Detection and Classification of Malicious Software based on Regional Matching of Temporal Graphs

Helen–Maria Dounavi
Department of Computer Science & Engineering,
University of Ioannina
Ioannina, Greece
edounavi@cse.uoi.gr

Anna Mpanti
Department of Computer Science & Engineering,
University of Ioannina
Ioannina, Greece
ampanti@cs.uoi.gr

Stavros D. Nikolopoulos
Department of Computer Science & Engineering,
University of Ioannina
Ioannina, Greece
stavros@cs.uoi.gr

Iosif Polenakis
Department of Computer Science & Engineering,
University of Ioannina
Ioannina, Greece
ipolenak@cs.uoi.gr

ABSTRACT

In this paper we present an integrated graph-based framework that utilizes relations between groups of System-calls, in order to detect whether an unknown software sample is malicious or benign, and to a further extent to classify it to a known malware family. A novel graph-based approach for the representation of software samples over the depiction of the structural evolution over time, the so-called Temporal Graphs, is discussed, and a method for measuring graph similarity among specific Regions of such graphs is proposed, the so-called Regional Matching. The partitioning of the Temporal Graphs that depicts their structural evolution over time is defined by specific time-slots, while the quantitative characteristics that depict the commonalities appeared over the weights of the vertices are measured by a similarity metric in order to conduct the malware detection and classification procedures. Finally, we evaluate the detection and classification ability of our proposed graph-based framework performing an experimental study over the achieved results utilizing a set of known malicious samples that are indexed into malware families.

CCS CONCEPTS

• **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; **Malware and its mitigation**; **Intrusion detection systems**; *Information flow control*; *Software reverse engineering*; • **Mathematics of computing** → **Hypergraphs**; *Graph algorithms*; • **Computing methodologies** → Cross-validation; • **Theory of computation** → **Pattern matching**; **Sorting and searching**; • **Information systems** → **Similarity measures**.

CompSysTech '21, June 18–19, 2021, Ruse, Bulgaria

ACM ISBN 978-1-4503-8982-2/21/06…$15.00
https://doi.org/10.1145/3472410.3472417

KEYWORDS

Malicious Software, Malware Detection, Malware Classification, Security.

ACM Reference Format:
Helen–Maria Dounavi, Anna Mpanti, Stavros D. Nikolopoulos, and Iosif Polenakis. 2021. Detection and Classification of Malicious Software based on Regional Matching of Temporal Graphs. In *International Conference on Computer Systems and Technologies '21 (CompSysTech '21), June 18–19, 2021, Ruse, Bulgaria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3472410.3472417

1 INTRODUCTION

Every day, thousands of new malware samples are developed by malware authors based on previous versions that inherit and preserve their main functionality utilizing mutations engines, or automated techniques and integrated tools [2] . The method followed over the gathering of information regarding the characteristics of malware distinguishes the two main types of malware analysis, namely the static and the dynamic malware analysis [16, 18] . Static analysis requires the examination of a given set of code artifacts (if available) [5] while the dynamic analysis [4] examines the interaction of a program with its hosting O.S. requiring its execution in a contained environment.

1.1 Protection against Malicious Software

The defense line against malicious software is based upon two basic techniques, namely malware detection and malware classification. The deployed malware detection approaches are based on the discrimination of either signatures, patterns or other static characteristics of the samples, or behaviour-based approaches that mainly focus on the detection of specific interactions (i.e., system-calls or API calls) that an executed program exposes [5]. Comparing the two approaches, it is observable that signature-based malware detection provides real-time protection against malicious threats but is less resilient against mutations of malicious samples than behaviour-based malware detection [2, 17] . On the other hand, malware classification mostly refers to the indexing of malicious samples into malware families that include malicious samples of the same functionality [7] , malware classification is an necessary

prerequisite on the development of generic signatures of malicious samples that cover a greater range of a malware family and probably potential mutations of its members [15].

1.2 Related Work

Through the recent literature, in malware detection, an algorithm for the extraction of common behavior graph alongside a graph matching algorithm for the computation if maximum weight subgraph is proposed by Ding *et al.* [6] in order to detect malicious code. Based on metrics over quantitative data flow graphs, Wüchner *et al.* developed a malware detection method presented in [20], while another behavioral malware detection technique is proposed in [19] based on the incremental construction of aggregated quantitative data-flow graphs. A novel detection method for android malware is proposed by John *et al.* [10] that utilizes graph convolutional nets based on centrality measures of the graph as input features. In malware classification, among various proposed models, in [9], Islam *et al.* utilize pattern recognition algorithms and statistical methods from function length and printable strings to develop an automated technique for malware classification. In [12], Nataraj *et al.* visualize malware binaries as gray-scale images and classify malware samples utilizing image processing techniques. Hassen and Chan, in [8], utilize function clustering to develop a linear time function-call graph vector representation combining graph features with non-graph features using an algorithm that finds the maximum weight sub-graph.

1.3 Contribution

In this work we design and propose a graph-based framework that utilizing graph representations of malicious software distinguishes malicious from benign samples, and further classifies the ones detected as malicious to a known malware family. The proposed model utilizes a specific type of graph representations of the structural evolution of dependency graphs, the so-called Temporal Graphs, deployed to depict the behavior of malicious samples. The technique deployed for the detection and the classification of malicious samples utilizes the partition of dependency graphs into Regions that depict specific states of their structural evolution over time. For a given software sample, we focus on the search of similar Regions in a set of known malicious samples across the whole extent of their dependency graph representations. Our intuition is that since the characteristic functionality of a malicious sample is indicated over a specific sub-graph through its dependency graph, then, during the mutation procedure, in order for the main functionality to be retained, this characteristic sub-graph should also appear throughout the whole extent of the mutated sample. Performing graph-similarity across such graph Regions, after a series of experiments, we achieved quite promising detection and classification results, that reinforce us with valuable information about the insights of malware mutations.

2 THEORETICAL BACKGROUND

In this Section we present the prerequisite theoretical background that consists the basis of our proposed model regarding the representation of software samples by their dependency graphs and the methodology we utilize in order to partition these graphs into Regions that depict their temporal evolution.

2.1 Representation of Malicious Software

The interaction that a software exhibits with its hosting environment during its execution time can be depicted through the invoked system-call dependencies captured though dynamic taint analysis, constructing a directed acyclic graph (dag). As described in [1, 3, 11, 14], the vertex set of the ScDG is consisted by the system-calls traces invoked during the execution of the software sample, while the edge set is consisted by the communication (i.e., data-flow dependencies) among these system-calls. Having the ScDG graph, let G, constructed for a software sample under consideration, utilizing the technique proposed in [11, 13] a graph abstraction of ScDG graph can be constructed by grouping disjoint subsets of its vertices (i.e. system-calls) based on their functionality into system-call groups. Utilizing these groups as super-vertices, the vertex set of the so-called Group Relation Graph (GrG), that we denote by G^*, is constructed, and based upon their inter-communication, an edge is added among the corresponding vertices of the GrG graph, i.e., the System-call Groups, constructing finally its edge set. To this end we ought to refer that the produced graph resulting from the directed acyclic graph ScDG is a weighted directed graph, since for each call from a System-call of a group to a system call of another System-call group we increment the weight of the corresponding edge between these groups.

2.2 Structural Evolution of Dependency Graphs

Starting from the beginning of the development of the GrG graph, utilizing information depicted over the sequence of edges added primarily to its corresponding ScDG graph, for a given number n of time-points (in our case, $n = 2^i, \forall 1 \leq i \leq 6$) we can define specific Regions in the evolution of the GrG graph. For a set of time-points, let $p_1, p_2, \ldots, p_n$ we can construct n instances of a given GrG graph denoting them by $T(G^*)$ as: $T_1(G^*), T_2(G^*), \ldots, T_n(G^*)$, that depict the structural evolution of a graph in terms of edges, vertex-degrees and vertex-weights of the corresponding GrG graph until specific time-points. Through this procedure, a series of Temporal Graphs is constructed, depicting the Regions of the structural evolution of the GrG through time. Each of the n Temporal Graphs includes the edges and the weights (i.e., structural modifications) developed in the GrG graph from the invocation of the first system-call until the invocation the system-call performed at that time-point. Throughout this procedure it is easy for our model to capture information regarding the temporal evolution of the graph through its construction procedure, w.r.t its structural modification. Then we utilize the corresponding graph instances created, for the computation of similarity between specific Regions of the graph. Further, we can compare different graphs by matching either the corresponding Regions or by investigating if this Regions exists inside another greater Region. The depiction of the structural evolution of a GrG graph, is represented by an aggregation procedure of the system-all invocation performed from the beginning of the execution of the program until a specific time-point. Hence, the Region defined from the start to that time-point, let p_i, through this aggregating procedure will also coincide inside all the next Regions starting also from

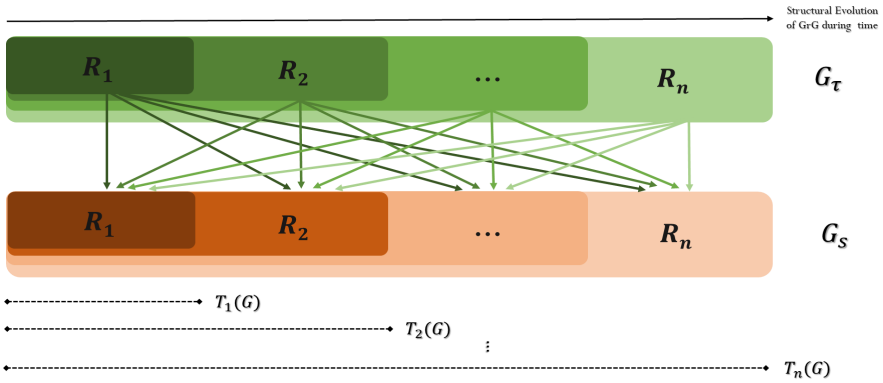

Figure 1: Regional matching method over Regions of various range.

the beginning of the execution and ranging until later time-points, let $p_{i+1}, p_{i+2}, \ldots, p_n$. Through this approach, our proposed model implements our intuition that if a characteristic sub-graph exists in an earlier point in the time-line of the evolution of a GrG graph of a malicious sample, then this characteristic sub-graph should also exist in a mutated strain independently of its chronological position (i.e., to the same or in a wider/later Region) in order to retain its main functionality.

3 SYSTEM ARCHITECTURE

In this section we present the architecture of the framework that deploys our proposed graph-based model for malicious software detection and classification based on Regional matching of Temporal Graphs.

3.1 Model Components

For the procedures of malware detection and classification we compare an unknown test sample, let τ, to known malicious ones, let s, utilizing their corresponding sequences of their Temporal Graphs, measuring the similarity among the defined Regions of the corresponding GrG graphs G^*_τ and G^*_s, as they are evolved over time. In order to measure the similarity of unknown samples and malicious samples, we consider their correlation regarding the quantitative characteristics exhibited among their Temporal Graphs, or, in other words, how similar is the creation of new edges or the increase of their weights in both graphs across their evolution. Denoting with $T(G^*_\tau)$, and $T(G^*_s)$, the Temporal Graphs of a given test sample τ and a known malicious sample s, respectively, we shall measure the similarity exhibited between $T(G^*_\tau)$ and $T(G^*_s)$ w.r.t. their quantitative characteristics utilizing the Cosine Similarity applying it over the weights of corresponding edges formed during the evolution of the GrG graph, measured as follows:

$$S(T(G^*_\tau), T(G^*_s)) = \frac{\sum_{i=1}^{n} w(e_{\tau,i}) \times w(e_{s,i})}{\sqrt{\sum_{i=1}^{n} w(e_{\tau,i})^2} \sqrt{\sum_{i=1}^{n} w(e_{s,i})^2}}, \quad (1)$$

where $w(e)$ refers to the weight of the edge $e = (u, v)$, and the edge e_i indicates the i-th edge in both Temporal Graphs (i.e., $e_{\tau,i}$ and $e_{s,i}$ for the Temporal Graph of the test and the Temporal Graph of the known sample, respectively) among the corresponding vertices $u_\tau, v_\tau \in V(T(G^*_\tau))$ and $u_s, v_s \in V(T(G^*_s))$, such that $(u_\tau, v_\tau) \in E(T(G^*_\tau))$ corresponds to the edge $(u_s, v_s) \in E(T(G^*_s))$.

Through the computation of the similarity between a test sample τ and a known sample s, we compute the similarity deploying our method to investigate if a characteristic sub-graph is located in a Region of a sample across the whole extent of the other. In our approach, we retain a number of graph instances that depict its structural evolution over time, with each one containing all the previous structural modification performed. Through the deployment of similarity measurement, we compute the similarity of each Region of the test sample, that potentially includes a characteristic sub-graph, with each Region of the malicious sample. Next, for each Region, let R_i, of the test sample, we register the maximum similarity computed with a Region of the malicious sample as $M_t = max\{S(T_t(G^*_\tau), T_k(G^*_s))\}, 1 \leq k \leq n$, where M_t is the maximum similarity exhibited between t^{th} Region R_t and the k^{th} Region R_k of the malicious sample. Iterating over the next Regions of the test sample, let $R_i, R_{i+1}, \ldots, R_n$, and the corresponding Regions of a specific known malicious we perform a comparison method that we call Regional Matching as we illustrate in Figure 1. To this end, we ought to notice that the general formula for the similarity computation between the Temporal Graph of test sample τ (i.e., $T(G^*_\tau)$) and the Temporal Graph of a known malicious sample s (i.e., $T(G^*_s)$) is computed as $S(T(G^*_\tau), T(G^*_s)) = \sum_{i=1}^{n} \{max\{S(T_i(G^*_\tau), T_j(G^*_s))\} : j \in [1, n]\}$, averaged by the n Regions, where $S(T_i(G^*_\tau), T_j(G^*_s))$ denotes the similarity between the R_i of G^*_τ with the R_j of G^*_s, while iterating over all the Regions of G^*_τ, $max\{S(T_i(G^*_\tau), T_j(G^*_s))\} : j \in [1, n]\}$ denotes the maximum similarity exhibited comparing each Region of the G^*_τ with each Region of G^*_s. Finally, in order to compute the maximum similarity exhibited between the test sample and any known malicious sample we iterate the computations of similarity between the test sample and all the known malicious samples of our knowledge base retaining the maximum value exhibited, as $\mathcal{S}_\tau = max\{S(T(G^*_\tau), T(G^*_s)), \forall\ 1 \leq s \leq \mathcal{M},\ s \in \mathbb{N}\}$, where $\mathcal{M}$ denotes the number of malicious samples in our knowledge base, recording also its malicious family name.

Figure 2: Detection results for the partitioning of Group relation Graphs into different numbers of Regions using the Cosine Similarity.

3.2 Malware Detection and Classification

In order to perform the malware detection procedure, we transform the initial ScDG graphs (i.e., G) to the corresponding GrG graphs (i.e., G^*), defining further the Regions that constitute the final graph representations that are the Temporal Graphs (i.e., $T(G^*)$), so for a given test sample, as also for our knowledge base that contains malicious samples. An unknown test sample is distinguished as malicious or benign based on the result of the computation of Cosine similarity metric when applied on the corresponding Regions of the Temporal Graph of the test sample and a set of Temporal Graphs that represent known malicious software samples, applying the procedure of Regional Matching across their Regions. Having computed the maximum similarity exhibited between the software sample under consideration and a known malicious sample, we compare this value against a pre-specified threshold value λ, where if the maximum value of similarity is above the value of λ the test sample is detected as malicious, or benign otherwise. Having detected a test sample as malicious, in the next phase, the one of malware classification, our proposed model decides the malware family in which the test sample should be indexed, based on the similarity exhibited by the samples belonging to each malware family. Based on the computation of Cosine similarity, our proposed model indexes the sample that has been detected as malicious in the previous phase selecting the malware family that contains the most similar sample (according to Cosine similarity) with the test sample, indexing the test sample into that family.

4 EVALUATION

For the evaluation procedure of our proposed malware detection and classification model, we utilized the same data-set used in [1] including 2631 malicious sample pre-classified into 48 malware families. For the evaluation of the detection procedure, regarding the detection ability of our model and the corresponding false-positive rates we took into account a set of 35 benign samples that cover a wide range of commodity software types. For the evaluation of the ability of our proposed model to index the test samples that have been detected as malicious into known malware families, we utilized the grouping of the 2631 malicious samples into 48 malware families by the utilization of heuristic rules as described in [1]. The evaluation of the procedures of malware detection and classification was performed implementing the five-file cross validation method, partitioning the data-set into an 80% train-set and a 20% test-set, averaging the achieved detection and classification results over the five folds, iterating the procedure over various partitioning of the Temporal Graphs into specific numbers of Regions.

4.1 Detection of Malicious Samples

In the plots presented in Figure 2 where is depicted the achieved detection ability of our model, the x-axis represents the values of threshold λ, while the y-axis represents the corresponding True Positive rates (TP-Rates) and False Positive rates (FP-Rates) depicting the percentage of the test samples that have been detected as malicious samples and indeed where malicious (red-color), alongside the percentage of benign samples that falsely detected as malicious (green-color), respectively. In Figure 2 there are presented the results achieved by the application of Cosine Similarity for the measurement of the similarity among Temporal Graphs partitioned into various number of Regions (i.e., 2, 4, 8, 16, 32, and 64) concerning the common quantitative characteristics appearing between test malicious and benign samples against known malicious samples represented also by their Temporal Graphs.

In Figure 2 (a) to (f) we present the achieved TP and FP-Rates exhibited by our proposed model for the partitioning of the Temporal Graphs into 2, 4, 8, 16, 32, and 64 Regions, respectively. More precisely, as we can see from the exhibited detection results of Figure 2 (a) and (b), for the partitioning of the Temporal Graphs into 2 and 4 Regions an adequately promising detection ability by an 90% TP-Rate and less than 10% FP-Rate is exhibited for the $\lambda = 0.995$, while for a more fine grained partitioning (i.e., partitioning of the

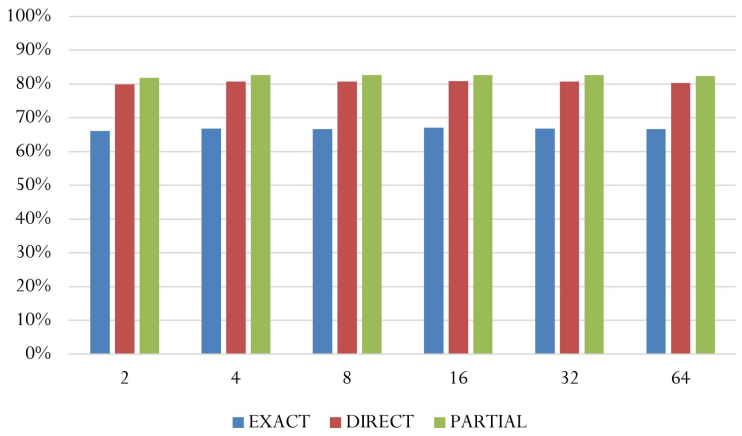

Figure 3: Classification results for the `Exact`, the `Direct`, and the `Partial Matching` classification types.

Temporal Graphs from 8 to 64 Regions) the detection ability is also retained on the same adequate levels exhibiting an 92% TP-Rate followed however by almost 18% FP-Rates. Finally, an interesting observation results from the fact that as we can observe in Figure 2 the application of a fine grained partitioning results to a more immediate decay in the number of FP-Rates for lower values of threshold λ resulting that this approach provides a better distinguishing ability among malicious and benign samples in that range of λ, (i.e., $\lambda \in [0.5, 0.9]$, whereas the detection rates are retained in obviously high rates. To this end, investigating the overall behavior of the detection procedure deployed by our proposed model, we can observe from Figure 2 that the exhibited TP-Rates appear to express a slower decay compared to the FP-Rates increasing the values of threshold λ across the number of Regions. Finally, through the increase of the number of Regions, the overall detection ability performs adequately well regardless of the precision of partitioning of the Temporal Graphs from more coarse grained to fine grained partitioning.

4.2 Classification of Malicious Samples

Since the 48 malware families that the malicious samples of our data set are pre-classified into have a label that is combined by two parts, and observing that malware families that tend to have common parts in their names also exhibit a similarity between their included samples, obviously as being classified according to their in common functionality, we define three different types of correct classification counting based upon the relation between the label of the family that the sample under consideration has been indexed in and its actual family's label, namely the `Exact`, the `Direct`, and the `Partial Matching`. More precisely, for a given test sample from malware family with label X_i, Y_i, if our model classifies it to a malware family with label X_j, Y_j, then the types of correct classification operate as the `Exact Matching`, the `Direct Matching`, and the `Partial Matching` that count a correct classification result if $(X_j \equiv X_i) \bigwedge (Y_j \equiv Y_i)$, if $(X_j \equiv X_i) \vee (Y_j \equiv Y_i)$, and if $(X_j \equiv X_i) \vee (Y_j \equiv X_i) \vee (X_j \equiv Y_i) \vee (Y_j \equiv Y_i)$, repetitively, computing in our performed evaluation experiments the three different values representing the classification ability of our proposed model.

In Figure 3 we present the achieved classification results exhibited by the application of the Regional Matching technique over various numbers of Regions (i.e., 2, 4, 8, 16, 32, and 64) of Temporal Graphs utilizing the Cosine similarity, averaged over the five folds. As we can observe from Figure 3, for the `Exact Matching`, the correct classifications are maximized for the partitioning of the Temporal Graph into 16 Regions achieving a 68% of correct classifications. For the case of `Direct Matching`, the correct classifications are maximized for the partitioning of the Temporal Graph into 4 and 16 Regions achieving an 81% of correct classifications. Finally, considering the `Partial Matching`, the correct classifications are maximized for the partitioning of the Temporal Graph into 4, 8, 16, 32 and 64 Regions achieving in all cases a 83% of correct classifications, results that are quite promising for the classification ability of our proposed model considering the commonalities exhibited among similar malware families.

5 CONCLUSION

In this paper we designed and developed a graph-based framework for malware detection and classification. We utilized the Temporal Graphs that depict the structural evolution of the ScDG graphs, and their corresponding GrG Graphs, over the execution time of the software sample under consideration. The exhibited detection and classification results achieved through a series of five-fold cross validation experiments provided a closer insight over the performance of our model against malicious samples and providing us with a feedback regarding the behavior of the proposed detection and classification techniques against mutated malicious samples. The promising exhibited results verified our intuitions leading us to focus to further improvements are oriented over the investigation of the deployment of other similarity metrics that may take account other types of graph characteristics of the Temporal graphs.

ACKNOWLEDGMENTS

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014- 2020" in the context of the project "Malicious Software Detection and Classification utilizing Temporal–Graphs of Discrete and Cumulative Structural Evolution" (MIS 5047642).

REFERENCES

[1] Domagoj Babić, Daniel Reynaud, and Dawn Song. 2011. Malware analysis with tree automata inference. In *International Conference on Computer Aided Verification*. Springer, 116–131.

[2] Mario Luca Bernardi, Marta Cimitile, Damiano Distante, Fabio Martinelli, and Francesco Mercaldo. 2019. Dynamic malware detection and phylogeny analysis using process mining. *International Journal of Information Security* 18, 3 (2019), 257–284.

[3] Alvaro Chysi, Stavros D Nikolopoulos, and Iosif Polenakis. 2020. An Algorithmic Framework for Malicious Software Detection Exploring Structural Characteristics of Behavioral Graphs. In *Proceedings of the 21st International Conference on Computer Systems and Technologies' 20*. 43–50.

[4] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. 2017. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques* 13, 1 (2017), 1–12.

[5] Baptiste David, Eric Filiol, and Kévin Gallienne. 2017. Structural analysis of binary executable headers for malware detection optimization. *Journal of Computer Virology and Hacking Techniques* 13, 2 (2017), 87–93.

[6] Yuxin Ding, Xiaoling Xia, Sheng Chen, and Ye Li. 2018. A malware detection method based on family behavior graph. *Computers & Security* 73 (2018), 73–86.

[7] Lars Strande Grini, Andrii Shalaginov, and Katrin Franke. 2018. Study of soft computing methods for large-scale multinomial malware types and families detection. In *Recent developments and the new direction in soft-computing foundations and applications*. Springer, 337–350.

[8] Mehadi Hassen and Philip K Chan. 2017. Scalable function call graph-based malware classification. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 239–248.

[9] Rafiqul Islam, Ronghua Tian, Lynn Batten, and Steve Versteeg. 2010. Classification of malware based on string and function feature selection. In *2010 Second Cybercrime and Trustworthy Computing Workshop*. IEEE, 9–17.

[10] Teenu S John, Tony Thomas, and Sabu Emmanuel. 2020. Graph Convolutional Networks for Android Malware Detection with System Call Graphs. In *2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP)*. IEEE, 162–170.

[11] Anna Mpanti, Stavros D Nikolopoulos, and Iosif Polenakis. 2018. A graph-based model for malicious software detection exploiting domination relations between system-call groups. In *Proceedings of the 19th International Conference on Computer Systems and Technologies*. 20–26.

[12] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*. 1–7.

[13] Stavros D Nikolopoulos and Iosif Polenakis. 2015. A graph-based model for malicious code detection exploiting dependencies of system-call groups. In *Proceedings of the 16th International Conference on Computer Systems and Technologies*. 228–235.

[14] Stavros D Nikolopoulos and Iosif Polenakis. 2017. A graph-based model for malware detection and classification using system-call groups. *Journal of Computer Virology and Hacking Techniques* 13, 1 (2017), 29–46.

[15] Younghee Park, Douglas Reeves, Vikram Mulukutla, and Balaji Sundaravel. 2010. Fast malware classification by automated behavioral graph matching. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*. 1–4.

[16] Giorgio Severi, Tim Leek, and Brendan Dolan-Gavitt. 2018. M alrec: compact full-trace malware recording for retrospective deep analysis. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 3–23.

[17] Alireza Souri and Rahil Hosseini. 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-centric Computing and Information Sciences* 8, 1 (2018), 1–22.

[18] Guosong Sun and Quan Qian. 2018. Deep learning and visualization for identifying malware families. *IEEE Transactions on Dependable and Secure Computing* (2018).

[19] Tobias Wüchner, Martín Ochoa, and Alexander Pretschner. 2014. Malware detection with quantitative data flow graphs. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*. 271–282.

[20] Tobias Wüchner, Martín Ochoa, and Alexander Pretschner. 2015. Robust and effective malware detection through quantitative data flow graph metrics. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 98–118.