HOLDMYBEER

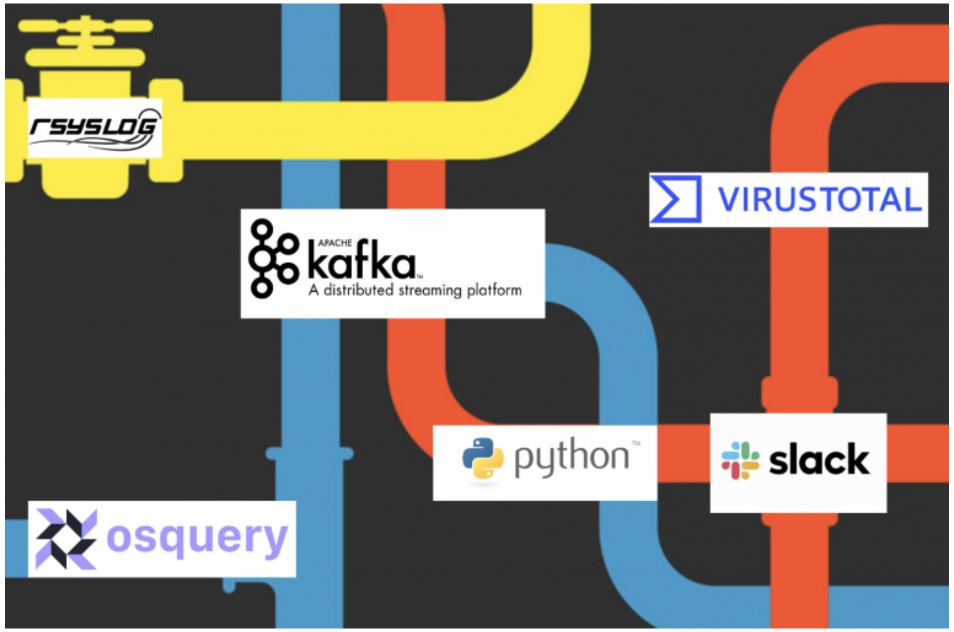
Cause every great story starts with "Hold my beer"

APR 25 2019
LEAVE A COMMENT

BY SPARTAN2194

INCIDENT RESPONSE, LOGGING, POC, TOOLS

DETECTING MALICIOUS DOWNLOADS WITH OSQUERY, RSYSLOG, KAFKA, PYTHON3, AND VIRUSTOTAL



This blog post will explore how to set up a simple logging pipeline to detect maliciously downloaded files. This setup will utilize technologies such as Osquery, Rsyslog, Kafka, Docker, Python3, and VirusTotal for a logging pipeline. If this pipeline detects a malicious file, a Slack alert will be triggered.

Abstract

First, Osquery will monitor file system events for newly created files. Rsyslog client on a macOS endpoint will ship logs to a Rsyslog server. The Rsyslog server will forward the logs to Kafka, and then Kafka will place the logs into a topic to be consumed by our Dockerized Python application. The Python application will extract the file hash from Osquery file events. These hashes will be submitted to VirusTotal for analysis. If VirusTotal reports that the file is malicious, a Slack alert will be triggered.

Goals

- Detect malicious downloads with Osquery and VirusTotal
- Osquery configs

- Learning to use Kafka with Python
- Learn how to leverage VirusTotal to detect malicious files
- Deploying Kafka and Rsyslog server on Docker

Assumptions

This blog post is written to be a proof of concept and not a comprehensive post. This post will **NOT** cover how Osquery, Kafka, or how Docker works, and this post assumes you know how these technologies work. Second, this blog post contains setups and configs that may **NOT** be production ready. The “future improvements” section discusses various improvements for this implementation.

Assumption

Background

What is osquery?

Osquery exposes an operating system as a high-performance relational database. This allows you to write SQL-based queries to explore operating system data. With Osquery, SQL tables represent abstract concepts such as running processes, loaded kernel modules, open network connections, browser plugins, hardware events or file hashes.

What is Rsyslog?

Rsyslog is a rocket-fast system for log processing. It offers high-performance, great security features and a modular design. While it started as a regular syslogd, rsyslog has evolved into a kind of swiss army knife of logging, being able to accept inputs from a wide variety of sources, transform them, and output to the results to diverse destinations.

Rsyslog can deliver over one million messages per second to local destinations when limited processing is applied (based on v7, December 2013). Even with remote destinations and more elaborate processing the performance is usually considered “stunning”.

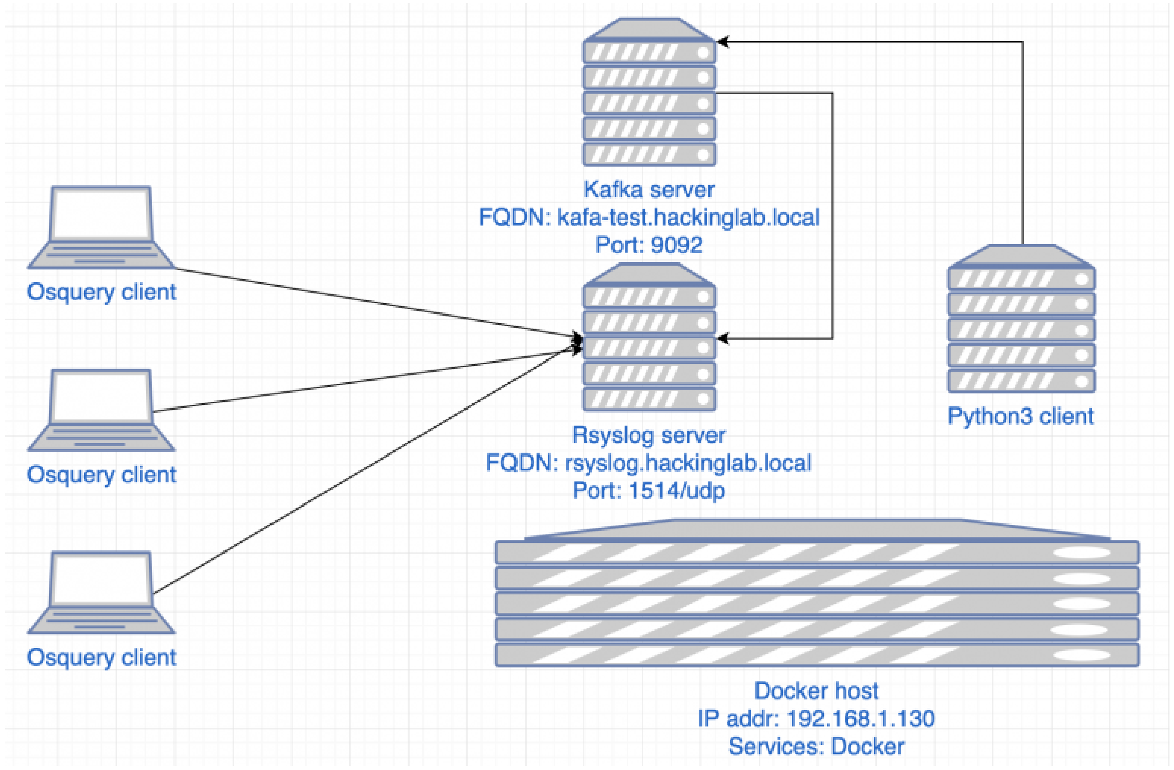
What is Kafka?

Apache Kafka is a community distributed event streaming platform capable of handling trillions of events a day. Initially conceived as a messaging queue, Kafka is based on an abstraction of a distributed commit log. Since being created and open sourced by LinkedIn in 2011, Kafka has quickly evolved from messaging queue to a full-fledged event streaming platform.

What is VirusTotal?

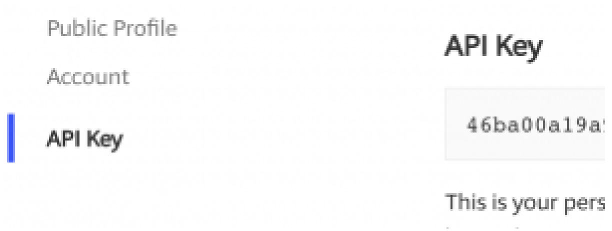
VirusTotal inspects items with over 70 antivirus scanners and URL/domain blacklisting services, in addition to a myriad of tools to extract signals from the studied content. Any user can select a file from their computer using their browser and send it to VirusTotal. VirusTotal offers a number of file submission methods, including the primary public web interface, desktop uploaders, browser extensions, and a programmatic API. The web interface has the highest scanning priority among the publicly available submission methods. Submissions may be scripted in any programming language using the HTTP-based public API.

Network diagram



Obtain VirusTotal API key

1. Browse to https://www.virustotal.com/#/home/upload and create an account
2. Login into your new account
3. Select your profile icon in the top right then select “Settings”
4. Select “API key” on the left



46ba00a19a

This is your pers

A. Copy this API key for later use

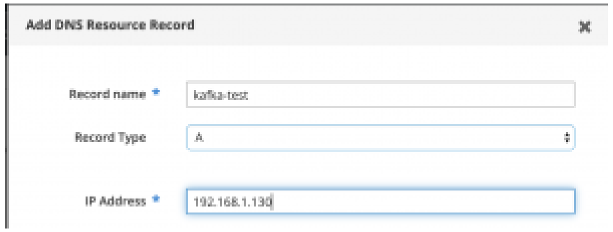
Setup Kafka + Rsyslog with Docker

Kafka

Create DNS A record for Kafka

Kafka needs to register itself to a static IP address **OR** a fully qualified domain name (FQDN – test.google.com). Therefore, you will need to create a DNS A record on your local DNS server that points at the Docker host.

This blog post will **NOT** cover how to set up a DNS A record because each person's DNS setup is different. However, I have posted a photo below of a DNS record for my Kafka container on my FreeIPA server. Within my home lab environment, I have two domains, `hackinglab.local` is used for development. I created an FQDN for `kafka-test.hackinglab.local` pointed at the IP address of 192.168.1.130 which is my Docker host.


Add DNS Resource Record

Record name * kafka-test

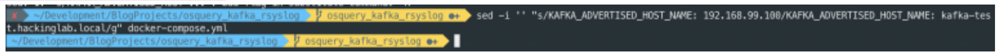
Record Type A

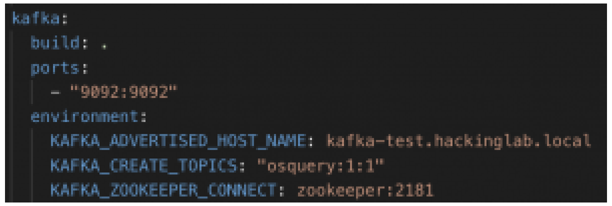
IP Address * 192.168.1.130

Configure Kafka

To keep this blog post short and targeted we will setup Kafka using Docker. This post assumes you know what Kafka is and how to operate it. If you are unfamiliar with Kafka, please take a look at these blog posts: Apache Kafka Tutorial—Kafka For Beginners, Thorough Introduction to Apache Kafka, and How to easily run Kafka with Docker for Development.

1. `sed -i '' "s/KAFKA_ADVERTISED_HOST_NAME: kafka-test.hackinglab.local/KAFKA_ADVERTISED_HOST_NAME: <FQDN for Kafka>/g" docker-compose.yml`


```
~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog ●+  sed -i '' "s/KAFKA_ADVERTISED_HOST_NAME: 192.168.99.100/KAFKA_ADVERTISED_HOST_NAME: kafka-tes
t.hackinglab.local/g" docker-compose.yml
~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog ●+
```

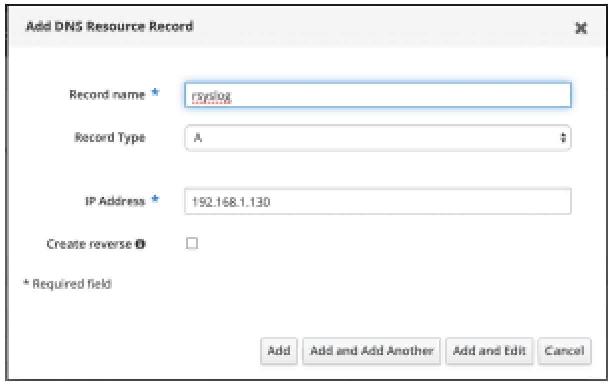


```
kafka:
  build: .
  ports:
    - "9092:9092"
  environment:
    KAFKA_ADVERTISED_HOST_NAME: kafka-test.hackinglab.local
    KAFKA_CREATE_TOPICS: "osquery:1:1"
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
```

Rsyslog server

Create DNS A record for Rsyslog server

I have posted a photo below of a DNS record for my Rsyslog server container on my FreeIPA server. This FQDN will allow Rsyslog clients to send their logs to the Rsyslog server. Lastly, the FQDN `rsyslog.hackinglab.local` is pointed at the IP address of 192.168.1.130 which is my Docker host.


Add DNS Resource Record

Record name *: rsyslog
Record Type: A
IP Address *: 192.168.1.130
Create reverse: ☐

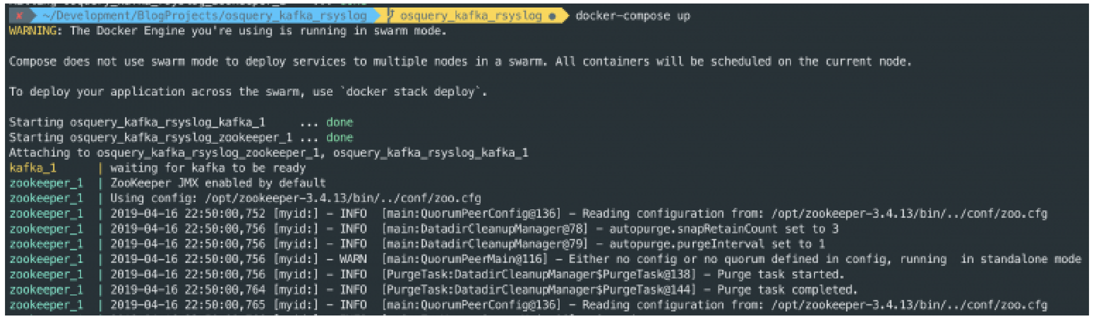
* Required field

Configure Rsyslog

1. `sed -i 's#broker=["kafka-test.hackinglab.local:9092"]#broker=["<FQDN of Kafka>:9092"]#g' conf/rsyslog-server/31-kafka-output.conf`

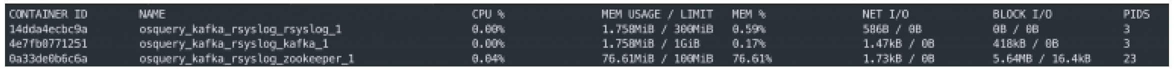
Spin up Docker stack

1. `docker-compose up -d`



```
✘  ~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog ●  docker-compose up
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Starting osquery_kafka_rsyslog_kafka_1     ... done
Starting osquery_kafka_rsyslog_zookeeper_1 ... done
Attaching to osquery_kafka_rsyslog_zookeeper_1, osquery_kafka_rsyslog_kafka_1
kafka_1       | waiting for kafka to be ready
zookeeper_1   | ZooKeeper JMX enabled by default
zookeeper_1   | Using config: /opt/zookeeper-3.4.13/bin/../conf/zoo.cfg
zookeeper_1   | 2019-04-16 22:50:00,752 [myid:] - INFO  [main:QuorumPeerConfig@136] - Reading configuration from: /opt/zookeeper-3.4.13/bin/../conf/zoo.cfg
zookeeper_1   | 2019-04-16 22:50:00,756 [myid:] - INFO  [main:DatadirCleanupManager@78] - autopurge.snapRetainCount set to 3
zookeeper_1   | 2019-04-16 22:50:00,756 [myid:] - INFO  [main:DatadirCleanupManager@79] - autopurge.purgeInterval set to 1
zookeeper_1   | 2019-04-16 22:50:00,756 [myid:] - WARN  [main:QuorumPeerMain@116] - Either no config or no quorum defined in config, running  in standalone mode
zookeeper_1   | 2019-04-16 22:50:00,756 [myid:] - INFO  [PurgeTask:DatadirCleanupManager$PurgeTask@138] - Purge task started.
zookeeper_1   | 2019-04-16 22:50:00,764 [myid:] - INFO  [PurgeTask:DatadirCleanupManager$PurgeTask@144] - Purge task completed.
zookeeper_1   | 2019-04-16 22:50:00,765 [myid:] - INFO  [main:QuorumPeerConfig@136] - Reading configuration from: /opt/zookeeper-3.4.13/bin/../conf/zoo.cfg
```

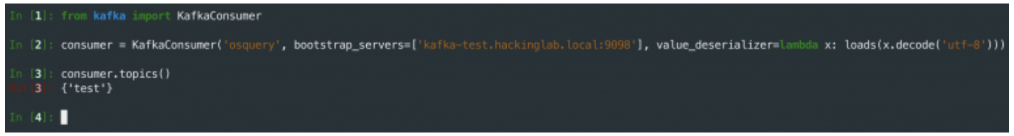
2. `docker stats`



```
CONTAINER ID        NAME                                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
14dda4ecbc9a        osquery_kafka_rsyslog_rsyslog_1     0.00%               1.758MiB / 300MiB     0.59%               586B / 0B           0B / 0B             3
4e7fb0771251        osquery_kafka_rsyslog_kafka_1       0.00%               1.758MiB / 1GiB       0.17%               1.47kB / 0B         418kB / 0B          3
0a33de0b6c6a        osquery_kafka_rsyslog_zookeeper_1   0.04%               76.61MiB / 100MiB     76.61%              1.73kB / 0B         5.64MB / 16.4kB     23
```

Test Kafka

1. `brew install ipython`
2. `pip3 install kafka-python`
3. `ipython`
   A. `from kafka import KafkaConsumer`
   B. `consumer = KafkaConsumer('osquery', bootstrap_servers=['<Kafka FQDN>:9092'], value_deserializer=lambda x: loads(x.decode('utf-8')))`
   C. `consumer.topic()`



```
In [1]: from kafka import KafkaConsumer

In [2]: consumer = KafkaConsumer('osquery', bootstrap_servers=['kafka-test.hackinglab.local:9098'], value_deserializer=lambda x: loads(x.decode('utf-8')))

In [3]: consumer.topics()
Out[3]: {'test'}

In [4]: 
```

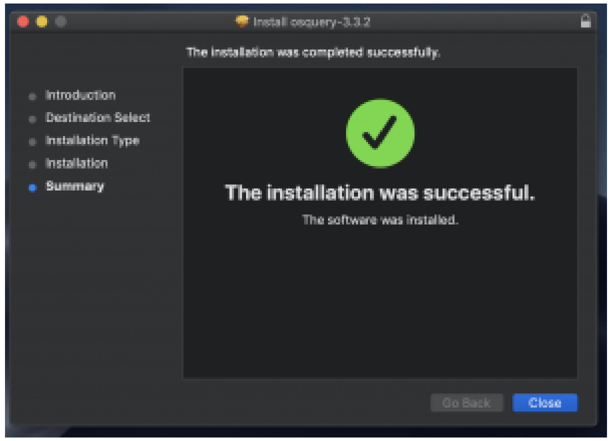
      i. Ignore port 9098

Install/Setup osquery on MacOS

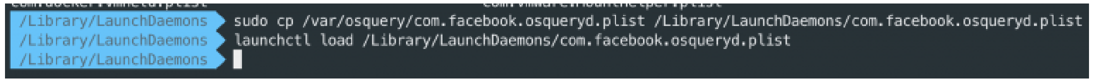
Install/Setup osquery on MacOS

1. Open a browser
2. Browse to https://osquery.io/downloads/official/3.3.2
3. Download the latest osquery installer

4. Install osquery


Install osquery-3.3.2
The installation was completed successfully.
Introduction
Destination Select
Installation Type
Installation
Summary
The installation was successful.
The software was installed.
Go Back
Close

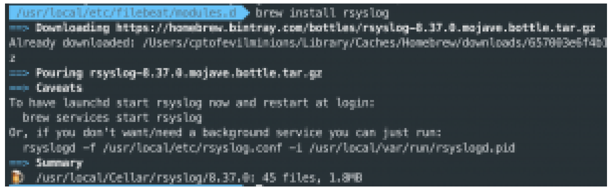
5. `sudo curl https://raw.githubusercontent.com/CptOfEvilMinions/BlogProjects/master/osquery_kafka_rsyslog/conf/osquery/osquery.conf -o /var/osquery/osquery.conf`
6. `sudo curl https://raw.githubusercontent.com/CptOfEvilMinions/BlogProjects/master/osquery_kafka_rsyslog/conf/osquery/osquery.flags -o /var/osquery/osquery.flags`
7. `sudo cp /var/osquery/com.facebook.osqueryd.plist /Library/LaunchDaemons/com.facebook.osqueryd.plist`
8. `sudo launchctl load /Library/LaunchDaemons/com.facebook.osqueryd.plist`



```
 /Library/LaunchDaemons  sudo cp /var/osquery/com.facebook.osqueryd.plist /Library/LaunchDaemons/com.facebook.osqueryd.plist
 /Library/LaunchDaemons  launchctl load /Library/LaunchDaemons/com.facebook.osqueryd.plist
 /Library/LaunchDaemons 
```

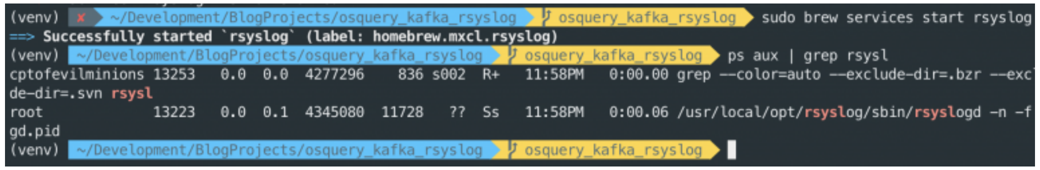
Install/Setup Rsylog client on MacOS

1. `brew install rsyslog`



```
 /usr/local/etc/filebeat/modules.d  brew install rsyslog
==> Downloading https://homebrew.bintray.com/bottles/rsyslog-8.37.0.mojave.bottle.tar.gz
Already downloaded: /Users/cptofevilminions/Library/Caches/Homebrew/downloads/657003e6f4b1
z
==> Pouring rsyslog-8.37.0.mojave.bottle.tar.gz
==> Caveats
To have launchd start rsyslog now and restart at login:
  brew services start rsyslog
Or, if you don't want/need a background service you can just run:
  rsyslogd -f /usr/local/etc/rsyslog.conf -i /usr/local/var/run/rsyslogd.pid
==> Summary
🍺  /usr/local/Cellar/rsyslog/8.37.0: 45 files, 1.8MB
```

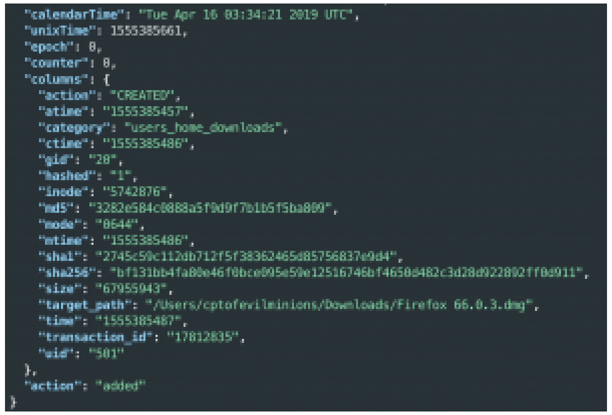
2. `sudo curl https://raw.githubusercontent.com/CptOfEvilMinions/BlogProjects/master/osquery_kafka_rsyslog/conf/rsyslog-client/rsyslog.conf -o /usr/local/etc/rsyslog.conf`
3. `sudo mkdir /etc/rsyslog.d`
4. `sudo curl https://raw.githubusercontent.com/CptOfEvilMinions/BlogProjects/master/osquery_kafka_rsyslog/conf/rsyslog-client/30-osquery.conf -o /etc/rsyslog.d/30-osquery.conf`
   A. `sed -i 's#Target="rsyslog.hackinglab.local"#Target="<FQDN of Rsyslog server>"#g' /etc/rsyslog.d/30-osquery.conf`
5. `sudo brew services start rsyslog`



```
(venv) ✘ ~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog  sudo brew services start rsyslog
==> Successfully started `rsyslog` (label: homebrew.mxcl.rsyslog)
(venv) ~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog  ps aux | grep rsysl
cptofevilminions 13253   0.0  0.0  4277296    836 s002  R+    11:58PM   0:00.00 grep --color=auto --exclude-dir=.bzr --exc
de-dir=.svn rsysl
root             13223   0.0  0.1  4345080  11728   ??  Ss    11:58PM   0:00.06 /usr/local/opt/rsyslog/sbin/rsyslogd -n -f
gd.pid
(venv) ~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog 
```

Test osquery config

1. Open browser
2. Browse to https://www.mozilla.org/en-US/firefox/new/ or any downloading website
3. Download a file
4. `tail -f /var/log/osqueryd.results.log`
   A. May take up to 5 minutes

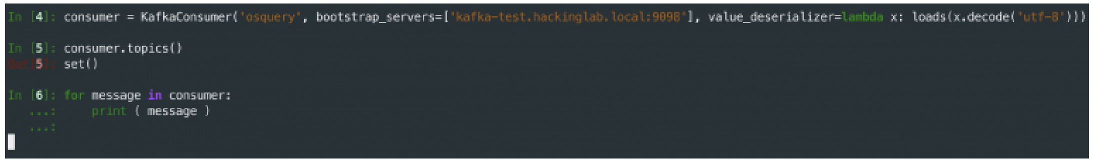


```
  "calendarTime": "Tue Apr 16 03:34:21 2019 UTC",
  "unixTime": 1555385661,
  "epoch": 0,
  "counter": 0,
  "columns": {
    "action": "CREATED",
    "atime": "1555385457",
    "category": "users_home_downloads",
    "ctime": "1555385486",
    "gid": "20",
    "hashed": "1",
    "inode": "5742876",
    "md5": "3282e584c0888a5f0d9f7b1b5f5ba809",
    "mode": "0644",
    "mtime": "1555385486",
    "sha1": "2745c59c112db712f5f38362465d85756837e9d4",
    "sha256": "bf131bb4fa80e46f0bce095e59e12516746bf4650d482c3d28d922892ff0d911",
    "size": "67955943",
    "target_path": "/Users/cptofevilminions/Downloads/Firefox 66.0.3.dmg",
    "time": "1555385487",
    "transaction_id": "17812835",
    "uid": "501"
  },
  "action": "added"
}
```

Pull events from Kafka with kafka-python

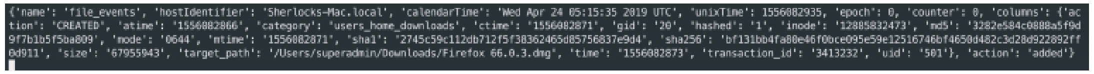
1. `brew install ipython`
2. `pip3 install kafka-python`
3. `ipython`
4.

```python
from kafka import KafkaConsumer
from json import loads

consumer = KafkaConsumer('osquery', bootstrap_servers=['<FQDN of Kafka>:9092'], value_deserializer=lambda x: loads(x.decode('utf-8')))

for message in consumer:
    message = message.value
    print (message)
```



```
In [4]: consumer = KafkaConsumer('osquery', bootstrap_servers=['kafka-test.hackinglab.local:9098'], value_deserializer=lambda x: loads(x.decode('utf-8')))

In [5]: consumer.topics()
Out[5]: set()

In [6]: for message in consumer:
   ...:     print ( message )
   ...:
```

5. Result

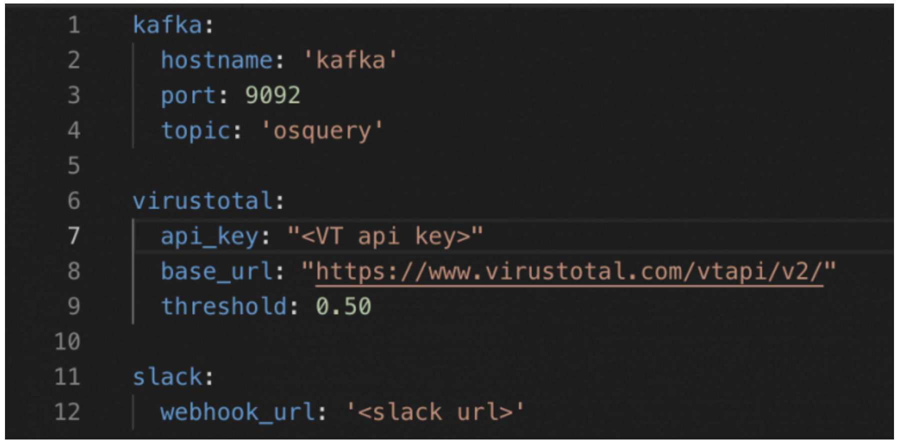


```
{'name': 'file_events', 'hostIdentifier': 'Sherlocks-Mac.local', 'calendarTime': 'Wed Apr 24 05:15:35 2019 UTC', 'unixTime': 1556082935, 'epoch': 0, 'counter': 0, 'columns': {'ac
tion': 'CREATED', 'atime': '1556082866', 'category': 'users_home_downloads', 'ctime': '1556082871', 'gid': '20', 'hashed': '1', 'inode': '12885832473', 'md5': '3282e584c0888a5f9d
9f7b1b5f5ba809', 'mode': '0644', 'mtime': '1556082871', 'sha1': '2745c59c112db712f5f38362465d85756837e9d4', 'sha256': 'bf131bb4fa80e46f0bce095e59e12516746bf4650d482c3d28d922892ff
0d911', 'size': '67955943', 'target_path': '/Users/superadmin/Downloads/Firefox 66.0.3.dmg', 'time': '1556082873', 'transaction_id': '3413232', 'uid': '501'}, 'action': 'added'}
```

Spin up Python app with Docker

Config.yaml

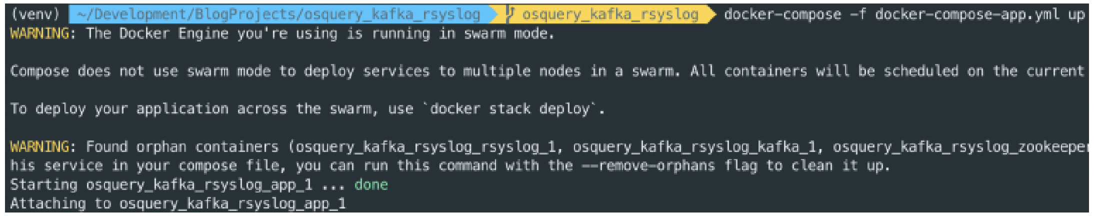
1. `cp app/config/config.yml vim app/config/config.yml.example`
2. `vim app/config/config.yml` and set:
   A. Kafka
      i. `hostname` – Set to the FQDN of Kafka
      ii. `port` – Set to the port of Kafka
      iii. `topic` – Can leave as default
   B. Virustotal
      i. `api_key` – Set API key
      ii. `threshold` – When to generate a Slack alert on a file
         i. The threshold is from 0-1.0 which is positive hits on file/total scanners
   C. Slack
      i. `webhook_url` – URL to send messages too
   D. save and exit



```
kafka:
  hostname: 'kafka'
  port: 9092
  topic: 'osquery'

virustotal:
  api_key: "<VT api key>"
  base_url: "https://www.virustotal.com/vtapi/v2/"
  threshold: 0.50

slack:
  webhook_url: '<slack url>'
```

Spin up app

1. `docker-compose -f docker-compose-app.yml up -d`

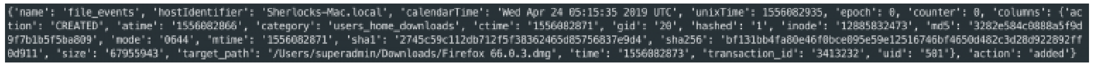


```
(venv) ~/Development/BlogProjects/osquery_kafka_rsyslog  osquery_kafka_rsyslog  docker-compose -f docker-compose-app.yml up
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current

To deploy your application across the swarm, use `docker stack deploy`.

WARNING: Found orphan containers (osquery_kafka_rsyslog_rsyslog_1, osquery_kafka_rsyslog_kafka_1, osquery_kafka_rsyslog_zookeepe
his service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Starting osquery_kafka_rsyslog_app_1 ... done
Attaching to osquery_kafka_rsyslog_app_1
```

Testing setup

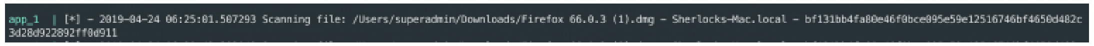
Benign file

1. Open browser
2. Browse to https://www.mozilla.org/en-US/firefox/new/ or any downloading website
3. Download a **NON-malicious** file
4. Kafka-python result



```
{'name': 'file_events', 'hostIdentifier': 'Sherlocks-Mac.local', 'calendarTime': 'Wed Apr 24 05:15:35 2019 UTC', 'unixTime': 1556082935, 'epoch': 0, 'counter': 0, 'columns': {'ac
tion': 'CREATED', 'atime': '1556082866', 'category': 'users_home_downloads', 'ctime': '1556082871', 'gid': '20', 'hashed': '1', 'inode': '12885832473', 'md5': '3282e584c0888a5f9d
9f7b1b5f5ba809', 'mode': '0644', 'mtime': '1556082871', 'sha1': '2745c59c112db712f5f38362465d85756837e9d4', 'sha256': 'bf131bb4fa80e46f0bce095e59e12516746bf4650d482c3d28d922892ff
0d911', 'size': '67955943', 'target_path': '/Users/superadmin/Downloads/Firefox 66.0.3.dmg', 'time': '1556082873', 'transaction_id': '3413232', 'uid': '501'}, 'action': 'added'}
```

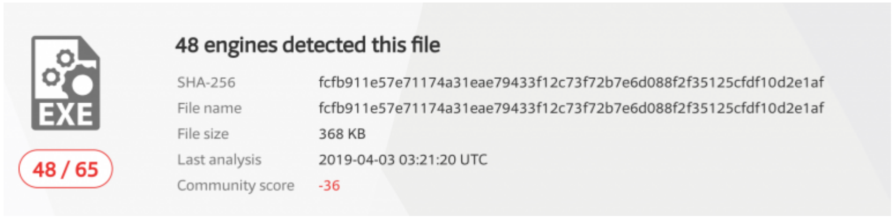
5. App result



```
app_1  | [*] - 2019-04-24 06:25:01.507293 Scanning file: /Users/superadmin/Downloads/Firefox 66.0.3 (1).dmg - Sherlocks-Mac.local - bf131bb4fa80e46f0bce095e59e12516746bf4650d482c
3d28d922892ff0d911
```

Malicious file

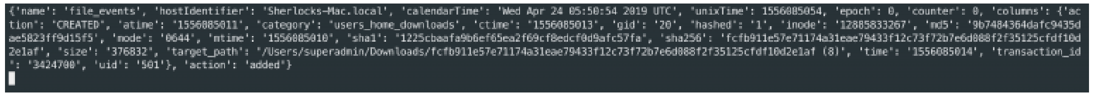
1. Open browser
2. **Link to DOWNLOAD MALICIOUS FILE**



48 engines detected this file

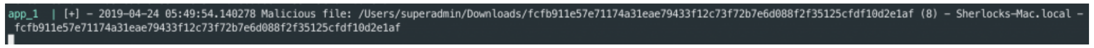
SHA-256	fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af
File name	fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af
File size	368 KB
Last analysis	2019-04-03 03:21:20 UTC
Community score	-36

3. Kafka-python result



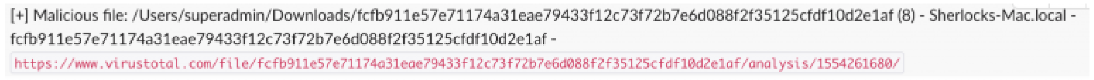
```
{'name': 'file_events', 'hostIdentifier': 'Sherlocks-Mac.local', 'calendarTime': 'Wed Apr 24 05:50:54 2019 UTC', 'unixTime': 1556085054, 'epoch': 0, 'counter': 0, 'columns': {'ac
tion': 'CREATED', 'atime': '1556085011', 'category': 'users_home_downloads', 'ctime': '1556085013', 'gid': '20', 'hashed': '1', 'inode': '12885833267', 'md5': '9b7484364dafc9435d
ae5823ff9d15f5', 'mode': '0644', 'mtime': '1556085010', 'sha1': '1225cbaafa9b6ef65ea2f69cf8edcf0d9afc57fa', 'sha256': 'fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d
2e1af', 'size': '376832', 'target_path': '/Users/superadmin/Downloads/fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af (8)', 'time': '1556085014', 'transaction_id
': '3424700', 'uid': '501'}, 'action': 'added'}
```

4. App result



```
app_1  | [+] - 2019-04-24 05:49:54.140278 Malicious file: /Users/superadmin/Downloads/fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af (8) - Sherlocks-Mac.local -
 fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af
```

5. Slack result



[+] Malicious file: /Users/superadmin/Downloads/fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af (8) - Sherlocks-Mac.local - fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af - `https://www.virustotal.com/file/fcfb911e57e71174a31eae79433f12c73f72b7e6d088f2f35125cfdf10d2e1af/analysis/1554261680/`

Future improvements

Osquery detection

This proof of concept(PoC) only monitors the User's download folder. In an enterprise environment, I would recommend monitoring the user's e-mail directory for e-mail attachments and potentially the user's entire home folder. However, this type of monitoring will generate **A LOT** of noise. I would also recommend generating a list of file types you would like to monitor for such as: .dmg, .docx, .pkg, .xlsx, and etc.

Osquery is detection, not prevention

```
"schedule": {
    "file_events": {
      "query": "SELECT * FROM file_events WHERE action=='CREATED' AND (
target_path NOT like '/Users/%/Downloads/Unconfirmed%' AND target_path
NOT like '/Users/%/Downloads/.com.google%');",
      "removed": false,
      "interval": 300
    }
```

This code segment was taken from `osquery.conf`. The `file_events` query is set to run every 300 seconds (once every 5 minutes). This means it will review the kernel's file events every 300 seconds looking for events that match our query. Because of this, you will always be 5 minutes behind detecting a malicious download. Secondly, Osquery is a detection tool and **NOT** a prevention tool. Therefore, if this pipeline detects a malicious file, it will **NOT** delete the file, nor will it stop the user from interacting with it.

Osquery file carving

A great feature to add to this Python app would be a trigger to initiate an Osquery file carving event. File carving is when you instruct Osquery to zip up a file on an endpoint and send the zip to a server. This would reduce the incident response team having to manually do this and hopefully we can obtain the malicious download before the user/malware deletes it.

References

- USING THE OSQUERY CARVER TO PULL FILES
- Github – Osquery: Filesystem hash data results
- imudp: UDP Syslog Input Module
- Github – wurstmeister/kafka-docker
- Kafka-Python explained in 10 lines of code
- osquery – Install on OS X
- osquery – kafka

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

← Previous post Next post →

Search

RECENT POSTS

- PoC: Monitoring user browser activity with Osquery
- Back in the saddle: Install/Setup Elastic stack 7.0 on Ubuntu 18.04
- Detecting malicious downloads with Osquery, Rsyslog, Kafka, Python3, and VirusTotal
- Detecting SSH brute forcing with Zeek

RECENT COMMENTS

- Lee on Part 1: Install/Setup Wazuh with ELK Stack
- Amit Srivastav on Install/Setup Doorman + OSQuery on Windows, Mac OSX, and Linux deployment
- Corbin on Part 1: Install/Setup Zeek + pf_ring on Ubuntu 18.04 on Proxmox 5.3 + openVswitch

- Part 1: Install/Setup Zeek + pf_ring on Ubuntu 18.04 on Proxmox 5.3 + openVswitch

- frank on Tales of a Blue Teamer: Detecting Powershell Empire shenanigans with Sysinternals
- spartan2194 on Detecting SSH brute forcing with Zeek

ARCHIVES

- October 2019
- May 2019
- April 2019
- March 2019
- February 2019
- December 2018
- November 2018
- June 2018
- April 2018
- January 2018
- December 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- February 2017
- January 2017
- December 2016
- November 2016
- September 2016
- June 2016

CATEGORIES

- ForFunAndWumbos
- Honeypot
- How to red team
- Incident Response
- Logging
- Malware Analysis
- Memory Forensics
- Network Security
- Pen Testing
- PoC
- Red Teaming
- System Administration
- Tales of a Blue Teamer
- Tales of a Red Teamer
- Threat Hunting
- Threat Intelligence
- Tools
- Uncategorized

META

- Log in
- Entries RSS
- Comments RSS
- WordPress.org