RSAConference2019
San Francisco | March 4–8 | Moscone Center

BETTER.

SESSION ID: TECH-F01

SOC Automation, Enterprise Blueprinting and Hunting Using Open-Source Tools

John Holowczak
Sr. Threat Analyst, Threat Analysis Unit
Carbon Black
@Skipwich

Brian Baskin
Sr. Threat Researcher, Threat Analysis Unit
Carbon Black
@bbaskin

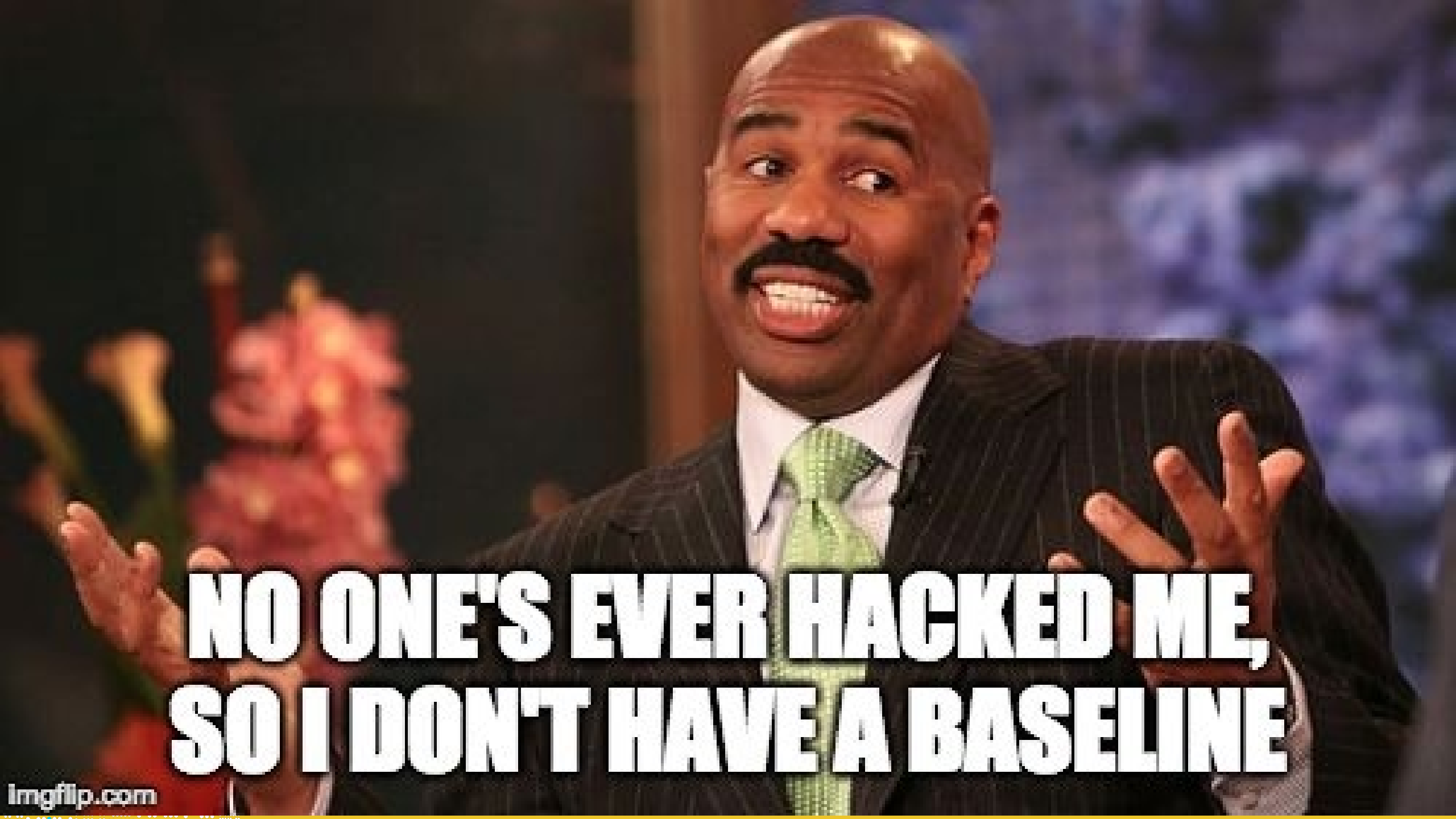
Agenda

- What and Why – Baselining
- Exercises in Blueprinting your Organization
- Automate the SOC
- Threat Hunting

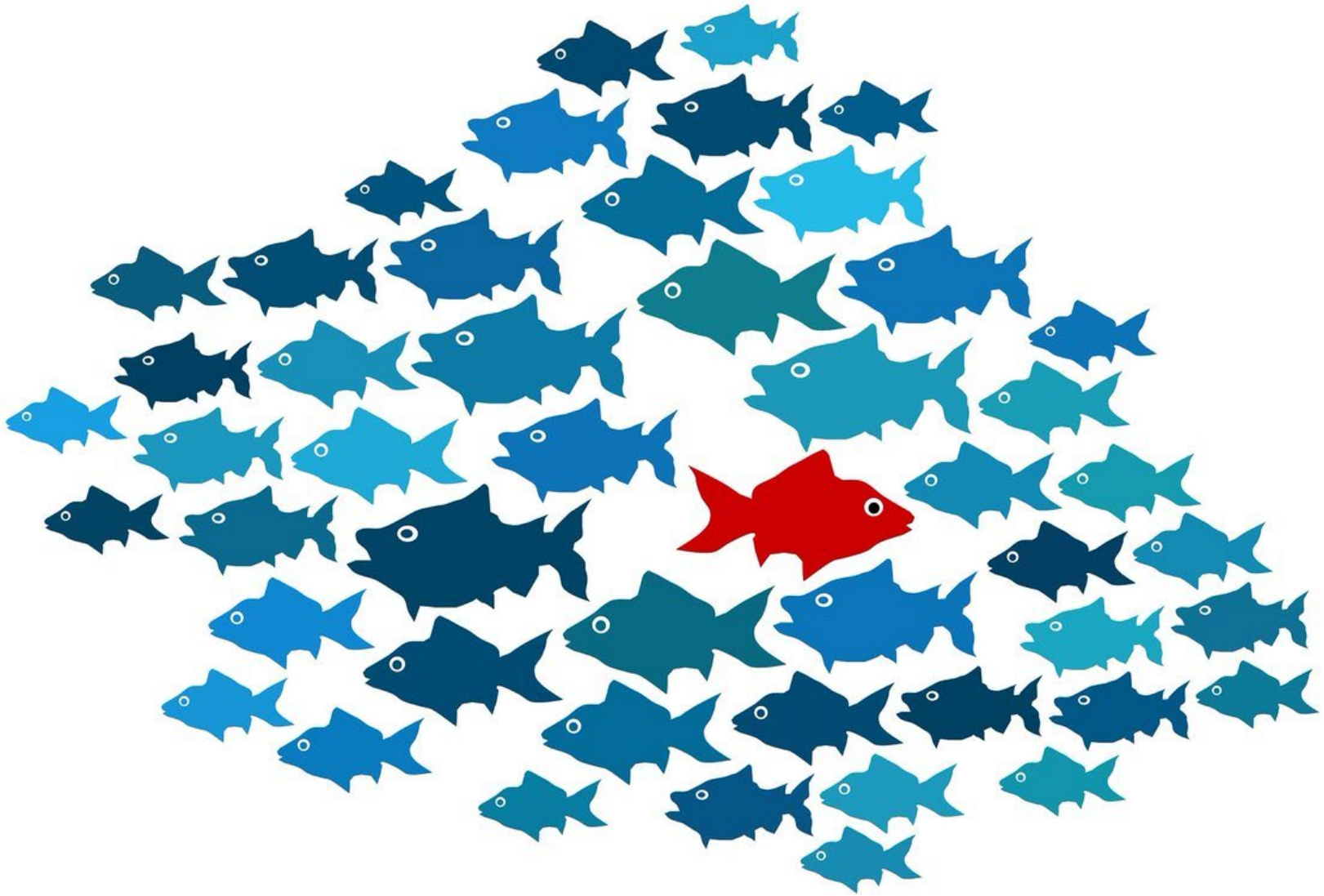
RSAConference2019

Baselining and Blueprinting

“No one’s ever hacked me, so I don’t have a baseline”


NO ONE'S EVER HACKED ME,
SO I DON'T HAVE A BASELINE
imgflip.com

Know Your Environment



Know Your Environment

- Turn over every stone; even normal behavior may be abnormal in reality
- When processing data, *classify* normal behavior and abnormal behavior
  - Certain behaviors can have multiple classifications
- Start your classification buckets large, add detail after each pass

“Blueprinting” Methods

Reactive

- Firehose
- Ingest all data into a SIEM
- Tune False Positives Forever

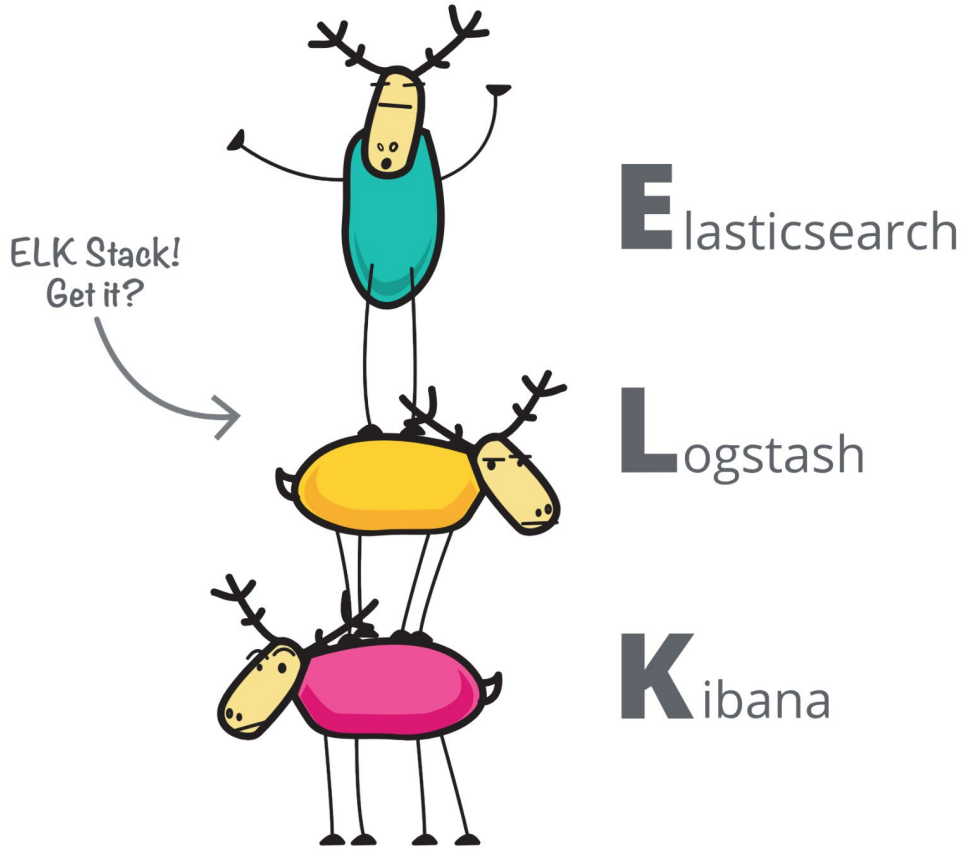
Proactive

- Blueprint First
- Create rules to find abnormal behavior
- Spend less time fighting False Positives and SIEM fires
- Later: Match against findings from threat hunts

Tools and Procedures

splunk>

sumologic



RSAConference2019

Enterprise Blueprinting

Intro to OSQuery

- Open source tool for querying endpoint metadata (at scale) like a database
- Utilizes SQL to expose data via a common interface
- Extensible in a number of languages
  - Add your own query-able data types

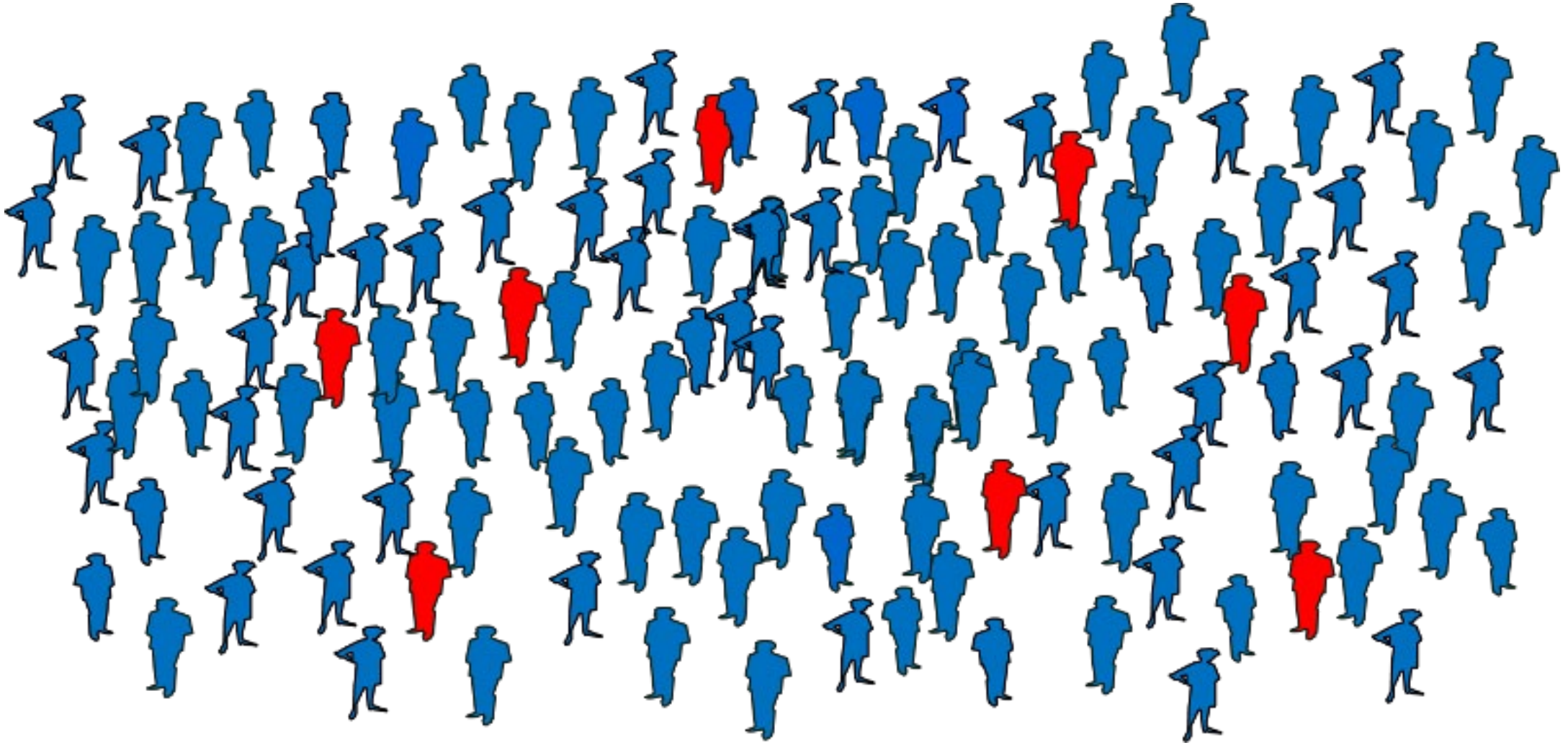
Pros/Cons

- Pros
  - Easy to get data from a number of endpoints at scale
  - Quickly query data using a common language (SQL)
  - Exhaustive list of metadata that is continually growing
- Cons
  - May be difficult to deploy across entire environment
    - Common orchestration tools can help with this (Ansible, Puppet, Chef)!

Further Information

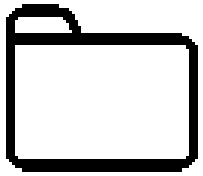
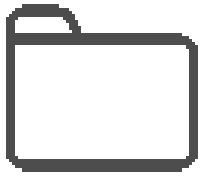
- List of schema available at https://osquery.io/schema/3.3.0 (latest version)
- Some schema offer event information such as *process_file_events* which includes timestamps with when an event took place
  - Can only get this info if running OSquery in daemon mode, as it is an *evented table*
- Other file information schema:
  - Signature information
  - Startup items
  - Scheduled tasks

Low Prevalence Executables



Low Prevalence Executables

- One-offs or rare applications
- Care less about the most common running programs
- Classify normal and abnormal for rarities to job functions

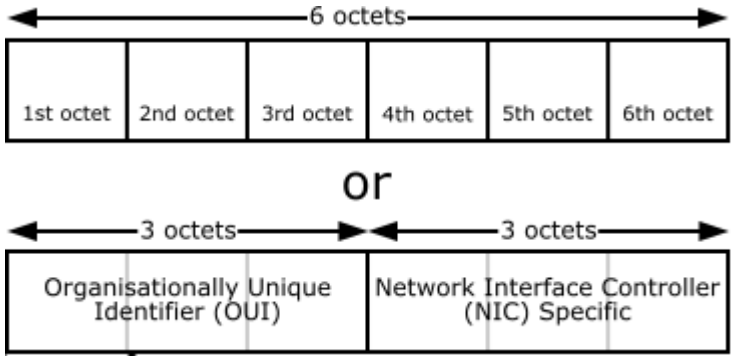




Leveraging OSQuery

```
osquery> SELECT name, pid, path, start_time FROM processes;
+----------------+-------+----------------------------------------------------------------------------+------------+
| name           | pid   | path                                                                       | start_time |
+----------------+-------+----------------------------------------------------------------------------+------------+
| systemd        | 1     | /usr/lib/systemd/systemd                                                   | 0          |
| rcu_sched      | 10    |                                                                            | 0          |
| migration/18   | 100   |                                                                            | 0          |
| udisksd        | 1002  | /usr/libexec/udisks2/udisksd                                               | 15         |
| ksoftirqd/18   | 101   |                                                                            | 0          |
| systemd-logind | 1024  | /usr/lib/systemd/systemd-logind                                            | 16         |
| gssproxy       | 1025  | /usr/sbin/gssproxy                                                         | 16         |
| irqbalance     | 1026  | /usr/sbin/irqbalance                                                       | 16         |
| smartd         | 1028  | /usr/sbin/smartd                                                           | 16         |
| kworker/18:0H  | 103   |                                                                            | 0          |
| lsmd           | 1032  | /usr/bin/lsmd                                                              | 16         |
| watchdog/19    | 104   |                                                                            | 0          |
| alsactl        | 1040  | /usr/sbin/alsactl                                                          | 16         |
| migration/19   | 105   |                                                                            | 0          |
| kworker/23:2   | 10555 |                                                                            | 17627181   |
| ksoftirqd/19   | 106   |                                                                            | 0          |
| mcelog         | 1068  | /usr/sbin/mcelog                                                           | 17         |
| kworker/19:0H  | 108   |                                                                            | 0          |
| kworker/16:2   | 10880 |                                                                            | 17627184   |
| watchdog/20    | 109   |                                                                            | 0          |
```

Networking Data

- SNMP (or equivalent) to pull data from your networking devices
- ARP Tables are a great start for network data collection
- Acquire IP and MAC addresses easily
- MAC Addresses are a great way to identify vendors on your network



Some common OUIs

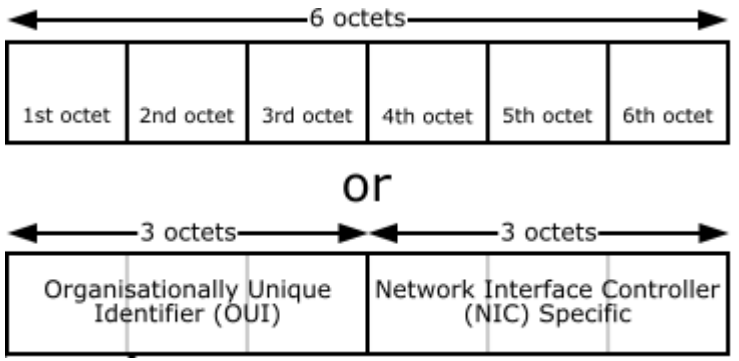
- https://www.wireshark.org/tools/oui-lookup.html

00:05:69 Vmware VMware, Inc.
00:0C:29 Vmware VMware, Inc.
00:1C:14 Vmware VMware, Inc.
00:50:56 Vmware VMware, Inc.

00:06:1B Notebook Notebook Development Lab. Lenovo Japan Ltd.
00:12:FE LenovoMo Lenovo Mobile Communication Technology Ltd.
00:59:07 Lenovoem LenovoEMC Products USA, LLC
0C:CB:85 Motorola Motorola Mobility LLC, a Lenovo Company
14:1A:A3 Motorola Motorola Mobility LLC, a Lenovo Company
14:30:C6 Motorola Motorola Mobility LLC, a Lenovo Company

00:00:97 DellEmc Dell EMC
00:01:44 DellEmc Dell EMC
00:06:5B Dell Dell Inc.
00:08:74 Dell Dell Inc.
00:0B:DB Dell Dell Inc.
00:0D:56 Dell Dell Inc.
00:0F:1F Dell Dell Inc.
00:11:43 Dell Dell Inc.
00:12:3F Dell Dell Inc.
00:12:48 DellEmc Dell EMC
00:13:72 Dell Dell Inc.
00:14:22 Dell Dell Inc.
00:15:30 DellEmc Dell EMC
00:15:C5 Dell Dell Inc.

00:60:B0 HP
08:00:09 HP
10:00:90 HP



Using OSQuery to Enrich our Networking Data

- OSQuery is a great tool to grab point-in-time endpoint data to supplement networking data
- Compare NICs and ARP tables on endpoint against Networking equipment ARP tables
- Great way to do full-coverage rogue detection

Getting ARP data from OSQuery

- Using the *osqueryi* command locally we can test out our queries before running against whole environment

```
osquery> SELECT * from arp_cache;
+-------------+-------------------+-----------------+-----------+
| address     | mac               | interface       | permanent |
+-------------+-------------------+-----------------+-----------+
| 10.1        | 00:50:56          | em1             | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 10.1        | 00:0c:29          | em1             | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | br-423ba2418f06 | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 172.        | 02:42:ac          | docker0         | 0         |
| 10.1        | 00:00:00          | em1             | 0         |
| 10.1        | 00:50:56          | em1             | 0         |
```

```
osquery> SELECT a.address AS address, a.mac AS mac, a.interface AS interface, i.address AS interface_address
    ...> FROM arp_cache AS a
    ...> INNER JOIN interface_addresses AS i
    ...> ON a.interface = i.interface
    ...> WHERE a.interface = "em1";
+-------------+-------------------+-----------+-----------------------------+
| address     | mac               | interface | interface_address           |
+-------------+-------------------+-----------+-----------------------------+
| 10.         | 00:50:56:         | em1       | 10.                         |
| 10.         | 00:50:56:         | em1       | fe8                         |
| 10.         | 00:0c:29:         | em1       | 10.                         |
| 10.         | 00:0c:29:         | em1       | fe8                         |
| 10.         | 00:00:00:         | em1       | 10.                         |
| 10.         | 00:00:00:         | em1       | fe8                         |
| 10.         | 00:50:56:         | em1       | 10.                         |
| 10.         | 00:50:56:         | em1       | fe8                         |
| 10.         | 00:50:56:         | em1       | 10.                         |
| 10.         | 00:50:56:         | em1       | fe8                         |
| 10.         | 00:0c:29:         | em1       | 10.                         |
| 10.         | 00:0c:29:         | em1       | fe8                         |
| 10.         | 00:0c:29:         | em1       | 10.                         |
| 10.         | 00:0c:29:         | em1       | fe8                         |
| 10.         | 00:50:56:         | em1       | 10.                         |
```

RSAConference2019

SOC Automation

Easing the task of baselining

Automation Overview

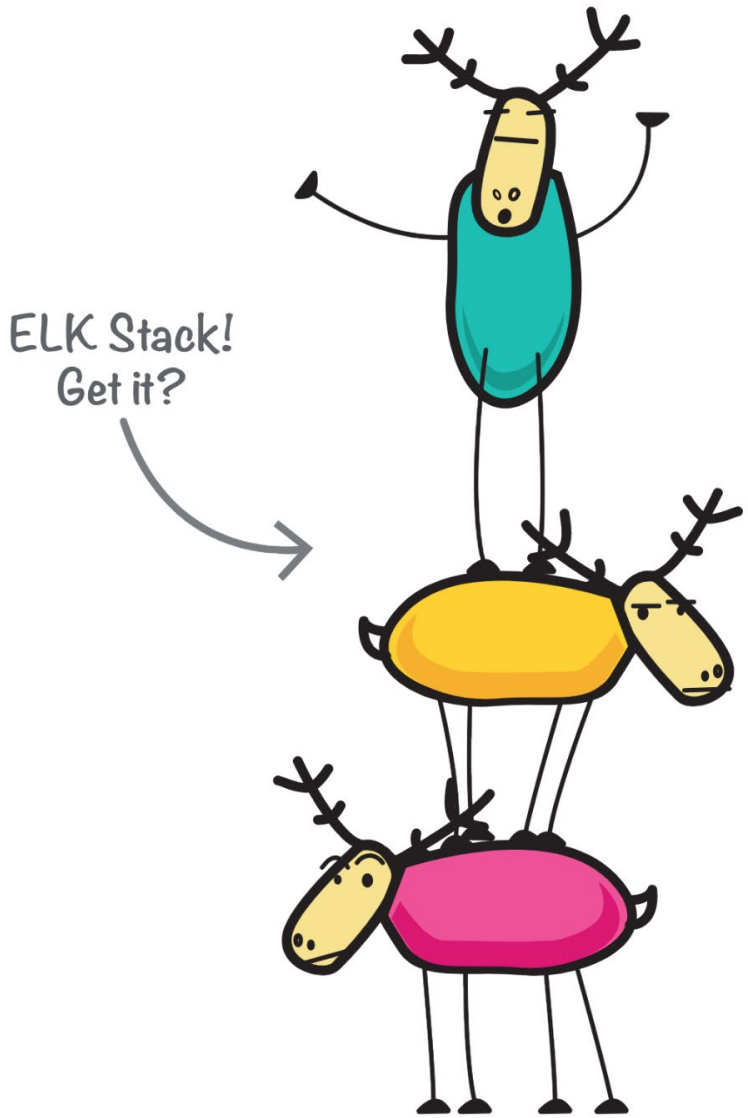
Is…

- Running repetitive tasks allowing your team to focus on studying of data
  - Data Collection
  - Aggregation

Isn't…

- Creating scripts/programs to find possible bad
  - This is what SIEM's do!
- Programs making and acting on decisions for us
  - "Should I automatically ban this machine because it doesn't match the baseline?"

Where do you put your data?



Elasticsearch

Logstash

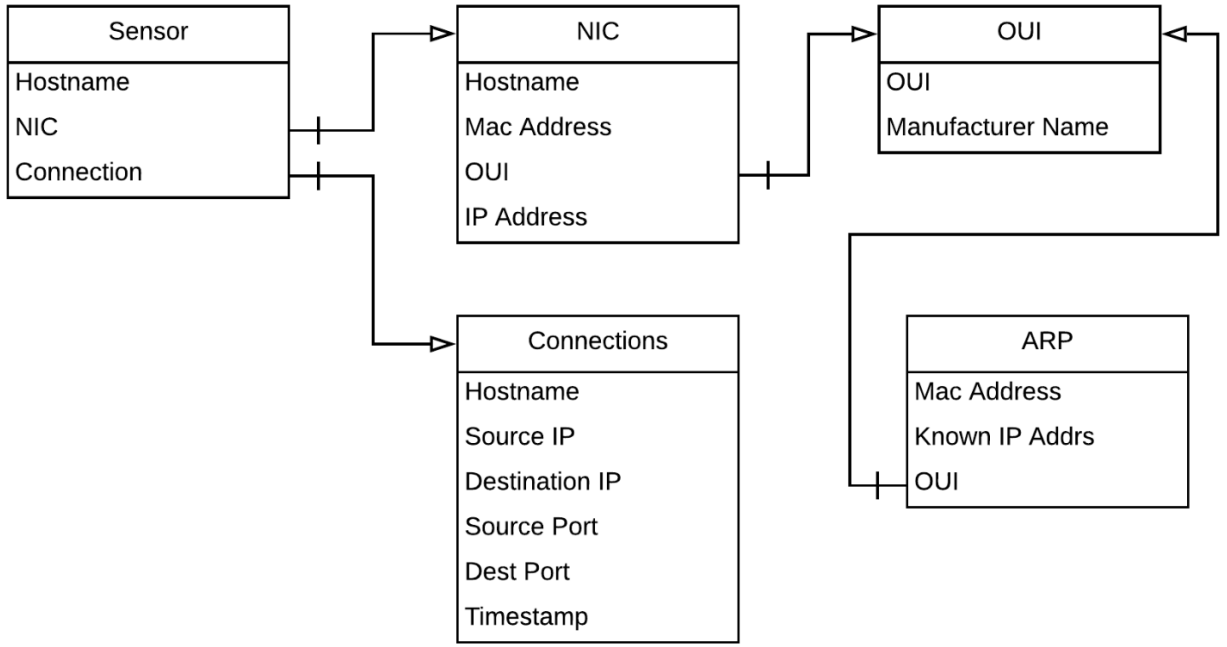
Kibana

Data Collection

- Many Examples in Python, Powershell, Bash
- Leverage libraries to do most of the heavy lifting
- Stack Overflow is your friend if you get stuck
- Choose a database or stack to store it all in

Data Storage

- Storage doesn’t need to be fancy, a single simple database will be fine
- Don’t throw all data into one table; Design a database for better organization
- POC with SQLite, easily stand up a database with Docker images

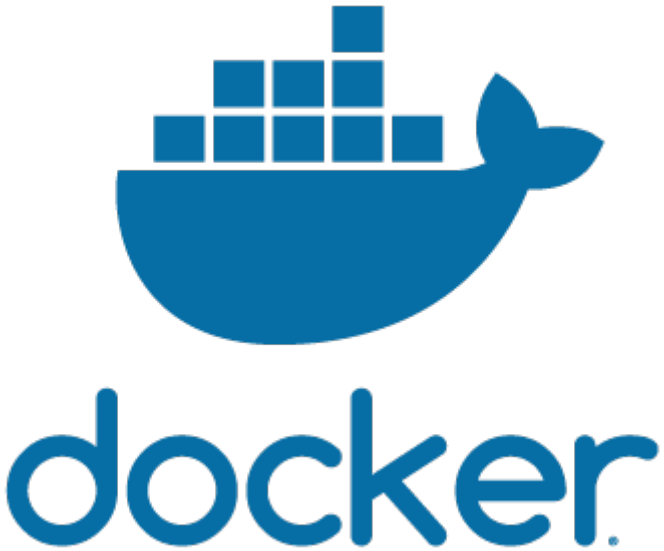


Querying Data

- Possible to leverage tools like Jupyter Notebook or R to automate some of your studies (programming heavy)
  - Takes less time to get started
- However, ELK is easier to query
  - May take longer to set up though there are some awesome tools out there to ease into the process

Docker

- Public Docker containers exist to get you started with a simple Elasticsearch, Logstash, and Kibana stack (easy)
- All that's left to you is setting up data inputs (more involved)
- Filebeat (part of ELK stack) allows for seamless integration with forwarding OSquery into logstash
- Easy-to-follow guide: https://elk-docker.readthedocs.io/



Filebeat

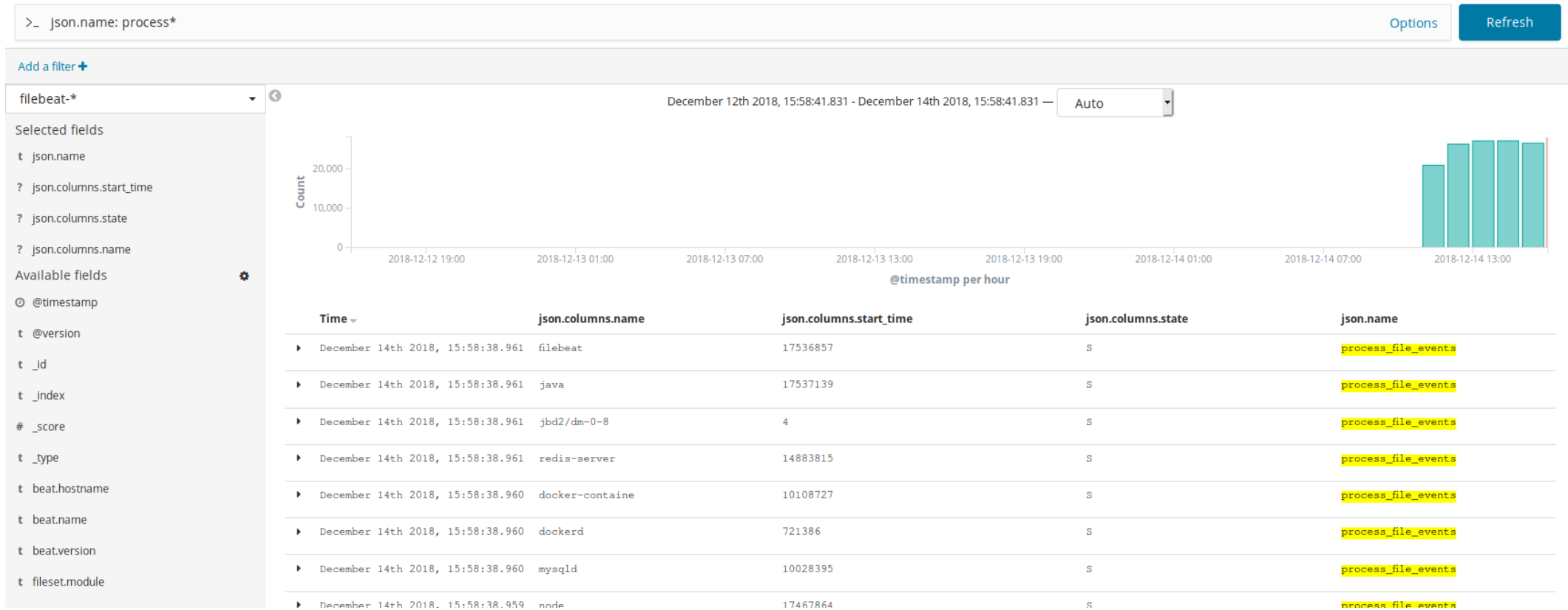
- Filebeat is a log forwarding service, part of the ELK stack
- Has built-in templates for transforming OSQuery data into an easily-digestible format.
- Somewhat involved setup to work properly with OSQuery
  - OSQuery also has built in support for pushing to LogStash

Configuring OSQuery for scheduled queries

```
{
  "options": {
    "host_identifier": “hostname",
    "schedule_splay_percent": 10
  },
  "schedule": {
    "arp_cache": {
      "query": "SELECT * FROM arp_cache;",
      "interval": 10
    }
  }
}
```

Next Steps

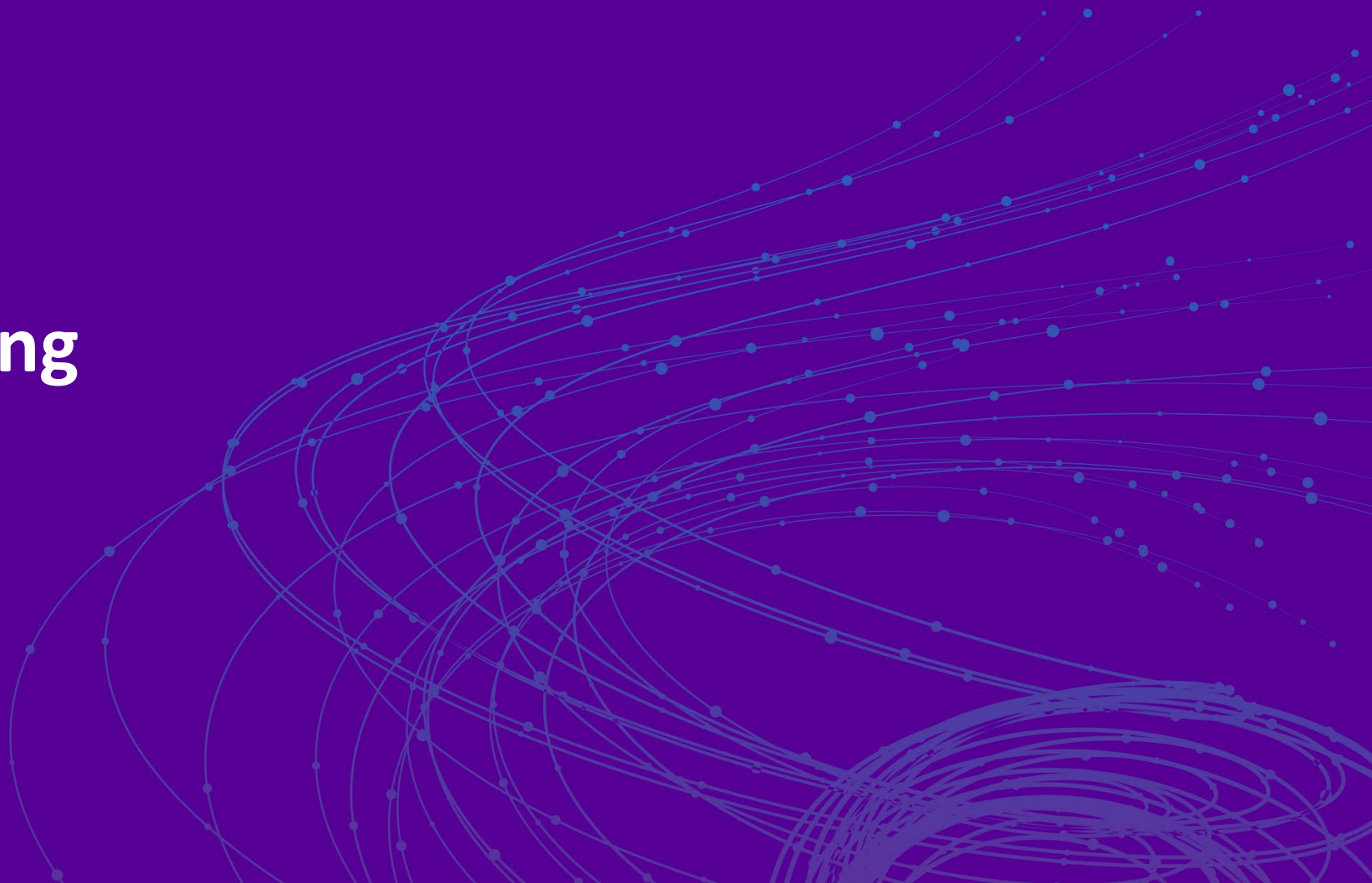
- Forward OSQuery (using Filebeat) to Logstash, start hunting with Kibana



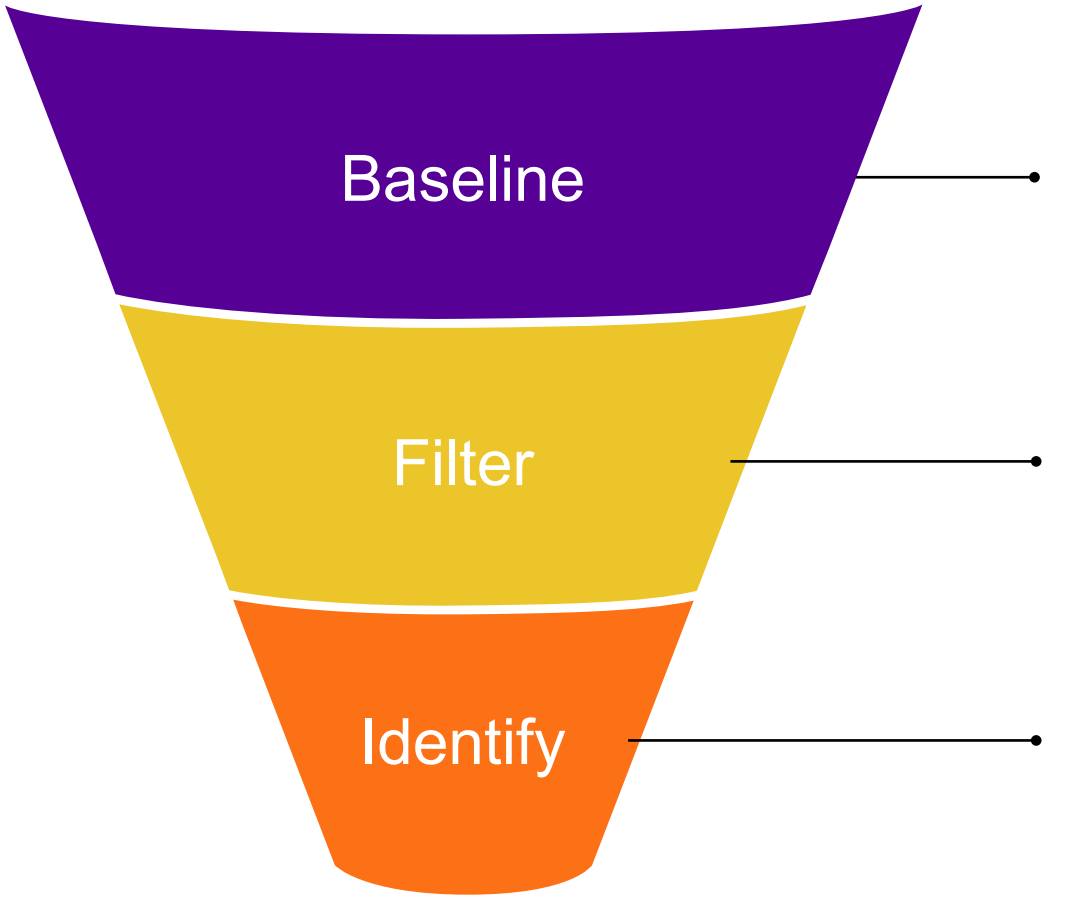
RSAConference2019

Threat Hunting

Tying it all together



Using Statistical Analysis for Threat Hunting

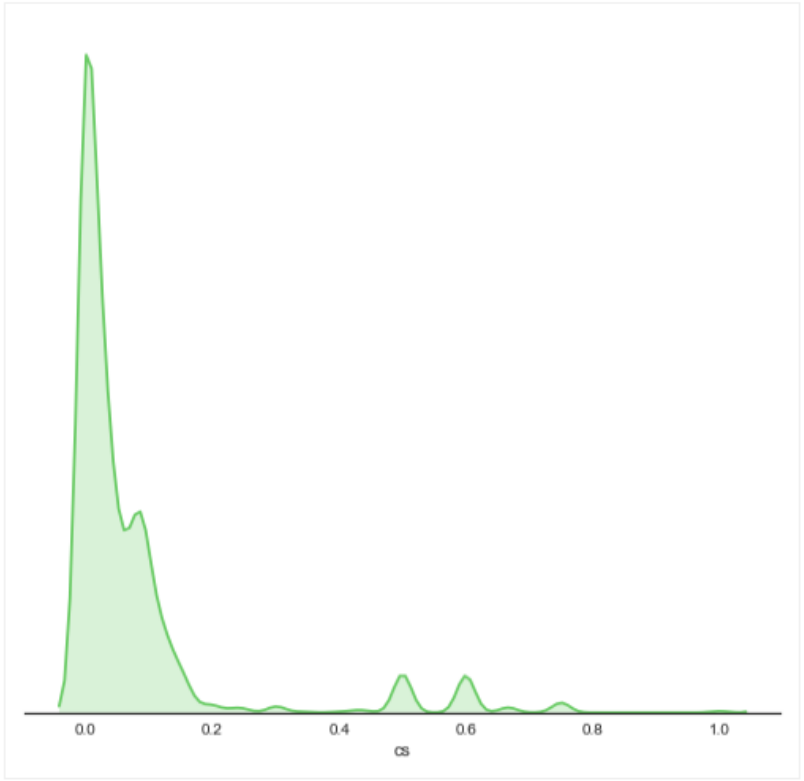


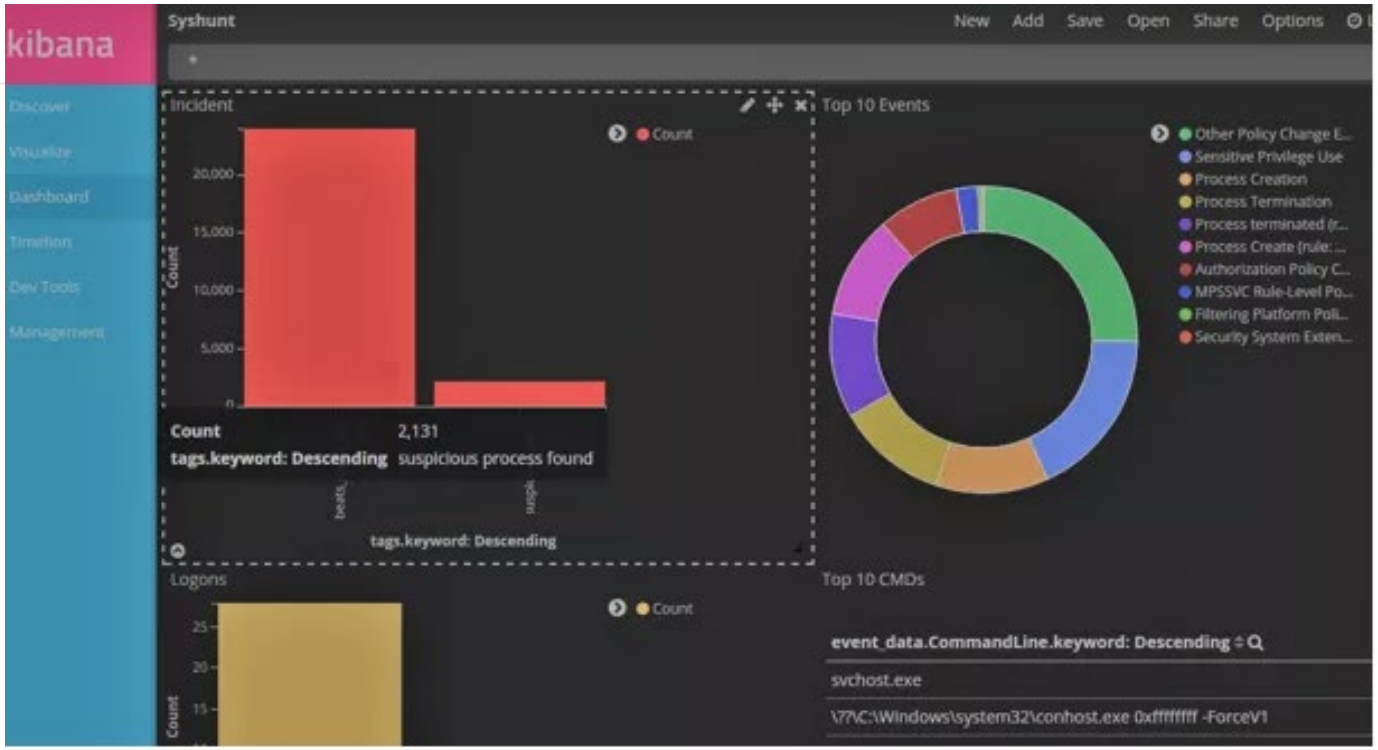
- Determine the “normal” dataset
- Eliminate baseline as noise
- Investigate anomalies & statistical deviations

Analyzing Data

In [6]:

```
sns.set(style="white", palette="muted", color_codes=True)
f, axes = plt.subplots(1, 1, figsize=(7, 7), sharex=True)
sns.despine(left=True)
sns.distplot(cs, hist=False, color="g", kde_kws={"shade": True})
plt.setp(axes, yticks=[])
plt.tight_layout()
```





Hunting Methodologies

- Back to the basics; now time to look for the abnormal
- Where to start?
- Search across environments for behavior and static IOC's
- Least prevalent occurrences tend to be most abnormal

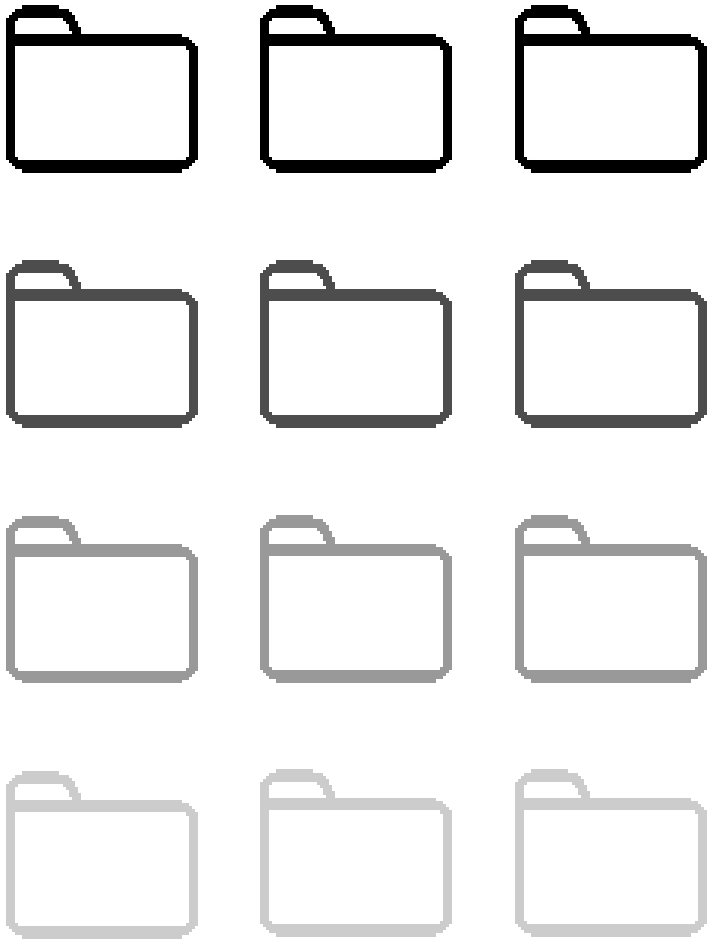
Mac Addresses – Uncommon Environmental OUIs

10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	52:54:00	RealtekU	Realtek (UpTech? also reported)
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.
10	00:50:56	Vmware	VMware Inc.

Prevalence of Executables

- Can you:
  - Identify abnormal software running on fewest endpoints?
  - Identify executables that are widespread but in unusual places?

- Yes!
  - Extract data on binaries from osquery
  - Combine into CSVs and perform text magic



Filtering Data

- Expressions to hunt for unusual indicators
  - Files that have a single character filename:
    ‘\\.\....,’
  - Files running one-folder deep from volume root:
    ‘(:\\[a-zA-Z0-9]{1,12}\\[a-zA-Z0-9]*\....,)’
  - Files run directly from Windows folder:
    ‘(:\\windows\\.{1,15},)’
  - Files with unusual extensions:
    ‘(\.bin,|\.dat,|\.log,|\.gif,|\.txt,|\.jpg,|\.rar,|\.sql,)’

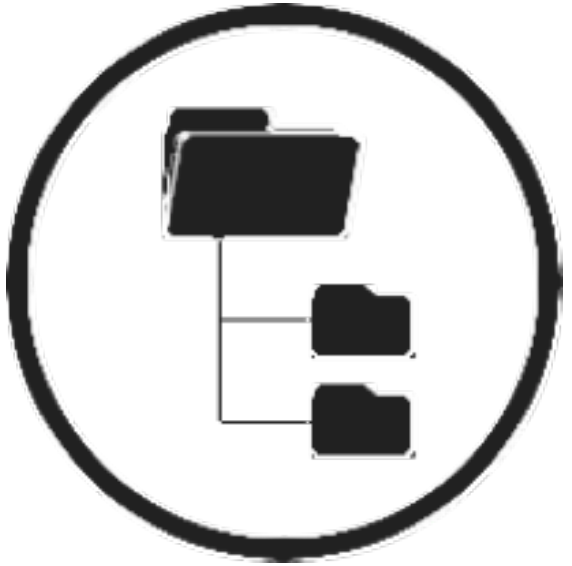
Mass Searching

- One-character file names:

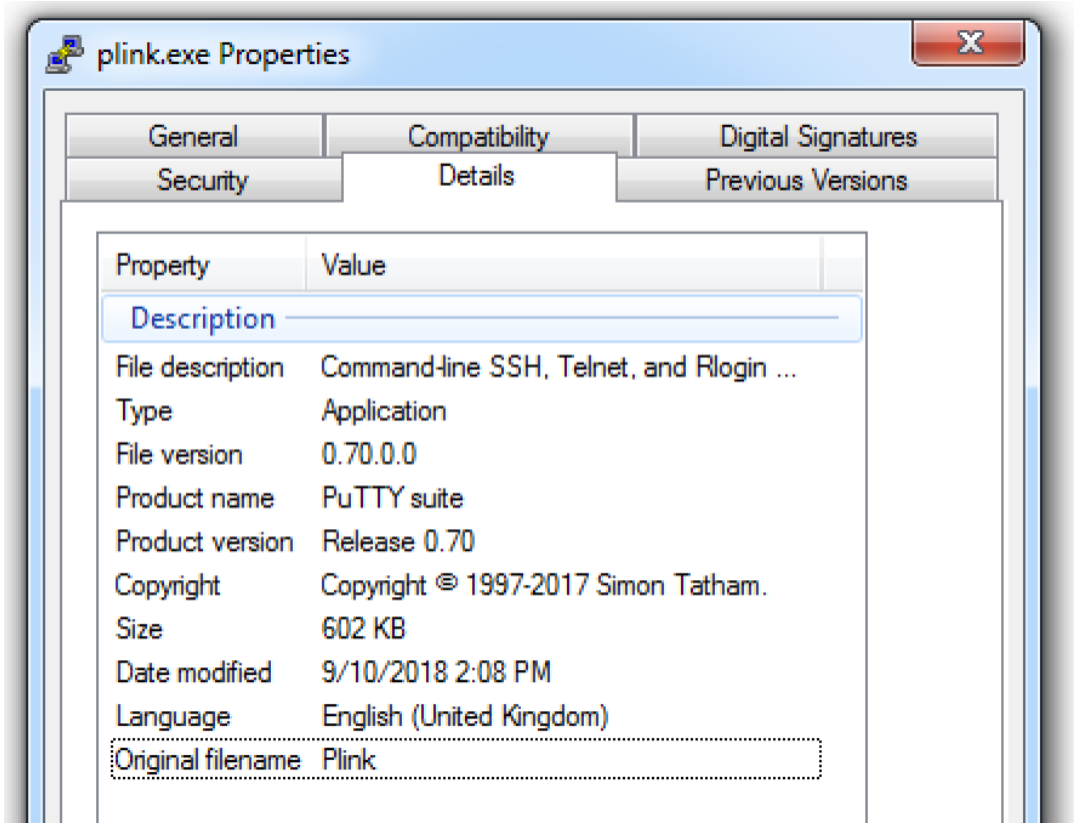
```
   6 c:\tdm-gcc-64_4.9.2\work\a.exe
   1 c:\accbk\agusta\y.bat
   1 c:\users\[redacted]\appdata\local\microsoft\windows\temporary internet
files\content.ie5\4unu162n\..exe
   1 sysvol\users\z9service\downloads\q.exe
   1 sysvol\program files (x86)\k2 for sharepoint 2013\z.bat
```

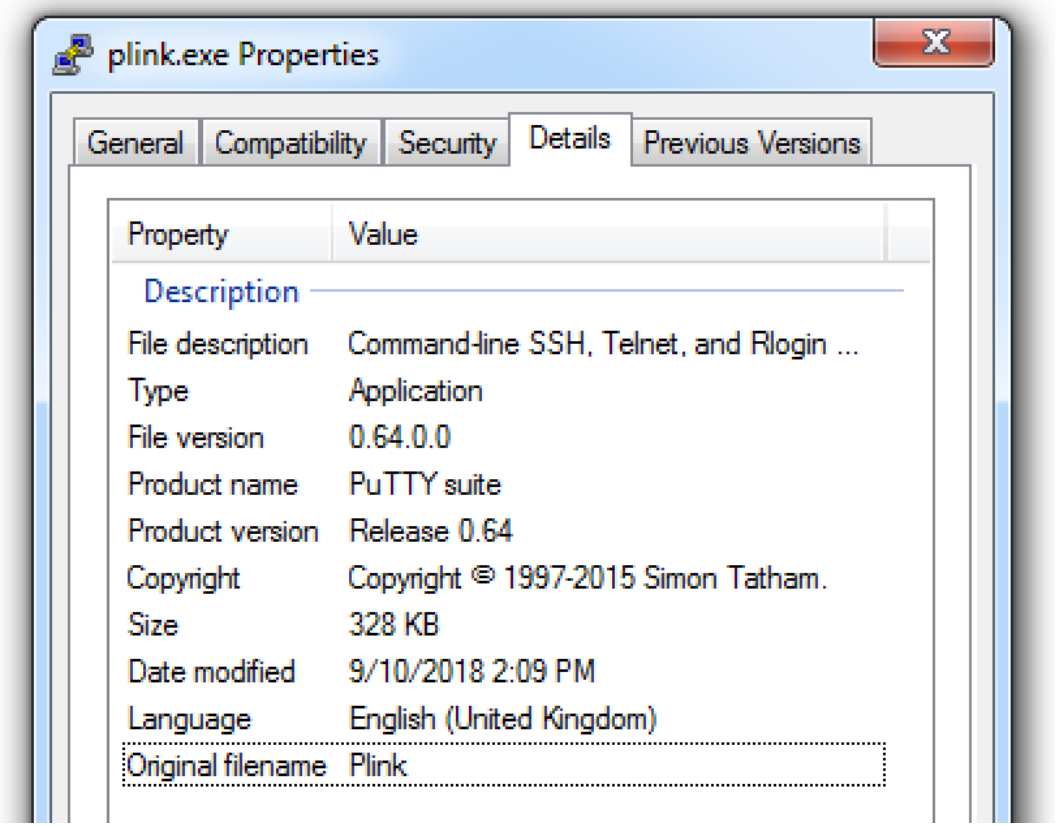
- Low prevalence in Windows Folder

```
  22 c:\windows\psexesvc.exe
   1 c:\windows\system32\oem\firstboot.cmd
   1 sysvol\windows\system32\dsget.exe
   1 c:\windows\system32\hpbpro.exe
   1 c:\windows\system32\scardsvr.exe
```



A Story of Two Executables (PLink)





Apply What You Have Learned Today

- Next week you should:
  - Create a quick plan on how to baseline your environment
    - Attempt some of the examples I showed today
- In the first three months following this presentation you should:
  - Minimizing previous bias, blueprint your environment as much as possible
    - Keep being proactive in mind!
- Within six months you should:
  - Automate your baselining tasks to focus on threat hunting and making your blueprinting more robust

Happy Hunting!