Synthetic Intrusion Alert Generation through Generative Adversarial Networks

Christopher Sweet, Stephen Moskal, Shanchieh Jay Yang
Department of Computer Engineering
Rochester Institute of Technology
Rochester, New York 14623
Email: Jay.Yang@rit.edu

***Abstract*—Cyber Intrusion alerts are commonly collected by corporations to analyze network traffic and glean information about attacks perpetrated against the network. However, datasets of true malignant alerts are rare and generally only show one potential attack scenario out of many possible ones. Furthermore, it is difficult to expand the analysis of these alerts through artificial means due to the complexity of feature dependencies within an alert and lack of rare yet critical samples. This work proposes the use of a Mutual Information constrained Generative Adversarial Network as a means to synthesize new alerts from historical data. Histogram Intersection and Conditional Entropy are used to show the performance of this model as well as it's ability to learn intricate feature dependencies. The proposed models are able to capture a much wider domain of alert feature values than standard Generative Adversarial Networks. Finally, we show that when looking at alerts from the perspective of attack stages, the proposed models are able to capture critical attacker behavior providing direct semantic meaning to generated samples.**

***Keywords*—Cyber Intrusion Alerts, GANs, Attack Stages**

I. Introduction

The use of Network Intrusion Detection Systems (NIDS) has become widely adopted by private corporations, government agencies, and research laboratories to monitor and analyze traffic flowing through their respective networks. The data collected from these systems allows system administrators and security professionals to identify typical versus anomalous traffic [1], [2], potential vulnerabilities in the network [3], and profile the behavior of adversaries [4]. Despite the depth of data provided through alert analysis, there is a low signal to noise ratio with respect to malignant traffic. Further, the low amount of malignant alerts may not provide analysts with the full picture of possible vulnerabilities or attack patterns.

This work seeks to solve this problem by introducing a framework for synthesizing new alerts from historical malignant data. Through the use of Generative Adversarial Networks (GANs) [5], alerts are synthesized with a high degree of fidelity. The models used also employ a neural Mutual Information constraint [6], forcing better coverage of feature value domains and critical feature interactions.

Generative Adversarial Networks were first proposed by Goodfellow *et al.* [5] and have since been improved through the minimization of the Wasserstein Distance with adaptive gradient penalty [7], [8]. Since, they have been expanded upon and used for data generation with respect to images [9], text [10], and sound [11]. Additionally, GANs have been applied to network traffic to modify and obfuscate malicious traffic [12], [13]. The traffic modified by these networks has been shown to successfully avoid detection by NIDS.

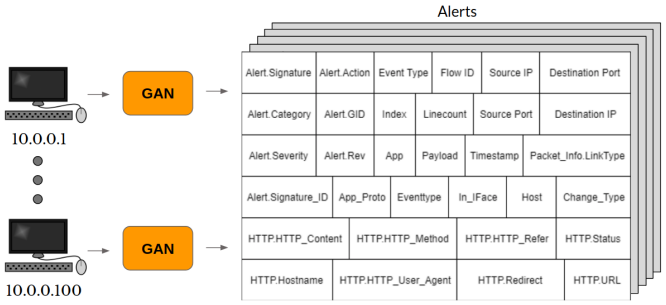

Fig. 1: Individual GAN models may be used to synthesize alerts based off Target IP and historical data.

Despite these successes, GANs suffer from several pathological issues when generating data. Namely, mode dropping is common for samples which occur with low probability in the training dataset. Failing to cover the entirety of the output domain is especially problematic in the field of cybersecurity as samples that occur with low probability may still contain significant data, such as critical vulnerabilities. By applying the work of Belghazi *et al.* [6], a Mutual Information constraint may be imposed on the GAN to encourage outputting all modes within the feature domain of the training dataset. Additionally, this constraint is shown to improve the model's ability to learn interactions between the features within a given alert compared to traditional GANs.

Using the Mutual Information Constrained GAN, alerts may be generated with a high degree of realism. Additionally, by organizing alerts on a per Target IP instance, individual models may learn to synthesize alerts for each possible *target* in a network as shown in Fig. 1. This allows a collection of models to be structured with inherent labels and provides data which may be analyzed directly for possible vulnerabilities on a target by target basis.

The remainder of the paper is structured as follows: Section II provides an overview of some of the existing challenges in Machine Learning for Cyber Security as well as existing applications of GANs for Cyber Security data. Section III and Section IV discusses the GAN model as well as preprocessing and analysis methods employed for CPTC'17 data respectively. Section V reviews the results of the trained models. Finally, Section VI gives the concluding remarks and future works of this research.

This research is supported by NSF SaTC Award #1526383.

II. Related Work

With the regularity and complexity of cyber attacks increasing, so has the interest in applying Machine Learning techniques to classify and predict attack actions. Several ongoing works in this field cite the need for more data as a limitation to their current research [14], [15], [16]. Specifically, Amit *et al.* [17] raise concerns about modern, openly available datasets, and the certainty of labels provided by the datasets which do exist.

Shen *et al.* [15] directly question the generalizability of their models despite having access to a dataset with over 3 billion individual alerts; unseen behaviors and alerts completely confound data driven models. Perry *et al.* [14] show a significant decrease in attack prediction and classification when models are trained with insufficient data.

Despite the challenge of limited datasets, multiple works have made use of alerts for anomaly detection and attack prediction. For example, Veeramachaneni *et al.* [1] used LSTMs to identify anomalous behavior given historical data. The anomalous behavior was then analyzed by experts to identify whether the anomaly was truly malicious, and the type of attack action taken in the alert. Shen *et al.* [15] also use LSTM, however they train their models to predict future attack actions given a sequence of prior alerts. Their models show promising ability to learn complex chains of alerts when trained with sufficient data.

Another avenue for research applying Machine Learning to cyber-security data has been the generation of adversarial traffic. Specifically, GANs have been used to obfuscate malicious traffic through the modification of packet behavior.

Rigaki *et al.* [12] proposed the use of GANs in generating network traffic which mimics other types of network traffic. In particular, real malware traffic was modified by a GAN to appear as legitimate network traffic. This allowed the malware to avoid detection from the Stratosphere Behavioral Intrusion Prevention System through the modification of three network traffic parameters; total byte size, duration of network flow, and time delta between current network flow and the last network flow. They showed that through the modification of these parameters detection rate could be dropped to 0%.

Similarly, Lin *et al.* [13] apply GANs to obfuscate traffic with the intention of directly deceiving a NIDS. Available attack actions include denial of service and privilege escalation. Their model is shown to drastically increase the evasion rate of malicious network traffic across several classifiers when benchmarked using the NSL-KDD benchmark provided in [18].

Despite the successes of these works, no current GAN model has been applied to recreation or expansion of Cyber Attack alert data. This research aims at generating malicious NIDS samples from the target perspective to expand limited datasets and enable further experiments using Machine Learning algorithms trained on intrusion alerts. Similar to the work provided by Veeramachaneni *et al.* our results could also be handed off to expert analysts to identify the attacker behavior or vulnerabilities within the network.

III. GAN Model

A Generative Adversarial Network is a class of neural network where two networks are pitted against each other. One network, the generator (G), attempts to create samples which emulate a dataset. The other network, the discriminator (D), takes inputs from the ground truth dataset as well as G, and flags samples as either real or fake. This structure minimizes the generator loss each time G successfully generates a sample that tricks D into marking the sample as real. Conversely, the discriminator loss is minimized when all samples from the ground truth set are marked as real and all samples created by G are marked as fake.

The Wasserstein GAN, proposed by Arjovsky *et al.* [7] extends the concept of a GAN but with increased stability during training. This was subsequently improved by Gulrajani *et al.* [8] by adding a gradient penalty term (WGAN-GP) to regularize the gradients of D, resulting in the loss function given by (1) .

Despite these improvements to the loss function for the discriminator, the generator loss was left mostly unmodified. Belghazi *et al.* [6] changed this by adding a mutual information term to the generator's loss. This contribution maximized the mutual information between the generator's noise input and it's output samples by minimizing the Donsker-Varadhan (DV) representation of the Kullback-Leibler (KL) divergence. This modification is shown in (2). The DV-KL term was added by using a neural network to learn how to estimate Mutual Information between two distributions. The rationale behind this added constraint was that it would force the generator to further explore the domain of the data when generating new samples; not exploring the dataset would result in a limit to the amount of mutual information which could be found between input noise and the output samples. Herein this model will be referred to as the WGAN-GPMI model.

$$D_{Loss} = \underbrace{\mathbb{E}[D(\mathbb{P}_r)] - \mathbb{E}[D(\mathbb{P}_g)]}_{\text{Wasserstein Distance}} + \underbrace{\lambda \mathbb{E}[(||\nabla_{\hat{x}} D(\mathbb{P}_z)||_2 - 1)^2]}_{\text{Gradient Penalty}} \tag{1}$$

$$G_{Loss} = \underbrace{-\mathbb{E}[D(\mathbb{P}_g)]}_{\text{Adversarial Loss}} + \underbrace{\mathbb{E}[\mathbb{P}_{gz}] + \log(\mathbb{E}[e^{\mathbb{P}_g \otimes \mathbb{P}_z}])}_{\text{DV KL Divergence}} \tag{2}$$

A WGAN-GPMI was implemented with the intention of learning to synthesize intrusion alerts. A λ value of 0.4 and hidden dimension size of 128 was used. G was configured to sample 64 points of Gaussian noise per alert generated. Both G and D were two-layer fully connected neural networks. The generator featured 4 independent fully connected layers in parallel on the output. These generate each of the 4 feature values used in the experiments discussed later on.

Due to the categorical nature of the data being generated, all features were one hot encoded and concatenated into a single vector per alert. These values were then transformed into real-world values by segmenting the vector into subcomponents whose length's equal the number of unique values for the given feature. The argmax of each of these subcomponents was then taken as a post-processing step to find the corresponding real world value generated.

The estimator (E) featured two independent inputs layers which mapped the noise and samples from G to a hidden representation. These hidden representation were then added together and then transformed into a mutual information estimate by the output layer. The network architecture may be seen visually in Fig. 2.

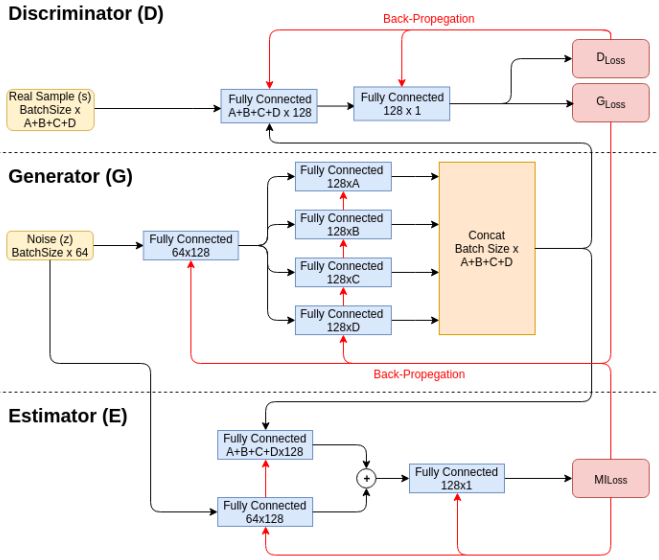

Fig. 2: Each network is trained end to end in order to create alerts which model the ground truth distribution as closely as possible.

Given that G receives gradient feedback from two independent loss functions, the adversarial loss and the mutual information estimate, the values of the gradient must be balanced to provide equal feedback to the network. Following the methodology of Belghazi *et al.* [6] all gradient updates to G were adaptively clipped to ensure that the Frobenius norm of the gradient resulting from the mutual information was at most equal to the adversarial gradient.

IV. Dataset for Evaluation

This work assesses how well WGAN-GPMI, in comparison with WGAN-GP, learns and generates synthetic intrusion alerts using the data collected through the 2017 National Collegiate Penetration Testing Competition (CPTC). For the competition, teams were tasked with penetrating into a mock network of election systems. The network topology featured several server systems hosted across a variety of subnets. Students were tasked with scanning, infiltrating, and exploiting vulnerabilities in the network within approximately 9 hours to exfiltrate and modify voter data. The dataset provides a unique opportunity for experimentation as it is completely comprised of actions known to be malicious. Though this data is unique to the competition it is worth noting that the preprocessing described herein is applicable to other datasets consisting of NIDS alerts.

The first preprocessing step applied to the data was to separate alerts on a per Destination IP basis. This allowed individual models to be trained for each system on the network, typifying the type of traffic seen at that target. Additionally, data from all of the teams could be compounded, allowing for the number of potential attacks taken on a single target to be more fully expressed during training. Segmentation on a per-target basis has several intuitive benefits: First, it allows for different vulnerabilities to be highlighted on each machine given commonly occurring alert features at that target. Additionally, it helps to remove noisy alert influence, such as scanning externally facing systems, from critical nodes in the network. Table I shows 2 Target IP addresses selected for experimentation from the CPTC'17 dataset; the operating system, high-level purpose, and number of alerts for the machine is also given.

TABLE I: Mapping of IP Address to Machine use/Purpose

IP Address	Operating System	Machine use	Number of Alert
10.0.0.27	Ubuntu	HTTP Server	3186
10.0.0.22	Ubuntu	MySQL Server	2974

Next, the dimensionality of the destination port feature was reduced based off common service categories run across a collection of ports provided by the Internet Assigned Numbers Authority [19]. This reduction drops the number of unique values from 1516 destination ports to 69 target services. Contextually, this has the effect of indicating what service is being targeted by attackers, rather than just knowing a specific port number.

Finally, a set of simple statistical criterion were used to segment timestamps into bins. Traditional modeling of cyber attacks use attack stages to segment actions into a series of contiguous stages with dependencies on previous stages. The beginning of an attack may consist of reconnaissance based actions, yielding information about which IP to target in later attack stages. Similarly, the CPTC dataset may be segmented to try and capture unique behaviors into different Time Bins.

Following the methodology shown by [14] bins were generated by smoothing the histogram timestamps and taking the first derivative to identify local minima and maxima. Then stages were cut if they contained at least 10% of the total data and consecutive events at the candidate point contained less than 0.5% of the total data. The goal of this ruleset was to capture significantly different types of traffic while not splitting bursts of data into multiple stages.

Table II shows the number of unique values present for each Target IP address tested after preprocessing the data. Additionally a single character symbol is defined for each feature in parenthesis to compact future plot labels.

TABLE II: Number of Unique Feature Values for Assorted target IPs

	Machine IP Address	
	10.0.0.27	**10.0.0.22**
Alert Signatures (A)	41	34
Dest. Port Category (D)	27	21
Source IPs (S)	6	6
Timebins (T)	8	8

V. Experiments & Results

Assessing the GAN models using the CPTC dataset was broken up into 4 stages. First, a GAN was trained to learn the distribution of the input data on a per Target IP basis and recreate it. Then a Histogram Intersection Score was calculated for all combinations of features to express how well the GAN had learned to recreate the dataset. Next, feature dependencies

for varying numbers of feature permutations were verified using the weighted, normalized, Conditional Entropy. Finally, the number of output modes dropped for each model was compared to show that the WGAN-GPMI model covered a larger percentage of the alert feature domain. To further demonstrate the value of improved output mode coverage, alerts were mapped to attack stages, showing that the output of the WGAN-GPMI model outputs attack stages not captured by the WGAN-GP model.

The WGAN-GPMI model was trained on each Target IP's alerts for a total of 300 epochs using the ADAM optimizer with a learning rate of $5e-5$, $\beta_1 = 0.5$, and $\beta_2 = 0.8$ with batches of 100 alerts.

Analyzing the degree of realism for artificially generated alert data is non-trivial. While other fields such as Computer Vision have created well defined metrics such as Inception Score [20] or allow for direct human analysis of image quality, no analogue exists for NIDS alerts. Several works have proposed the use of graph based metrics such as comparing nodes and their connectivity for both generated and real network traffic [21], or looking at low level parameters such as distributions of packets [22]. However, despite these works, there is no widely accepted methodology. To address this, we propose using the two aforementioned metrics, Histogram Intersection and Condition Entropy, to gauge the fidelity of our models. Additionally, we review the distributions directly and note the output modes and attacker behaviors captured by the models.

A. Histogram Intersection

The Histogram Intersection metric compares the similarity of two histograms within the same domain by computing the amount of overlap between them. It is naturally bounded between 0 and 1, intuitive to understand, extends to joint distributions of features, and may be graphed to directly visualize results. Let P represents the ground truth data histogram and Q represents the generated data histogram, each with N samples. The Histogram Intersection (G) is defined as

$$G(P,Q) = \frac{\sum_{i=0}^{N} min(P_i, Q_i)}{max(\sum_{i=0}^{N} P_i, \sum_{i=0}^{N} Q_i)} \quad (3)$$

The Histogram Intersections (G-scores) were computed for several targets when using both WGAN-GP and WGAN-GPMI. This paper reports the results for the Target IPs presented in Tables I and II: 10.0.0.27 and 10.0.0.22. Each model was sampled 1000 times to compute the standard deviation of the G-scores. Table III shows the average and standard deviation of these results. First, note that both WGAN-GP and WGAN-GPMI achieves reasonably good performance, even when considering the combination of all 4 feature values; Samples from WGAN-GP are able to achieve up to 60% intersection with the ground truth distribution while samples generated by the WGAN-GPMI model achieve up to 71% intersection. Secondly, note that for both IP addresses the Mutual Information constraint is able to increase the G-score. This is a direct reflection of the model having a closer match to the probability distribution of the ground truth due to the increase in output mode coverage. These results will be reviewed fully in Section V-C.

TABLE III: Histogram Intersection for all Feature Combinations

	Target Machine IP Address			
	WGAN-GP		WGAN-GPMI	
Features	**10.0.0.27**	**10.0.0.22**	**10.0.0.27**	**10.0.0.22**
A	0.658 ± 0.007	0.844 ± 0.005	0.833 ± 0.005	0.847 ± 0.006
D	0.660 ± 0.006	0.843 ± 0.005	0.846 ± 0.005	0.823 ± 0.007
S	0.867 ± 0.009	0.846 ± 0.008	0.909 ± 0.005	0.755 ± 0.005
T	0.760 ± 0.007	0.818 ± 0.008	0.815 ± 0.007	0.844 ± 0.008
A,S,D,T	0.548 ± 0.007	0.601 ± 0.007	0.718 ± 0.006	0.626 ± 0.008

B. Conditional Entropy

We introduce the use of weighted conditional entropy as a means to analyze how well WGAN-GPMI learns the dependencies between the features. The weighted conditional entropy (4) provides a numerical value expressing the amount of randomness in an output feature given another feature(s) value. This calculation weights the entropy of each possible input combination based off the probability that input i occurs as $\frac{|w_i|}{|w|}$ in order to obtain a single value expressing the weighted average conditional entropy for all values of a given feature.

$$\widehat{H}_{Y|X_0,X_1,...,X_m} = \sum_{i=0}^{N} \left(\frac{|w_i|}{|w|} * \sum_{j=0}^{Z} \left(p_{i|j} * \log(\frac{1}{p_{i|j}}) \right) \right) \quad (4)$$

In order to provide a consistent comparison between different feature combinations, the result of (4) is normalized by the entropy maximizing distribution for a discrete distribution with the same finite support, *i.e.,* the uniform distribution $\mathbb{U}$ with cardinality equal to the number of unique values in the conditional distribution. This bounds the metric between 0 and 1.

TABLE IV: Weighted Normalized Conditional Entropy for two Targets using WGAN-GPMI.

	Target Machine IP Address			
	Ground Truth Results		Generated Results	
Features	**10.0.0.27**	**10.0.0.22**	**10.0.0.27**	**10.0.0.22**
A|T	0.238	0.153	0.238	0.153
T|S	0.463	0.515	0.463	0.516
T|A	0.339	0.695	0.339	0.695
S|T	**0.252**	**0.263**	**0.186**	**0.252**
S|A	**0.752**	**0.831**	**0.239**	**0.526**
D|S	0.445	0.253	0.385	0.207
A|D	0.222	0.070	0.026	0.007
T|D	0.346	0.655	0.346	0.655
D|T	0.234	0.152	0.234	0.152
A|S	0.385	0.271	0.376	0.214
S|D	**0.779**	**0.856**	**0.260**	**0.474**
D|A	0.246	0.006	0.009	0.048
S|A,D	**0.747**	**0.829**	**0.226**	**0.474**
D|S,T	0.118	0.013	0.118	0.013
S|D,T	**0.025**	**0.101**	**0.025**	**0.101**
T|A,D	0.335	0.650	0.335	0.650
A|S,D	0.206	0.056	0.003	0.007
A|S,T	0.117	0.013	0.117	0.013
A|D,T	0.018	0.001	0.018	0.001
D|A,S	0.243	0.005	0.007	0.003
T|S,D	0.340	0.587	0.144	0.334
D|A,T	0.238	0.003	0.006	0.010
S|A,T	**0.178**	**0.100**	**0.012**	**0.100**
T|A,S	0.312	0.561	0.144	0.328
A|S,D,T	0.172	0.028	0.003	0.006
D|A,S,T	0.232	0.002	0.004	0.008
T|A,D,S	0.167	0.498	0.144	0.328
S|A,T,D	**0.306**	**0.558**	**0.004**	**0.100**

The weighted and normalized Conditional Entropy was tabulated in Table IV to demonstrate the WGAN-GPMI model's

ability to learn feature dependencies within an alert. Samples where Source IP is dependent upon other features within an alert are bolded. This feature in particular is not typically dependent upon other features within the dataset. Samples where the model distribution was significantly more deterministic than the ground truth distribution were highlighted in red. For the remaining features, the model performs well at learning feature dependencies within the data. In fact, there are 23 cases where the model learned dependencies within 10% of the ground truth conditional entropy.

C. Generated Attack Behavior Analysis

An important feature of this work is to leverage the learning capabilities of a GAN to generate diverse artificial attack data that is reflective of the behaviors observed in the ground truth dataset. We define the attacker behavior based off the type of actions taken, such as reconnaissance and data exfiltration. Alert attributes such as category or signature gives the inclination of attack type; however, the category is an arbitrary high-level description of the attack type that may not accurately represent the outcome of the action whereas the signature may be at too fine of granularity to depict the attack behavior. Thus, this work also assess the synthetically generated alerts by mapping the alert signatures to one of the defined attack stages given in Table V, by manually examining the objective and outcome described in the signature description.

TABLE V: Attack Stage Types

Attack Stage	Description
IP Scan	Scan to reveal IP addresses
Service Scan	Scan to reveal services active on a target
Targeted Scan	A targeted and specific scan on a machine
Social Engineering	Deceiving / manipulating individuals for malicious intent
Surfing	Browsing publicly available information to research target
Specific Exploits	Using a specific vulnerability on a target
Escalate Privledges	Gaining unauthorized privileges
Zero Day	Conducting an action not recorded and/or observed before
Malware Injection	Delivering malware to target
Degrade Operations	Reduce or interrupt “normal” functionality of a target
Data Exfiltration	Steal and extract sensitive information

Using this mapping we can see if GAN captures latent behaviors within the dataset even when it fails to output specific alert signatures that occurred explicitly in the dataset. Additionally, the output domain coverage for each model is shown to compare the model's performance on fine grained generation to that of the attack action distribution.

Figure 3 shows the attack stage coverage within the ground truth data as well as those generated from WGAN-GP and WGAN-GPMI. Note that the WGAN-GPMI model shows almost identical attack stage distribution as the ground truth data, exhibiting improvement over the WGAN-GP case. Specifically, it generated alerts pertaining to the Targeted Scanning stage with probability very close to the ground truth distribution. Meanwhile, the standard WGAN-GP model could not capture this output mode with probability greater than 1.8%, leaving a large gap in the generated data sample.

We further examine the number of output modes dropped or added by WGAN-GPMI. This was done by looking at all the unique alert feature combinations across A/D/S/T values that existed in the ground truth dataset versus those existed in the generated dataset. These sets of unique values were compared to see which modes were dropped, which were covered, and which existed in the generated set but not the ground truth set. We refer to these values as *Dropped*, *Covered*, and *Noisy* respectively.

Figure 4 shows these results for the two Target IPs as a series of bars. The bars marked ”Coverage” show the number of unique alert combinations (modes) that fall into each category. The bars marked ”Distribution” show the percentage of alerts from the generated distribution belonging to each category. Note that *Dropped* samples are not shown in the distribution as they inherently have 0% probability of occurring in the generated dataset.

First, note that a large percentage of alerts were covered for both Target IPs, accounting for 78$\sim$83% of the generated output distributions. A detailed look at the results reveal differences in the two Target IP cases. Target 10.0.0.22 shows superior coverage with only 4 output modes being dropped while adding a larger number of novel modes (though the percentage of alerts is still a minority). On the other hand, Target 10.0.0.27 has less noisy modes and alerts but still drops 14 modes from the ground truth data. It is possible that these dropped modes represent samples which have an extremely low probability of occurring; so much so that the mutual information constraint is insufficient to encourage the generation of these values. Further supporting this is the fact that even with less than half of the total output modes covered there is still an 83% chance that the outputs from this model do exist in the ground truth distribution.

VI. Concluding Remarks

This work shows the promise of using Generative Adversarial Networks as a means to expand Cyber Intursion Alert datasets. The fidelity of generated data was measured through Histogram Intersection and Conditional Entropy, both of which show the potential for GANs in Cyber Security. Through the use of a Mutual Information constraint on the generator critical output modes from the ground truth dataset are also generated by the proposed models. This work has provided the basis for future experimentation with larger cyber intrusion alert datasets. Additionally, using Recurrent GANs would allow for the generation of alert sequences, accounting for temporal dependencies in the data. Finally, modification of the Mutual Information constraint could allow for expert's to guide generator output. This would allow for targeted generation of particularly interesting aspects of the network, such as specific vulnerabilities or a particular type of attacker behavior.

References

[1] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li, “AI2 : Training a big data machine to defend,” in *Proceedings of IEEE 2nd International Conference on Big Data Security on Cloud*, Beijing, China, June 2016, pp. 49–54.

[2] P. Filonov, F. Kitashov, and A. Lavrentyev, “RNN-based early cyber-attack detection for the tennessee eastman process,” in *Proceedings of ICML Time Series Workshop*, Sydney, Australia, August 11 2017.

[3] S. Noel and S. Jajodia, “Advanced vulnerability analysis and intrusion detection through predictive attack graphs,” *Critical Issues in C4I, AFCEA Solutions Series. International Journal of Command and Control*, 2009.

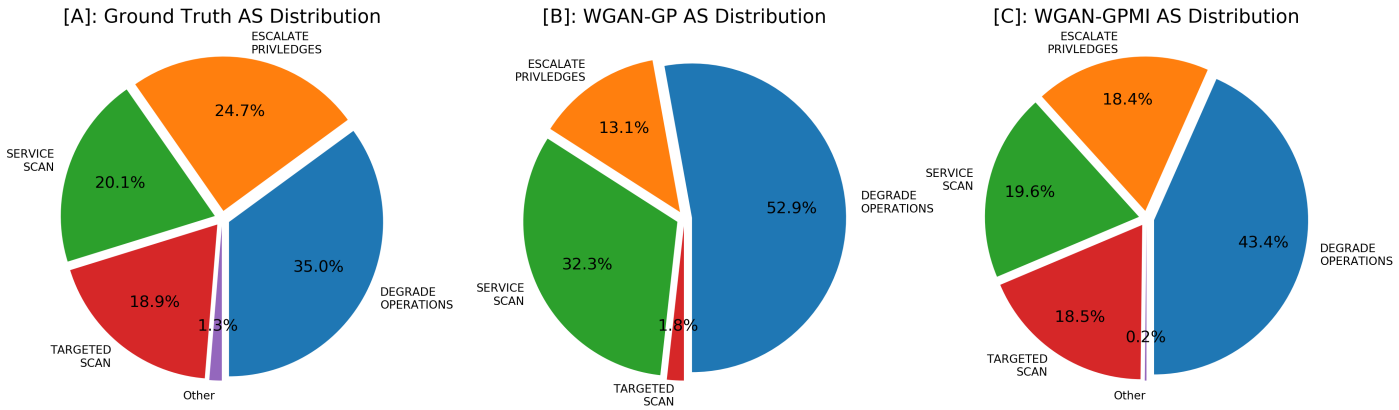

Fig. 3: Distribution of Attack Stages (AS) on Target IP 10.0.0.22. Note that the WGAN-GPMI model results [C] have a much closer probability distribution to the ground truth data [A] then the WGAN-GP Model [B].

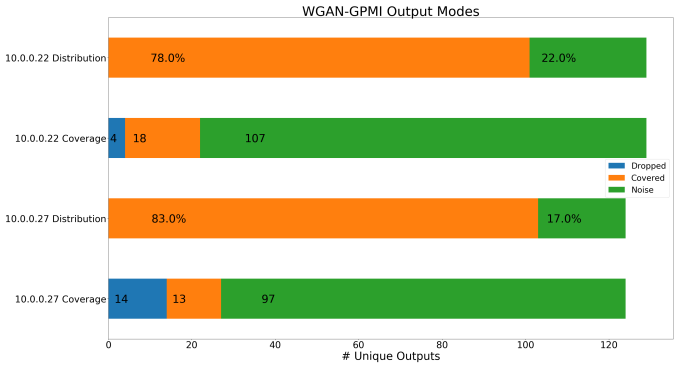

Fig. 4: The WGAN-GPMI model features less mode dropping than the WGAN-GP model, however the amount of probability mass assigned to noisy samples also increases.

[4] D. S. Fava, S. R. Byers, and S. J. Yang, "Projecting cyberattacks through variable-length markov models," *IEEE Transactions on Information Forensics and Security*, vol. Vol. 3, no. 3, pp. 359–369, September 2008.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[6] I. Belghazi, S. Rajeswar, A. Baratin, R. D. Hjelm, and A. C. Courville, "MINE: mutual information neural estimation," in *Proceedings of International Conference on Machine Learning*, Stockholmsmssan, Stockholm Sweden, July 10-15 2018.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML*, Sydney, NSW, Australia, August 6-11 2017, pp. 214–223.

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Proceedings of Advances in Neural Information Processing Systems 30*, Long Beach, California, USA, December 4-9 2017, pp. 5769–5779.

[9] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (To Appear)*, vol. abs/1812.04948, Long Beach, California, USA, June 15-21 2019. [Online]. Available: http://arxiv.org/abs/1812.04948

[10] H. Su, X. Shen, P. Hu, W. Li, and Y. Chen, "Dialogue generation with gan," in *Proccedings of AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, February 2-7 2018.

[11] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proccedings of AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, February 2-7 2018.

[12] M. Rigaki and S. Garca, "Bringing a gan to a knife-fight: Adapting malware communication to avoid detection," in *Proceedings of IEEE Security and Privacy Workshops (SPW)*, San Francisco, California, USA, May 24 2018.

[13] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: generative adversarial networks for attack generation against intrusion detection," *CoRR*, vol. abs/1809.02077, 2018. [Online]. Available: http://arxiv.org/abs/1809.02077

[14] I. Perry, L. Li, C. Sweet, S. Su, F. Cheng, S. J. Yang, and A. Okutan, "Differentiating and predicting cyberattack behaviors using lstm," in *IEEE Conference on Dependable and Secure Computing*, Kaohsiung, Taiwan, December 10-13 2018.

[15] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 592–605.

[16] I. Faber and G. Malloy, "Deep security: Cyber security threat behavior classification," "http://cs230.stanford.edu/projects_spring_2018/reports/8285947.pdf", 2018.

[17] I. Amit, J. Matherly, W. Hewlett, Z. Xu, Y. Meshi, and Y. Weinberger, "Machine learning in cyber-security - problems, challenges and data sets," vol. abs/1812.07858, 2019. [Online]. Available: http://arxiv.org/abs/1812.07858

[18] L. Hu, Z. Zhang, H. Tang, and N. Xie, "An improved intrusion detection framework based on artificial neural networks," in *Proceedings of 2015 11th International Conference on Natural Computation (ICNC)*, Zhangjiajie, China, August 15-17 2015, pp. 1115–1120.

[19] J. Touch, E. Lear, A. Mankin, M. Kojo, K. Ono, M. Stiemerling, L. Eggert, A. Melnikov, W. Eddy, A. Zimmermann, B. Trammell, J. Iyengar, A. Mankin, M. Tuexen, E. Kohler, and Y. Nishida, "Service name and transport protocol port number registry," 2018. [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proceedings of Conference on Neural Information and Processing Systems*, vol. abs/1701.00160, Barcelona, Spain, December 2016, pp. 2234–2242.

[21] S. Iannucci, H. Kholidy, A. Dhakal Ghimire, R. Jia, S. Abdelwahed, and I. Banicescu, "A comparison of graph-based synthetic data generators for benchmarking next-generation intrusion detection systems," in *Proceedings of IEEE Cluster Conference*, Hawaii, USA, September 5-8 2017.

[22] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. Vol. 56, no. 15, October 2012.