Clustering IoT Malware based on Binary Similarity

Márton Bak[1], Dorottya Papp[1,3], Csongor Tamás[1,2], and Levente Buttyán[1]

[1]CrySyS Lab, BME-HIT
[2]Ukatemi Technologies
[3]AIT Austrian Institute of Technology GmbH.

***Abstract*—In this paper, we propose to cluster malware samples based on their TLSH similarity. We apply this approach to clustering IoT malware samples as IoT botnets built from malware infected IoT devices are becoming an important trend. We study the performance of two distance-based clustering algorithms, k-medoid and OPTICS, on a large corpus of IoT malware samples when they are used with the TLSH difference metric to measure distances between samples. Our results show that neither of the two algorithms have acceptable clustering performance. Hence, we propose a new clustering algorithm, which achieves a performance superior to both k-medoid and OPTICS.**

I. Introduction

The concept of malicious software – or malware, as it is called in the computer security community – is almost as old as computers themselves. While in early years, malware was mainly created for fun or for experimental purposes, with the growing number of personal computers and the proliferation of Internet connectivity, malware development became a profitable business for miscreants at the end of the last century. Later, smart phones appeared, and attackers started developing malware for mobile devices. Today, we are witnessing a new trend: all sorts of embedded devices are being connected to the Internet, which is rapidly transforming into an *Internet of Things*, or IoT for short. Not surprisingly, malware development followed this new trend, and malware is now developed for embedded IoT devices as well.

A significant problem is that the number of IoT devices is already large and grows exponentially, which means that they can be converted into a substantial attack infrastructure by infecting them with malware and organizing them into botnets. Actually, such botnets have already appeared in the wild. An infamous example is the *Mirai* botnet, and the importance of the problem is illustrated by the fact that it holds the record for the most intensive DDoS attack in history ever [1]. Of course, malware infected IoT devices can be used not only for building botnets, but also for all sorts of other misdeeds, such as click fraud and bitcoin mining.

Anti-virus companies rely on malware classification methods to identify relating malware samples. Clustering malware into families makes sense, as members of the same family, while being different at the binary level, exhibit similar behavior. So ultimately such clustering reduces the load on virus analysts by allowing them to focus on samples that are not similar to any known sample.

978-1-7281-4973-8/20/$31.00

In this paper, we propose to cluster malware samples based on their TLSH difference, where TLSH is a fuzzy hash algorithm developed by Trend Micro [2]. While this approach can be used for clustering any malware, here we use it to cluster IoT malware samples due to the importance of this new trend. We study the performance of two distance-based clustering algorithms, k-medoid and OPTICS, on a large corpus of IoT malware samples when they are used with the TLSH difference metric to measure distances between samples. Our results show that neither of the two algorithms have acceptable performance: k-medoid produces clusters with unacceptably large diameter, meaning that it puts unrelated samples into the same cluster, whereas OPTICS fails to cluster more than half of the samples in our data set. To overcome these problems, we propose a new clustering algorithm, which is based on OPTICS and achieves a performance superior to both k-medoid and OPTICS.

The organization of the paper is as follows: In Section II, we provide some background on program similarity measures, and clustering. In Section III, we give an overview of our research methodology, including how we obtained our initial set of IoT malware samples, how we cleaned this initial corpus, and how we determined the TLSH difference threshold under which two samples are considered variants of the same malware. In Section IV, we present the performance of the k-medoid and OPTICS clustering algorithms on our corpus and explain why they are not appropriate for malware clustering. We describe our own clustering algorithm in Section V, and evaluate its performance and compare it to that of k-medoid and OPTICS in Section VI. Finally, we conclude our report and sketch some possible future work in Section VII.

II. Background

In response to polymorphism and metamorphism employed by modern malware, anti-virus companies have begun to utilize multi-layered approaches [3], [4], [5] in order to detect malware. The layers include techniques for metadata-analysis, static analysis, dynamic analysis and machine learning, as well as network-based techniques. Our work is concerned with static analysis and clustering, therefore, we focus on these.

During static analysis, the instructions and bytes of the sample are analyzed. Because the sample is not executed, such approaches typically scale better and can provide a quick insight into the sample's functionality. Fuzzy hash algorithms, such as the Trend Micro Locality Sensitive Hash (TLSH) [6],

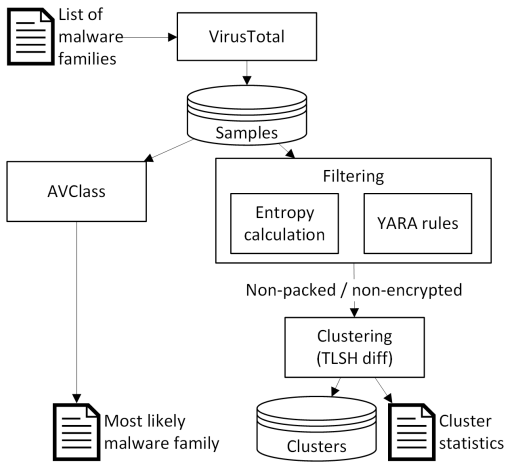

Fig. 1. Overview of Methodology

are a popular group of static analysis techniques that construct a fingerprint or digest of the input sample. Two output hashes can then be compared to measure the similarity of the original samples. Recent research [7], [8] have shown that TLSH is not only more precise than previous methods, including the state-of-the-art approaches `ssdeep` [9] and `sdhash` [10], it is also applicable for malware classification. In order to measure similarities between files, TLSH digests have to be computed and compared. TLSH performs well as long as the input binary is not packed or encrypted.

In this paper, we study the applicability of two clustering algorithms, k-medoid and OPTICS. K-medoid [11] is a PAM-based algorithm in which clusters can have only valid data points as their centers (also called medoids). The algorithm has one input parameter, k, which determines how many clusters will be present in the output of the algorithm. The algorithm first selects k medoids, then tries to fit all data points to the nearest cluster head. Medoid selection and re-clustering is repeated iteratively until an optimum is reached. The measure of goodness for the algorithm is $s(k)$ [11], which measures the gain in assigning data points to specific clusters based on distance. The closer the metric is to 1, the better the setup.

OPTICS [12] is a density-based algorithm. It is capable of identifying sparse and dense regions in the data set in order to create clusters. It takes two parameters, ϵ_{max} and $minPts$. ϵ describes the radius of an area, while $minPts$ is the minimum number of data points in that area. The algorithm dynamically calculates ϵ values for data points such that data points have at least $minPts - 1$ samples in their ϵ radius. OPTICS also has a built-in clustering algorithm, ξ, which clusters data points by detecting abrupt changes in the ϵ-values.

III. Methodology

The high-level overview of the methodology we followed during this research is shown in Figure 1. The methodology can be divided into three main steps:

1) data collection, which results in a data set of IoT malware samples,
2) filtering, which removes packed and/or encrypted samples from the data set, and
3) clustering, which identifies variants in the data set by grouping malware samples based on their pair-wise TLSH differences.

To evaluate cluster configurations, we need the TLSH difference threshold denoting variants of malware families.

A. Data Collection

In order to acquire a data set of IoT malware samples, we first need to select an IoT-relevant embedded architecture which the data set should target. This is a required step as the different instruction sets could cause TLSH to measure big differences between variants compiled for different architectures. For this study, we select samples targeting the ARM architecture due to its widespread use in the IoT world. Secondly, we compile a list of 29 malware family names based on previous studies of the IoT malware landscape [13], [14], [15]. These malware families specifically target the IoT ecosystem. Many of them implement the ability to infect other machines and connect them to an existing botnet. The botnet is remotely administered by the attacker via various channels, e.g. IRC or HTTP-based communication. Samples from these families take various commands from the attacker via a command & control server. The families also share similar traits as they are known to copy and develop features from each other, e.g. after Mirai's source code was leaked [16], several modifications led to the branches Satori, Okiru, Masuta and PureMasuta.

We use the compiled list to search for and download malware samples from VirusTotal [17], a publicly available site to which users can upload executables and submit URLs. The site scans uploaded executables with a number of anti-virus tools and returns to the user the collected results, including the malware family names assigned by anti-virus tools. We downloaded 12 993 samples and their corresponding anti-virus scan results.

The scan results from VirusTotal are fed to AVClass [18], which outputs the most likely malware family name based on a majority vote cast of anti-virus tools' assigned labels. We made changes to the tool's source code as initially, it could not provide a label for a number of samples. In order for AVClass to cast a majority vote, it needs at least 4 detections per sample. As some of our samples have lower detection rates, we remove this requirement. Throughout our study, we use AVClass's output as the ground truth for all samples.

B. Filtering

The second step in our methodology is to filter the downloaded samples. The step is required because TLSH has difficulty identifying similarities between packed and/or encrypted samples. We use two approaches for filtering our data set. Firstly, we use binary entropy calculation, which calculates the empirical entropy of a file based on the contained bytes. There exist best practices for calculated entropy values signaling packed and/or encrypted executables [19]. The measured

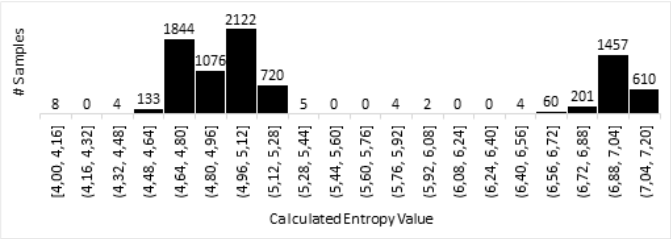

Fig. 2. Entropy distribution of our dataset

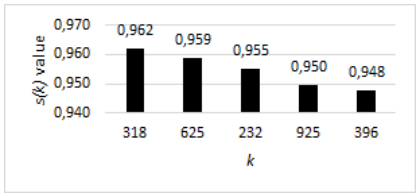

Fig. 3. Best $s(k)$ values of our dataset

empirical entropy values of our data set is shown in Figure 2. There is a clear cut between the set of native executables and the set of packed and/or encrypted samples. As a result, we excluded 2817 samples.

TABLE I
DISTRIBUTION OF MALWARE FAMILIES IN THE FILTERED CORPUS

Family	# Samples	Family	# Samples
mirai	6108	dnsamp	5
gafgyt	3711	oneeva	2
dofloo	163	ditertag	2
tsunami	92	zergrush	1
ddostf	63	luabot	1
presenoker	19	lightaidra	1
cloxer	1	no name recovered	7

Binary entropy calculation has one major limitation, namely, that large sections of low-entropy bytes can lower the calculated overall entropy. In order to remove this limitation, we also use YARA-rules [20], as packers can leave traces in the binary, e.g. specific strings and/or byte sequences unique to the packer. We run YARA-rules for UPX and other packers on our whole data set, looking for packed binaries. We found a total of 980 packed samples, all of which were packed with UPX and have already been filtered using binary entropy calculation. Table I shows the distribution of malware families in our filtered data set.

C. TLSH difference threshold selection

Before being able to cluster our data set, we needed a TLSH difference threshold signaling variants of malware families that produce zero false matches. In [8], a threshold of 70 was suggested to be used for few false positives. This suggestion, however, was based on a relatively small set of 477 samples with most of the samples belonging to two families. In order to define a globally applicable TLSH difference threshold for malware similarity, we carry out a measurement at a much bigger scale.

We use the EMBER data set [21], the largest available labeled malware data set at the time of writing. We process its test set, containing 100 000 malicious samples from 917 malware families. As the actual malware binaries required to calculate the TLSH digests are not included in the data set, our measurements are carried out on the 62 863 samples available in Ukatemi Technologies's malware repository. In order to find the maximal zero false positive threshold, we search the available sample set with a candidate threshold. If false matches are found, the threshold is reduced. This process yielded the zero false positive threshold of 1, which is a much lower threshold than we anticipated. Manual investigation based on VirusTotal details and behavior pages, as well as IDA PRO [22] with the Diaphora plugin [23] reveals, that the EMBER data set labels are often incorrect: samples with very similar functionality and behavior are labeled as members of different malware families.

As a result, we resort to manual verification. We randomly select samples from a a data set of 355 795 714 unlabeled malware samples, courtesy of Ukatemi Technologies. The selected samples include 9 Azorult samples, 3 Lightneuron samples and 10 Pioneer samples. We compare these samples to the rest of the data set using the above mentioned methodology. Our analysis yielded the TLSH difference threshold of 48.

D. Clustering

The final step is clustering the data set. Our goal is to group samples based on their TLSH difference, thereby detecting variants of malware families. Initially, we clustered the data set with two widespread algorithms, k-medoid and OPTICS, using TLSH difference as the distance metric. The results, presented in more details in Section IV, show that k-medoid often puts unrelated data into the same clusters, while OPTICS fails to cluster more than half of the samples in our data set. Therefore, we develop a new clustering algorithm, which we present in Section V.

IV. CLUSTERING RESULTS

A. K-Medoid

There are several rationales behind choosing k-medoid as the clustering algorithm. It is unsupervised, ie. there is no need to supply any additional data, only the similarity measurements between samples. In addition, the algorithm only selects existing data points as cluster heads. This is useful in our scenario, because cluster heads can represent other malware samples in the same cluster. The disadvantage of this algorithms is that the input k has to be specified.

Unfortunately, we do not know how many variants there are in our data set, therefore, we calculate cluster configurations for all potential k values. We compute the $s(k)$ metric for all cluster configurations in order to rank our setups. As shown in Figure 3, the best $s(k)$ values of the calculated cluster configurations barely differ, however, the corresponding k values have a wide range, making it unclear which setup

TABLE II
STATISTICS FOR K-MEDOID CLUSTER CONFIGURATION WITH k = 17

s(k)	0.4054158410234883
Maximum diameter	1038
Minimum diameter	180
Mean diameter	467.823529411
Largest cluster size	1110
Smallest cluster size	35
Mean cluster size	573.294118

to choose. We also observed that several cluster heads have small TLSH differences when compared to other cluster heads, which suggests that clusters could be merged. As variants have a TLSH difference lower then 48, cluster heads of different clusters should have a TLSH difference score higher than 48. With this requirement in mind, we looked at our cluster configurations and found, that with TLSH thresholds ranging from 30 to 70, $k = 17$ achieves the best $s(k)$ value.

Statistics of the cluster configuration $k = 17$ is shown in Table II. In this configuration, the calculated cluster diameters range from 180 to 1038, the mean being 468. In our scenario, cluster diameter is interpreted as the largest TLSH difference between any two samples in the same cluster. Taking our TLSH difference threshold for variants of 48 into consideration, we can conclude that clusters in this setup contain very different samples; clusters should be split into smaller clusters.

Cluster sizes in the setup range from 35 to 1110 samples. However, as shown in Table I, certain families contribute only a small number of samples to our data set. The configuration does not reflect that distribution as it does not have any singleton or small clusters. Thus, we conclude, that the k-medoid algorithm does not perform well for the goal of clustering malware samples based on TLSH differences.

B. OPTICS

The second algorithm we tested was the density-based algorithm, OPTICS [12]. The algorithm is able to detect and cluster dense and sparse regions in the data set. This characteristic makes it favorable in our scenario as we have families with only a few samples as well as families with thousands of samples, as shown in Table I.

The algorithm takes two additional parameters besides a precomputed distance matrix: $minPts$ and ϵ_{max}. We can specify an upper bound for ϵ_{max} as the maximum TLSH difference in our data set. However, selecting $minPts$ is a challenge without knowledge about the internal structure of our data set. To gain this knowledge, we ran OPTICS with different parameter setups: ϵ_{max} values were set to be 40, 50, 60 and 70, while $minPts$ was set to be 1, 2, 5, 10, 20, 40, 50, 70, 100, 150 and 200.

The resulting cluster configurations are again unsatisfactory. In all configurations, the number of unclassified samples is very high. Different values of ϵ_{max} does not seem affect this trait: setting $minPts$ to 2, $\epsilon_{max} = 40$ yielded 1800 unclassified samples, while $\epsilon_{max} = 70$ resulted in 1721 unclassified samples. The more we increased $minPts$, the more unclassified samples were returned. The configuration $\epsilon_{max} = 70$, $minPts = 200$ resulted in 6934 unclassified samples, which is 68% of our data set. Such a high number of unclassified samples is disadvantageous in our scenario, as many samples would require additional analysis.

C. Discussion

The results of both tested algorithms present issues for malware analysis. Firstly, the k-medoid algorithm produces clusters whose diameters are too large to represent variants of malware families. We determined that the threshold TLSH difference for variants is 48, however, k-medoid's diameters range between 180 and 1038. OPTICS's cluster diameters are more in line with our threshold value, however, as much as 68% of our samples are detected as outliers.

These algorithms were originally developed to cluster measurements which may be noisy. In order to remove noise, the data set must be cleaned and it must also be balanced. A balanced data set in our scenario requires exclusion of samples from families with very high and very low sample counts. Such a step, however, is undesirable as potential outliers may represent previously unseen variants or entirely new families.

V. NEW CLUSTERING ALGORITHM

In light of our experiment presented previously, we develop a new clustering algorithm which meets the following requirements. Firstly, it has to cluster samples based on their binary similarity expressed as TLSH differences. Secondly, it has to be able to find even the smallest clusters in a varying density data set. The input data set may contain singleton clusters, i.e. samples dissimilar to every other sample, however, these must not be treated as noise because these are the most interesting samples for malware analysis.

Our algorithm is based on OPTICS, however, we replace its default clustering algorithm, ξ. Our algorithm can be divided into three major phases. In the first phase, we extract information about the structure of the data set. In the second phase, we generate a greedy, initial cluster configuration based on TLSH differences. In the last phase, we merge clusters in order to compensate for the greedy mechanism in the previous phase.

In order to extract structural information from the data set, we reuse OPTICS with input parameters $minPts$ and ϵ_{max}. OPTICS can compute the ϵ values required to form a cluster around individual samples. Samples with low ϵ values represent dense regions while samples with high ϵ values represent sparse regions. We sort the resulting ϵ values such that samples in dense regions come first in the list.

Our initial clustering begins with no clusters and chooses the first sample with the lowest ϵ value. This sample represents the densest region of the data set and it becomes the first cluster head. We then put all samples into the selected head's cluster which are considered similar enough, captured by a threshold parameter. Subsequent cluster heads are selected such that they have the lowest corresponding ϵ value and they are significantly dissimilar to previously selected cluster

TABLE III
COMPARISON OF THE CLUSTERING METHODS

	k-medoid	OPTICS	Our algorithm
Number of clusters	17	13	745
Number of singletons	0	6058	353

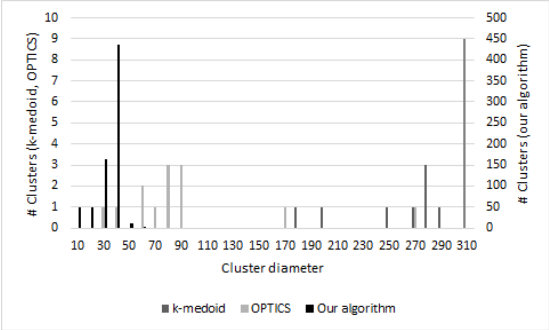

Fig. 4. Diameters Produced by Clustering Algorithms

heads. There may be cases when new cluster heads cannot be selected this way. In such cases, we select the sample with the maximum dissimilarity to every other cluster head. We repeat this process until all samples are clustered.

The data set may contain dense regions whose diameter is larger than the maximum allowed dissimilarity. In such cases, the initial clustering strategy faces a limitation as it groups the center of the region into one large cluster and generates several small clusters on its perimeter. In order to overcome this challenge, we try to detect such perimeter clusters and merge them with the center cluster. We combine two clusters if the combined cluster's diameter either remains under 48 (our empirical threshold for variants), or merging increases the diameter of the center cluster by a fixed parameter.

VI. EVALUATION

In order to evaluate the efficiency of our proposed clustering algorithm, we compared it against the results of both k-medoid and OPTICS. During evaluation, we took into consideration cluster diameters, the number of singleton clusters generated and two new measures of goodness.

The number of generated clusters and singleton clusters are shown in Table III. K-medoid and OPTICS both generate considerably fewer clusters than our algorithm does, more in line with the number of malware families our data set contains. Our algorithm generates 745 clusters of which 353 are singletons, a negligible amount compared to OPTICS's performance.

The diameters of cluster configurations from all three algorithms are shown in Figure 4. Because our algorithm produced much more clusters than k-medoid and OPTICS, we use a different scale for the number of clusters in its case. Our experiments have shown that in order to detect variants of malware families, cluster diameters must be below 48. The figure shows that both k-medoid's and OPTICS's cluster diameters are too large to denote variants. The cluster

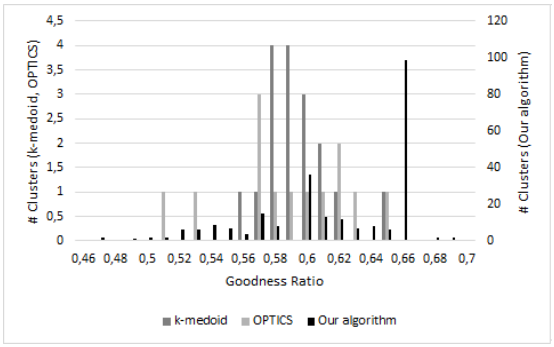

Fig. 5. Goodness Ratios Produced by Clustering Algorithms

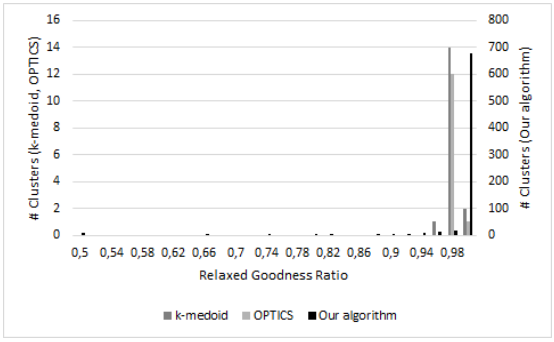

Fig. 6. Relaxed Goodness Ratios Produced by Clustering Algorithms

configuration of our algorithm, however, is much closer to this threshold with 93.69% of our clusters having diameters below 50. As a result, our clusters are more likely to represent malware variants.

The first measure of goodness we present shows how "pure" a cluster is, i.e. how many samples of the cluster are of the family with the most samples in that cluster. This metric can only be computer for non-singleton clusters as clusters containing only a single sample automatically achieve the measure of 1. Figure 5 shows the algorithms' performance with respect to this measure of goodness. Cluster configurations of both k-medoid and OPTICS typically achieve ratios between 0.56 and 0.63. By contrast, of the 392 non-singleton cluster produced by our algorithm, 185 have ratios over 0.6 of which 103 outperform both OPTICS and k-medoid.

We need to take into consideration that malware families share and/or copy features from each other. In response, the relaxation of our measure of goodness considers not only the family with the most samples in a given cluster but also families with which it is known to share features. Singleton clusters can be included, since non-singleton clusters have a chance of achieving the ratio 1 due to the relationships between malware families. The relaxed measure of goodness ratios achieved by the compared algorithms are shown in Figure 6. The figure shows that even though both OPTICS and k-medoid achieve ratios above 0.95, our algorithm outperforms both with almost all clusters achieving the measure of 1. However, our algorithm produces clusters whose relaxed ratios

are well below those achieved by k-medoid and OPTICS. While investigating this issue, we found an indication of poor anti-virus labels. Specifically, all clusters achieving ratios of 0.5 contain 2 samples and their malware family labels do not match. However, the diameters of these clusters is quite low, the mean diameter being 30.71. Checking the samples' VirusTotal pages, we also saw that there were only a few labels on their scan pages. As a result, the low ratios could be the result of misclassifications.

VII. Conclusion

In this paper, we proposed to cluster malware samples based on their TLSH fuzzy hash values. We applied this approach to cluster IoT malware samples as IoT botnets built from malware infected IoT devices are becoming an important trend, however, we note that our approach is generic and can be applied to other malware as well. We studied the performance of two distance-based clustering algorithms, k-medoid and OPTICS, on a large corpus of IoT malware samples when they are used with the TLSH difference metric to measure distances between samples. We found that neither of the two algorithms had acceptable performance: k-medoid produced clusters with unacceptably large diameter, meaning that it put unrelated samples into the same cluster, whereas OPTICS failed to cluster more than half of the samples in our corpus.

To overcome these problems, we proposed a new clustering algorithm, which was based on OPTICS, but achieved a better clustering performance. Our algorithm identifies dense regions in the data set and considers the data points in the center of the dense regions as cluster heads. A data point is included in a given cluster if and only if its TLSH difference to the cluster head is below a threshold value. We also merge clusters if they are close to each other in terms of TLSH difference and the diameter of the resulting merged cluster does not exceed a pre-specified threshold. We determined the TLSH difference threshold below which two samples are considered to be variants of the same malware by empirical analysis of an independent set of malware samples (the EMBER data set).

Our future work consist of analyzing the relationships between clusters by looking at their content more closely. Our suspicion is that this analysis will help us identify evolving features of variants in malware families. If this proves to be true, then our method can be used to identify individual variants within malware families, allowing for a finer grained classification of samples than that based on family labels.

Acknowledgement

The presented research has been partially supported by the SETIT Project (no. 2018-1.2.1-NKP-2018-00004), which has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2018-1.2.1-NKP funding scheme, and by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[2] J. Oliver, C. Cheng, and Y. Chen, "Tlsh–a locality sensitive hash," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*. IEEE, 2013, pp. 7–13.

[3] K. Lab, "Advanced cybersecurity technologies: How it works," https://www.kaspersky.com/enterprise-security/wiki-section/home, last visited: Feb 28, 2020.

[4] s. s. r. ESET, "ESET leading-edge technology," https://www.eset.com/int/about/technology/, last visited: Feb 28, 2020.

[5] Broadcom, "Symantec content and malware analysis," https://www.symantec.com/products/atp-content-malware-analysis, last visited: Feb 28, 2020.

[6] J. Oliver, C. Cheng, and Y. Chen, "Tlsh – a locality sensitive hash," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, Nov 2013, pp. 7–13.

[7] F. Pagani, M. Dell'Amico, and D. Balzarotti, "Beyond precision and recall: understanding uses (and misuses) of similarity hashes in binary analysis," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. ACM, 2018, pp. 354–365.

[8] C. Tamás and B. Bencsáth, "Method for similarity searching in large malware repository," 2018.

[9] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digit. Investig.*, vol. 3, pp. 91–97, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2006.06.015

[10] V. Roussev, "Data fingerprinting with similarity digests," *IFIP Advances in Information and Communication Technology*, vol. 337 AICT, pp. 207–226, 2010, cited By 88. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-78651093858&doi=10.1007%2f978-3-642-15506-2_15&partnerID=40&md5=d72d586c1e2186fdc9519c8ca35661f9

[11] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[12] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.

[13] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of ddos-capable iot malwares," in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2017, pp. 807–816.

[14] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 161–175.

[15] B. Vignau, R. Khoury, and S. Hallé, "10 years of iot malware: a feature-based taxonomy," 2019.

[16] "Source code for iot botnet 'mirai' released," https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/, Oct 2016, last visited: Feb 28, 2020.

[17] "Source code for iot botnet 'mirai' released," https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/, last visited: Feb 28, 2020.

[18] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 230–253.

[19] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Security & Privacy*, vol. 5, no. 2, pp. 40–45, 2007.

[20] "YARA," https://yara.readthedocs.io/en/latest/, last visited: Feb 28, 2020.

[21] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," *ArXiv e-prints*, Apr. 2018.

[22] Hex-Rays, "Ida interactive disassempler," https://www.hex-rays.com/products/ida/.

[23] joxeankoret, "diaphora," https://github.com/joxeankoret/diaphora.