GLEAM: GAN and LLM for Evasive Adversarial Malware

Dharani Devadiga*
Green Level High School
Cary, North Carolina
dhar.devadiga@gmail.com

Gordon Jin*
Seven Lakes High School
Katy, Texas
gordonj2016@outlook.com

Bisti Potdar*
Vernon Hills High School
Vernon Hills, Illinois
bisti12018@gmail.com

Hankyu Koo
Bellarmine College Prep.
Cupertino, California
hankyukoo@gmail.com

Andrew Han
Westlake High School
Austin, Texas
andrewhanlt@gmail.com

Anusha Shringi
Lynbrook High School
San Jose, California
Anushashringi@gmail.com

Angad Singh
Robbinsville High School
Robbinsville, New Jersey
angad.thirteen@gmail.com

Kinjal Chaudhari
University of Illinois Urbana-Champaign
Champaign, Illinois
chaudharikinjal2004@gmail.com

Saurav Kumar
University of Illinois Urbana-Champaign
Champaign, Illinois
sauravk4@illinois.edu

***Abstract*—The rapid evolution of cybersecurity threats poses formidable challenges for effective malware detection. Traditional methods often struggle to keep pace with the continuously changing landscape of new malware variants. To address this issue, researchers have turned to machine-learning techniques to optimize malware detection. Yet, these approaches have overlooked the fact that evasive malware is meticulously crafted to elude detection through tactics such as exploiting software vulnerabilities, utilizing encryption, and employing obfuscation techniques. Malware authors have a strong incentive to attack malware detection systems, yet the features and methods that they exploit are limited. We propose a model named GLEAM- GAN and LLM for Evasive Adversarial Malware. This model infuses hex code and opcode features with LLM (Large Language Model) embeddings and GANs (Generative Adversarial Networks) to generate synthetic samples that closely resemble evasive malware to bypass black-box machine learning detectors. Through extensive evaluation, our model achieved an average evasion rate increase of 22.6%, demonstrating its ability to effectively attack detection systems. By expanding the space for adversarial malware generation, we give modern detection systems the capability to counter the nuanced tactics of evasive malware, thus enhancing proficiency in preempting and neutralizing potential threats with heightened precision.**

* ***Index Terms*—Generative adversarial networks, Large Language Models, Cybersecurity, Adversarial Malware Examples**

I. Introduction

Within the intricate ecosystem of computer systems and networks, the constant presence of malware poses an imminent threat to the security landscape. Spanning a diverse array of forms, including viruses, worms, trojans, ransomware, and spyware, malware orchestrates a sophisticated infiltration of digital infrastructure, highlighting the criticality of robust detection and mitigation strategies in the realm of cybersecurity.

Traditional approaches to malware detection have primarily relied upon the matching of known patterns or signatures associated with documented malware instances. However, this approach implies that those malware instances have been previously detected, by which point major damage will have been inflicted on the machine. Furthermore, this flaw renders the approach ineffective in discerning emerging and novel malware variants, consequently exposing vulnerable systems to the dangers of highly sophisticated attacks. As such, a pressing need has emerged for advanced detection techniques capable of effectively identifying evasive malware.

In recent years, the research community has diverted its attention toward harnessing the potential of machine learning algorithms, notably the practices of deep learning and neural networks, to bolster the efficacy of malware detection mechanisms. [1],[2] These cutting-edge approaches leverage the innate capacities of pattern recognition and anomaly detection to expose potentially malicious behaviors. While exhibiting promising outcomes in terms of detection accuracy enhancement, they remain constrained when confronted with the evasive characteristics of malware deliberately designed to evade contemporary detection systems, known as adversarial malware examples. [3],[4] This spurred a new field of adversarial malware generation using gradient-based GANs (Generative Adversarial Networks) pioneered by Goodfellow et al. [5] Subsequently, Zhong et al. proposed MalFox, which creates adversarial Windows Portable Executable files, or PE files, without the need to convert their features into an image, as is done commonly with the traditional GAN. [6] Hu et al. proposed MalGAN which extracts features from API function

*These co-first authors are listed in alphabetical order and contributed equally to this work.

calls that are translated into a feature set to generate adversarial examples with GAN. [7] Zhu et al. used n-gram feature extraction on hex codes as a simple static analysis approach to adversarial malware generation.[8] However, many of these approaches are limited by their scope of feature sources and only explore possibilities with the GAN model- a model susceptible to mode collapse and catastrophic forgetting.

To address the challenges imposed by sophisticated malware detection systems as well as limited adversarial malware generation tactics and feature sources, this research project pursues developing highly functional GANs as well as LLMs combined with a synthesis of hex code and opcode features to generate adversarial malware examples through our GLEAM model. This specifically structured framework generates synthetic malicious data, with the sole purpose of eluding detection systems. Through the creation of realistic and novel evasive malware samples, this project aims to provide the cybersecurity field with a robust evaluation framework to scrutinize the accuracy of detection systems, thereby fortifying overall defense against the incessant onslaught of cyber threats.

Utilizing a balanced dataset, our research incorporates novel techniques to address the inherent complexity of extracting local features from such malicious agents. Drawing upon the profound knowledge and findings of existing literature, such as the application of GANs in enhancing the accuracy of malware classification and the obstacles imposed by static and dynamic analysis, we strive to maximize the effectiveness of our approach. Ultimately, the GLEAM framework serves to bolster cybersecurity defenses and mitigate the ever-growing threats posed by malicious attacks.

The remainder of this paper is structured as follows: Section II discusses the pertinent literature for this field, Section III describes our approach in detail, Section IV evaluates the performance of GLEAM, and Section V concludes the paper with a future research discussion.

II. Literature Review

In 2015, Buczack and Guven conducted a comprehensive survey focused on cyber analytics for intrusion detection. The study highlighted three primary approaches: misuse-based (signature-based), anomaly-based, and hybrid techniques. Misuse-based techniques excel at detecting known attacks, but face challenges when encountering novel (zero-day) attacks. On the other hand, anomaly-based techniques model normal system behavior and identify deviations as anomalies, offering the advantage of detecting zero-day attacks, but burdened with the risk of high false alarm rates. Hybrid techniques combine both of these approaches to improve detection rates for known and unknown attacks as a whole.

In the aftermath of the WannaCry malware that garnered global attention, Lu et al. conducted a study exploring the value of the BERT model in constructing malware datasets. They addressed the limitations of existing malware detection strategies and focused on how difficult it is to obtain malware samples and their API sequences due to obfuscation techniques and the ever-changing variants present within the malware.[9] The BERT model utilizes a public API call sequence dataset of obfuscation-free malware as input, using adversarial training in conjunction to enhance its performance. It averaged 87 percent accuracy, higher than other classification methods such as KNN and CNNs.

In 2023, Smith et. al. conducted a research study further exploring the challenges that malware poses, specifically due to the increasing amount of malware variants that go undetected by conventional methods. This study conducted a comparative analysis of two malware-related datasets, evaluating their correlation with malware samples. They were able to achieve a high correlation with the first batch of data by using both supervised and unsupervised learning algorithms, such as K-Means and Random Forest. The Random Forest achieved a 99.88 percent accuracy on the Malware-Exploratory dataset and a 99.99 percent accuracy on the CIC-MalMem-2022 dataset. This paper is able to achieve very high accuracy rates and improve the effectiveness of malware detection systems, but further research is needed to explore how these algorithms can be applied to large, realistic, and more diverse malware datasets.

Jang et al. utilized a GAN model for improved accuracy and effectiveness in the analysis of malware detection and classification. [10] This paper introduces global-image-based local feature visualization and a global and local image merging approach. They aim to enhance malware classification by utilizing global and local images generated from byte and assembly language source files. They achieved a 100 percent accuracy in unobfuscated malware datasets and a 96.87 percent accuracy in obfuscated malware. First, they extracted the ASM/byte files and then created global and local images. The global images included information about the malware obtained from binaries, while the local images presented information such as API functions and opcodes. [11] The GAN model then merged the global and local images, ultimately training a Convolutional Neural Network on malware classification.

III. Method

This section outlines the rationale behind our dataset selection, the pre-processing steps for that specific dataset, the black-box detectors and how they were paired with the GAN, and finally, our large learning model.

A. Dataset

The PE files were sampled from the PE Malware Machine Learning Dataset by Michael Lester published by Practical Security Analytics LLC. To create samples of evasive malware, we obtained raw labeled portable executable files from Michael Lester's dataset of Practical Security Analytics [12]. Specifically, from the total of 200000 PE files, we randomly sampled 5000 malicious and 5000 benign files, of which 70% is used as the training set and 30% as the testing set.

B. Black-box detector

The dataset was run through five anti-malware machine-learning algorithms considered as the black-box detector whose internal parameters are invisible: Logistic Regression (LR), K-Nearest Neighbors (KNN), Random Forest (RF), Decision Trees (DT), and Support Vector Machine (SVM).[13],[14],[15] The True Positive Rates (TPRs) on each of these detectors were 84%, 78%, 92%, 81%, and 93%, respectively.

C. GAN

GLEAM applies GAN to generate adversarial malware examples. We expand the feature extraction process by incorporating hex code and opcode n-gram features in our feature matrix.

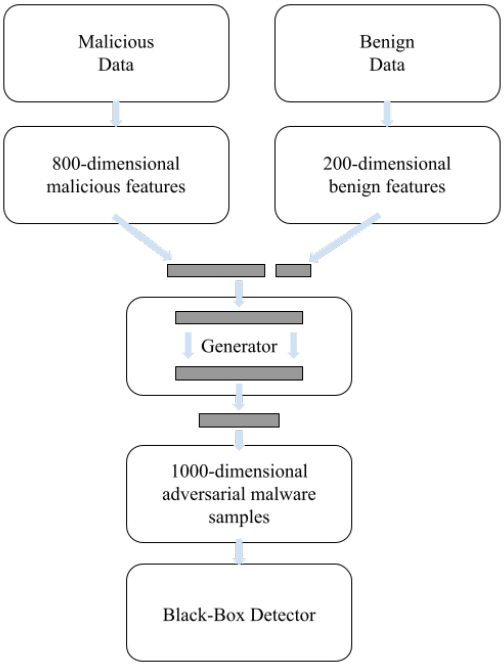

Fig. 1. The architecture of the GLEAM GAN

1) N-gram: In the field of Natural Language Processing (NLP), n-grams are utilized for text generation such as in the recent chatbot, Chat-GPT. It analyzes the frequency and co-occurrence patterns of a sequence of words, providing crucial information about relevant phrases in a document. We borrow this tool for cybersecurity purposes. N-gram feature selection removes the step of translating the PE files into a different form, such as the popular method of making malware "images"[16], and allows the model to work on the raw data itself.

2) Hex code: The first source of feature extraction used in our GLEAM model is the hexadecimal representation of machine code. Each hexadecimal value contains meaningful information about the PE file, such as instruction codes and data. We use a simple Python program to extract hexadecimal codes from the PE files, and 6 grams are obtained as they perform best in malware classification[17].

3) Opcodes: The second source of feature extraction for our GLEAM model is the assembly language source code, which contains the symbolic machine code of the PE files. Opcodes are contained within the assembly code and represent the rudimentary function calls made by the program. They are similarly extracted in Python by reading the first non-byte token in each line of the assembly file. 2-grams are selected to represent opcode features as they are shown to have the highest accuracy in malware classification[18].

4) Feature extraction: We combine hex codes and opcodes into a 1000-dimensional feature matrix. [19] We use a 600:100 split between malicious and benign hex codes and a 200:100 split between malicious and benign opcodes. This feature matrix is used to train the GLEAM GAN.

5) GLEAM: We combine an 800-dimensional malicious feature vector with a 200-dimensional benign feature vector and map each sample, benign or malicious, to the new 1000-dimensional vector. To generate adversarial examples and maintain malicious content, we take the logical 'OR' operation between malicious and benign feature matrices. The new feature matrix is fed into the generator. The GLEAM GAN consists of a generator and a black-box detector. The generator is a feed-forward neural network that uses CopulaGAN, a variation of the CTGAN, which is introduced in the SDV open-source library. [20] It uses the Cumulative Distribution Function (CDF)- based transformation, which is applied through GaussianCopula, allowing it to learn the correlation between random variables[21].

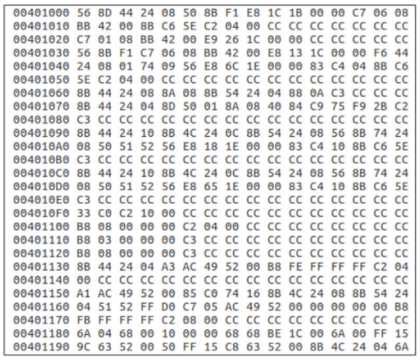

Fig. 2. Hexadecimal representation of PE binaries

D. LLM

1) Language Model Initialization: As a preliminary step, the language model took in grayscale images of the hexadecimals. [22],[23] It extracted the grayscale representation and translated it back into its corresponding hexadecimal strings. To facilitate the generation of synthetic hexadecimals, a pre-trained language model (LLM) was employed. The choice of language model was instrumental in achieving coherent and contextually relevant synthetic data. In this regard, the GPT-3 language model, made accessible through the OpenAI API,

was selected for its demonstrated prowess in text generation tasks.[24] For the purpose of this study, the following specifications were configured:

- Model Name: "EleutherAI/gpt-neo-1.3B"[25]
- Tokenizer: The AutoTokenizer component from Hugging Face's Transformers library
- Model Architecture: The AutoModelForCausalLM module from Hugging Face's Transformers library, configured as a decoder

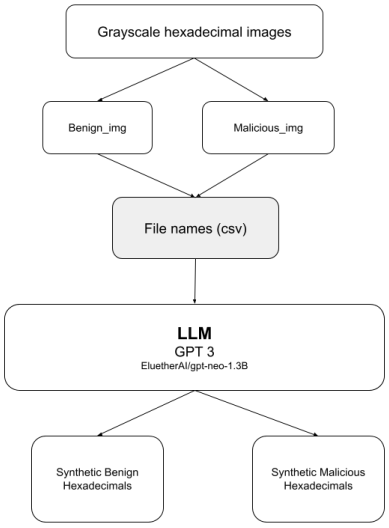

Fig. 3. The architecture of the GLEAM LLM

2) Hexadecimal Generation: The generation process revolved around presenting the language model with hexadecimal strings derived from grayscale image data, which were made from the pe files from the dataset. Prior to inputting these hexadecimal strings, it was essential to preprocess them to ensure compatibility with the model's tokenizer. Subsequently, the language model was invoked to generate sequences of hexadecimal characters, which, in turn, contributed to the generation of longer hexadecimal sequences.

E. Iterative Generation and Validity Filtering

The generation of synthetic hexadecimals was done iteratively to extend the length of each sequence. At each iteration, a portion of the text was presented to the language model, which generated additional segments of hexadecimal data in return. Post-generation filtering was carried out to maintain validity and coherence in the generated hexadecimals.

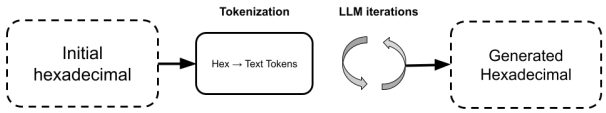

Fig. 4. Structure of tokenization

Extraneous characters filtered out as the filtering process left only hexadecimal values (0-9, A-F) to be retained. The file names generated by the CNN were used as indicators for which images should be looked at when generating synthetic samples. The file names produced by the CNN were where the classifiers failed, so to create evasive samples, the LLM was trained on these missed samples. These images were then looked at for their hexadecimals, and used to generate more.

F. Storing Organized Synthetic Data

The generated hexadecimals were saved as ".bytes" files, effectively simulating synthetic image data. A structured folder arrangement was implemented to ensure clarity in the differentiation of synthetic data based on their classification (benign or malicious). This also ensured an easier classification process when put through the black-box detectors such as logistic regression and random trees. For example, hexadecimals originating from the malware images folder were stored in their own synthetic malware folder. This systematic organization creates efficient tracking and management. In the end, there were 105 synthetic samples and 110 malware samples created.

IV. Results/Discussion

In this section, we present the results of our study on generating synthetic malware samples using two different methods: Generative Adversarial Networks (GANs) and Language Models (LLMs). The objective of this study was to evaluate the evasiveness of the synthetic malware samples generated by each method by assessing their detection rates across various classifiers commonly used in malware detection.

Two different methods were explored for generating synthetic malware samples: GANs, and LLMs. By employing these methods, we assessed how evasive the generated synthetic malware samples were with respect to each method through the use of classifiers. The experimental design consisted of passing the synthetic malware samples through a battery of classifiers and evaluating their classification accuracy. We then compared the performance of the GAN-generated and LLM-generated malware samples to determine which method yielded more evasive samples.

In our use case of malware detection, evasiveness refers to the ability of generated malware samples to evade detection by classifiers. A higher detection rate or accuracy indicates that the classifiers are able to correctly identify the synthetic malware samples, which implies lower evasiveness. Thus, a lower detection rate or accuracy suggests that the synthetic malware samples are successfully evading detection, resulting in higher evasiveness.

In our experiments, we use the True Positive Rate (TPR) to assess accuracy. The TPR is the proportion of malicious samples that were accurately classified as such.

A. GANs

The classifiers' performance on GAN-generated samples varied across different models. For instance, the Logistic Regression classifier achieved an accuracy of 57 percent on GAN-generated samples, while the Random Forest and Decision Trees classifiers achieved accuracies of 74 and 66

TABLE I
TPRS OF DIFFERENT CLASSIFIERS ON ORIGINAL DATA AND GLEAM

	Original (%)	**GAN (%)**	**LLM (%)**
LR	84	57	61
RF	92	74	63
DT	81	66	62
SVM	93	83	59
KNN	78	47	58

percent respectively. The SVM classifier achieved an accuracy of 83 percent, and the K-Nearest Neighbors (KNN) classifier exhibited an accuracy of 47 percent on GAN-generated samples. These results suggest that GAN-generated samples possess certain traits that challenge the classifiers' ability to distinguish them from genuine malware instances, making them potentially more evasive.

B. LLMs

When employed for generating synthetic malware samples, LLMs exhibited the ability to produce hex sequences that closely resemble the syntax and structure of real malware code.

Evaluation of the classifiers on LLM-generated samples resulted in varied classification accuracy. The Logistic Regression classifier achieved an accuracy of 61 percent, while the Random Forest and Decision Trees classifiers achieved accuracies of 63 and 62 percent respectively. Surprisingly, the SVM classifier exhibited a lower accuracy of 59 percent on LLM-generated samples, and the KNN classifier achieved an accuracy of 58 percent. The lower classification accuracy on LLM-generated samples indicates that these samples possess more evasive characteristics, as they challenge the classifiers' ability to correctly classify them.

C. Discussion

There are many advantages associated with each individual approach. Firstly, the GAN was able to excel at producing a variety of diverse samples mimicking the statistical properties of the original dataset, whereas the LLM focused on creating coherent and structured samples that modeled after real malware patterns. Additionally, the lower classification accuracy observed on LLM-generated samples suggests their potential for being more evasive, as they were able to closely replicate traits present in genuine malware. This is significant, as it implies for improving the robustness of malware detection systems by augmenting training datasets with LLM-generated samples.

V. CONCLUSIONS

Our model, GLEAM, is used for evasive, adversarial malware generation. We achieved this goal by increasing the average evasion rate by 20.2% with GANs and 25.0% with LLMs, demonstrating the possibilities of expanded feature sources and even LLMs for adversarial examples. Through this investigation, we reveal the vulnerabilities in machine-learning-based malware detection algorithms.

In future works, we encourage the development of novel malware detection approaches, such as safe dynamic analysis of malware PEs. Further research could focus on optimizing and fine-tuning the model's architecture to improve its capacity to replicate the nuances of real evasive malware. Finally, future research on the incorporation of additional sources and technologies beyond hex code and opcode features could help to broaden the field of adversarial malware development.

A. Acknowledgment

We would like to thank Michael Lester (michael.lester.main@gmail.com) of Practical Security Analytics LLC (practicalsecurityanalytics.com) for providing the dataset.

REFERENCES

[1] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review," Journal of Cybersecurity and Privacy, vol. 2, no. 3, pp. 527–555, Jul. 2022, doi: https://doi.org/10.3390/jcp2030027

[2] M. S. Akhtar and T. Feng, "Evaluation of Machine Learning Algorithms for Malware Detection," Sensors, vol. 23, no. 2, p. 946, Jan. 2023, doi: https://doi.org/10.3390/s23020946

[3] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016, doi: https://doi.org/10.1109/comst.2015.2494502

[4] D. S. Keyes, B. Li, G. Kaur, A. H. Lashkari, F. Gagnon, and F. Massicotte, "EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics," IEEE Xplore, May 01, 2021. https://ieeexplore.ieee.org/document/9452002

[5] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv (Cornell University), Jun. 2014, doi: https://doi.org/10.48550/arxiv.1406.2661.

[6] Zhong, F., Cheng, X., Yu, D., Gong, B., Song, S., and Yu, J, "MalFox: Camouflaged adversarial malware example generation based on Conv-Gans against black-box detectors." Jun. 06, 2022. Available: https://arxiv.org/pdf/2011.01509.pdf

[7] W. Hu and Y. Tan, "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN," arXiv.org, Apr. 2017. https://arxiv.org/pdf/1702.05983.pdf

[8] E. Zhu, J. Zhang, J. Yan, K. Chen, and C. Gao, "N-gram MalGAN: Evading machine learning detection via feature n-gram," Digital Communications and Networks, vol. 8, no. 4, pp. 485–491, Aug. 2022, doi: https://doi.org/10.1016/j.dcan.2021.11.007

[9] F. Lu, Z. Cai, Z. Lin, Y. Bao, and M. Tang, "Research on the Construction of Malware Variant Datasets and Their Detection Method," Applied Sciences, vol. 12, no. 15, p. 7546, Jan. 2022, doi: https://doi.org/10.3390/app12157546

[10] S. Jang, S. Li, and Y. Sung, "Generative Adversarial Network for Global Image-Based Local Image to Improve Malware Classification Using Convolutional Neural Network," Applied Sciences, vol. 10, no. 21, p. 7585, Oct. 2020, doi: https://doi.org/10.3390/app10217585

[11] G. Kaur, Habibi Lashkari, Arash, A. Rahali, L. Taheri, and F. Gagnon, "DIDroid: Android malware classification and characterization using deep image learning," Dec. 2020. doi: https://doi.org/10.1145/3442520.3442522

[12] L. Michael, "PE Malware Machine Learning Dataset," Practical Security Analytics LLC, Jun. 08, 2021. https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/

[13] S. Nari and A. A. Ghorbani, "Automated malware classification based on network behavior," IEEE Xplore, Jan. 01, 2013. https://ieeexplore.ieee.org/document/6504162

[14] S. Mahdavifar, A. F. Abdul Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning," IEEE Xplore, Aug. 01, 2020. https://ieeexplore.ieee.org/abstract/document/9251198

[15] D. Smith, S. Khorsandroo, and K. Roy, “Supervised and Unsupervised Learning Techniques Utilizing Malware Datasets,” easychair.org, Feb. 2023, Accessed: Aug. 10, 2023. [Online]. Available: https://easychair.org/publications/preprint/4bMc

[16] Kargaard, J., Drange, T., Kor, A.-L., Twafik, H., Butterfield, E.: Defending IT systems against intelligent malware. Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018, 411–417 (2018). doi:10.1109/dessert.2018.8409169

[17] E. Raff et al., “An investigation of byte n-gram features for malware classification,” Journal of Computer Virology and Hacking Techniques, vol. 14, no. 1, pp. 1–20, Sep. 2016, doi: https://doi.org/10.1007/s11416-016-0283-1.

[18] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, “Detecting unknown malicious code by applying classification techniques on OpCode patterns,” Security Informatics, vol. 1, no. 1, Feb. 2012, doi: https://doi.org/10.1186/2190-8532-1-1.

[19] R. Killam, P. Cook, and N. Stakhanova, “Android Malware Classification through Analysis of String Literals.” Available: https://www.ta-cos.org/sites/ta-cos.org/files/Strings.pdf

[20] A. Dunmore, J. Jang-Jaccard, F. Sabrina, and J. Kwak, “Generative Adversarial Networks for Malware Detection: a Survey,” Apr. 2023. Available: https://arxiv.org/pdf/2302.08558.pdf

[21] S. Bourou, A. El Saer, T.-H. Velivassaki, A. Voulkidis, and T. Zahariadis, “A Review of Tabular Data Synthesis Using GANs on an IDS Dataset,” Information, vol. 12, no. 9, p. 375, Sep. 2021, doi: https://doi.org/10.3390/info12090375.

[22] H. Hadian Jazi and A. A. Ghorbani, “Dynamic graph-based malware classifier,” IEEE Xplore, Dec. 01, 2016. https://ieeexplore.ieee.org/document/7906945/

[23] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, “An Efficient DenseNet-Based Deep Learning Model for Malware Detection,” Entropy, vol. 23, no. 3, p. 344, Mar. 2021, doi: https://doi.org/10.3390/e23030344

[24] M. Labonne, “Fine-Tune Your Own Llama 2 Model in a Colab Notebook,” Medium, Jul. 31, 2023. https://towardsdatascience.com/fine-tune-your-own-llama-2-model-in-a-colab-notebook-df9823a04a32

[25] S. Black, G. Leo, P. Wang, C. Leahy, and S. Biderman, “GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow,” Zenodo, Mar. 21, 2021. https://zenodo.org/record/5297715 (accessed Aug. 13, 2023).