs/manage`
- Fleetdm developers focus on paid version and “Teams” feature
- “Teams” feature can provide great experience for scope restriction
- But Fleetdm wants to use Teams as replacement for Packs and Labels
- Fleet will support Packs until the next major version release

https://osquery.slack.com/archives/C01DXJL16D8/p1683537869964549
Concerns for the future of Fleetdm

How to live further?
We still didn’t decide, but we can

- Try another osquery manager
- Fork specific Fleetdm version, support and develop it ourselves
- Just live with old version until it breaks
- Live without Packs and Label and query all hosts anytime
- Maybe something else

https://osquery.slack.com/archives/C01DXJL16D8/p1683537869964549
6. Conclusions
What did we get as result?

- Get current remote machine state! => Different Osquery tables, can write new with extensions
- Identify security misconfigurations! => Augeas and other tables
- Do lightweight Threat Hunting! => Different Osquery tables
- Collect artifacts for investigations! => Carving table
- Make simple vulnerability checks! => Yara rules and packages gathering with Osquery (for checks with a third-party API)
- More security alerts for SOC team! => Different Osquery tables
- Save money! => Osquery is free, Fleetdm also has free core version
Thank you for the attention!
Q&A

My LinkedIn

OWASP Limassol