DEFENDER'S ADVANTAGE

You (and your SIEM) know what's "normal" for your organization.

@PwnieFan@infosec.exchange

Notes
(https://blogs.adelaide.edu.au/environment/2012/12/20/jens-amazing-lotus-flower-research/)

Be a savvy consumer when you're looking at macOS EDR tools. Some tools marketed as macOS tools are really just Windows EDR tools that were ported over to macOS. These tools might compile and run just fine, but they don't always understand what artifacts they should collect or what those artifacts mean. For example, I've seen several EDR tools that only record the parent process when gathering process data on Macs. The responsible process is way more important for filtering alerts than the parent process. Some tools that collect the responsible process don't actually make use of it when building/filtering their alerts. Make sure the ppl maintaining or selling the tool you choose actually understand macOS.

Once you pick out your telemetry tools, you need to think about how to keep your SIEM healthy. SIEMs and the pipelines that feed them don't maintain themselves. Expect to build this engineering capability in-house or pay a vendor.

Some tools/sources for telemetry on macOS systems in no particular order:

OpenBSM logs - These are recorded by default on macOS, but you need to use the audit binary to process them into something useful. Also, the default auditd config doesn't collect much. If you're relying on these logs for detailed, comprehensive information you'll want to change the config.

Unified logs - Most new Apple-generated logs are going into unified logs. They're a gold mine for all sorts of things, but not always easy to access or use. If you're going to use unified logs to send data to your SIEM, make sure your unified log queries are performant. Searching for keywords in messages is a double-wildcard search on a long field and is likely to cause you problems. This means you need to test on a machine where you can look at the all the unified logs in a specific timeframe. Once you know what you're looking for, you can build a good unified log query that filters on subsystem, process path, etc in addition to keywords in the message.

Osquery - Osquery is good open-source tool for asking questions about the configuration of a machine. You can ask what autoruns are installed, what the hash of a specific file is, network config and much, much more. I've found it to be most useful for getting a snapshot of a machine's state, then using those snapshots to establish normal/abnormal. The docs say osquery can collect process data as well, though I've never tested this and can't speak to how well it works.

Compliance Reporter - CR is a JAMF product that mainly collects process and network connection data and a bunch of other useful events as well. CR will record when changes are made to most autorun file locations and can help you track the configuration of a machine. One of my favorite features of CR is the ability to add specific unified log queries and have the results go to your SIEM along with all the other data. Unified log queries are good monitoring USB events, as well as getting more detail on some lateral movement methods (see 'How to Sink the Ducky'). The biggest bonus of CR, imho, is having a tool that was built for macOS and understands macOS system internals.

Crowdstrike/Falcon - Crowdstrike's agents runs on Macs and collects responsible process info (yay!) but doesn't let you build custom alerts that filter on responsible process info. You can pay for a feature to ship the data collected by Crowdstrike to a SIEM, where you could run whatever queries you like.

LimaCharlie - LimaCharlie's agent works on the macOS platform as well. I don't have a lot of experience with it, so I can't say too much. I will say that when I tested it, they weren't collecting responsible process information which limits the usefulness of their data.

ESF Playground - This is a super cool open source project that gathers whatever Endpoint Security Framework (ESF) events you're interested in. Really easy to work with on individual machines, but unfortunately I don't know of an easy way to use ESF Playground to forward data to a SIEM. May be there's a way to hide the ESF Playground window so you can run the tools in the background as an agent? ESF events do include a responsible process pid, but not a responsible process name. If you're ingesting these logs in a SIEM and you want the responsible process name, you'll have to do some enrichment after you ingest.

ALERT ON UNSIGNED LAUNCH AGENTS THEY SAID . . .

What if we alerted anytime we saw an adhoc/unsigned LaunchAgent or LaunchDaemon?

On busy days, 2-3 false positives per day. If we constantly update the exclusion list.

Attacker	QA	SOC	Detection Engineering
•`_´•	•‿•	(⊙_⊙')	(⊙_⊙')
✓	✓	✗	✗

@PwnieFan@infosec.exchange

Notes
Internally, we evaluate alerts on a bunch of (often competing) factors:
- Will this alert catch attacker activity?
- Can we verify that this alert will catch attacker activity? Yes, this sounds like an obvious question but with fancier “AI” or machine-learning tools it can be hard to tell. If the criteria for what generates an alert changes daily based on a proprietary, super-secret algorithm I can’t see, how do I know the alert will work when we need it to work? We don’t actually have a full QA department inside detection engineering but we do try to test and verify when we can.
- Is the false positive rate acceptable? The threshold for an acceptable false positive rate varies based on the situation. I might tolerate a higher false positive rate for an alert that fires on a critical event, or doesn’t fire very often. But alerts that fire a lot and have high false positive rates pretty quickly get a reputation as unreliable. Your SOC will start to ignore them.
- Is the false negative rate acceptable? Unfortunately, it’s rare to have an indicator specific enough that you can alert on attacker activity and only attacker activity. In most cases, you have to increase the false positive rate to decrease the false negative rate and vice versa. Visibility limitations are another gotcha with false negative rates. Always verify that the activity you’re looking for appears in your logs the way you expect. For example, xattr -w com.apple.quarantine "`xattr -p com.apple.quarantine known_good`" known_bad. Depending on how your EDR gathers process data you might see this as 2 separate processes, one process record for xattr -p and one process record for xattr -w. Or your EDR might see this as one command line. Your rule logic has to account for how the data will appear in your SIEM.
- Maintainability/Resilience is important too. The most obvious example of this is maintaining exclusion lists for a rule, if an exclusion list has to be manually updated every other day, your rule probably isn’t working. Other less obvious examples are rules that depend on updates to avoid false negatives - if you’re alerting on traffic to Tor exit nodes but you’re not automatically updating the list of exit node IP addresses, your rule is going to be stale pretty quickly. Yet another example is when a rule depends on an external data source. How stable is that GitHub repo you’re pulling from? What happens if the project disappears or publishes bad data?

ALERT ON UNSIGNED LAUNCH AGENTS THEY SAID . . .

What if we alerted anytime we saw ***a new*** adhoc/unsigned LaunchAgent or LaunchDaemon ***with a negative/unknown reputation on VirusTotal***?

1 false positive since implementation

Attacker	QA	SOC	Detection Engineering
•`_´•	•‿•	•‿•	•‿•
✓	✓	✓	✓

IOC

Context/ Enrichment

Better alert

@PwnieFan@infosec.exchange

Notes
The context/enrichment step can be from external or internal sources, like a SIEM. Being able to add context based on what's normal in your environment is incredibly valuable.

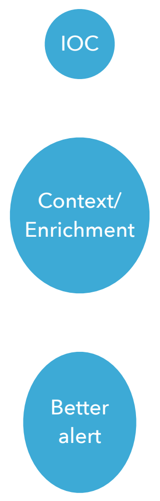
ALERT ON TOUCH BACKDATING THEY SAID . . .

- The 2020 variant of OceanLotus used touch –t to backdate the plist file.
  - Developer and compiling tools also backdate files.
- Other problematic commands:
  - xattr –d or xattr –c
  - security
  - chmod +x or chmod 777
  - uuidgen
  - …

@PwnieFan@infosec.exchange

Notes
I really appreciate a good, deeply technical malware write-up. What would make them even better is if every suspicious process indicator included the full process tree with responsible process and parent process information. I can't count how many times I've read something calling out a process as suspicious, only to go in our environment and find that xattr or security command gets run several times a day. Chainbreaker uses the security command to dump the keychain for password cracking. Some legit tools also run the same exact command-line. Process tree information helps establish the context for suspicious commands and gives me possibilities for filtering.

CONTEXT FOR LOOBINS/LOLBINS?



LOOBins/LOLBins used by attackers

- Has this command been run in my environment recently?
- Has this responsible process run this command in my environment recently?
- How common is this responsible process in our environment?
- What does VT say about the responsible process?

NORMALIZED BASELINE DETECTION (NBD)

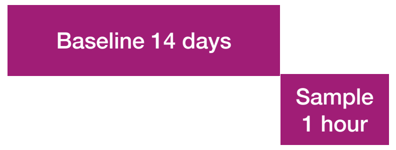
An example

Command line seen in BASELINE	Responsible process seen in BASELINE
xattr -d -r com.apple.quarantine /Applications/ Google Chrome.app	/Library/Google/GoogleSoftwareUpdate/ GoogleSoftwareUpdate.bundle/Contents/Helpers/ GoogleSoftwareUpdateDaemon

Command line seen in SAMPLE	Responsible process seen in SAMPLE	Matches entry in baseline results?	Is this activity suspicious?
xattr -d -r com.apple.quarantine /Applications/	Same as above	Yes	No, this is expected behavior from Google Chrome.

@PwnieFan@infosec.exchange

NORMALIZED BASELINE DETECTION (NBD) FOR SUS COMMANDS

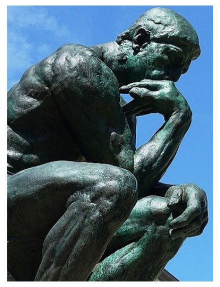


For each command we care about:

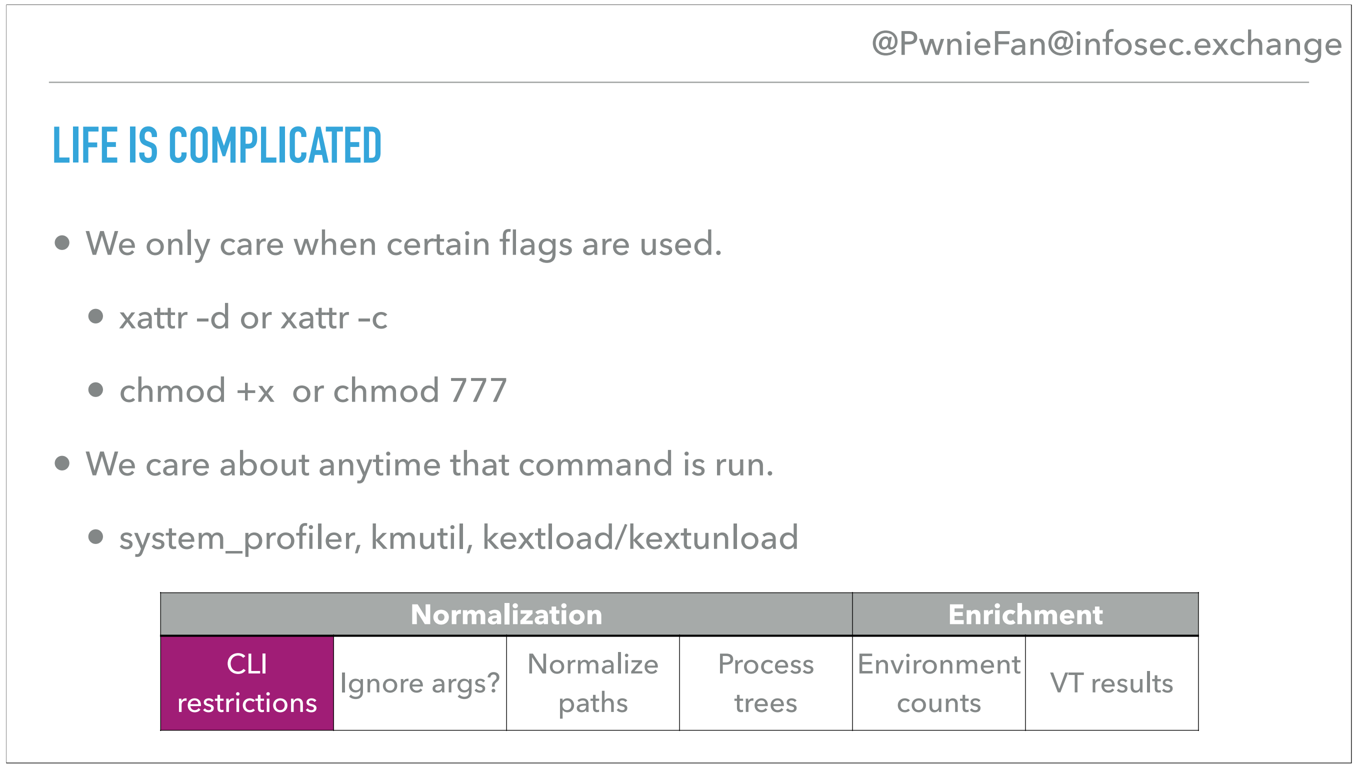
- baseline_results = instances of cmd in baseline
- sample_results = instances of cmd in sample period

For each result in sample_result:

- is this command new?
- is this responsible process/command pair new?
- if yes to either question: write enriched record to SIEM


What *is* new?

@PwnieFan@infosec.exchange



@PwnieFan@infosec.exchange

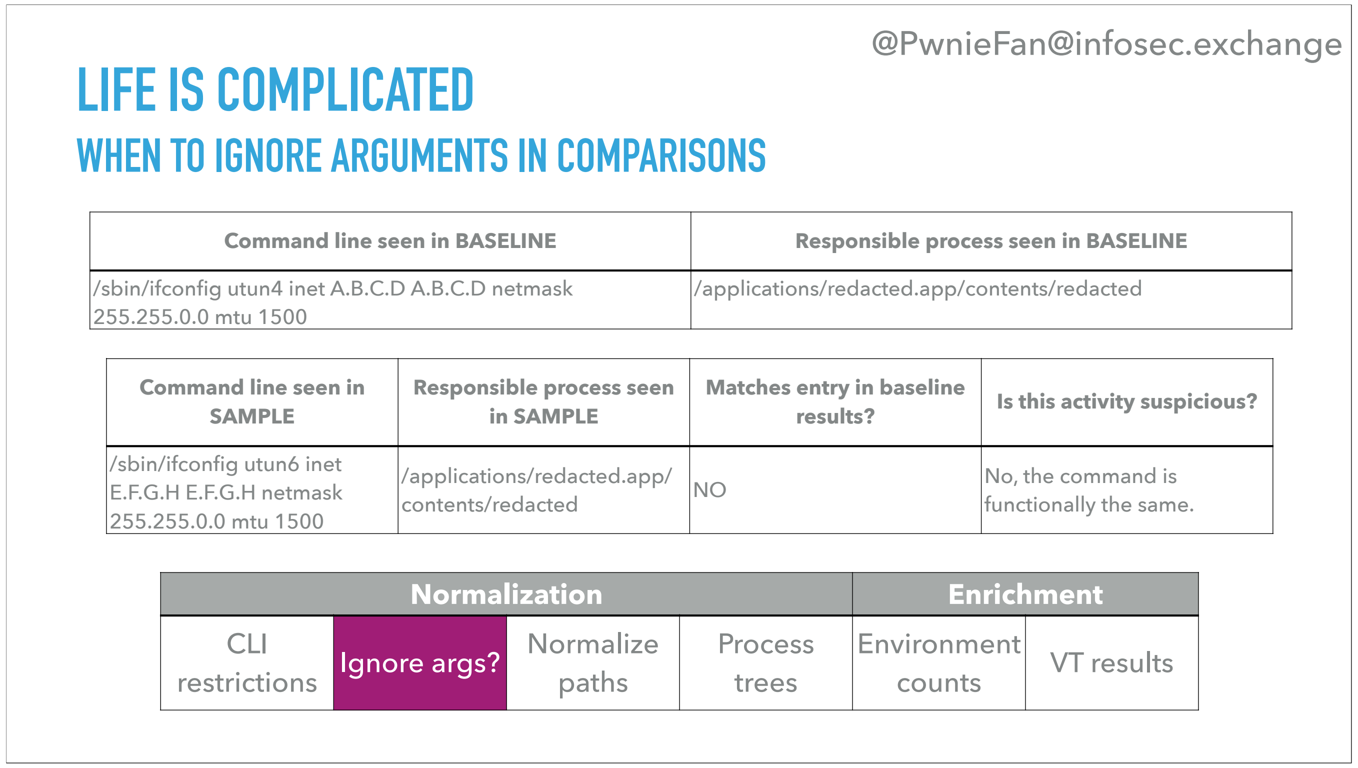
LIFE IS COMPLICATED

- We only care when certain flags are used.
  - xattr -d or xattr -c
  - chmod +x or chmod 777
- We care about anytime that command is run.
  - system_profiler, kmutil, kextload/kextunload

Normalization				Enrichment	
CLI restrictions	Ignore args?	Normalize paths	Process trees	Environment counts	VT results

Notes
My code uses a configuration file that looks something like this:

```
cmds_list.append(
    {
        "command": "kmutil",
        "process_arg_restrictions": ""
    }
)
cmds_list.append(
    {
        "command": "xattr",
        "process_arg_restrictions": '(process.args:("com.apple.quarantine" && "-d") OR process.args:("-c"))'
    }
)
.... and many more commands
```



@PwnieFan@infosec.exchange

LIFE IS COMPLICATED

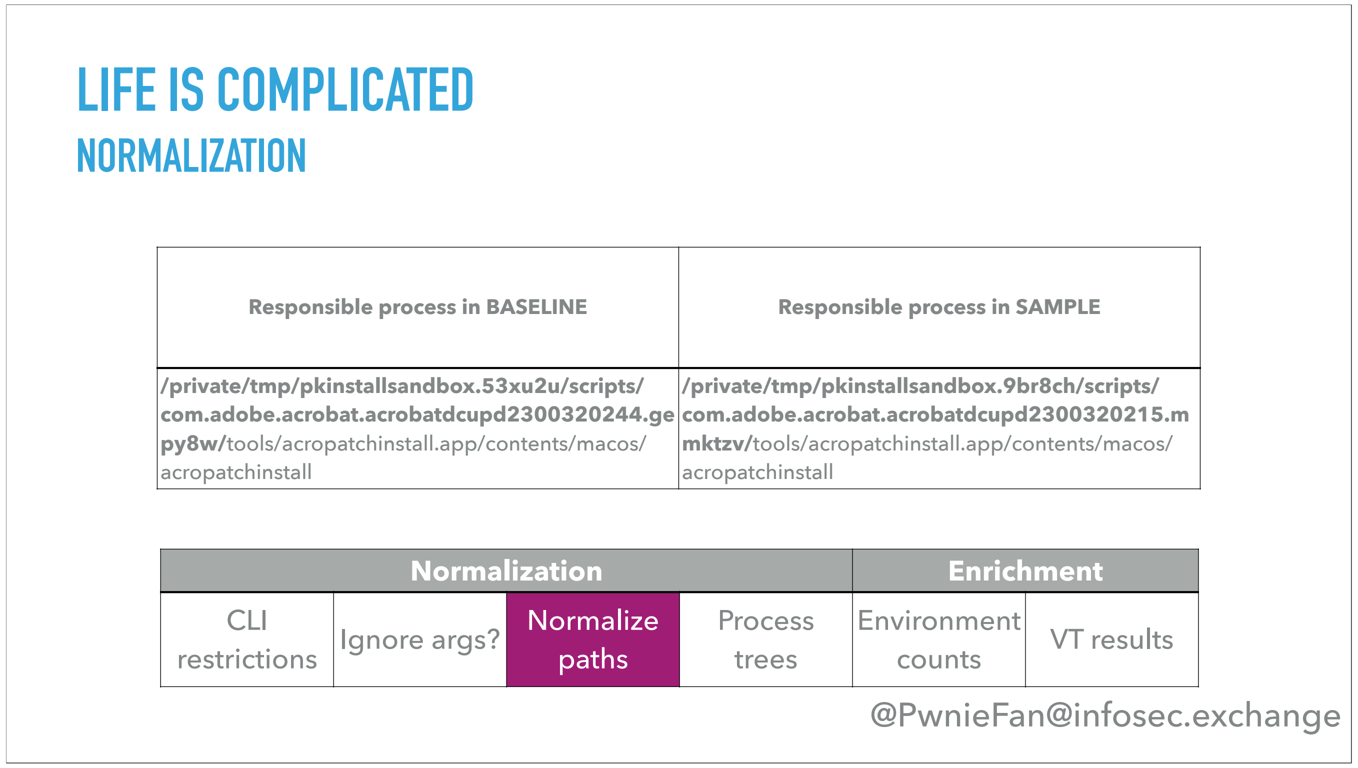
WHEN TO IGNORE ARGUMENTS IN COMPARISONS

Command line seen in BASELINE	Responsible process seen in BASELINE
/sbin/ifconfig utun4 inet A.B.C.D A.B.C.D netmask 255.255.0.0 mtu 1500	/applications/redacted.app/contents/redacted

Command line seen in SAMPLE	Responsible process seen in SAMPLE	Matches entry in baseline results?	Is this activity suspicious?
/sbin/ifconfig utun6 inet E.F.G.H E.F.G.H netmask 255.255.0.0 mtu 1500	/applications/redacted.app/contents/redacted	NO	No, the command is functionally the same.

Normalization				Enrichment	
CLI restrictions	Ignore args?	Normalize paths	Process trees	Environment counts	VT results

Notes
My code actually does both versions of the analysis (with and without ignoring params) for all commands. This way I have all the data in the SIEM and I can decide which approach makes sense for individual rules. Even though I think with commands like ifconfig the answer will almost always be ignore params.



LIFE IS COMPLICATED

NORMALIZATION

Responsible process in BASELINE	Responsible process in SAMPLE
/private/tmp/pkinstallsandbox.53xu2u/scripts/ com.adobe.acrobat.acrobatdcupd2300320244.gepy8w/tools/acropatchinstall.app/contents/macos/ acropatchinstall	**/private/tmp/pkinstallsandbox.9br8ch/scripts/ com.adobe.acrobat.acrobatdcupd2300320215.mmktzv/**tools/acropatchinstall.app/contents/macos/ acropatchinstall

Normalization				Enrichment	
CLI restrictions	Ignore args?	Normalize paths	Process trees	Environment counts	VT results

@PwnieFan@infosec.exchange

Notes
Both responsible processes and command lines need normalization. You will probably find some things to normalize that are specific to your environment.

Responsible processes:
User directories
```
		re.sub("\/users\/<yourusernameregex>\/", "/*/", responsible_process)
```
AppTranslocation
```
		re.sub(“\/private\/var\/folders\/[a-z0-9_]{2}\/[a-z0-9_]{30}\/[a-z]{1}\/apptranslocation\/[a-z0-9\-]{36}\/[a-z]{1}\/","/*/",new_responsible_process)
```
Temp dirs created by installer
```
		re.sub(“\/private\/tmp\/pkinstallsandbox\.[a-z0-9]{6}\/","/*/",new_responsible_process)
		re.sub(“\/var\/folders\/[a-z0-9_]{2}\/[a-z0-9_]{30}\/[a-z]{1}\/“,”/*/",new_responsible_process)
```
System extension activity
```
		re.sub("\/library\/systemextensions\/[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}\/","/library/systemextensions/*/",new_responsible_process)
```

Command-lines:
User directories
```
		re.sub("\/users\/<yourusernameregex>\/", "/*/", command_line)
```
IPv4 and IPv6 addresses
```
		re.sub(" [0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}", " *", new_command_line)
		re.sub(" ([a-f0-9:]+:+)+[a-f0-9]+", " *", new_command_line)
```
Temp dirs created by installer
```
		re.sub(“\/var\/folders\/[a-z0-9_]{2}\/[a-z0-9_]{30}\/[a-z]{1}\/“,”/*/“,new_command_line)
```
Interface names in ifconfig commands (most common way they would appear, doesn’t cover everything)
```
		re.sub("ifconfig [a-z0-9]{3,10} ", "ifconfig * ", new_command_line)
```
Google Chrome stuff
```
		re.sub(“\/tmp\/ksdownloadaction\.[a-z0-9]{10}/“,"/tmp/ksdownloadaction.*/",new_command_line)
		re.sub(“\/tmp\/ksinstallaction\.[a-z0-9]{10}/","/tmp/ksinstallaction.*/",new_command_line)
```
UUIDs
```
		re.sub(“-uuid [a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}”,"-uuid *",new_command_line)
```

LIFE IS COMPLICATED
RESPONSIBLE PROCESS ANOMALIES

Command line	Responsible process
/usr/sbin/system_profiler -nospawn -xml SPConfigurationProfileDataType	/usr/sbin/system_profiler

Normalization				Enrichment	
CLI restrictions	Ignore args?	Normalize paths	Process trees	Environment counts	VT results

@PwnieFan@infosec.exchange

Notes
Here are some ways to find a responsible process when the data is confusing:
Look for an executable in the thread that isn't a built-in executable. This means looking through the process tree for a process that doesn't start with /usr/ or /bin/* or /sbin/*.
Look for a responsible process in the thread that isn't a built-in executable. Same search as above, but we're looking at the responsible process field instead of the executable path.
Look for an open command or a shell command running a shell script. These are commands like open /Applications/Google Chrome.app or /bin/sh ashellscript.sh.

LIFE IS COMPLICATED

NEW != BAD

- More context
  - How common is this responsible process hash in our environment?
  - How common is this responsible process path in our environment (more normalization!)?
  - And, yeah, VirusTotal info.

Normalization				Enrichment	
CLI restrictions	Ignore args?	Normalize paths	Process trees	Environment counts	VT results

@PwnieFan@infosec.exchange

Notes

After all of that, we end up with this more detailed pseudocode. This “simple” code took me several months to develop. Not just to code and cover edge cases, but to determine what normalizations were necessary. Because multiple queries against the SIEM were necessary for each step, there’s also a lot of error-handling questions that come up. Questions like “If this query fails, what should I return?” “If no data is returned, do I err on the side of reducing noise or making more noise?” Also, if we can’t find a responsible process hash do we still write a record to the SIEM? I decided we should, but it means we don’t have any VT information to filter on in that record.

If you do detection engineering, you’re probably thinking this looks a lot like an enrichment pipeline. And you’re right. :)

```
for each command:
    baseline = normalize(instances of cmd in baseline (maybe with flag restrictions))
    sample = normalize(instances of cmd in sample (maybe with flag restrictions))

    for each result in sample:
       seen in the baseline?
       seen with the same responsible process in the baseline?*
        if no to either question: enrich with context and write record to SIEM

    for each result in sample:
      seen in the baseline ignoring arguments?
      seen with the same resp. process in the baseline ignoring arguments?*
      if no to either question: enrich with context and write record to SIEM

* if responsible process == self, look in process tree for a better answer
```

TOUCH COMMANDS FROM OCEANLOTUS

```
touch -t 1910071234 ~/Library/LaunchAgents/com.apple.launchpad.plist
touch -t 1910071234 ~/Library/WebKit/b2NlYW5sb3R1czIz
touch -t 1910071234 ~/Library/WebKit/com.apple.launchpad
```

Responsible process:

```
/bin/bash /Applications/conkylan.app/Contents/MacOS/conkylan
```

@PwnieFan@infosec.exchange

@PwnieFan@infosec.exchange

RECORD IN SIEM

```
 "rule": { "meta": {"reason_for_alert": "not seen in the baseline period"},
"process": {
   "responsible": {"name": "bash", "executable": "/bin/bash" },
   "name": "touch",
   "normalized_command_line": "\"touch -t 1910071234 /*/Library/LaunchAgents/
com.apple.launchpad.plist\"",
   "command_line": "touch -t 1910071234 /Users/loonicorn/Library/LaunchAgents/
com.apple.launchpad.plist"
"stats": {
   "processes_seen_in_baseline": 1116, "processes_seen_in_sample": 1000,
   "other_machines_with_file": 7628, "other_machines_with_hash": 6955 },
virustotal: {
 "malicious": 0, "tags": "64bits,multi-arch,macho,arm,signed"
 "signature_info": { "signers": "Apple Inc; Apple Inc.; Apple Inc.", "verified": "Valid"}
```

DETECTING OCEANLOTUS' TOUCH COMMAND WITH NBD

```
INDEX enriched_commands
process.name:“touch”
NOT process.responsible.executable:"REDACTED"
```

@PwnieFan@infosec.exchange

Notes
Seriously, the original rule had a very long exclusion list that had to be updated every couple of months. Here we have to exclude just one thing.

WHAT'S NEXT

- Links posted at @PwnieFan@infosec.exchange or go to https://github.com/megancarney
- Other NBD ideas currently in testing
  - Abnormally busy MS application process trees
  - Unexpected executables in MS application process trees