Understanding Malwares and Anti-Virus Engines in the Real World

paper 127 in 13 pages

Abstract

In this paper, we conducted a thorough empirical study on real-world malwares and anti-virus engines by examining the largest real malware repository, VirusTotal. Our study is performed in three dimensions. First, we study the correlation between malwares metadata and their detection rates. Second, we model how influence propagates across different anti-virus vendors. Finally, we explore the feasibility to build machine learning malware detectors based only on hash values of files. Our study results confirmed a set of hypothesis and anecdotal assumptions and revealed a few surprising new insights into malwares and anti-virus engines. Together with our machine learning framework, these results can shed light on future research directions and assist both anti-virus vendors and normal users in their fight against malwares.

1. Introduction

There is and will continue to be a constant competition between anti-virus tools and malwares. Malwares grow exponentially [1] and place an imperative threat to human society. For example, more than 390000 new malwares are registered in AVTest institute every day [1]. As another example, a new type of threat, ransomware, has caused more than 1 billion dollars this year [13]. In order to keep up and detect these new types of malwares, anti-virus tools are also improving rapidly, by constantly updating their signature database, by using more advanced techniques like deep learning [4], or by utilizing more data.

In order to improve anti-virus tools and defend against emerging threats from malwares, it is essential to understand malwares and existing anti-virus tools in the real world. Previous works on studying malwares and anti-virus engines do provide valuable insights [5, 9, 23] such as how malware writers create new malwares and how malwares escape from the detection of anti-virus engines. However, these previous works only studied limited malwares and did not provide insights into the relationship between malwares and anti-virus engines.

We propose to use the vast amount of existing "big data" to study real-world malwares and their relationship with anti-virus engines. To conduct such study, we utilized an open data repository that contains billions of real-world malwares, *VirusTotal*.

VirusTotal is a free online malware scanning service that applies a set of state-of-the-art anti-virus engines to analyze user-submitted files and sends a detection report back to user. VirusTotal provides public access to all its submitted files and analysis results. VirusTotal is a valuable resource to study and understand real-world malwares and anti-virus engines.

First, there are huge amount of suspicious files submitted to VirusTotal. For example, within our data collection window, there were around 40 million submissions to VirusTotal each month. These submissions cover a large variety of file types, and are conducted by a large variety of VirusTotal users from all over the world. This amount of diverse data on VirusTotal serves as a good representation of malwares in the real world.

Second, for almost all submissions, VirusTotal applies no less than 50 state-of-the-art anti-virus engines to analyze them. VirusTotal keeps detailed detection results and provides an open access to these results. Analyzing historical detection results can help capture how anti-virus engines evolve over time.

Third, VirusTotal provides rich metadata for each submission. Besides the detection results of various anti-virus engines, VirusTotal also provides file type information, submitter information, and a hash string of the original submitted file. These types of information is valuable to study

Unfortunately, there has only been limited attention to VirusTotal in the past. Researchers tried to capture malware writers who leverage VirusTotal as the testing platform during malware development [7, 10]. Anti-virus vendors use VirusTotal to detect possible false positives and false negatives in their products. But none of them study the rich malware data provided by VirusTotal in a large scale.

We collected 4 months of metadata of all submissions to VirusTotal and conducted an extensive study on them. After collecting and pre-processing data from VirusTotal, we first perform a set of basic analysis to gain an overall knowledge of the VirusTotal data repository, including how big the files submitted to VirusTotal are, who submit to VirusTotal, and how many submissions are labeled as malware.

On top of these basic findings, we developed a set of more advanced studies in three directions.

First, we study the correlation between submissions metadata and their *detection rates*, the percentage of engines labeling a submission as malware. We found high correlation between detection rate and three factors: submission file size, the history of submitting a file, and the reputation of the submitters. These results shed light on what types of submission are more likely to be malware. With this result, future researchers and anti-virus vendors can have more guided direction into what they should have further investigation.

Second, we study the question of whether or not different anti-virus vendors can influence each other and if detection rate is a perfect measurement of the likelihood of a file being a true malware. Anecdotally, anti-virus vendors frequently

leverage VirusTotal to identify false negatives in their products, which are malwares detected by others products but not detected by their own products. To verify this hypothesis, we use the detection history from VirusTotal and developed statistical models based on a known technique from from the web mining area to quantify the influence between vendors. With this method, we confirmed that there do exist high influence between vendors; certain vendors are highly influenced by the detection results of other vendors and use this information to change their detection results. This result alerts VirusTotal users and researchers to be more careful about the detection results from vendors and use detection rate more cautiously.

Third, we explore the feasibility of building malware classifiers or detectors based only on hash values of files. We obtained hash strings of submitted files to VirusTotal and use the similarity of these hash string to build classifiers. We design several machine learning experiments to evaluate our hash-based malware classifiers. Surprisingly, our evaluation results show that by just using hash string, we can obtain high accuracy for certain classification tasks. At the same time, other classification tasks are not as accurate. We further developed a mechanism to predict which classification task is likely to have high accuracy. With these classification and detection mechanisms and the large amount of data, it will be possible to just use a compressed representation of files instead of the original files to detect malware. This result implies that future users can have higher privacy and do not need to reveal their files to know whether or not they are malware.

Our study advances the understanding of malwares and anti-virus engines in the real world and provides various valuable insights for future researchers and vendors.

2. Data Collection and Basic Analysis

This section introduces VirusTotal and discusses how we collect data from VirusTotal and pre-process them. We also present the analysis results of their basic properties, before delving into more advanced analysis in later sections.

2.1 VirusTotal

VirusTotal is a free online malware scan service. It was founded in 2004 and was acquired by google in 2012. VirusTotal is widely used by both normal users and anti-virus vendors. Normal users submit suspicious files to VirusTotal when they do not have any anti-virus software, or when they want to check for viruses possibly missed by their own anti-virus software or false alarms from their own anti-virus software. Anti-virus vendors use VirusTotal to verify false positives and false negatives in their products [7, 10].

For each submission, VirusTotal applies a set of anti-virus engines to analyze it. VirusTotal keeps information about whether the submission is labeled as malware by each engine, and detailed tags for identified malware from each engine.

Metadata Field	Explanation
name	submitted file name
link	where to download the file
timestamp	timestamp when the submission was made
source_country	the country where the submission was made
source_id	user ID making the submission
size	file size
type	file type
tags	labels with more specific information for each **type**
first_seen	when the same file was first submitted
last_seen	when the same file was last submitted
hashes	sha1, sha256, md5, and vhash
ssdeep	ssdeep information digest string
total	number of engines analyzing the file
positives	number of engines that flagged the file as malicious
positives_delta	changes in **positives** across different submissions
report	detailed detection report from each AV engine

Table 1: VirusTotal Metadata. ***(Fields for each submission retrieved from the VirusTotal distribution API and their relation explanation. The same file can be submitted multiple times by different users.)***

VirusTotal provides open APIs to access and download both the metadata of all submissions and detection results. One API, named distribution API, works like a pipe. After a user opens the distribution API and starts to download data from VirusTotal, VirusTotal will keep returning metadata for latest submission to the user.

VirusTotal provides rich metadata. Table 1 shows the metadata fields and their meaning. The original submitted files from VirusTotal can be downloaded using the **link** field in the metadata.

2.2 Data Collection and Preprocessing

We collected all metadata for all submissions to VirusTotal from May 7th, 2016 to September 6th, 2016, with a total of 151 million submissions. Our collected data is larger than or comparable to previous works on studying VirusTotal [10, 16]. According to Lastline Labs [17], common lag time for an anti-virus engine to detect a new malware is two weeks. Thus, four months' data is more than enough to analyze most malware and anti-virus engine behavior. In deed, we drew several meaningful conclusions with this dataset, as will be presented in the rest of the paper.

We performed our data collection using VirusTotals distribution API. We insert all collected metadata into a table in Cassandra [2] using the combination of sha256, source_id, and timestamp as the key. We then used Spark [21] and wrote Spark programs to efficiently analyze the vast amount of data. All our analysis is conducted by using Spark 1.4.0 on a cluster with 19 nodes, 266 cores, and 560 GB memory.

Figure 1 shows the file type distribution for all submissions. Among all file types, Windows *Portable Executable* (*PE*) files, or "Win32EXE" and "Win32DLL" files, are the most frequently submitted type, accounting for 51% of all submissions. Web pages and Android apps account for the second and third largest submissions, with 12% and 8% of

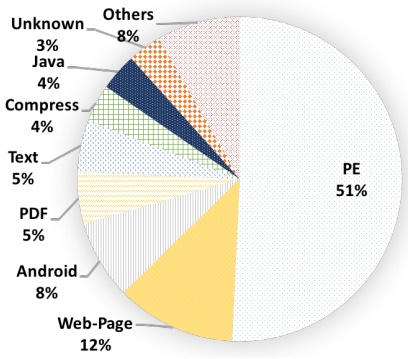


Figure 1: File types for all submissions. ***(File types and their distributions for all VirusTotal submissions from 05/07/2016 to 09/06/2016.)***

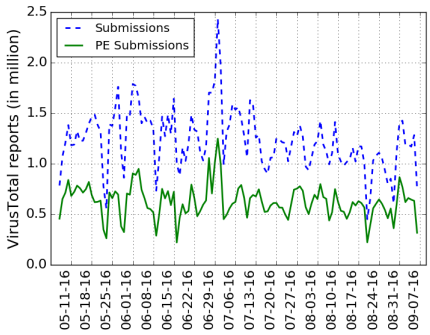


Figure 2: The number of files and PE files. ***(The number of suspicious files and the number of PE files submitted to VirusTotal from 05/07/2016 to 09/06/2016.)***

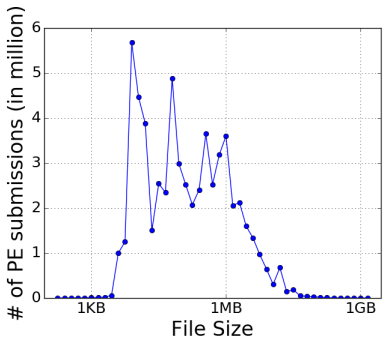


Figure 3: File size distribution for PE submissions. ***(How file size distributes among all PE submissions we collect. Results from log2 are rounded up to nearest 0.5.)***

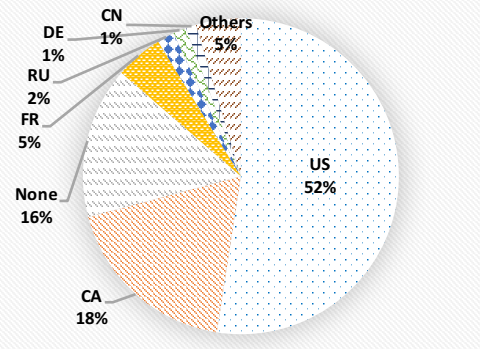


Figure 4: How PE submissions distribute among different countries. ***(Only countries with more than 1% PE submissions are shown.)***

all submissions respectively. Other popular file types include PDF, Text, compressed files, and Java files. This result shows that even though new types of malware such as Android apps have increased significantly in recent years, traditional types of malwares such as PE files and web pages are still the most commonly targeted by attackers.

Since PE files are the most common type, we focus our study in this paper on PE files and leave studies on other types of malwares for future. In total, we collected 76 million PE submissions.

2.3 Basic Analysis

After collecting data, we first conducted a set of analysis to learn various basic properties of submission files, submission sources, and anti-virus engines. These properties give an overview of how real-world submissions and anti-virus engines are like and serve as the foundation for our more advanced analysis in the next two sections.

Submission temporal, geo-location, and size distribution. Figure 2 plots the number of submissions of all types and the number of PE submissions per day over the whole collection period. There are a large amount of submissions every day and the amount of submissions is fairly stable over the whole collection period. The same conclusion can be made to PE submissions.

One PE file could be submitted more than once to VirusTotal. On average, each PE file was submitted 1.19 times and 6.72% PE files were submitted more than once.

For around 16% of PE submission, VirusTotal fails to provide their source country information. All other submissions are conducted from 221 countries. The top 6 countries in PE submission number are US, Canada, France, Russia, Germany, and China, as shown in Figure 4.

Figure 3 shows the file size distribution for PE submissions. The smallest PE file is only 187 bytes, and the largest one is more than 1 GB. 99.58% of PE file fall into the range from 4 KB to 32 MB.

Observation 1: *VirusTotal constantly receives large amount of submissions from all over the world. Most submitted files have small to middle sizes and are only submitted once, but some files are submitted many times.*

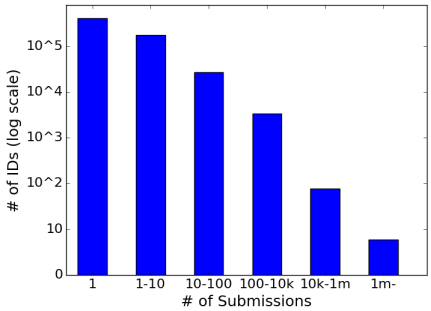


Figure 5: Source ID Category. ***(We categorize source IDs according to the number of submissions they make. Y axis represents the number of IDs in log scale that have submission numbers in a category.)***

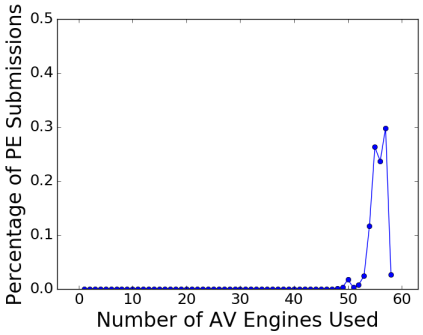


Figure 6: The distribution for the number of used anti-virus engines. ***(How the number of used anti-virus engines distributes among all PE submissions we collect.)***

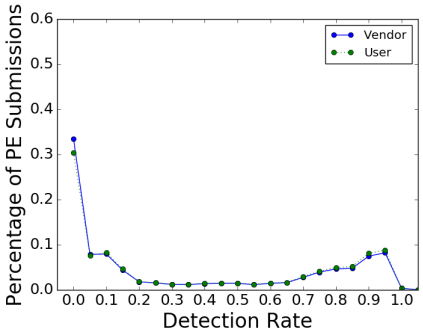


Figure 7: The distribution for detection rate. ***(How detection rate distributes among all PE submissions we collect. Each detection rate is rounded up to nearest 0.05.)***

Submission sources. Both normal users and anti-virus vendors use VirusTotal to detect malwares or evaluate products. Usually, vendors tend to make more submissions than normal users, since they need to test their products on many files. Thus, we use the number of submissions to measure what type of users a source ID is.

We separate the number of submissions into six categories: one submission, one to ten submissions, ten to 100 submissions, 100 to 10000 submissions, 10000 to 1 million submissions, and larger than 1 million submissions. Figure 5 plots the number of source IDs in log scale in the first five categories. There are only 6 source IDs that have more than 1 million submissions and we suspect that these IDs are either bogus or robots that constantly make submissions to VirusTotal. For the rest, we find that the number of source IDs dramatically drops when the number of submissions is more than 100. Thus, we categorize the source IDs as normal user if they have less than 100 submissions and as vendors if they have more than 100 submissions.

Observation 2: *Most VirusTotal users are normal users that make occasional submissions, while a small set of anti-virus vendors use VirusTotal and make a huge amount of submissions.*

Anti-virus engines and detection basic analysis. After learning the basic properties of submissions and submission sources, we now move to the basic analysis of anti-virus engines and their detection of malware. Figure 6 shows the distribution of the number of anti-virus engines used by VirusTotal on submissions. More than 99% of PE submissions are analyzed by at least 50 anti-virus engines. We suspect that VirusTotal sets some threshold when running anti-virus engines and will abort an engine when it takes too long to run on a submission, causing few submissions having fewer than 50 engines.

For the same submission, different anti-virus engines can make different detection results, *i.e.*, some mark the submission as malware while others don't. To quantify the detection results of a submission, we use a value we call *Detection Rate*. Detection rate represents the percentage of engines labeling the submission as malware.

$$Detection\ Rate = \frac{positives}{total + 1}$$

We add one to **total** to avoid divide-by-zero exception, and more importantly, to have higher detection rate to submissions labeled as malwares by more engines. For example, submissions analyzed by 50 engines and detected by 50 engines has a higher detection rate than submissions analyzed by one engine and detected by one engine.

Figure 7 shows the distribution of detection rates over all PE submissions. We separately consider submissions made by normal users and by anti-virus vendors (as defined above). Interestingly, overall most submissions to VirusTotal are likely to be benign files, *i.e.*, has low detection rate. Around half of the submissions were labeled as benign by all engines (a detection rate of zero), and only 31.5% submissions were labeled as malware by at least half of the engines. Surprisingly, the detection rates of submissions made by normal user and by anti-virus vendors are very similar. One

probable explanation is that anti-virus vendors obtain their suspicious files from end users and thus have similar behavior as normal users' submissions. Another possible reason is that as suspicious files are announced publicly, both normal users and vendors will try to test them with VirusTotal.

Observation 3: *Most submissions to VirusTotal are labeled as benign and submissions made by normal user and by anti-virus vendors have similar probabilities of being labeled as malware.*

2.4 Caveats

Like all other empirical study works, our findings and conclusions need to be considered with our methodology in mind. We use the distribution API provided by VirusTotal to download submissions' metadata from VirusTotal. There is no guarantee that this API returns all submissions to VirusTotal. It could be possible that some files are submitted to VirusTotal, but are not downloaded. Although we have collected huge amount of malware information from VirusTotal, we do believe that there are malwares never submitted to VirusTotal, or submitted to VirusTotal much later than they appear in the real world. However, there are no conceivable ways to study them. We believe that the 4-month malware information we collect can serve as a representative sample for malwares in the real world.

3. Correlation between Submission Properties and Detection Rate

This section presents our study of the correlation between various properties of PE submissions and the detection rate of PE submissions. Detection rate is what most VirusTotal users use to decide whether their submissions are benign or malware and the first thing that a user of VirusTotal uses. Therefore, it is important to study what factors affects or correlates to detection rate and the results can guide security researchers and vendors to have targeted investigation over *suspicious* files, files that have the factors that we identify as highly correlated to detection rate.

We studied a range of properties and their correlation with detection rate and found that three factors have higher correlation: submission file size, historical submission properties, and the reputation of source IDs. We present these correlation study results in this section. In the next section, we will present our further analysis of what can affect the detection result of anti-virus engines and if detection rate is a perfect measurement of the likelihood of malware.

3.1 File Size

Anecdotally, PE malwares are most likely to have small to medium size. To verify this, we analyzed the relationship between submission file size and detection rate. Figure 8 presents the average detection rate and the 95% confidence interval for different file sizes. A wider confidence interval implies that the calculated average detection rate is farther

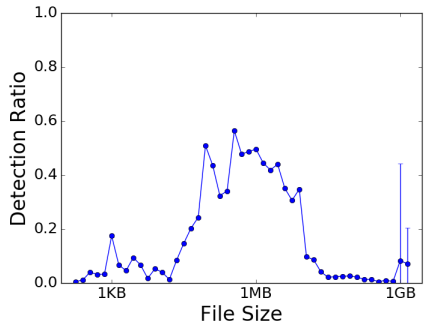


Figure 8: How detection rate changes with file size. ***(How detection rate changes with log2-based file size. 95% confidence interval is also drawn for each point. Results of log2 of the original file sizes are rounded to the nearest 0.5 value.)***

away from the real average detection rate. We use discrete file sizes of powers of two in our analysis and in this graph, i.e., we calculate the log2 of each original file size and round it to the nearest 0.5 value. Overall, we find that files with size from 90KB to 4MB have higher detection rate, more than 20% on average. Except for the last two points which represent a small amount of files that are bigger than 1GB, all other sizes have high confidence.

Observation 4: *PE malwares are mostly likely to have small to medium size.*

An immediate question to raise is whether the high detection rate of files with small to medium size is because these files also contribute to most of the submissions as shown in Figure 3. As we will discuss in Section 3.2, the correlation of submissions and detection rate is more complex and non-linear. Thus, there is a more fundamental reason behind the file size correlation with detection rate. One likely reason of this correlation is that files that are too small are not enough to express the malware functions while files that are too big are difficult to spread.

3.2 Submission History

As discussed in Section 2.3, there are files that have been submitted more than once to VirusTotal. It is worthwhile to investigate this *history of submission* and how history affects future. We study the correlation between submission history and the detection rate of the current submission. Among the different types of historical information that we study, we find that two types have higher correlation to detection rate: the number of submissions made in history and the number of source IDs that made submissions of the same file in history.

To study the submission history, we first sort all submissions for each file chronologically and then collect the number of all submissions made before submission s for each submission s in VirusTotal. Next, we calculate the correlation between detection rate and the number of submission

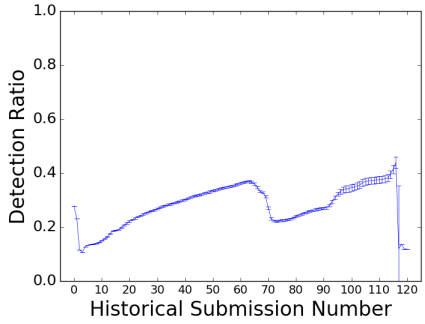


Figure 9: The relation between historical submission number and detection rate. ***(How detection rate changes with historical submission number. Each historical number is rounded up to nearest 5. 95% confidence interval is also drawn for each point.)***

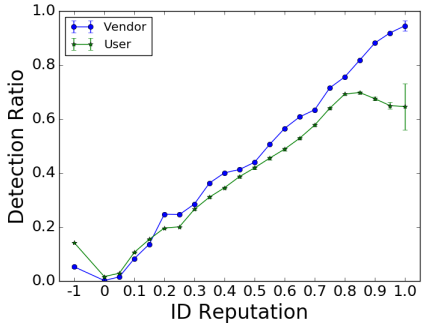


Figure 10: The relation between source id's previous reputation and detection rate. ***(How detection rate changes with the value of source id's reputation. Each reputation is rounded up to nearest 0.05. Reputation -1 means the source id did not make any submission before. 95% confidence interval is also drawn for each point.)***

in history. Figure 9 plots how detection rates change over the number of historical submissions. Interestingly, there are two main ranges that steadily increase and there are three drops in the beginning, middle, and end.

To explain this effect, we first look at the two factors that can both influence detection rates: the percentage of benign files submitted and the amount of anti-virus engines labeling the submission as malware. Obviously, with more benign files submitted, detection rate will decrease and with more engines labeling malwares, detection rate will increase. In the the two stages of detection rate increasing, with more submissions, more engines are able to identify malwares and the increase of percentage of engines identifying submitted malwares dominates. In the range that detection rate drops, VirusTotal users stop submitting files that have already been identified as malwares, and the increase of percentage of benign files submitted to VirusTotal dominates.

Observation 5: *The number of submissions made in history has correlation with detection rate, but the correlation is non-linear and is affected by multiple factors.*

3.3 Reputation of Source ID

Different users of online services such as VirusTotal often have different *reputation* because of different motivations, objectives, and backgrounds. For example, some users can randomly trying out VirusTotal with no specific objective and thus submit random files while other users have clear goals and motivations and thus only submit suspicious files. The rationale behind reputation is that VirusTotal users would have a roughly constant submission pattern, and it is promising to use their historical submission to predict their future submission. Previous work [8] reported that there is a correlation between bug reporters reputation and the likelihood for the bug being fixed. We also observe correlation between the reputation of source ID and submissions detection rate.

Intuitively, a user that have submitted more files that were detected as malware suggest that she is more likely to use VirusTotal to detect malwares in suspicious files and thus should be given higher reputation. To quantify this, we define the reputation of a source ID as the follow. The reputation of a source ID is not a fixed value and can change over time.

DEFINITION 3.1. *Reputation: Given a submission S with source ID N, we define the reputation of N when conducting the submission S as the average detection rate for all submissions conducted by N before S. If N did not make any submission before, we define the reputation to be* -1.

Before conducting our source ID reputation analysis, we first filter out submissions without source ID information (for a small amount of submissions, VirusTotal fails to provide source ID), files that are only submitted once, and files that are submitted more than 1 million times (bogus or robots). To calculate source ID reputation, we first sort all submissions from the same source ID chronologically, and calculate reputation for each source id when conducting each submission. We further separate normal users from vendors and analyze the correlation of their ID reputation and detection rate.

Figure 10 plots the average detection rate and the 95% confidence interval as the source ID reputation increases for normal users and for anti-virus vendors. We round up reputation values to their nearest 0.05 values.

Detection rate steadily increase as the reputation increases from 0 to 1 for vendors and from 0 to 0.8 for normal users. Except the points when reputation is 1, all other results have high confidence. Interestingly, for both normal user and vendors the detection rate with reputation -1 is higher than with reputation 0, which means that the first submissions conducted by all source IDs (reputation -1) are overall have

higher detection rate than the submissions conducted by IDs that have only submitted benign files (reputation 0).

For normal users, the detection rate drops when reputation is greater than 0.8. This means that for normal users, it is difficult for them to always submit suspicious files detected by almost all vendors. Vendors do not have this behavior and their detection rate always increases with higher reputation (other than reputation -1).

Observation 6: *There is a high correlation between user reputation and detection rate of their submissions for both normal users and anti-virus vendors.*

3.4 Discussion

From our analysis, file size, number of submissions in history, and source ID reputation are all highly correlated with detection rate. These results suggest that future researchers and vendors can use these values to have an initial estimation or prediction of future submissions. Doing so can help reduce researchers' and vendors' effort to focus on more interesting files—files that are more likely to be malicious. Anti-virus vendors can also inspect results which are variant from our studying results to identify possible false positives or false negatives in their products.

4. Influence Propagation

As discussed in Section 2, many files are submitted to VirusTotal more than once, and more than 99% submissions are analyzed by at least 50 anti-virus engines. We observe that some engines fail to identify some malwares during early submissions, but they catch up when analyzing later submissions. Anti-virus vendors frequently use VirusTotal to identify false negatives in their products, which are malwares they fail to detect but detected by other vendors. We hypothesize that there are influence among different anti-virus vendors. This section presents our verification of this hypothesis by modeling the cross-vendor influence. We also provide the prediction model of whether an engine will identify a file as malware in the future after labeling the file as benign.

4.1 Influence Model

Influence propagation on social graph is a well-studied topic in the web mining area. Borrowing this idea, we use graphs to represent the relationship between vendors and model influence among different vendors based on static models originally proposed to analyze Flickr data [6]. We now explain our graph-based model for anti-virus vendor influence.

We first connect all vendors with a complete directed graph $G = (V, E)$, where the nodes V are vendors. The graph is complete because we assume that influence from any vendor to any other vendor is possible. We define an action (a) at time t as (u, a, t) or $(u, \bar{a}, t)$, as the former represents an anti-virus engine identifies the submission as malware and the latter represents it identifies the submission as benign. Since the goal of our study is to detect if the change from labeling a file as benign to labeling it as malware by a vendor is influenced by other vendors, we do not consider the case of changing from labeling benign to malware and assume that after a vendor labels a file as malware, it will not change its decision. A similar model can be applied to study the latter type.

We associate each edge $(u, v) \in E$ in the graph G with an influence probability $p_{u,v}$, which represents the probability that after u takes an action v will follow u to take the same action. We only consider cases of changing from labeling benign to malwares, which means that when calculating $p_{u,v}$, we only consider cases where u has labeled a submission as malware, v will label a submission as malware in the future, and v labeled the submitted file as benign earlier. For an action a, we use $S_v(a)$ to represent the set of v's neighbors taking action a before v in time. The probability that v will follow its neighbors to take the same action can then be calculated as:

$$p_v(S_v(a)) = 1 - \prod_{u \in S_v(a)} (1 - p_{u,v}) \quad (1)$$

Thus, if we know the probability of a node influencing another node ($p_{u,v}$) for all node pairs, we can know the probability of a node (an anti-virus vendor) being influenced by other nodes (other anti-virus vendors). The goal of our learning model is thus to estimate $p_{u,v}$.

Specifically, during training stage, we learn $p_{u,v}$ for each edge. We designed four static models for the training stage, which differ in how to estimate $p_{u,v}$ by using the data in training set.

During prediction stage, we test a node v that has not taken the action a yet and predict if it will take this action following other nodes. Specifically, we use a tunable threshold θ and predict v will follow its neighbors to take the action a in the future if $p_v(S_v(a)) > \theta$.

Before delving into the details of how our four static models estimate $p_{u,v}$ during the training stage, we first gives a set of definitions that we make to assist the development of these models. We use A_u to represent the set of submissions identified by u as malwares in its history and $\bar{A}_u$ represents the set of submissions ever labeled as benign by u. Further, we define the set of actions taken by both u and v as $A_{u\&v}$ and the set of actions taken by either u or v as $A_{u|v}$. Thus, $|A_{u|v}| = |A_u| + |A_v| - |A_{u\&v}|$.

Finally, we define *action propagation* as follow and use A_{u2v} to represent the set of all the actions that are propagated from u to v.

DEFINITION 4.1. *Action Propagation: We say that an action a propagated from u to v iff: (i) $\exists\ (v, \bar{a}, t_i) \in \bar{A}_v$ and $(v, a, t_k) \in A_v$, with $t_i < t_k$; (ii) $\exists\ (u, a, t_j) \in A_u$, with $t_j < t_k$ and $u \neq v$.*

With these definitions, we now present our four static models.

Bernoulli distribution estimates $p_{u,v}$ as the ratio of the number of actions propagated from u to v over the total number of actions taken by u.

$$p_{u,v} = \frac{|A_{u2v}|}{|A_u|}$$

Jaccard index estimates $p_{u,v}$ as the number of actions propagated from u to v divided by the number of actions taken either by u or by v.

$$p_{u,v} = \frac{|A_{u2v}|}{|A_{u|v}|}$$

On top of the Bernoulli distribution and the Jaccord index, we add an additional consideration called *partial credit*. When v takes an action a, it may be influenced by the combination of all its neighbors $S_v(a)$ taking the action a before v. To account for this effect, we use partial credit and calculate the partial credit for u who takes an action a before v as

$$credit_{u,v}(a) = \frac{1}{|S_v(a)|}$$

Bernoulli distribution with partial credit estimates $p_{u,v}$ as the sum of all partial credits taking by u for actions propagated from u to v, dividing by the number of actions taken by u.

$$p_{u,v} = \frac{\sum_{a \in A_{u2v}} credit_{u,v}(a)}{|A_u|}$$

Jaccard index with partial credit estimates $p_{u,v}$ as the sum of all partial credits taking by u for actions propagated from u to v, dividing by the number of actions taken either by u or by v.

$$p_{u,v} = \frac{\sum_{a \in A_{u2v}} credit_{u,v}(a)}{|A_{u|v}|}$$

4.2 Model Evaluation

Training stage methodology. We split all the PE submissions we collected into a training set and a testing set based on the SHA256 values of the submitted files. We place all submissions with SHA256 values starting with a numeric character, i.e., from 0 to 9, into training set and all the rest into the testing set.

We implement the training process by several stages of map, filter, and reduce in Spark. Specifically, we first reduce submission records according to submitted files. We then sort submission records chronologically. For each remaining file, we compute four maps based on its submissions. There is one one-dimension map, containing the number of actions taken by each vendor, or the number of malwares identified by each vendor. There are three two-dimension maps, containing an entry for each pair of vendors (u,v), representing actions taken by both u and v, actions propagated from u to v, and partial credits taken by u for actions propagated from u to v. Finally, we reduce these 4 maps from different files, and calculate $p_{u,v}$ for the four static models.

After training the four static models, we filter out engines taking fewer than 10000 actions, since engines taking too few actions cannot produce meaningful results. There are 59 engines left and the average number of actions taken by these engines is around 190K. For confidentiality reasons, we omit vendors name in the following discussion and use numbers from 0 to 58 to represent these vendors.

Training results. For each static model, we put all learned $p_{u,v}$ in a influence table, by using u as row number and v as column number. For any $p_{u,v}$, Bernoulli distribution has the largest value, Jaccard index has the second largest value, Bernoulli with partial credit has the third largest value, and Jaccard index with partial credit has the least value.

We draw heat maps for all learned influence tables in Figure 11. Given a $cell(u,v)$ with u as row number and v as column number, red color means more influence from u to v. The four heat maps share a similar pattern. There are columns with almost all cells in red color, which means there are vendors influence by all other vendors. There are no rows with all cells in red color, and this means that there are no vendors which can influence all other vendors.

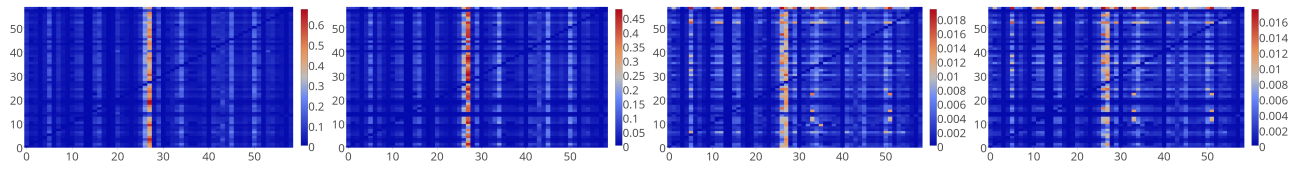
For each row in all influence tables, we sum all its number together. The results can serve as a quantitative measurement for engines' influence in malware detection community. Influence rankings from all the four static models also share a similar pattern.

Observation 7: *Certain anti-virus vendors are influenced by all other vendors in their malware detection decision.*

Testing stage methodology. We also used Spark to implement the testing process and express the process using several stages of map, filter, and reduce. Similar to our training stage process, we first reduce all submissions based on submitted files, next filter out files without action propagation, and then sort submissions chronologically.

To test a submission, we first use Equation 1 to calculate $p_v(S_v(a))$ for all engines that labeled the submitted file as benign when analyzing early submissions but have not labeled the file as malware. S is the set of engines which labeled the file as malwares before. We compare $p_v(S_v(a))$ with a given threshold θ to decide whether we predict v will label the file as malware. We then check what v actually labels in the submission to count true positive (TP) and false negative (FN). After processing all submissions for a file, we calculate $p_v(S_v(a))$ for all engines that label the file as benign but have not labeled the file as malware. We compare $p_v(S_v(a))$ with θ to count false positive (FP) and true negative (TN). In the final stage, we reduce TP, TN, FP, and FN numbers from different files.

Testing results. We change the probability threshold θ from 0.1% to 99.9% and We measure the accuracy of the four static models using ROC (Receiver Operating Characteris-



(a) Bernoulli (b) Jaccard Index (c) Bernoulli with Partial Credit (d) Jaccard Index with Partial Credit

Figure 11: Heat Maps for Learned Models. **(Heat maps drawn for $p_{u,v}$ tables of the 4 models. $Cell(u,v)$ contains influence probability from u to v, where u is row number, and v is column number. A larger number means more influence from u to v.)**

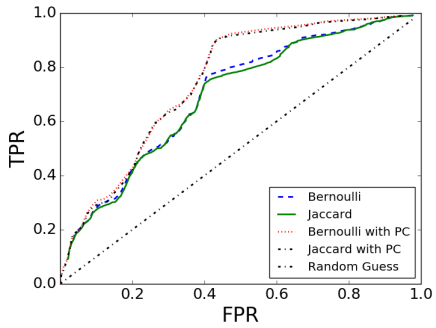


Figure 12: ROC comparisons of Static Models. ***(How true positive rate (TPR) changes with false positive rate (FPR). We change probability threshold from 0.1% to 99.9% with step 0.1%. We compute TPR and FPR for each probability threshold to draw the curve.)***

tic) curves, with true positive rate ($TPR = TP/(TP+FN)$) as X-axis and false positive rate ($FPR = FP/(FP+FN)$) as Y-axis. Figure 12 plots the ROC curves for the four static models. The larger the area under the ROC curve, the better the performance. We find that, all models beat random guesses (the diagonal line between (0,0) to (1,1)) significantly. Using partial credits improves the accuracy of both Bernoulli and Jaccard index.

Observation 8: *The influence model borrowed from social media analysis contains information that helps the prediction of the temporal behavior of malware detection.*

4.3 Discussion

Our findings in this section rings an alert toward using detection rate as the pure measurement of the likelihood of a file being malware and shed light on how vendors can influence each other. Our study results can serve as a quantitative measurement for vendors' influence in malware detection community. Previously, when combining results from different vendors, security experts simply treat each vendor equally and use the percentage of vendors labeling a file as malware as likelihood of the file to be a malware. Our model can provide a weight, when combining results from different vendors. For anti-virus vendors, the prediction part of our models can help alarm possible false negatives in their products.

5. Malware Classification

One potential use case of the massive amount of data available on VirusTotal is to build machine learning models for applications such as malware classifications and clustering. In this section, we study the question that *"Is the data provided by VirusTotal enough to support classification and clustering tasks?"*

Our study reveals both positive and negative answers to this question. Surprisingly, by just using the fuzzy hash strings of files submitted to VirusTotal, we are able to build an automatic classifier whose accuracy is higher than 80% for some classification tasks and we expect the quality to keep increasing given more data. Meanwhile, we identify some classification tasks whose accuracy hardly beat random guesses.

To differentiate these cases and offer future researchers with more insights, we propose the use of a simple metric to predict whether a given task belongs the high-quality category or the low-quality category. We hope our study shed lights on future design of signatures for malware detection.

5.1 Data Preparation

We set out to answer an interesting yet challenging question: can we use compressed representation of files to detect malware? To answer this question, we use features based on ssdeep [12], a program to compute fuzzy hashes. Specifically, ssdeep computes a hash value for each block of the input file. The end of each block only depends on the content of the block. All calculated hash values are concatenated into a fuzzy hash string. So identical blocks in original files can be matched. Similarity between calculated hash strings can serve as an estimation for similarity between the two original files. VirusTotal provides ssdeep hash strings as one of its metadata files and we collected them together with other metadata.

We create training and testing datasets using the detection results from Microsoft, since Microsoft is more reliable in successfully detecting PE malwares [16] than other anti-virus engines. For each detected malware, Microsoft assigns

	Basic Properties				Classification	
Index	**Microsoft Tag**	**# of Malwares**	**% of Tailing**	**% of Distinct**	**Best K**	**Accuracy**
1	SoftwareBundler:Win32/Penzievs	49380	3.51%	99.98%	1	81.27%
2	Adware:Win32/Hotbar	132161	1.88%	99.98%	1	82%
3	TrojanDropper:Win32/Lamechi!rfn	39205	0.01%	6.13%	1	82.55%
4	Virus:Win32/Nabucur.D	1190132	99.95%	100.00%	2	59.81%
5	Virus:Win32/Virut.BO	84600	21.84%	99.95%	1	76.23%
6	Worm:Win32/Mydoom.L@mm	76259	0%	99.44%	3	81.79%
7	Virus:Win32/Ramnit.I	412052	3.07%	96.76%	1	81.79%
8	Trojan:Win32/Dorv.A	54324	2.80%	80.53%	1	82.26%
9	Trojan:Win32/Dynamer!ac	145402	65.17%	99.20%	1	64.41%
10	Virus:Win32/Ramnit.A	181524	26.79%	99.82%	1	74.06%

Table 2: Experimental Data and Classification Results. ***(Information for malwares used in our classification and clustering experiments. # of Malwares: # of distinct malwares we collected for each sampled malware group from 05/07/2016 to 09/06/2016. % of Tailing: % of tailing examples in each group. % of Distinct: number of distinct ssdeep hash strings divided by group size. Best K: best k value selected through cross validation. Accuracy: accuracy from each two-class classifier.)***

a tag that contains type, platform, family, and variant information [3].

We divide PE malwares detected by Microsoft engine into different groups, and malwares in the same group share the same Microsoft malware tag. We randomly sample 10 groups that have more than 10000 malwares. Table 2 presents the Microsoft tag sand the number of malwares in each sampled group we collected from VirusTotal. Within each group, we randomly sample 10000 malwares and use these malwares in our classification and clustering experiments.

5.2 Classification Methodology and Results

We install ssdeep library in a local machine, and compute similarity by using ssdeep's compare API for downloaded ssdeep fuzzy hash strings. We use 1 minus ssdeep similarity to get ssdeep distance.

5.2.1 KNN Classification

We first use KNN [19], as our classifier. KNN looks for k nearest neighbors for a given testing instance, and classifies the test instance as a majority vote of its k nearest neighbors We design several experiments to understand the accuracy of the combination of KNN and ssdeep distance (or similarity) when handling different classification tasks.

Pair-wise two-class classification. We first build a two-class classifier for each of the 10 sampled groups. We use the 10000 sampled malwares in each group as positive examples (positive means being labeled as malware) and randomly sample 10000 benign files as negative examples. Benign files are files submitted to VirusTotal but are not labeled as malwares by any anti-virus engines. We randomly choose 5000 malwares and 5000 benign files for training and use the remaining files for testing.

We run cross validation for k from 1 to 10 to pick up best k value and use the best k value to test KNN using the testing set. Table 2 presents the best k used for testing and classification results. $k = 1$ is always the best k value for testing, except for two groups.

The classification accuracy of the two-class classifier differs in different groups. For five out of ten malware groups, we achieve an accuracy higher than 80%. On the other hand, for groups such as "Virus:Win32/Nabucur.D", the classification accuracy is significantly lower. We get an accuracy of 59% when the accuracy of random guesses would be 50%. Overall, for the two-class classifier, there are not enough malwares and benign files in the training set to generate high-accuracy results.

Complete ten-class classification. We also build a ten-class classifier. We randomly choose 2000 malwares from each group and put half of them in the training set and the remaining half in the testing set. We run cross validation to pick up best k from 1 to 10. The accuracy on the testing set with the best k value 1 is 70.2%.

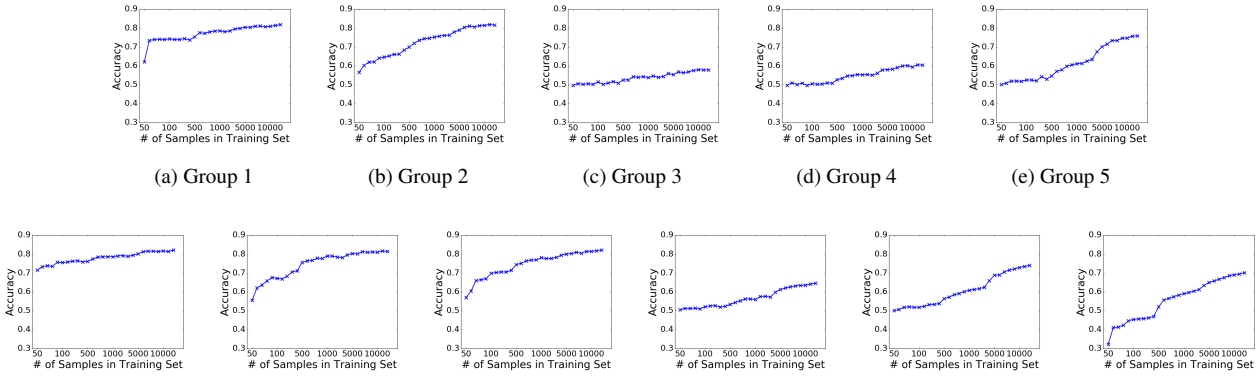
Effect of training set size. For each designed classifier, we further investigate how the size of training sets influences its accuracy. We keep the test sets unchanged and increase the size of training sets from 50 to 10000. We use the best k value got from cross validation in earlier experiments. Figure 13 presents how accuracy changes with the size of training sets. As expected, for all classifiers, accuracy increases as there are more data in the training sets. This constant increase in accuracy implies that with more data from VirusTotal, the classification accuracy is expected to be even higher.

Classifications with semantics labels. To make classification based on more criteria, we further design a set of two-class classifications.

First, we test the classification of malwares with the same *class* but with different *variants* in the class. Specifically, we classify malwares from Group 7 with malwares from Group 10. They are both from the same class "Ramnit", but in different variants. The accuracy we got is 85.3%.

We then target to classify malwares in different *families*. To do this, we classify malwares from Group 3 with malwares from Group 7 and Group 10 and classify malwares from Group 5 with malwares from Group 7 and Group 10. We get accuracies of 94.2% and 85.8%, respectively.

Finally, we test the classification of different *types* of malware. Specifically, we classify malwares from Group 8



(a) Group 1 (b) Group 2 (c) Group 3 (d) Group 4 (e) Group 5

(f) Group 6 (g) Group 7 (h) Group 8 (i) Group 9 (j) Group 10 (k) 10-class

Figure 13: How accuracy changes with more data in traning set. **(Figure 13a - Figure 13j show how accuracy of each two-class classifier changes with the size of training set changing from 50 to 10000. Figure 13k shows how accuracy of the ten-class classifier changes with the size of training set increaing from 50 to 10000. For all figures, the range of Y-axis is from 0.3 to 0.9.)**

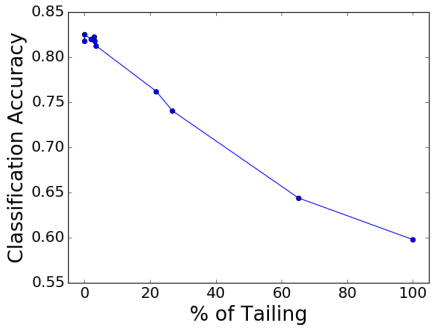


Figure 14: The relation between classifer's accuracy and the percentage of tailing. ***(How classifier's accuracy changes with the percentage of tailing.)***

and Group 9 with malwares from Group 4, Group 5, Group 7, and Group 10, the former groups have Trojan malwares and the latter group contains Virus. For this classification, we get 77.0% accuracy.

Observation 9: *Different types of classification on VirusTotal file ssdeep hash strings result in different accuracy, and increasing the training set size results in higher accuracy.*

5.3 Tailing Malwares

From the above study of various types of classification, we find that using ssdeep hash strings to learn the similarities of VirusTotal submission files is not always accurate. But in some cases, the accuracy is very high. To make use of these high-accuracy classification, we need a good technique to separate them from low-accuracy classification.

We identify a new metric to predict whether a classification task falls into the high-accuracy category or the low-accuracy category. The intuition behind our idea is simple: the ssdeep hash similarity of a data sample to other samples in the same group in the training set is a proxy of the upper bound of accuracy that we can expect. If no samples in a group are similar to each other, then no classifier can have high accuracy in predicting this class. On the other hand, more similar samples will make it easy for a classifier to have high accuracy.

We call malwares that have no similarity with any other samples in the same group *tailing malwares* and compute the percentage of tailing malwares in each group. Table 2 lists the percentage of tailing malwares for each sampled group. Figure 14 plots the classification accuracy for all two-class classifiers against the percentage of tailing malwares. The percentage of tailing malwares has a perfect linear relationship with classification accuracy, implying that we can use tailing malware to predict the accuracy of a classifier.

Observation 10: *The percentage of tailing malwares can predict whether a given malware classification task is likely to have high or low accuracy.*

5.4 SVM Classification

Our previous experiments all used KNN as the classification algorithm. We now discuss the potential of using more sophisticated classifiers such as support vector machines (SVM). SVM could take distance matrix as input, but the size of the distance matrix is quadratic to the number of samples in the training set, which would consume large memory and long execution time. Nystrom methods [22] are popular ways to approximate a distance matrix with clusters. Using Nystrom methods, each instance is transformed to a feature vector, where each dimension is the distance to a cluster center.

To understand the potential of applying Nystrom methods, we run hierarchical clustering [18] on our data. Hierarchical clustering starts with each instance as a cluster, and

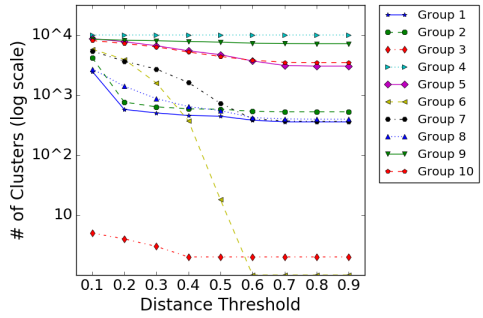


Figure 15: The relation between resulting clusters and distance threshold. ***(How the number of resulting clusters change with different distance threshold.)***

then it iteratively merges two clusters with minimum distance until distance threshold or cluster number threshold is reached. We use distance as threshold and the single linkage distance [20] as distance between two clusters.

We change the distance threshold from 0.1 to 0.9 and count the resulting clusters under each experiment. Figure 15 presents the experimental results. As we increase the distance threshold, the number of resulting clusters decreases for each group. If we want to change a ssdeep hash string to a feature vector based on distance of the string to the center of all clusters in training set, the size of the resulting feature vector would have a very large variance across different malware groups. This illustrates a challenge of directly applying classic Nystrom method to our problem. However, we think that it is promising to explore how to develop new approaches to accommodate this observation.

5.5 Discussion

Our study found that by just using the ssdeep fuzzy hash string, we can achieve high accuracy in certain classification tasks, especially so with more data, and we developed a technique to predict when classification accuracy will be high. The combination of these results point to one interesting direction for future researchers and security vendors, and to VirusTotal: instead of original files, we can use compressed representation of files to identify malwares.

With the "big data" available in the current world, we believe that this is a fruitful area to investigate. The implication of this finding is significant. Users can have stronger privacy and do not need to submit, or reveal, their files to know if these files are likely to be malware.

6. Related Works

We are not the first to investigate VirusTotal. There have been several works on VirusTotaland increasing attention has been paid to the VirusTotal repository recently.

The closest related work is a recent study on VirusTotal [16]. Similar to our work, they downloaded one-month data from VirusTotal and focused on PE files. They studied a set of basic properties of PE malwares, including size distribution, malware family distribution, and temporal properties. We collected a longer period of data from VirusTotal. More important, we performed deeper analysis into what are the fundamental factors that correlate and influence malware detection. Apart from studying malwares, we also study anti-virus engines and how they influence each other and perform classification on VirusTotal data. These findings provide far more insights than the previous work and will be able to help future researchers more. Finally, the previous work only relied on detection results from Microsoft anti-virus engines, while we conducted our study using all engines.

Recently, researchers noticed that some malware writers use VirusTotal as testing platform [7, 10]. They explored this phenomenon and developed techniques to identify malware writers. Huang *et al.* downloaded four months of VirusTotal metadata to identify Android malware development cases [10]. Their technique first alerts suspicious source IDs and then conducts program analysis to further validate whether these source IDs are really testing their Android malware prototypes on VirusTotal. Graziano *et al.* tried to identify Windows malware development cases on Anubis [7]. They used data on VirusTotal to validate their findings. Their works targets to identify suspicious source IDs, while we focus on understanding correlations between source IDs' history and detection rate of their submissions.

There have also been many works on exploring how to leverage open-source code repositories that contain vast amount of source code projects, such as GitHub, BitBucket, and CodePlex [11, 14, 15]. These systems analyze source codes by leveraging large code bases in open-source code repositories. We have different objectives and methods: we analyze malwares and anti-virus engines by leveraging an open malware repository.

7. Conclusion

VirusTotal serves as a fruitful resource to understand malwares and anti-virus engines in the real world. In this paper, we conduct a thorough study on four months of VirusTotal submission metadata. We first study which factors correlate to submissions' detection rate and identified three factors: file size, submission history, and submitter reputation. We then investigate the influence between different vendors using a graph model borrowed from the social media analysis area. Our experimental results show that some vendors are highly influenced by all other vendors and cautions should be taken to use these detection results. Finally, we study whether hash strings of files are enough to build accurate machine learning classifiers and predictors. We design several classifiers based on ssdeep similarity and these classifiers have varied accuracy. We also identify a simple metric that can predict accuracy for a given classification task.

References

[1] AV-TEST. Malware Statistics. URL: https://www.av-test.org/en/statistics/malware/.

[2] Cassandra. Apache Cassandra database. URL: http://cassandra.apache.org/.

[3] M. M. P. Center. Naming malware. URL: www.microsoft.com/security/portal/mmpc/shared/malwarenaming.aspx.

[4] A. Davis and M. Wolff. Deep learning on disassembly data. BlackHat'15.

[5] I. Gashi, V. Stankovic, C. Leita, and O. Thonnard. An experimental study of diversity with off-the-shelf antivirus enginess. NCA'09.

[6] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 241–250, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. . URL http://doi.acm.org/10.1145/1718487.1718518.

[7] M. Graziano, D. Canali, L. Bilge, A. Lanzi, and D. Balzarotti. Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 1057–1072, Berkeley, CA, USA, 2015. USENIX Association. ISBN 978-1-931971-232. URL http://dl.acm.org/citation.cfm?id=2831143.2831210.

[8] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy. Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 495–504, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-719-6. . URL http://doi.acm.org/10.1145/1806799.1806871.

[9] A. Gupta, P. Kuppili, A. Akella, and P. Barford. An empirical study of malware evolution. In *Proceedings of the First International Conference on COMmunication Systems And NETworks*, COMSNETS'09, pages 356–365, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2912-7. URL http://dl.acm.org/citation.cfm?id=1702135.1702182.

[10] H. Huang, C. Zheng, J. Zeng, W. Zhou, S. Zhu, P. Liu, S. Chari, and C. Zhang. Android malware development on public malware scanning platforms: A large-scale data-driven study. In *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)*, BIG DATA '16, Washington, DC, USA, 2016. IEEE Computer Society.

[11] S. Karaivanov, V. Raychev, and M. Vechev. Phrase-based statistical translation of programming languages. In *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software*, Onward! 2014, pages 173–184, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3210-1. . URL http://doi.acm.org/10.1145/2661136.2661148.

[12] J. Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digit. Investig.*, 3:91–97, Sept. 2006. ISSN 1742-2876. . URL http://dx.doi.org/10.1016/j.diin.2006.06.015.

[13] D. Palmer. This ransomware is now one of the three most common malware threats. URL: http://www.zdnet.com/article/this-ransomware-is-now-one-of-the-three-most-common-malware-threats/.

[14] V. Raychev, M. Vechev, and E. Yahav. Code completion with statistical language models. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 419–428, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2784-8. . URL http://doi.acm.org/10.1145/2594291.2594321.

[15] V. Raychev, M. Vechev, and A. Krause. Predicting program properties from "big code". In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, pages 111–124, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3300-9. . URL http://doi.acm.org/10.1145/2676726.2677009.

[16] L. Song, H. Huang, W. Zhou, W. Wu, and Y. Zhang. Learning from big malwares. In *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys '16, pages 12:1–12:8, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4265-0. . URL http://doi.acm.org/10.1145/2967360.2967367.

[17] G. Vigna. Antivirus Isn't Dead, It Just Can't Keep Up. URL: http://labs.lastline.com/lastline-labs-av-isnt-dead-it-just-cant-keep-up.

[18] Wikipedia. Hierarchical clustering. URL: https://en.wikipedia.org/wiki/Hierarchical_clustering, .

[19] Wikipedia. k-nearest neighbors algorithm. URL: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, .

[20] Wikipedia. Single-linkage clustering. URL: https://en.wikipedia.org/wiki/Single-linkage_clustering, .

[21] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1863103.1863113.

[22] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nystrÖm low-rank approximation and error analysis. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1232–1239, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. . URL http://doi.acm.org/10.1145/1390156.1390311.

[23] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 95–109, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4681-0. . URL http://dx.doi.org/10.1109/SP.2012.16.