setting up the virtual machine

Install Virtual Box (or VMWare Workstation should work as well)

Go to File, Import Appliance, and point the wizard at the .ova file (you can copy the file locally first if you want).

login with learnsoquery/learnsoquery, *or*

After install and start, you can access your virtual box locally via ssh

```
ssh -p 2222 learnosquery@127.0.0.1
```

Catching Up with osquery

A Talk & Workshop

Doug Wilson – Director of Security, Uptycs

FIRST
Improving Security Together

Uptycs

Hi, I'm Doug

~20 years doing "security"
FIRST Speaker 2013, 2014, 2015
Ex-Mandiant, FireEye

Today's Workshop on osquery

Talk turned into interactive workshop

- **Introduce people to what osquery can do**
- **Give a "state the art" about the project and how to get involved**

Designed for folks with no or moderate experience with osquery

Focused on local use of osquery, but happy to take questions on distributed osquery later or after

If you have played with osquery before, the basic SQL may be "known" but there may still be tips and tricks that are new

Why osquery?

Open-source endpoint

Originally developed at Facebook

Ask Questions and Get Answers!

Turns system calls into virtual SQL tables

(**S**tructured **Q**uery **L**anguage)

Universal Endpoint

Cross-platform
Cross-virtualization (level)

Cross-maturity

Design Principles of osquery

Read Only

Only modifies files needed to run

Polite & respectful of privacy

Developer choice of tables/content

Non-intrusive

user mode, controls impact on machine

Universal language of SQL

Tries to be consistent across all OS

(when possible)

Abstract the Operating Systems to SQL

```
ps -ef | grep apached
```

or

```
ps -C apached
```

or

```
ps ax | grep apache
```

But if you standardize on SQL

```
Select * from processes where name like ‘%apache%’
```

acpi_tables
ad_config
alf
alf_exceptions
alf_explicit_auths
alf_services
app_schemes
apps
apt_sources
arp_cache
asl
augeas
authorization_mechanisms
authorizations
authorized_keys
block_devices
browser_plugins
carbon_black_info
carves
certificates
chrome_extensions
cpu_time
cpuid
crashes
crontab
curl
curl_certificate
device_file
device_firmware
device_hash
device_partitions
disk_encryption
disk_events
dns_resolvers
docker_container_labels
docker_container_mounts
docker_container_networks
docker_container_ports
docker_container_processes
docker_container_stats
docker_containers
docker_image_labels
docker_images
docker_info
docker_network_labels
docker_networks
docker_version
docker_volume_labels
docker_volumes
etc_hosts
etc_protocols
etc_services
event_taps
extended_attributes
fan_speed_sensors
file
file_events
firefox_addons
gatekeeper
gatekeeper_approved_apps
groups
hardware_events
hash
homebrew_packages
interface_addresses
interface_details
iokit_devicetree
iokit_registry
kernel_extensions
kernel_info
kernel_panics
keychain_acls
keychain_items
known_hosts
last
launchd
launchd_overrides
listening_ports
lldp_neighbors
load_average
logged_in_users
magic
managed_policies
mounts
nfs_shares
nvram
opera_extensions
os_version
osquery_events
osquery_extensions
osquery_flags
osquery_info
osquery_packs
osquery_registry
osquery_schedule
package_bom
package_install_history
package_receipts
pci_devices
platform_info
plist
power_sensors
preferences
process_envs
process_events
process_memory_map
process_open_files
process_open_sockets
processes
prometheus_metrics
python_packages
quicklook_cache
routes
safari_extensions
sandboxes
shared_folders
sharing_preferences
shell_history
signature
sip_config
smbios_tables
smc_keys
startup_items
sudoers
suid_bin
system_controls
system_info
temperature_sensors
time
time_machine_backups
time_machine_destinations
uptime
usb_devices
user_events
user_groups
user_interaction_events
user_ssh_keys
users
virtual_memory_info
wifi_networks
wifi_status
wifi_survey
xprotect_entries
xprotect_meta
xprotect_reports
yara
yara_events

Universal vs. OS Specific

Osquery was developed on *nix systems

- Reverse problem of a lot of security software
- Ported to Windows later on
- Tries to be universal, but there are specific ~~idiosyncrasie~~s structures of different OS that make this hard

- macOS and Windows have the most OS specific tables
  - Windows is structured differently, period
  - macOS has a lot of customized features that are not standard *nix
  - More “custom” work has gone into tables for Windows & macOS

Overview of Tables in osquery

System Utilities vs. SQL
Over 200 tables

Special Tables

- **`osquery_`** tables
- Events and **`_event`** tables
- Add-on Utilities
  - Augeas, Prometheus, Docker, Extensions and more!

Osquery Files

Binaries: /usr/bin/

- osqueryi
- osqueryd

Config: /etc/osquery/

- osquery.conf
- osquery.flags

Database /var/osquery/

- osquery.db

Logs /var/log/osquery

- osquery.INFO

In depth config docs: https://osquery.readthedocs.io/en/stable/deployment/configuration/

Osquery Flags

There are a LOT of flags – too many to cover here*

```
osquery> select count(*) from osquery_flags;
count(*) = 169
```

--verbose & **--config-check**
debugging

--config_path & **--flagfile**
configuration

--disable_events
event listeners

*** https://github.com/facebook/osquery/blob/master/docs/wiki/installation/cli-flags.md**

Queries vs Query Packs

you'll hear references to both queries and query packs

A **query** is a set SQL request to get a specific set of data:

```
select * from processes
```

A **query pack** is a **group of queries designed to be distributed together**.

We are ***not*** going to cover query packs in depth today. Think of them as a batch of queries that can be distributed together.

Interactive Lab 1
Install, Launch, & Test

FIRST
Improving Security Together

Uptycs

Let’s find osquery

Osquery is already on your VM at /var/tmp/osquery

Otherwise, best place to start is at https://osquery.io/downloads/
If you want to fetch it, it would look like this:

```
wget https://pkg.osquery.io/deb/osquery_3.2.6_1.linux.amd64.deb
```

```
shasum –a 256 osquery_3.2.6_1.linux.amd64.deb
```

Your result should be **3627e6931c97a27439b33147a7ae1496027be789**

If you don’t get the right checksum, try again, or just use the local copy

Install & Verify

To install

```
sudo dpkg -i /var/tmp/osquery_3.2.6_1.linux.amd64.deb
```

To verify

```
$ osqueryi <enter>
osquery>
osquery> select * from uptime; <enter>
```

Shockingly Easy (hopefully)

So, that took what, 2 minutes?

Very easy to get osquery deployed on a host

Works with almost all Package Managers

Used to be a bit trickier on Windows

Likely need to do code signing if you “roll your own.”

Interactive Lab 2
Shell & Simple SQL

FIRST
Improving Security Together

Uptycs

osquery Shell Commands

If you're not still in osquery, go ahead and type **`osqueryi`** again.

Type **`.help`** at the **`osquery>`** prompt to see some of the osquery shell commands.

Some osquery Shell Commands

.exit / **.quit** - gets you out of the shell

.show & **.features** – show you some of osquery's settings in one place

.mode pretty – "default" view that shows SQL "tables"

.mode line – different view that shows one result per line

.tables $search – lists (some) tables currently available in osquery

.schema $table -- shows you the build statement for a table

SQLite Syntax

osquery uses SQLite* as its SQL interpreter

only SELECT statements

Read Only (in osquery core, at least)

a few functions don’t work, most do

*** https://www.sqlite.org/lang.html**

Simple Queries

select * from <table_name>;

```
osquery> select * from uptime;
```
← protip: the semicolon

```
osquery> select
    ...> *
    ...> from
    ...> uptime
    ...> ;
```

Why Run osquery with sudo?

Try this:

```
osqueryi> select * from shadow;
```

What results do you get?

Now do .exit, and try it again running osqueryi with sudo:

```
$ sudo osqueryi
```

What is different? Why?

SQL - Simple Queries

```
select * from users;

select * from users limit 5;
```

protip: (use **`limit 1`** or **`.schema users`** to get column names)

```
select count(*) from users;

select uid, gid, username, description, directory
from users limit 5;
```

SQL – ORDER BY

Let's take what we have here and "sort" the results

```
select uid, gid, username, description, directory
from users limit 5;
```

```
select uid, gid, username, description, directory
from users order by uid asc limit 5;
```
← Protip – *before* **LIMIT**

How would you do this by username? Why was it not by uid in the first place?

SQL - WHERE and LIKE

If you want to get data from a specific row, you can get ones that match data by using **`WHERE`** (protip – quotes!)

```
select uid, gid, username, description, directory
from users where username='systemd-timesync';
```

This gets you the one specific row. However, what if you want all the "systemd" accounts?

SQL - WHERE and LIKE

You can use the **LIKE** operator and wildcards before or after a string to find partial matches

```
select uid, gid, username, description, directory
from users where username like 'system%';
```

Can you figure out how to get the same results using the directory column?

SQL - JOINing Table Data

Take a look at the **`users`** table, and the **`processes`** table

Processes table is very "noisy" – try just a few columns

```
select * from processes limit 1;
select pid, name, cmdline from processes limit 5;
```

You may want to see what the name of a user is for a given process.

What columns do the users and processes table have in common?

SQL - JOINing Table Data

Both tables have a "**`uid`**" column for the user ID number

Let's take the process data we need with user id, and then map the corresponding user name from the users table.

```
select p.pid, p.name, u.uid, u.username
from processes p
join users u on u.uid=p.uid;
```

“consider JOINing against the users table”

```
select * from shell_history;
```

```
W0617 21:41:10.583434  1534 virtual_table.cpp:557]
The shell_history table returns data based on the
current user by default, consider JOINing against
the users table
```

```
select * from shell_history WHERE shell_history.uid
IN (SELECT uid from users);
```

Tables that Require "join against users:"

account_policy_data
authorized_keys
browser_plugins
chrome_extensions
firefox_addons
known_hosts
opera_extensions
safari_extensions
shell_history

Date Functions

```
osquery> .mode line
select local_time from time;
```

```
> local_time = 1529608143
```
<-- in unix epoch time

```
select datetime(local_time, 'unixepoch', 'localtime') as
formatted_time from time;
```

```
> formatted_time = 2018-06-21 15:09:09
```

* https://www.sqlite.org/lang_datefunc.html

Math

```
osquery> select path, type, blocks_available, blocks_size from
mounts where path = '/';

    path = /
    type = ext4
    blocks_available = 22653804
    blocks_size = 4096

osquery> select path, type, round((blocks_available * blocks_size
*10e-10),2) as gigs_free from mounts where path='/';

    path = /
    type = ext4
    gigs_free = 92.79
```

Back to Presentation

Uptycs

Special Tables

`osquery_`

`_events`

Augeas

Prometheus

Docker

ATC

Special Tables - osquery_

tables that start with "**osquery_**" are diagnostic tables for osquery

osquery_

- **events** – shows current event publishers and subscribers
- **extensions** – show registered extensions
- **flags** – show all recognized flags, and current status
- **info** – status of current installation
- **packs** – shows any registered query packs
- **registry** – summary of components registered with osquery
- **schedule** – scheduled queries from config & query packs

Special Tables - _events

_events tables do not work like “normal tables”

```
select * from processes;
```

Show you all processes at a given point in time

Query later, shows you the delta between point 1 in time and point 2

```
select * from process_events;
```

Saves up events after first query at point 1 in time

Query at point 2 in time gets _ALL_ events since point 1, unless buffer has been overwritten

Special Tables - _events

`_events` tables use a pub/sub model

There are a few different event publishers per OS

There are standardized listeners in osquery

NOT consistent across all OS yet, unfortunately

Windows is specifically lacking consistency with others

*** Catching Everything with osquery Events**
https://www.youtube.com/watch?v=yFfWv9wAhyA

Special Tables – Augeas*

Augeas – a separate open-source project*

Reads configuration files into key-value pairs

Used by osquery to make *nix config files parse-able by osquery without having to write a unique table for each one

Lenses

- What comes with osquery
- Rolling your own
- This is also an open-source project – contribute back!

*** http://augeas.net/**

Special Tables – Prometheus*

Osquery is not that great for performance metrics

Prometheus is an open source metrics collection & publishing project

Prometheus has a LOT of metrics it returns

With the Prometheus table, you can query a Prometheus API and get results inside of osquery

*** https://prometheus.io/**

Special Tables – Docker*

Contributed by Uptycs in May of 2017 – **docker_** tables

Converts Docker API calls into osquery tables

- Allows a lot (but not all) of the information from inside a running docker container to be read in the parent container (check terminology)
- There are some gaps
- There are some areas from improvement
- But much simpler than previous options, and allowed osquery to be a viable solution for Dockerized environments without running osquery in each container

*** https://www.docker.com/**

Special Tables – Auto Table Construction*

Contributed by Facebook in April of 2018

Allows you to take a SQLite database that is sitting on disk, and dynamically turn it into a table in osquery without having to write code for each instance

REALLY useful on macOS, where the OS uses a bunch of these for various different configuration options

Needs to be created in config file

*** https://github.com/facebook/osquery/pull/4271**

Extensions

Extensions are code that runs alongside osquery, but is not a part of the osquery code

Can be written in Python or Go (and possibly other languages)

Extensions can do all sorts of things – including things that violate osquery design principles

- Extensions can read things that are not in osquery core tables
- Extensions can now _WRITE_ to the endpoint, changing configurations
- Extensions can allow for the capture of additional forensic data

Files, Monitoring, "Carving" and Compliance

osquery provides several different ways to get information about files

ALL are path-specific!

The **`file`** table give information about a file when you do the query

The **`file_events`** table gives you information about *changes* to specific files and file paths

The **`carves`** table gives you the ability to carve files from an operating system (but not trivial)

Interactive Lab 3 Viewing Events in osqueryi

FIRST
Improving Security Together

Uptycs

osquery_events Table

```
sudo osqueryi
osquery> select * from osquery_events
```

Try to see the entire table – currently all zeros.

Events are turned on/off through config & flags.
Base osqueryi sees none of them

Configuring osqueryi to See Events

For this we need to use flags:

`--disable_events` is default set to "true" – needs to be "false"

`--disable_audit` & `--audit_allow_config` need to be set

The latter two are in the config file. But the first is not.

Launching osqueryi to See Events

We need to feed osqueryi the flags settings.
Run osqueryi as follows (all on one line):

```
sudo osqueryi --disable_events=false
--config_path=/etc/osquery/osquery.conf
--flagfile=/etc/osquery/osquery.flags
```

What flags are being sent via the flags file? What flags are being sent via the command line?

Querying Events

now, take a look at osquery_events again

```
osquery> select * from osquery_events;
```

What is different?

Querying Events

let's refine the events we are looking at

```
select * from osquery_events where active=1;
```

What are we seeing here?

All publishers and subscribers are not equal.

file_events and yara_events require additional config.

File Events & FIM

open a new shell, your choice

navigate to /etc/osquery and examine the osquery.conf file

$ less /etc/osquery/osquery.conf

find the "file_paths" section

This is where File Event / File Integrity Monitoring (FIM) is configured

File Events & FIM

```
"file_paths": {
  "monitor_this": [
    "/var/tmp/filetest/%%"
  ]
}
```

JSON snippet – specifies a label and a file path.

The file path can include % or %% wildcards.
They do NOT always work the way you expect*.

*** https://www.uptycs.com/blog/wildcards-and-globbing-in-osquery**

File Events & FIM

```
"file_paths": {
  "monitor_this": [
    "/var/tmp/filetest/%%"
  ]
}
```

This **will** monitor any file changes underneath the "filetest" directory

cd to the **/var/tmp/filetest** directory
but do NOT do anything there yet.

Let's Create Some Events

in the osquery terminal, look at **`osquery_events`**

you may see some process events already

`osqueryi> select * from file events`

should still return nothing, though

go to your **`filetest`** directory, and create or "touch" a few files.

Let's Create Some Events

in the osquery terminal, look at **`osquery_events`** again
you should now see some file events as well.

`osqueryi> select * from file events`
will return events now. What do you see about them?

Feel free to create (or delete!) a few more files. What do you see?

How many events are there per action that you take?

Let's Create Some (user) Events

looking at **`osquery_events`** you should file and process events

Let's generate some user events.
These often involve authentication (or lack there of).
Go back to your file window.

Type

`passwd`

and then just hit <enter> again. Don't actually change your password.

Let's Create Some (user) Events

Then type

```
sudo fileblah.txt
```

And then at the prompts, don't actually put in your password. Just hit enter, and wait for the timeouts to finish.

Go back to the osquery window, and look at **`user_events`**

What do you see?

If you see events from cron – can you filter them out so you only see your user events? What are some different ways you could do that?

Bonus Question

when querying **`user_events`**, can you also figure out how to display the user name?

can you do this and edit out "cron" jobs at the same time?

Bonus Question

when querying user_events, can you also figure out how to display the user name?

```
select u.username, ue.pid, ue.message, ue.path,
ue.terminal, ue.time from user_events ue join users
u on ue.uid=u.uid;
```

```
select u.username, ue.pid, ue.message, ue.path,
ue.terminal, ue.time from user_events ue join users
u on ue.uid=u.uid where ue.path!='/usr/sbin/cron';
```

Thus Endeth the Lab for Now

Back to the Presentation

osquery project status and overview

Uptycs

osquery – Project Status and Overview

Project status and who's using osquery now

Current pain points and roadmap items

Improvements in the past year

What to expect in the next year

How to get involved

Osquery – Current Project Status

Can osquery become the “Apache for Endpoint?”

One of the top open-source projects for security (as per github)

Immense growth & activity

Adoption in “early adopter” as well as some more conservative ones

Who's Using osquery Now?

DIY community

Adoption in Silicon Valley

Facebook (obviously)

Airbnb, Apple, Etsy, Google, Netflix, Uber, Stripe and many more . . .

Large scale IT companies

Large financial institutions

Even Security Companies

(shout out to Carbon Black)

Current Pain Points and Roadmap Items

Needs “better” QA and reliability

Doesn’t have full functionality in hosted container environments

Doesn’t have great performance counters

Lacking “full” documentation and FAQs

Lacking full “events” for Windows

Current Pain Points and Roadmap Items

~~Needs "better" QA and reliability~~

~~Doesn't have full functionality in hosted container environments~~

Doesn't have great performance counters

Lacking "full" documentation and FAQs

Lacking full "events" for Windows

Important Improvements in the Past Year

Support for hashes in SQLite (tables)

Windows Authenticode Support (table)

Standardized JSON output (overall)

Support for Docker API (tables)

ATC Tables (table)

Dedicated team at Facebook working on performance (overall)

What to Expect in the Next Year w/ osquery

"Write" extensions* ← HUGE

Forensic data support for NTFS file systems*

Integration with Google Santa*

Improved container support (Kubernetes & hosted containers)

Event triggered Response Actions (still to be defined)

Proposed Improved Code Review Process

Likely integrate with EBPF (Extended Berkeley Packet Filter) on Linux

* - These are already written, just not adopted fully yet

How You can Join and Contribute

Download a binary and try it out - **https://osquery.io/downloads/**
Download or clone from Github - **https://github.com/facebook/osquery/**
Join the osquery slack - **https://osquery-slack.herokuapp.com/**
Follow @osquery and contributors on Twitter

Facebook has designated some “starter” items for less experienced coders
Contribute an IOC for malware
Help improve documentation
Write a blog post about solving a problem w/ osquery and share it

The Challenge of osquery at Scale

Uptycs

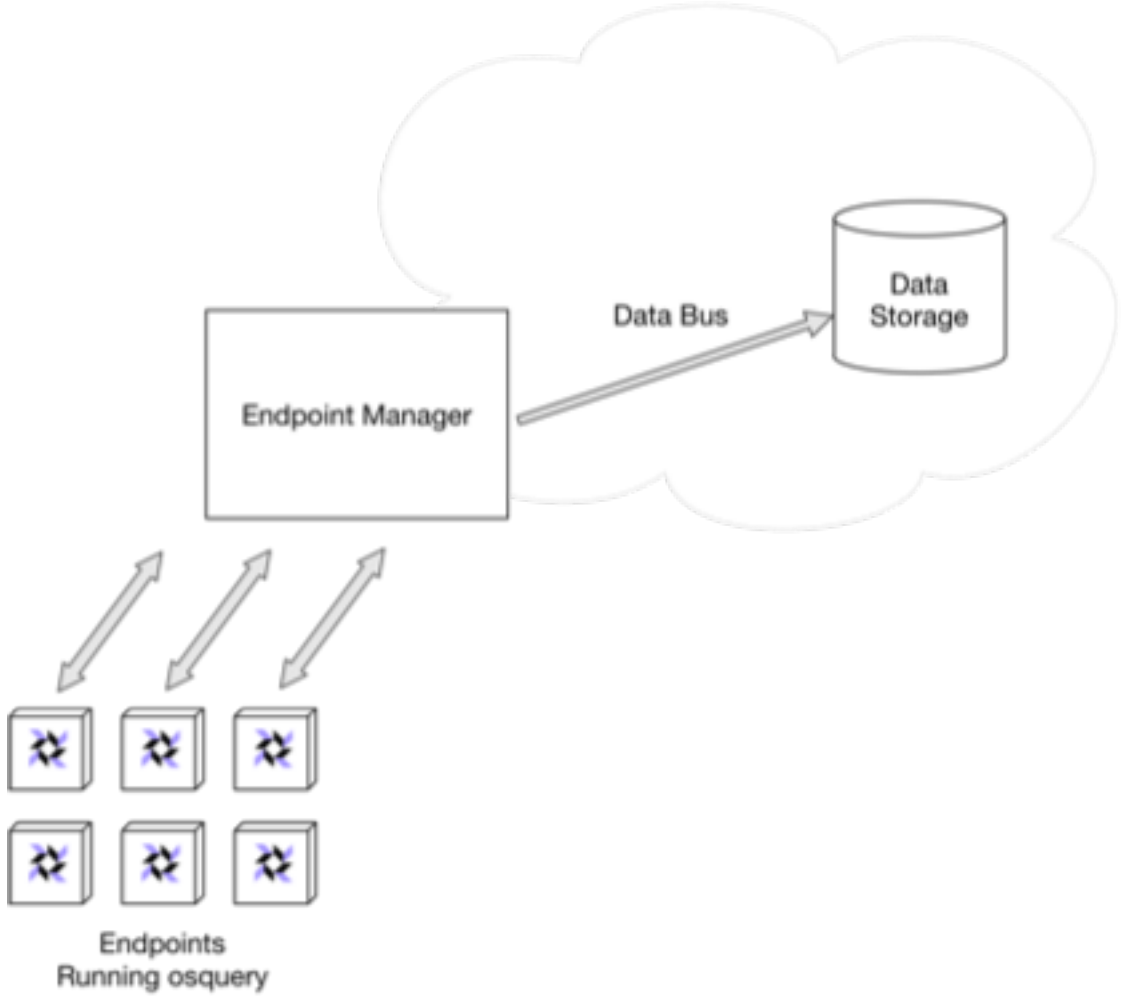
Osquery at Scale – the Challenges

osquery is really cool

but there's a lot more to deploying it a scale beyond the open-source project

You have to create an ecosystem, with osquery as the endpoint

Osquery at Scale – General Architecture



Stitching Components Together

Endpoints	Management	Data Bus	Data Storage
Chef	Zentral	Logstash	Elastic Search
Puppet	Kolide Fleet	Kafka	Postgres
Munki	Doorman	<several more>	<several more>
<many more>	Okta SGT		

And you need to get the data back out

In most cases, you still have to be able to query the data store as well.

If you have a classic ELK stack, you use Kibana.

Wide variety of other choices for how to query and present the data.

Data over time

If you gather enough data over time using osquery, you can recreate most of the state of a machine at different points in time.

Your data store and retrieval method has to take this into account.

Very useful for investigations and timelines.

However, with the **same tool** you can go back and query in real time.

osquery at Scale Summary

you need to configure and deploy the following

- Endpoint Configuration
- osquery Management & TLS or logging server
- Data Pipeline(s)
- Data Store(s)
- UI & Retrieval System
- Timelining System

Classic "Open-Source" time vs. resources trade-off

so, What Have We Learned?

FIRST
Improving Security Together

Uptycs

In summary: osquery . . .

Is a **universal, open-source endpoint** agent that originated at Facebook.

Aims to be **read-only**, "**polite**," to users, and **non-intrusive** to the host.

Is **easy to install** and start using, though **requires additional work at scale**.

Uses **SQLite syntax** for formatting queries, and **has over 200 tables** to format information from endpoints.

Can **gather data** about a point in time, collect series of **events**, and leverage **third party tools** and **extensions**.

Provides the ability to **gather information over time** as well as **ask questions in real time** in the same tool.

Time Check (yes)
Bonus Lab (maybe)

FIRST
Improving Security Together

Uptycs

Thanks for coming!

Doug Wilson
dwilson@uptycs.com
@dallendoug

https://osquery.io
https://osquery-slack.herokuapp.com

https://www.uptycs.com
https://www.uptycs.com/blog
@uptycs
https://medium.com/Uptycs

Interactive Lab 4 Augeas and Prometheus

FIRST
Improving Security Together

Uptycs

Augeas

Augeas can load configuration files into key value pairs

Does this through the **augeas** table

The query MUST have a path to a config file to work efficiently

```
select * from augeas where path='/etc/sudoers';
```

Augeas

```
select label, value from augeas where
path='/etc/sudoers' and label not like '%comment%';
```

```
select label, value from augeas where
path='/etc/ssh/sshd_config' and label not like
'%comment%';
```

/etc/crontab – compare to **'select * from crontab'**

/etc/hosts – compare to **'select * from hosts'**

Prometheus

Prometheus is an open-source monitoring solution

Prometheus exports counters and publishes them to a local API endpoint on a small server it runs.
You can query this API in a table built into osquery.

Let's start up the services

```
sudo systemctl start prometheus
sudo systemctl start node_exporter
```

Prometheus

Prometheus initially just reports on its own Go application.

Node Exporter is a Prometheus module that gathers a large variety of metrics from the computer in question and publishes them into Prometheus.

Prometheus publishes to

http://localhost:9100/metrics
http://localhost:9090/metrics

You can see the entries telling osquery about this in the **`/etc/osquery/osquery.conf`** file.

Prometheus

let's try to take a look at prometheus in osquery.

Prometheus can be _very_ noisy

```
select * from prometheus_metrics limit 10;
```

This is where LIKE statements will come in handy

Prometheus

First you see statistics on the actual Go application.
Let's look for things from the node_exporter and limit columns

```
select metric_name, metric_value from
prometheus_metrics where metric_name like
'node_cpu%';
```

Try **node_disk**, **node_filesystem**, **node_memory**, **node_netstat**, **node_network** to see some of the other sets of info available.

Augeas and Prometheus – and much more

Augeas was one way of extending osquery to read config files

Prometheus is another – for metrics

There is also the **`curl`** table for hitting any specific endpoint (kind of what Prometheus does, but not purpose built)

Extensions can do almost anything you can dream up

Thanks for coming!

Doug Wilson
dwilson@uptycs.com
@dallendoug

https://osquery.io
https://osquery-slack.herokuapp.com

https://www.uptycs.com
https://www.uptycs.com/blog
@uptycs
https://medium.com/Uptycs