Demo: Benchmarking Label Dynamics of VirusTotal Engines

Shuofei Zhu[1], Ziyi Zhang[2], Limin Yang[3], Linhai Song[1], Gang Wang[3]
[1]Pennsylvania State University
[2]University of Science and Technology of China
[3]University of Illinois at Urbana-Champaign

ABSTRACT

VirusTotal is the largest online anti-malware scanning service. It is widely used by security researchers for labeling malware data or serving as a comparison baseline. However, several important challenges of using VirusTotal are left unaddressed (*e.g.*, when VirusTotal labels are stable and can be trusted), severely harming the correctness of research projects depending on VirusTotal.

In this paper, we present VTSet, which contains daily VirusTotal labels on more than 14,000 files over one year. VTSet can be used to build and evaluate various tools to tackle the existing challenges and facilitate the future usage of VirusTotal. Besides the data, VTSet also provides a demonstration tool to display many measurement results and a query tool to ease the access of its data. A video demonstration of VTSet is located at the following link: https://youtu.be/aSVaUGHxFi4.

ACM Reference Format:
Shuofei Zhu[1], Ziyi Zhang[2], Limin Yang[3], Linhai Song[1], Gang Wang[3]. 2020. Demo: Benchmarking Label Dynamics of VirusTotal Engines. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3372297.3420013

1 INTRODUCTION

As the largest online anti-malware scanning service, VirusTotal applies more than 70 state-of-the-art anti-malware engines to analyze user-submitted files and returns back engines' detection results (*e.g.*, whether a file is malicious). VirusTotal has been heavily used by industrial practitioners to identify false positives and false negatives in their products, and researchers in the security community for malware data annotation [2, 3].

In our previous work [9], we collected 115 top-tier conference papers that used VirusTotal to label their evaluation datasets. We studied how the researchers used VirusTotal and identified several common methodologies. For example, since different VirusTotal engines often disagree with each other, researchers often used a threshold-based method to aggregate labels from different engines, and considered a file is malicious, if a threshold is met in terms of the number of engines that identify the file as malicious. In addition, most researchers submitted their samples to VirusTotal only once without considering possible label changes on VirusTotal, largely due to their limited VirusTotal query quota.

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11.
https://doi.org/10.1145/3372297.3420013

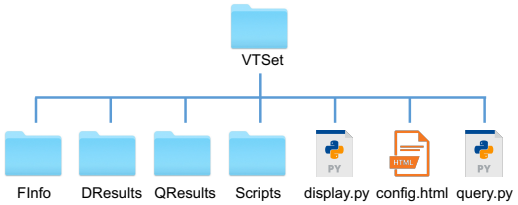


Figure 1: The directory structure of VTSet.

We also conducted a measurement to validate or justify researchers' VirusTotal usage [9]. We collected daily VirusTotal labels on 14,423 PE files without prior histories on VirusTotal for more than one year. Our measurement on the data identified some engines whose labels are very similar to each other and engines whose detection decisions are highly influenced by other engines. These engines may cause problems for the threshold-based label aggregation widely used by researchers. Our measurement also revealed label flips are very common on VirusTotal, and a label aggregation result can be changed after submitting the same file to VirusTotal for the second time.

Although we identified questionable approaches of using VirusTotal and provided several alternative suggestions [9], there are still many key questions unanswered. First, given a file's VirusTotal labels, how can we know whether the labels are already stable and will not change anymore? If a file's VirusTotal labels are already stable, they can be used to infer the true label of the file (using the threshold-based label aggregation). Otherwise, they cannot be trusted. Second, since users may not be able to wait for a file's VirusTotal labels to become stable, can we predict the file's stable labels using its current VirusTotal labels? Third, can we predict whether a file will eventually be detected by a given number of engines for the threshold-based aggregation method? In the end, how can we exclude the impact of engines highly influenced by others when aggregating VirusTotal labels? The answers to these questions can significantly improve the future usage of VirusTotal.

The data we collected in our previous work contain fine-grained VirusTotal labels on a large set of randomly sampled PE files over a long period of time. Naturally, it can provide training data, ground-truth labels, and features for various VirusTotal label prediction tasks, which can answer the key questions discussed above. Therefore, we transform our collected data into a dataset, VTSet, and release it to encourage future researchers to explore how to predict label changes on VirusTotal.

In this paper, we present VTSet, which is built by conducting a daily measurement on more than 14,000 files over a year and can serve as a benchmark set for many label prediction tasks. Besides the data, VTSet also contains an interactive tool, which shows the measurement results in [9] and helps users better use VirusTotal (*e.g.*, selecting suitable anti-malware engines). Since analyzing the whole dataset takes a long time, VTSet also provides a query tool

to ease the data access. The difference between this paper and our previous paper [9] is that our previous work focuses on how to collect the data and how to conduct the measurement, but this paper describes how the data is organized and how it enables future research opportunities.

Our contributions are summarized as the following.

- We organize a large set of VirusTotal labels into VTSet and build tools to access it.
- We identify several prediction tasks that can be conducted and evaluated using VTSet.

VTSet can be downloaded using the following link: https://drive.google.com/file/d/1jSCPeEM3pHFkceSjQyngEuVlWFVbzjFb.

2 DATASET DETAILS

This section presents VTSet in detail. As shown in Figure 1, VTSet contains three components: the collected VirusTotal data, an interactive tool ("`display.py`") to display measurement results in [9], and a query tool ("`query.py`") to select a part of the labeling data for further analysis.

2.1 Directory Structure and Data Format

There are four sub-directories in VTSet. We explain their content and related file formats as follows.

`FInfo`. For each of the 14,423 monitored files, `FInfo` contains a JSON file with the file information returned by VirusTotal. The file information can be roughly divided into hash values (*e.g.*, SHA256), metadata (*e.g.*, file size), and information extracted by relatively deep analysis (*e.g.*, imported DLLs).

`DResults`. `DResults` contains 378 JSON files, each of which is named as a date in the data collection window and contains VirusTotal engines' detection results on that day. Each JSON file indexes the 14,423 monitored files using their SHA256 hash values, and then indexes engines' detection results using engines' names.

`QResults`. VTSet allows its users to select a portion of the labeling data using the query tool. All selected data will be saved under a sub-directory of `QResults`.

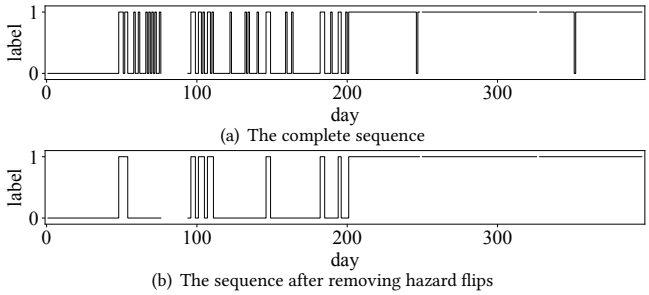
The preprocessed data used in [9] is placed under sub-directory `QResults/usenix`, and it can serve as an illustration for the format of files under `QResults`. Since we focus on how engines change their labels (*e.g.*, from malicious to benign) in [9], we create a separated file for an engine's detection results. Each file has 14,423 lines representing an engine's results on the 14,423 files. Each line has two components, one is a file's SHA256 hash, and the other is a label sequence containing 396 numbers with "0" representing benign, "1" representing malicious, and "2" representing an unavailable result. Figure 2(a) shows the visualized label sequence when AegisLab analyzes a file throughout the data collection window.

`Scripts`. All scripts that support the functionality of `display.py` and `query.py` are placed under directory `Scripts`.

2.2 Displaying Measurement Results

VTSet provides an interactive tool `display.py` to exhibit measurement results discussed in [9]. `display.py` can be executed in a terminal using a command of the following form.

```
python3 display.py <dataset> <options>
```



(a) The complete sequence

(b) The sequence after removing hazard flips

Figure 2: An example file's label sequence from AegisLab.

`<dataset>` specifies a sub-directory under `QResult` containing a piece of preprocessed data generated by `query.py`, *e.g.*, `usenix`. `<options>` can be any of the following alternatives:

1) `-cf` or `--count-flips`: count the number of flips. A flip refers to a change between two consecutive labels, *i.e.*, "malicious" → "benign" or "benign" → "malicious". We use "0" to represent a benign label and use "1" for a malicious label. Thus, a flip can be represented as "01" or "10". Users can count flips per file, per engine, or per week, by using `--file`, `--engine`, or `--week` as an extra parameter.

2) `-ch` or `--count-hazard-flips`: count hazard flips. Given a file, a VirusTotal engine sometimes flips its label on one day and then quickly changes it back the next day. We use "hazard" to represent this phenomenon. Specifically, a hazard is represented as "010" or "101". A hazard contains two hazard flips. We can see that the label sequence in Figure 2(a) contains many hazards. A hazard can last for multiple days. Users can count hazard flips per file, per engine, or per week, by providing the corresponding parameter.

3) `-cnh` or `--count-non-hazard-flips`: count non-hazard flips. After removing hazard flips, a label sequence only contains non-hazard flips (*e.g.*, Figure 2(b)).

4) `-csf <x>` or `--comp-stable-files <x>`: compute the percentage of files whose VirusTotal labels do not change since day-x until the end of the label sequences. Users can use `--exclude <n>` to exclude the top n engines with most flips.

5) `-cfi <t>` or `--comp-flip-impact <t>`: compute the percentage of files with only benign aggregated labels, with only malicious aggregated labels, and with both benign and malicious aggregated labels during the data collection, when considering a file as malicious if t or more than t engines detect the file as malicious. To measure the percentage of files having different aggregated labels on different days, users can estimate the impact of flips on the threshold-based aggregation method.

6) `-cs <A> <B>` or `--compute-similarity <A> <B>`: compute the average similarity of detection results between engine A and B on the same file over the whole dataset.

7) `-c <t>` or `--clustering <t>`: report the hierarchical clustering results when choosing t as the threshold. The distance between a pair of engines is computed as one minus the similarity between the two engines' detection results.

8) `-pir` or `--print-influence-ranking`: print the influence ranking. In [9], we used a graph to measure influence between VirusTotal engines. In this complete, directed graph, nodes represent engines, and weights on directed edges represent the influence score between the two engines. An engine with larger weights on its outgoing edges has more influence on other engines, while an engine with larger weights on its incoming edges is more likely to

be influenced by others. We designed an active model and a passive model to compute the weights, and users can choose between the two models by using `--active` or `--passive`. Users can also choose between "01" flips and "10" flips using `--one` or `--zero`.

9) `-vi` or `--visualize-influence`: we use Gephi [1] to visualize the computed influence graph. Similarly, users can choose between the passive model and the active model, and choose between the usage of "01" flips and the usage of "10" flips.

2.3 Querying VTSet

`query.py` is a query tool to select a part of the labeling data based on a user-provided configuration file. `config.html` is a GUI tool to facilitate users to specify their requirements and generate corresponding configuration files. `query.py` can be invoked using a command in the following format:

```
python3 query.py <config> <dataset>
```

<dataset> specifies a sub-directory name under `QResults` to save the selected data. <config> is a json file that describes what labeling data to be selected, and it is generated by `config.html`.

3 RESEARCH OPPORTUNITIES

VTSet captures VirusTotal engines' labeling behavior on a daily basis over a year. Thus, it enables many prediction tasks (by providing training data and ground-truths). We briefly discuss these tasks as follows.

First, can we predict an engine's label on a given file in the near future or in the long run? In our previous work, we identified engines whose labels that are highly correlated with each other and engines whose labeling decisions are highly influenced by other engines [9]. Therefore, it is promising to predict an engine's detection decision on a file based on other engines' detection results on the same file. Since an engine may take some time to react to other engines' behavior, we think it is more reasonable to predict an engine's label on a file after a relatively long period of time.

Second, can we predict the stability of an engine's label on a given file? We found that some engines have many more label flips than other engines, and label flips are more likely to happen on some files [9]. An interesting prediction task is to know how likely an engine would change its label on a file, or whether an engine's label has become stable on the file. If an engine's label on a file has already been stable, the label is more trustworthy, compared with a label that is predicted to be changed in the future.

Third, can we predict whether a file's VirusTotal labels can eventually become stable? In our previous measurements, we found that some files' VirusTotal labels are very difficult to become stable, and VirusTotal engines still change their detection decisions on these files even after the files have been submitted to VirusTotal for more than a year [9]. It is interesting to categorize these files' features and detect files whose VirusTotal labels remain dynamic. For these files, leveraging manual inspection to infer their true labels is more reasonable than using the anti-malware analysis on VirusTotal.

Fourth, can we predict an engine's behavior for a particular malware family? VirusTotal also provides malware family names assigned by its engines. VTSet also records this information. Intuitively, different engines have different capabilities of analyzing different malware families and an engine's behavior on a particular malware family may be different from its overall behavior when analyzing all families. Thus, we can explore how to conduct fine-grained prediction on each malware family using VTSet.

Fifth, how to predict the results of threshold-based label aggregation? Since label changes are very common on VirusTotal, it is important to predict whether a threshold can be met for the number of engines that detect a file. It is promising to leverage our observations in [9] (*e.g.*, highly influenced engines, engines with similar results) to do the prediction.

4 RELATED WORK

There are previous works on evaluating VirusTotal engines or aggregating VirusTotal labels. Peng et al. [7] inspected how VirusTotal's URL scanning engines detect phishing URLs over a month. Kantchelian et al. [5] proposed a machine learning model to aggregate VirusTotal labels. Previous researchers also tried to aggregate malware family names provided by different VirusTotal engines [4, 6, 8]. Different from the data collected by these previous works, VTSet contains daily snapshots of VirusTotal labels on a large set of randomly sampled PE files. VTSet is built through monitoring VirusTotal over one year. Users can observe fine-grained label changes or malware family changes using VTSet and explore how to predict these changes.

5 CONCLUSION

In this paper, we presented VTSet, which is the first available dataset with fine-grained recording of VirusTotal engines' labeling behavior over a long period of time. VTSet can serve as a benchmark set for many prediction tasks on how VirusTotal labels change over time. Future work can consider adding more data to VTSet and extending it to other file types.

ACKNOWLEDGEMENT

This research was supported in part by a Seed Grant award from the Institute for Computational and Data Sciences at the Pennsylvania State University.

REFERENCES

[1] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *ICWSM*, 2009.
[2] Kai Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Nan Zhang, Heqing Huang, Wei Zou, and Peng Liu. Finding unknown malice in 10 seconds: Mass vetting for new threats at the google-play scale. In *USENIX Security*, 2015.
[3] Sean Ford, Marco Cova, Christopher Kruegel, and Giovanni Vigna. Analyzing and detecting malicious flash advertisements. In *ACSAC*, 2009.
[4] Médéric Hurier, Guillermo Suarez-Tangil, Santanu Kumar Dash, Tegawendé F Bissyandé, Yves Le Traon, Jacques Klein, and Lorenzo Cavallaro. Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware. In *MSR*, 2017.
[5] Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Brad Miller, Vaishaal Shankar, Rekha Bachwani, Anthony D. Joseph, and J. D. Tygar. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *AISec*, 2015.
[6] Aziz Mohaisen and Omar Alrawi. Av-meter: An evaluation of antivirus scans and labels. In *DIMVA*, 2014.
[7] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *IMC*, 2019.
[8] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In *RAID*, 2016.
[9] Shuofei Zhu, Jianjun Shi, Limin Yang, Boqin Qin, Ziyi Zhang, Linhai Song, and Gang Wang. Measuring and modeling the label dynamics of online anti-malware engines. In *USENIX Security '20*, Boston, MA, 2020.