Foundational Models for Malware Embeddings Using Spatio-Temporal Parallel Convolutional Networks

Dhruv Nandakumar*[a], Devin Quinn[a], Elijah Soba[a],
Eunyoung Kim[a], Christopher Redino[a], Chris Chan[a],
Kevin Choi[a] Abdul Rahman[a], Edward Bowen[a]
[a]Deloitte & Touche LLP
*Corresponding author: dnandakumar@deloitte.com

***Abstract*—As the complexity of malicious tactics, techniques, and procedures (TTPs) continuously grows to evade detection, so does the need for advanced methods capable of capturing and characterizing malware behavior. Current methodologies for task-specific models in cyber threat detection are time-intensive to build and do not generalize well to other tasks. In this paper, the authors introduce a novel method that combines convolutional neural networks, standard graph embedding techniques, and a metric learning objective to extract meaningful information from network flow data and create strong embeddings characterizing malware behavior. These embeddings enable the development of highly accurate, efficient, and generalizable machine learning models for tasks such as malware strain classification, zero day threat detection, and closest attack type attribution as demonstrated in this paper. A shift from task specific objectives to strong embeddings will not only allow rapid iteration of cyber-threat detection models, but also allow different modalities to be introduced in the development of these models.**

I. Introduction

In today's interconnected digital landscape, the proliferation of malware poses a significant threat to the security and stability of computer networks and systems worldwide. As the complexity of malicious tactics, techniques, and procedures (TTPs) continuously grows to evade detection, so does the need for advanced methods capable of capturing and characterizing malware behavior. The current state of the art in malware classification and detection uses task specific objectives; however, this method fails to generalize to other downstream tasks involving the same malware class. It is crucial to create embeddings for malware behavior that enable the development of highly accurate, efficient, and generalizable machine learning models for a variety of downstream tasks. A shift from task specific objectives to generalized embeddings will not only allow rapid iteration of cyber-threat detection models, but also allow different modalities to be introduced in the development of these models.

In this paper, the authors introduce a novel method for creating embeddings of malware behavior using network flow telemetry. The embeddings capture behavioral similarities between similar malware strains while disambiguating distinct strains in the latent space. Furthermore, the authors demonstrate how the pre-trained embedding model can be used for transfer learning to other cyber-threat detection tasks with strong results. The methodology combines convolutional neural networks, standard graph embedding techniques, and a metric learning objective to extract meaningful information from network flows and separate them in the latent space for use in downstream tasks. The authors also introduce a novel methodology for feature engineering in the cybersecurity domain that orders network flow telemetry Spatio-Temporally, i.e., by how devices connect on a network both topologically and over time. Overall, the authors endeavour to introduce a methodology for representing malware-specific behaviour in the form of embeddings which can be used for a variety of downstream tasks including, but not limited to, malware classification.

The first few sections of this paper will focus on a literature review of related work, an exploration of the proposed methodology, and an overview of experimental design. The authors will then present findings and experimental results on the performance on the methodology's ability to create strong clusters as well as performance on downstream tasks. The paper will conclude with a summary and discussion of next steps.

II. Related work

There have been several past works related to classifying malware using a variety of different methodologies. Sethi et al. [1] utilized application programming interface (API) based features extracted from sandboxed malware samples to train Decision Tree and Random Forest based models to detect and classify malware strains with good results. Nguyen et al. [2] also utilize tree based techniques to classify malware strains using features generated from static analysis of malware executables. Ma et al. [3] and Or-Meir et al. [4] also propose novel approaches for malware classification using techniques such as system call analysis and attention-based models.

There has also been research conducted on malware classification using techniques that are conceptually related to the work presented here. Anderson et al. [5] utilized graph representations of malware instruction sequences to identify malware executions. They demonstrated strong classification results but limited their scope to identifying only one malware strain distinctly. Similarly, Ding et al. [6], Hu et al. [7], and Kinable et al. [8] all utilized graph-based feature vectors of system call graphs during malware executions to classify malware strains. In contrast with the work presented here, all of the above approaches use malware execution traces or static analyses to create graphs for classification instead of

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

network flow information. Furthermore, all the methods above are used to train models in a task specific manner and do not lend themselves to transfer learning or other objectives easily. The methodology proposed in this work, however, involves generating representations of malware behavior that can be used for multiple downstream tasks and is, therefore, conceptually distinct.

Similarly, several works [9] [10] [11] utilized convolutional neural networks to classify malware strains. However, the input features consisted of malware images which were grey-scale, bit-map representations of malware files. Hu et al. [12] used graph convolutional networks to classify malware strains with graphs constructed on data-flow information. The graphs were constructed to represent data flow within the device on which the malware executed, not across various devices. Prior work [13] [14] also utilized graph features on network flow data for zero day threat detection using autoencoders and metric learning. The approach demonstrated strong performance in identifying anomalies in network flow behavior but only reasonable performance in identifying specific malware types or zero day threats. Furthermore, the approaches utilized were task specific and prone to instability during training.

There has also been work done in the field of producing embeddings of malware behavior using reconstruction of malware binaries [15] or using text-based embeddings of open-source threat intelligence [16] [17]. However, these approaches either require extraction of the actual malware binary in production settings, which may be unfeasible, or are more predictive in nature rather than detective. This stands in contrast to our work which aims to provide a light-weight methodology for detecting malware behavior in near real time.

This paper aims to analyze behavioral differences of malware strains with respect to network connections and information flow across various devices. Our approach captures nuanced information regarding patterns of interaction with command and control servers, etc. which other methods do not. Furthermore, this paper also utilizes a metric learning objective to create embeddings that cluster malware and separate different strains. This allows our model to be more general and extensible to other tasks such as and Zero Day Threat Detection and Malware Classification.

III. Shortcomings and considerations in existing work and motivations for this work

IV. Methodology

A. Datasets

The datasets used for our novel architecture were required to contain flow-level information about each event (connection duration, port numbers, timestamp, and number of forward and backward bytes transmitted), the source and destination IP addresses, and attack class labels. Two datasets were used during training and evaluation. The first is Gigas, an organization proprietary dataset consisting of network data over five years. It represents more than 100 real malware sample detonations in our internal malware cyber-range. A subset containing a sample of 20 diverse malware classes and 10,000 examples per class were ordered by timestamp and used as training data.

The second dataset, Malnet, was collected in the form of packet capture files available on malware-traffic-analysis.net [18]. Packet Captures were collected for 16 malware types and converted to flow-level features using the CICFlowmeter tool [19]. Both the Malnet and Gigas datasets were labelled with class labels referring to the malware strain executed. Class imbalance was more prevalent in Malnet and served as a benchmark for evaluating model performance in imbalanced scenarios. The per-class support varied from 42,000 to 110 examples per class and had a median of 2,570 examples per class.

B. Feature engineering

The first novel approach we take is the construction of our directed network-connection graph for network flows captured during malware executions. The nodes on the graph represent distinct Internet Protocol (IP) addresses, and the directed edges represent an aggregation of flow level information between source and destination IP pairs. In this paradigm, duplicate edges can exist between nodes if multiple connections exist between two IP addresses. Each edge is weighted using the connection's network flow attributes as given by the following formula:

$$weight(edge) = \frac{sourceBytes - destinationBytes}{\alpha^{duration}}$$

where α is a hyperparameter, *duration* is the duration of a connection in seconds, and *sourceBytes* and *destinationBytes* represent the amount of information, in bytes, sent from source to destination and vice-versa. Next, we compute an embedding for each node on the graph using FastRP. FastRP begins by assigning random vectors as embeddings to each node and iteratively averages over a nodes neighbors. The dimension of the embeddings are a tunable hyperparameter and can be modified based on intended downstream use; in this work, we refer to the embedding dimension as ϵ. A node's ϵ-dimensional embedding is a combination of its vector and the average embedding vector of its neighbors. The final embedding of a node *n* is given by:

$$embedding_n = weight_0 \cdot norm_{l2}(vector_{initial}) + \sum_{i=1}^{n} weight_i \cdot norm_{l2}(embedding_i)$$

where $embedding_i$ is the intermediate embedding of the node given neighbors at the i^{th} degree and $vector_{initial}$ is the initial random vector assigned to the node. It follows that embeddings of nodes seen during inference would be consistent with those generated during training only if the IP behavior in inference data is similar to training data. However, we believe that this assumption is reasonable given the malware behavior is broadly consistent with the objectives of the malware rather

than the type of the device on which it executes. The node embeddings will form the basis of the features used in the remainder of this work.

A key benefit of using FastRP to produce node embeddings is the ability to remove the normalization of raw features. Previously proposed methods rely on the normalization of network flow behavior between several networks to compare them to behavior of malware executions. This methodology can be unreliable, especially when the structure and behavior of the originating networks are vastly different. Our approach eliminates this concern by observing network flow behavior on a per-asset level for a given duration and producing node embeddings for those flows compared to malware behavior. Furthermore, using FastRP removes the need for specific graph feature engineering as seen in the work by [13] and [14] and allows models to learn from richer behavior-specific embeddings.

C. Spatio-temporal example creation

Once all nodes from malware executions have emeddings, we begin the creation of training and evaluation examples. Malware executions are differentiated not only by the characteristics of individual connection, but also the order in which they occur temporally and spatially ([20]). That is, we believe that we can generate richer embeddings of malware behavior if we consider sequences of network flows and the various devices they connect to as opposed to single network flows. Malware execution examples are represented such that they capture the interaction of various nodes on a graph exhibit.

For each malware execution, we first order the network flows in ascending order by timestamp and subset β flows. Of the β flows, we select the first γ IP addresses (either source or destination) seen in the connection. We then construct a (γ, ϵ) dimensional feature matrix, *F*, that contains the FastRP embedding for each selected IP address in the order in which they were seen. *F* now contains feature vectors for all nodes participating in the β flows temporally. Next, we construct a binary adjacency matrix for all nodes present in γ such that each entry i, j in the matrix will be given by the formula:

$$adjcency(i,j) = \begin{cases} 1 & \exists connection_{i,j} \in \beta \\ 0 & otherwise \end{cases}$$

The adjacency matrix, *A*, represents spatial connections between interacting nodes on a graph and has dimension (γ, γ). *A* and *F* form the input features for one example. Similarly, a sliding window of width β can be used to create several examples per malware execution for all malware executions with a corresponding label.

D. Model architecture

We propose a spatio-temporal parallel convolutional network (ST-PCN) architecture (Figure 1) that processes the spatial and temporal aspects of malware execution graphs in parallel for the creation of strong embeddings. Our model architecture begins with two sets of convolutional layers that convolve

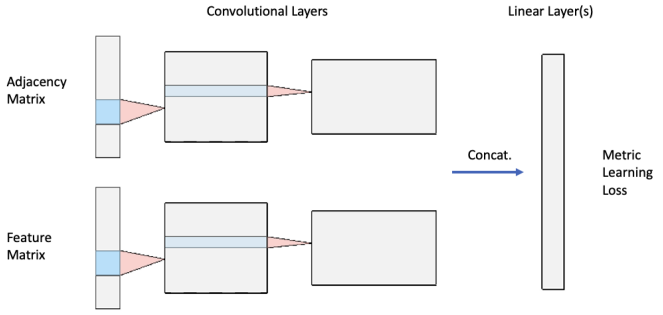

Fig. 1: ST-PCN Architecture

A and *F* independently to produce a 32 dimensional vector embedding for each matrix. Both vector embeddings are then concatenated to produce a final spatio-temporal embedding which is passed into the metric learning objective function. It should be noted here that, while the concatenation of inputs from Neural Networks for multi-modal learning [21] has been seen in literature previously, combining the architecture with the feature engineering and example creation for security purposes is a novel contribution.

E. Training and evaluation

The ST-PCN architecture is trained using a metric learning objective to maximize inter-class separation in the embedding space while simultaneously minimizing intra-class separation. Particularly, we utilize a softmax-based additive angular margin loss [22] as our loss function which is backpropagated through the entire ST-PCN model and is computed on the concatenated embedding and corresponding malware label.

Furthermore, we will introduce a 'holdout' malware class which will be held out from training and validation data entirely when training the ST-PCN models and evaluating them on downstream tasks. Downstream task evaluation will also assess the performance of our models on holdout data to estimate the performance of the approach on novel malware. This is particularly helpful when evaluating the efficacy of the ST-PCN on a zero day threat detection task. All experiments conducted below require a compute instance with at least 64 gigabytes of Random Access Memory (RAM) and a 32 core processor.

V. Experimental design

Training and testing are performed on a random 70/30 train-test split of input matrices. Given that the objective of the ST-PCN is to produce strong embeddings of malware behavior for downstream tasks, the primary evaluation will be focused on the embedding clusters and associate metrics such as silhouette scores, Rand indices, and cluster completeness and homogeneity scores. Furthermore, performance of the embeddings will also be evaluated based on the performance of complex downstream tasks. All downstream tasks will use pre-trained embeddings from the ST-PCN with no fine-tuning.

A. Embedding by attack class

The metric-learning objective involves training an ST-PCN model on a metric learning objective that aims to separate malware by their respective strain. Compared to recent work ([14]) which uses metric learning on cyber attack types such as botnet or ransomware attacks, this work uses more granular malware strain labels because each malware strain exhibits distinct behavior that could be used for multiple attack campaigns. For example, a Bazaloader trojan could be used in an overall attack campaign to deliver ransomware, keystroke loggers, or any other malware. We believe this will lead to better separability in the latent space. As mentioned in the Datasets section, we use malware labels from two datasets consisting of a total 36 malware classes.

The produced FastRP embeddings only consider a node's neighbors up to the second degree. Consequently, each FastRP embedding is a weighted sum of three intermediate embeddings: the node's initial vector, it's vector relative to it's neighbors, and it's vector relative to it's neighbors' neighbors. The weights of each vector are 1, 0.5, and 0.5, respectively. Furthermore, all experiments and results set α as 1.15, β as 128, and γ and ϵ as 32.

B. Malware classification

This downstream task evaluates the performance of pre-trained ST-PCN embeddings on classifying malware types. Evaluation is conducted on embeddings produced for the test-set of malware classes and the holdout malware class. In the malware classification experiments that follow, we utilize a Random Forest Classifier fit on the training set of embeddings.

C. Zero day threat detection

In this task, we evaluate the models' performance on differentiating a malware class seen during training to a holdout class. In order to estimate the performance of ST-PCN embeddings on a zero day threat detection task, we utilize a Euclidean distance-weighted K-Neighbor classification model fit on the training set of embeddings with the number of neighbors set to 350.

During evaluation, we compute maximum class membership probability for each test and holdout example on the training classes. The complement of this probability is then computed to represent the probability of an example being a Zero Day Threat, referred to henceforth as the ZDT probability. The ZDT probabilities are then thresholded to compute metrics. Given the class imbalance between test examples and holdouts, we utilize precision, recall, and area under the precision-recall curve as evaluation metrics.

D. Closest attack type attribution

The objective of this downstream task is to determine, for any given holdout example, what the most similar malware seen during training to it is. This allows us to evaluate the fidelity of the organization of the latent space and examine if similar malware types are truly located close to each other.

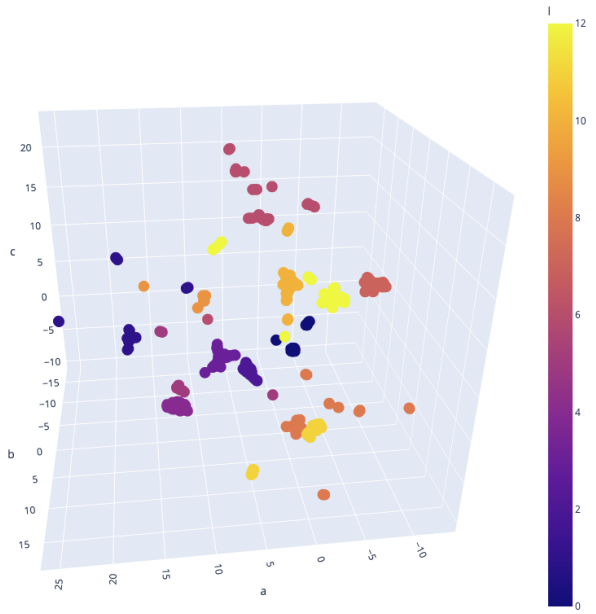

Fig. 2: Malnet dataset embeddings

The same K-Neighbors classification used in the Zero Day Threat detection task is used to compute the closest attack type and probability for every holdout example. Then, for each holdout class, we extract the two most common classifications and compute the average probability for each classification.

E. Additional considerations for experimental design

Several open source datasets included to evaluated performance in previous work do not lend themselves will to the ST-PCN methodology, particularly due to lack of source and destination IP address fields in the data. Examples of popular datasets include CICIDS2017, NSL-KDD, and KDD-99 which have been used extensively in prior work [23] [24] [25] [26]. Consequently, this work does not benchmark model performance on those datasets. Future work will include benchmarking other proposed methods in the literature using the Malnet dataset.

Furthermore, it is also important to note that the scope of this work is not to propose a model for the sole purpose of malware classification or ZDT detection. Instead, this work aims to provide a generalized methodology for creating representations of malware behavior for downstream use. Some examples of this use could be, but are not limited to, malware severity estimation, malware family classification, or other multi-modal threat detection tasks.

VI. Results and Discussion

A. Cluster and embedding analysis

In order to visualize and evaluate the embeddings produced by the ST-PCN model, we produce embeddings for all example matrices in the test-set. Once embeddings are computed, we utilize Uniform Manifold Approximation and Projection (UMAP) ([27]) to produce 3-dimensional representations of the

TABLE I: Zero Day Threat detection and CATA metrics

Dataset	Holdout	AUC	Precision	Recall	CATA	Probability
Gigas	Nanocore	0.90	0.90	0.85	Meterpreter	91%
Gigas	Azorult	0.96	0.99	0.91	Ursnif	88%
Gigas	Ursnif	0.89	0.98	0.72	Nymeria	91%
Gigas	Trickbot	0.96	0.98	0.92	Xtremerat	93%
Gigas	Lokibot	0.95	0.99	0.92	Netwire	88%
Malnet	Bazaloader	0.97	0.96	0.94	SquirrelWaffle	81%
Malnet	Astaroth	0.82	0.95	0.70	Hancitor	96%
Malnet	Mantabuchus	0.68	0.76	0.65	Monsterlibra	76%
Malnet	Valak	0.92	0.94	0.91	Hancitor	90%
Malnet	Qakbot	0.88	0.93	0.84	Gozi	89%
Average		**0.89**	**0.94**	**0.83**		

TABLE II: ST-PCN embedding metrics

Dataset	Silhouette	Completeness	Homogeneity	Rand
Gigas	0.69	0.86	0.94	0.97
Malnet	0.66	0.71	0.97	0.96

TABLE III: Malware classification metrics on Gigas test data

Value	AUC	Precision	Recall
Macro Avg.	0.99	0.99	0.99
Minimum	0.98	0.97	0.97

embeddings for visualization. A visualization Malnet datasets is shown in Fig 2, where the x, y, and z axes each correspond to a dimension of the 3-dimensional UMAP embedding. In the figure, the points are colored by malware class. We also see similar results in the Gigas dataset.

From the figures, we see that the ST-PCN produces well separated clusters for malware classes in both datasets. We also see that a few malware strains tend to have multiple tight clusters as opposed to a single larger cluster. We believe that this is due to sub-variations in malware behavior based on external factors such as the operating system on which they execute. In general, however, malware classes are well separated into either tight clusters or slightly diffuse cluster clouds. This analysis is also supported by strong evaluation metrics presented in Table II.

TABLE IV: Malware classification metrics on Malnet test data

Value	AUC	Precision	Recall
Macro Avg.	0.99	0.99	0.99
Minimum	0.98	0.94	0.97

TABLE V: Malware classification metrics on Gigas test data with holdout

Value	AUC	Precision	Recall
Macro Avg.	0.95	0.97	0.91
Minimum	0.40	0.32	0.15

TABLE VI: Malware classification metrics on Malnet test data with holdout

Value	AUC	Precision	Recall
Macro Avg.	0.93	0.93	0.90
Minimum	0.31	0.19	0.25

B. Malware classification

Performance of the ST-PCN model on the classification task showed consistently strong performance on complete test sets of both Gigas and Malnet datasets with 19 and 13 classes respectively. Macro and Lowest class-specific precision, recall, and AUC scores are provided in Tables III and IV.

In order to test performance of the ST-PCN when a novel class is present in training data, we re-ran the above experiment with introducing a distinct holdout class in the embedding data and training data for the classification model 5 times. Aggregate results show good performance on both datasets but with a noticeable drop in performance in tables V and VI. However, overall metrics still indicate that the ST-PCN can generalize with reasonably strong performance.

C. Zero day threat detection

The Zero Day Threat Detection task was carried out and evaluated for 5 different holdouts per dataset independently. This borrows directly from the methodology described in literature previously, which showed robust experimental results [13] [14]. Overall, the ST-PCN exhibited consistently strong performance on the Gigas dataset. Performance of the model on the Malnet dataset was also generally strong, but some classes had lower performance. This is ostensibly due to the significantly lower number of training examples per class in the Malnet dataset compared to Gigas. All results are shown in Table I in the *AUC*, *Precision*, and *Recall* columns. Furthermore, we already see strong performance improvements compared to recent work by [13] and [14] with no fine tuning and a relatively simple classification model.

D. Closest attack type attribution (CATA)

The CATA for every holdout evaluated in the Zero Day Threat Detection section is provided in Table I in the *CATA* and *Probability* columns. For each holdout, the closest attack

type in the training set is provided alongside a probability score.

Analyzing some results above reveals that the embedding space produced by the ST-PCN is, in fact, well-ordered by behavior. For example, we see that the downstream model indicated that the *Nanocore* holdout is closest to the *Meterpreter* malware. External analysis indicates that both malware types are Remote Access Trojans (RATs) and therefore exhibit similar behavior. We see similar results for

- *Bazaloader* and *SquirrelWaffle* which are both 'malware downloaders' which perform the function of downloading malware execution suites from remote sources.
- *Astaroth* and *Hancitor* which are also software packages used to facilitate dropping or downloading of malware.
- *Qakbot* and *Gozi* which are malware strains used for information stealing and keylogging as part of larger attack campaigns.

We believe that enriching model results with CATA attributions allow for enhanced model interpretability as well as increased utility of model alerts to end users such as Threat Hunters who could use model outputs for further investigation in an network.

VII. Conclusion, Limitations, and Future Work

In this work, we have presented a novel, foundational method for producing embeddings of network-agnostic malware behavior for downstream use. In the proposed methodology, we utilize graph-based embeddings of asset behavior as a feature engineering step to a ST-PCN. We also demonstrate the strong performance of the embedding model both independently and using historically non-trivial tasks such as Zero Day Threat Detection and CATA. We believe our results show the potential to prevent cyber-operator fatigue and allow rapid iteration of malware-specific machine learning techniques.

While the metrics on downstream tasks indicate strong results, we hypothesize that more complex, task specific architectures can achieve even better performance. The out of the box methods used in our experiments primarily leverage the well organized latent space to perform their respective tasks. However, task specific architectures can further manipulate the latent space and extract meaningful features in a way that could not be done with our reported methodologies. In tasks such as Zero Day Threat Detection, even a small increase in performance can have a big impact.

Furthermore, we also hypothesize that incorporating multi-dimensional or multi-modal raw feature data will further strengthen the ST-PCN model. Data such as malware severity scores or endpoint based logs can provide context about malware executions beyond network flow patterns. Incorporation of this type of data could not only alleviate the strict dependence on network flow data, but also introduce new downstream tasks our model can generalize to.

A limitation to this work is the limited availability of malware strains present in our training and validation data. Although we made a concerted effort to capture a wide diversity of malware, the set of known and unknown malware is larger than the data available to us. Adding more strains has the potential to make our embeddings more general but could impact performance of the model. Consequently, further training and validation on wider sets of malware data is an important next step in the development of this framework. Some methods that future work will explore to alleviate this concern include, but are not limited to, training with data augmentation techniques such as synthetic or simulated attack generation or using other few-shot learning techniques such as Model Agnostic Meta-Learning [28]. Furthermore, the authors also believe that online training of the ST-PCN model run during deployment using Federated Learning [29] could greatly combat the dearth of labelled training data by learning attack patterns across deployments while alleviating data privacy concerns.

Future work will aim to improve the ST-PCN architecture and performance by incorporating multi-modal data and introducing more complex downstream task models. In addition, we will attempt to use more malware strains and investigate how it affects our performance across downstream tasks. If performance maintains or exceeds what is reported, it will indicate that our model is a good candidate for generalizing malware in cyber specific machine learning applications.

References

[1] K. Sethi, R. Kumar, L. Sethi, P. Bera, and P. K. Patra, "A novel machine learning based malware detection and classification framework," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1–4.

[2] C.-D. Nguyen, N. H. Khoa, K. N.-D. Doan, and N. T. Cam, "Android malware category and family classification using static analysis," in *2023 International Conference on Information Networking (ICOIN)*, 2023, pp. 162–167.

[3] X. Ma, Q. Biao, W. Yang, and J. Jiang, "Using multi-features to reduce false positive in malware classification," in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, 2016, pp. 361–365.

[4] O. Or-Meir, A. Cohen, Y. Elovici, L. Rokach, and N. Nissim, "Pay attention: Improving classification of pe malware using attention mechanisms based on system call analysis," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[5] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *Journal in Computer Virology*, vol. 7, pp. 247–258, 11 2011.

[6] Y. Ding, X. Xia, S. Chen, and Y. Li, "A malware detection method based on family behavior graph," *Computers and Security*, vol. 73, pp. 73–86, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817302146

[7] X. Hu, T.-c. Chieuh, and K. G. Shin, "Large-scale malware indexing using function-call graphs," 2009. [Online]. Available: http://www-personal.umich.edu/~huxin/papers/xin_SMIT.pdf

[8] J. Kinable, "Malware classification based on call graph clustering," 2010. [Online]. Available: https://arxiv.org/pdf/1008.4365.pdf

[9] S. A. Roseline, A. D. Sasisri, S. Geetha, and C. Balasubramanian, "Towards efficient malware detection and classification using multilayered random forest ensemble technique," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–6.

[10] M. Alam, A. Akram, T. Saeed, and S. Arshad, "Deepmalware: A deep learning based malware images classification," in *2021 International Conference on Cyber Warfare and Security (ICCWS)*, 2021, pp. 93–99.

[11] W. W. Lo, X. Yang, and Y. Wang, "An xception convolutional neural network for malware classification with transfer learning," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2019, pp. 1–5.

[12] X. Hu, T.-c. Chieuh, and K. G. Shin, "Large-scale malware indexing using function-call graphs," 2020. [Online]. Available: http://www-personal.umich.edu/~huxin/papers/xin_SMIT.pdf

[13] C. Redino, D. Nandakumar, R. Schiller, K. Choi, A. Rahman, E. Bowen, A. Shaha, J. Nehila, and M. Weeks, "Zero day threat detection using graph and flow based security telemetry," in *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2022, pp. 655–662.

[14] D. Nandakumar, R. Schiller, C. Redino, K. Choi, A. Rahman, E. Bowen, M. Vucovich, J. Nehila, M. Weeks, and A. Shaha, "Zero day threat detection using metric learning autoencoders," in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2022, pp. 1318–1325.

[15] E. Downing, Y. Mirsky, K. Park, and W. Lee, "Deepreflect: Discovering malicious functionality through binary reconstruction," in *Proceedings of the 30th USENIX Security Symposium*, ser. Proceedings of the 30th USENIX Security Symposium. USENIX Association, Jan. 2021, pp. 3469–3486.

[16] Y. Shen and G. Stringhini, "Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks," ser. SEC'19. USA: USENIX Association, 2019, p. 905–921.

[17] N. Tavabi, P. Goyal, M. Almukaynizi, P. Shakarian, and K. Lerman, "Darkembed: Exploit prediction with neural language models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11428

[18] B. Malware Traffic Analysis, "Malware traffic analysis," 2023. [Online]. Available: https://www.malware-traffic-analysis.net/

[19] H. L. Arash, D. G. Gerard, S. I. M. Mohammad, and A. G. Ali, "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*, INSTICC. SciTePress, 2017, pp. 253–262.

[20] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," 1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-70542-0_6#citeas

[21] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:45998148

[22] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4685–4694.

[23] A. G. P. Lobato, M. A. Lopez, I. J. Sanz, A. A. Cardenas, O. C. M. B. Duarte, and G. Pujolle, "An adaptive real-time architecture for zero-day threat detection," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[24] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, p. 1684, 10 2020.

[25] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3854–3861.

[26] Z. Zhang, Q. Liu, S. Qiu, S. Zhou, and C. Zhang, "Unknown attack detection based on zero-shot learning," *IEEE Access*, vol. 8, pp. 193 981–193 991, 2020.

[27] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," Sep 2020. [Online]. Available: https://arxiv.org/abs/1802.03426

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:6719686

[29] M. Vucovich, A. K. Tarcar, P. Rebelo, N. R. Gade, R. Porwal, A. Rahman, C. Redino, K. Choi, D. Nandakumar, R. Schiller, E. Bowen, A. West, S. Bhattacharya, and B. Veeramani, "Anomaly detection via federated learning," *ArXiv*, vol. abs/2210.06614, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252872967