

Use EVTX files on VirusTotal with Timesketch and Sigma (Part1)

 osdfir.blogspot.com/2021/11/use-evt-x-files-on-virustotal-part1.html

Alexander Jäger

TDLR:

VirusTotal added a new feature to allow VirusTotal Enterprise customers to download Windows XML EventLog files (.evtX) for a sandbox execution of submitted samples. This article covers how this feature can help incident responders and digital forensic analysts develop detections and how to use the new API to test an existing detection pipeline.

Over the course of the article, tools like DFTimewolf, Plaso and Timesketch will be used.

Disclaimer

Most of our other blog posts cover open source techniques. The API feature described in this post is part of a commercial offering from VirusTotal and is not available to free tier accounts. Similar files could be created with Cuckoo Sandbox, an open source malware analysis system.

Prerequisite

In order to follow this guide, we will need a running Timesketch server and docker on our local computer and installed DFTimewolf. In addition we need access to the private API of VirusTotal.

Context

Windows EventLogs are an important source for incident response and digital forensics. Analysts usually extract events from the EVTX files to put them into timelines and use events to detect anomalous activity. Several open source tools are available to extract and parse EVTX files and create timelines out of them (e.g. Plaso).

Besides EVTX data from real incidents, there are situations where analysts need EVTX files to test or improve tools or rules. To make or create use of that data, analysts usually need access to a clean system. So if they want to create new signatures like Sigma rules that are based on event log entries, analysts have to either create mock entries or maintain a lab, execute a sample, capture event logs, parse them and then start with writing detection rules.

Maintaining such a lab environment comes with a cost. Cost of hardware or software, and time for maintenance are just the tip of the iceberg of these costs. Wouldn't it be handy to have a system where we can upload a sample, it would run and provide us with the event logs after the execution?

VirusTotal has been a central point of collecting and executing samples since 2004 and its first sandbox reports were made available during 2012, which has helped analysts to create detection rules for malware. In 2021 VirusTotal added support for Sigma to show Sigma rule matches against the EventLogs produced when samples are run. The VirusTotal team added a feature in October 2021 to enable users with access to the private API to download EVTX files for samples.

VirusTotal API

The VirusTotal API has two tiers, a private and a public. While the majority of API endpoints and information is freely available to registered users, the private API is restricted to premium customers. The first one being free to any user and the other one is limited to Enterprise users. The description of all the API endpoints and features can be found on the VirusTotal website.

How can this be used with one sample?

Say we want to create a detection rule in our environment for a given sample, the only thing we need to start with is the SHA-256 hash of the sample. With this hash, we can go to VirusTotal and download the EVTX file. For this example, we will have a look at one sample that we will first download manually to add it to Timesketch and after that a scalable method with DFTimewolf. For this we will focus on the following sample:

3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306. This sample was uploaded to VirusTotal on 2021-10-07 06:18:24 UTC and its VirusTotal report contains several hints on why it is bad, for example several hits on Sigma rules (which is only visible for enterprise VirusTotal users).

||| CRITICAL 1 HIGH 3 MEDIUM 4 LOW 17

- ⚠️ 1 match for rule **Suspicious Use of Procdump on LSASS** by Florian Roth from Sigma Integrated Rule Set (GitHub)
Detects suspicious uses of the Sysinternals Procdump utility by using a special command line parameter in combination with the lsass.exe process. This way we're also able to catch cases in which the attacker has renamed the procdump executable.
- ⚠️ 2 matches for rule **LSASS Memory Dumping** by E.M. Anhaus (originally from Atomic BL... from Sigma Integrated Rule Set (GitHub)
Detect creation of dump files containing the memory space of lsass.exe, which contains sensitive credentials. Identifies usage of Sysinternals procdump.exe to export the memory space of lsass.exe which contains sensitive credentials.
- ⚠️ 1 match for rule **Suspicious Use of Procdump** by Florian Roth from Sigma Integrated Rule Set (GitHub)
Detects suspicious uses of the Sysinternals Procdump utility by using a special command line parameter '-ma' and '-accepteula' in a single step. This way we're also able to catch cases in which the attacker has renamed the procdump executable.
- ⚠️ 4 matches for rule **Procdump Usage** by Florian Roth from Sigma Integrated Rule Set (GitHub)
Detects uses of the Sysinternals Procdump utility
- 🔍 1 match for rule **Usage of Sysinternals Tools** by Markus Neis from Sigma Integrated Rule Set (GitHub)
Detects the usage of Sysinternals Tools due to accepteula key being added to Registry

Getting the EVTX from VirusTotal

The first thing we do, for convenience, is to create an environment variable that contains our VirusTotal API key. We can find our API key following: <https://developers.virustotal.com/reference#getting-started>. Copy that value and go to our shell:

```
export VT_API_KEY="AAAAAAAAAAAAA"
```

And do the same for the hash

```
export HASH="3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306"
```

Search the sample on VirusTotal

Ok now let's check if the sample in question has EVTX data:

```
curl --header "x-apikey: ${VT_API_KEY}" \
```

```
https://www.virustotal.com/api/v3/files/\${HASH}/behaviours > ${HASH}.json
```

Check if EVTX is available for the sample

In the response the "has_evtx" boolean needs to be checked using jq:

```
$ cat ${HASH}.json | jq '.data[] | select(.attributes.has_evtx) | .links'
```

```
{
  "self":
  "https://www.virustotal.com/api/v3/file_behaviours/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306_VirusTotalZenBox"
}
```

Download and extract the EVTX

Fetch the EVTX ZIP file containing all EVTX files from the sandbox executions of that sample.

```
curl --location -v --request GET --url "https://www.virustotal.com/api/v3/file_behaviours/${HASH}_VirusTotal%20ZenBox/evtx" --header "x-apikey: ${VT_API_KEY}" -o ${HASH}_evtx.zip
```

Now we have a zip that we can unpack, but first let's move the file in a sub directory..

```
mkdir $HASH
```

```
mv $HASH_evtx.zip $HASH
```

```
Unzip
```

```
cd $HASH
```

```
unzip $HASH_evtx.zip
```

Process EVTX with Plaso

Now we parse the EVTX file with Plaso (of course adjust the directory of our folder):

```
username@host:~/dev/vt_evtx/${HASH}$ docker run -v ~/dev/./data log2timeline/plaso log2timeline --storage_file /data/vt_evtx/${HASH}.plaso /data/vt_evtx/${HASH}/
```

That will generate a .plaso file with an output similar to (output is trimmed):

```
plaso - log2timeline version 20211024
```

```
Source path : /data/vt_evtx/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306
```

```
Source type : directory
```

```
Processing time : 00:00:06
```

Tasks:	Queued	Processing	Merging	Abandoned	Total
	0	0	0	313	

Identifier	PID	Status	Memory	Sources	Events	File
Main	7	completed	152.4 MiB	313 (0)	14313 (0)	

```
[...]
```

```
Processing completed.
```

Now if we do the following to check the number of events and more metadata:

```
docker run -v ~/dev/vt_evtx/./data log2timeline/plaso pinfo /data/${HASH}.plaso
```

This will give us some indication of how many events have been parsed, if the number is very low, there might be issues:

```
***** Plaso Storage Information *****
```

```
Filename :
```

```
3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.plaso
```

```
Format version : 20210514
```

```
Storage type : session
```

```
Serialization format : json
```

```
***** Sessions *****
```

```
8a903a57-6e36-45dd-b03a-8e38359e3822 : 2021-10-22T14:21:38.407377Z
```

```
c7847b56-3dd9-413f-8598-05183c22866b : 2021-10-22T14:23:05.795549Z
```

```
***** Event sources *****
```

```
Total : 315
```

```
***** Events generated per parser *****
```

Parser (plugin) name : Number of events

filestat : 945

winevtx : 13374

Total : 14319

Ingest .plaso file to Timesketch

We want to ingest the Plaso file in [Timesketch](#), an open source DFIR tool to collaborate on timelines.

We want to import the file with the Notebook shipped with Timesketch, as explained here:

<https://github.com/google/timesketch/blob/master/docs/learn/notebook.md>

Visit:

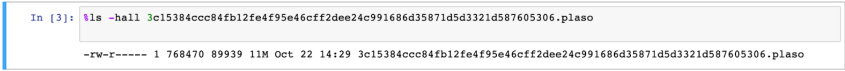
<http://localhost:8844/?token=timesketch>

Before we continue, move the Plaso file that was just created to the tmp directory because the dev notebook only has read access to that directory.

```
cp ${HASH}.plaso /tmp/
```

We can run the following in our notebook and see if our find the file:

```
%ls -hall ${HASH}.plaso
```



```
In [3]: %ls -hall 3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.plaso
-rw-r----- 1 768470 89939 11M Oct 22 14:29 3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.plaso
```

```
-rw-r----- 1 768470 89939 11M Oct 22 14:29 3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.plaso
```

We might need to add read permissions to the Plaso file so the notebook is able to read it:

```
chmod a+r /tmp/${HASH}.plaso
```

The following code makes use of the Timesketch [import_client](#) and will upload the file to our sketch (make sure the [celery_workers](#) are running)

```
from timesketch_api_client import config
```

```
from timesketch_import_client import importer
```

```
ts = config.get_client()
```

```
my_sketch = ts.get_sketch(2)
```

```
with importer.ImportStreamer() as streamer:
```

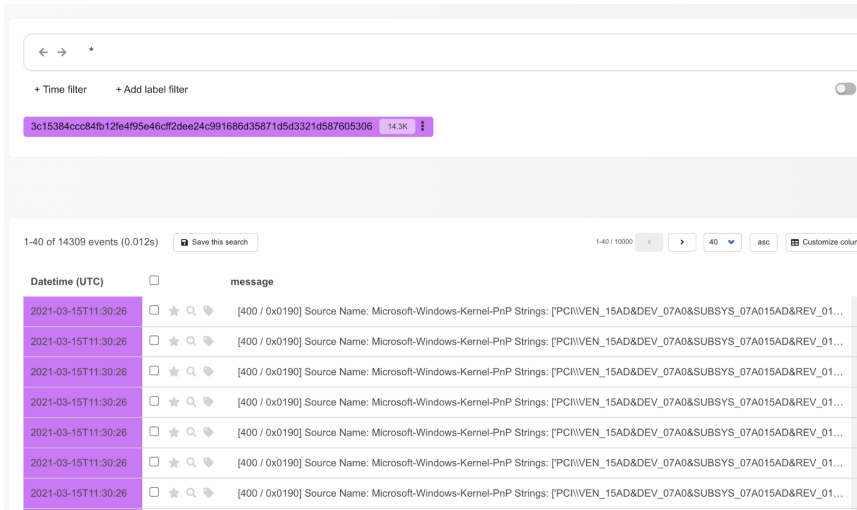
```
    streamer.set_sketch(my_sketch)
```

```
    streamer.set_timeline_name('3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306')
```

```
    streamer.set_timestamp_description('VT_EVTX_3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306')
```

```
    streamer.add_file('3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.plaso')
```

Which will show up in Timesketch like following



We now have the timeline in Timesketch. We'll leverage the Sigma module of Timesketch to write, test and adjust our Sigma rules against EventLogs that are generated when the sample that was run in the VirusTotal sandbox.

To read more how Sigma in Timesketch works, have a look at: <https://osdfir.blogspot.com/2020/11/sigma-in-timesketch-lets-rule-sketch.html>

Automate the process

To not have to manually repeat all the steps previously mentioned a [DFTimewolf](#) recipe called `vt_evtx` is available to automate this. DFTimewolf is a free and open source tool that has integrations (among others) with Plaso and Timesketch.

The recipe is invoked like this:

```
dfimewolf vt_evtx e2a24ab94f865caeacdf2c3ad015f31f23008ac6db8312c2cbfb32e4a5466ea2 /var/tmp/
```

That command will try to download the EVTX files for the hash. If we pass a SHA-256 hashes to the tool that are unknown to VirusTotal or do not have an EVTX file to download, it will tell us. The only thing we need to add if we do not want to provide our API key in the command line argument is (or add `-vt_api_key ${VT_API_KEY}` in your `dfimewolf` command):

```
cat ~/.dfimewolfrc
```

```
{
  "vt_api_key": "${VT_API_KEY}"
}
```

```
[2021-10-19 12:04:40,681] [dfimewolf ] DEBUG Logging to stdout and /tmp/dfimewolf.log
[2021-10-19 12:04:40,682] [dfimewolf ] SUCCESS Success!
[2021-10-19 12:04:40,682] [dfimewolf ] DEBUG Recipe data path: ~/dev/dfimewolf/data
[2021-10-19 12:04:40,682] [dfimewolf ] DEBUG Configuration loaded from: ~/dev/dfimewolf/data/config.json
[2021-10-19 12:04:40,682] [dfimewolf ] DEBUG Configuration loaded from: ~/.dfimewolfrc
[2021-10-19 12:04:40,704] [dfimewolf ] INFO Loading recipe vt_evtx...
[2021-10-19 12:04:40,704] [dfimewolf ] DEBUG Loading module VTCollector from dfimewolf.lib.collectors.virustotal
[2021-10-19 12:04:40,842] [dfimewolf ] DEBUG Loading module LocalPlasoProcessor from dfimewolf.lib.processors.localplaso
[2021-10-19 12:04:40,844] [dfimewolf ] INFO Loaded recipe vt_evtx with 2 modules
[2021-10-19 12:04:40,844] [dfimewolf ] DEBUG VTCollector:
[2021-10-19 12:04:40,844] [dfimewolf ] DEBUG hashes
'3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306,aaa,bbbb'
[2021-10-19 12:04:40,845] [dfimewolf ] DEBUG vt_api_key '${VT_API_KEY}'
```

```

[2021-10-19 12:04:40,845] [dfetimewolf ] DEBUG vt_type 'evtx'
[2021-10-19 12:04:40,845] [dfetimewolf ] DEBUG directory '/var/tmp/output22'
[2021-10-19 12:04:40,845] [dfetimewolf ] DEBUG LocalPlasoProcessor:
[2021-10-19 12:04:40,845] [dfetimewolf ] DEBUG timezone None
[2021-10-19 12:04:40,845] [dfetimewolf ] DEBUG
[2021-10-19 12:04:40,845] [dfetimewolf ] INFO Running preflights...
[2021-10-19 12:04:40,845] [dfetimewolf ] INFO Setting up modules...
[2021-10-19 12:04:40,845] [dfetimewolf ] INFO Setting up module: VTCollector
[2021-10-19 12:04:40,846] [dfetimewolf ] INFO Setting up module: LocalPlasoProcessor
[2021-10-19 12:04:40,847] [dfetimewolf ] INFO Modules successfully set up!
[2021-10-19 12:04:40,847] [dfetimewolf ] INFO Running modules...
[2021-10-19 12:04:40,847] [dfetimewolf ] INFO Running module: VTCollector
[2021-10-19 12:04:40,848] [VTCollector ] DEBUG Trying to find
3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306 on VirusTotal...
[2021-10-19 12:04:42,598] [VTCollector ] INFO Found the following files on VT:
['3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306']
[2021-10-19 12:04:43,179] [VTCollector ] INFO evtx for 3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306:
https%3A/www.virustotal.com/api/v3/file_behaviours/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306_VirusTotal%20;
[2021-10-19 12:04:43,179] [VTCollector ] INFO Download link
https%3A/www.virustotal.com/api/v3/file_behaviours/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306_VirusTotal%20;
[2021-10-19 12:04:44,413] [VTCollector ] DEBUG
/var/tmp/output22/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.evtx file extracted to
/var/tmp/output22/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.evtx_extract
[2021-10-19 12:04:44,414] [VTCollector ] INFO Finished writing EVTX to
/var/tmp/output22/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.evtx_extract
[2021-10-19 12:04:44,414] [dfetimewolf ] INFO Module VTCollector finished execution
[2021-10-19 12:04:44,414] [dfetimewolf ] INFO Running module: LocalPlasoProcessor
[2021-10-19 12:04:44,415] [LocalPlasoProcessor ] INFO Log file: /tmp/tmptgy5m6jn/plaso.log
[2021-10-19 12:04:44,415] [LocalPlasoProcessor ] INFO Running external command: "/usr/bin/log2timeline.py -q --status_view none --
partition all --logfile /tmp/tmptgy5m6jn/plaso.log --storage_file /tmp/tmptgy5m6jn/826d057c4abe4c0e9a7aa4d6bf886063.plaso
/var/tmp/output22/3c15384ccc84fb12fe4f95e46cff2dee24c991686d35871d5d3321d587605306.evtx_extract"
[2021-10-19 12:05:14,771] [dfetimewolf ] INFO Module LocalPlasoProcessor finished execution
[2021-10-19 12:05:14,772] [dfetimewolf ] INFO Recipe vt_evtx executed successfully!

```

After that execution, we have the final Plaso file:

```
/tmp/tmptgy5m6jn/826d057c4abe4c0e9a7aa4d6bf886063.plaso
```

Which can be added to Timesketch and all other compatible tools very easily. To make it even easier, there is also a recipe called "[vt_evtx_ts](#)" which will push the Plaso file directly to Timesketch.

Exploring the data in Timesketch

In Timesketch we can now start to explore the sketch and find interesting events e.g. by searching for

```
*accepteula* AND *-ma* AND *lsass.exe*
```

🔍	message	[1 / 0x0001] Source Name: Microsoft-Windows-Sysmon Strings: [*, 2021-10-11 13:24:00.589', '[C784477D-3AF0-6164-6306-000000003800]', '6328', 'C:\\Users\\george\\Desktop\\procdump.exe', '5,00', 'Sysinternals process dump utility', 'ProcDump', 'Sysinternals - www.sysinternals.com', 'procdump', 'C:\\Users\\george\\Desktop\\procdump.exe -accepteula -ma lsass.exe C:\\Users\\george\\Desktop\\1.dmp', 'C:\\Users\\george\\Desktop\\', 'DESKTOP-B0T93D6\\george', '[C784477D-B139-6141-075D-030000000000]', '0x00000000000035d07', '1', 'High',
---	---------	--

The event matching this search term indicates execution of procdump to copy memory of lsass. This is a technique used by attackers to retrieve passwords.

In part two of this blog series, we will look into ways to use a VirusTotal EVTX file to test a Sigma rule and adjust Sigma config in Timesketch to make the rule work.

If you have any questions please reach out on the Open Source DFIR Slack community.

Incident Response in the Cloud

Parsing the \$MFT NTFS metadata file

Forensic Disk Copies in GCP & AWS
