CSE 509: System Security Fall 2023

Polymorphic Malware Detection and Evasion

Yash Kothadiya (115734533) Dhruv Charan (115689145) Pranav Dani (115812293) Shreyas Habade (115911132)

State University of New york (SUNY) at Stony Brook

Abstract - This report presents a comparative study of different Generative Adversarial Network (GAN) models for generating adversarial network attack data and evading conventional detection systems. We investigate various GAN variants' effectiveness, advantages, and limitations, such as cGAN (conditional), wGAN (Wasserstein), and Basic GAN. We also evaluate the performance of these models on different types of Intrusion Detection Systems (IDSs), such as signature-based, anomaly-based, and hybrid-based. Our results show that some GAN models can generate realistic and diverse adversarial network traffic, while others suffer from mode collapse, instability, or high data dependency. We also discuss the implications and challenges of using GANs for polymorphic adversarial cyberattacks and suggest possible future research directions.

I. INTRODUCTION

Polymorphic malware refers to a type of malicious software that constantly changes its code and appearance while retaining its core functionality. This makes it challenging for traditional signature-based antivirus and intrusion detection systems (IDSs) to detect and mitigate. The issue is that the training data is increasingly time-consuming to capture and then subsequently curate and as a result, the attack vectors are always going to be way ahead of the training data we would have to train these models since in this case of attacks carried out using Polymorphic malware the signatures are constantly evolving at a rate faster than we can reasonably capture. But what if we could turn this imbalance on its head and train our IDS on attacks that have never been seen before, or concocted, what if we could have a component as part of our IDS system that would scan these incoming attacks and gather a sense of them well enough to the point that it could with reasonable confidence recreate with some changes and perhaps concoct fresh attacks. This is where the idea of generative AI comes in. In recent years, there has been a growing interest in using Generative Adversarial Networks (GANs) to generate polymorphic malware and network traffic, as well as to deceive and evade detection systems. GANs are a class of neural networks that consist of two competing components: a generator and a discriminator. The generator tries to produce fake data that resembles the real data, while the discriminator tries to distinguish between the real and fake data. The two components are trained in an adversarial manner until the generator can fool the discriminator. GANs have shown remarkable results in generating realistic and diverse images, texts, audio, and videos. However, there are also various challenges and limitations associated with GANs, such as mode collapse, instability, data dependency, and evaluation metrics.

In this report, we aim to answer the following questions:

RQ1: How effective are different GAN models when used as IDSs instead of conventional detection systems?
RQ2: What are the advantages and limitations of different GAN models for generating adversarial network attack data and using it as an IDS?
RQ3: What are the challenges and opportunities of using GANs for polymorphic adversarial cyberattacks?

To answer these questions, we conduct a comprehensive literature review of the existing

works on using GANs for polymorphic malware and network traffic generation and evasion. We also implement and compare some GAN variants, such as cGAN (conditional), wGAN (Wasserstein), and Basic GAN on our discriminator-based IDS. We evaluate the performance of these models using various metrics, such as accuracy, precision, recall, and F1-score. We also analyze the strengths and weaknesses of these models and discuss the implications and challenges of using GANs for polymorphic adversarial cyberattacks.

II. LITERATURE REVIEW

The pervasive use of the internet has led to an alarming surge in web-based attacks, underscoring the critical need for robust defenses against sophisticated malware. The classification of malware into generational categories has aided in understanding its evolution, particularly the second generation marked by internal structural variations: Encrypted, Oligomorphic, Polymorphic, and Metamorphic strains [1].

Among these, Polymorphic Malware stands out for its adaptive nature, capable of altering its appearance and evading traditional signature-based detection systems. Recent research has delved into innovative methodologies enabling the creation and evasion of such malware, challenging the efficacy of existing intrusion detection systems (IDSs) [2].

Chauhan et al. introduced an emergent concept employing Adversarial Machine Learning techniques against operational IDSs, generating functionally identical attacks with altered signatures. This approach, while momentarily exploiting vulnerabilities, underscores the evolving sophistication of attackers aiming to bypass intrusion filters [2].

Moreover, advancements in leveraging deep learning models, such as Wasserstein Generative Adversarial Networks (WGAN), have demonstrated exceptional efficacy in evading detection systems. Zhan, Yan, and Wang illustrated a novel evasion attack strategy specifically targeting malicious PDF file detection systems. Their approach achieved an unprecedented 100% evasion rate, highlighting the potency of generative AI in creating evasive malware variants [3].

Concurrently, the realm of polymorphic malware detection has seen a multifaceted evolution. Conventional signature-based methods prove inadequate due to the dynamic nature of this malware. Various strategies have emerged, encompassing data mining through system calls, sandbox analysis, string searching algorithms, and hybrid clustering algorithms [4],[5]. Alazab et al. proposed statistical measures through system call analysis, while Gavriluţ et al. and Cesare and Xiang developed hybrid-clustering frameworks and dynamic emulation methods, respectively, to classify and detect polymorphic malware [4],[7].

Additionally, the integration of deep learning techniques has exhibited promising outcomes in malware detection. Vinay Kumar et al. and Masabo introduced frameworks utilizing image-based deep learning and feature engineering algorithms, showcasing enhanced classification abilities [9],[11].

Furthermore, innovations like DeepLocker and network traffic analysis-based techniques have contributed to identifying polymorphic malware based on behavior and encrypted communication patterns [6],[8]. Kim, Bu, and Cho pioneered the use of transferred deep-convolutional generative adversarial networks (tDCGAN) for zero-day malware detection, displaying robustness against previously unseen attacks [10].

To sum up, the field of polymorphic malware is constantly changing, making detection and prevention extremely difficult. Combating these elusive threats has shown promise when advanced AI tools and multidisciplinary methodologies are included. However, to protect against the constantly changing strategies of polymorphic malware, attackers and defenders must constantly innovate and change their defenses.

III. GAN VARIANTS

In this section, we provide an overview of *Basic* GAN (*refer to Fig. 1*) architecture which is augmented further when we discuss other GANs. Generative Adversarial Network is a paradigm based on machine learning models that can generate synthetic data from the original input data. It consists of two neural networks known as the Generator and the Discriminator.

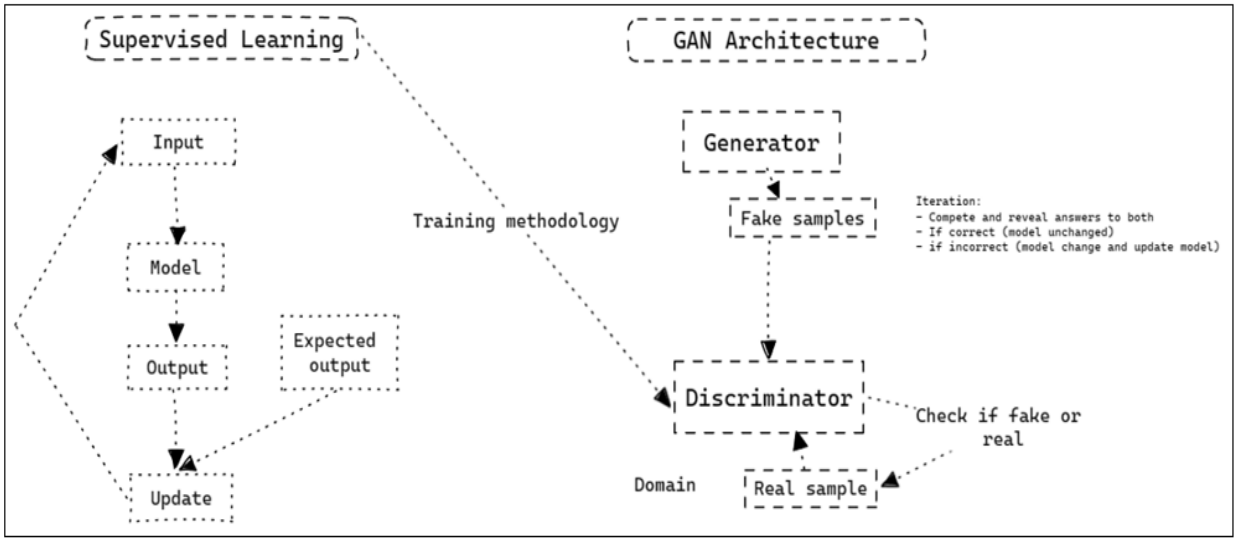


Figure 1: GAN Architecture

The discriminator essentially acts as a sort of classifier, distinguishing between genuine and fake data. It assesses two categories of information: the original dataset and the data produced by the generator. In the training process, the discriminator treats the original data as positive examples and the generated data as negative or adversarial instances. The term LD stands for the penalty assigned to the discriminator. This penalty fluctuates; it increases when the discriminator fails to identify or accurately differentiate the data and decreases otherwise. Backpropagation is the mechanism employed to adjust the discriminator's weights.

Additionally, there's another term, LG, indicating the loss incurred by the generator during the training phase.

Methodology

The generator produces a synthetic dataset by receiving feedback from the discriminator and learns to produce data so that the discriminator classifies the synthetic data as the original.

The training of the Generator includes the following steps:

• A random input noise.

• The generator, to produce synthetic data from the random data.
• The discriminator, to distinguish the synthetic data and original data.
• Loss LG that fines the generator if it is unable to produce data, which can deceive the discriminator.

It needs to train the generator and the discriminator alternately. Moreover, it is important to check if GAN is convergent or not. The training works as follows:
• step -1 - Training of the generator runs for some epochs.
• step -2 - Training of the discriminator runs for some epochs.
• Continue repeating steps 1 and 2 until the GAN converges.

Generally speaking [17, 18, 19] most of the common challenges in training a GAN particularly when it comes to generating adversarial examples within the context of training a discriminator are as follows:
- Convergence problem: Classification performance of the discriminator decreases in the following cycles. Training the GAN from this point indicates that the generator trains with less meaningful data. This state is known as the convergence problem.
- Mode collapse: A GAN can produce a good variety of data in ideal conditions. However, suppose a generator learns to produce a specific set of data so that the discriminator classifies them as the original. In that case, the generator will only produce these sets of data and easily deceive the discriminator. This condition is called mode collapse and oftentimes even though the mode collapse is not full-fledged the generator can still often bias to specific classes in the generated data.

Wasserstein GAN

Wasserstein GAN (WGAN) offers an improved approximation of distributed data present in the training set. WGAN employs a discriminator equipped with a critic, which furnishes a score indicating the authenticity of generated data. In comparison, the discriminator in traditional GANs simply predicts and categorizes the generated data as either original or fake, resulting in a binary classification.

Conditional GAN

Conditional Generative Adversarial Networks (cGANs) differ from Wasserstein Generative Adversarial Networks (WGANs) in that cGANs introduce conditional information during the generation process. The generator in a cGAN not only aims to produce realistic data but also incorporates a conditioning factor, such as a label or auxiliary data, to guide the generation. The loss function of cGANs includes both the standard GAN loss for realism and a conditional term enforcing alignment with the given condition. While cGANs address the task of generating data based on specific conditions, WGANs, on the other hand, focus on improving training stability and mitigating issues like mode collapse by utilizing the Wasserstein distance in the loss function. The choice between the two depends on the specific generative task and requirements.

In our research endeavors, we commenced our exploration using basic GANs as a foundational baseline. We integrated Wasserstein GAN (WGAN) into our framework to augment our approach and elevate accuracy. Moreover, leveraging the advantages of conditional information, particularly when working with labeled data, we further refined our models by implementing conditional GANs (cGANs). This progression through GAN variants allowed us to refine our generative models, harnessing their distinct capabilities to elevate accuracy, generate

diverse outputs, and incorporate specific control over the generated data.

IV. DATASETS AND IDS

In response to the evolving landscape of detection systems, the Canadian Institute for Cyber Security has contributed significantly by developing the CICIDS 2017 dataset. This dataset, specifically curated for Intrusion Detection Evaluation, encompasses both benign and a variety of well-known attacks, including Brute Force, SSH, DoS, Web Attacks, Botnets, and DDoS.

The dataset's construction adheres to critical criteria, including complete network configurations, comprehensive traffic data, labeled datasets, thorough interaction scenarios, exhaustive capture data, adherence to available protocols, a diverse range of attacks, a robust feature set, and detailed metadata. Notably, the CICIDS2017 dataset surpasses its predecessors by incorporating over 80 features aligned with contemporary network standards, many of which were previously unavailable.

The CICIDS2017 data consists of eight different files that contain regular traffic and attack traffic data. Table 1 provides a succinct overview of the key properties characterizing the CICIDS2017 dataset, making it an essential tool for researchers and practitioners in intrusion detection.

We used the CICIDS2017 dataset, specifically focusing on weekly attack data. Our data vectors comprised packet details, port information, network activity, and flag indicators from the dataset. Notably, the dataset exhibited a significant bias towards regular, non-benign network flows. To address this imbalance, we implemented SMOTE undersampling and other techniques to enhance the data's suitability for integration with GAN models.

Our goal was to enable these GAN models to consistently learn attack patterns and generate realistic representations of the corresponding data streams. The dataset encompassed various attack types, including 'Infiltration,' 'Bot,' 'PortScan,' 'DDoS,' 'FTP-Patator,' 'SSH-Patator,' 'DoS slow loris,' 'DoS Slowhttptest,' 'DoS Hulk,' 'DoS GoldenEye,' 'Heartbleed,' 'Web Attack Brute Force,' 'Web Attack XSS,' and 'Web Attack SQL Injection.'

Feature	Value
Total number of flows	2,830,540
Total number of features	83
Number of labels	15

Table 1 - *Properties of dataset*

V. METHODOLOGY AND SETUP

The following are the libraries used in the overall work of this research:
For our training and inference processes, we employed Tensorflow and its Tensorflow. Keras wrapper for model development. Additionally, we utilized the numpy, sci-kit-learn, and Pandas libraries to handle data manipulation and implement undersampling techniques.

Model Architecture

The Generator model is a sequential neural network that starts with a 256-unit dense layer, taking a 100-element input vector. This layer is followed by batch normalization and a LeakyReLU activation. The next layer is another dense layer with 512 units, also followed by batch normalization and LeakyReLU. The final

output layer is a dense layer with an output size of 77 units corresponding to our feature set and uses a 'tanh' activation function.

The Discriminator model, also a sequential network, begins with a 512-unit dense layer designed to process inputs of size defined by the number of features which in our case is 77.This is followed by LeakyReLU activation and a 0.3 dropout layer. A second 256-unit dense layer continues the architecture, again followed by LeakyReLU and another dropout layer. The final layer is a single-unit dense layer, used for binary classification to determine if the input data is real or generated.

In essence, the generator model creates data from noise, and the discriminator model classifies data as real or synthetic.
We trained this model on the reduced CICIDS2017 dataset which after undersampling and dataset pruning came out to around 2425727 input samples with an 80:20 mix of BENIGN and NON-BENIGN labels which happen to contain all of our aforementioned attack types. We then generated a mix of samples from our input set as well as certain samples generated entirely from noise through our generator model and made the Discriminator identify and classify the results as one of the possible classes of attacks to be able to give our discriminator more training data than could be acquired directly from the dataset thereby strengthening its ability to detect unforeseen attack modules. We then moved on to our other GAN vectors namely the Wasserstein GAN and the Conditional GAN. For the conditional GAN the architecture would stay largely the same with the only change being that the generator would now be able to take into account the identity of the label it was learning from and also generate vectors pertaining to the same labels, this is very useful since our dataset even with all the sampling implementations is still heavily biased towards Benign network flows and so the generators and the discriminators both do not have enough incentive to learn or generate results from the other classes which a) leads to mode collapse on some scale and b) leads to both the models not generalizing enough to the types of attacks that we need them to be generalized towards,with a conditional GAN we can force the generator to specifically train for and generate results by feeding it the input labels and asking it to generate results from the same mold,which in turn gives ur discriminator a better sampling of attacks of those kind in our artificially generated input dataset which ultimately leads to better performance from the generator.

As for the Wasserstein GAN, it brings about a significant alteration in the loss functions. Departing from conventional adversarial losses, the Wasserstein distance metric is employed to create a smoother convergence landscape during training. The discriminator's aim shifts towards maximizing the disparity between the mean scores of real and generated data, ensuring a more perceptive assessment of data authenticity.

Furthermore, a pivotal aspect of our implementation of the Wasserstein GAN involves the incorporation of a gradient penalty. This regularization mechanism is introduced to address potential drawbacks in training stability. The gradient penalty acts as a deterrent against an excessive focus on specific regions of the data distribution, fostering a more consistent learning process. By penalizing the norm of the discriminator's gradients concerning interpolated data, we mitigate the risk of the discriminator becoming overly responsive to particular data features.

The integration of the gradient penalty into the training step augments the discriminator's capacity to generalize across diverse instances. This regularization not only cultivates stable

learning dynamics but also plays a role in averting mode collapse, ensuring that the generator explores a more comprehensive range within the data distribution.

However to understand the behavior of the GANs is super necessary since we have no way of really peeking into the inner machinations and figuring out whether it is effectively learning the patterns behind these attacks or whether it has merely learned to mimic them in a very vague and uneventful way which is the primary challenge when working with generative AI, that is to sniff out the chaff from the wheat and detect inconsequential learnings.We have in our attempts to evaluate the behavior of our systems and compare them against the decision tree implementation decided to use certain metrics as outlined below as they gave us the best sense of where the models stand concerning each other

The following metrics are employed to evaluate the performance of IDS in this work:

• Detection Rate (DR) or Precision—a ratio between the correctly detected attack samples over all the samples classified as an attack by the IDS. It is also known as a ratio between True Positives and the sum of True Positives and False Positives.

$$DR = TP / (TP + FP)$$

• Recall—a ratio between the correctly detected attack samples over the total attack sample data. It is also known as a ratio between True Positives and the sum of True Positives and False Negatives.

$$Recall = TP / (TP + FN)$$

• F1_Score—represents the harmonic mean of precision and recall.

$$F1\_Score = 2 \times (precision \times recall) / (precision + recall)$$

• True Negative Rate (TNR)—a ratio between the correctly detected benign samples over the total benign sample data. It is also known as a ratio between True Negatives and the sum of True Negatives and False Positives.

$$TNR = TN / (TN + FP)$$

VI. RESULTS AND DISCUSSION

This section describes the results of various experiments for different scenarios centered around the metrics outlined previously and the analyses of the findings.

For our baseline comparison, we chose the random forest classifier-based model implementation on our CICIDS dataset as its results were readily available. Unlike the Wasserstein GAN's reliance on novel loss functions, the Random Forest Classifier operates within the conventional supervised learning framework, offering a different perspective on model refinement. The Random Forest Classifier employs an ensemble of decision trees to collectively make predictions. This ensemble approach introduces a level of robustness, enabling the model to discern patterns and relationships within the data.

The below-given results and comparisons of our Implementations are juxtaposed with our baseline model mentioned above.

Decision Tree			
Accuracy	Precision	Recall	F-1 Score
0.954	0.946	0.951	0.954

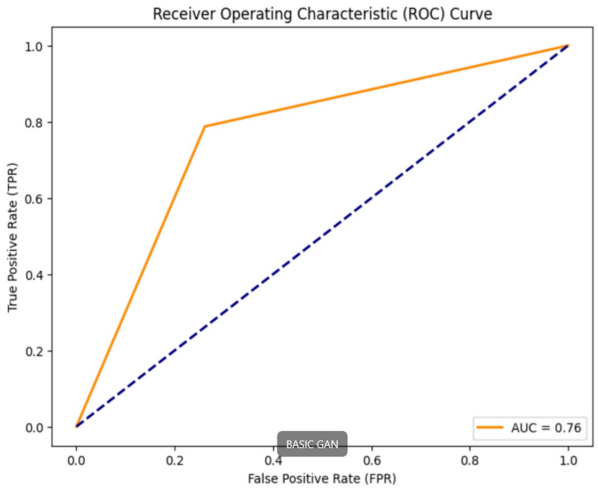
In the evaluation of the Random Forest Classifier using the provided confusion matrix, an impressive accuracy level of approximately 95% is observed. This notable figure, however, raises concerns regarding potential overfitting owing to an imbalanced distribution within the

dataset, particularly in the representation of benign and non-benign instances.

The disproportionately skewed distribution of benign and non-benign data points appears to have influenced the model's learning process, potentially leading to overemphasis or bias towards the more prevalent class. Consequently, this could have resulted in an inflated accuracy metric, indicating a robust performance that might not generalize well to new, unseen data.

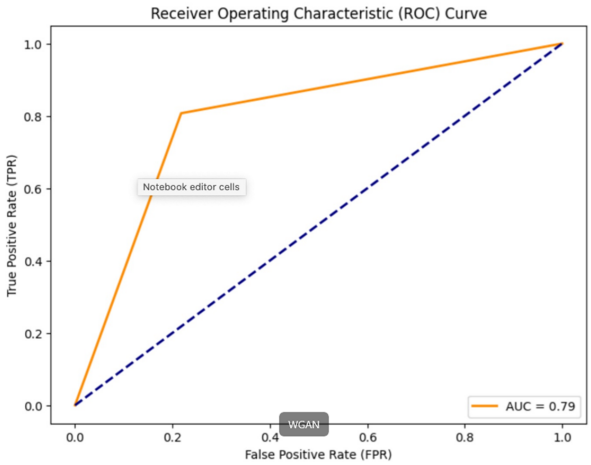
A. Basic GAN (Baseline Model)

Confusion Matrix				
	Precision	Recall	F-1 Score	
0	0.78	0.74	0.76	390144
1	0.75	0.79	0.77	390144
Classification Report				
Accuracy			0.76	780288
Macro avg	0.76	0.76	0.76	780288
Weighted avg	0.76	0.76	0.76	780288



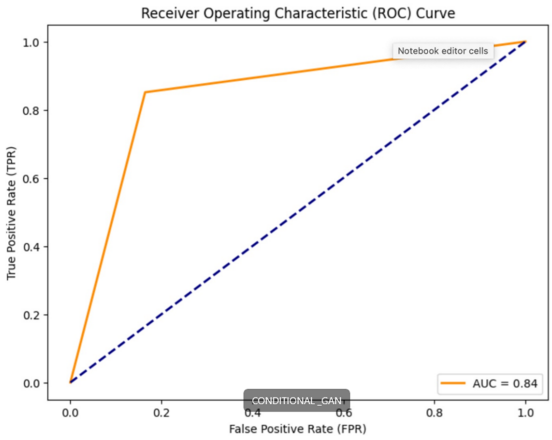
B. Wasserstein GAN

Confusion Matrix				
	Precision	Recall	F-1 Score	
0	0.80	0.78	0.79	390144
1	0.79	0.81	0.80	390144
Classification Report				
Accuracy			0.79	780288
Macro avg	0.79	0.79	0.79	780288
Weighted avg	0.79	0.79	0.79	780288



C. Conditional GAN

Confusion Matrix				
	Precision	Recall	F-1 Score	
0	0.85	0.84	0.84	390144
1	0.84	0.85	0.84	390144
Classification Report				
Accuracy			0.84	780288
Macro avg	0.84	0.84	0.84	780288
Weighted avg	0.84	0.84	0.84	780288



Our study underscores the potential of Generative Adversarial Networks (GANs) in enhancing the performance of Intrusion Detection Systems (IDS). We began with a standard GAN and progressively tested more suitable architectures, which led us to identify Conditional GANs (CGANs) as the most promising model for IDS training, identification, and classification of future attacks.

The CGAN's ability to switch between various modes, as dictated by the subset of the dataset that we used, provides the IDS with a more representative training dataset for each class we aim to recognize. This adaptability is a significant advantage of CGAN.

While the Wasserstein GAN (WGAN) showed slightly inferior analytical performance, it still demonstrated an improvement over the basic GAN. This is primarily because WGANs can better avoid mode collapse due to the implementation of the Wasserstein distance in the loss function.

These findings not only demonstrate the effectiveness of these specific GAN architectures but also contribute to the existing body of work on the efficacy of various GANs in tackling the issue of polymorphic malware detection.

However, it's important to note that these results should be interpreted with caution. The high computational cost of training GANs meant that we could only train on a subset of the actual 15.6 GB dataset. Although this subset was representative and fairly sampled, it did not provide the models with a comprehensive view of the various attack types.

Furthermore, our models were trained to classify attacks into various classes, rather than simply distinguishing between benign and non-benign categories, which is the ultimate goal of an IDS. It's possible that training our models for this binary classification task and evaluating them accordingly could yield different results. It might also reveal that certain architectures are better suited to identifying irregular network flow rather than strictly classifying the types of attacks. Therefore, further research is warranted in this area.

VII. CONCLUSION AND FUTURE WORK

In conclusion, our experiments have demonstrated that CGANs are potentially well-suited for training Intrusion Detection Systems (IDSs) to identify harmful attack patterns within a system and generate new data with sufficient confidence to keep the IDSs alert. However, due to computational limitations, we were unable to further extend this experiment to tune the GAN architectures to find the optimal fit for the given problem statement, which could theoretically yield slightly different results.

Moreover, we were unable to train our models on the full-scale dataset, which is considerably more comprehensive and diverse. There is also the potential for incorporating transfer learning, where we could use our existing models that

have learned to classify attacks and train them to simply recognize attacks rather than classify them. We attempted to implement this but were unsuccessful in resolving the issue due to problems with training convergence, which appears to be related to how the baseline models learn the ground truths and how the loss is calculated. Further work is needed to determine if different loss definitions could enhance the systems' performance.

In addition to this, the non-determinism introduced by this implementation also presents the challenge of completeness versus robustness as it relates to false positives and the tolerance of our systems. IDSs are one of the first licenses against rogue intrusions, but a high rate of flagging harmless behavior could lead to significant disruptions. Our paper does not delve deeply into the trade-offs between accuracy and false positive rate and how it can be optimized concerning the architecture and hyperparameter selection.

VIII. ACKNOWLEDGMENTS

We express our sincere gratitude to Professor Amir Rahmati for his invaluable guidance, unwavering support, and insightful feedback throughout this research endeavor. Their expertise and dedication have been instrumental in shaping the direction and quality of this work.
Furthermore, we would like to acknowledge the contributions and support received from our fellow coursemates during the duration of this study.
Finally, we would like to extend our appreciation to all individuals who, directly or indirectly, contributed to this research effort. Your encouragement, feedback, and support have been invaluable in shaping this work.

IX. REFERENCES

1. Sharma, Ashu. 2012. "Evolution and Detection of Polymorphic and Metamorphic Malware: A Survey." https://arxiv.org/pdf/1406.7061.pdf

2. Chauhan, Sabeel, Polymorphic Adversarial Cyberattacks Using WGAN *J. Cybersecur. Priv.* **2021**,*1*(4),767-792; **https://doi.org/10.3390/jcp1040037**

4. D. Gavriluţ, M. Cimpoeşu, D. Anton and L. Ciortuz. Malware detection using machine learning, International Multiconference on Computer Science and Information Technology, Mragowo, 2009, pp 735- 741.

5. P. Sharma, S. Kaur, and J. Arora. An Advanced Approach to Polymorphic/Metamorphic Malware Detection using Hybrid Clustering Approach (IRJET) 2016.

6. R. Kaur and M. Singh, A Survey on Zero-Day Polymorphic Worm Detection Techniques. IEEE Communications Surveys & Tutorials vol. 16, no. 3, pp. 1520-1549, Third Quarter 2014.

7. S. Cesare, Y. Xiang and W. Zhou, Malwise— An Effective and Efficient Classification System for Packed and Polymorphic Malware. IEEE Transactions on Computers, vol. 62, no. 6, pp. 1193-1206, June 2013.

8. M. A. Atici, S. Sagiroglu and I. A. Dogru. Android malware analysis approach based on control flow graphs and machine learning algorithms. 4th International Symposium on Digital Forensic and Security (ISDFS), 2016, Little Rock, AR, 2016, pp. 26-31.

9. Wu, Songyang, Pan Wang, Xun Li, and Yong Zhang. Effective detection of Android malware

based on the usage of data flow APIs and machine learning. Information and Software Technology 75 2016, pp 17-25.

10. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P. and Venkatraman, S., 2019. Robust Intelligent Malware Detection Using Deep Learning. IEEE Access, 7, pp.46717-46738.

11. Masabo, E., Kaawaase, K.S., Sansa-Otim, J. and Hanyurwimfura, D., 2017, November. Structural Feature Engineering approach for detecting polymorphic malware. In 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC /PiCom/DataCom/CyberSciTech) (pp. 716-721). IEEE

12. Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. (2018, May 18). Zero-day Malware Detection Using Transferred Generative Adversarial Networks Based on Deep Autoencoders - ScienceDirect. https://doi.org/10.1016/j.ins.2018.04.092

13. Evasion Attacks Based on Wasserstein Generative Adversarial Network. (n.d.). Evasion Attacks Based on Wasserstein Generative Adversarial Network | IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9018647

14. *Malware Detection by Analysing Network Traffic with Neural Networks*. (n.d.). Malware Detection by Analysing Network Traffic With Neural Networks | IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8227308

15. *DeepLocker: When malware turns artificial intelligence into a weapon*. (n.d.). ZDNET. https://www.zdnet.com/article/deeplocker-when-malware-turns-artificial-intelligence-into-a-weapon/

16. Nguyen, Vinh. (2018). A study of polymorphic virus detection. 10.13140/RG.2.2.19853.79842.

17. Shahpasand, M.; Hamey, L.; Vatsalan, D.; Xue, M. Adversarial Attacks on Mobile Malware Detection. In Proceedings of the 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile), Hangzhou, China, 24–24 February 2019; pp. 17–20.

18. Hui, J. GAN—DCGAN (Deep Convolutional Generative Adversarial Networks). Available online: https://jonathan-hui.medium.com/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f

19. Zhang, Z.; Li, M.; Yu, J. On the convergence and mode collapse of GAN. In Proceedings of the SIGGRAPH Asia 2018 Technical Briefs, Tokyo, Japan, 4–7 December 2018; p. 21.

20. CICIDS2017-Intrusion Detection Evaluation Dataset https://www.kaggle.com/datasets/cicdataset/cicids2017