Global Information Assurance Certification Paper

Exploring Osquery, Fleet, and Elastic Stack as an Open-source solution to Endpoint Detection and Response

Author: Christopher Hurless, christopher.hurless@gmail.com

Advisor: Lenny Zeltser

GIAC (GCIA) Gold Certification

Accepted: August 31, 2019

Abstract

Endpoint Detection and Response (EDR) capabilities are rapidly evolving as a method of identifying threats to an organization's computing environment. Global research and advisory company, Gartner defines EDR as: "Solutions that record and store endpoint-system-level behaviors, use various data analytics techniques to detect suspicious system behavior, provide contextual information, block malicious activity, and provide remediation suggestions to restore affected systems" (Gartner, 2019). This paper explores the feasibility and difficulty of using open-source tools as a practical alternative to commercial EDR solutions. A business with sufficiently mature Incident Response (IR) processes might find that building an EDR solution “in house” with open-source tools provides both the knowledge and the technical capability to detect and investigate security incidents. The required skill level to begin using and gaining value from these tools is relatively low and can be acquired during the build process through problem deconstruction and solution engineering.

1. Introduction

A business that has already implemented "Incident Response" (IR) processes and is now considering adding "Endpoint Detection and Response" (EDR) to reduce investigation times and improve endpoint threat detection is a business that is facing a solution that can be costly and time-consuming. While EDR provides greater endpoint visibility, finding actionable data from EDR systems can be challenging as it requires qualified staff to make sense of logged events and remove false positives. Performing investigations using an EDR system is an advanced skill set that must be present, developed, or acquired. There are many paid and managed EDR solutions, but it is also possible to implement EDR using open-source technologies that system administrators or security analysts with minimal understanding could setup and configure using readily available documentation as detailed later in this paper.

In a blog post, "Build vs. Buy: Not Mutually Exclusive," Chief Security Officer and Co-Founder of Red Canary Keith McCammon points out that "Builders prioritize flexibility and control" and that "This is doubly true as it relates to security solutions. By definition, an effective security solution requires an understanding of both the target as well as the new system designed to defend that target. Building demands deconstructing the problem you're trying to solve and a deep understanding of the problem's environment" (McCammon, 2018). With the idea in mind that building an EDR system is a means of acquiring the knowledge to operate that system, it becomes reasonable to take steps to determine if an open-source installation of Osquery, Fleet, and Elastic Stack is a viable solution for EDR.

2. Overview of Osquery, Fleet, Elastic Stack

Deconstructing and understanding each of these open-source tools is an essential step in understanding how they would work together to create a solution for EDR. Osquery offers the ability to query endpoints. Fleet is a control server for Osquery, which manages multiple endpoints at once. Elastic Stack is for the collection and visualization of logs collected from the endpoints.

2.1. Osquery in Depth

Osquery is an open-source multi-platform (Windows, Linux, Mac) application created by Facebook that enables the ability to query a database of recorded system states. The installation on its own is lightweight and makes no assumptions for basic queries. A wealth

of queries publicly shared and available to use, both at Osquery's command-line interface "osqueryi" or using the persistent daemon "osquery" are available. Osquery's GitHub page lists many examples (Osquery, 2019).

The Osquery schema documentation (Facebook, 2019) is well written and readily available online. The schema contains information about running processes, installed applications, startup items, network connections, and many more. The power of Osquery, combined with an active user base, is making this tool more accessible to a wide range of security analysts and IT professionals. Facebook has made its list of starter queries freely available on the Github site, which hosts Osquery; you can find these in the "packs" directory (Osquery, 2019). With these freely available queries alone, a great deal of information on endpoints is available with no immediate skill required to get started. Figure 1 shows an example of a scheduled query that would run from the "osquery.conf" file on the endpoint system.

This query would log all users on a given system every 86400 seconds (1 Day).

```
{
"scheduled_query": {
    "users_snapshot": {
        "query": "SELECT * FROM users;",
        "interval": 86400,
        "snapshot": true,
        "description": "Retrieves all users accounts on the system.",
        }
    }
}
```

Figure 1: osquery.conf example

The query in Figure 2 would log all users changed since the last snapshot on a given system every 3600 seconds (60 minutes).

```
{
"scheduled_query": {
    "users_differential": {
        "query": "SELECT * FROM users;",
        "interval": 3600,
        "description": "Retrieves any new user accounts created since the
last snapshot of the users table."
        }
    }
```

```
}
```

Figure 2: osquery.conf example

With several methods for installation available such as the standalone installer via the osquery.io/downloads page, Chocolatey on Windows, brew on Mac, or Kolide's "launcher" addon for Fleet (Kolide, 2019), there are many ways to approach the installation and configuration of Osquery.

An unconfigured installation of Osquery is simple on every platform (Windows/Mac/Linux) if using the stand-alone installer. Configuration of the Osquery daemon and scheduled queries can be a bit more complicated as there are several more moving parts. The standard set of queries provided on the Osquery GitHub page (Osquery, 2019) provides several query packs that are available as a starting point for a new installation.

2.2. Fleet in Depth

Fleet manages Osquery through the creation of "queries" and "packs." Queries are the same as those that would be presented directly to Osquery, or Osqueryi. Once an Osquery client connects to Fleet, some significant changes happen. First, it starts sending all of its logs to the Fleet server, and second, it allows the Fleet server to define and schedule queries to run against the endpoints. For investigative and testing purposes, single run queries can be run from Fleet with the output placed directly on the screen rather than written to logs. Using this feature is excellent for testing queries before putting them in the schedule or for investigations that need some quick answers to important questions.

A query pack is a collection of queries and target endpoints. The pack defines, by name or by platform, which hosts should subscribe to a set of queries. Within the pack, the queries have some additional traits such as "the interval between runs," "which platforms", and "which versions" of Osquery should run a given query. Therefore, within a pack, there can be both Windows and Mac hosts. There are queries which run only on Windows, or only on Mac, but some are OS agnostic and run on many different operating systems. This capability is handy for grouping all of your compliance queries into a single pack, without having to differentiate the type of query by the platform.

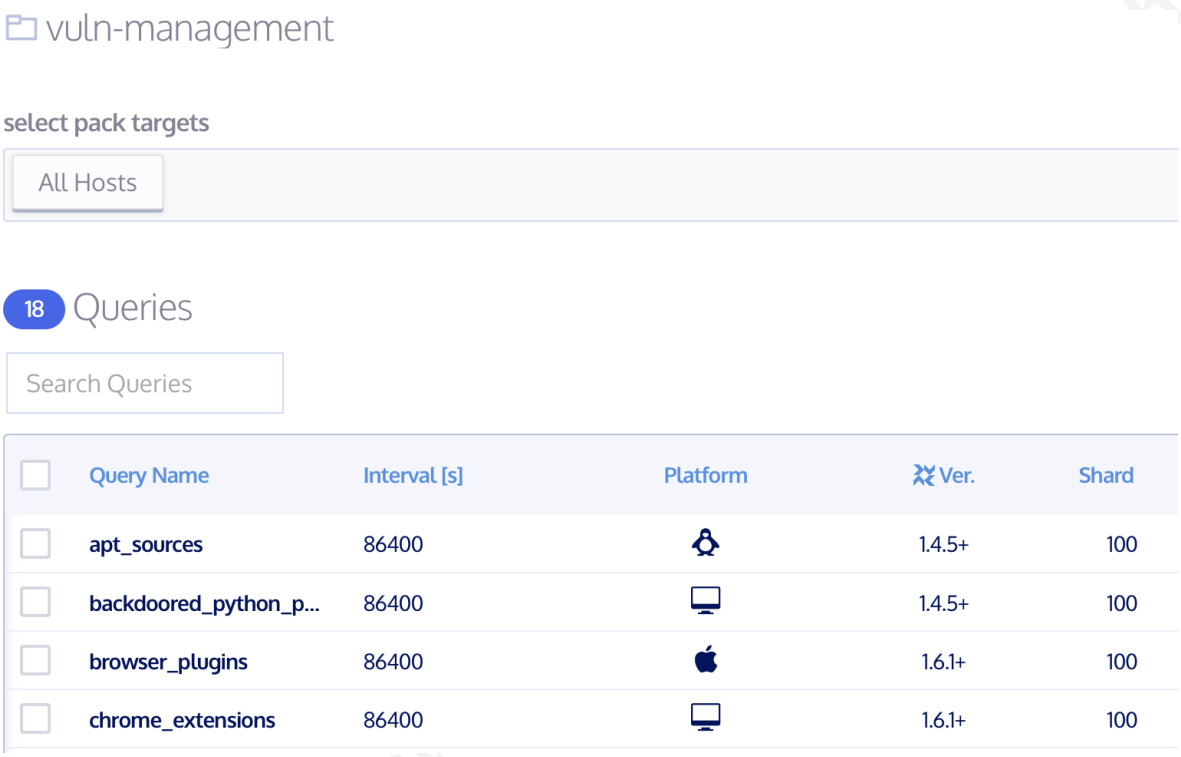

Figure 3: Fleet query pack example

One convenient feature of Fleet is the "fleetctl" binary that takes existing Osquery configuration files and imports them into "Fleet ready" YAML files. Converting the default Osquery packs on the Osquery GitHub site means the initial query pack configuration is mostly complete after this step.

The documentation on the Fleet GitHub page is accurate and easy to follow. Instructions include operating systems such as Centos, Ubuntu, and Kubernetes (Kolide, 2019). Because of the quality of documentation, the initial setup requires little or no skill to reproduce, merely following step-by-step instructions to complete. Configuration of query packs can initially be done using a wealth of available resources, including the query packs included on the Osquery GitHub page (Osquery, 2019).

2.3. Elastic Stack in Depth

Elastic Stack is three individual projects that have been built by Elastic, a single open-source company. These three projects—Elasticsearch, Logstash, and Kibana—combine to form an enterprise-grade search platform akin to Splunk and Graylog. The application Filebeat acts as a log collector and shipper. Filebeat runs as an agent and monitors specified locations for log files and then forwards those log files to Logstash or Elasticsearch for indexing.

Elasticsearch is a scalable, database-driven, multi-tenant log search tool built in Java. It accepts unstructured data and presents it to an HTTP RESTful API using JSON for data exchange. This ability makes Elasticsearch ideal for analyzing large volumes of unstructured and semi-structured data.

Logstash is a data processing tool that takes any log type you send it via Filebeat or other means and normalizes it and then stores it. It's the engine that sits in between Elasticsearch and the front-end tool, Kibana.

Kibana is a web-based front end for creating indexes, reports, and visualizations. It accesses data directly from the Elasticsearch cluster and gives users the ability to create charts and graphs from large volumes of indexed data.

The setup and installation process for Elastic Stack is well and widely documented. Digital Ocean provides several setup pages for installation and configuration on multiple Linux platforms (Digital Ocean, 2019). Instructions are also available for Ubuntu LTS 14.04, 16.04, and 18.04. Because of the quality of these instructions, the initial setup requires little or no skill to reproduce. Ongoing maintenance, however, requires some understanding of Linux and Apache.

2.4. Osquery, Fleet, and Elastic Stack Unified

A platform-agnostic tool for querying system state and tracking change, on its own, is a powerful capability for an open-source tool like Osquery. Fleet's ability to manage, schedule, and aggregate Osquery results to a central location such as Elastic Stack can dramatically enhance the usefulness of both Osquery and Fleet. Several commercial EDR providers, such as Carbon Black (Carbon Black, Inc, 2019) and AlienVault (AlienVault, 2019) use Osquery as a sensor for their solutions.

While Osquery, Fleet, and Elastic stack are straightforward to install individually, integration of these three technologies introduces its own set of challenges. Jordan Potti's blog post entitled "Elk + Osquery + Kolide Fleet = Love" contains instructions for the installation and configuration on Ubuntu 16.04. Most notably is the configuration of the Filebeat application, which is used to move logs from Fleet into the Elastic Stack (Potti, 2018). Connecting Osquery to Fleet is simple to do by configuring the “osquery.flags” file with the correct location of the Fleet server and its SSL certificate.

Configuring logs to be moved from Fleet to Elastic Stack is not well documented and may require some additional time and understanding to get working as needed. Connecting

Osquery to Fleet is well documented, and while some steps require additional customization for your environment, the process is simple to understand and replicate.

For step-by-step instructions for full Osquery, Fleet, and Elastic Stack installation used in this research, see the Appendix.

3. Research Methodology

Lab setup begins with a host computer running VMware Fusion Professional version 11.1.0. The hardware is an iMac (Late 2015, 4GHz Intel Core i7, 32GB 1667 MHz DDR3) running MacOS Mojave version 10.14.5. This machine runs six virtualized operating systems or "guests" used for experimentation.

Hostname	IP	Role	OS	CPU	RAM
fleet-elastic	192.168.13.128	Elastic Search/Fleet Server	CentOS 7.6	1	4096
osq01-win7	192.168.13.129	osquery	Windows 7sp1 Ent, 64-bit	1	2048
osq02-win8	192.168.13.130	osquery	Windows 8 Ent, 64-bit	1	2048
osq03-win10	192.168.13.133	osquery	Windows 10 Pro, 64-bit	2	2048
osq04-osx1013	192.168.13.132	osquery	Mac OS X 10.13.6	2	2048
osq05-osx1012	192.168.13.131	osquery	Mac OS X 10.12.5	2	2048

The host named "fleet-elastic" runs both the Fleet and the Elastic Stack services. In a production environment, these would likely run on separate machines. Fleet packs and queries installed are available on Facebook's Osquery GitHub page (Osquery, 2019). These query packs are converted from .conf to .yaml and then imported into the Fleet server using the fleetctl. These imported query packs run on the endpoints Osquery and generate logs that forward to the Elastic Stack. Indications of unwanted software installations would present in the Elastic Stack visualizations module called Kibana. Investigations based on log data may require additional queries and new visualizations to represent how useful these tools can be at catching and displaying information about endpoints.

Each VMware guest is running an unpatched fresh install of the operating system version indicated by its hostname. System malware defenses are disabled. The Osquery agent

was installed via soquery.io/downloads and configured manually. Full step-by-step instructions for setup of the fleet-elastic server and the installation of Osquery on Windows and Mac is available in the Appendix.

3.1 Experiment Process

Lenny Zeltser's webpage on "Free Malware Sample Sources for Researchers" (Zeltser, 2019) provides a wealth of information and the locations used for finding the malware samples needed for this research. Each sample includes its common name, MD5 hash, and a brief description included. Each sample is installed onto a VMware guest. The Elastic Search logs will then be read to find indications of the malware sample installation. If an indicator of the installation is found in Elastic Search, an investigation will be performed using Osquery as the primary tool for uncovering additional details and attributes such as files created, and system settings changed. Based on the investigation results, a final report will include:

- Whether or not the default Osquery packs queries detect the malware.
- What attributes was Osquery able to uncover about an installed malware.
- How were Fleet and Osquery used for the investigation?
- Without prior knowledge of the installation being malware, would this indicator begin an investigation?
- Would creating additional queries based on the investigation results improve the ability of this solution to detect future installations?
- What is the skill level required to complete this investigation?

3.2 Experiment 1: Operation AppleJeus: Lazarus (Mac Version)

For the first experiment, AppleJeus: Lazarus is installed onto a MacOS virtual machine running version 10.13. AppleJeus: Lazarus is a trojanized cryptocurrency trading application that is installed by tricking an unsuspecting person into believing it is a legitimate program (GReAT, 2018). Elastic Stack logs will be monitored to find if any queries provided by the default installation from the Osquery GitHub page (Osquery, 2019) indicate that the trojan has been installed. If an indicator is found, subsequent queries will be run to find additional traits of the malware.

Malware sample can be obtained from Objective See's website. https://objective-see.com/downloads/malware/AppleJeus.zip (Objective See, 2019)

The hash for the sample is:

sha256 = 7a52d986cfab16abd0c6197a3c1f10299124c1c8738a148898591bc8a717b29a

The malware "Trojanized cryptocurrency application" (GReAT, 2018) is installed voluntarily by the user as it appears to be a functional crypto-trading application. Because the application "CelasTradePro.app" is copied into the /Applications/ directory, administrator-level privileges are required. When the application is run, a new startup daemon is created "/Library/LaunchDaemons/com.celastradepro.plist" which launches a persistent app /Applications/CelasTradePro.app/Contents/MacOS/Updater. This "updater" attempts to start an outbound connection on port 443 to www.celasllc.com.

3.2.1 Indicators of AppleJeus: Lazarus Installation

The Osquery query pack "it-compliance" runs the query "Installed applications" a straightforward differential query.

```
osqueryi> SELECT * FROM apps;
```

According to the Osquery schema, the apps table lists "OS X applications installed in known search paths (e.g., /Applications)" (Facebook, 2019). Applications installed outside of known application installation paths would not be detected with this search. However, the "file" table in Osquery can watch specific folders looking for changes or specific file types.

In Figure 4, you can see the output from Kibana, the visual front end for Elastic Stack, which shows a logged change to "installed applications."

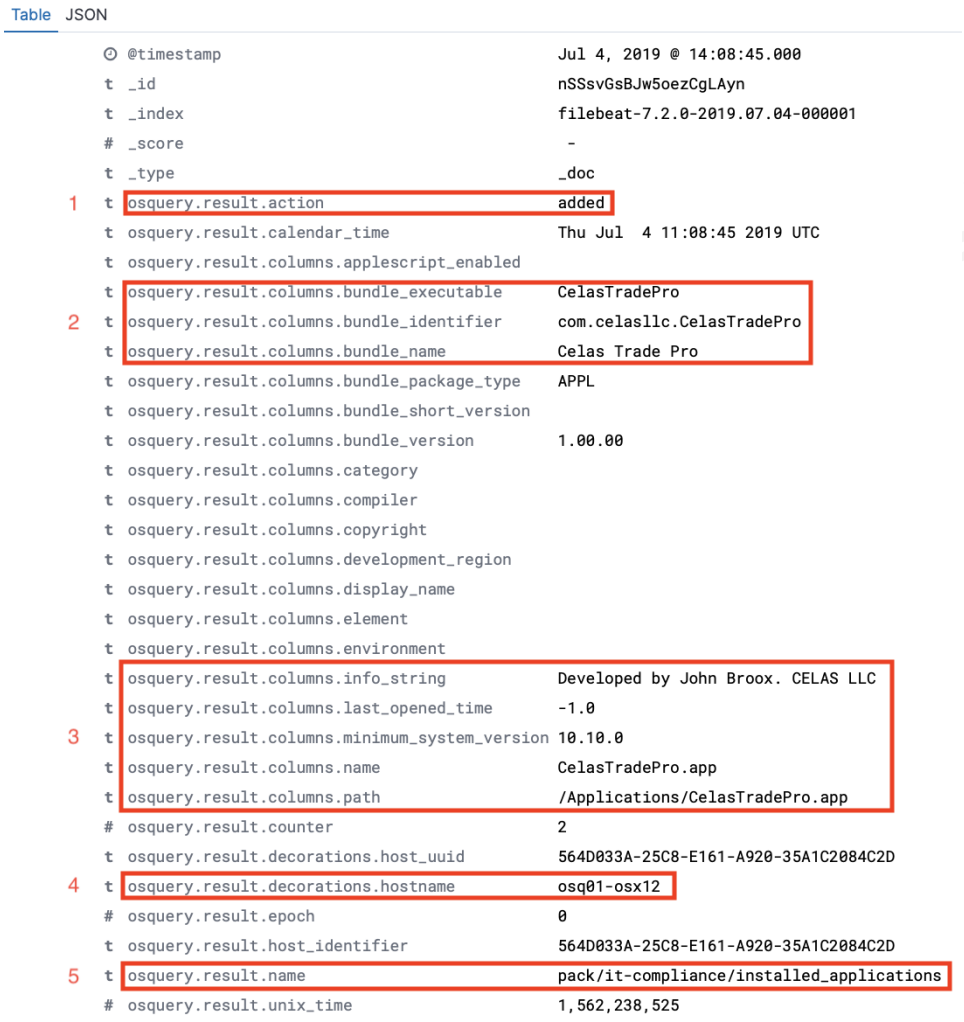

Figure 4: Elastic Stack log visualization

- Figure 4.1: osquery.result.action Shows the action taken, in this case, “added,” meaning this entry did not exist in the previous run.
- Figure 4.2: osquery.result.columns.bundle.* A first look at some characteristics of the newly installed application, based on its bundle attributes.
- Figure 4.3: osquery.result.columns.last_opened, which shows the last time the application was opened, a -1.0 is never otherwise an epoch time will be present. Also osquery.result.columns.path shows where the application is installed. Some applications seen will be installed in ~/Downloads, or some other nonstandard directory creating a near-immediate need for an investigation.
- Figure 4.4 osquery.result.decorations.hostname is the name of the endpoint, which recorded this result.
- Figure 4.5: osquery.result.name shows the Fleet "pack" and "query" that was run from the fleet server, which produced this result. For more details on the exact query, you

can reference back to your Fleet server or find the query pack run entry in Kibana which also shows the SQL of the query pack.

This malware is trying to hide in plain sight by looking like a legitimate application that connects to what appears to be an official website with its update daemon running in the background. With no prior knowledge of this being malware, this application could easily survive many people looking at it without concern or discovery.

3.2.2 Using Fleet or Osqueryi to Start an Investigation

If an investigation was started for this newly discovered application, there are several options for the next step, looking deeper into the Kibana logs, running additional queries from Fleet, or running queries from the endpoint directly using Osqueryi. Each method has its own merits. The Fleet server is handy because it can search across the entire organization or just a specific subset of endpoints all at once without interruption to the operations of the endpoints being queried. Ad hoc queries in the Fleet server are logged to Elastic stack giving a record of the process. However, the query results are not logged. This gives a record of the queries, but for results logged, a new pack query will need to be created and put into the schedule. Osqueryi is quick, but its drawback is the need to have access to the endpoint, in person or via a command shell. None of the osqueryi results are logged.

Consider this query, first from Fleet, then from Osqueryi.

```
osqueryi> SELECT name, path FROM launchd WHERE path LIKE '/Library/%';
```

The "launchd" table returns LaunchAgents and LaunchDaemons from the default launch paths in MacOS. Considering MacOS runs launch items from three locations, this location, "/Library/" is responsible for processes that run for all logged-in users and is the location typically associated with an installation that was installed with privilege. Since CelasTradePro.app was installed into /Applications/ it can be inferred that the application was installed with privilege.

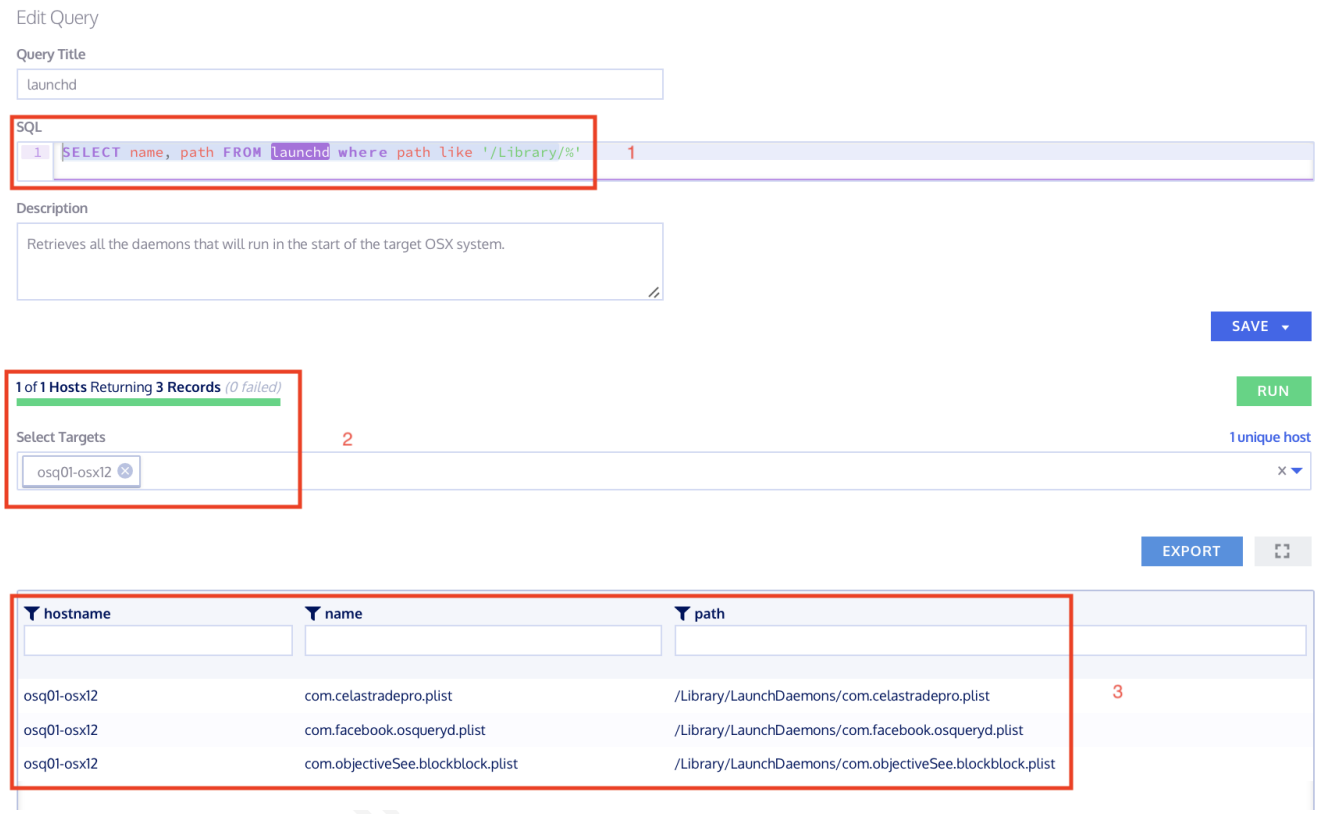

Figure 5: Fleet query for launch daemons

- Figure 5.1: The previously stated query – 'SELECT name, path FROM launchd WHERE path LIKE '/Library/%'; ' any query can be run from this Fleet screen.
- Figure 5.2: Any number of hosts can be selected to run this query. The choice is by ALL, OS, or Specific hosts. This one is just a specific host.
- Figure 5.3: These are the results of the query. Notice com.celastradepro.plist is present along with two other LaunchDaemons.

Figure 6 is the same query as Figure 5 but runs from the Osqueryi command line on the endpoint. The host returning the results is assumed so that the host field does not exist in Osqueryi.

```
osquery> SELECT name, path FROM launchd WHERE path LIKE '/Library/%';
+-----------------------------------+-----------------------------------------------------------------+
| name                              | path                                                            |
+-----------------------------------+-----------------------------------------------------------------+
| com.celastradepro.plist           | /Library/LaunchDaemons/com.celastradepro.plist                  |
| com.facebook.osqueryd.plist       | /Library/LaunchDaemons/com.facebook.osqueryd.plist              |
| com.objectiveSee.blockblock.plist | /Library/LaunchDaemons/com.objectiveSee.blockblock.plist        |
+-----------------------------------+-----------------------------------------------------------------+
```

Figure 6: osqueryi query for launch daemons

Switching to Osqueryi from Fleet to continue the investigation, the same information is available. The next query returns information about running processes associated with the CelasTradePro application.


```
osqueryi> SELECT pid, name, path FROM processes WHERE path LIKE
'/Applications/CelasTradePro.app%';
```

```
osquery> select pid, name, path from processes where path like '/Applications/CelasTradePro.app%';
+-----+---------+-----------------------------------------------------------+
| pid | name    | path                                                      |
+-----+---------+-----------------------------------------------------------+
| 436 | Updater | /Applications/CelasTradePro.app/Contents/MacOS/Updater |
+-----+---------+-----------------------------------------------------------+
```

Figure 7: osqueryi query for CelasTradePro running processes

In Figure 7, the result shows that the CelasTradePro.app has a launch daemon related to the item we found earlier at /Library/LaunchDaemons/com.celastradepro.plist:

```
<array>
    <string>/Applications/CelasTradePro.app/Contents/MacOS/Updater</string>
    <string>CheckUpdate</string>
</array>
```

Figure 8: plist showing the launch application path

Figure 8 shows a snippet from "com.celastradepro.plist" the file which references the launch daemon, which in turn is referenced in the Osquery search.

This "Updater" app attempts to reach out to www.celasllc.com which is no longer a live site, but at this point the next step in the investigation is to check network connections, associated binaries, and their ports:

```
osqueryi> SELECT processes.name, process_open_sockets.remote_address,
process_open_sockets.remote_port FROM process_open_sockets LEFT JOIN
processes ON process_open_sockets.pid = processes.pid WHERE
process_open_sockets.remote_port != 0 AND processes.name != '';
```

Figure 1 – osqueryi query for active network connections

```
osquery> select processes.name, process_open_sockets.remote_address, process_open_sockets.remote_port fro
m process_open_sockets LEFT JOIN processes ON process_open_sockets.pid = processes.pid WHERE process_open
_sockets.remote_port != 0 AND processes.name != '';
+------+----------------+-------------+
| name | remote_address | remote_port |
+------+----------------+-------------+
| sshd | 192.168.13.1   | 54769       |
| sshd | 192.168.13.1   | 54769       |
+------+----------------+-------------+
```

Figure 9: osqueryi query for active network connections

Figure 9 shows that the only current active connection is the secure shell to the malware-infected machine.

3.2.3 Catching Future Installations of AppleJeus with Kibana Visualizations

Three indicators of the AppleJeus: Lazarus application was uncovered. An application, a plist, and a running process. Osquery has uncovered no evidence that this application is malware. However, it is an unexpected installation. To ensure subsequent installations of this application are noticed more quickly, create a query pack and Kibana visualizations that look for attributes that are unique to the application.

Based on the queries that were run during the investigation, create a new query pack Called "AppleJeus" with three “snapshot” style queries should be capable of catching future installations of this application:

applejeus_application:

SELECT name, path FROM apps WHERE name LIKE 'CelasTradePro.app'

applejeus_launchd:

SELECT name, path FROM launchd WHERE path LIKE '/Library/LaunchDaemons/com.celas%'

applejeus_process:

SELECT pid, name, path FROM processes WHERE path LIKE '/Applications/Celas%'

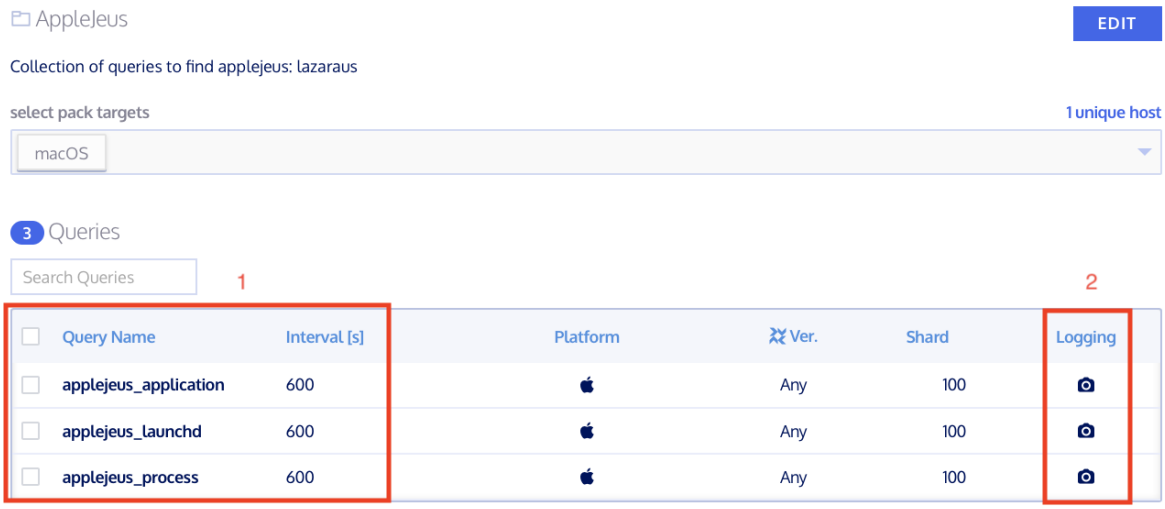

Figure 10: Fleet query pack to detect installations of AppleJeus.

- Figure 10.1: The three queries created to track AppleJeus installs are here, running at a very aggressive interval of 600 seconds, for queries like this, daily (86400) is more appropriate.
- Figure 10.2: The Logging type is a snapshot (camera) instead of differential (+). This takes more resources but ensures that the results are captured in every run of the query.

Figure 11 shows a dashboard item with the result data of the new AppleJeus query packs. The presence of an item these queries have a result other than empty, meaning the "CelasTradePro.app" is installed and running.

01_applejeuse_table

host name	action type	query name	Count
osq01-osx12	snapshot	pack/AppleJeus/applejeus_application	2
osq01-osx12	snapshot	pack/AppleJeus/applejeus_process	2
osq01-osx12	snapshot	pack/AppleJeus/applejeus_launchd	1

Figure 11: Kibana dashboard showing hosts with AppleJeus installed

To verify this dashboard is working, installing AppleJeus on a second machine should show in the visualization.

01_applejeuse_table

host name	action type	query name
osq01-osx12	snapshot	pack/AppleJeus/applejeus_application
osq01-osx12	snapshot	pack/AppleJeus/applejeus_launchd
osq01-osx12	snapshot	pack/AppleJeus/applejeus_process
osq02-osx13	snapshot	pack/AppleJeus/applejeus_application
osq02-osx13	snapshot	pack/AppleJeus/applejeus_launchd
osq02-osx13	snapshot	pack/AppleJeus/applejeus_process

Figure 12: Kibana dashboard showing the second host with AppleJeus installed

Figure 12 shows that the new installation shows up on the dashboard as expected. The final test is to remove from osq01-osx12 and see if it drops from the dashboard.

01_applejeuse_table

host name	action type	query name
osq02-osx13	snapshot	pack/AppleJeus/applejeus_application
osq02-osx13	snapshot	pack/AppleJeus/applejeus_launchd
osq02-osx13	snapshot	pack/AppleJeus/applejeus_process

Figure 13: Kibana dashboard showing AppleJeus removed from the first host

In Figure 13, osq01-osx12 drops from the dashboard after AppleJeus is removed from the endpoint showing that all future instances of AppleJeus: Lazarus will be easy to find via the dashboard.

3.2.4 Final Report: AppleJeus: Lazarus

The query packs provided by the Osquery GitHub page were able to detect the installation of a new application on the endpoint by showing the changes made to the "/Applications/" folder. The full scope of the installation was not uncovered initially, but a few additional simple queries run via Fleet and Osqueryi were able to find a launch daemon and a running process associated with this application. Finally, creating visualizations based on discovered attributes of the applications is an effective means of catching additional installations of the application. Overall, the skill level to complete this investigation is relatively low as this application is not designed to hide in the operating system but relies on looking like a legitimate application.

3.3 Experiment 2: jRAT/Adwind (Windows Version)

For the second experiment, jRAT/Adwind is installed onto a Windows virtual machine running Windows version 10. Elastic Stack logs then checked if any queries provided by the default installation from the Osquery GitHub page (Osquery, 2019) indicate that the trojan has been installed. If an indicator is found, subsequent queries will be run to find additional traits of the malware. "jRAT (also called Adwind) is a commercial cross-platform remote access Trojan that is written in Java. It is designed to control and collect data from a victim's machine regardless of whether it is running Windows, Linux, Mac OS X, or BSD. While jRAT is not very new, it keeps upgrading its technology" (Zhang, 2018).

Malware sample can be obtained from virus share's website, registration is required.: https://virusshare.com/ (VirusShare.com, 2019)
sha256 = 2098ae8c4256a354283a4e0d64175402f4c7fbf69bb6a50098703db715e7f818

3.3.1 Indicators of jRAT/Adwind Installation

After the installation of jRAT, checking the logs on Elastic Stack revealed that the Osquery query pack "windows-hardening" which runs the query "UAC_Disabled" caught a change at the time jRAT was installed. "UAC" or User Account Control is "a security feature

of Windows, which helps prevent unauthorized changes to the operating system" (Rusen, 2017). UAC should not be turned off.

The following query noticed the change to the UAC:

```
SELECT * FROM registry WHERE
path='HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA' AND data=0;
```

The registry table in Osquery needs to have particular locations to watch. Otherwise, changes to the registry are easy to miss. This high-value location, however, was being monitored and Osquery logged the change.

Table JSON

	t	@timestamp	Aug 3, 2019 @ 10:47:26.000
	t	_id	yLQqWGwBK7ive928Qrlt
	t	_index	filebeat-7.2.0-2019.08.03-000002
	#	_score	-
	t	_type	_doc
1	t	osquery.result.action	added
2	t	osquery.result.calendar_time	Sat Aug 3 15:47:26 2019 UTC
	t	osquery.result.columns.data	0
	t	osquery.result.columns.key	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System
3	t	osquery.result.columns.mtime	1506692919
	t	osquery.result.columns.name	EnableLUA
4	t	osquery.result.columns.path	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA
	t	osquery.result.columns.type	REG_DWORD
	#	osquery.result.counter	2
	t	osquery.result.decorations.host_uuid	48B34D56-C0B5-E574-AC18-A950A788365D
5	t	osquery.result.decorations.hostname	osq03-win10
	#	osquery.result.epoch	0
	t	osquery.result.host_identifier	48B34D56-C0B5-E574-AC18-A950A788365D
6	t	osquery.result.name	pack/windows-hardening/UAC_Disabled
	#	osquery.result.unix_time	1,564,847,246

Figure 14: Kibana log visualization

- Figure 14.1: osquery.result.action Shows the action taken by the query, in this case, "added," meaning this entry did not exist on the previous run.
- Figure 14.2: osquery.result.calendar_time shows the time that this query result was reported.
- Figure 14.3: osquery.result.column.mtime shows the timestamp of the registry write.
- Figure 14.4: osquery.result.columns.path shows the registry key where the change was made.
- Figure 14.5 osquery.result.decorations.hostname is the name of the endpoint, which recorded this result.

- Figure 14.6 osquery.result.name shows the Fleet "pack" and "query" that was run from the Fleet server, which produced this result. If you want more details on the exact query, you can reference back to Fleet server or find the query pack run entry in Kibana which also shows the SQL of the query pack.

Very little is known about this change. From the logs on Elastic Stack, there are no additional entries which would indicate that an application has been installed or why the UAC is now disabled. This type of change to an endpoint's security posture makes for a worthwhile investigation.

3.3.2 Using Fleet or Osqueryi to Start an Investigation

Having only the change to the UAC to start an investigation, broad strokes are needed to find further evidence of what caused the change to the UAC. Checking startup locations with Osquery is easy.

```
SELECT name, path FROM startup;
```

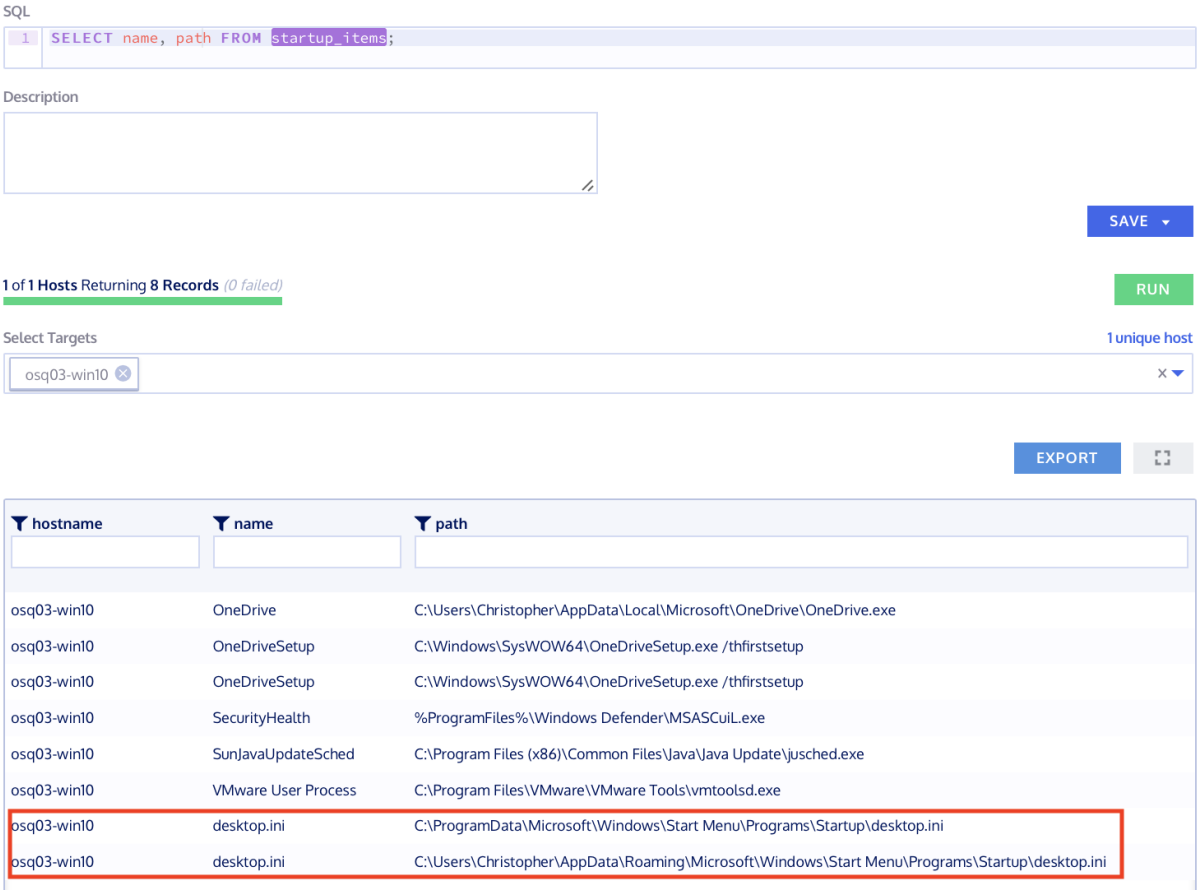

Figure 15: Fleet query for startup items

The results in Figure 15 reveal two new startup items. These two changes warrant further investigation. Pulling the endpoint from service for further study would be warranted

due solely to the changes to the UAC, adding changes to startup is a noticeable attribute of unwanted application installation.

The query "pack/windows-hardening/UAC_Disabled" runs on a 600-second schedule. The query logged the event on August 3 at 15:47:26 UTC. Using the epoch timestamp for that date is “1564847246”. To find out if the UAC_Disabled event from Elastic Search correlates to the unexpected startup items:

```
SELECT btime, path FROM file WHERE path LIKE
'C:\Users\Christopher\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\desktop.ini';
```

hostname	btime	path
osq03-win10	1564839017	C:\Users\Christopher\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\desktop.ini

Figure 16: Fleet query for “btime” of new startup entries

The resulting "btime" or file birth time is epoch 1564839017 compared to the UAC_Disable event at epoch 1564847246 indicates roughly two hours and fifteen minutes between events. The startup items appear to follow the UAC_Disable event. Since the discovered startup entries exist in the user’s folder, and the time of change is known, a reasonable next step is looking at what other files were created around the time of the UAC_Disable event and the startup item creation event.

```
SELECT filename, path, btime FROM file WHERE path LIKE
'C:\Users\Christopher\%%' AND btime LIKE '15648470%';
```

osq03-win10	1564847050	local_policy.jar		C:\Users\Christopher\AppData\Roaming\Oracle\lib\security\policy\limited\local_policy.jar
osq03-win10	1564847050	.		C:\Users\Christopher\AppData\Roaming\Oracle\lib\security\policy\unlimited\
osq03-win10	1564847050	US_export_policy.jar		C:\Users\Christopher\AppData\Roaming\Oracle\lib\security\policy\unlimited\US_export_policy.jar
osq03-win10	1564847050	local_policy.jar		C:\Users\Christopher\AppData\Roaming\Oracle\lib\security\policy\unlimited\local_policy.jar
osq03-win10	1564847050	trusted.libraries	1	C:\Users\Christopher\AppData\Roaming\Oracle\lib\security\trusted.libraries
osq03-win10	1564847049	sound.properties		C:\Users\Christopher\AppData\Roaming\Oracle\lib\sound.properties
osq03-win10	1564847049	tzdb.dat		C:\Users\Christopher\AppData\Roaming\Oracle\lib\tzdb.dat
osq03-win10	1564847049	tzmappings		C:\Users\Christopher\AppData\Roaming\Oracle\lib\tzmappings
osq03-win10	1564847049	release		C:\Users\Christopher\AppData\Roaming\Oracle\release
osq03-win10	1564847026	.	2	C:\Users\Christopher\Desktop\VirusShare_75b86ff8bd2e615eda7ad052e9c99cc2\
osq03-win10	1564847049	.		C:\Users\Christopher\YLMhgyupQoN\
osq03-win10	1564847049	ID.txt		C:\Users\Christopher\YLMhgyupQoN\ID.txt
osq03-win10	1564847049	.		C:\Users\Christopher\YLMhgyupQoN\KAbBQnroDYp\
osq03-win10	1564847050	.	3	C:\Users\Christopher\fUTkALeaTxM\
osq03-win10	1564847050	.		C:\Users\Christopher\fUTkALeaTxM\DdWDtpinxpf\
osq03-win10	1564847050	ID.txt		C:\Users\Christopher\fUTkALeaTxM\ID.txt

Figure 17: Fleet query result for files with “btime” like 15648470%

- Figure 17 is a clipping of the 273 results from the query, most of which is the Java version being copied into the user folder — all created within a few seconds of each other.
- Figure 17.1 This is the majority of the entries, most of which are the copying of Java to the user folder.
- Figure 17.2 This is the extracted malware sample folder.
- Figure 17.3 These are very suspect and worth additional investigation, though little more can be done about these from Osquery.

Breadcrumbs leading to the registry or "C:\Windows\System32\" directory were not uncovered. Recursive searches such as:

```
SELECT path, name, type, data FROM registry WHERE path LIKE
'HKEY_LOCAL_MACHINE\%\%%' AND btime LIKE '15648470%'
```

Looking for changes in the registry at a similar mtime to the recorded file changes did not adequately show any useful results showing that the ability for Osquery to search recursively through the registry for timed changes is limited since changes were made, but the results are not reported.

3.3.3 Catching Future Installations of jRAT/Adwind with New Queries

Now that indicators beyond just the UAC_Disabled query have been uncovered creating visualizations and additional queries to help with discovery are vital to ensure future instances of this malware are quickly caught.

One of the most important new queries to run is something to monitor startup locations on Windows machines. Curiously, this query exists for MacOS systems in the default packs, but not for Windows systems.

windows_startup_items:

```
SELECT name, path FROM startup;
```

windows_registry_user_startup_items:

```
SELECT * FROM registry WHERE path LIKE
'HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\%';
```

windows_registry_machine_startup_items:

```
SELECT * FROM registry WHERE path LIKE
'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
\%';
```


3.3.4 Final Report: jRAT/Adwind

The query packs provided by the Osquery GitHub page were able to detect the installation of a new application on the endpoint by recording changes the User Account Control (UAC) state changing from on to off. However, to uncover additional attributes of the malware, several advanced concepts and queries were required. The Fleet server was a useful tool for executing additional queries beyond what was included in the query packs. Without prior knowledge of this being malware, the alert raised by this tool would likely kick off an investigation due to the blatant User Account Control change. The lessons learned from this malware required the creation of improved query packs. New queries that watch startup items are likely to be important indicators for future investigations. The skill level to catch this application is with Osquery, Fleet, and Elastic Search is low. However, the investigation requires a nuanced understanding of the Windows operating system, file systems, the Windows Registry, and epoch times.

4. Recommendations and Implications

The research indicates that Osquery, Fleet, and Elastic Stack are successful at catching some fundamental changes to an endpoint using only those queries included in the GitHub page (Osquery, 2019). Furthermore, Fleet and Osquery are capable of uncovering a several additional changes made to an endpoint after unexpected software installation. Finally, the log aggregation features of Elastic Stack make the timeline and results reported by Osquery and Fleet easy to trace and track. An organization with skilled professionals looking to add EDR to their existing portfolio should consider this solution as an option.

4.1 Incident Detection Recommendations

The ability of Osquery, Fleet, and Elastic Stack to detect and log changes to monitored systems is evident. However, the query packs included in the GitHub repository did not perform to the full degree that this platform is capable of doing. Further research and documentation are needed to expand on these queries. Carbon Black is currently using ATT&CK with Osquery to help define standards for Osquery detection (Tympanick, 2019). Translating ATT&CK into Fleet and Osquery could become a remarkable tool for catching and dealing with many types of malware quickly.

4.2 Investigation Capabilities and Recommendations

The power of Osquery in the hands of an experienced investigator is pretty remarkable. AT&T Cybersecurity (Formerly AlienVault) hosts many blog posts on using the "AlienVault Agent," which leverages Osquery "to enable threat hunting" (Ruiz, 2019). These and many other blogs form a growing community of users interested in expanding the conversation about how to use this open-source product.

5. Conclusion

The difficulty of implementing and using this solution is relatively low. Most system administrators or security analysts are capable of benefiting from this solution. Osquery with the GitHub query packs is effective at finding initial indicators of unwanted software on endpoints and could be used as a starting point for investigations.

IT professionals of moderate ability will be capable of using Fleet and Osquery as an investigation medium for discovering unwanted software.

Lessons learned from investigations can be used to enhance the system for detecting future indicators through the creation of additional queries that forward to the Elastic Stack for monitoring.

Log events presented in Kibana, the visualization front end of Elastic Stack, are well organized. Finding meaningful data or creating visualizations of essential events is straightforward and requires very little time or experience with the user interface.

5.1 Recommendations

Organizations that need to enhance detection and investigation capabilities would find using Osquery, Fleet, and Elastic Stack to be a practical EDR solution, provided the organization's security processes are mature enough to handle response and containment after discovery. Organizations that consider their security processes ready for EDR are most likely capable of implementing these products and would find an immediate ability to detect and investigate with very little initial training using the underlying technologies. Advanced understanding of operating systems and SQL would significantly enhance the benefits but are not required. A sufficient amount of documentation and examples exist and less experienced professionals are capable of finding and building the queries they need based on existing documented cases. Based on similar research articles by AT&T Cybersecurity (formerly AlienVault) (Ruiz, 2019) and Carbon Black (Tympanick, 2019), OSQuery, Fleet, and Elastic Stack provide similar features and use cases to these commercial applications.

References

AlienVault. (2019). *The AlienVault Agent*. Retrieved from AlienVault: https://www.alienvault.com/documentation/usm-anywhere/deployment-guide/data-source/alienvault-agents.htm

Arpaia, M. (2017, October 18). *Kolide Fleet: An open-source osquery fleet manager.* Retrieved from blog.kolide.com: https://blog.kolide.com/kolide-fleet-an-open-source-osquery-fleet-manager-26e8094fab

Carbon Black, Inc. (2019). *Carbon Black Introduces Cb LiveOps for Real-Time Query and Response, Surpassing Tanium and CrowdStrike With Its Complete, Cloud-Delivered Security Platform*. Retrieved from Globe News Wire: https://www.globenewswire.com/news-release/2018/08/02/1546373/0/en/Carbon-Black-Introduces-Cb-LiveOps-for-Real-Time-Query-and-Response-Surpassing-Tanium-and-CrowdStrike-With-Its-Complete-Cloud-Delivered-Security-Platform.html

Digital Ocean. (2019, July 31). *How To Install Elasticsearch, Logstash, and Kibana (Elastic Stack) on CentOS 7*. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-centos-7

Facebook. (2019). Retrieved from Welcome to osquery: https://osquery.readthedocs.io/en/stable/

Facebook. (2019). *OSQeury Schema*. Retrieved from osquery.io: https://osquery.io/schema/3.3.2#launchd

Facebook. (2019). *osquery*. Retrieved from github: Introduction to osquery for Threat Detection and DFIR

Firstbrook, P. (2018, November 26). *Market Guide for Endpoint Detection and Response Solutions*. Retrieved from Gartner: https://www.gartner.com/document/3894086?ref=solrAll&refval=223298784&qid=2b4fc913bedaeeaa9382c7284a6

Gartner. (2019, July 05). *Reviews for Endpoint Detection and Response Solutions*. Retrieved from Gartner: https://www.gartner.com/reviews/market/endpoint-detection-and-response-solutions

GReAT. (2018, August 23). *Operation AppleJeus: Lazarus hits cryptocurrency exchange with fake installer and macOS malware*. Retrieved from securelist: https://securelist.com/operation-applejeus/87553/

Kolide. (2019, August 01). *Kolide Fleet Infrastructure Docs*. Retrieved from github: https://github.com/kolide/fleet/tree/master/docs/infrastructure

Kolide. (2019, August 01). *Kolide Launcher*. Retrieved from github: https://github.com/kolide/launcher

Kolide. (2019). *Kolide/launcher*. Retrieved from github: https://github.com/kolide/launcher

Malik, A. (2019). *Threat Hunting with Osquery: macOS Malware Techniques & How to Find Them*. Retrieved from uptycs: https://www.uptycs.com/blog/macos-malware-threat-hunting

McCammon, K. (2018, July 18). *Build vs. Buy: Not Mutually Exclusive*. Retrieved from red canary: https://redcanary.com/blog/build-vs-buy-not-mutually-exclusive/

objective see. (2019, July 04). *Mac Malware*. Retrieved from objective-see.com: https://objective-see.com/malware.html

Objective-See. (2019, June 26). *Mac Malware*. Retrieved from objective-see: https://objective-see.com/malware.html

Osquery. (2019, July 29). Retrieved from https://github.com/osquery/osquery

Potti, J. (2018, February 16). *Elk + Osquery + Kolide Fleet = Love*. Retrieved from jordanpotti.com: https://jordanpotti.com/2018/02/16/elk-osquery-kolide-fleet-love/

Rapid7. (2016, May 09). *Introduction to osquery for Threat Detection and DFIR*. Retrieved from rapid7 blog: https://blog.rapid7.com/2016/05/09/introduction-to-osquery-for-threat-detection-dfir/

Ruiz, J. (2019, August 07). *Malware Analysis using Osquery Part 1*. Retrieved from AT&T Cybersecurity: https://www.alienvault.com/blogs/labs-research/malware-analysis-using-osquery-part-1

Rusen, C. A. (2017, 07 11). *What is UAC (User Account Control) and why you should never turn it off*. Retrieved from Digital Citizen: https://www.digitalcitizen.life/uac-why-you-should-never-turn-it-off

Tympanick, S. (2019, 10 28). *ATT&CK +osquery = Love*. Retrieved from carbonblack.com: https://www.carbonblack.com/2018/10/29/attck-osquery-love/

VirusShare.com. (2019, August 02). *VirusShare.com*. Retrieved from VirusShare.com: https://virusshare.com/

vmray. (2018, march 12). *vmray*. Retrieved from vmray analyzer: https://www.vmray.com/analyses/2098ae8c4256/report/ioc.html

Zeltser, L. (2019). *Free Malware Sample Sources for Researchers*. Retrieved from Lenny Zeltser: https://zeltser.com/malware-sample-sources/

Zhang, X. (2018, February 16). *New jRAT/Adwind Variant Being Spread With Package Delivery Scam*. Retrieved from Fortinet Blog: https://www.fortinet.com/blog/threat-research/new-jrat-adwind-variant-being-spread-with-package-delivery-scam.html

Appendix

Build Process

CentOS 7.6
Install Dependencies

```
$ sudo yum update
$ sudo yum install unzip
$ sudo yum install java-1.8.0-openjdk
$ sudo yum install epel-release
$ sudo yum install nginx
$ sudo systemctl start nginx
```

Install Elasticsearch, Logstash, and Kibana
https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-centos-7

Elasticsearch

```
$ sudo rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
$ sudo vi /etc/yum.repos.d/elasticsearch.repo
$ sudo yum install elasticsearch
$ sudo vi /etc/elasticsearch/elasticsearch.yml
$ sudo systemctl start elasticsearch
$ sudo systemctl enable elasticsearch
$ curl -X GET "localhost:9200"
```

```
{
  "name" : "DX2KuVz",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Mscq8fVcR5-xgxFB3l35lg",
  "version" : {
    "number" : "6.5.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "816e6f6",
    "build_date" : "2018-11-09T18:58:36.352602Z",
    "build_snapshot" : false,
    "lucene_version" : "7.5.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Kibana Dashboard

```
$ sudo yum install kibana
$ sudo systemctl enable kibana
$ sudo systemctl start kibana
$ echo "kibanaadmin:`openssl passwd -apr1`" | sudo tee -a
/etc/nginx/htpasswd.users
$ sudo vi /etc/nginx/conf.d/192.168.13.128.conf$ sudo vim
/etc/nginx/sites-available/192.168.13.128
```

```
      server {
          listen 80;
          server_name 192.168.13.128;
          auth_basic "Restricted Access";
          auth_basic_user_file /etc/nginx/htpasswd.users;
          location / {
              proxy_pass http://localhost:5601;
              proxy_http_version 1.1;
              proxy_set_header Upgrade $http_upgrade;
              proxy_set_header Connection 'upgrade';
              proxy_set_header Host $host;
              proxy_cache_bypass $http_upgrade;
          }
      }

$ sudo nginx -t
$ sudo systemctl restart nginx
$ sudo setsebool httpd_can_network_connect 1 -P
In a web browser to go http://192.168.13.128/status
```

Logstash

```
$ sudo yum install logstash
$ sudo vim /etc/logstash/conf.d/02-beats-input.conf

      input {
        beats {
          port => 5044
        }
      }

$ sudo vim /etc/logstash/conf.d/30-elasticsearch-output.conf
      output {
        elasticsearch {
          hosts => ["localhost:9200"]
          manage_template => false
          index => "%{[@metadata][beat]}-%{[@metadata][version]}-
      %{+YYYY.MM.dd}"
        }
      }
$ sudo -u logstash /usr/share/logstash/bin/logstash --path.settings
/etc/logstash -t
$ sudo systemctl start logstash
$ sudo systemctl enable logstash
```

Filebeat

```
$ sudo yum install filebeat
$ sudo filebeat modules enable osquery
$ sudo vim /etc/filebeat/modules.d/osquery.yml
> var.paths: ["/var/log/osquery/*.log"]

$ sudo filebeat setup
$ sudo systemctl start filebeat
$ sudo systemctl enable filebeat
```

Fleet

```
https://github.com/kolide/fleet/blob/master/docs/infrastructure/fleet-on-
centos.md
$ sudo yum install wget
```

```
$ wget https://github.com/kolide/fleet/releases/latest/download/fleet.zip
$ sudo apt install unzip
$ unzip fleet.zip 'linux/*' -d fleet
$ sudo cp fleet/linux/fleet /usr/bin/fleet
$ sudo cp fleet/linux/fleetctl /usr/bin/fleetctl
$ sudo rpm -i mysql57-community-release-el7.rpm
$ sudo yum update
$ sudo yum install mysql-server
$ sudo systemctl start mysqld
$ sudo cat /var/log/mysqld.log
-- grep for 2019-06-30T05:52:41.219581Z 1 [Note] A temporary password is
generated for root@localhost: rEv?+0fIFdyd
$ mysql -u root -p
      mysql> ALTER USER "root"@"localhost" IDENTIFIED BY "toor?Fl33t";
      mysql> flush privileges;
      mysql> exit
$ sudo yum install redis
$ sudo service redis start
$ /usr/bin/fleet prepare db --mysql_address=127.0.0.1:3306 --
mysql_database=kolide --mysql_username=root --mysql_password=toor
Migrations completed.

$ sudo mkdir -p /etc/pki/tls/certs
$ sudo mkdir -p /etc/pki/tls/private
$ sudo mkdir /var/log/osquery
$ sudo openssl genrsa -out /etc/pki/tls/private/server.key 4096
$ sudo openssl req -new -key /etc/pki/tls/private/server.key -out
/etc/pki/tls/certs/server.csr
$ sudo openssl x509 -req -days 366 -in /etc/pki/tls/certs/server.csr -
signkey /etc/pki/tls/private/server.key -out
/etc/pki/tls/certs/server.cert

Verify this next command runs, then create fleet service
$ sudo /usr/bin/fleet serve --mysql_address=127.0.0.1:3306 --
mysql_database=kolide --mysql_username=root --mysql_password=toor --
redis_address=127.0.0.1:6379 --server_cert=/etc/pki/tls/certs/server.cert
--server_key=/etc/pki/tls/private/server.key --logging_json --
osquery_result_log_file=/var/log/osquery/osqueryd.result.log --
osquery_status_log_file=/var/log/osquery/osqueryd.status.log --
auth_jwt_key=supersecretkey
```

Fleet Service
https://github.com/kolide/fleet/blob/master/docs/infrastructure/systemd.md

```
$ sudo vim /etc/systemd/system/fleet.service

      [Unit]
      Description=Fleet
      After=network.target

      [Service]
      LimitNOFILE=8096
      ExecStart=/usr/bin/fleet serve \
        --mysql_address=127.0.0.1:3306 \
        --mysql_database=kolide \
        --mysql_username=root \
        --mysql_password=toor \
        --redis_address=127.0.0.1:6379 \
        --server_cert=/etc/pki/tls/certs/server.cert \
```

```
        --server_key=/etc/pki/tls/private/server.key \
        --osquery_result_log_file=/var/log/osquery/osqueryd.result.log \
        --osquery_status_log_file=/var/log/osquery/osqueryd.status.log \
        --auth_jwt_key=supersecretkey \
        --logging_json

      [Install]
      WantedBy=multi-user.target

$ sudo systemctl enable fleet.service
$ sudo systemctl start fleet.service

Useful commands after editing the .service file:
$ sudo systemctl daemon-reload
$ sudo systemctl restart fleet.service


https://192.168.13.128:8080/setup

From Add Computer Page:
OSQuery Enroll Secret:
enroll_secret
UPDFjN8+aOytMVmooOfJ87RrKM4+l0Wr

Fleet server certificate:
fleet_cert.pem
-----BEGIN CERTIFICATE-----
MIIFBjCCAu4CCQClDTcGPQRu3TANBgkqhkiG9w0BAQsFADBFMQswCQYDVQQGEwJB
VTETMBEGA1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50ZXJuZXQgV2lkZ2l0
cyBQdHkgTHRkMB4XDTE5MDYxMzEzMTg0NloXDTIwMDYxMzEzMTg0NlowRTELMAkG
A1UEBhMCQVUxEzARBgNVBAgMClNvbWUtU3RhdGUxITAfBgNVBAoMGEludGVybmV0
IFdpZGdpdHMgUHR5IEx0ZDCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIB
ALHb+RYdg0keBARy6/0t1EQsHzGZTI3IqyE/KGbXCnMTkASziwSCPyfQVJCm4eY4
OIjDvTGz6HyTH7va2AJUcsec6a6FOLmNSR+FjIs4k7k3MvWtPNUm0pdwlvcW1TDP
RgAX9kgUvzPyNrnUgq2YeRqG9Pvn4RrYpiBIpYNa5I+jLzQgtsmZavlBGLWwFtkk
bFcQ4XPo7O/nxNkW0ob2cMSMEM+ICakayQApJQH8KH2nGGX2uQ6dZuSHZw6qfFT3
AZMAdwbFfQwCDNtmTmeOmJIDW9SLITh9AE+PKhIXO4a4O0DvD9DEg3JwN49aLKC/
hor0sWOLkm2EuxgcOdPu3JdD0TUny+d9/7jPDGNl98i2gB+qoLnLzDLlpHSPpeAy
5W/q9tOPMLSYhaaO2UYLLi7TO2CY/jShc97GiNthC/N5ZUlpZie8E4/dFe+d+tmX
yy0jkdlJun/kgFSu2Asl2ZMfOvtu8jYNfQlYI8cdH/e9lrJEZM7kHZDO7Qf2yQ0y
U7DPJmwUvnxNurLIF8YaOdvuwKwJkWeqN1dla00R6u3v7kIcs1GZdTvlaf4I1mvB
RzZflNpid1hrI/1L3HGU/+6MOKBB9AZHUpK1p/rdWk/qHN+7D0fcV4kgHtW0by+m
GG+8k5761MMvewZxlnK95jDlSliWKWrEwIDXSyW7ABQ7AgMBAAEwDQYJKoZIhvcN
AQELBQADggIBAELkrmSwovq9uI/5hBHqBsxlN3vUDTweUFi9sjoB9bsnA4+dvSEF
vGoxTYdPye5SKz2pEP9K2FBQ0qbXhFvHmu6AVeuNjsEIDw7NhJpxKFxww6FW14ky
oDOmsEhznsSzfmXclkmzqzN2FCZF9LAvO1swQ+/K/4S0rgHVSACQDyR+lmu0R5Ls
Beh7lSB8+tDY4tmnsrJZ4Ub60kp8Bg1gCFwM6pL76sT95PjTeK93Bk/lZTszUwfY
sCOn/oD3xDimulVRo6EMdtMKs0Aj0ZxyieRis7F/HWep/3//iptWt//sQeYnS8O+
6WcMUEiqo5uLXFeszM66logknPKO3GR0vC5vLiKl0a9K/7H5IuZUYFxPwjHqdcQ+
Mv2QL9kkW09u70OiE6gYXW2BF8+4fBJZOvgcuBoOTYxNi+TucERCPRMkpCmhJBwa
H6+fLIQNgJET0qwUGi3iLLXD5cU+FLmtmQid+LyldWWMFlA+zZtklknTE6Ytki/A
qBg+VK/olb0JEcZ7YFdZsAUQxPhtGfQsFfXJ/6Vq+cDwZsXuKZv1Z7FwqnwVpTft
WfxpQPFbF0NkOtuOYqUfiply0Tbg3/miFOy+wnfoytsrb08LJemOcwb0B2plp/J4
UasBFzH1fKQ0m5vctaUcUYI4auEAt7Im5zKowleYnYxKDSTaz/rakGuS
-----END CERTIFICATE-----
```

Fleetctl setup
https://github.com/kolide/fleet/blob/master/docs/cli/setup-guide.md

```
$ fleetctl config set --address https://192.168.13.128:8080 -tls-skip-
verify
$ fleetctl login
email:password

**Importing and converting Query Packs**
https://github.com/facebook/osquery
https://gist.github.com/marpaia/9e061f81fa60b2825f4b6bb8e0cd2c77

$ git clone https://github.com/facebook/osquery.git

for each pack you want to import to fleet do:
$ fleetctl convert -f *.conf >> *.yaml
$ fleetctl apply -f ./*.yaml
$ cd ~/osquery/packs/

$ vim install packs.sh
fleetctl convert -f hardware-monitoring.conf >> hardware-monitoring.yaml
fleetctl convert -f incident-response.conf >> incident-response.yaml
fleetctl convert -f it-compliance.conf >> it-compliance.yaml
fleetctl convert -f osquery-monitoring.conf >> osquery-monitoring.yaml
fleetctl convert -f ossec-rootkit.conf >> ossec-rootkit.yaml
fleetctl convert -f osx-attacks.conf >> osx-attacks.yaml
fleetctl convert -f unwanted-chrome-extensions.conf >> unwanted-chrome-
extensions.yaml
fleetctl convert -f vuln-management.conf >> vuln-management.yaml
fleetctl convert -f windows-attacks.conf >> windows-attacks.yaml
fleetctl convert -f windows-hardening.conf >> windows-hardening.yaml

fleetctl apply -f ./hardware-monitoring.yaml
fleetctl apply -f ./incident-response.yaml
fleetctl apply -f ./it-compliance.yaml
fleetctl apply -f ./osquery-monitoring.yaml
fleetctl apply -f ./ossec-rootkit.yaml
fleetctl apply -f ./osx-attacks.yaml
fleetctl apply -f ./unwanted-chrome-extensions.yaml
fleetctl apply -f ./vuln-management.yaml
fleetctl apply -f ./windows-attacks.yaml
fleetctl apply -f ./windows-hardening.yaml

Osquery
https://osquery.readthedocs.io/en/stable/installation/

Osquery Ubuntu
$ export OSQUERY_KEY=1484120AC4E9F8A1A577AEEE97A80C63C9D8B80B
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
$OSQUERY_KEY
$ sudo add-apt-repository 'deb [arch=amd64] https://pkg.osquery.io/deb deb
main'
$ sudo apt-get update
$ sudo apt-get install osquery

Ubuntu Osquery configuration
$ sudo vim /var/osquery/enroll_secret
from above

$ sudo vim /var/osquery/fleet_cert.pem
$ sudo /usr/bin/osqueryd --enroll_secret_path=/var/osquery/enroll_secret -
-tls_server_certs=/var/osquery/fleet_cert.pem --
```



```
tls_hostname=192.168.13.128:8080 --host_identifier=elastic --
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=3 --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
```

Osquery Windows
https://osquery.io/downloads/official/3.3.2

```
download the windows osquery-3.3.2.msi
double click to install
with an admin boosted notepad create
C:\ProgramData\osquery\certs\enroll_secret
<insert enroll secret>
C:\ProgramData\osquery\certs\fleet_cert.pem
<insert certificate block>

C:\ProgramData\osquery\osquery.flags
      --enroll_secret_path=C:\ProgramData\osquery\certs\enroll_secret.txt
      --tls_server_certs=C:\ProgramData\osquery\certs\fleet_cert.pem
      --tls_hostname=192.168.13.128:8080
      --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

restart the Osquery service.

If the host does not appear in fleet, use the following to troubleshoot.

With admin boosted cmd prompt:
c:\programdata\osquey\osqueryd\osqueryd.exe --
flagfile=c:\ProgramData\osquery\osquery.flags --verbose --tls_dump
```

Osquery Mac

```
https://osquery.io/downloads/official/3.3.2
$ sudo vim /private/var/osquery/certs/enroll_secret
$ sudo vim /private/var/osquery/certs/fleet_cert.pem
$ sudo vim /var/osquery/osquery.flags

      --enroll_secret_path=/private/var/osquery/certs/enroll_secret
      --tls_server_certs=/private/var/osquery/certs/fleet_cert.pem
      --tls_hostname=192.168.13.128:8080
      --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

$ sudo rm /var/osquery/osquery.example.conf
$ sudo cp /var/osquery/com.facebook.osqueryd.plist /Library/LaunchDaemons/
$ sudo launchctl load /Library/LaunchDaemons/com.facebook.osqueryd.plist

for troubleshooting:
$ sudo /usr/local/bin/osqueryd --flagfile /var/osquery/osquery.flags

----------End of Instructions---------
```