This is an introductory workshop
You probably won’t hear/see a lot of new things if you have:
- Already used osquery;
- Followed SANS SEC599, etc.;

If you are stuck, please do not suffer in silence!
Workshop VM

- ais_workshop_xubuntu-18.04.2-desktop-amd64
- VMware Workstation, Player, or Fusion
  - You can try VirtualBox too, but you are on your own with that... sorry! 😊
- 8 GB RAM
- 30-50 GB disk space
- **Keyboard layout: EN-US !!!**
- Workshop VM (Ubuntu) user/pass: **user / Workshop1234%**
  - Normally, it should not require password for login and sudo
About David

• Managing partner at Alzette Information Security (@AlzetteInfoSec)
• Network penetration testing, security architectures, security monitoring, incident response
• Instructor at SANS Institute: FOR572
• BSides Luxembourg organizer https://bsideslux.lu
• Twitter: @DavidSzili
• E-mail: david.szili@alzetteinfosec.com
• Blog: http://jumpespjump.blogspot.com
Introduction to Osquery

2019 Pass the SALT Workshop
About Osquery

What is osquery?
• Build for:
  • Security
  • Compliance
  • Operations (DevOps)
• Everything in SQL!
  • Exposes the operating system as a relational database
• Developed by Facebook

Why osquery?
• (Free) Open Source Software
• Cross-platform
  • One platform for monitoring
  • Native packages for supported operating systems
• Large-scale host monitoring or threat hunting
• Growing Community
Osquery History

2014 OCT 29: Announcement

2016 SEP 27 / 2016 Oct 4: Osquery for Windows (Trail of Bits)

2018 APR 25: v3.2.4 - First stable release in 3.0.0 series

2019 JUN 28: osquery 4.0.0 released
Osquery flavours

Carbon Black LiveOps™

osql

- Osquery open source "soft-fork" from Trail of Bits
  - https://osql.io
Installation

- Built and signed by the osquery team
- Uses minimal number of run-time library dependencies
  - Binaries are a bit big (~20MB)
- Packages for:
  - macOS
  - Linux (Tarball, RPM, DEB)
  - Windows (MSI)
- https://osquery.io/downloads
- Alternative downloads: darwin, apt, yum, freebsd, chocolatey repositories
Getting Help

- **Osquery Documentation**

- Osquery Slack
  - [https://osquery-slack.herokuapp.com/](https://osquery-slack.herokuapp.com/)

- Osquery E-mail (for long-form questions)
  - osquery@fb.com

- Osquery Github
  - [https://github.com/facebook/osquery/issues](https://github.com/facebook/osquery/issues)
## Main Components

<table>
<thead>
<tr>
<th>osqueryi</th>
<th>osqueryd</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Interactive query console</td>
<td>• Host monitoring daemon</td>
</tr>
<tr>
<td>• Provides an SQL interface</td>
<td>• Distributed, high-performance, low-footprint</td>
</tr>
<tr>
<td>• Completely standalone, no communication with a daemon</td>
<td>• Schedules queries to be executed across an entire infrastructure</td>
</tr>
<tr>
<td>• Does not require elevated privileges (root/Administrator), but not every table can be queried in this case</td>
<td>• Aggregates query results and generates logs</td>
</tr>
</tbody>
</table>
Osquery SQL and schema

• Superset of SQLite’s SQL
  • SELECT only! (without using extensions)
  • You can still create run-time tables/VIEWS

• "SQL As Understood By SQLite“:
  • [https://www.sqlite.org/lang.html](https://www.sqlite.org/lang.html)

• Osquery schema documentation:
  • [https://osquery.io/schema](https://osquery.io/schema)

• More than 200 tables in total!
  • All platforms: ~40
  • MacOS: ~160
  • FreeBSD: ~40
  • Linux: ~130
  • Windows: ~73
Using osqueryi

- **Used for:**
  1. Developing queries
  2. Exploring a single system

- **Side note:**
  - There is no connection between interactive and daemon mode
  - However, osqueryi and osqueryd are the same binary!
  - You can run osqueryi in daemon mode and osqueryd interactively 😊

- **Linux/BSD/MacOS:**
  - `$ sudo osqueryi`

- **Windows:**
  - Osqueryi is not in the path by default
  - C:\ProgramData\osquery\osqueryi.exe {in an Administrator console}
Osquery Shell and Schema Hands-On

2019 Pass the SALT Workshop
SELECT (1)

- **SELECT** statement
  - **FROM**: defines input data
  - **WHERE**: boolean expression evaluated for each row
  - **GROUP BY**: Groups the result-set by one or more columns
  - **HAVING**: boolean expression evaluated once for each group (can use aggregate functions)
  - **DISTINCT/ALL**: no duplicate rows/all rows displayed
- [https://www.sqlite.org/lang_select.html](https://www.sqlite.org/lang_select.html)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>BETWEEN</td>
<td>Between a certain range</td>
</tr>
<tr>
<td>LIKE</td>
<td>Search for a pattern</td>
</tr>
<tr>
<td>IN</td>
<td>Specify multiple values</td>
</tr>
</tbody>
</table>
More on SELECT statement:

- **ORDER BY**: the list of expressions in the ORDER BY determine the order in which rows are returned
  - **ASC**: smaller values returned first
  - **DESC**: larger values returned first
- **LIMIT**: upper bound on the number of rows returned
  - **OFFSET**: the first X number of rows are omitted from the results

Compound SELECT Statements

- **UNION ALL**: returns all the rows from two SELECTs
- **UNION**: like UNION ALL, but duplicate rows are removed
- **INTERSECT**: returns the intersection of the results of two SELECTs
- **EXCEPT**: Returns the subset of rows returned by the left SELECT that are not returned by the right-hand SELECT
# Aggregate Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>avg(X)</code></td>
<td>Returns the average value of all non-NULL X within a group</td>
</tr>
<tr>
<td><code>count(X)</code></td>
<td>Returns a count of the number of times that X is not NULL in a group</td>
</tr>
<tr>
<td><code>count(*)</code></td>
<td>Returns the total number of rows in the group</td>
</tr>
<tr>
<td><code>group_concat(X)</code></td>
<td>Returns a string which is the concatenation of all non-NULL values of X</td>
</tr>
<tr>
<td><code>group_concat(X,Y)</code></td>
<td><code>group_concat(X)</code> and Y is used as the separator between instances of X</td>
</tr>
<tr>
<td><code>max(X)</code></td>
<td>Returns the maximum value of all values in the group</td>
</tr>
<tr>
<td><code>min(X)</code></td>
<td>Returns the minimum non-NULL value of all values in the group</td>
</tr>
<tr>
<td><code>sum(X)</code></td>
<td>Returns the (integer) sum of all non-NULL values in the group</td>
</tr>
<tr>
<td><code>total(X)</code></td>
<td>Returns the (float) sum of all non-NULL values in the group</td>
</tr>
</tbody>
</table>
JOIN

- **INNER JOIN** (or just **JOIN**): combines column values of two tables based upon the join predicate (**ON** keyword)
  - **USING**: specifies a list of one or more columns as a condition
  - **NATURAL INNER JOIN**: automatically tests for equality between the values of every column that exists in both tables

- **LEFT OUTER JOIN** (or just **LEFT JOIN**): returns all values from the left table, even if there is no match with the right table
  - **ON, USING, NATURAL**: works the same way as in INNER JOINs

- **CROSS JOIN**: matches every row of the first table with every row of the second table
### Osquery Complex Query Example

```sql
osquery> SELECT datetime(logged_in_users.time,'unixepoch') AS datetime, logged_in_users.type, logged_in_users.user, users.uid, logged_in_users.tty, logged_in_users.pid, processes.name AS process_name, processes.path
   ...> FROM logged_in_users
   ...> LEFT JOIN processes USING(pid)
   ...> LEFT JOIN users ON users.username = logged_in_users.user;
```

<table>
<thead>
<tr>
<th>datetime</th>
<th>type</th>
<th>user</th>
<th>uid</th>
<th>tty</th>
<th>pid</th>
<th>process_name</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-03-26 21:35:00</td>
<td>boot_time</td>
<td>reboot</td>
<td></td>
<td>~</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019-03-26 21:35:13</td>
<td>login</td>
<td>LOGIN</td>
<td></td>
<td>tty1</td>
<td>834</td>
<td>agetty</td>
<td>/sbin/agetty</td>
</tr>
<tr>
<td>2019-03-26 21:35:14</td>
<td>user</td>
<td>user</td>
<td>1000</td>
<td>tty7</td>
<td>1248</td>
<td>sh</td>
<td>/bin/bash</td>
</tr>
<tr>
<td>2019-03-26 21:35:39</td>
<td>runlevel</td>
<td>runlevel</td>
<td></td>
<td>~</td>
<td>53</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Osquery Configuration and Extensions

2019 Pass the SALT Workshop
Using osqueryd

- Osqueryd is the host monitoring daemon
- It aggregates query results over time and generates logs
- Allows to:
  1. Schedule queries
  2. Record OS state changes, including file and directory changes, hardware events, network events, etc.

  ```json
  {  
    "osquery_info": {  
      "query": "SELECT * FROM osquery_info; ",  
      "interval": 300,  
      "snapshot": true
    }
  }
  ```

- Configuration and query schedule
- Logging and reporting
- Query Packs
Flags and Flagfile

- Osqueryi and osqueryd use optional command line (CLI) flags to:
  - Control initialization
  - Disable/enable features
  - Select plugins


- **Flagfile**: flags can be set within environment variables or via a "master" flag file

```
--tls_hostname=ws-vm
--tls_server_certs=C:\ProgramData\osquery\ws-vm.pem
--host_identifier=uuid
--enroll_tls_endpoint=/api/v1/osquery/enroll
--config_plugin=tls
--config_tls_endpoint=/api/v1/osquery/config
--config_tls_refresh=10
--disable_distributed=false
--distributed_plugin=tls
--distributed_interval=10
--distributed_tls_max_attempts=3
--distributed_tls_read_endpoint=/api/v1/osquery/distributed/read
--distributed_tls_write_endpoint=/api/v1/osquery/distributed/write
--logger_plugin=tls
--logger_tls_endpoint=/api/v1/osquery/log
--logger_tls_period=10
--enroll_secret_path=C:\ProgramData\osquery\osquery.secret
```
Configuration

- Osquery "configuration" is read from a config plugin
  - Set to filesystem by default
  - HTTP/TLS request using the tls config plugin
- The response data must be in JSON format

- Components in a configuration include
  - Daemon options and feature settings
  - Query Schedule: the set of SQL queries and intervals
  - File Change Monitoring: categories and paths of monitored files and directories
- Filesystem config plugin default locations:
  - Windows: C:\ProgramData\osquery\osquery.conf
  - Linux: /etc/osquery/osquery.conf and /etc/osquery/osquery.conf.d/
  - MacOS: /var/osquery/osquery.conf and /var/osquery/osquery.conf.d/
• Configuration supports sets of queries called packs
• Packs are distributed with osquery and labeled based on broad categories
• In an osquery configuration JSON
  • Packs can be defined as a top-level-key and consist of pack name to pack content JSON data structures
  • Pack value may also be a string. In case of the filesystem plugin, these strings are considered paths.
Logging

• Osqueryd uses logger plugins:
  • filesystem (default)
  • tls
  • syslog (for POSIX),
  • windows_event_log (for Windows)
  • kinesis
  • firehose
  • kafka_producer

• Log types: status and result logs

• Status logs:
  • Generated by the Glog logging framework
  • Logger plugins may intercept these

• Results logs: Results of scheduled queries are logged to the "results log"
  • Differential logs: Differential changes between the last (most recent) query execution and the current execution
  • Snapshot logs: A snapshot is an 'exact point in time' set of results, no differentials
Eventing Framework

• Scheduled queries have limitations
  • Volatile events like process execution
• To overcome this, osquery has the Eventing (pubsub) Framework
  • Aggregating operating system information asynchronously at event time
  • Storing related event details in the osquery backing store
  • Performing a lookup to report stored rows query time

• Almost every pubsub-based table ends with a _events or _changes
• Note that this reporting pipeline is much more complicated!
  1) Requires additional configuration
  2) As events occur, the rows returned by a query will compound, so queries should always include a time range
  3) The buffered events will eventually expire! Buffer is set to 1 day by default
  4) Eventing Framework will not really work with osqueryi
File Integrity Monitoring
• Available for Linux and Darwin
• The list of files/directories to monitor is defined in the osquery configuration
• Can use standard wildcards "*" or SQL-style wildcards "%" for the path definitions
  • %: Match all files and folders for one level
  • %%%: Match all files and folders recursively

```json
{
  "schedule": {...},
  "file_paths": {
    "homes": [
      "/root/.ssh/%%",
      "/home/%/.ssh/%%"
    ],
    "etc": [
      "/etc/%%"
    ]
  },
  "exclude_paths": {
    "homes": [
      "/home/user/.ssh/%%"
    ]
  }
}
```
Extensions

• Osquery supports proprietary tables, config plugins, and logger plugins
• Thrift-based extensions API
• Osqueryd may "autoload" these extensions and monitor their performance
• Trail of Bits extensions:
  • [https://github.com/osql/extensions](https://github.com/osql/extensions)

• CLI flags for extension auto-loading:
  - `-e autoload=/etc/osquery/extensions.load`
  - `-r timeout=3`
  - `-r interval=3`

• Extensions.load file example (osquery.ext is an executable):
  ```
  /usr/lib/osquery/extensions/osquery.ext
  ```

• Manually Loading Extensions:
  ```
  osqueryi --allow_unsafe --extension /path/to/extension.ext
  ```

Osquery Configuration and Extensions Hands-On

2019 Pass the SALT Workshop
Fleet Management

2019 Pass the SALT Workshop
Fleet Management Options

- **Kolide Fleet:** [https://kolide.com/fleet](https://kolide.com/fleet)
  - (Free) Open Source Software from Kolide: [https://github.com/kolide/fleet](https://github.com/kolide/fleet)
  - Paid: Kolide Cloud (SaaS)
- **Doorman:** [https://github.com/mwielgoszewski/doorman](https://github.com/mwielgoszewski/doorman)
  - (Free) Open Source Software from Marcin Wielgoszewski
- **STG:** [https://github.com/OktaSecurityLabs/sgt](https://github.com/OktaSecurityLabs/sgt)
  - (Free) Open Source Software from Okta
  - "Built Entirely on AWS"
- (osquery-fleet? : [https://github.com/sandstorm/osquery-fleet](https://github.com/sandstorm/osquery-fleet))
About Kolide Fleet (and Kolide Launcher)

- Open Source Osquery Manager
- Compatible with every major platform
- Designed to work with Launcher (Osquery deployment)
- Features:
  - Query dynamic sets of hosts
  - Run queries repeatedly with Packs
  - Create labels populated with hosts matching a query
  - Export results
- `fleetctl`: provides scriptable, CLI based access to osquery on your entire fleet
Kolide Fleet Installation and Configuration

1) Install and configure MySQL
2) Install Redis
3) Generate TLS certificate for Kolide Fleet server
4) Install Kolide Fleet (https://dl.kolide.co/bin/fleet_latest.zip)
5) Configure Kolide Fleet
   a) Create fleet.config
   b) Create MySQL database
   c) Create fleet.service
6) Start Kolide Fleet
Kolide Fleet Interface and Deployment

It's Kinda Lonely In Here...
Get started adding hosts to Kolide.
This can be done individually or across your entire fleet.

- **Add New Host**
- **All Hosts**
- **Description**
- **NEW added in last...**
- **ONLINE**
- **OFFLINE**
- **MIA (offline > 30 days)**
- **LABELS**
- **ADD NEW LABEL**
- **ADD NEW HOST**

Add New Host
Follow the instructions below to add hosts to your Kolide Instance.

- **Manual Install**
  Fully Customize Your Osquery Installation

- **Kolide / Osquery - Install Docs**
  In order to install osquery on a client you will need the following information:

- **Retrieve Osquery Enroll Secret**
  The following is your enroll secret: [Reveal Secret]

- **Download Server Certificate (Optional)**
  If you use the native osquery TLS plugins, Osquery requires the same TLS certificate that Kolide is using in order to authenticate. You can fetch the certificate below:

- **FETCH KOLIDE CERTIFICATE**

RETURN TO APP
Kolide Fleet Hands-On

2019 Pass the SALT Workshop
Osquery and Elastic Stack

2019 Pass the SALT Workshop
Filebeat Configuration

- Filebeat osquery module can be used
- JSON messages can be sent to:
  - Elasticsearch
  - Logstash

```yaml
# Filebeat inputs
filebeat.inputs:
  # Each - is an input. Most options can be set at the input level, so # you can use different inputs for various configurations.
  # Below are the input specific configurations.
  - type: log
    # Change to true to enable this input configuration.
    enabled: true
    # Paths that should be crawled and fetched. Glob based paths.
    paths:
      - /tmp/osquery_result

# Filebeat modules
filebeat.config.modules:
  # Glob pattern for configuration loading
  path: $(path.config)/modules.d/*.yml
  # Set to true to enable config reloading
  reload.enabled: false
  reload.enabled: true
  # Period on which files under path should be checked for changes
  reload.period: 10s

# Logstash output
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```
Logstash Pipeline Configuration

- Logstash file needs to be placed to:
  - /etc/logstash/conf.d/
- /etc/logstash/logstash.yml has:
  - config.reload.automatic: true
  - config.reload.interval: 5s
Kibana Discovery
Osquery and Elastic Stack Hands-On

2019 Pass the SALT Workshop
Questions and Answers

2019 Pass the SALT Workshop
References

• Osquery Website and Osquery Schema
  • https://osquery.io
  • https://osquery.io/schema

• Osquery Docs
  • https://osquery.readthedocs.io

• Kolide Website
  • https://kolide.com

• Elastic Website
  • https://www.elastic.co