Relationship of Jaccard and Edit Distance in Malware Clustering and Online Identification

by

Shlomi Dolev, Mohammad Ghanayim, Alexander Binun, Sergey Frenkel and Yeali S. Sun

Technical Report #17-05

October 5, 2017

The Lynne and William Frankel Center for Computer Science Department of Computer Science, Ben-Gurion University, Beer Sheva, Israel.

Relationship of Jaccard and Edit Distance in Malware Clustering and Online Identification

Shlomi Dolev* Mohammad Ghanayim* Alexander Binun* Sergey Frenkel† Yeali S. Sun‡

* Department of Computer Science, Ben-Gurion University of the Negev, Israel, {dolev, ghanayim, binun}@cs.bgu.ac.il
† The Institute of Informatics Problems of FRC IC, Russian Academy of Sciences, Russia, fsergei51@gmail.com
‡ Department of Information Management, National Taiwan University, Taiwan, sunny@ntu.edu.tw

***Abstract*—In this paper, we examine the possibility to utilize the well-known approximations of Jaccard metric in order to reduce computational complexity of Edit Distance metric estimation. The scope of our analytical results is the *representing strings* rather than the original (raw) textual data, still in practice we obtained a solid indication that the results can be applied to (raw) strings that have low n-gram repetitions. We formulate inequalities between the Jaccard metric and the Edit Distance, that impose upper and lower bounds on the Edit Distance values in terms of the Jaccard values. We validate our inequality over strings of API call traces where (the small) clusters obtained are refined by applying Edit Distance.**

Jaccard is a measure of similarity between two sets, while Edit Distance is a measure for two strings, such as traces of API calls. The computation associated with creating n-grams and using Jaccard similarity is much more efficient than the computation of Edit Distance (linear versus quadratic time complexity). Thus, our new bounds on the Edit Distance given the Jaccard value are of practical interest. Another new aspect we coped with in our research is the inherent imbalance between malicious and benign API traces that are harvested from the system, as most of the traces are benign. We performed clustering only on the malware traces where each cluster concentrates malware with some specific common essence. The obtained clustering is used with great success in classifying new query traces for being either benign or malware. The traces for our research were obtained from the KVM hypervisor Runtime Execution Introspection and Profiling (REIP) system based on Virtual Machine Introspection (VMI) techniques to profile hooked Windows API calls.

I. Introduction

There is an inherent connection between security and machine learning. For example, cryptography is used by the system administrators and also attackers to avoid leakage of information from the activity (e.g., communication) by means of exhibiting high and/or constant entropy. On the other side, the opponent uses machine learning to detect imperfect entropy hiding for establishing countermeasures.

A sequence of events monitored and recorded during operation can be used to classify and identify risks and threats based on their run time behavior. In this paper, we learn and classify API call traces of programs as a form of behavioral (or dynamic) analysis of malware. API call behavioral analysis has been proposed in the literature. Santos et al. [10] and Islam et al. [5] proposed malware detectors based on API calls in combination with other static features. Younjoon et al. [2] analyzed API calls using DNA sequence alignment algorithms. Gupta et al. [6] clustered hash signatures of API calls to detected the type of malware.

A widely-used approach of classification is based on similarity measurement. Presently various similarity metrics were suggested [8], one of the most discussed is the Edit Distance, namely, the number of edit operations (delete, insert and substitute of a single symbol) required to convert one sequence to the other. Unfortunately, the computational complexity of the Edit Distance is high, with relation to the commonly used Jaccard measure. Moreover, when computing Jaccard measure, one can employ several approximation techniques, such as MinHash, to dramatically speedup the clustering, classification and identification.

We show, to the best of our knowledge, for the first time, that there is a relationship between Edit Distance and Jaccard measure, which can serve as a theoretical base for the estimation of Edit Distance by Jaccard metric.

A clustering of the malicious API call traces, based on Jaccard similarity, is utilized for online classification where the query trace is declared as benign/malicious by its distance from the clusters medoids (see e.g., [4]).

The experiment outcome supports our theoretical results, namely, high quality clustering of malware traces and on-line classification of malware sequences based on Jaccard measure yields high quality results, even though the semantics of the API calls were ignored and their ordering is not explicitly captured by the Jaccard measure as opposed to the Edit Distance measure.

Our analytical results concerning the possibility to use Jaccard distance for Edit Distance estimation, for the sake of clustering sequences and identification of similar sequences, appear in section II. Later in section III we train a malicious/benign classifier and finally in section IV we describe experiments and results.

II. Jaccard Similarity Versus Edit Distance

A useful property of the metric of Jaccard similarity $J_S(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$ of two sets X and Y is the ability to approximate the value of $Jaccard(X,Y)$ by the probability that two uniformly selected members, one from X and another from Y, are hashed (by universal hash function) to the

We thank Jeffrey Ullman for many comments. This research was partially supported by the Rita Altura Trust Chair in Computer Sciences; the Lynne and William Frankel Center for Computer Science; grant of the Ministry of Science, Technology and Space, Israel, and the National Science Council (NSC) of Taiwan; the Ministry of Foreign Affairs, Italy; the Ministry of Science, Technology and Space, Infrastructure Research in the Field of Advanced Computing and Cyber Security, RFBR grant 15-07-05316 and the Israel National Cyber Bureau.

978-1-5386-1465-5/17/$31.00

same value. When Jaccard similarity, or alternatively, Jaccard distance $J_D = 1 - J_S(X, Y)$, is computed among two sets of symbols, X and Y, the order of the symbols in the sets is ignored. Jaccard Distance can be expressed in terms of symmetric difference between two sets, i.e. $J_D = \frac{|X \Delta Y|}{|X \cup Y|}$, where Δ denotes the symmetric of two sets - the set of elements in either of the sets but not in both.

On the other hand, the well-known Levenshtein distance (Edit Distance, ED e.g., [14]) can identify ordering differences, and therefore, better fits scenarios in which the ordering encapsulates important information. It is well-known that if two strings have small Edit Distance, they will have many close (in terms of Hamming distance) substrings whose locations in the respective strings are almost similar. It can be interpreted that Edit Distance reflects the structural differences between two strings while Jaccard reflects the content differences only. $ED(x, y)$ is the minimal number of edit operations (delete, insert and substitute) of (single) symbols done on one sequence x to obtain the other sequence y. Both x and y can be considered as strings that are sets with an ordering of their elements. ED satisfies the conditions of metric functions, namely the triangle inequality (sub-additivity) condition. In the sequel, we use both the notation $ED(x, y)$ and ED to represent the Edit Distance.

However, the computation of ED is time-consuming [15], while there are efficient hashing based linear algorithms for approximating Jaccard distance for large data sets. There are several heuristics to approximate the ED value with efficient computation, however, these heuristics do not guarantee approximation bound on their results [3]. Moreover, the triangular property is not always respected, which in turn, may lead to unsatisfying clustering. Therefore, it would be useful to have the ability of estimating the range of ED values using other easily calculated measures of similarity, e.g., Jaccard, for which there are some efficient approximation algorithms [8].

Jaccard is based on probabilistic arguments, thus, we seek and suggest analogous (ready for probabilistic approaches) normalized forms of the Edit Distance. The most common normalized Edit Distance form in literature is the *Normalized Edit Distance* (NED), which is obtained by normalizing the Edit Distance by the maximal of strings' lengths as follows (e.g., [14], [16]):

$$NED(x, y) = \frac{ED(x,y)}{max\{|x|,|y|\}}$$

and, the *Normalized ED Similarity* ($SimNED$) is:

$$SimNED(x, y) = 1 - \frac{ED(x,y)}{max\{|x|,|y|\}}$$

The following properties of the ED may assist us in utilizing Jaccard for estimating $SimNED$:
(a) In accordance with its definition, the Edit Distance is bounded by the length of the longest string:

$$0 \leq ED(x, y) \leq max\{|x|, |y|\},$$

it holds that:

$$0 \leq \frac{ED(x,y)}{max\{|x|,|y|\}} \leq 1, \text{ i.e., } NED(x, y) \in [0, 1].$$

(b) $SimNED(x, y) \leq 1 - \frac{||x|-|y||}{max\{|x|,|y|\}}$ as shown by the next lemma.

Lemma 1:

$$SimNED(x, y) \leq 1 - \frac{||x| - |y||}{max\{|x|, |y|\}} \tag{1}$$

Proof: By the Edit Distance definition it holds:

$$ED(x, y) \geq ||x| - |y||$$

Hence,

$$\frac{ED(x,y)}{max\{|x|,|y|\}}| \geq \frac{||x|-|y||}{max\{|x|,|y|\}}$$

And therefore:

$$1 - \frac{ED(x,y)}{max\{|x|,|y|\}} \leq 1 - \frac{||x|-|y||}{max\{|x|,|y|\}}$$

$$SimNED(x, y) = 1 - NED(x, y) = 1 - \frac{ED(x,y)}{max\{|x|,|y|\}}$$

$$SimNED(x, y) \leq 1 - \frac{||x| - |y||}{max\{|x|, |y|\}} \tag{2}$$

■

An obvious hurdle in the technique suggested, namely, using Jaccard as the basis for the approximation of Edit Distance is that these metrics are based on different mathematical objects. Jaccard is defined over (unordered) sets, in which each different element appears only once, despite that it may occur many times in different parts of the document. However, Edit Distance is defined over strings, and depends on the order of the symbols in the underlying strings.

In order to overcome the difficulties associated with this discrepancy, we confine the argument to certain types of sets and strings, both derived from the original documents; the documents in question go through a *shingling process* (which collects all the substrings of certain length of appearing in the document), which is the first necessary stage in most modern methods of similarity estimation [8]. The outcome of the shingling process are *sets of* n*-grams* (i.e., without repetitions), which will be used for computing Jaccard. Then, we create *representing strings* of the sets, by sorting and concatenating their elements according to, say, lexicographic ordering, as it described in Section 3.9.2 in [8]. As a result, we get strings of n-grams (string over the alphabet of the n-grams) that are sorted and have no repetitions. These representing strings will be used for Edit Distance estimation. We denote sets with uppercase letters e.g., X, and its representing strings with the same letter in lowercase e.g., x.

The representing strings do differ from the original documents. However and roughly speaking, the lower the frequencies of the n-grams in the original document the higher the correspondence is between the document and its representing string (See Figure 1 and Figure 2). The frequencies of the n-grams can be tuned by varying the length, the n, of n-grams (See Figure 1).

The experiments results summarized in Figure 2 indicate that it is possible to choose n-grams that yield 10% average difference between the NED over the original documents, and the NED over the representing strings. Thus, justifying our choice to concentrate in analyzing the representing string as we do in the sequel. We note that in general, one may sample a given data set and tune the length of the n-grams for the given data set, taking into account the correspondence between the original document and representing strings, prior to proceeding with the clustering of the representing strings.

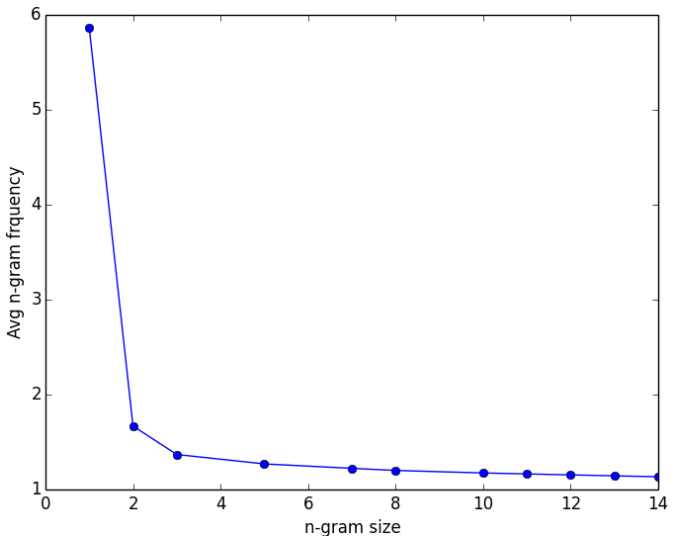

Fig. 1: n-gram length vs. the average n-gram frequency Done on a sample of 20 API call traces.

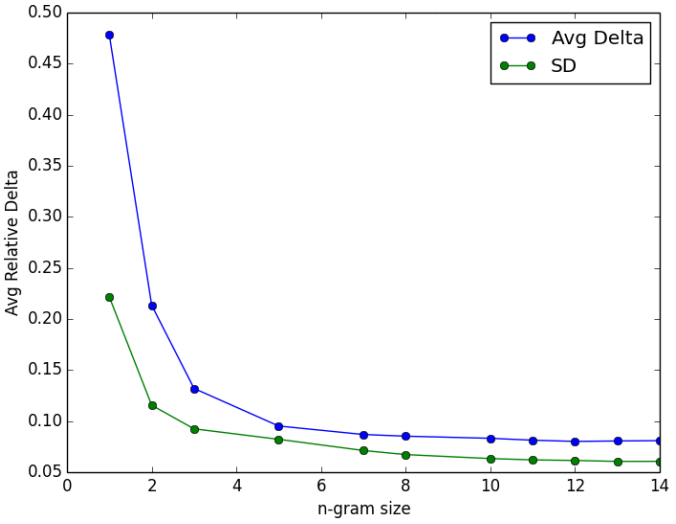

Fig. 2: The average and standard deviation of the difference (delta) between the NED on pairs of original documents and the NED on the same pair of representing strings, as a function of the n-gram length.

Next we investigate the mathematical basis of both metrics, Jaccard and Normalized Edit Distance. In fact, when a refinement process of the obtained clusters has been employed by the use of the original ED no outliers were found.

Lemma 2:

$$J_D(X,Y) = \frac{E(x,y)}{E(x,y)+C(x,y)} \tag{3}$$

where $E(x,y)$ is the *insertion/deletion Edit Distance*, and $C(x,y)$ is the length of the longest common subsequence (LCS) of x and y [8] .

Proof: The symmetric difference between two sets X, Y consists of the elements in Y but not in X and vice versa i.e., $X\Delta Y = X/Y \cup Y/X$. It holds that $E(x,y) = |X\Delta Y|$, since x and y are the representing string of the n-grams sets X and Y, respectively.

The edit operations required for transforming string x into y; are exactly deleting the elements that are in x but not in y, and inserting the element that are in y but not in x i.e., it is the symmetric difference.

Moreover, since the elements in the representing strings are sorted and have no repetitions, we have that the length of the strings' LCS is exactly the size of the corresponding sets intersection, $C(x,y) = |X \cap Y|$. Therefore,

$$\frac{E(x,y)}{E(x,y)+C(x,y)} = \frac{|X\Delta Y|}{|X\Delta Y|+|X\cap Y|} = \frac{|X\Delta Y|}{|X\cup Y|} = J_D(X,Y)$$

■

Thus, for strings which are representing string of n-gram sets there is a direct relationship between $J_D(X,Y)$ and delete/insertion based Edit Distance E. Further, we notice the fact that for any pair of strings x, y, $ED(x,y)$ does not exceed $E(x,y)$. For example [11]:

Let $x = dave; y = dav$;
Set of all 2-grams of x: $Bx = \{-d, da, av, ve, e-\}$
Set of all 2-grams of y: $By = \{-d, da, av, v-\}$
$E = 5 + 4 - 2 \cdot 3 = 3$, (3 is $|LCS| = |-d, da, av| = 3$), but Edit Distance ED=2 as we may just change the token ve with $v-$ and add $e-$.

Theorem 1: The following inequality holds for Jaccard distance and NED:

$$1-\alpha \leq NED(x,y) \leq (1+\alpha)\frac{J_D(X,Y)}{2-J_D(X,Y)} \tag{4}$$

where $\alpha = \frac{min\{|x|,|y|\}}{max\{|x|,|y|\}}$

Proof: It holds that,

$$E(x,y) = |X\Delta Y| = |x| + |y| - 2C(x,y)$$

And by Lemma 2,

$$J_D(X,Y) = \frac{2E(x,y)}{E(x,y)+|x|+|y|} \tag{5}$$

Note that, $ED(x,y) \leq E(x,y)$, as one substitution operation is equivalent to a sequence of two operations insert and then delete (or vice versa).

$$J_D(X,Y) \geq \frac{2ED(x,y)}{ED(x,y)+|x|+|y|} \tag{6}$$

$$ED(x,y) \leq J_D(X,Y) \cdot \frac{|x|+|y|}{2-J_D(X,Y)}$$

Note that the function on the right side of equality (5) is increasing in $E(X,Y)$, and once $|x|+|y| \geq C(x,y)$, equality (5) becomes inequality (6). Converting to the normalized Edit Distance we obtain:

$$NED(x,y) = \frac{ED(x,y)}{max\{|x|,|y|\}} \leq J_D(X,Y) \cdot \frac{|x|+|y|}{(2-J_D(X,Y))max\{|x|,|y|\}}$$

Taking into account that $|x| + |y| = max\{|x|,|y|\} + max\{|x|,|y|\}$, we obtain:

$$\frac{||x|-|y||}{max\{|x|,|y|\}} \leq NED(x,y) \leq (1+\alpha)\frac{J_D(X,Y)}{2-J_D(X,Y)}$$

where $\alpha = \frac{min\{|x|,|y|\}}{max\{|x|,|y|\}}$

And by Lemma 1,

$$\frac{||x|-|y||}{max\{|x|,|y|\}} \leq NED(x,y) \leq (1+\alpha)\frac{J_D(X,Y)}{2-J_D(X,Y)}$$

Notice that,

$$\frac{||x|-|y||}{max\{|x|,|y|\}} = \frac{max\{|x|,|y|\}-min\{|x|,|y|\}}{max\{|x|,|y|\}} = 1-\alpha$$

Hence,

$$1-\alpha \leq NED(x,y) \leq (1+\alpha)\frac{J_D(X,Y)}{2-J_D(X,Y)}$$

■

Corollary 1: $SimNED$ can be bounded as follows:

$$\frac{2J_S(X,Y)-\alpha(1-J_S(X,Y))}{1+J_S(X,Y)} \leq SimNED(x,y) \leq \alpha \tag{7}$$

Clearly, since $SimNED(x,y) = 1 - NED(x,y) =$

$$1-(1+\alpha)\frac{J_D(X,Y)}{2-J_D(X,Y)} = \frac{2J_S(X,Y)-\alpha(1-J_S(X,Y))}{1+J_S(X,Y)}$$

Note that $\frac{2J_S(X,Y)-\alpha(1-J_S(X,Y))}{1+J_S(X,Y)} \leq 1$.

Inequality (4) of Theorem 1 has several important implications as we discuss next. First of all, if Jaccard distance is 0 – the strings are identical and $\alpha = 1$, then the right-side of inequality (4) is zero.

The right side of both inequalities may not turn out to be greater than 1 as $J_D, J_S \leq 1$ that is, in fact, the bounds provided by the inequalities have the property of a probability measure, and may be used as an estimation for a normalized metric in any applicable domain of clustering.

Taking into account that the true values of the NED (or $SimNED$) lie between the boundaries of the inequality, we use the arithmetic mean of the upper and lower bounds of the inequality as an estimate of the NED value.

Let $\widetilde{NED}$ denote the average of the lower and upper bounds of NED, as an approximation of it.

$$\widetilde{NED}(x,y) = \frac{(1-\alpha)+(1+\alpha)\frac{J_D(x,y)}{2-J_D(x,y)}}{2} = \frac{1+\alpha(J_D(X,Y)-1)}{2-J_D(X,Y)}$$

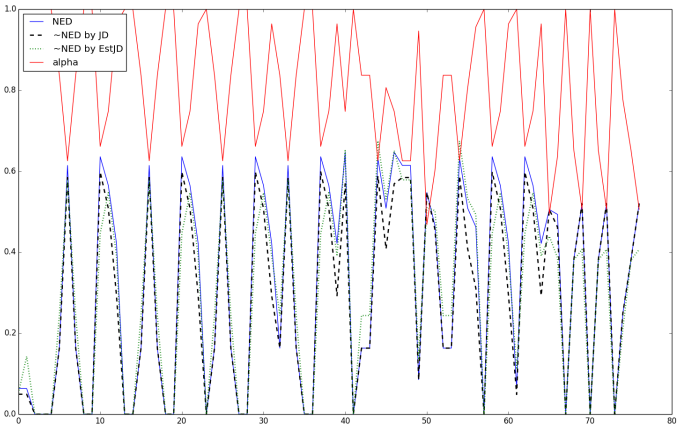

Fig. 3: NED approximations.

Figure 3 depicts the actual NED and its approximation $\widetilde{NED}$ computed as arithmetic mean of the upper and lower bounds of inequality (4). $\widetilde{NED}$ is computed once with the actual Jaccard distance J_D, and once with the MinHash estimation of Jaccard $ESTJ_D$ (as described in Section 3). The measures are done over 77 pairs of malicious traces arbitrary chosen from the KVM data set. We can see good approximation of the NED for the majority of cases, both for J_D based $\widetilde{NED}$ and $ESTJ_D$ based $\widetilde{NED}$. One may clearly notice that for the pairs whose $ESTJ_D$ based $\widetilde{NED}$ is slightly deviated from the actual NED, there is a large difference between the lengths of the objects pair (by the value of $alpha$).

The use of such NED estimation allows us to supplement the clustering technique outlined in the next section, by a scheme, where LSH provides Jaccard similar strings (traces) in the same clusters, in a way that allows us to check the ED for any item in the cluster, without accurate ED computation.

We may also exclude from the cluster those traces for which the value of NED is significantly higher than J_D, when measured from the cluster's medoid. Lastly, we may include the excluded traces in other clusters according to the ED from the representatives of the clusters (medoids). If the number of elements in a cluster is small, then the computation cost is significantly reduced with relation to the square complexity required among all the pairs of elements of the original set.

In the following section, equipped with the provable relationship between the powerful (but hard to compute) Edit Distance and the (efficiently computed) Jaccard metric, we use the efficient Jaccard's MinHash approximation to learn and identify malware based on their API call sequences.

III. Malware Identification via API Call Traces

In this section we will show that our model of Jaccard-Edit Distance relation is valid for real-life malicious traces. First of all, let us describe our technique of the traces clusterization.

A. Finding Similar Traces

Finding similar items in datasets is a major data mining problem. Next we list efficient techniques that are used for finding similar textual documents (strings , sequences or traces) in data sets.

One trivial way to find groups of similar items in a dataset is to calculate the similarity between each and every pair of traces, and then find all pairs of items that are above some similarity threshold $sim(x_i, x_j) \geq t$ and group them into one cluster. This method is time consuming, since in dataset of size n there are $\binom{N}{2} = O(n^2)$ pairs of items.

Similar traces can be grouped together using LSH in linear time with only a small increase in false negative results. Locality sensitive hashing, see e.g., [8], [13], hashes items into buckets several times, such that:

- Similar items are hashed into the same bucket with high probability, while
- Items that are not similar enough are hashed into a common bucket with low probability.

Hence, there is a benefit of using large number of buckets for maximizing the probability of collision of similar items. Items that are mapped to the same bucket are considered *candidates* for being similar. In case more accuracy is desired, the candidate items should be further investigated by explicitly computing the similarity function among pairs of items in a bucket. Note, that a bucket may consist of significantly less items compared to the total number of items. Typically, the similarity is computed only for a limited amount of item pairs within the same bucket.

Another technique we use is MinHashing [8], [9] which is a compression method for sets of items that preserves the Jaccard similarity. With this method it is possible to replace long text files by much shorter same-length *MinHash signatures*;

First, the text files are *shingled* into n-grams and stored as

```
1376 malware.exe
#308810000
RegQueryValue
hKey=HKEY_LOCAL_MACHINE\System\CurrentControlSet\
 Services\LDAP\Ldap...
Return=SUCCESS
type=REG_DWORD
data=1
#308120000
LoadLibrary
lpFileName=adsldpc.dll
Return=SUCCESS
#313460000
LoadLibrary
lpFileName=adsldpc
Return=SUCCESS
....
```

Fig. 4: The beginning of a malware trace, which shows 3 timestamped Windows API Calls (RegQueryValue, LoadLibrary, LoadLibrary).

binary bag of words vectors, which have one coordinate for each possible string of length n over the alphabet Σ i.e, the length of these vectors is $|\Sigma|^n$. A coordinate gets a value 1 iff its corresponding string appears in the text file at least once.

Second, since binary bag of words vectors are impractical due to their size and sparsity, these vectors are compressed into much shorter signatures using MinHashing. The outcome of MinHashing are integer vectors that have the following property: the Jaccard similarity of the binary bag of words vectors is the same as the expected similarity of their MinHash signatures. MinHash signatures' length is determined by the number of MinHash functions used in the process, which is user-defined.

The above techniques are combined together for efficiently solving the problem of finding similar textual items. For improved efficiency the text items are MinHashed into signatures, then LSH is performed on these signature (integer vectors) using the *banding technique* [8]. According to this technique a signature is cut into b subsequent portions, which are hashed separately, giving each pair of items b opportunities to be declared similar. The outcome of this process is a hash-table, with similar items in its buckets.

B. Offline Clustering - Online Classification

We regard the provided API Calls traces [1] (see example trace in Figure 4) as plain text, ignoring any semantics of the API calls.

Clustering malware traces. Having a decent distance or similarity measure between pairs of API calls traces, makes it possible to obtain high quality clustering of API calls traces data sets.

In our approach, and under the assumption of low similarity between malware and benign traces, the malicious/benign classifier training is done by clustering malicious traces only (the number of clusters is not predefined). Then during the identification (prediction) phase, the task is to identify whether a new unseen trace is malicious or benign. The classification is done by taking a new trace and trying to fit it into one of the malware clusters. The new trace is classified as malicious when it matches (with high similarity rate) one of the clusters. Otherwise, if the new trace is found as an outlier or as anomalous to the rest of the traces inside the clusters, it is classified as benign.

C. Jaccard-Edit distance relationship model validation.

We now turn to evaluate the suggestion to use the Jaccard distance estimation for Edit Distance according to the results in Section 2. We use consequences of inequality (1) as a validity verification property. Figure 5 and Figure 6 show the relations between J_D and NED for the traces of API calls, hashed into two different buckets, considered as some strings transformed in the shingles, like the strings x, y in Section 2. Where the "average" means the average values of corresponding distances over the traces inside the bucket.

By inequality (1) we have that $NED \leq J_D$ if $J_D \leq 1-\alpha$ (we just resolve inequality), and in our experiments we deal with $\alpha \geq 0.5$. Hence, $NED \leq J_D$ holds for all $J_D \leq 0.5$. Our experiments show that this actually holds, demonstrating that the assumptions made in Theorem 1 are valid and relevant.

Indeed, from the inequality we can see, that $NED \leq J_D$ if $J_D \leq 1-\alpha$ (we just resolve inequality). Since, as it follows from Figure 3, the α in our experiments ≥ 0.5, this inequality must hold for all $J_D \leq 0.5$. Figure 5 and Figure 6 illustrate that this actually holds, demonstrating that the assumptions made in Theorem 1 are valid and relevant.

Note, that in case $J_D \geq 1-\alpha$, the relation between NED and J_D can be both $\leq, \geq$, as in this case there is an obvious contradiction between the inequalities.
$NED \leq (1+\alpha)\frac{J_D(x,y)}{2-J_D(x,y)}$ and $(1+\alpha)\frac{J_D(x,y)}{2-J_D(x,y)} \geq J_D$.

However, if, for example, we received a bucket with small J_D, namely with high Jaccard similarity rate, but NED significantly exceeds J_D, then it can indicate that this bucket should be excluded from the clustering as NED is a stronger characteristic of similarity and large NED indicates the irrelevance of LSH result in this case.

IV. Experiments, Results and Discussion

Do malware and benign traces have low similarity? We now examine the assupmtion of low similarity between malware and benign traces. In addition, we test that the differences between malware and benign traces are tangible even when the API calls are treated as plain text, without taking into account any semantics of API name, arguments or return value.

We clustered 1652 malicious traces using MinHash of 20 hash functions (signature length is 20), and LSH with 4 bands and 517 buckets hash table. The traces were mapped into 501 buckets. The medoid [7] of each bucket was chosen as its representative. Two test datasets, 48 traces each, were used. The first consists of all malicious traces and the second consists of all benign traces. We have compared each test trace to all the medoids, and kept its Jaccard similarity to the most similar medoid. The obtained distribution of value in each test dataset as follows:

Dataset	Max	Median	Average	Standard Deviation
Malicious	1	0.984819	**0.923450**	0.112049
Benign	0.897219	0.506525	**0.513685**	0.148236

The table above shows the significant distinction between the high similarity rates of the malicious traces to the medoids (all the medoids are malicious) on one hand, and the low similarity rates of the benign traces to the medoids on the

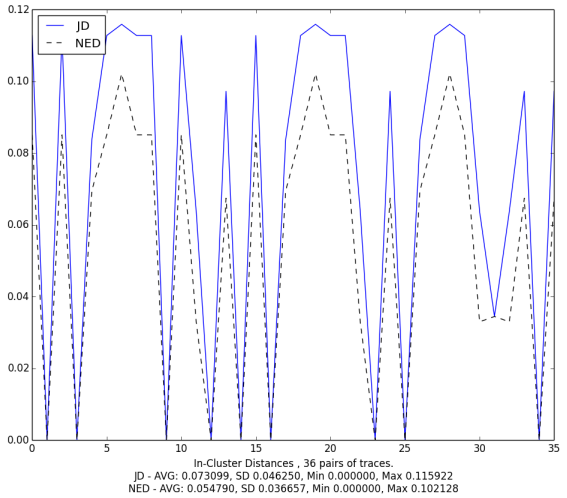

Fig. 5: Example 1, In-Cluster distances of 36 pairs.

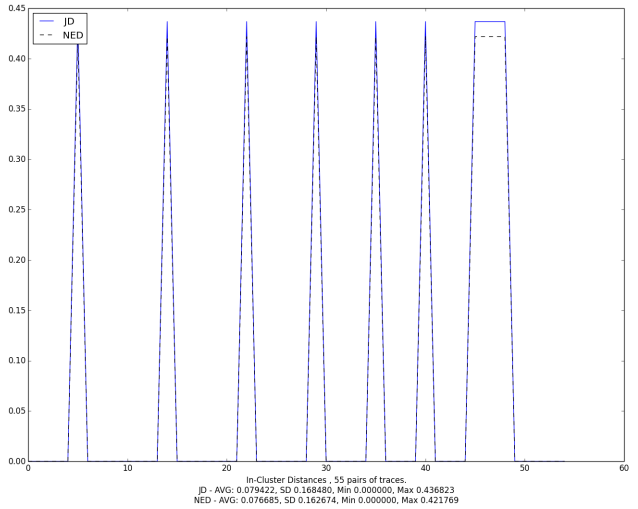

Fig. 6: Example 2, In-Cluster distances of 55 pairs.

other hand. This supports our assumption of low similarity between malware and benign traces, even without any semantic considerations. Hence, after clustering only malicious traces, benign traces are expected to be anomalous to the clusters' representatives.

On using only the malicious clusters for classification. We used the same training dataset and test dataset for testing our clustering-backed classifier, which is based on the aforementioned assumptions. A new query trace is compared to all the medoids; if its maximal similarity to one of the medoids exceeds a threshold $t = 0.7$ then it is classified as malicious, otherwise it is classified as benign. We got 9% classification error on the test dataset, 6% error on the malicious traces and 12% on the benign traces. Taking into account that ED is more powerful for the traces similarity analysis (but hard to compute) we can use the efficient Jaccard's MinHash approximation for approximation of the ED as $\widetilde{NED}$ which can be used to learn and identify malware based on their API call sequences. In particular, the fact that $\widetilde{NED}$ exceed the estimated J_D in some cluster can be used for the clustering reasonability checking (see subsection C of Section 3).

REFERENCES

[1] Shun-Wen Hsiao, Yi-Ning Chen, Yeali S. Sun, Meng Chang Chen, "A Cooperative Botnet Profiling and Detection in Virtualized Environment", *IEEE Conference on Communications and Network Security (CNS 2013)*, 2013.

[2] Ki, Youngjoon, Eunjin Kim, and Huy Kang Kim."A novel approach to detect malware based on API call sequence analysis." International Journal of Distributed Sensor Networks 11.6 (2015): 659101.

[3] Peisen Yuana, Haoyun Wanga, Jianghua Chea, Shougang Rena, Huanliang Xua, Dechang Pib, "Approximate String Similarity Join using Hashing Techniques under Edit Distance Constraints", *Journal of Software*, Vol. 9, No. 10, 2014.

[4] Roberto Perdisci, Wenke Lee, and Nick Feamster, "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces," *Proceeding of the 7th USENIX conference on Networked systems design and implementation* (NSDI10), 2010.

[5] Islam, Rafiqul, et al. "Classification of malware based on integrated static and dynamic features." Journal of Network and Computer Applications 36.2 (2013): 646-656. APA

[6] Gupta, Sanchit, Harshit Sharma, and Sarvjeet Kaur."Malware Characterization Using Windows API Call Sequences." International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer International Publishing, 2016.

[7] Wikipedia, Medoid, `https://en.wikipedia.org/wiki/Medoid`.

[8] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman, *Mining of massive datasets*, Cambridge university press, 2014

[9] Andrei Z. Border, "On the resemblance and containment of documents", *Proc. of the IEEE Compression and Complexity of Sequences* 1997.

[10] Santos, Igor, et al. "Opem: A static-dynamic approach for machine-learning-based malware detection." International Joint Conference CISIS12-ICEUTE 12-SOCO 12 Special Sessions. Springer, Berlin, Heidelberg, 2013.

[11] Doan, AnHai, Alon Halevy, and Zachary Ives. Principles of data integration. Elsevier, 2012.

[12] Shlomi Dolev, Mohammad Ghanayim, Alexander Binun, Sergey Frenkel, Yeali S. Sun. "Relationship of Jaccard and Edit Distancein Malware Clustering and Online Identification", Department of Computer Science, Ben-Gurion University of the Negev, Technical Report, #17-05, `https://www.cs.bgu.ac.il/frankel/TechnicalReports/2017/17-05.pdf`, October 2017.

[13] Aristides Gionis, Piotr Indyk, and Rajeev Motwani, "Similarity search in high dimensions via hashing" VLDB. Vol. 99. No. 6. 1999.

[14] Xavier Dupré, "Some thoughts about normalizing an edit distance", `www.xavierdupre.fr`, 2013.

[15] Arturs Backurs, Piotr Indyk, "The Edit Distance Cannot Be Computed in Strongly Subquadratic Time", Arxive: 1412.0348v3, April 2017.

[16] Seung-Shik Kang, "Word Similarity Calculation by Using the Edit Distance Metrics with Consonant Normalization", *Journal of Information Processing Systems*, Vol.11, No.4, pp.573-582, 2015.