Understanding Real-World Malware and Anti-Virus Engines

Anonymous Author(s)

ABSTRACT

In this paper, we conducted a thorough empirical study on real-world malware and anti-virus engines by examining the largest real malware repository, VirusTotal. Our study is performed in three dimensions. First, we perform an analysis on various properties of submissions, submitting entities, and anti-virus engines in VirusTotal. Second, we study the correlation between malware's metadata and their detection rates. Finally, we model how influence propagates across different anti-virus vendors. Our study results confirmed a set of hypothesis and anecdotal assumptions and revealed a few surprising new insights into malware and anti-virus engines. These results can shed light on future research directions and assist both anti-virus vendors and normal users in their fight against malware.

ACM Reference format:
Anonymous Author(s). 2017. Understanding Real-World Malware and Anti-Virus Engines. In *Proceedings of 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, 4–8 September, 2017 (ESEC/FSE 2017),* 11 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Malware grow exponentially [5] and place an imperative threat to human society. For example, more than 390000 new malware are registered in AVTest institute every day [5]. As another example, a new type of threat, ransomware, has caused more than 1 billion dollars this year [15]. To fight against malware and detect these new types of malware, anti-virus tools are also improving rapidly, by constantly updating their signature database, by using more advanced techniques like deep learning [7], or by utilizing more data.

In order to improve anti-virus tools and defend against emerging threats from malware, it is essential to understand malware and existing anti-virus tools in the real world. Previous works on studying malware and anti-virus engines do provide valuable insights [8, 12, 24] such as how malware writers create new malware and how malware escape from the detection of anti-virus engines. However, most previous works focus only on one or few problems of malware with limited scale. There is also a lack of study on anti-virus engines and how they relate to malware.

Fortunately, we live in the "big data" era and there is no shortage of data on malware and anti-virus engines in the real world. One such type of data repositories is online malware analysis services. There are many online malware analysis services [1–4] that use malware sandboxes and state-of-the-art anti-virus engines to analyze user-submitted files or URLs and produce detailed detection reports.

We propose to use the vast amount of malware big data in online malware services to study real-world malware and their relationship with anti-virus engines. To conduct such study, we utilized the biggest open data repository that contains billions of real-world malware, *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 malware and anti-virus engines for the following reasons.

First, there are a 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 malware 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 detailed detection results from various anti-virus engines, VirusTotal also provides file type information, submitter information, and a hash string of the original submitted file. These *metadata* can be used to better understand malware and assist both security experts and normal users in malware detection.

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 [10, 13]. 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.

In this paper, we conduct an extensive, large-scale study on both malware and anti-virus engines using data collected from VirusTotal. We collected 4 months of metadata of all file submissions to VirusTotal, a total of 151 million submissions and 120 million files. We focus on analyzing malware files and leave URLs for future work, since file submissions contribute to the majority of VirusTotal repository.

After collecting and pre-processing data from VirusTotal, we first perform a basic analysis to gain an overall knowledge of the VirusTotal data repository. This analysis answers a rich set of fundamental questions about the VirusTotal data repository including what types of files and how many of them were submitted to VirusTotal; who submitted files to VirusTotal, at what time, and from where; how big are the files submitted to VirusTotal; how many anti-virus engines

ESEC/FSE 2017, Paderborn, Germany

DOI: 10.1145/nnnnnnn.nnnnnnn

were used to analyze a submission and how many of them labeled it as malware.

On top of these basic findings, we developed a set of more advanced studies in two 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. We further developed a regression model to capture the effect of all these factors on detection rates and found that using just these three factors, we can predict malware with a much higher accuracy than random guess. 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.

To fight against malware, it is not enough to just understand malware; we should also understand anti-virus engines and their detection results. Specifically, 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 malware detected by others' products but not detected by their own products.

To verify this hypothesis, we used the detection history from VirusTotal to create a new type of graph that we call *influence graph* to model the influence relationship across vendors. We further developed a set of statistical models on top of the influence graph to quantify the influence between vendors and a prediction model to predict how likely the decision of a vendor is influenced by others. 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.

Our study advances the understanding of malware and anti-virus engines in the real world and provides various valuable insights for future researchers and vendors, some of which are surprising and have never been revealed in the past. Specifically, this paper makes the following main contributions. We hope that our findings and our methodologies can help future security researchers and practitioners to better understand malware and fight against them.

- Online malware detection services offer rich sets of real-world data that are representative of both the latest malware and malware in history. Our study show that it is not only feasible and but also valuable to study these data in large scale. Thus, we call attention from our research community to investigate these online repositories more closely.
- We show that with just file metadata, we can already achieve malware detection that is significantly more accurate than random guess. Researchers and anti-virus vendors can leverage these properties to perform initial filtering through the vast amount of file submissions and focus their efforts on more suspicious files. Moreover, although still not comparable to anti-virus engines that scan files to detect malware, our results open up the possibility of using metadata only to detect malware. The implication is huge: users may not need to transfer their original files to an untrusted anti-virus service but only submit file metadata.
- As far as we know, we are the first to perform a big data analysis on anti-virus engines themselves. Our influence study results in the surprising finding that there are anti-virus vendors who are influenced by almost all other vendors, while some vendors influence many other vendors. This result alerts normal users and security experts to be treat detection results from anti-virus vendors with more caution.
- We developed a set of analytical methodologies, statistical models, and prediction methods to study malware and anti-virus engines. They can assist future researchers and practitioners to perform more analysis on other malware data and develop more advanced analytical techniques.

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 who made 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 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 VirusTotal through distribution API and their related explanation. One file could be submitted multiple times by different users.)

The rest of this paper is organized as follows. Section 2 introduces VirusTotal and discusses how we collect data from VirusTotal. Section 3 presents the basic analysis of the VirusTotal data we collected. Section 4 studies the correlation between various metadata properties and detection rate. Section 5 analyzes how anti-virus vendors influence each other. Finally, we discuss related works in Section 6 and conclude in Section 7.

2 DATA COLLECTION

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 [10, 13].

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.

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. We use many of these metadata types in our study throughout this paper.

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 [13, 19]. According to Lastline Labs [20], 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 behaviors for most malware and anti-virus engines. Indeed, we drew several meaningful conclusions with this dataset, as will be presented in the rest of the paper.

We performed our data collection using VirusTotal's distribution API. We insert all collected metadata into a table in Cassandra [6] using the combination of sha256, source_id, and timestamp as the key. We then used Spark [23] 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 file type distributions 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 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 malware 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 malware for future. In total, we collected 76 million PE submissions.

2.3 Caveats

Like all other empirical studies, 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 malware never submitted to VirusTotal, or submitted to VirusTotal much later than when 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 malware in the real world.

3 BASIC PROPERTIES

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.

3.1 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 3.

Figure 4 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.*

3.2 Source ID

Both normal users and anti-virus vendors use VirusTotal to detect malware 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.*

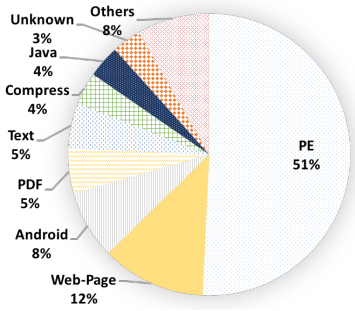


Figure 1: File type distributions. (File types and their distributions for all VirusTotal submissions from 05/07/2016 to 09/06/2016.)

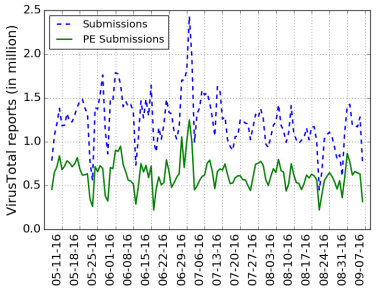


Figure 2: The number of files and PE files. (The number of suspicious files and the number of PE files submitted to VirusTotal from 05/07/2016 to 09/06/2016.)

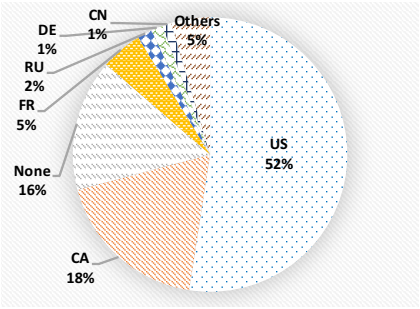


Figure 3: How PE submissions distribute among different countries. (Only countries with more than 1% PE submissions are shown.)

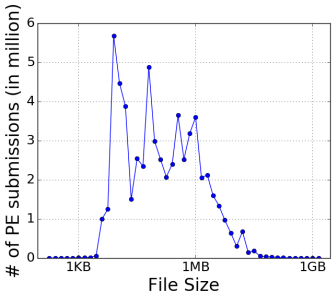


Figure 4: File size distribution for PE submissions. (How file size distributes among all PE submissions. Results from log2 are rounded up to nearest 0.5.)

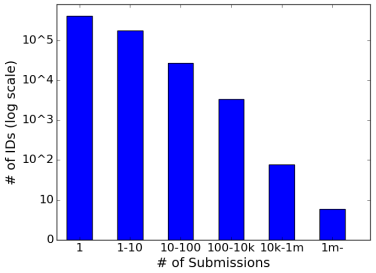


Figure 5: Source ID Category. (We categorize source IDs according to the number of submissions they make. Y axis represents the number of IDs in log scale for each category.)

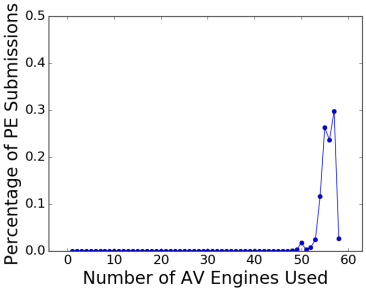


Figure 6: The distribution for the number of used anti-virus engines. (How the number of applied anti-virus engines distributes among all PE submissions.)

3.3 Anti-virus Engines

After learning the basic properties of submissions and source IDs, 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 each submission. 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 malware 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 in Section 3.2). 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.*

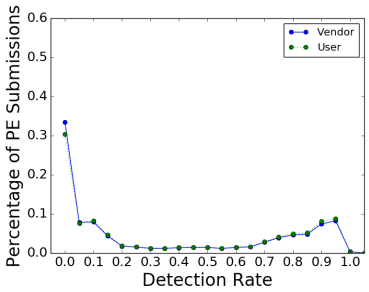


Figure 7: The distribution for detection rate. (How detection rate distributes among all PE submissions. Each detection rate is rounded up to nearest 0.05.)

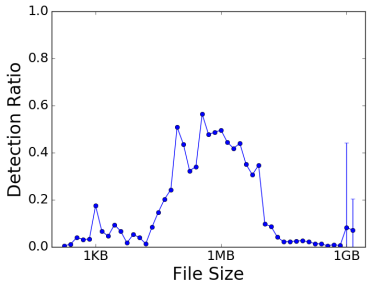


Figure 8: How detection rate changes with file size. (95% confidence interval is also drawn for each point. Results of log2 for the original file sizes are rounded to the nearest 0.5 value.)

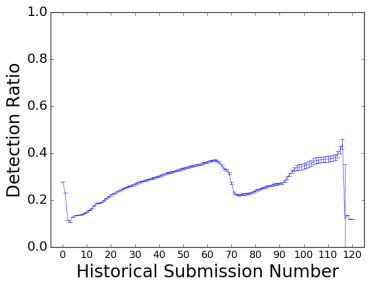


Figure 9: The relation between historical submission number and detection rate. (How detection rate changes with historical submission number. Each historical submission number is rounded up to nearest 5. 95% confidence interval is also drawn for each point.)

3.4 Discussion

This section presents the basic properties of the VirusTotal data repository. It demonstrates the rich set of data that comes from the real world. These analysis results motivated this paper and we hope that they can encourage future researchers to investigate online malware big data as well.

Our experience here also showed that while VirusTotal and other online detection services provide many types of submission metadata, not all of them are important and of interest to malware and anti-virus engine analysis, as will be seen in the next two sections.

4 CORRELATION BETWEEN SUBMISSION PROPERTIES AND DETECTION RATE

Detection rate is what most VirusTotal users refer 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 affect or correlate to detection rates.

While anti-virus engines detect malware relying on the direct analysis of files themselves, scanning and analyzing the large amount of files can be time-consuming and require lots of computing resources, especially in today's big-data era. Moreover, obtaining original files may not always be possible or desirable; for example, users may not want to disclose their files to an online service like VirusTotal but still want to detect malware in their files. Thus, we raise the question of *if there is any way to assist or even conduct malware detection with file properties only?*

This section presents our study of the correlation between various properties of PE submissions and their detection rate. We focus on VirusTotal submissions' properties that can be obtained without executable binary files and studied their correlation with detection rate. We found that three factors have higher correlation: submission file size, historical submission properties, and the reputation of source IDs. We also built a regression model based on these three factors to predict detection rate. We present these correlation study results and our regression model in this section.

4.1 File Size

Anecdotally, PE malware are most likely to have small to medium size. To verify this hypothesis, 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 away from the real average detection rate. To simplify calculation, 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 malware are mostly likely to be of small to medium size.*

An immediate question that follows 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 4. As we will discuss in Section 4.2, 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.

4.2 Submission History

As discussed in Section 3.1, many files are submitted multiple times, from one or more users to VirusTotal. It is worthwhile to investigate this *history of submission* to check 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 including the number of submissions of a file made in history, the number of users that submitted the same file in history, and the number of countries that have submitted a

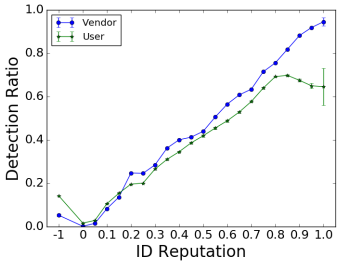


Figure 10: The relation between source id's previous reputation and detection rate. (How detection rate changes with the value of source id's reputation. Each reputation is rounded up to nearest 0.05. Reputation -1 means the source id did not make any submission before. 95% confidence interval is also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

file, we find that the number of submissions made in history has the highest correlation to detection rate.

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 in history. Figure 9 plots how detection rates change over the number of historical submissions.

The overall trend from this result is that more historical submissions result in higher detection rate. Intuitively, if a file is submitted many times, it means that many users suspect this file as malware and/or a user insist her suspicion that the file is malware and submit it over and over. In both cases, there is a higher chance that this file indeed is malware and is thus being detected as malware by more anti-virus engines.

Interestingly and counter-intuitively, there are three drops in the beginning, middle, and end. We think that for files with high detection rate, users are satisfied with VirusTotal's results, and they stop submitting these files. Percentage of submitted files with low detection rate increases, and this causes average detection rate drops.

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

4.3 Submitting User

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 [11] reported that there is a correlation between bug reporter's reputation and the likelihood for the bug being fixed. We also observe correlation between the reputation of source ID and submission's 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 malware in suspicious files and thus should be given higher reputation. The reputation of a source ID is not a fixed value and can change over time. To quantify this value, we define the reputation of a source ID, the unique identifier of a VirusTotal user, as the follow.

Definition 4.1. 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 that have no source ID information and submissions by source IDs with more than 1 million submissions (these are likely to be bogus or coming from robots). For the remaining source IDs, we then sort submissions from the same source ID chronologically and calculate each source ID's reputation when it conducts a 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) generally 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, which means that for normal users, it is difficult for them to always submit suspicious files detected by almost all vendors. Vendors do not exhibit this behavior and their detection rate always increases with higher reputation (other than reputation -1), showing that vendors' reputation can almost always be trusted.

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

4.4 Comprehensive Correlation Model

From the above study in this section, we find three factors that are correlated to detection rate: submission file size, the number of submission made in history, and source ID reputation. Two natural questions that follow are *how the combination of these three factors correlate to detection rate* and *whether or not we can use these finding to predict the likelihood of a file being malware*. A positive answer to the second question can largely assist security researchers and practitioners to narrow down the vast suspicious files to a much smaller set. More important, such an answer can open up the possibility to perform an initial prediction of malware *with metadata only* and *no original file*.

Methodology. To answer these questions, we build a linear regression model using the three factors as features of the model to predict the detection rate of a submission. We first filter out submissions without

source ID reputation information (Section 4.3). We then randomly divide remaining submissions into a training set and a testing set with equal size. We use training set to train our regression model and two ways for testing the trained model: in-sample testing where we use the whole training set for testing, and out-of-sample testing where we use the testing set for testing.

For comparison to our regression model, we use a model that is based on random guesses; it guesses the detection rate for any submission as the mean of detection rate for all training submissions.

We calculate the mean absolute error (*mae*) [21] and the mean squared error (*mse*) [22] to compare our regression model and the random model. Smaller mae and mse values mean lower error rates.

Modeling results. For in-sample testing, mae and mse for our regression model are 0.2227 and 0.0718 respectively, while for baseline model they are 0.3345 and 0.1327. For out-of-sample testing, mae and mse for our regression model is 0.2223 and 0.0716 respectively, and they are 0.3342 and 0.1325 for baseline model. For both in sample testing and out of sample testing, our regression model improves the prediction accuracy over the random model (from 33.4% to 46.0%). Errors of our regression model under in-sample testing and out-of-sample testing are almost the same, which shows that our model generalizes well across different data set.

After obtaining satisfactory accuracy results with our regression model above, we further analyze the importance of each of the three factors in achieving good model accuracy. The learned coefficients from our regression model for submission file size, the number of submission made in history, and source ID reputation are $7.12*10^{-3}$, $1.76*10^{-4}$, and $6.03*10^{-1}$ respectively. To understand the impact of each factor on prediction results, we exclude the three factors one by one and use the remaining two to train three regression models with out-of-sample testing. Excluding submission file size, the number of submissions in history, and source ID reputation increase mse to 0.0844, 0.716, 0.1390 respectively (17.9%, 0%, 94.1% worse), This result shows that source ID reputation has the biggest influence on detection rate, while the number of submissions in history has insignificant influence. There are two possible reasons why the number of submission in history has only a very small influence on detection rate. First, the majority of files are only submitted once in VirusTotal. Thus, all these submissions have the same number of submission made in history, but have various detection rate. Second, the relationship between the number of submission and detection rate is not linear as shown in Figure 9. Thus, only one coefficient cannot describe how detection rate change with the number of submission precisely.

Observation 7: *Regression model built with the three studied factors can help the prediction of detection rate for malware submissions. Among these three factors, source ID reputation has the most significant impact on detection rate, while the number of submissions made in history has only a minimal impact.*

4.5 Discussion

From our analysis, file size, number of submissions in history, and source ID reputation are all correlated with detection rate. Our regression model based on these three factors can effectively help predict detection rate for VirusTotal submissions.

All the three studied factors can be obtained without analyzing binary executable files. These factors plus our regression model provide an easy way to perform an estimated malware prediction of a file without knowing the file content. Researchers and vendors can use this method to perform an initial filtering of the huge amount of malware submissions, so that they can easily focus on more interesting files—files that are more likely to be malicious. More important, our results here shed light on the possibility for normal users to not disclose their files while still be able to have an initial prediction of suspicious files.

5 INFLUENCE AMONG ANTI-VIRUS ENGINES

Anti-virus vendors frequently use VirusTotal to identify false negatives in their products, which are malware they fail to detect but detected by other vendors [18]. 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. Interestingly, we observe that some engines fail to identify some malware during early submissions, but they catch up when analyzing later submissions.

This observation led us to ask one important question that have never been studied before: *is there influence across different anti-virus vendors?* That is, can an anti-virus vendor's decision be affected by other vendors' malware detection results? With the large number of anti-virus engines in the wild, it is important to understand if an anti-virus engine is reliable and can be trusted.

This section presents our answer to this question by quantifying the influence across anti-virus engines. Specifically, we study the historical submissions of a file and how anti-virus engines change their labels of the same file over time. We also provide a prediction model on whether an engine will identify a file as malware in the future after labeling the file as benign—a warning flag that this engine's decision is highly likely to be resulted from the influence of other engines.

5.1 Influence Graph

We now discuss our proposed mechanism to analyze the influence among anti-virus engines. Our goal is to estimate the *trustworthiness* of an anti-virus engine i, T_i.

We propose to model the anti-virus engine influence problem as a graph problem. Influence propagation in social networks is a well-studied topic in the web mining area. Inspired by social graph solutions [9], we propose to use graphs to represent the relationship between vendors and model influence among different vendors based on static models.

We first build a complete directed graph $G = (V, E)$ called *influence graph* for the influence problem, where the nodes V are vendors and edges model the likelihood of influence between two vendors. We choose to use a complete graph because we initially assume that it is possible to have influence from one vendor to any other vendor.

We use *action* to describe the detection result of an engine to a submission. Since a file can be submitted more than once and an engine can analyze it multiple times, we need to associate action with time. We formally define an action, a, at time t as (u, a, t) or $(u, \bar{a}, t)$, where the former represents an anti-virus engine identifies

the submission as malware and the latter represents it identifies the submission as benign.

We limit the goal of this study to detecting whether or not a vendor changes its labeling of a file from benign to malware because other vendor(s) have labeled it as malware (what we call *positive influence*). Thus, we do not consider the case of changing from labeling malware to benign (*negative influence*). We also assume that after a vendor labels a file as malware it will not change its decision. A similar model can be applied to study negative influence and we leave it for future work.

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. Since we only consider positive influence, when calculating $p_{u,v}$, we only consider cases where after u has labeled a submission as malware, v will change its label from benign to malware in a later time, i.e., $\exists\ (u, a, t_1)$, $(v, \bar{a}, t_2)$, (v, a, t_3) where $t_1 < t_3$ and $t_3 > t_2$.

To capture the engines influence an engine, we use the neighbor relationship in the influence graph. We define $S_v(a)$ to be the set of v's neighbors that take action a before v in time. The probability that v will follow its neighbors to take the same action can then be calculated as:

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

We can use $p_{u,v}$ to further estimate the trustworthiness of an engine, T_v. Intuitively, if an anti-virus engine gets more influence from others, it is less reliable and trustworthy. Thus, we have

$$T_v = \frac{1}{\sum_{u \neq v} p_{u,v}} \tag{2}$$

5.2 Influence Probability Estimation Model

From the above analysis, we can see that the influence probability is the key of our anti-virus engine influence problem. After knowing the value of the influence probability $p_{u,v}$ between all engine pairs u and v, it is easy to calculate the trustworthiness of all engines. The problem now boils down to estimating $p_{u,v}$.

To estimate the influence probability between two engines, we propose to use the statistics of actions taken by an engine on a file and the action propagation between two engines. Specifically, we use the *happen-after* relationship to define *action propagation*; if an action of an engine is taken after an action of another engine, we say that this action is propagated.

To assist the definition of action propagation, we first define a few types of submission sets. We use A_u to represent the set of submissions ever identified by u as malware in its history and $\bar{A}_u$ to represent the set of submissions that have been labeled as benign and yet not labeled as malware 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, $|A_{u|v}| = |A_u| + |A_v| - |A_{u\&v}|$. With these sets defined, we formally define *action propagation* in the following equation and use A_{u2v} to represent the set of all the actions that are propagated from u to v.

Definition 5.1. Action Propagation: 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 proposed static models to estimate $p_{u,v}$.

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.

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

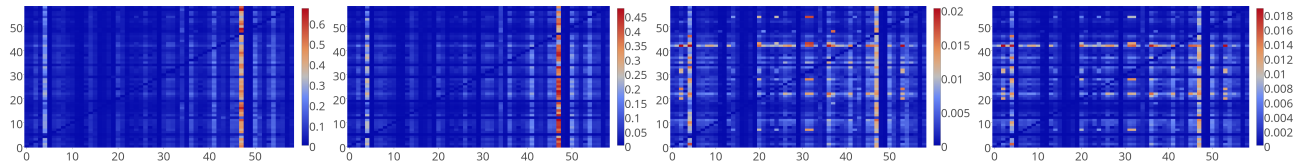
Since $|A_{u|v}|$ is not less than $|A_u|$, $p_{u,v}$ calculated by Jaccard index is not larger than $p_{u,v}$ calculated by Bernoulli distribution. For Bernoulli distribution and Jaccard index, considering partial credit will decrease $p_{u,v}$ calculated by both of them.

5.3 Influence Probability Analysis

Using the four static models discussed above, we built an influence graph with anti-virus engines on VirusTotal and analyzed the influence probability ($p_{u,v}$) between each pair of engines.

Methodology. We implement our analysis using several stages of map, filter, and reduce in Spark [23]. Specifically, we first map submission records according to their corresponding files and filter out submissions without action propagation.

We then filter out engines taking fewer than 20000 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 392K. For confidentiality reasons, we



(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.)

omit vendors' name in the following discussion and use numbers from 0 to 58 to represent these vendors.

Analysis results and implications. For each static model, we construct an influence table with calculated influence probability values ($p_{u,v}$), with u as row number and v as column number. We visualize these influence tables with heat maps in Figure 11. Each $cell(u,v)$ represents the relative heat color of $p_{u,v}$, and red means more influence from u to v.

We also sum all $p_{u,v}$ values in a column in each influence table to calculate trustworthiness of engine v using Equation 2. Table 2 lists the maximum, minimum, and average trustworthiness values of all the engines under four different models. The results can serve as a quantitative measurement for how trustworthy anti-virus engines are.

In all the four heat maps, there are two columns with almost all cells in red, which means that these vendors influenced by all other vendors and are less trustworthy. On the other hand, quite a few vendors are not influenced by any vendors (blue columns). This result suggests that most vendors perform predictions on their own and thus can be trusted. However, our result should also ring an alarm to online malware detection service users and security experts that we cannot treat all engines with the same level of trust and the results from some of them should be at least examined more closely if not discarded.

At the same time, there are several rows with many cells in red, especially in the last two heat map, which means that there are vendors which influence many vendors, where there are a set of vendors that do not influence other vendors (blue rows). Different from the effect of columns which shows the trustworthiness of a vendor by users, the effect of rows is related to how vendors view (and trust) each other. Certain vendors are highly regarded by many other vendors so that their decision results are often referenced by other vendors.

Interestingly, the vendors that are heavily influenced by others are not the ones that influence others, suggesting that the vendors with lower trustworthiness by users are also not trusted by other vendors.

Model	Max	Min	Average
Bernoulli	108.78	0.04	5.40
Jaccard Index	118.20	0.05	6.15
Bernoulli with Partial Credit	3545.34	1.63	138.45
Jaccard Index with Partial Credit	4177.98	2.02	155.18

Table 2: Trustworthiness. (Maximum, minimum, and average trustworthiness values under four different models.)

Finally, we observe that although the four static models show similar trends of influence relationship and the relative rankings of engines in terms of influence are similar, their absolute influence probability values differ. Bernoulli distribution and Jaccard index have influence probability values that are about 30 to 40 times higher than Bernoulli and Jaccard index with partial credit.

Observation 8: *Certain anti-virus vendors are influenced by almost all other vendors in their malware detection decisions and are less trustworthy, while quite a few trustworthy vendors have influence on many other vendors.*

5.4 Influence Probability Prediction

The analysis results of vendor influence above are encouraging. With these results, we take a step further and ask *if it is possible to predict whether or not an engine's prediction of a file submission should be trusted (i.e., not influenced by other engines)?* We now discuss the prediction model we built to answer this question.

Methodology. We first 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 (starting from 'a' to 'f') into the testing set. We use Spark to implement both the training and the testing process.

During the training stage, we use the training set to build an influence graph and learn $p_{u,v}$ of each edge in the graph using the four static models discussed in Section 5.2. The calculation of $p_{u,v}$ is the same as in Section 5.3.

During the prediction stage, for each submission in the testing set, we predict whether or not an engine v will take an action a following other engines if it has not taken that action yet. Specifically, to test a submission, we first use Equation 1 to calculate $p_v(S_v(a))$, the probability that v will follow its neighbors to take the same action, for all engines that have labeled the submitted file only as benign in their history. These engines are of interest to us because positive influence, the type of influence in this study, only happens when an engine changes its prediction decision from benign to malware.

We then compare $p_v(S_v(a))$ with a *tunable threshold* θ to predict whether v will label the file as malware (if $p_v(S_v(a)) > \theta$) or not. Next, we compare this prediction with the actual action that v took in the testing set to deduct true positives (TPs) where both our prediction and the actual fact label the submission as malware, false negatives (FNs) where our prediction labels the submission as benign and the actual labeling is malware. Thus, TP means our prediction is the same as the fact and both show the engine changes its labeling

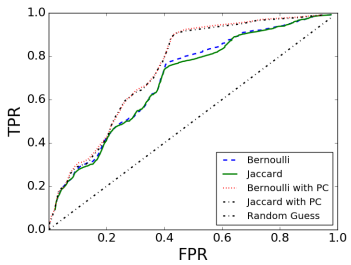


Figure 12: ROC comparisons of Static Models. (How true positive rate (TPR) changes with false positive rate (FPR). We change probability threshold from 0.1% to 99.9% with step 0.1%. We compute TPR and FPR for each probability threshold to draw the curve.)

from benign to malware, an indication that the decision of this engine on the submission is in deed affected by other engines. Similarly, we can obtain false positives (FPs) and true negatives (TNs). In the final stage, we calculate the overall TP, TN, FP, and FN rates for all different files.

Prediction results. We change the threshold θ from 0.1% to 99.9% and measure the accuracy of the four static models using ROC (Receiver Operating Characteristic) 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. A larger area under the ROC curve means higher accuracy. We compare our prediction models with random guess, which is represented by the diagonal line between (0,0) to (1,1) in the figure.

We find that all models are more accurate than random guess and using partial credits improves the accuracy of both Bernoulli and Jaccard index. This result is encouraging; even with a simple prediction model like ours, we can already obtain satisfactory prediction accuracy of whether an engine's decision has been influenced by others.

Observation 9: *Our influence prediction model can accurately predict whether or not an anti-virus engine's labeling of a submission should be trusted.*

5.5 Discussion

Our findings in this section shed light on how vendors can influence each other and rings an alert towards using detection rate as the only measurement of the likelihood of a file being malware. 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 to each vendor when combining results from different vendors. Anti-virus vendors can also use our prediction model to detect possible false negatives in their products.

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 [19]. 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 malware, 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 malware, 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 [10, 13]. 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 [13]. 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 [10]. 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 [14, 16, 17]. SLANG [16] trains statistical models using extracted sequences of API calls from large code bases and applies the trained models to fill uncompleted programs with call innovations. JSNICE [17] aims to predict identifier types and obfuscated identifier names for JavaScript programs. A score function based on learned CRF model is optimized to make all predictions. Phrase-based statistical translation approaches have also been proposed to translate C# programs to Java programs [14]. These systems analyze source codes by leveraging large code bases in open-source code repositories. We have different objectives and methods: we analyze malware and anti-virus engines by leveraging an open malware repository.

7 CONCLUSION

VirusTotal serves as a fruitful resource to understand malware 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. Our experimental results show that some vendors are highly influenced by all other vendors and cautions should be taken to use these detection results.

REFERENCES

[1] Anubis and Wepawet Discontinued. http://anubis.iseclab.org/.

[2] Malwr. https://malwr.com/.

[3] VirusTotal. URL: https://www.virustotal.com/.

[4] Vxstream. https://www.hybrid-analysis.com/.

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

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

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

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

[9] 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. doi: 10.1145/1718487.1718518. URL http://doi.acm.org/10.1145/1718487.1718518.

[10] 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.

[11] 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. doi: 10.1145/1806799.1806871. URL http://doi.acm.org/10.1145/1806799.1806871.

[12] 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.

[13] 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.

[14] 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. doi: 10.1145/2661136.2661148. URL http://doi.acm.org/10.1145/2661136.2661148.

[15] 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/.

[16] 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. doi: 10.1145/2594291.2594321. URL http://doi.acm.org/10.1145/2594291.2594321.

[17] 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. doi: 10.1145/2676726.2677009. URL http://doi.acm.org/10.1145/2676726.2677009.

[18] C. S. Solutions. VirusTotal Access to be Limited: Google. URL: https://antivirus.comodo.com/blog/computer-safety/virustotal-access-to-be-limited-google/.

[19] 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. doi: 10.1145/2967360.2967367. URL http://doi.acm.org/10.1145/2967360.2967367.

[20] 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.

[21] Wikipedia. Mean absolute error. URL: https://en.wikipedia.org/wiki/Mean_absolute_error, .

[22] Wikipedia. Mean squared error. URL: https://en.wikipedia.org/wiki/Mean_squared_error, .

[23] 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.

[24] 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. doi: 10.1109/SP.2012.16. URL http://dx.doi.org/10.1109/SP.2012.16.