JABBIC lookups: a backend telemetry-based system for malware triage

Octavian Ciprian Bordeanu[1], Gianluca Stringhini[2], Yun Shen[3], and Toby Davies[1]

[1] University College London, London WC1E 6DH, UK
octavian.bordeanu.16@ucl.ac.uk, toby.davies@ucl.ac.uk
[2] Boston University, MA 02215, US
gian@bu.edu
[3] NortonLifeLock Research Group
yun.shen@nortonlifelock.com

Abstract. In this paper, we propose JABBIC lookups, a telemetry-based system for malware triage at the interface between proprietary reputation score systems and malware analysts. JABBIC uses file download telemetry collected from client protection solutions installed on end-hosts to determine the threat level of an unknown file based on telemetry data associated with files already known to be malign. We apply word embeddings, and semantic and relational similarities to triage potentially malign files following the intuition that, while single elements in a malware download might change over time, their context, defined as the semantic and relational properties between the different elements in a malware delivery system (*e.g.*, servers, autonomous systems, files) does not change as fast. To this end, we show that JABBIC can leverage file download telemetry to allow security vendors to manage the collection and analysis of unknown files from remote end-hosts for timely processing by more sophisticated malware analysis systems. We test and evaluate JABBIC lookups with 33M download events collected during October 2015. We show that 85.83% of the files triaged with JABBIC lookups are part of the same malware family as their past counterpart files. We also show that, if used with proprietary reputation score systems, JABBIC can triage as malicious 55.1% of files before they are detected by VirusTotal, preceding this detection by over 20 days.

Keywords: Malware triage · Word embeddings.

1 Introduction

The anti-malware industry is confronted by the challenge of identifying malware in large amounts of telemetry data from files downloaded by end-hosts using their client protection solutions [31, 38]. Typically, antivirus software collect and transmit telemetry data from a large number of client machines when users download files that are potentially malicious and unknown to the proprietary security vendor. Since security vendors do not possess unknown downloaded

binaries but only telemetry reports, they rely on file reputation score systems to prioritize and collect those binaries that require utmost attention. At this stage, however, malware analysts need to make decisions as to which files should be prioritized for collection and analysis based only on telemetry data since state-of-the-art triage systems require the downloaded binaries [11, 37, 18, 19].

A widely used solution is built upon file reputation score systems. Generally, antivirus (AV) engines rely on a backend system which assign reputation scores to unknown files. Hash values of unknown files that are found in the backend system are assigned reputation scores accordingly. When a hash value is not found in the backend system, the AV engine sends back on-device activity data. The backend system verifies if the data matches any predefined behavioral rules and returns a reputation score. Lastly, if the backend system is uncertain then a negative reputation score is arbitrarily assigned. To this end, we develop the JABBIC (judge a book by its cover) triage system which uses download contextual information when looking up a file. This way, the backend reputation score system is provided with additional clues about the extent to which an unknown file can be deemed as malicious.

JABBIC runs at the interface between file reputation score systems and malware analysts, and leverages download telemetry data to match unknown files to known malicious files such that the semantic and relational similarities between their associated telemetry data is maximized. The match itself is accompanied by a confidence score which, if above a set threshold, indicates whether an unknown file and the known malicious file it has been matched with are likely to belong to the same malware family or be siblings by parent files, file naming patterns, signers, or a combination of these. Malware analysts are thus able to better guide their decisions as to which unknown files with negative reputation scores, potentially in the order of millions, to collect and analyze first.

JABBIC uses word embeddings which are an approach developed within natural language processing (NLP) to capture semantic and relational properties between words [24]. Words are projected to a latent space - an n-dimensional vector space - in such a way that vectors that are close in latent space have corresponding data points that have some degree of relatedness [23]. Word2Vec [24], a word embedding algorithm, takes as input a corpus of natural text and, given a window size relative to each sentence, outputs a vector representation for each unique word which reflects its meaning. We adapt this idea by applying the approach to abstract 'sentences' which describe file download events, the 'words' of which refer to characteristics of the download telemetry data (such as ASes, network and host IDs, domain names, and URL paths).

2 Related Work

Researchers have developed state-of-the-art systems that show great promise on detecting malware using word embeddings. The authors in [11] proposed a methodology that used Word2Vec and TF-IDF algorithms to embed API *syscalls* functions from malware binaries. These embeddings captured the infection be-

havior of malicious files and were used to cluster the malware samples into families. In [37], authors presented an Android malware detection system, called ANDRE, which utilized the raw labels from antivirus vendors, meta-info of binaries, and insights from source code analysis to learn latent representations of labeled malware samples. Clustering and deep learning were then used to assign malware families to weakly labeled samples based on how similar they were to strongly labeled samples. Scalable triage malware triage systems that have also shown promising results include BitShred [18], MAST [8], and SigMal [19], all of which rely on binary analysis. Nevertheless, these state-of-the-art methods require the binary files and rely on a sequence of static and/or dynamic analysis stages. Such detection systems are thus not suitable for the telemetry-based detection of malware, particularly when it is uncommon for client protection solutions installed on end-hosts to send a copy of the downloaded binary to antivirus vendors.

3 Background and Motivation

JABBIC is positioned at the interface between a proprietary reputation score system – widely used by security companies to flag potentially malicious files using telemetry data – and malware analysts who decide whether files should be submitted for further analysis (Fig. 1). When a file is downloaded, the antivirus software sends telemetry data to a reputation score system that is proprietary to the vendor. The telemetry data of unknown files with negative reputation scores are forwarded to JABBIC which then searches for known malicious files whose telemetry data are similar both semantically and relationally to that of the unknown files. The unknown and known malicious files, their reputation scores, and the confidence score with which JABBIC matched them, are sent to the malware analyst who then decides whether to initiate a file collection request to the end-host. As shown later in this paper, the confidence score quantifies the likelihood that an unknown file and a known malicious file found by JABBIC share the same malware family.

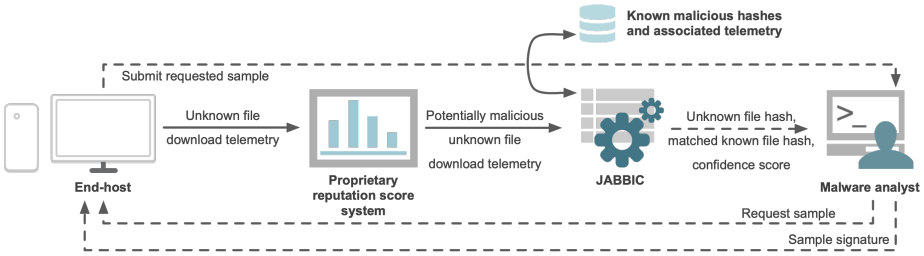

Fig. 1: Improved triage of unknown files using JABBIC at the interface between proprietary reputation score systems and malware analysts.

Table 1: JABBIC triage report sample based on telemetry data.

Row index	Query file (unknown)			Match file (known)			JABBIC confidence score
	Hash	Malware family	Risk given reputation score	Hash	Malware family	Risk given reputation score	
1	AF4EF...	convertad	low	01A28...	convertad	high	0.85
2	F3841...	installcore	medium	DF20D...	installcore	high	0.76
3	1C233...	yakes	low	90CDC...	yakes	high	1
4	D6483...	opencandy	low	1B487...	downloadadmin	low	0.56
5	38F6E...	swisyn	high	27A50...	ardamax	low	0.49

Prior to the application of JABBIC, the reputation score provides the only means to rank the risk (and therefore priority) of unknown files (see Table 1). However, the matching of these files with known malicious files (with high confidence) suggests that this implied ranking is misleading. The low-risk query files, Table 1 at row indices 1, 3 and 4, are not only matched by JABBIC with known files having high risk levels but also the same malware families. These files would thus be incorrectly given the lowest priority which can lead to significant detection delays. Furthermore, file risk levels based on reputation scores are not necessarily representative of how malicious they are. For example, the unknown and known files in Table 1 with the same malware family have a very large discrepancy in their risk levels despite performing similar malicious activities. Therefore, we propose that the triage process is based on the JABBIC scores once files are flagged as suspicious by the reputation score system. JABBIC scores are also meaningful when compared to reputation scores. For instance, scores of at least 0.6, a threshold which we set in Section 6, indicate that two matched files are likely to belong to the same malware family or have some form of sibling relationship, as shown in Table 1.

4 Data

We leverage a dataset from NortonLifeLock's data sharing platform. The telemetry used in this study does not contain any personal identifiable information. The dataset contains information on 33,188,789 download events for the entire month of October 2015, and include the SHA2 string of the downloaded file, the SHA2 string of the parent file that initiated the download, the IP of the file host, the URL of the file host, and the time and date the file was first observed by NortonLifeLock. The data also include file reputation scores denoting low, medium, and high-risk levels. We augmented this dataset with autonomous system (AS) information. The AS number for each download IP was identified using IP to AS mappings published on University of Oregon Route Views Archive [2].

4.1 Ground truth labels

We identified 115,705 query file hashes that were downloaded solely on 31 October 2015 to ensure that neither JABBIC nor the baseline methods, which we discuss in subsection 8.4, had any prior information about these files. For each query file, we used JABBIC to search its match file in every previous day of the

same month; thus each query file has 30 matches, one for each previous day, resulting in 515,571 unique matches across all days. This allows us to evaluate the performance decay of JABBIC over time. Query files can have the same matches across multiple days and two or more query files can have the same matches for one or more days. Similarly, we used the baseline methods to search matches for query files among known files downloaded on 1 October 2015. We limit the search to one day for the baseline methods because we want to compare their performance to that of JABBIC when identifying matches among known files downloaded 30 days prior to when the query files were downloaded. The query file hashes and the returned match hashes were then searched on VirusTotal [3].

Table 2: Query hashes and corresponding match hashes that were identified by Jabbic and variant baselines, found on VirusTotal and labeled by AVClass with a malware family.

Matching method	# hashes	% found on VT	% labelled by AVClass
Query files	115,705	6.71	3.73
Jabbic* matches	515,571	15.34	7.4
Skip-gram** matches	3,012	27.08	12.71
CBOW** matches	4,454	29.88	14.51
fastText** matches	11,618	17.79	9.96
TF-IDF** matches	15,586	22.22	12.38
Bloom filters** matches	25,026	18.57	10.12
LSH** matches	25,099	18.18	9.98
Dropper files	45,567	66.83	63.82

* Unique match and dropper hashes were searched for all 30 days.
** Variant baselines whose performance was evaluated based on their ability to identify match files downloaded on 1 0ctober 2015. For this reason, the numbers for unique match and dropper hashes are much smaller when compared to those reported for JABBIC.

A breakdown of how many queries and unique matches were found as malicious on VirusTotal and labeled with a malware family is shown in Table 2. We used the AVClass labeler[30] with the VirusTotal scanning reports to retrieve their malware families. We also searched the dropper hashes of queries and their matches whose malware families were different but had JABBIC scores within the threshold set in Section 7. This allowed us to determine whether a query file and its match were dropped by files with the same malware family despite the query and the match having different malware families.

5 Methods

5.1 Embedding Download Events

Word2Vec is an approach which can be applied to data which is structured as a sequence of 'words'; typically formed into sentences. Unlike natural language, in

which words are linked by semantics and sentences arise naturally, in order to apply it in the present context we need to artificially formulate a sentence-like representation of the file download information. We conceptualized a download event as a structured fragment of information of the form *an **AS** that owned an **IP address** facilitated the download of a **file** from a **domain** at a particular location given by the **URL path***. Each download event is therefore represented as the sentence *(AS, network ID, host ID, file SHA2, domain, URL path)* before being fed to a Word2Vec model. The structure of the sentence also preserves the hierarchical structure of the malware delivery infrastructure: **(1)** the domains and URL paths form the full file host URLs, **(2)** ASes own the file host IPs and hence the latent relationships among host IPs are better captured during the training; and **(3)** the position of the file hashes in the sentence ensures that the latent relationships between host IPs and domains are also indirectly preserved due to the chosen window size.

Choice of Word2Vec model There are two main types of Word2Vec model – Skip-gram and CBOW – each with different properties. We choose Skip-gram for training word embeddings because **(1)** the aim is to predict the context of a given word rather than predict the word given a context (*e.g.*, predict context given a downloaded file, IP, URL, or AS as input rather than vice versa), **(2)** Skip-gram performs better with infrequent words [26], which is advantageous considering the high cardinality of file hashes (*e.g.*, there are 968,174 unique file hashes for the 1 October 2015 dataset with 1,129,239 file hashes, meaning that each file appears in the dataset 1.17 times on average), and **(3)** Skip-gram does not require large amounts of data to produce high quality embeddings when compared to CBOW [26]. The weights in Word2Vec models are adjusted using negative sampling [25]. The recommended negative samples is between 5 and 20 for small training datasets, and 2 to 5 for large datasets [25]. We chose negative sampling with 5 negative samples to update the model weights. Subsampling of frequent words and discarding of rare words are omitted to avoid both removing valuable file hashes, IP addresses, domain names, and URL paths from the dataset and reducing the vector space within which the search for the local match is carried out.

Temporal granularity NortonLifeLock collected and aggregated the download events telemetry on a daily basis. We choose to retain this temporal slicing and train a separate Word2Vec model for each day. JABBIC lookups can be carried out sequentially or concurrently on multiple days. A daily granularity allows us to limit the search space to most recent download events and thus increase training and lookup speeds; for instance, we find that the strongest matches were identified a day prior to that of query files and hence there is no need to increase the search space to an entire month. Lastly, in real-world settings, training a separate Word2Vec based model on a daily basis minimizes the update delay of the search space.

Training hyperparameters We set the starting learning rate (α) at its default value of 0.025 and the number of epochs at 10. How to choose a suitable embedding size for Word2Vec (*i.e.*, the dimensionality of the latent space) is the subject of much debate [35]: values between 100 and 300 are usually recommended in the literature, while a guideline introduced by Google is the fourth root of the number of categories (*e.g.*, unique words in vocabulary) [4]. Here we adopt the latter approach: the average vocabulary size across all 30 days of October 2015 is 1,111,775.29, and the average fourth root is 32.44, and so we chose an embedding size of 32. A further choice is the window size to use. Relative to the size of each sentence, larger windows capture topic-specific information about words, while smaller windows capture more functional (*i.e.*, syntactical relationships) information [22]. We set the window size to 2 since we are interested in learning syntactical relationships between words.

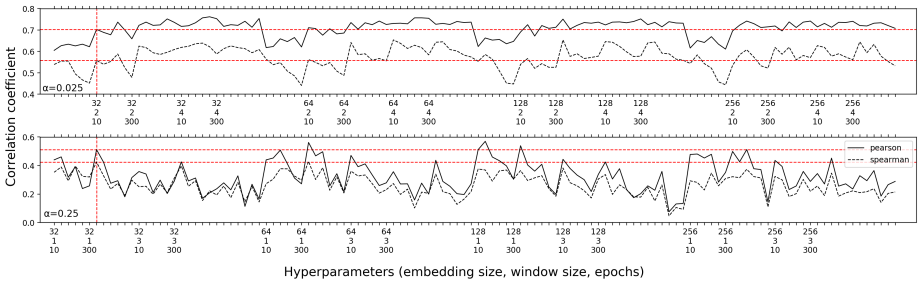

Fig. 2: Correlation values between calculated relatedness scores and cosine similarities, across all pairs of file hashes, for a range of hyperparameter values.

We tested the sensitivity of word embeddings to these choices by training a Word2Vec model for 1 October 2015 data, containing 1,129,239 download events, for all combinations of the following hyperparameter values: *embedding size*: 32, 64, 128, 256; α: 0.025, 0.25; *window size*: 1, 2, 3, 4, 5; and *epochs*: 10, 20, 30, 100, 200, 300. We built a dataset that contained file hash pairs together with an assigned similarity that measured the relatedness between them. The relatedness score was calculated using the R/O score, described in subsection 5.2, such that two file hashes that shared similar contexts had higher scores and vice versa. We calculated these scores for 1000 observations that contained the top 200 most downloaded files. We then calculated the Pearson and Spearman [14] correlation coefficients and associated two-tailed p-values between the relatedness scores and the cosine similarities across all pairs of file hashes. A similar approach was used in [28] and [12] with datasets containing 999 [15] and 200 word pairs, respectively.

The accuracy of word embeddings using α*=0.025*, *embedding size = 32*, *window size = 2*, and *epochs = 10* was fairly high as indicated by the Pearson and Spearman correlation coefficients of 0.703 and 0.558, respectively, both with two-tailed p-values lower than 0.001 (Fig. 2). The highest correlation coefficients

were obtained using α=*0.025*, *embedding size = 128*, *window size = 3* and *epochs = 10* (*r=0.751, rho=0.654, p<0.001*) and α=*0.025*, *embedding size = 32*, *window size = 4* and *epochs = 200* (*r=0.762, rho=0.622, p<0.001)*. However, these correlation coefficients were only marginally higher than those obtained with the hyperparameters we initially proposed. We chose to trade off this small difference in embedding quality for highest training speeds; smaller embedding, window size, and number of epochs meant faster training speeds.

5.2 Local Matching

To identify matches for a given input file, we adapt the search method developed by Zhang *et al.* [36] that finds terms in previous time slices that are semantically closest to a given input term in the present time slice. The method of Zhang *et al.* [36] was designed for querying archives and collections of past documents.

Consider a base vector space, which is the set of vectors trained on the data containing query files (*e.g.*, Word2Vec model trained on 31 October 2015 data), and a target vector space, which is the set of vectors trained on the data where the local match searching is carried out (*e.g.*, Word2Vec model trained on 1 October 2015 data). Given a query file q (unknown file represented by its hash) in base vector space, the aim is to find its best representative file hash m in target vector space (known malicious file represented by its hash). We formulate this problem as follows.

Let $S_{sim}(q, m) = cos(M \cdot q, m)$ be the across-time *semantic similarity* between a query file q in base vector space and a local match file m in target vector space, where M is the transformation matrix which projects the base vector space to the target vector space. Let $F_q = (V, \prod)$ be a download event associated with query file q, defined by a set of download event elements V – such that $V = q \cup C$, where $C = \{c_1, c_2, c_3, ..., c_u\}$ is a set of u context words (*e.g.*, ASes, network and host IDs, domains, and URL paths) – and a set of associations $\prod = \{\pi | \text{association between } q \text{ and each context word in } C\}$. Similarly, let $F_m = (V^{'}, \prod^{'})$ be a download event associated with file m in target vector space, with download event elements $V^{'}$ and associations $\prod^{'}$ defined equivalently. The objective is to find the file m^* such that F_{m^*} in target vector space is most similar to F_q in base vector space. *Relational similarity* is then defined as $R_{sim} = cos(M \cdot (q - c_i), (m - c_i^{'}))$, where $q - c_i$ is the subtraction between the vector of query file q and the vector of its context word c_i, and $m - c_i^{'}$ is the subtraction between the vector of file m and the vector of its context word $c_i^{'}$.

The local similarity LS between F_q and F_m is calculated as per equation 1, where v_0 and $v_0^{'}$ denote the vector representations of query file hash q in base space and its counterpart file hash m in target space, respectively; v_i and $v_i^{'}$ denote the vector representations of all i and $i^{'}$ delivery infrastructure elements (*e.g.*, file hash, AS, network ID, host ID, second-level domain, and host URL) of q and m, respectively; and u is the number of context words. LS is a weighted combination of semantic and relational similarity, with higher λ values corresponding to higher weighting for semantic similarity.

$$LS = \lambda \cdot \frac{\sum_{i=0}^{u} S_{sim}(v_i, v_i^{'})}{u} + (1-\lambda)\frac{\sum_{i=1}^{u} R_{sim}((v_0, v_i),(v_0^{'}, v_i^{'}))}{u} \quad (1)$$

The best local match of query file hash q is given by $argmax_m(LS(q,m))$, which returns the file hash m^* in target space that is most representative of the query file hash q semantically, relationally, and by delivery infrastructure.

Offline training, whereby words are grouped into time bins and then a Word2Vec model is separately trained for each slice, is useful for capturing the semantic content of words at discrete intervals. However, each embedding is specific to its corresponding time bin, which means that embeddings of a particular word across different time bins cannot be meaningfully compared. Vector alignment is required in order to compare vector representations of words in different vector spaces. For vector space alignment, we use orthogonal Procrustes, an assumption-free method [34], to project the base vector space onto the target vector space.

The Ratcliff/Obershelp (R/O) pattern matching algorithm compares two strings and outputs a value between 0 and 1, where 1 denotes a complete match between the two strings and 0 indicates no substrings in common. The iterative process by which the R/O algorithm works is described in [16]. In our context, we use the R/O score to quantify the overlap between the contextual information of pairs of query and match files in a string-like sense. Put simply, this measures the similarity between the contexts of the two files. As shown below, this can be used as a confidence measure that allows us to quantify the certainty with which JABBIC can associate a local match file with a given query file.

6 JABBIC Lookups **Architecture**

The system architecture is illustrated in Fig 3. ❶ Determine the temporal granularity by which telemetry data from known malicious files is to be split (*e.g.*, by day). ❷ Train separate target vector-based models for each day, and then store them into a lookup database. This allows for the selection of a specific past timeframe within which searching for a local match is carried out. ❸ Train a base vector space for the day which contains the query files; that is, those files that have not been downloaded in any previous timeframes. ❹ Select a target vector space within which local matches are to be searched given the query files from the base vector-based model (*e.g.*, find local match files downloaded on 1 October 2015 given query files downloaded on 31 October 2015). ❺ Align base and target vector spaces to a common vector space in which cosine similarities can be computed. ❻ Identify query files as those files that have not been seen before. ❼ For each query file in the base vector space, find its most representative match file in target vector space from a previous timeframe and then assess the confidence with which these files are matched using the R/O score. After lookups are completed, the current base vector space can then be added to the lookup database, which can then be queried when local matches are searched for queries downloaded the next day.

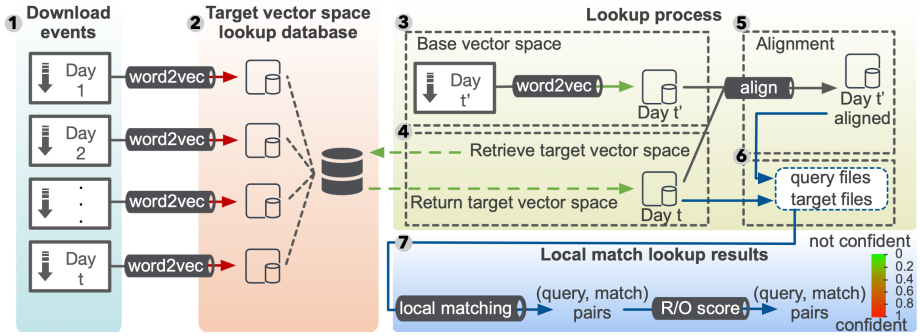

Fig. 3: JABBIC lookups architecture.

7 JABBIC Confidence Score Threshold

In this section, we assess the triage performance of JABBIC on download events from October 2015 for which malware family labels, parent files, and file names are available from both the VirusTotal scan reports and our datasets. First, we identify 115,705 unique hashes of files downloaded solely on 31 October 2015 to ensure that no prior information about these files is known to JABBIC in the lookup process. For each of these files, JABBIC lookups are used to search their match file hashes in every previous day. This allows us to assess the triage accuracy when lookups are performed on data up to thirty days prior to when query files were downloaded. Lastly, we relate triage accuracy to the corresponding R/O scores and then set an optimal confidence score threshold. Unknown query files matched with malicious known files to a confidence R/O score above the set threshold are very likely to be related by malware family.

7.1 Matched files and malware families

On average, across all days of October 2015, 81.18% of unique and labeled *(query, match)* pairs have the same malware family irrespective of the R/O score. The percentage of *(query, match)* pairs with the same malware family increased as the daily proportion of *(query, match)* pairs with R/O scores in ranges $[0.6, 0.8)$ and $[0.8, 1]$ also increased while peaks in R/O scores in range $[0, 0.6)$ led to a percentage decrease (Fig. 4). The highest density of *(query, match)* pairs with the same malware family was associated with R/O scores in range $[0.6, 1]$; 89.43% and 87.43% of *(query, match)* pairs with R/O scores in ranges $[0.8, 1]$ and $[0.6, 0.8)$, respectively, had the same malware family, while the percentage is much lower for scores in the range $[0, 0.6)$. Thus, for any given *(query, match)* pair, a higher R/O score increases the probability that the malware family of the query file is the same as that of its match. The certainty with which JABBIC found matches with the same malware family as that of their corresponding query files was highest up to seven days prior to when queries were downloaded; on average,

85.68% of the matches found in these days had the same malware family as their counterpart query files compared to an average of 79.81% across previous days.

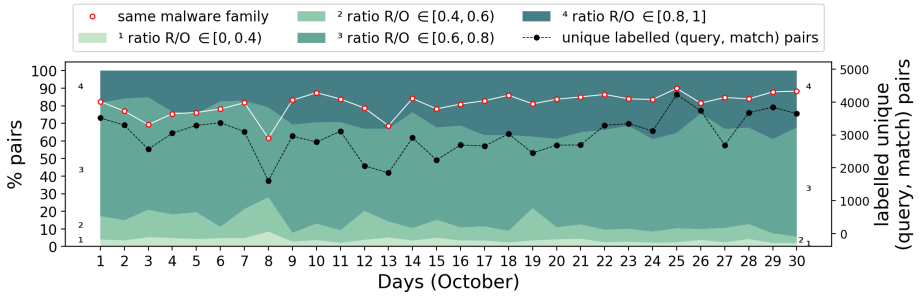

Fig. 4: Percentage of query file hashes and their match file hashes from each previous day of October 2015 with the same malware families. The plot also shows the proportion of pairs falling into each R/O band.

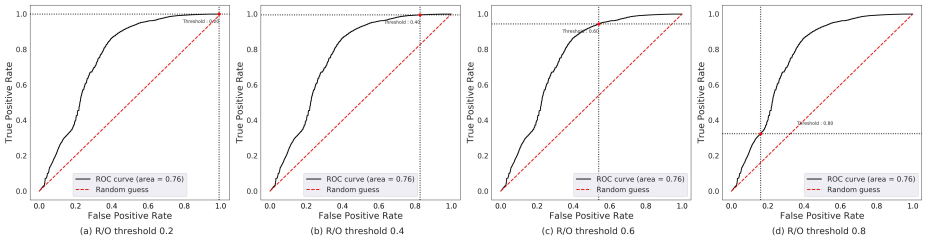

Fig. 5: ROC curves for the R/O score metric ($R/0 \in [0, 1]$ and thresholds 0.2, 0.4, 0.6, 0.8).

Next, we evaluate the reliability of setting the confidence score threshold at 0.6 when deciding whether a query file (unknown) and a match file (known as malicious) are likely to belong to the same malware family. This is particularly useful when, despite missing malware family labels, analysts can rely on the assumption that a query file and a match file, which are paired by JABBIC with a confidence score of at least 0.6, are also likely to have the same malware family. Thus, the query file can be prefiltered for collection and analysis by more than just its reputation score which, as we show in the next section, is not a reliable risk indicator. First, we calculated the weighted precision, recall, and F1 scores across all classes (*i.e.*, malware families) to evaluate the average performance of JABBIC when matching query and known files by malware families. Second, we assessed how the certainty with which JABBIC assigned malware families to previously unseen files changed based on different R/O score thresholds us-

ing the receiver operating characteristic (ROC) curve and area under the ROC curve (AUC) metrics. The matching process is framed as a binary classification problem, with each *(*query, match) pair being assigned label 1 if their families are the same, and 0 otherwise.

Table 3: JABBIC performance at matching unknown and known malicious files by malware families for different R/O confidence score ranges.

R/O range	Weighted precision	Weighted recall	Weighted F1 score	Labeled pairs
$R/O \in [0, 1]$	0.87	0.82	0.84	89,158
$R/O \in [0, 0.6)$	0.5	0.36	0.40	11,355
$R/O \in [0.6, 1]$	0.92	0.89	0.89	77,803

For a second stage triage, we aim for the highest true positive rate – identify as many queries as possible that have the same malware families as their matches – while accepting a higher false positive rate since the cost of collecting and analyzing mismatched unknown files has no significant practical implications. The initial triage of files as either benign or potentially malign has already been done by the reputation score system. Thus, the only implication of JABBIC matching a small proportion of unknown files to malicious known files with different malware families, is that the unknown files are collected and analyzed earlier than they otherwise would. For this reason, we accept a confidence R/O score threshold of 0.6 as optimal, which not only correctly matches the highest proportion of unknown query files to known malicious files by malware families but also achieves the lowest false positive rate (Fig. 5). Most importantly, however, is that 89% of matched files with confidence scores in range $[0.6, 1]$ had the same malware families (F1 score of 0.89 in Table 3). Therefore, malware analysts can rely on the assumption that if a query file is matched to a malicious known file with a confidence score of at least 0.6, then the query file is most likely to be related to the malicious known file by malware family.

7.2 Lambda Parameter

The λ parameter value was set to 0.5, meaning that semantic and relational similarities were given equal weights in the match lookup process. In Table 4 we also show the percentage of *(query, match)* pairs with R/O confidence scores falling in different ranges had 0.2 and 0.8 been used instead. We find that the confidence scores of *(query, match)* pairs were not sensitive to the three λ values; that is, the proportion of *(query, match)* pairs with confidence scores in ranges $[0.6, 8)$ and $[0.8, 1]$ did not differ significantly as the λ parameter value was changed. One reason for this could be that there was a relatively high number of files that were good candidates as local matches for a each query file. Despite the lack of sensitivity of local match lookups to the three λ parameter values, it appeared that setting λ to 0.5 achieved the highest proportion of R/O scores in ranges $[0.6, 0.8)$ and $[0.8, 1]$, although just marginally.

Table 4: Matching R/O scores for different λ parameter values.

λ	$R/O \in [0, 0.2)$	$R/O \in [0.2, 0.4)$	$R/O \in [0.4, 0.6)$	$R/O \in [0.6, 0.8)$	$R/O \in [0.8, 1]$
0.2	1.19%	9.78%	18.90%	41.84%	28.29%
0.5	1.15%	8.89%	18.85%	41.91%	29.20%
0.8	1.02%	8.68%	19.47%	41.83%	29.00%

8 Evaluation

Here we begin by evaluating how the confidence with which JABBIC assigns a match to a query file changes across time. This allows us to determine the optimal search timeframe prior to when query files were downloaded. We then evaluate the extent to which JABBIC can improve the prioritization for collection and analysis of unknown files already prefiltered by reputation scores. We then compare the detection times by VirusTotal with those expected from JABBIC if used as a second stage triage system. Lastly, we compare the triage performance of JABBIC with n-gram, co-occurrence, and set-based matching baselines.

8.1 Longitudinal decay of triage performance

The R/O scores of local matching provide a measure of the confidence associated with the triage of a given query file. On this basis, the 115,705 query files, downloaded solely on 31 October 2015, can be triaged with very high confidence based on files downloaded up to 7 days before; 35.17% and 41.53% of R/O scores are in ranges $[0.6, 0.8)$ and $[0.8, 1]$, respectively (Fig. 6). We recommend that the confidence with which a query and match file are paired by malware family is evaluated based on the following confidence levels: high ($R/O \in [0.6, 1]$) and cannot tell for $R/O \in [0, 0.6)$. We justify this recommendation based on the results from subsection 7.1.

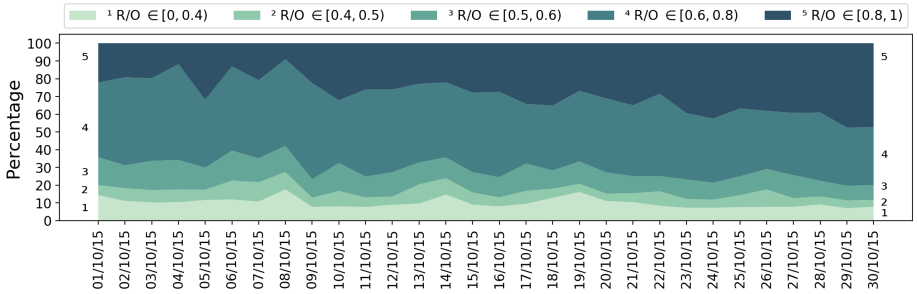

Fig. 6: Confidence decay when query files downloaded on 31 October 2015 are triaged based on matches from every previous day.

8.2 Reliability of file reputation scores

File reputation scores are not consistent across files belonging to the same malware families. For example, 616 files with malware family *zusy* had reputation scores denoting low, medium, and high levels of risk. To understand the scale of this observation, we look at the reputation scores of files associated with each malware family. We use 87,532 labeled unique file hashes and their reputation scores, including files that are neither queries nor matches, which span across 2,460 malware families. For each malware family, we count the number of files that fall within it and calculate the standard deviation of their reputation scores. The higher the standard deviation of reputation scores of files belonging to a malware family, the lower the triage reliability of those scores since these files are expected to pose similar threat levels and thus have reputation scores as close as possible. We find that for 42.13% of labeled files, belonging to 21.22% of malware families, the standard deviation of reputation scores is 30. For 84.28% of labeled files, belonging to 29.63% of malware families, the standard deviation of reputation scores is 20. Files with low-risk levels are thus not necessarily less suspicious than those with higher risk levels.

Next, we exemplify how JABBIC can prefilter unknown files with negative reputation scores. Consider that the current date is 31 October for which a large volume of telemetry is received by malware analysts who then identify 115,705 unknown files flagged as suspicious by the reputation score system. We also hold telemetry data of files which were downloaded the previous day and known to be malicious. For each unknown file, JABBIC lookups are then carried out to identify their local matches from the previous day. JABBIC results show that 80.44% of *(query, match)* pairs have R/O scores of at least 0.6, meaning that they are most likely to have the same malware family.

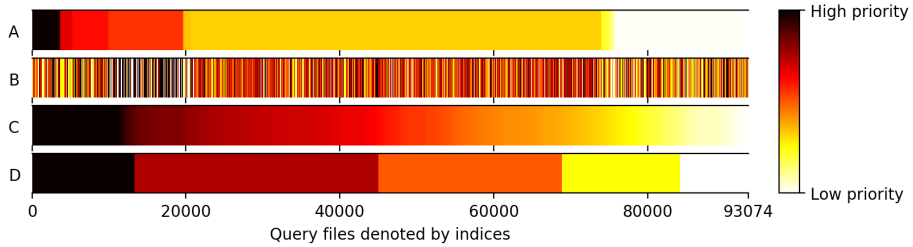

Fig. 7: Triage of query files with R/O scores in range $[0.6, 1]$. Prioritization for collection and analysis of query files. **A:** Reputation scores of query files from highest to lowest priority); **B:** Confidence R/O scores of query files ordered by reputation scores; **C:** Query files ordered by confidence R/O scores (1 for highest priority and 0.6 for lowest priority) at low granularity; **D:** Query files ordered by confidence R/O scores at high granularity ($[0.9, 1]$, $[0.8, 0.9)$, $[0.7, 0.8)]$, and $[0.6, 0.7)$)

In Fig. 7 we illustrate the prioritization of files using the reputation score system and JABBIC. Query files, sorted in ascending order by their reputation scores, are matched with known malicious files whose reputation scores do not reflect similar levels of maliciousness despite being very likely to have the same malware family (heatmaps A and B). Most notably, query files with lowest priority would have been prioritized for collection and analysis much earlier judging by the risk levels of their match files. Similarly, most of query files with highest priority for collection and analysis are matched with known files having much lower priority. Heatmaps A and B indicate what we have already shown: *(query, match)* pairs have significantly different levels of risk despite belonging to the same malware families. Therefore, file reputation scores are at most reliable for an initial triage of unknown files as either potentially malign or benign but not for further prioritization of malign files. In heatmaps C and D from Fig. 7, query files are sorted in descending order by the confidence R/O scores with which JABBIC paired them with known malicious files.

8.3 JABBIC triage vs. VirusTotal

VirusTotal reports both the date when a file was submitted for analysis and the date it was detected as malign. We show the number of days elapsed since query files were submitted to VirusTotal until they were detected as malicious. Out of 113,642 unique query files which had at least one match in any previous day with an R/O score in range $[0.6, 1]$, 7,320 (6.44%) were submitted for scanning on VirusTotal by third parties and found malicious (include singletons which are malignware not belonging to a malware family). The detection times of these files by VirusTotal are shown in Table 5. More than half of the query files found as malicious on VirusTotal were detected over 20 days after they were submitted for scanning. We also include the detection times for all hashes that we scanned and found on VirusTotal to show that similar detection times are relevant beyond the sample of query files (Table 5). JABBIC could have triaged and flagged as suspicious over half the query files in less than a day if not hours, depending on the search space size, when compared to over 20 days as per VirusTotal.

Table 5: Number of days until detection by VirusTotal.

	0 days	1-9 days	10-19 days	>= 20 days
Detected query hashes (*N=7,320*)	33.10%	8.31%	3.13%	55.10%
All detected hashes (*N=216,582*)	31.53%	7.41%	3.62%	57.12%

8.4 Variant Baselines

State-of-the-art triage systems require binary-level analysis for which reason we are not able to compare their performance against that of JABBIC which

is a telemetry-based triage approach. Instead, we compare the performance of JABBIC with Skip-gram and CBOW, fastText [7], TF-IDF [29], Bloom filters [21], and LSH [9] which are commonly used in malware detection systems based on string-level similarity of opcodes, network packet payloads, application programming interface (API) method calls, instruction sequences, and binary strings [17, 33, 32, 5, 6]. We used fastText with 3 to 6 grams, meaning that the embedding of each download event component was represented by a bag of 3 to 6 n-grams. For LSH and Bloom filters we converted each download event into 5-shingle sets. For example, download event, identified by its file hash `CA756...`, '`China169 Backbone 218.58.225.7 dxdown.rilibiao.com.cn/cx/2015092118/`' is shingled into 5-character substrings '`China`', '`hina1`', '`ina16`', and so forth. Given the small number of 5-character shingles (*e.g.*, not exceeding 50 shingles) per download event, the size of each Bloom filter was set to 1000 with 7 hash functions to ensure a false positive rate not higher than 0.001, as confirmed by this online Bloom filter calculator [1]. As per Table 6, the overall performance of JABBIC not only surpassed that of all baseline methods but was also much faster than those whose performance was relatively close, particularly set-based matching with LSH and Bloom filters.

Table 6: Triaging performance of Jabbic compared with n-gram, co-occurrence, and set-based matching models.

Matching method	*(a)* Individual performance based on all labeled (query,match) pairs		*(b)* Performance comparison based on (query, match) pairs with queries for which both Jabbic and the baseline found a labeled match				Searching time (≈ hours)
	% same malware family	N	% same malware family (Jabbic)	N	% same malware family (baseline)	N	
Jabbic	82.4	3,517	–		–		5
Skip-gram	23.95	1,027	68.10	840	28.04	731	1
CBOW	7.56	1,680	88.41	2,260	11.37	888	1
fastText (3-6 grams)	68.60	1,258	72.38	1,086	73.18	548	1
TF-IDF	74.74	1,437	72.41	1,131	80.56	607	12
Bloom filters (5 grams)	81.26	1,446	83.58	1,657	89.18	647	124
LSH (5 grams)	80.85	1,504	84.14	1,766	89.57	671	310

JABBIC and variant baselines are compared on query files and their matches downloaded on 31 and 1 October 2015, respectively, irrespective of their R/O scores. ***(a)*** The individual performance results are reported based on how many *(query, match)* pairs, out of all labeled pairs, were found to have the same malware family; ***(b)*** Results are reported based on those query files for which both JABBIC and each baseline method have found a match in order to determine their relative performance on the same query files. For example, 348 query files were found in 840 and 731 labeled *(query, match)* pairs identified by JABBIC and Skip-gram, respectively.

9 System Performance

Training Word2Vec models for each day was carried out on a home machine using 5 CPU cores with 2 worker threads per core. The lookup process was carried out in batches of 1,500 query file hashes, required 1 CPU core and approximately 140GB of RAM. The average training and search times are shown in Table 7. The lookup time ought to scale approximately linearly with the number of unique

files in the search space. The search time of 115,705 query files in a sample of 1,070,606 download events (daily average) took close to 5 hours, so even an order-of-magnitude increase should still produce results on useful timescales (and could be mitigated further by the use of multiple machines/cores).

Table 7: System performance assuming an average training/search space size of 1,070,606 download events.

	Machine specifications	Running time
Training	6-core CPU at 2.2GHz, 32GB RAM	$\approx$ 1.3 minutes/model
Lookups	32-core CPU at 2.4GHz, 256GB RAM	$\approx$ 5 hours/115,705 queries

9.1 JABBIC limitations

JABBIC lookups rely on the alignment of one vector space to the other, which requires that some file hashes are found in both vector spaces (*i.e.*, downloaded in both days). If no file hashes are common to both vector spaces, then the alignment of one vector space to another is not possible. However, we found that a sufficiently high number of file hashes were downloaded not only from one day to another but also across multiple days, thereby the alignment of vector spaces should not pose any issues in similar analyses, particularly in the case of malware or potentially unwanted programs (PUP) ecosystems. It is also necessary to know whether file hashes from all previous timeframes are either benign or malicious, otherwise they cannot be used to infer the threat level of a query file.

9.2 Data limitations

The malware ecosystem has changed since the period between 2015 and 2016 for which the results in our analysis are reported. NortonLifeLock [10] reported an 8,500% and 46% increase in file-based coinminer and ransomware infections on endpoint computers in 2017 alone, respectively, when compared to previous years when these types of attacks were not as common. Nevertheless, the means of propagation continue to rely on droppers or a combination of multiple vectors among which droppers are still present. For instance, file-based cryptojacking spreads in a similar fashion as traditional malware and continues to be prevalent [27]. As such, despite the malware ecosystem increasingly shifting towards ransomware and cryptojacking attacks, it is still common that their distribution vectors rely on users downloading the payloads from malicious hosts. For this reason, malware contextual information – such as payload domains, URLs, IPs, and ASes – is as relevant to newer types of attacks that rely on dropper vectors. We thus argue that JABBIC is a useful triage tool as long as file-based malware – where their distribution is leveraged by drive-by downloads and other social engineering techniques – continue to be part of the malware ecosystem landscape.

9.3 Evasion

JABBIC lookups work under the assumption that the context of malicious file download for a given malware campaign presents similarity over time. JABBIC does not make any assumptions about the number of times a file SHA2 has been downloaded. Indeed, it makes no difference whether or not all files hashes are unique as long as some or all delivery infrastructures are being reused. This makes JABBIC less prone to evasion when compared to signature-based or heuristic detection. One way JABBIC could thus be evaded is if malware writers would avoid reusing, either fully or partially, the same delivery infrastructures over periods longer than 7 days. Another way to evade JABBIC is to drop a higher number of benign files using the same delivery infrastructures that they use to distribute malware and PUPs. This would increase the chance that the returned local match is benign, tricking the system into assessing the level of threat of its corresponding query file as inconclusive. However, these evasion techniques are both atypical for the pay-per-install ecosystem and financially infeasible given the scalability of the required pay-per-install services and delivery infrastructures.

10 Conclusion

We proposed a back-end file triage system that operates on top of file reputation score systems used by security vendors to prioritize the collection and analysis of unknown files downloaded by their clients. When compared to state-of-the-art malware triage systems that use binary-analysis, JABBIC can function along client protection solutions installed on end-hosts and using only download telemetry data. We showed that JABBIC is more reliable for the second stage triage of files flagged as potentially malign by the reputation score systems, which allows for better prioritization of files for collection and analysis. Lastly, we showed that the files triaged by JABBIC could have been flagged as suspicious much earlier than online scanning services such as VirusTotal.

11 Acknowledgements

This research was funded by the Dawes Centre for Future Crime at UCL.

References

1. Bloom filter calculator. `https://hur.st/bloomfilter/?n=50&p=&m=1000&k=7`.
2. University of oregon route views archive project. `http://routeviews.org/`.
3. Virustotal. `https://www.virustotal.com/`.
4. Introducing tensorflow feature columns, Nov 2017. `https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.htm`.
5. Yara Awad, Mohamed Nassar, and Haidar Safa. Modeling malware as a language. pages 1–6, 05 2018.

6. Mahmood Azar, Len Hamey, Vijay Varadharajan, and Shiping Chen. *Learning Latent Byte-Level Feature Representation for Malware Detection: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part IV*, pages 568–578. 01 2018.
7. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
8. Saurabh Chakradeo, Bradley Reaves, Patrick Traynor, and William Enck. Mast: Triage for market-scale mobile malware analysis. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 13–24, 2013.
9. Sachendra Singh Chauhan and Shalini Batra. Finding similar items using lsh and bloom filter. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 1662–1666. IEEE, 2014.
10. Gillian Cleary, Mayee Corpin, Orla Cox, Hon Lau, Benjamin Nahorney, Dick O'Brien, Brigid O'Gorman, John-Paul Power, Scott Wallace, Paul Wood, et al. Symantec internet security threat report, 2018. `https://docs.broadcom.com/doc/istr-23-2018-en`.
11. Hugo Leonardo Duarte-Garcia, Carlos Domenick Morales-Medina, Aldo Hernandez-Suarez, Gabriel Sanchez-Perez, Karina Toscano-Medina, Hector Perez-Meana, and Victor Sanchez. A semi-supervised learning methodology for malware categorization using weighted word embeddings. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 238–246. IEEE, 2019.
12. Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. volume 20, pages 406–414, 01 2001.
13. Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31:1–10, 02 1966.
14. Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, 30(2):87–93, 2011.
15. Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41, 08 2014.
16. Ilya Ilyankou. Comparison of jaro-winkler and ratcliff/obershelp algorithms in spell check. *IB Extended Essay Computer Science*, 2014.
17. Jiyong Jang, Abeer Agrawal, and David Brumley. Redebug: finding unpatched code clones in entire os distributions. In *2012 IEEE Symposium on Security and Privacy*, pages 48–62. IEEE, 2012.
18. Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320, 2011.
19. Dhilung Kirat, Lakshmanan Nataraj, Giovanni Vigna, and BS Manjunath. Sigmal: A static signal processing based malware triage. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 89–98, 2013.
20. Malwarebytes Labs. 2020 state of malware report, 02 2020. `https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf`.
21. Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Finding Similar Items*, page 68–122. Cambridge University Press, 2 edition, 2014.
22. Omer Levy and Yoav Goldberg. Dependency-based word embeddings. volume 2, pages 302–308, 06 2014.

23. Yang Liu, Eunice Jun, Qisheng Li, and Jeffrey Heer. Latent space cartography: Visual analysis of vector space embeddings. *Comput. Graph. Forum*, 38:67–78, 2019.
24. Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, 01 2013.
25. Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 10 2013.
26. Marwa Naili, Anja Habacha, and Henda Ben Ghezala. Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science*, 112:340–349, 12 2017.
27. B O'Gorman. Cryptojacking: A modern cash cow. *Internet Security Threat Report, Symantec, září*, 2018. `https://docs.broadcom.com/doc/istr-cryptojacking-modern-cash-cow-en`.
28. Chakaveh Saedi, António Branco, João Rodrigues, and Joao Silva. Wordnet embeddings. 07 2018. `https://www.aclweb.org/anthology/W18-3016/`.
29. Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.
30. Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, 2016.
31. Jack W Stokes, John C Platt, Helen J Wang, Joe Faulhaber, Jonathan Keller, Mady Marinescu, Anil Thomas, and Marius Gheorghescu. Scalable telemetry classification for automated malware detection. In *European Symposium on Research in Computer Security*, pages 788–805. Springer, 2012.
32. Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. Guilt by association: Large scale malware detection by mining file-relation graphs. 08 2014.
33. Ke Wang, Janak J Parekh, and Salvatore J Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *International workshop on recent advances in intrusion detection*, pages 226–248. Springer, 2006.
34. Huiling Xiong, Dapeng Zhang, Christopher J Martyniuk, Vance Trudeau, and Xuhua Xia. Using generalized procrustes analysis (gpa) for normalization of cdna microarray data. *BMC bioinformatics*, 9:25, 02 2008. `https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-25`.
35. Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 895–906, USA, 2018. Curran Associates Inc.
36. Y. Zhang, A. Jatowt, S. S. Bhowmick, and K. Tanaka. The past is not a foreign country: Detecting semantically similar terms across time. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2793–2807, Oct 2016.
37. Yanxin Zhang, Yulei Sui, Shirui Pan, Zheng Zheng, Baodi Ning, Ivor Tsang, and Wanlei Zhou. Familial clustering for weakly-labeled android malware using hybrid representation learning. *IEEE Transactions on Information Forensics and Security*, 15:3401–3414, 2019.
38. Malwarebytes Labs. 2020 state of malware report, 02 2020. `https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf`.