On the Variety and Veracity of Cyber Intrusion Alerts Synthesized by Generative Adversarial Networks

CHRISTOPHER SWEET, Department of Computer Engineering, Rochester Institute of Technology
STEPHEN MOSKAL, Department of Computer Engineering, Rochester Institute of Technology
SHANCHIEH JAY YANG*, Department of Computer Engineering, Rochester Institute of Technology

Many cyber attack actions can be observed but the observables often exhibit intricate feature dependencies, non-homogeneity, and potentially rare yet critical samples. This work tests the ability to learn, model and synthesize cyber intrusion alerts through Generative Adversarial Networks (GANs), which explore the feature space by reconciling between randomly generated samples and data that reflect a mixture of diverse attack behaviors without apriori knowledge. Through a comprehensive analysis using Jensen-Shannon Divergence (JSD), Conditional and Joint Entropy, and mode drops and additions, we show that the Wasserstein-GAN with Gradient Penalty and Mutual Information (WGAN-GPMI) is more effective in learning to generate realistic alerts than models without Mutual Information constraints. We further show that the added Mutual Information constraint pushes the model to explore the feature space more thoroughly and increases the generation of low probability, yet critical, alert features. This research demonstrates the novel and promising application of unsupervised GANs to learn from limited yet diverse intrusion alerts to generate synthetic alerts that emulate critical dependencies, opening the door to proactive, data-driven cyber threat analyses.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Neural networks**; • **Information systems** → *Similarity measures.*

Additional Key Words and Phrases: GAN, Intrusion Alert Analysis, Cyberattack Characterization

ACM Reference Format:
Christopher Sweet, Stephen Moskal, and Shanchieh Jay Yang. 2020. On the Variety and Veracity of Cyber Intrusion Alerts Synthesized by Generative Adversarial Networks. *ACM Trans. Manag. Inform. Syst.* 1, 1, Article 1 (January 2020), **??** pages. https://doi.org/10.1145/3394503

1 INTRODUCTION

The prevalence of cyber intrusion activities has led to diverse observables that often puzzle analysts and researchers while determining the intent and actions of the attack. Utilizing machine learning techniques to assist in extracting behavioral patterns from the intrusion alerts is a logical step; such techniques will need to be unsupervised since it is unlikely to obtain truths for adversary behaviors. With a diverse mix of attack behaviors and potentially rare yet critical feature combinations in any given set of observables, the unsupervised technique must go beyond just learning the exact feature dependencies exhibited in the data. Generative Adversarial Networks (GANs) serve as a

*This research is supported by NSF SaTC Awards #1526383 and #1742789.

Authors' addresses: Christopher Sweet, crs4263@rit.edu, Department of Computer Engineering, Rochester Institute of Technology, 1 Lomb Memorial Drive; Stephen Moskal, Department of Computer Engineering, Rochester Institute of Technology, 1 Lomb Memorial Drive; Shanchieh Jay Yang, jay.yang@rit.edu, Department of Computer Engineering, Rochester Institute of Technology, 1 Lomb Memorial Drive.

2158-656X/2020/1-ART1 $15.00
https://doi.org/10.1145/3394503

plausible solution to discover and synthesize data with similar, but not exactly the same, feature dependencies exhibited in historical data. The generated intrusion alerts not only help characterize the intrusion observables but also can augment the often limited true observables to strengthen prediction and other analytics for cyber defense.

Cyber intrusion alerts have been used to identify anomalous activities [**?**] [**?**] [**?**], discover network vulnerabilities [**?**], and profile bad-actor behaviors [**?**]. Imagine research works like these enhanced with synthetic data that resembles intrusion alerts from previous attacks. This work explores and investigates the use of two types of GANs in their effectiveness to generate intrusion alert data when given representative real world data. We consider eight target IP addresses attacked by two sets of ten independent teams as part of the Collegiate Penetration Testing Competitions in 2017 and 2018. The application of unsupervised machine learning, in the form of GANs, to these data includes a means for driving better coverage of the feature domain in model outputs and allowing more rare but critical events to be synthesized. To the best of our knowledge, this work is the first attempt to synthesize intrusion alerts using GANs *to analyze intricate, and potentially rare, attacker behaviors from observable malicious activities.*

First proposed by Goodfellow *et al.* , Generative Adversarial Networks (GANs) [**?**] are **unsupervised** deep learning models that learn to emulate data from a training dataset, by reconciling between generated and real samples. This framework was subsequently improved by Arjovsky *et al.* [**?**] and Gulrajani *et al.* [**?**] by optimizing models via the Earthmover Distance. Belghazi *et al.* [**?**] further introduced an additional loss term to drive diverse model outputs. Through these improvements, GANs have achieved state of the art results in generating data with respect to images [**?**] [**?**] [**?**], text [**?**], and sound [**?**] [**?**].

Additionally, GANs have been applied to network traffic to modify and obfuscate malicious traffic [**?**] [**?**] [**?**] [**?**]. These adversarial samples are created to avoid being flagged by Network Intrusion Detection Systems (NIDS). Despite the widespread usage of GANs, there is a lack of research in using GANs to synthesize NIDS alerts for analysis from the *target* IP perspective, which is essential to understanding the attack behaviors exhibited during an attack. Given the low critical-to-noisy-alert ratio for malicious activities, a means to generate meaningful data based off limited observables could enable researchers to reveal network vulnerabilities, understand attacker behaviors, and augment other data driven models relying on malicious alerts for training.

To address the current void of data driven generative models for Cyber Intrusion data, we propose using WGAN-GPMI [**?**]. This work specifically investigates how this GAN architecture can be used to generate synthetic intrusion alerts by learning the sparsely distributed categorical features of said alerts from samples of malicious network intrusions. This unsupervised learning problem is particularly challenged by the need to generate rare yet critical alerts. This makes the unique application of WGAN-GPMI well-suited to the domain of Cyber Intrusion alerts; better than the standard WGAN formulation. Through well-established Information Theoretic metrics such as Jensen-Shannon Divergence and Conditional Entropy we show that our models are able to generate new alerts which exhibit behavior similar to that of the training data, and do so better than comparative models without Mutual Information constraints. We further show that GANs learn to generate alerts which emulate attacker behaviors without explicit tasking to do so.

This research realizes these claims by applying WGAN-GPMI to NIDS data collected via Suricata (`https://suricata-ids.org/`) from the 2017 and 2018 Collegiate Penetration Testing Competition (CPTC) (`https://nationalcptc.org/`). CPTC'17 had ten student teams attempt to penetrate and exploit vulnerabilities of a virtualized network that manages election systems. CPTC'18 tasked new student teams with penetration into an autonomous driving IT infrastructure including virtualized embedded systems, mobile applications, and processing servers. Rather than directly focusing on the specific behaviors exhibited by each team for the two datasets, the data was segmented based off

the IP address being attacked. Segmentation on a *per target* basis yielded independent datasets for each target, featuring unique attack strategies from each team. The use of data from CPTC'17 and CPTC'18 illustrates the unsupervised nature of these models, as they were applied to completely disparate datasets without explicit labels.

The remainder of the paper is structured as follows: Section **??** 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 **??** and Section **??** discusses the GAN model as well as preprocessing and analysis methods employed for generating synthetic intrusion alerts. Finally, Section **??** discusses the observations made from reviewing generated data and Section **??** gives the concluding remarks and future works of this research.

2 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. However, with the use of Deep Learning models comes the need for massive amounts of quality training data; several ongoing works in this field cite the need for more data as a limitation to their current research [**?**] [**?**] [**?**] [**?**].

In particular, LSTM models are shown by Perry *et al.* [**?**] to suffer significantly lower accuracy when the dataset provided for training is not large enough to be representative of previous observations. This holds true for both classifying cyber attackers and for predicting the next attack action taken. This message is echoed by Faber and Malloy [**?**] despite having a dataset of over 600,000 alerts and promising classification accuracy. They note that the availability of quality labeled data and a low signal-to-noise ratio for malicious activity are both outstanding issues.

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.* [**?**] proposed the use of GANs in generating malicious network traffic which appeared as benign network traffic. This allowed 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 down to 0%. Similarly, Lin *et al.* [**?**] apply GANs to obfuscate traffic with the intention of directly deceiving a NIDS. Their model makes use of 9 discrete features and 32 continuous features to modify attack actions to avoid detection. 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 different classifiers when benchmarked using the NSL-KDD benchmark provided by [**?**]. Despite the promise of these results, several of the pathological issues identified in the original KDD Cup Dataset remain in the improved NSL-KDD benchmark [**?**] [**?**].

None of the aformentioned works apply GAN models to the generation of Intrusion Alert datasets from the target perspective. Intuitively, a target machine attracting many malicious activities may suffer from some kind of coherent attack strategies that could be learned by an unsupervised model and generated en masse for further study. We hypothesize that WGAN-GPMI can learn these attack strategies to generate high fidelity alerts resembling the original dataset. GANs have been previously shown to augment small semi-labeled datasets across a number of fields [**?**] [**?**] [**?**] [**?**] [**?**]. The application of Wasserstein-GAN with Gradient Penalty and Mutual Information (WGAN-GPMI) specifically addresses the learning of attack strategies from relatively 'small' datasets, each representing intrusion alerts observed for a target. A total of eight targets are considered from

isolated instances of two different networks. Methods to quantify, analyze, and confirm the veracity of these synthesized alerts are also introduced.

Applying GANs to cyber intrusion alerts is non trivial as the challenges posed by the data directly affect the training of GANs. The distribution of alert features cannot be modeled trivially and critical alert features may occur with low probability. The potential for mode dropping is simultaneously high and problematic due to the contextual meaning of results. In order to try and address this in other fields, Belghazi *et al.* [**?**] proposed adding a Mutual Information constraint on the Generator. Applied to cyber intrusion alerts, the Mutual Information constraint would encourage the generation of all alert features, including the rare actions that are indicative of targeted attacker behavior. Given the unsupervised nature of GANs no class labels for each alert are required to train the models, allowing any NIDS data, from any network, to be used as training data.

This work applies two types of GAN models to cyber intrusion alerts and studies the results in depth to judge the veracity of synthetic alerts. A generalized novel set of preprocessing steps are also introduced to provide contextually useful information from the samples generated by the GAN models. Training on CPTC'17 and CPTC'18 data illustrates the ability of these models to recreate small imbalanced datasets that exhibit different attack strategies tailored to the target under attack. Additionally, intra-alert feature dependencies are captured and revealed by the data sampled from the GANs' output, showing that critical interactions between feature values are preserved by the models. Finally, using a mapping of alert signatures to attack stages opens the potential for attacker behavior to be inferred and learned by GANs.

3 GAN MODELS FOR CYBER INTRUSION ALERTS

Generative Adversarial Networks are comprised of a pair of networks which learn the structure of a dataset, commonly referred to as the training dataset, in an unsupervised manner and emulate it to synthesize new datasets. One network, the generator (G), attempts to create samples which seem to belong to a training dataset. The other network, the discriminator (D), takes inputs from the training 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 training data set are marked as real and all samples created by G are marked as fake. Throughout training each network improves, resulting in more and more realistic output samples.

The Wasserstein GAN, first proposed by Arjovsky *et al.* [**?**] extends the concept of a GAN but with increased stability during training. This was subsequently improved by Gulrajani *et al.* [**?**] by adding a gradient penalty term to regularize the gradients of D. The gradient penalty creates a 1-Lipschitz constraint on the discriminator during training by sampling noise from $\mathbb{P}_z$ and constraining the gradient of the L2 norm of D($\mathbb{P}_z$) to 1. Additionally, D is given real samples $\mathbb{P}_r$ and generated samples $\mathbb{P}_g$ in a 5:1 ratio per epoch of training; this is done to increase the utility of gradients provided by D. These modifications resulted in the discriminator loss function provided in (**??**). This model is referred to as Wasserstein GAN with Gradient Penalty (WGAN-GP).

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

Despite these improvements to the loss function for the discriminator, the generator loss was left unmodified. Belghazi *et al.* [**?**] changed this by adding a mutual information term to the generator's loss. This contribution maximized an approximation of the mutual information between

the generator's noise input $\mathbb{P}_z$ and it's output samples $\mathbb{P}_g$ by minimizing the Donsker-Varadhan (DV) representation of the Kullback-Leibler (KL) divergence as shown in (**??**).

The DV KL term was added by using a neural network (E) to maximize an estimate of 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.

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

Since mutual information is theoretically unbounded, gradient updates resulting from it could overwhelm the adversarial gradients resulting from the loss provided by D. In order to address this all of the gradient updates to the generator 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 [**?**], as shown in (**??**). Note that g_{norm} is the normalized gradient, g_a is the adversarial gradient resulting from (**??**), and g_m is the gradient resulting from the DV KL portion of (**??**).

$$g_{norm} = g_a + min(||g_a||, ||g_m||)(\frac{g_m}{||g_m||}) \tag{3}$$

Given the addition of these terms, the WGAN-GPMI model has a means to synthesize data with high fidelity, without exhibiting output mode collapse, and will attempt to drive exploration of the feature space. These attributes are ideally suited to the generation of Cyber Intrusion alert data as they provide scalability down to small sample-limited datasets which would historically be outside the scope of Deep Learning models.

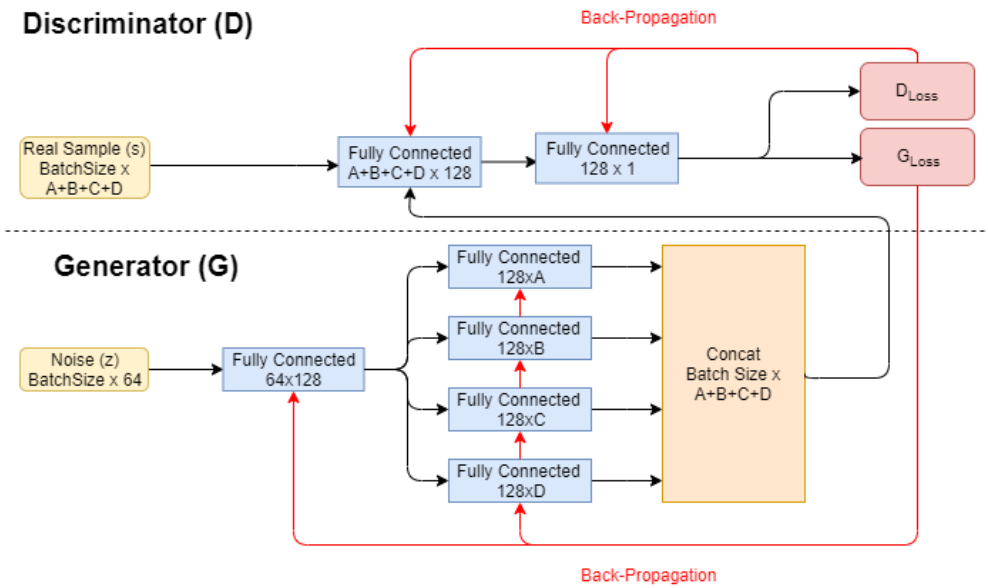

Fig. 1. WGAN-GP Model Architecture: The real samples provided to the Discriminator are one hot encoded in the same fashion as the Generator's output

WGAN-GP and WGAN-GPMI models were implemented to generate malicious cyber intrusion alert features from historical data, herein referred to as alerts. Each network in the models were configured with layers that had a hidden dimension size of 128. Both models used a vector of 64 normally distributed points per sample as input to G.

Fig. **??** provides a visualization of the WGAN-GP architecture used. The first layer of the discriminator took input from the training data as well as G. A second layer then generated a probability that the sample was from the training data set by using the Sigmoid activation function.

The generator featured 4 independent fully connected layers in parallel on the output. These generated each of the 4 features tested. Due to the categorical nature of the data being generated all features were one hot encoded and concatenated into a single vector per alert when input to the discriminator. For analysis of generated samples, each output was 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's first layer featured two fully connected layers; one for the input to G and one for the output of G. The second layer took the linear combination of these layers and computed a single output value representing the mutual information estimate. The addition of the estimator in WGAN-GPMI may be seen in Fig. **??**.

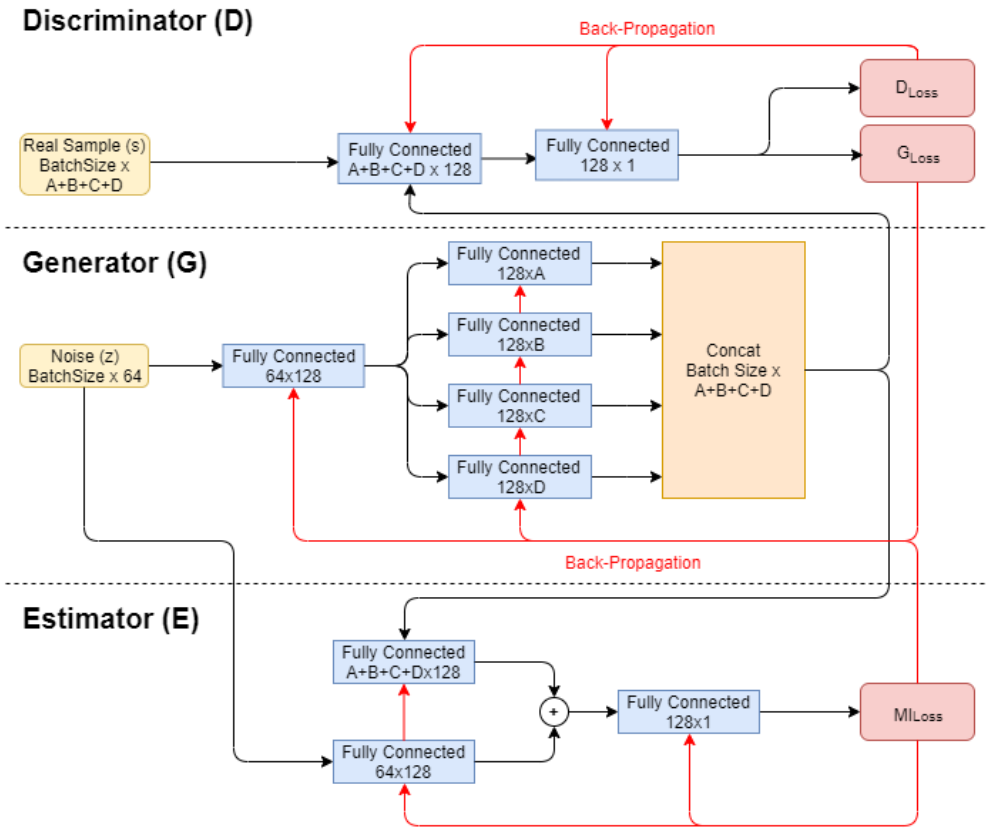

Fig. 2. WGAN-GPMI Model Architecture: The Generator and Discriminator are left unmodified from the WGAN-GP model

In both figures yellow boxes represent inputs to the network. The blue boxes represent weight layers of the network which are updated via back-propagation. The concatenation box at the end of the generator is a post processing step to form the aforementioned one hot encoded alert vector from each of the feature outputs. Finally, the red boxes and lines represent feedback paths which update network parameters during each step of training.

4 EXPERIMENTAL DESIGN AND VERACITY ANALYSIS

Training and testing of the CPTC dataset was broken up into four stages. First, a GAN was trained to learn the distribution of the input data on a per target IP basis and emulate it. Then the JSD was calculated for all feature combinations to quantify how well the GAN had learned to emulate the dataset. Next, feature dependencies were analyzed by computing the Conditional Entropy for all unique feature permutations. 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. These results are also shown to highlight specific attacker behaviors captured by the WGAN-GPMI models output which are lacking in WGAN-GP alerts.

Given that the proposed models were all fully connected, temporal dependencies between chains of alert are not considered. If using CNN or RNN based models the above metrics would need to be augmented with metrics which consider similarity between chains of alerts. Such metrics fall outside of the scope of this work; we refer the reader to review graph theoretic metrics such as longest common subsequence or transition matrix similarity if pursuing time-based feature analysis.

4.1 CPTC Dataset & Preprocessing

The data used for these experiments comes from the National Collegiate Penetration Testing Competition from 2017 and 2018. In 2017, teams were tasked with penetrating and exploiting network vulnerabilities in a virtualized network managing election systems. In 2018 teams were required to attack a multifaceted system handling autonomous cars; this included host based systems, servers, and mobile assets such as cell phones running an app. Each team had around 9 hours to scan, infiltrate the network, and exfiltrate information from the target. Both datasets provide a unique opportunity for Machine Learning experimentation as they are completely comprised of malicious actions as teams attempt to compromise the target network.

Prior to being input to the models as training data, significant preprocessing was performed. This not only reduced the dimensionality of each of the features but also increased the contextual utility of generated alerts. Though this data is unique to the competition it is worth noting that the preprocessing described herein is applicable to any dataset consisting of NIDS alerts. For both datasets four features were considered; Alert Signature, Destination Port, Source IP and Timestamps. The usage of these features provided contextual information regarding what type of action took place, where it originated from and targeted, and when it occurred.

The first preprocessing step applied to the data was the segmentation of alerts on a per-Destination IP basis. This allowed individual models to be trained for each system on the network. Additionally, data from all of the teams could be compounded, allowing more unique attacker behaviors to be captured for each target. Segmentation on a *per-target* basis has several intuitive benefits for analysis as well: First, it allows for different vulnerabilities to be highlighted on each machine given commonly occurring alert features at that target. Secondly, it helps to remove noisy alert influence from critical nodes in the network. For example, internet facing IPs may contain a significant amount of scanning activity, drowning out exfiltration related alert features at nodes further embedded in the network. Finally, the information extracted from alerts on a per target basis

is actionable, as network administrators can use commonly targeted services to modify network settings at the given system to mitigate future attacks.

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 [**?**]. This reduction drops the number of unique values from 1516 destination ports to 69 destination services for the CPTC'17 dataset across all the data. Contextually, this has the effect of indicating what service is being targeted by attackers, rather than just knowing a specific port number. Herein the processed Destination Ports are referred to as Destination Services. Additionally, this step can easily be expanded or customized on a per network basis given individualized configuration of services.

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. It is the goal of this preprocessing step to segment timestamps into discrete bins that capture these unique attack stages. Following the methodology shown by Perry *et al.* [**?**], bins were generated by smoothing the histogram timestamps and taking the first derivative to identify local minima and maxima. The data were segmented into different bins at each extrema 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. This ruleset captured significantly different types of traffic in each bin while not splitting bursts of data into multiple stages.

Tables **??** and **??** shows the number of unique values present for each target IP tested after preprocessing the data for CPTC'17 and CPTC'18. Additionally a single character symbol is defined for each feature in parenthesis for future analysis. We limit our research to the four IPs with the greatest number of alerts from each competition dataset, however the number of models could be scaled up to match the number of systems in the network, provided that each featured sufficient data to train. Despite the significant increase in the number of alerts captured in the CPTC'18 data, the number of unique features does not vary proportionally. This feature sparsity in addition to the overarching data sparsity are two key challenges in Cyber Intrusion alert data.

Table 1. Number of Unique Feature Values for Assorted Target IPs from CPTC'17

	Target Machine IP Address			
	10.0.0.100	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**
Number of Alerts	3388	3186	2974	2182
Alert Signatures (A)	13	14	8	15
Destination Service (D)	10	9	7	12
Source IPs (S)	10	10	8	6
Timebins (T)	8	8	5	6

It is worth noting that if applied to real world attack data, segmenting Source IP by subnet could provide useful context to the originator of the attack. However, given the limited scope of the CPTC environments this processing was not applied here. Additionally a well planned or socially engineered attack which looked like normal behavior within the network could be completed without generating any alerts. These shortcomings remain outstanding challenges of NIDS as a whole and fall outside the scope of this work.

Table 2. Number of Unique Feature Values for Assorted Target IPs from CPTC'18

	Target Machine IP Address			
	10.0.1.46	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**
Number of Alerts	9861	8695	7996	7475
Alert Signatures (A)	13	10	10	8
Destination Service (D)	14	9	10	8
Source IPs (S)	8	7	8	5
Timebins (T)	15	12	6	6

4.2 Jensen Shannon Divergence

Several statistical metrics were considered for the comparison of generated and training data alerts. These included Kullback Leibler Divergence (KLD) and Jensen Shannon Divergence (JSD). Additionally, combinations of features were considered in both the training data and generated datasets. For example, the divergence of all possible combinations of values for Alert Signature and Destination Service is one class of combinations. The divergence of these combinations of features was taken by representing feature combinations as joint distributions; Herein, the number of features included in each distribution is referred to as an m-tuple. The JSD of varying m-tuple distributions were then reviewed to judge the quality of data synthesized by each model compared to the training data used for training.

Output mode collapse, a key challenge for GANs, occurs when G begins to output a single value repetitiously. This is especially challenging with imbalanced datasets such as CPTC, as learning to emulate only the most probable attack action could still yield a distribution that appears similar to the training data. To counter this, a non-linear comparison needs to be made between the two distributions.

KLD, given in (**??**), was considered as a candidate given that it incurs an exponential penalty for failing to represent output modes. However, the KL Divergence is zero intolerant and asymmetric. This would require special handling of null outputs as well as a defined convention of which probability distribution is considered P and which is given by Q.

$$D_{KL}(P||Q) = -\sum_{x \in X} P(x) \log_2 \frac{Q(x)}{P(x)} \tag{4}$$

The JSD, given in (**??**), was then considered. It is both zero tolerant and symmetric while maintaining the penalty for failing to accurately represent the training data probability distribution. Additionally by using a base 2 logarithm for each term in the JSD it is naturally bounded between 0 and 1, with all attributes measured in bits. Intuitively, the JSD for comparing generated results to training data samples can be thought of how much additional information needs to be learned by the GAN model in order to perfectly emulate the training dataset; when the divergence is high a large amount of information needs to be learned. When it is low, only a small discrepancy exists. These attributes make JSD an excellent metric for the comparison of high fidelity synthetic distributions to their training data counterparts. Furthermore, JSD has been analyzed as a part of the basic modeling of GAN's since their inception in [**?**].

$$M(P, Q) = \frac{1}{2}(P + Q) \tag{5}$$

$$D_{JS}(P||Q) = \frac{1}{2}(D_{KL}(P||M) + D_{KL}(Q||M)) \tag{6}$$

4.3 Dependencies within Alert Features

To confirm that the models correctly learned feature dependencies from the training data the Conditional Entropy was computed. The previous m-tuple notation was extended to include a conditional equivalent, $Y|X$-tuples, which define a single feature Y given a vector of features X. The Conditional Entropy of $Y|x_1, x_2, ..., x_m$ is given in (**??**) and was computed for both the generated and training data. In this equation the weight term $p_{x_1,x_2,...,x_m}$ represents the probability of the input conditioning values occurring. This weighting enables a single Conditional Entropy score to be given for each m-tuple combination, even though the Joint Entropy may vary depending on the value of $x_1, x_2, ..., x_m$.

$$\widehat{H}_{Y|x_1,x_2,...,x_m} = \sum_{x \in X} \left(p_{x_1,x_2,...,x_m} * - \sum_{y \in Y} \left(p_{y|x_1,x_2,...,x_m} * \log(p_{y|x_1,x_2,...,x_m}) \right) \right) \tag{7}$$

This score was then normalized in (**??**) by dividing the weighted entropy by the entropy maximizing distribution for a discrete dataset with finite support; the uniform distribution $\mathbb{U}$ with cardinality equivalent to the number of unique elements in the feature value Y being considered. Note that this uses the cardinality of Y from the generated distribution, not the cardinality from the training data Y distribution. Using this metric, entropy values close to 0 indicated that a given input resulted in a particular output with near determinism. Entropy values of 1 indicated that given a particular input condition all outputs are equally probable. By normalizing the Conditional Entropy, varying Target IPs may be compared directly despite having different feature distributions. Most importantly, comparing the normalized Conditional Entropy of the training data and generated data provide a numerical means to evaluate how well the GAN learned to mimic feature interactions seen in various attacks in the training data.

$$\overline{H}_{Y|x_1,x_2,...,x_m} = \frac{\widehat{H}_{Y|x_1,x_2,...,x_m}}{H(\mathbb{U}_Y)} \tag{8}$$

Similarly, the Joint Entropy was computed for all m-tuples using (**??**). This metric provides a baseline for analyzing the results of the normalized Conditional Entropy by illustrating the randomness of the data if feature dependence is not considered. Additionally, the relationship between JSD and Joint Entropy is considered to demonstrate how additional randomness in the distributions correlates to additional divergence in output distributions.

$$H(x_1, x_2, ..., x_m) = - \sum_{x_1,x_2,...,x_m} p(x_1, x_2, ..., x_m) * \log \left(p(x_1, x_2, ..., x_m) \right) \tag{9}$$

By computing conditional and Joint Entropy the ability of GANs to learn latent interactions in the training data despite no explicit requirement to do so is demonstrated. These interactions are directly related to the attack actions taken, providing insight into attacker behavior and dependencies within an attack.

4.4 Output Modes and Attack Stages

Finally, the purpose of adding in the mutual information constrained model (WGAN-GPMI) was to palliate mode dropping. In order to evaluate this, a two step analysis process was employed. First, the number of output modes dropped for all feature combinations was collected. Then, to provide a result with direct contextual meaning to cybersecurity, the generated alerts were mapped to attack stages to show that WGAN-GPMI is capable of synthesizing alerts pertaining to more unique attack stages than the WGAN-GP model is. Furthermore these attack stages occurred with probabilities far closer to that of the training data, even when dealing with sparse feature values.

Following the work in [**?**], we define the attack stages 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 a granularity to depict the attack behavior. Thus, this work also assesses the synthetically generated alerts by mapping the alert signatures to generalized attack stages based off the objective and outcome described in the signature description.

Using this mapping we can see if GANs captured latent attacker behaviors within the dataset even when they failed 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.

5 ANALYSIS AND FINDINGS

Each model used was trained using individualized hyper-parameter settings. After training each model was tested using the proposed metrics in Sections **??**, **??**, and **??**.

Section **??** covers the tuning of these hyper-parameters. Additionally, the fundamental structure of the CPTC'17 and CPTC'18 data are used to formulate a set of criterion for application to other datasets. Sections **??** through **??** cover the application and analysis of the proposed metrics.

5.1 Hyperparamter Tuning and Training Considerations

Following the work in [**?**], an exhaustive hyperparameter sweep was performed for both WGAN-GP and WGAN-GPMI models to find the values which resulted in optimal results across all m-tuples of features. Table **??** illustrates the range of values swept over for the initial search performed on CPTC'17 data. From this sweep the parameters were then tested against the CPTC'18 data. Output mode collapse was then observed for the WGAN-GPMI, resulting in the need for an increased *lambda* value to help regularize the gradients of the Discriminator. Additionally, it was found empirically that increasing the number of epochs for the WGAN-GPMI model to 300 resulted in improved results across all m-tuples.

Table 3. Candidate Parameters for WGAN-GP and WGAN-GPMI

WGAN-GP Parameters						**WGAN-GPMI Parameters**			
Lambda	0.05	0.1	0.2			0.2	0.3	0.4	
Batch Size	10	25	50	100	150	50	100		
Learning Rate	5e-5	5e-4	1e-3			5e-5	1e-4	5e-4	1e-3
Hidden Dimension	128	256	384			64	128	256	
Epochs	100	150	200			150	200	250	
Number of Unique Combinations	405					216			

Ultimately this parameter search resulted in the following hyper-parameter settings: The WGAN-GP model was trained for a total of 200 epochs, while the WGAN-GPMI model was trained for 300 epochs. The *lambda* value was set to 0.1 and 1.0 for the CPTC'17 and CPTC'18 data respectively. All other hyperparameter values were held constant for both models. *Batch_size* was set to 100, and the *hidden_dimension* was set to 128. The parameters of the ADAM optimizer also followed the work performed in [**?**]. The *learning_rate* was set to $5e-5$ with $\beta_1 = 0.5$ and $\beta_2 = 0.8$. Finally, the entirety of the alerts collected for each target IP were used in training, as there is no separation of training and test sets in unsupervised models. For each target IP, the disciminator had it's weights updated 5 times for each time the generator's weights were. Given that each target contained a

different number of alerts in the training dataset, the overall number of updates each network went through was variable.

By forcing all models trained on a given CPTC dataset to use the same hyperparameters, we demonstrate that the proposed models are able to generalize their output. Concisely, a single set of hyperparameter values is suitable to the generation of a wide variety of alert structures, across a variety of targets IPs. Treating the two datasets as separate experiments is a critical test of model generalizability, as performing an exhaustive hyperparameter search for all systems in a network would quickly become infeasible.

When training on an Intel Core i7 4558U running at 2.8 Ghz with 8 GB of RAM and no GPU, it took an average of 10 minutes to train with an average of 7626 alerts as the training set. Training times are hindered primarily by the difficulty of training unsupervised models, such as WGAN-GPMI, which have loss functions that oscillate as the Generator, Discriminator, and Estimator compete. Despite this, the training times for these models are low compared to other works in the field of Deep Learning where training time may be on the order of hours or days. The relatively small number of alerts required to train and use of fully connected networks was a driving factor of this speed, however came at a cost; temporal dependencies were not directly learned by the models. Moving to models such as LSTM that do model temporal dependencies would greatly increase the amount of time required to train each model.

Upon completion of training, alert synthesis was shown to be quick. Using an average taken over 100 sample dataset generations, with cardinality equal to the respective training dataset, it took 0.0092 seconds to synthesize a new set of alerts for a given target. These tests were performed using the same hardware as described above. Given the speed of model evaluation, it is the amount of time and data required for training that hinder the ability to scale these models to generating alerts for the entirety of a network. Superior hardware, such as GPU acceleration and specialized Tensor Processing Units, offer solutions to the time required for training.

To address the question of how much data is required for training, we consider the criterion for applying these models to other Cyber Intrusion Alert datasets. Given that a neural network acts as an approximator of high-order non-linear functions, no specific distribution is required in the training data. The current models assume that only 4 features of each alert are used for training and generation; Alert Signature, Source IP, Destination Port, and Timestamp. In order to scale these models to include other alert features an additional fully connected layer would be required mapping the last hidden layer of the generator to an output. Also, the cardinality of each alert feature when one hot encoded must be known in order to provide an output vector with size great enough to map each possible output value for the given feature.

The work presented here demonstrates that a relatively small number of alerts is required to learn intricate feature dependencies from sparse data. In the alerts derived from CPTC'17, around 3000 alerts were included for training a WGAN-GPMI model on each Target IP. Despite this, our metrics show that the models are able to learn the latent structure of the alerts. Determining a definitive lower bound for data sufficiency with respect to training GANs remains an open problem. However, empirical study from [**?**], show that unique samples that occurred less than 100 times were rarely generated, even when implementing WGAN-GPMI. Thus datasets which are sparse to the point that most feature combinations have less than 100 unique instances are prone to poor results. Historically, insufficient training of GAN based models results in output mode collapse or noisy outputs, which have no semantic meaning, *dominating* output. Additionally, in order to scale these models to large-scale, real world networks, a large corpus of data representative of numerous attack vectors must be obtained for each system. Outside of ethical hacking in the form of penetration tests and actual attacks on the network, such datasets do not exist.

5.2 Jensen Shannon Divergence

The quality of data generated using WGAN-GP and WGAN-GPMI models were assessed using the JSD metric discussed in Section **??**. Intrusion alerts targeting a total of eight machines were used as discussed in Section **??**. Each case was run 1000 times to determine the standard deviation of JSD. Table **??** shows the JSD values and their standard deviations for each m-tuple and each target from CPTC'17 and CPTC'18. Each table is oriented with the results from WGAN-GP on the left and the results from WGAN-GPMI on the right. By comparing corresponding values for each model and bolding the entry that is closest to 0, it is readily apparent that WGAN-GPMI learns to synthesize data which diverges less from the training dataset.

First, note that both WGAN-GP and WGAN-GPMI achieved reasonably good performance, even when considering the combination of all 4 feature values; When trained on CPTC'17 data, samples from the WGAN-GP model did not exceed 0.28 bits while the WGAN-GPMI samples did not exceed a divergence of 0.26 bits; similarly the CPTC'18 data shows improvements when using the WGAN-GPMI model and a maximum divergence of 0.2501 bits. Secondly, note that for many of the IPs tested the Mutual Information constraint in the WGAN-GPMI model is able to decrease the amount of divergence between the generated and training data samples. This is a result of the model learning a probability distribution which is closer to that of the training data.

It is interesting to note that the effect of the mutual information constraint varies from target IP to target IP. For Target IP 10.0.0.22 from CPTC'18 saw an increase in the divergence when using the WGAN-GPMI model. On the other hand, Target IP 10.0.0.27 saw a large benefit from using the mutual information constraint. Palliating mode dropping is directly related to decreasing histogram divergence in many cases because it distributes output sample entropy across more output values than standard GAN models do when exhibiting mode dropping. Recall that the JSD is a non-linear distance metric where outputting all modes with some probability is more important than outputting the most probable mode with excessively high probability.

Another interesting result of Table **??** is that the JSD is that low divergence in a single feature does not guarantee low divergence in all joint distributions that feature is a part of. Consider the divergence of Timestamp (T) on target IP 10.0.0.22. This feature has a low JSD of 0.0561, potentially leading to the fallacious expectation that any combination with T will also be low. However, When moving to testing 2-tuple combinations such as Timestamp (T) + Source IP (S) the divergence more than doubles.

Overall, the JSD demonstrates that both models learned an approximate representation of each Target IP's alerts. The JSD never exceeded 0.275 bits compared to the upper bound of 1 bit for maximally diverging distributions. Additionally, the models showed consistency in their outputs as the standard deviation of divergences was low; never exceeding 0.0040 bits.

5.3 Dependencies within Alert Features

In order to further validate the output feature distributions learned by each model, feature dependencies in the training data were analyzed and confirmed to exist in the generated alerts. This was accomplished using the normalized Conditional Entropy and Joint Entropy introduced in Section **??**.

Conditional entropy was computed for all target IPs from the CPTC'17 dataset, across all potential $Y|X$ feature-tuples. These results are shown in Table **??**. By computing the Conditional Entropy in Table **??** it is apparent that the WGAN-GPMI model closely imitates the dependencies of the training data. In fact, several of the small valued m-tuples such as A|T, T|D, and D|S,T all have identical Conditional Entropy values to the training data distribution. These values, as well as those within within 10% of the training data entropy value, are bolded for clarity. These cases illustrate

Table 4. Jensen Shannon Divergence for all Feature Combinations: (TOP) CPTC'17 (BOTTOM) CPTC'18

	CPCT'17 Target Machines (IP Addresses)							
	WGAN-GP				WGAN-GPMI			
Features	**10.0.0.100**	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**	**10.0.0.100**	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**
A	**0.0313 ± 0.0012**	0.1378 ± 0.0016	0.0561 ± 0.0007	0.1367 ± 0.0010	0.0404 ± 0.0015	**0.0703 ± 0.0009**	**0.0232 ± 0.0014**	**0.0542 ± 0.0009**
D	**0.0114 ± 0.0013**	0.0271 ± 0.0014	0.0726 ± 0.0009	0.0236 ± 0.0013	0.0239 ± 0.0013	**0.0112 ± 0.0010**	**0.0305 ± 0.0012**	**0.0223 ± 0.0014**
S	**0.0181 ± 0.0016**	0.1352 ± 0.0019	0.0755 ± 0.0010	0.0849 ± 0.0014	0.0301 ± 0.0014	**0.0699 ± 0.0010**	**0.0213 ± 0.0018**	**0.0379 ± 0.0013**
T	**0.0194 ± 0.0011**	0.1403 ± 0.0018	0.0776 ± 0.0010	0.0906 ± 0.0012	0.0204 ± 0.0011	**0.0514 ± 0.0013**	**0.0200 ± 0.0017**	**0.0228 ± 0.0016**
A,T	**0.0884 ± 0.0020**	0.2136 ± 0.0020	0.1019 ± 0.0010	0.1508 ± 0.0016	0.1016 ± 0.0019	**0.1250 ± 0.0014**	**0.0556 ± 0.0022**	**0.1035 ± 0.0020**
A,S	**0.0825 ± 0.0025**	0.1500 ± 0.0020	0.0774 ± 0.0014	0.1592 ± 0.0018	0.0945 ± 0.0020	**0.0965 ± 0.0016**	**0.0317 ± 0.0023**	**0.0948 ± 0.0021**
S,D	0.1178 ± 0.0027	0.2129 ± 0.0019	0.1192 ± 0.0014	0.1976 ± 0.0020	**0.1161 ± 0.0020**	**0.1213 ± 0.0017**	**0.0726 ± 0.0023**	**0.1279 ± 0.0024**
D,T	**0.0544 ± 0.0027**	0.1967 ± 0.0022	0.1119 ± 0.0013	0.1037 ± 0.0021	0.0659 ± 0.0024	**0.1151 ± 0.0015**	**0.0583 ± 0.0023**	**0.0744 ± 0.0019**
S,T	**0.0576 ± 0.0025**	0.1662 ± 0.0024	0.1308 ± 0.0012	0.1179 ± 0.0020	0.0650 ± 0.0026	**0.0884 ± 0.0016**	**0.0851 ± 0.0027**	**0.0544 ± 0.0020**
A,D	0.0949 ± 0.0026	0.2127 ± 0.0020	0.1228 ± 0.0018	0.1737 ± 0.0019	**0.0924 ± 0.0023**	**0.1236 ± 0.0011**	**0.0723 ± 0.0025**	**0.1014 ± 0.0023**
A,S,T	**0.1339 ± 0.0029**	0.2264 ± 0.0021	0.1139 ± 0.0020	0.1778 ± 0.0023	0.1468 ± 0.0027	**0.1482 ± 0.0020**	**0.0663 ± 0.0031**	**0.1361 ± 0.0025**
A,S,D	**0.1954 ± 0.0027**	0.2669 ± 0.0019	0.1713 ± 0.0021	0.2293 ± 0.0024	0.1956 ± 0.0026	**0.1727 ± 0.0027**	**0.1522 ± 0.0029**	**0.2349 ± 0.0024**
A,D,T	0.1573 ± 0.0030	0.2216 ± 0.0023	0.1234 ± 0.0024	0.2084 ± 0.0024	**0.1554 ± 0.0027**	**0.1417 ± 0.0025**	**0.0811 ± 0.0032**	**0.1598 ± 0.0024**
S,D,T	0.1797 ± 0.0029	0.2618 ± 0.0024	0.1722 ± 0.0022	0.2134 ± 0.0026	**0.1740 ± 0.0024**	**0.1717 ± 0.0023**	**0.1499 ± 0.0035**	**0.2259 ± 0.0022**
A,S,D,T	0.2292 ± 0.0032	0.2759 ± 0.0026	0.1729 ± 0.0028	**0.2390 ± 0.0029**	**0.2257 ± 0.0028**	**0.1898 ± 0.0025**	**0.1563 ± 0.0037**	0.2563 ± 0.0030

	CPCT'18 Target Machines (IP Addresses)							
	WGAN-GP				WGAN-GPMI			
Features	**10.0.1.46**	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**	**10.0.1.46**	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**
A	0.0854 ± 0.0018	0.0852 ± 0.0008	0.0301 ± 0.0010	**0.0202 ± 0.0017**	**0.0403 ± 0.0017**	**0.0388 ± 0.0010**	**0.0214 ± 0.0011**	0.0452 ± 0.0012
D	0.0547 ± 0.0016	0.0671 ± 0.0011	0.0483 ± 0.0006	**0.0142 ± 0.0015**	**0.0307 ± 0.0017**	**0.0457 ± 0.0014**	**0.0270 ± 0.0013**	0.0299 ± 0.0012
S	0.0816 ± 0.0017	0.0211 ± 0.0006	0.0292 ± 0.0012	**0.0153 ± 0.0014**	**0.0391 ± 0.0014**	**0.0264 ± 0.0009**	**0.0196 ± 0.0009**	0.0328 ± 0.0017
T	0.0778 ± 0.0014	0.0441 ± 0.0012	0.0183 ± 0.0012	**0.0124 ± 0.0014**	**0.0238 ± 0.0013**	**0.0315 ± 0.0011**	**0.0147 ± 0.0014**	0.0219 ± 0.0015
A,T	0.1528 ± 0.0026	0.1399 ± 0.0020	0.0796 ± 0.0016	**0.0398 ± 0.0023**	**0.0977 ± 0.0023**	**0.0963 ± 0.0021**	**0.0619 ± 0.0019**	0.0819 ± 0.0021
A,S	0.1108 ± 0.0026	0.0944 ± 0.0019	**0.0313 ± 0.0017**	**0.0274 ± 0.0022**	**0.0806 ± 0.0024**	**0.0805 ± 0.0022**	0.0345 ± 0.0025	0.0788 ± 0.0022
S,D	0.1567 ± 0.0026	0.1441 ± 0.0014	0.0523 ± 0.0017	**0.0316 ± 0.0020**	**0.0947 ± 0.0028**	**0.1024 ± 0.0022**	**0.0487 ± 0.0023**	0.0795 ± 0.0024
D,T	0.1427 ± 0.0027	0.0857 ± 0.0019	0.0782 ± 0.0015	**0.0342 ± 0.0020**	**0.0984 ± 0.0026**	**0.0635 ± 0.0029**	**0.0588 ± 0.0024**	0.0650 ± 0.0026
S,T	0.1410 ± 0.0026	0.1490 ± 0.0023	0.0718 ± 0.0017	**0.0309 ± 0.0020**	**0.0975 ± 0.0024**	**0.1111 ± 0.0025**	**0.0609 ± 0.0023**	0.0613 ± 0.0023
A,D	0.1451 ± 0.0023	**0.0774 ± 0.0017**	0.0509 ± 0.0013	**0.0315 ± 0.0023**	**0.0951 ± 0.0021**	0.0842 ± 0.0026	**0.0455 ± 0.0026**	0.0647 ± 0.0025
A,S,T	0.1733 ± 0.0036	0.1489 ± 0.0029	0.0803 ± 0.0024	**0.0448 ± 0.0026**	**0.1405 ± 0.0038**	**0.1339 ± 0.0029**	**0.0714 ± 0.0026**	0.1054 ± 0.0030
A,S,D	0.2594 ± 0.0030	0.2200 ± 0.0023	**0.1116 ± 0.0018**	**0.0549 ± 0.0025**	**0.2122 ± 0.0033**	**0.1875 ± 0.0030**	0.1213 ± 0.0030	0.1106 ± 0.0031
A,D,T	0.1756 ± 0.0032	0.1528 ± 0.0025	**0.0534 ± 0.0017**	**0.0368 ± 0.0025**	**0.1412 ± 0.0035**	**0.1366 ± 0.0025**	0.0585 ± 0.0031	0.0997 ± 0.0029
S,D,T	0.2503 ± 0.0038	0.1895 ± 0.0028	**0.1097 ± 0.0023**	**0.0563 ± 0.0026**	**0.2275 ± 0.0036**	**0.1715 ± 0.0027**	0.1184 ± 0.0029	0.1062 ± 0.0028
A,S,D,T	0.2724 ± 0.0040	0.2270 ± 0.0032	**0.1120 ± 0.0024**	**0.0593 ± 0.0027**	**0.2501 ± 0.0039**	**0.2154 ± 0.0033**	0.1270 ± 0.0030	0.1264 ± 0.0029

the ability of WGAN-GPMI to learn feature important dependencies between individual features despite never receiving specific reinforcement of these dependencies. Additionally it is significant that data generative models learn dependencies such as these as it reflects the attacker behaviors seen in the training data.

Table 5. CPTC'17: Normalized Conditional Entropy Values for all target IPs: WGAN-GPMI Result

	Target Machine IP Address							
	Training Data Results				**Generated Results**			
Features	**10.0.0.100**	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**	**10.0.0.100**	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**
A|T	**0.244**	**0.238**	**0.153**	**0.333**	**0.244**	**0.238**	**0.153**	**0.334**
T|S	**0.593**	**0.463**	**0.515**	**0.695**	**0.593**	**0.463**	**0.516**	**0.695**
T|A	**0.330**	**0.339**	**0.695**	**0.246**	**0.330**	**0.339**	**0.695**	**0.246**
S|T	**0.262**	0.252	**0.263**	**0.405**	**0.263**	0.186	**0.252**	**0.406**
S|A	0.800	0.752	0.831	0.711	0.229	0.239	0.526	0.222
D|S	0.346	0.445	0.253	0.278	0.509	0.385	0.207	0.558
A|D	0.080	0.222	0.070	0.288	0.149	0.026	0.007	0.097
T|D	**0.479**	**0.346**	**0.655**	**0.383**	**0.479**	**0.346**	**0.655**	**0.383**
D|T	**0.287**	**0.234**	**0.152**	**0.379**	**0.287**	**0.234**	**0.152**	**0.379**
A|S	0.346	**0.385**	0.271	0.475	0.403	**0.376**	0.214	0.543
S|D	0.822	0.779	0.856	0.785	0.436	0.260	0.474	0.301
D|A	0.006	0.246	0.006	0.016	0.055	0.009	0.048	0.000
S|A,D	0.799	0.747	0.829	0.705	0.228	0.226	0.474	0.222
D|S,T	**0.171**	**0.118**	**0.013**	**0.149**	**0.171**	**0.118**	**0.013**	**0.149**
S|D,T	**0.107**	**0.025**	**0.101**	**0.024**	**0.107**	**0.025**	**0.101**	**0.024**
T|A,D	**0.316**	**0.335**	**0.650**	**0.246**	**0.316**	**0.335**	**0.650**	**0.246**
A|S,D	0.069	0.206	0.056	0.245	0.018	0.003	0.007	0.055
A|S,T	0.131	**0.117**	**0.013**	**0.130**	0.112	**0.117**	**0.013**	**0.131**
A|D,T	**0.038**	**0.018**	**0.001**	**0.004**	**0.038**	**0.018**	**0.001**	**0.004**
D|A,S	0.005	0.243	0.005	0.012	0.054	0.000	0.000	0.000
T|S,D	0.393	0.340	0.587	0.348	0.176	0.144	0.334	0.170
D|A,T	0.004	0.238	0.003	0.007	0.044	0.006	0.000	0.000
S|A,T	0.211	0.178	**0.100**	0.228	0.055	0.012	**0.100**	0.019
T|A,S	0.365	0.312	0.561	0.302	0.170	0.144	0.328	0.089
A|S,D,T	0.041	0.172	0.028	0.195	0.005	0.003	0.000	0.001
D|A,S,T	0.002	0.232	0.002	0.004	0.044	0.000	0.000	0.000
T|A,D,S	0.209	0.167	0.498	0.222	0.157	0.144	0.328	0.089
S|A,T,D	0.362	0.302	0.558	0.294	0.055	0.004	0.100	0.019

The Conditional Entropy for samples from CPTC'18 were also computed as shown in Table **??**. The same criteria for bolding values within 10% of the training data values was applied. Note that there are far fewer samples which meet this criteria in CPTC'18 data. One explanation for this is that the constraints which inhibit the WGAN-GPMI model from output mode collapse actually prevent the network from modeling the true distribution of the training data. Several $Y|X_1, X_2, ..., X_m$ in the training data have a Conditional Entropy below the three decimal places used. These samples include $D|A$, $D|A, S$, $D|A, T$, and $D|A, S, T$ for all but Target 10.0.1.46. Additionally, combinations such as $A|S, D$, $A|D, T$, and $A|D, S, T$ also exhibit very low entropy across each of the targets. These samples illustrate the tight coupling between Alert Signature and the Destination Service being targeted by the attackers in CPTC'18.

In order to better understand the relationship between randomness in each feature distribution and the model's ability to synthesize realistic data, the JSD was plotted against the Joint Entropy

Table 6. CPTC'18: Normalized Conditional Entropy Values for all target IPs: WGAN-GPMI Result

	Target Machine IP Address							
	training data Results				Generated Results			
Features	**10.0.1.46**	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**	**10.0.1.46**	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**
A|T	0.287	0.175	0.265	0.151	0.412	0.371	0.304	0.209
T|S	0.303	0.355	**0.321**	0.231	0.479	0.565	**0.343**	0.297
T|A	0.317	0.395	0.169	0.113	0.518	0.628	0.196	0.254
S|T	0.334	0.246	**0.367**	0.171	0.495	0.399	**0.375**	0.225
S|A	0.272	0.350	0.223	0.135	0.452	0.529	0.256	0.252
D|S	0.270	0.234	0.249	0.223	0.395	0.533	0.317	0.307
A|D	0.091	0.274	0.005	0.069	0.202	0.460	0.069	0.157
T|D	0.346	0.681	**0.176**	0.216	0.525	0.810	**0.195**	0.281
D|T	0.307	0.153	0.283	0.164	0.448	0.445	0.307	0.230
A|S	0.221	0.235	0.238	0.205	0.337	0.443	0.317	0.271
S|D	0.336	0.618	0.223	0.238	0.486	0.690	0.253	0.293
D|A	0.062	0.000	0.000	0.000	0.212	0.257	0.073	0.147
S|A,D	0.236	0.350	**0.223**	0.135	0.411	0.522	**0.243**	0.223
D|S,T	0.043	0.063	0.013	0.076	0.305	0.420	0.155	0.183
S|D,T	0.070	0.160	0.115	0.049	0.358	0.385	0.223	0.173
T|A,D	0.248	0.394	**0.169**	0.113	0.461	0.618	**0.182**	0.215
A|S,D	0.002	0.040	0.005	0.001	0.130	0.280	0.059	0.099
A|S,T	0.040	0.054	0.012	0.070	0.257	0.313	0.153	0.158
A|D,T	0.0121	0.048	0.000	0.000	0.146	0.255	0.057	0.010
D|A,S	0.026	0.000	0.000	0.000	0.169	0.244	0.059	0.121
T|S,D	0.052	0.174	0.036	0.027	0.383	0.505	0.162	0.173
D|A,T	0.000	0.000	0.000	0.000	0.158	0.238	0.061	0.107
S|A,T	0.058	0.107	0.115	0.049	0.334	0.344	0.225	0.164
T|A,S	0.080	0.125	0.030	0.027	0.387	0.443	0.160	0.175
A|S,D,T	0.001	0.002	0.000	0.000	0.103	0.205	0.049	0.077
D|A,S,T	0.000	0.000	0.000	0.000	0.137	0.227	0.050	0.095
T|A,S,D	0.051	0.125	0.030	0.027	0.352	0.435	0.150	0.149
S|A,D,T	0.057	0.107	0.115	0.049	0.313	0.339	0.214	0.150

for each Target IP in Fig. **??**. As the number of features in each m-tuple increased, so did the Joint Entropy of the resultant joint distribution. This is due to Joint Entropy being additive when not considering feature dependencies. As the Joint Entropy for each distribution rose, so did the JSD between the training data and generated results. This positive correlation occurs as there is more information for the GAN models to learn in order to accurately synthesize alerts from distributions with a higher amount of entropy. Despite this, the JSD never exceeds a value of 0.275 bits even as the Joint Entropy for each Target reaches it's maximum when considering 4-tuple combinations.

5.4 Output Modes and Attack Stages

Finally, to assess output modes captured by each model, we 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 training dataset versus those existing 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 training data set. We refer to these values as *Dropped*, *Covered*, and *Noisy* respectively.

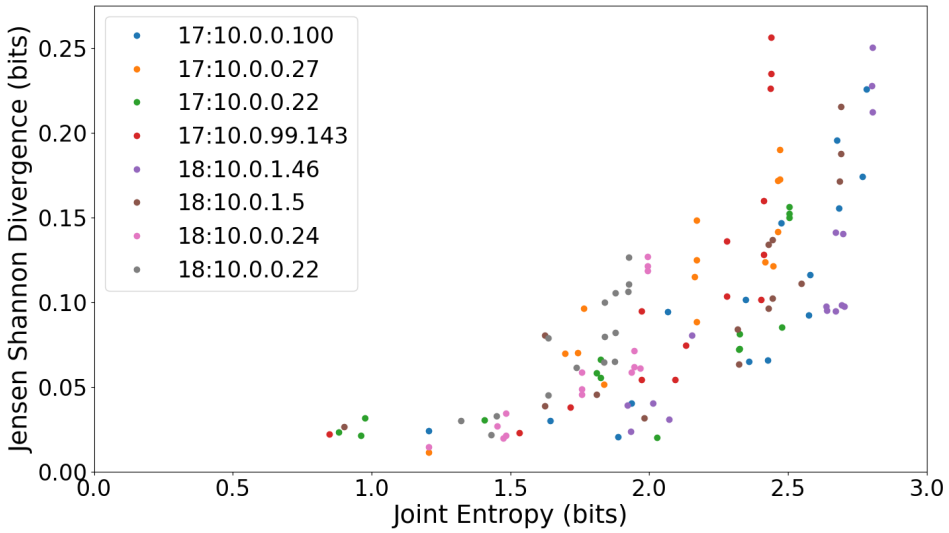

Fig. 3. CPTC'17 and CPTC'18 Targets: Jensen Shannon Divergence is positively correlated with Joint Entropy of the feature distribution.

The top portion of Table **??** shows the number of Dropped and Noisy outputs for each GAN model when trained on CPTC'17 data. The bottom four rows show the number of alerts, the number of unique 4-feature combinations, % of output modes dropped, and ratio of noisy outputs to outputs within the domain of the training data. Note that this table shows the direct benefit of mutual information maximization, as the number of output modes missed by the model decreases across the board for the WGAN-GPMI model. Some of the target IPs learn more output modes than others when moving to the WGAN-GPMI model; 10.0.0.100, as well as 10.0.0.22, halve the number of output modes dropped. On the other hand, 10.0.0.27 and 10.0.99.143 only see a minor improvement when adding in the mutual information constraint. These IPs instead see a large decrease in the number of noisy output modes when used for training the WGAN-GPMI model instead of the WGAN-GP model. It's also important to note that these output modes aren't inherently wrong since the individual feature value do exist in the training dataset. However, there should be no gradient feedback to encourage the generation of these combinations of feature values since they don't occur in the training dataset for these targets.

The bottom portion of Table **??** shows the equivalent information for each generative model trained on the CPTC'18 data. Interestingly for all but target IP 10.0.0.22 the inverse relationship between Dropped and Noisy output modes holds true. Target IP 10.0.1.5 sees no change in the number of output modes dropped but does see an increase in noisy outputs. Despite these two IPs, Table **??** still showed a decrease in the divergence between the training data and generated histograms for the WGAN-GPMI model in 3 out of the 4 Target IPs from CPTC'18. This points to WGAN-GPMI model learning to output alerts with probabilities much closer to those of the training data distribution.

One method to demonstrate this would be to view the percent of alerts generated that are within the training data compared to the percentage of alerts generated that are not. Viewing the alert distribution at this macro scale could be thought of as a means to view the power of the noisy alerts. Even if the generator synthesizes a large number of alert modes which do not exist in the training data distribution those modes would have a low power if they occur rarely.

Table 7. Output Modes Dropped and Noisy Outputs: (TOP) CPTC'17 (BOTTOM) CPTC'18

	Target Machine IP Address									
		WGAN-GP				WGAN-GPMI				
Features	**10.0.0.100**	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**	**10.0.0.100**	**10.0.0.27**	**10.0.0.22**	**10.0.99.143**		
Noise	168	235	76	321	213	97	107	98		
Dropped	21	15	9	10	10	14	4	8		
# Alerts	3388	3166	2974	2182						
# Unique Modes	32	27	22	27						
% Modes Dropped	0.6563	0.5556	0.4091	0.3704	0.3125	0.5185	0.1818	0.2963		
Noise Ratio	5.250	8.704	3.455	11.889	6.656	3.593	4.864	3.630		
		WGAN-GP				WGAN-GPMI				
Features	**10.0.1.46**	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**	**10.0.1.46**	**10.0.1.5**	**10.0.0.24**	**10.0.0.22**		
Noise	138	59	69	69	365	200	147	150		
Dropped	7	14	10	8	5	14	5	13		
# Alerts	7475	8695	9861	7996						
# Unique Modes	33	31	22	29						
% Modes Dropped	0.2121	0.4516	0.4545	0.2759	0.1515	0.4516	0.2272	0.4483		
Noise Ratio	4.182	1.903	3.136	2.379	11.060	6.452	6.682	5.172		

Figure ?? shows these output mode distributions as a series of bar graphs for two target IPs from both CPTC'17 and CPTC'18. 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. Across all of the distributions there are significantly more noisy alert modes than those which occur in the training

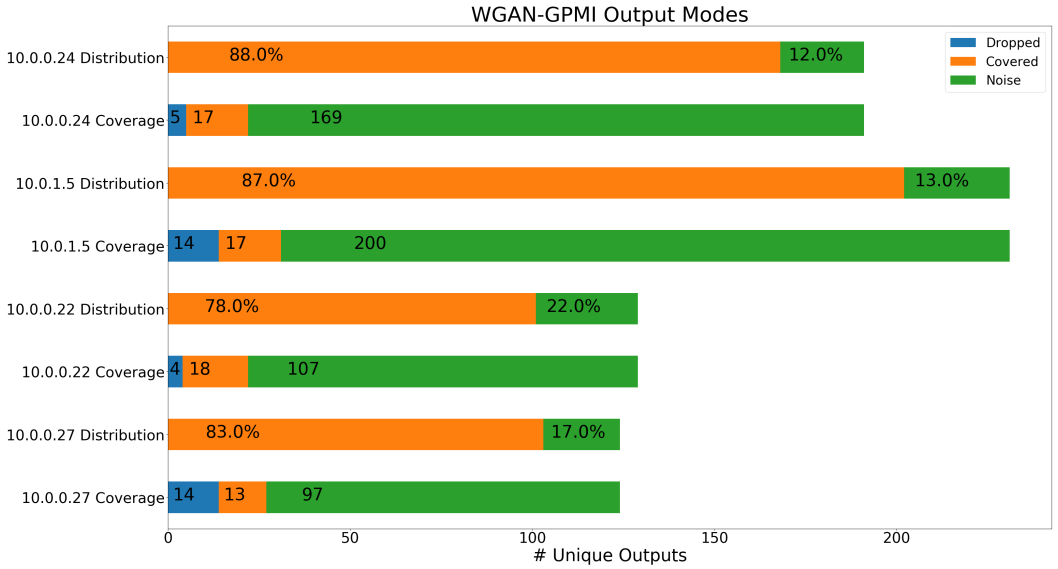

Fig. 4. CPTC'17 Target IPs 10.0.0.22 and 10.0.0.23: The WGAN-GPMI model features less mode dropping than the WGAN-GP model, however the amount of probability mass assigned to noisy samples also increases.

data distribution. However, these noisy modes have a low power as they occur far less often than the modes which do exist in the training data distribution. In particular, Target IP's 10.0.0.24 and 10.0.1.5 from CPTC'18 generate noisy alerts no more than 15% of the time despite having 1.5 times more noisy output modes than the IP's from CPTC'17. Additionally, these two targets have significantly more unique outputs than the samples from CPTC'17. This is predominantly due to the higher number of noisy output modes. For the two targets from CPTC'17 the noisy alerts exhibit more power as they occur with a higher probability of 17%-22%.

A detailed look at the results reveal differences in theses two target IPs. Target 10.0.0.22 shows superior coverage with only 4 output modes being dropped while adding a larger number of novel modes which occur with low probability. On the other hand, Target 10.0.0.27 has less noisy modes and alerts but still drops 14 modes from the training 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 training data distribution. Note that these results only represent a subset of the IP addresses tested due to space limitations; the observations above hold for each of the IPs tested.

Finally, a means to identify the type of behavior associated with the additional output modes captured would provide contextual information to what type of network behaviors are most recoverable from data driven models such as GANs. To accomplish this Alert Signatures were mapped to attack stages such as Host, Service, and Vulnerability Discovery. Figure **??** shows the attack stage coverage within the training data as well as those generated from WGAN-GP and WGAN-GPMI for the CPTC'17 and CPTC'18 datasets respectively. Note that the WGAN-GPMI model shows attack stage behavior with probabilities far closer to the training data distribution, exhibiting improvement over the WGAN-GP case.

Specifically for CPTC'17, the WGAN-GPMI model synthesized alerts pertaining to the Host Discovery stage with probability only 1% off from the training data distribution. Meanwhile, the

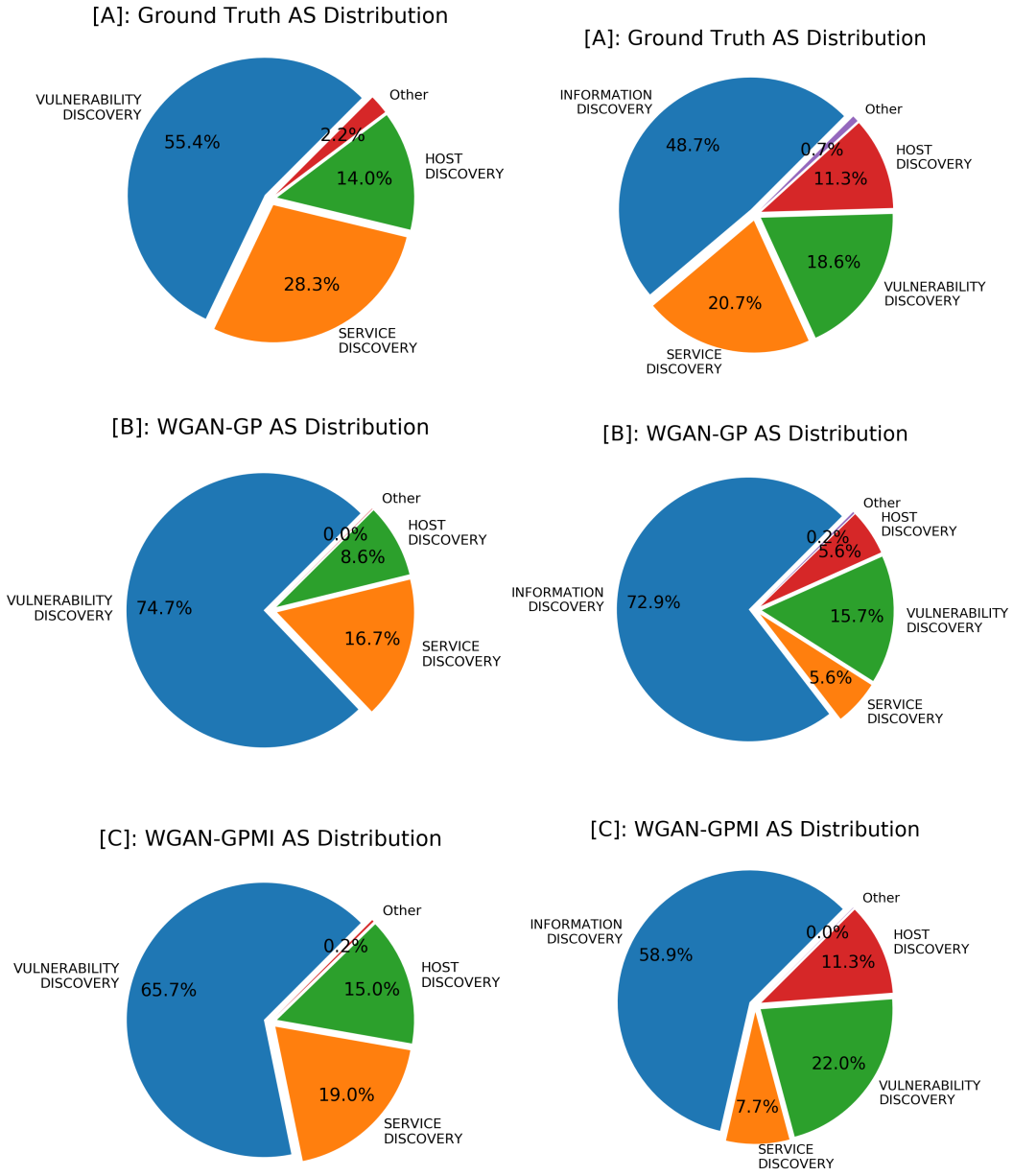

Fig. 5. (Left): Distribution of Attack Stages (AS) on target IP 10.0.99.143 from CPTC'17. (Right): Distribution of Attack Stages (AS) on target IP 10.0.1.46 from CPTC'18. Note that the WGAN-GPMI model results [C] have a much closer probability distribution to the training data [A] then the WGAN-GP Model [B].

standard WGAN-GP model could not capture this output mode with probability greater than 8.6%, leaving a large gap in the generated data sample. In CPTC'18 alerts exhibiting this behavior were synthesized with the exact same probability as the training data, 11.3%. Finally, in both CPTC datasets the WGAN-GP model generated alerts pertaining to Vulnerability Discovery around 20%

more frequently than they occurred in the training data. The WGAN-GPMI model cut this down by 10%, contributing to the overall probability distribution diverging from the training data less.

6 CONCLUDING REMARKS

This research showed the promise of using unsupervised Deep Learning models, GANs, to synthesize target based cyber-alert data from known malicious data. Usage of the WGAN-GPMI model is shown to lower the divergence of the generated distribution from the training data distribution, better maintain intra-alert feature dependencies, and generate alerts pertaining to real attacker behavior far more often than WGAN-GP based models. Future works using synthetically generated alerts could pursue multiple directions. Synthesizing alerts which emulate historical attacker behavior could be used to increase the utility and responsiveness of network intrusion prevention systems by updating detection rule-sets automatically and continuously on a per system basis; all based off the traffic previously seen. Furthermore, these models offer the potential to augment limited datasets for usage in other Machine Learning systems. Finally, future experimentation with varying GAN models is of great interest. Through the usage of LSTM or CNN architectures, temporal dependencies in network behavior for a given machine may be exploited to build complex multistep attacker models. And through the usage of Transfer Learning, models trained on a combination of malicious data and network configuration parameters could be applied to another network to discover possible attack vectors.