Scanning VirusTotal's firehose

skyblue.team/posts/scanning-virustotal-firehose/

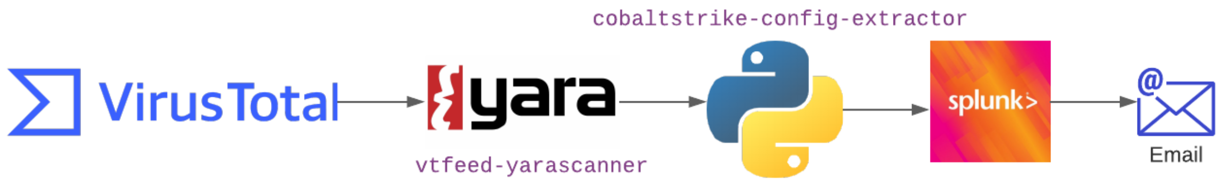
Sep 21, 2021 · 584 words · 3 minute read

Let's say one of your adversaries is known for using a given malware family, custom or off-the shelf. Even if the coverage is biased and limited, samples on VirusTotal (VT) are the low-hanging fruits that keep on giving.

At $WORK, we are lucky to have access to the Virus Total feeds/file API. This API endpoint is the firehose of VirusTotal: it allows downloading each sample submitted to VT in pseudo-real-time. The feed is unfiltered (we are not talking about VT's LiveHunt feature) so the volume is HUGE.

We set the crazy objective to **extract and push IOC in real-time for a given malware family submitted to VirusTotal**. For this blog post, as an example, we will focus on Cobalt Strike.

The steps are:



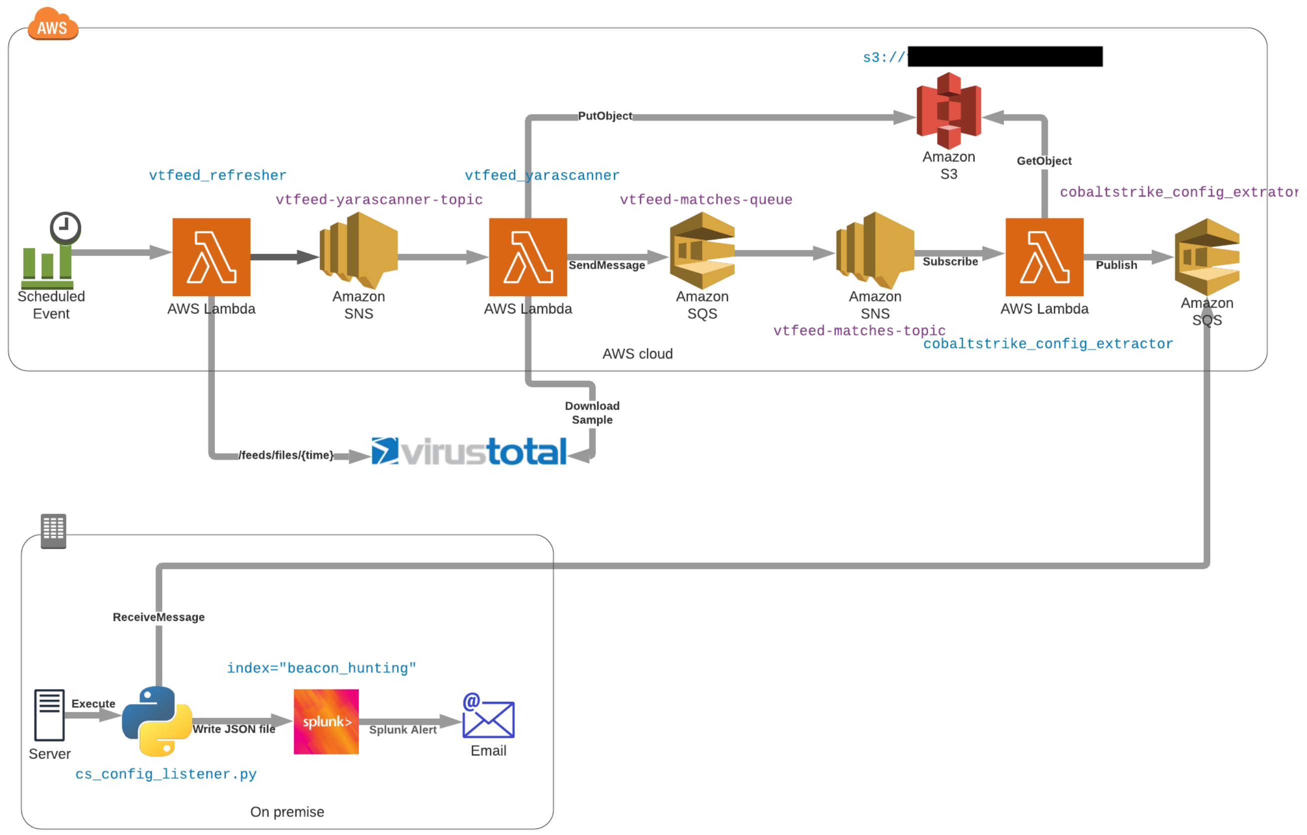
1. Download each sample submitted
2. Apply Yara rules matching the malware families we are interested in
3. Automatically extract C2 configuration
4. Disseminate IOC

Initially, we used our on-premises infrastructure with 2-3 servers. Quickly, the operational maintenance killed us:

- Our Celery cluster was regularly KO.
- Everything had to be very carefully tuned (memory limits, batch size, timeout, retries), we were constantly juggling with the balance between completeness, stability, and speed.
- Adding an under-performing Yara rule could break the platform.
- It was also not a good use of our computing resources as VT's activity is not evenly spread across the day: our servers were under-used most of the day while overloaded during the peaks.

Going Serverless 🔗

Taking a step back, it jumped out at us that this was a textbook example for a Serverless architecture. It was easy to refactor our on-prem code into self-contained functions and *glue* them together with Amazon SQS:

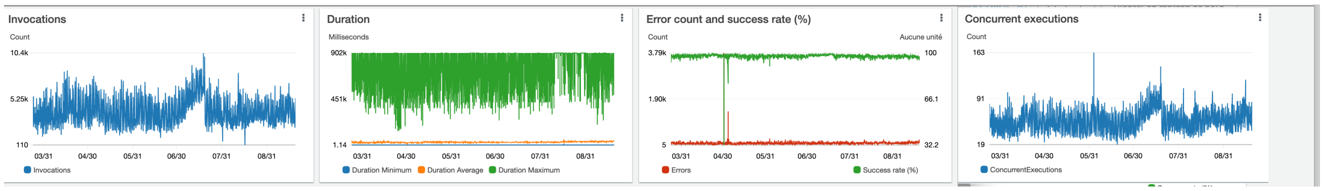


The platform has been running smoothly for 18 months, and from an operational point of view, we love it:

- The scalability of the platform allowed us to not mind anymore about the performance of each rule: we can add our Yara rules quite freely instead of cherry-picking and evaluating carefully each addition.
- SQS handles the whole retry mechanism.
- Adding a new dissector is as easy as plugging a new Lambda function to the Amazon Simple Notification Service (SNS) topic.
- Everything is decoupled, it is easy to update one part without touching the rest.
- Each new release of libyara increases its performance and it is directly correlated to the execution duration's average.
- Everything is instrumented, we learned to love the AWS Monitoring Console.

Performance Stats 🔗

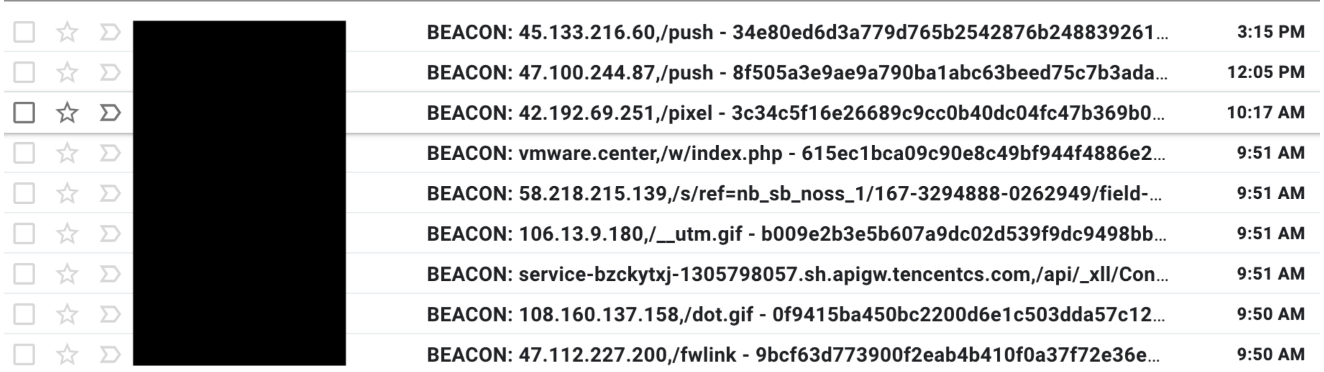
For those who like numbers, here is a screenshot of the activity of the last 6 months:



On average:

- A batch of samples is scanned in less than 30s
- There are always 45 Lambda functions running at any given time
- 97% of the executions are successful
- We send 150 samples per minute (before deduplication) to dissectors

CobaltStrike 🔗



[redacted]

BEACON: 45.133.216.60,/push - 34e80ed6d3a779d765b2542876b248839261… **3:15 PM**
BEACON: 47.100.244.87,/push - 8f505a3e9ae9a790ba1abc63beed75c7b3ada… **12:05 PM**
BEACON: 42.192.69.251,/pixel - 3c34c5f16e26689c9cc0b40dc04fc47b369b0… **10:17 AM**
BEACON: vmware.center,/w/index.php - 615ec1bca09c90e8c49bf944f4886e2… **9:51 AM**
BEACON: 58.218.215.139,/s/ref=nb_sb_noss_1/167-3294888-0262949/field-… **9:51 AM**
BEACON: 106.13.9.180,/__utm.gif - b009e2b3e5b607a9dc02d539f9dc9498bb… **9:51 AM**
BEACON: service-bzckytxj-1305798057.sh.apigw.tencentcs.com,/api/_xll/Con… **9:51 AM**
BEACON: 108.160.137.158,/dot.gif - 0f9415ba450bc2200d6e1c503dda57c12… **9:50 AM**
BEACON: 47.112.227.200,/fwlink - 9bcf63d773900f2eab4b410f0a37f72e36e… **9:50 AM**

We are using CobaltStrikeParser from Sentinel One to parse the beacons, then we are sending the JSON output to our Splunk instance.

There are two uses of this data:

- Threat Hunting: tracking some Threat Actors
- Proactive protection: adding proactively the IOC to a watchlist in our scope

For Threat Hunting perspectives, we implement alerting for things like:

- Specific watermark identifiers
- Patterns in the C2 domain
- Non-standard values for some fields
- Use of some options or specific malleable profile

Regarding proactive Defense, there is currently no automatic pipeline to push the IOC into a WatchList/DenyList for one reason: it is not uncommon to see trolling BEACONs using legitimate and “assumed safe” domains. To mitigate that, we plan to have a kind of Slack/Mattermost bot that will make us approve each entry seamlessly.