Cyber Creative Generative Adversarial Network for Novel Malicious Packets

John Pavlik

Nathaniel D. Bastian
Army Cyber Institute, U.S. Military Academy, nathaniel.bastian@westpoint.edu

Follow this and additional works at: https://digitalcommons.usmalibrary.org/aci_ja

Part of the Artificial Intelligence and Robotics Commons, Data Science Commons, and the Information Security Commons

Recommended Citation

This Conference Proceeding is brought to you for free and open access by the Army Cyber Institute at USMA Digital Commons. It has been accepted for inclusion in ACI Journal Articles by an authorized administrator of USMA Digital Commons. For more information, please contact dcadmin@usmalibrary.org.
Cyber creative GAN for novel malicious packets

John Pavlik, Nathaniel Bastian
Cyber Creative GAN for Novel Malicious Packets

John A. Pavlik and Nathaniel D. Bastian

Army Cyber Institute, United States Military Academy, West Point, NY, USA

ABSTRACT

Machine learning (ML) requires both quantity and variety of examples in order to learn generalizable patterns. In cybersecurity, labeling network packets is a tedious and difficult task. This leads to insufficient labeled datasets of network packets for training ML-based Network Intrusion Detection Systems (NIDS) to detect malicious intrusions. Furthermore, benign network traffic and malicious cyber attacks are always evolving and changing, meaning that the existing datasets quickly become obsolete. We investigate generative ML modeling for network packet synthetic data generation/augmentation to improve NIDS detection of novel, but similar, cyber attacks by generating well-labeled synthetic network traffic. We develop a Cyber Creative Generative Adversarial Network (CCGAN), inspired by previous generative modeling to create new art styles from existing art images, trained on existing NIDS datasets in order to generate new synthetic network packets. The goal is to create network packet payloads that appear malicious but from different distributions than the original cyber attack classes. We use these new synthetic malicious payloads to augment the training of a ML-based NIDS to evaluate whether it is better at correctly identifying whole classes of real malicious packet payloads that were held-out during classifier training. Results show that data augmentation from CCGAN can increase a NIDS baseline accuracy on a novel malicious class from 79% to 97% with a minimal degradation in accuracy on benign classes (98.9% to 98.7%).

Keywords: Cybersecurity, Generative Machine Learning, Synthetic Data Generation, Data Augmentation

1. INTRODUCTION

The dynamic and adversarial nature of the cyberspace domain causes machine learning (ML) based methods for network intrusion detection to lack resilience due to the dearth of well-marked and fresh datasets. Enabling ML to evolve ahead of our adversaries’ dynamic behavior is recognized as an “as of yet” unattainable requirement. Generative machine learning (GML) has the power to enable data augmentation to enlarge ML training sets to improve the ability of trained classifiers to detect out-of-distribution (OOD) inputs, but this area still needs to be fully leveraged within the cyberspace domain and its evolutionary properties need to be explored. GML has demonstrated its ability to generate very realistic data in the computer vision and audio domains, for example, yet cyber GML needs to be further understood and used to enable cyber ML evolution to enable network intrusion detection systems (NIDS) to detect both known and novel (zero-days) malicious intrusions.

NIDS can be divided into signature-based systems and anomaly-based systems. Signature systems look for known attack patterns, while anomaly systems identify normal activity and flag anything outside of the normal. Signature systems are limited by the availability of signatures, and will fail to detect new and novel attacks. Anomaly systems are subject to many false positive detection events when the distribution of normal behavior shifts. Many NIDS focus on network flows using tools like Zeek Network Security Monitor that produce information about an entire stream of packets. However, network flows can only be analyzed after the session has been closed. A network monitoring system that relies solely on flow information will be too late to identify and interrupt an in-progress attack. As such, a system needs to examine individual packets in order to identify cyber attacks in (near) real-time. Common packets like TCP and UDP have each packet divided into header information that describes the packet and an encapsulated payload of data being transmitted. Our paper makes the following contributions:

Further author information: (Send correspondence to J.A.P)
J.A.P.: E-mail: john.pavlik@westpoint.edu
N.D.B.: E-mail: nathaniel.bastian@westpoint.edu
• We present a novel GML model architecture, Cyber Creative Generative Adversarial Network (CCGAN), that generates synthetic malicious network traffic payloads that are related to the training dataset without matching any of the training classes (i.e., OOD attack classes).

• We conduct extensive computational experimentation using CCGAN and two different NIDS datasets to evaluate the impact, in terms of classification robustness and accuracy, of using the synthetically generated malicious payloads to augment the training of ML-based NIDS.

• We demonstrate that CCGAN-based data augmentation can improve network intrusion detection performance for both known and OOD malicious intrusions, thereby helping enable cyber ML evolution.

This paper is organized as follows. Section 2 describes relevant existing works, while Section 3 introduces the CCGAN model architecture and design. Section 4 describes the experimental setup for testing CCGAN. Section 5 details the results and analysis of the experiments, while Section 6 presents our conclusions and future work.

2. RELATED WORKS

There are many common approaches for detection and filtering benign and malicious network traffic, such as classical statistics, expert and rule-based systems, ML classification, and anomaly detection. CCGAN improves ML classification of network traffic through the use of data augmentation, leading to assurance gains.

2.1 Generative Machine Learning

A Generative Adversarial Network (GAN)12,13 differs from many other ML model architectures in that it is setup as a two-player adversarial pair of networks where each network trains the other as a minimax game. The two networks in a GAN are the discriminator and the generator. The generator samples from a latent space and produces synthetic data. The discriminator takes both real and synthetic data and learns to classify real versus synthetic. The generator learns to “fool” the discriminator into misclassifying the synthetic samples as real. The optimum result is that the generator learns to produce samples which are indistinguishable from real samples, and therefore has captured the original distribution.

Our work builds upon the Creative Adversarial Network (CAN) developed by Elgammal et al.,14 which is based upon the idea of creating new styles of art through the use of a GAN trained on known art styles. The discriminator is similar to a conditional GAN, where the discriminator learns to identify real from generated images and also learns to identify their style. The generator is where the CAN differs from a regular conditional GAN. The generator learns to make images which are classified as real, but also attempts to maximize style ambiguity. Style ambiguity is when the discriminator produces a uniform distribution of style classification.

2.2 Data Augmentation

Data augmentation is a technique for improving the quality of a dataset through the use of adding new features or examples.15 Liu et al.16 examine the use of a variational autoencoder (VAE) for data augmentation of a NIDS. Chaâ and Bastian17 explore how Markov Chain Monte Carlo (MCMC) methods can be used for realistic synthetic data generation and compares it to several existing GML techniques. The performance of MCMC is compared to GAN and VAE methods to estimate the joint probability distribution of NIDS data. Chaâ and Bastian18 extend this work by testing the capabilities of a conditional tabular GAN and a tabular VAE at generating synthetic network traffic to augment training of ML-based NIDS. They find that roughly 85% synthetic to 15% real data is needed to train high-performing ML-based NIDS classifiers.

2.3 Raw Network Traffic Detection

While most NIDS focus on network flows or packet headers, there are some areas of research focused on NIDS that use the payload of a packet in order to identify malicious behavior. This has some weaknesses, for example it can be defeated by encryption schemes, but when it works it allows for immediate and accurate identification of the type of malicious behavior without waiting for a complete flow to be produced. De Lucia et al.19 demonstrates packet binary malicious/benign classification using only the payload of the packets, without any features from the header or network flow, in a 1D-Convolutional Neural Network (1D-CNN). Bierbrauer et al.20 extend this work by using transfer learning for raw traffic detection. They test the accuracy of different ML models in a normal environment, then transfer part of each model to a low-power edge device and perform limited retraining.
2.4 Adversarial Machine Learning

Adversarial ML is the approach of using ML techniques against an adversary targeting the adversary’s own ML models. Examples include data poisoning against the data used for training a ML model to deliberately introduce weaknesses, as well as evasion attacks that exploit vulnerabilities in models to allow attacks to be misclassified. CCGAN could be viewed as using an adversarial model reinforcing a friendly model against attacks by finding them before they are used. Talty et al. demonstrate poisoning datasets in order to create evasion attacks that can bypass an ML-based NIDS that was trained on the poisoned dataset. Alhajjar et al. focus on crafting evasion attacks with the use of particle swarm optimization, genetic algorithms, and GANs. They demonstrate the importance of model robustness by examining the interactions of the adversarial example generators with different types of underlying classification algorithms. Schneider et al. extends this work to further evaluate model robustness in ML models used to create ML-based NIDS. They look at possible vulnerabilities and methods of defense against adversarial sample generation.

Chalé et al. create adversarial examples of raw packets using a genetic algorithm inspired meta-heuristic algorithm to bypass an ML-based NIDS that classifies based on packet payloads instead of network flow features. This is significant because packet payloads are significantly easier for a malicious actor to manipulate than the network flow features. Unlike in computer vision where many pixels can subtly manipulated without impacting the overall image, each byte has a specific meaning in a malicious payload and must be changed carefully to preserve the impact of the payload. Hore et al. develop Deep PackGen, which employs deep reinforcement learning to generate adversarial packets, which takes raw malicious network packets as inputs and systematically makes optimal perturbations on them, camouflaging them as benign packets while still maintaining functionality.

3. METHODOLOGY

CCGAN is based on CAN but is modified to support network traffic payloads (1-dimensional sequence of bytes) instead of art (2-dimensional array of pixels). The goal of CCGAN is to produce synthetic malicious network payloads that augment the training of a NIDS that uses ML. Unlike a regular GAN, CCGAN attempts to be creative by producing packets that are similar to the malicious packets in the training set but different enough that they are not classified as any of the known classes of packets that exist in the training set. This can allow a NIDS that has been augmented with data from CCGAN to identify new classes of malicious packets that did not exist in the training data. Figure 1 shows the architecture of CCGAN. It consists of two different networks, the generator and discriminator.

The discriminator of CCGAN receives a payload and makes two predictions, how real does the packet look and which malicious class is the payload from. 50% real and 50% generated payloads use binary cross entropy loss to train the discriminator to recognize real from generated payloads. For the real packets only, a cross-entropy loss against a one-hot encoding of the correct class is also added to the loss function. The discriminator minimizes its loss when it correctly predicts whether all packets are real or generated and when it correctly predicts the correct class for all real packets.

CCGAN has a generator loss function that combines two conflicting goals. The CCGAN generator tries to produce realistic data while also not matching any of the known classes. The generator receives a latent space vector of 100 random standard normal values and produces a 1500 byte payload. The loss function is the sum of two different loss values. The first is a binary cross entropy loss that is minimized when the discriminator incorrectly identifies generated packets as real. The second is a binary cross entropy loss against a zero vector, which minimizes the loss when all the classes are considered equally likely by the discriminator.

Both the generator and discriminator use very similar 1D-CNN architectures, as shown in Figure 2. The only difference is the input and output sizes. The generator’s input is a 1D vector of 100 random normal values representing a latent space and outputs a vector of 1500 values in the range of 0-255. The discriminator takes a 1500 byte vector representing a packet payload as input and outputs a vector equal to the number of classes plus one extra value for real or generated classification. The output values of the generator are activated with a tanh function. The output values of the discriminator are activated with a sigmoid function.

The common architecture of the generator and discriminator is a series of blocks containing a 1D convolutional layer, batch normalization layer, and activation (ReLU or tanh) layer. The number of channels in the
convolutional layer increases as the depth increases. The total number of convolutional blocks varies in the experiments, taking the first 7, 14, or 28 1D-CNN layers from Figure 2. All of the convolutional layers use kernel size set to three and padding set to one. After the convolutional blocks, there is a single 1D maxpool layer with kernel size and stride both equal to two to reduce the dimensionality by half before passing the data to a series of three fully connected blocks. Each fully connected block consists of a 50% dropout layer, a fully connected layer that outputs 4096 values, and an activation layer. The final layer of the generator outputs 1500 floating point values in the range of \([-1, 1]\) by applying a tanh activation, then rescales the output to \([0,255]\) and applies a floor function to convert the values to integers. The final layer of the discriminator is broken into the real/generated prediction which is activated with a sigmoid and a class prediction activated with a softmax.

4. COMPUTATIONAL EXPERIMENTATION

4.1 Data

We use two datasets to test CCGAN, UNSW NB-15\(^{29,30}\) and CIC-IDS 2017\(^{31,32}\). Each dataset provides labeled malicious and benign network flows with associated packet capture information. UNSW has one benign class (‘normal’) and nine malicious classes: fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode and worms. CIC-IDS 2017 has one benign class and 14 malicious classes: Bot, DDoS, DoS Hulk, DoS GoldenEye, DoS Slowhttptest, DoS slowloris, FTP-Patator, Infiltration, Heartbleed, PortScan, SSH-Patator, Web Attack - Brute Force, Web Attack - Sql Injection, and Web Attack - XSS.

Both datasets provide labeled flow data and unlabeled packet capture (PCAP) data, so Payload Byte\(^{33}\) is used to produce labeled payloads from both datasets by combining flow information and timestamps to find the most likely label match. The total number of payloads by class in both datasets are listed in Table 1. For CIC-IDS2017, the DoS slowloris and Web Attack - XSS classes are held out of training and only used in the final tests to provide novel distributions of real data for classification.

The holdout classes are reserved for augmentation test, and both datasets excluding the holdout classes are split into 64% CCGAN training, 16% CCGAN validation, 16% augmentation training and 4% augmentation.
test datasets. In total, that produces 8x datasets, 4x for UNSW NB-15 and 4x for CIC-IDS2017. The CCGAN is trained exclusively on malicious payloads, so all benign or normal payloads are dropped from the CCGAN training and validation datasets.

The augmentation training dataset is balanced to be 50% benign/normal and 50% all others by under-sampling the larger benign/normal classes and keeping all malicious classes. All payloads are resized to be exactly 1500 bytes by either padding with zeros or removing bytes from the end of the payload.

<table>
<thead>
<tr>
<th>CIC-IDS 2017</th>
<th>UNSW-NB15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>45,159,909</td>
</tr>
<tr>
<td>DoS Hulk</td>
<td>2,219,061</td>
</tr>
<tr>
<td>DDoS</td>
<td>618,544</td>
</tr>
<tr>
<td>SSH-Patator</td>
<td>181,147</td>
</tr>
<tr>
<td>FTP-Patator</td>
<td>110,636</td>
</tr>
<tr>
<td>Infiltration</td>
<td>41,725</td>
</tr>
<tr>
<td>Heartbleed</td>
<td>41,283</td>
</tr>
<tr>
<td>DoS GoldenEye</td>
<td>34,293</td>
</tr>
<tr>
<td>Web Attack – Brute Force</td>
<td>28,920</td>
</tr>
<tr>
<td>DoS slowloris</td>
<td>20,877</td>
</tr>
<tr>
<td>DoS Slowhttptest</td>
<td>9,778</td>
</tr>
<tr>
<td>Web Attack – XSS</td>
<td>6,767</td>
</tr>
<tr>
<td>Bot</td>
<td>5,143</td>
</tr>
<tr>
<td>PortScan</td>
<td>946</td>
</tr>
<tr>
<td>Web Attack – SQL Injection</td>
<td>45</td>
</tr>
</tbody>
</table>

Table 1: CIC-IDS2017 and UNSW NB-15 Payload Counts by Category (with Holdout Classes Indicated)
4.2 Test Design

The primary purpose of CCGAN is to provide synthetic data to augment a NIDS, so this is the primary test we execute. We setup a variety of different configurations for the CCGAN and trained it for 10 epochs each time to see if any of the hyperparameter choices affect the quality of data produced by CCGAN.

The configuration choices we looked at are the dataset, activation function, number of layers, and optimizer. The two datasets are UNSW-NB15 and CIC-IDS 2017. The activation function is either ReLU or tanh and affects every convolutional and linear layer of both the discriminator and generator networks. The number of layers is the first 7, 14, or 28 layers from Figure 2. The optimizer is either Adam or RMSProp. After training, each variation produces 100,000 random packets. We setup a NIDS based on the same architecture as the discriminator from CCGAN, using 14 layers of 1D-CNN, with a single output value to determine benign or malicious for each packet payload. The augmentation experiment combines the UNSW-NB15 and CIC-IDS 2017 datasets into a single dataset.

The baseline experiment is the NIDS trained on 100% real data, about 920,000 network packets, with 50% benign and 50% malicious packets of all classes excluding the holdout classes. The augmentation experiments each add 100,000 synthetic packets labeled as malicious generated by CCGAN to the NIDS training and then over-sample the benign packets to make the dataset balanced again. Then, the NIDS is tested against 100% real data, about 2,760,000 packets, with 95.6% benign packets and 4.4% malicious packets. The test set is larger than the train set because the benign packets have not been down-sampled to make a balanced set.

5. RESULTS AND DISCUSSION

Unlike images, network traffic payloads make poor visualizations, but the following string is a representative sample of the kind of output produced by CCGAN after converting to ASCII:

```
?}{{?"\?t\?y\?p\?"??????}{t?????yv??u??z??h??y"u??}u?w??v???\?w???|????z|zz|t
"\?}???y??{?x"?y"z|??????}??????v??}?????u???}wp???w|?r???x|??p?x?t??u??w??
v?u??{y?????u}|"w????x????v????"n????"??????{u\?y\\?v|?\?}????{y???y|z
???????{q??r??}{n??y??{y?z????z????um\?t|y?????????}??????????????????????{t??
txx~?x??"v"~?x}ur\?u\?x}xw?w|?????q?v? ??????????????z|?????u|"s |r |"z
|???????{p\?yv\?x\?}pxv}???z\?xr\?z\?x\?zu??????"??|y?????|x???????|yzu????w
??????{s\?y|????x"??}wo\?tx???"x??z\?u\{z\}?www\?w|p{???????{|"z |???|zz????p
????}???|y\?v\?yw|?\?x?????ax?|w???x|?}\?????"u?????k????}|???|y\?w|www
{???}\????{!!}}zzs?z?????r????z}\?n??y}\{v?????????\{x"uw??????????x"????
y\?x"?w|\}?{t\?yy\?nn}|"????|????|yz?"u|{'z|'}z????"u\?u\?u\?|t}\{q\\?{r????z??y
|?????}|?????xw???????z????????v\?y?????y????y???????????????????????????{|}xpt\?t{|www|}\}v
|""""x??????|y????q\?z}\???????????|?\?x\?t??????v\"\|\?q\?
\?y????\?s?\?y?"v"~\?x|}\? wvv?uq\?y\?q\?{zz?{"j "??xy"??"\"}|\"{urt?"|}\"}|}z\?ww|\?{|}???\?x\?uu????s\?w
?????x?z\?|?t??"yy\?w\?\{zz\?v\?w|wv"??|"v?y|x|v?z???????\}???x????{\{t\?wy|\?z??
\?u?"y\?g?????o\{w?zs\?{?x?????u\?uto\?w\}w|u\?ow\?o\?u\?y\?v\?yt\?{x?????p\?}\m
|{???}\??????p\"??}\}xr\?{u?????wt\?zs\?uxv"??|q\}\\|\{\|\}x????}\?ur\??\??\??\??\??\??\?
zy?"t?????|z??????y?????????|z|z???????????\?u?????\?u\?k?\{kz\}zzw\?vwq????????z|?
x\?yz|"\?|??????\{\?ny\?y\?\?\?xu|\??\|\?x\?u|\?vq\?ww\}\r\?|????r?\?\}??{s}\?z|yvtpz
{"z\?{qzw\?u\?v\?x\?"??|\?x|\zt\?y\?xw\?w\?\{u\?x\?tx\}w\?x???|z????z????\wyt\w|wv\{p\?z\}?y????\tv
???}\{x\?

The '?' characters represent unprintable characters because they are either an ASCII control character or above 127. CCGAN produces values in the range of 0-255, so any values between 128 and 255 are not valid ASCII characters, and attempting to decode as both cp1252 and utf-8 produces invalid characters to compile for this paper. Most values produced are in the range of 110-146. That means most values are near zero from the final tanh in the output layer. After the output layer the values are rescaled from [-1, 1] to [0, 256], so a value of 0 from the tanh produces a final output payload byte value of 128. This is a common occurrence among all variations of CCGAN, so this limited range of outputs is likely caused by the architecture of the model.
This particular example is generated by CCGAN trained with the UNSW dataset, tanh activation function, seven layers, and RMSProp optimizer. However, the differences are not visually obvious in the outputs of different instantiations of CCGAN. The differences only become apparent when the data is used in the augmentation test.

Table 2 shows the results of each experiment. The results are broken up into accuracy against three different categories, ‘All Classes’, ‘Holdout Classes’ and ‘Non-Holdout Classes’. The all classes category is the entire dataset including both holdout and non-holdout classes, while the other two are each represent part of the dataset. Holdout classes are only the classes of packets that were not seen in training by either the CCGAN or the NIDS. The non-holdout classes were only the classes that were present in training.

<table>
<thead>
<tr>
<th>GAN Configuration</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All Classes</td>
</tr>
<tr>
<td>Baseline (no GAN augmentation)</td>
<td>0.966</td>
</tr>
<tr>
<td>CICIDS ReLU 14 Adam</td>
<td>0.964</td>
</tr>
<tr>
<td>CICIDS ReLU 14 RMSProp</td>
<td>0.971</td>
</tr>
<tr>
<td>CICIDS ReLU 28 Adam</td>
<td>0.968</td>
</tr>
<tr>
<td>CICIDS ReLU 28 RMSProp</td>
<td>0.967</td>
</tr>
<tr>
<td>CICIDS ReLU 7 Adam</td>
<td><strong>0.971</strong></td>
</tr>
<tr>
<td>CICIDS ReLU 7 RMSProp</td>
<td>0.965</td>
</tr>
<tr>
<td>CICIDS tanh 14 Adam</td>
<td>0.964</td>
</tr>
<tr>
<td>CICIDS tanh 14 RMSProp</td>
<td>0.965</td>
</tr>
<tr>
<td>CICIDS tanh 28 Adam</td>
<td>0.962</td>
</tr>
<tr>
<td>CICIDS tanh 28 RMSProp</td>
<td>0.966</td>
</tr>
<tr>
<td>CICIDS tanh 7 Adam</td>
<td>0.961</td>
</tr>
<tr>
<td>CICIDS tanh 7 RMSProp</td>
<td>0.962</td>
</tr>
<tr>
<td>UNSW ReLU 14 Adam</td>
<td>0.967</td>
</tr>
<tr>
<td>UNSW ReLU 14 RMSProp</td>
<td>0.968</td>
</tr>
<tr>
<td>UNSW ReLU 28 Adam</td>
<td>0.971</td>
</tr>
<tr>
<td>UNSW ReLU 28 RMSProp</td>
<td>0.967</td>
</tr>
<tr>
<td>UNSW ReLU 7 Adam</td>
<td>0.537</td>
</tr>
<tr>
<td>UNSW ReLU 7 RMSProp</td>
<td>0.968</td>
</tr>
<tr>
<td>UNSW tanh 14 Adam</td>
<td>0.968</td>
</tr>
<tr>
<td>UNSW tanh 14 RMSProp</td>
<td>0.962</td>
</tr>
<tr>
<td>UNSW tanh 28 Adam</td>
<td>0.966</td>
</tr>
<tr>
<td>UNSW tanh 28 RMSProp</td>
<td>0.966</td>
</tr>
<tr>
<td>UNSW tanh 7 Adam</td>
<td>0.965</td>
</tr>
<tr>
<td>UNSW tanh 7 RMSProp</td>
<td>0.708</td>
</tr>
</tbody>
</table>

Table 2: Data Augmentation Test (with the Highest Accuracy Results Highlighted in Bold)

There is no obvious best configuration of the CCGAN. However, at least one of the augmentation experiments has a higher accuracy than the baseline test in every category. The CCGAN trained with the CIC-IDS 2017 dataset, a ReLU activation function, seven layers, and an Adam optimizer scored 97.1% accuracy on the all classes category against a baseline of 96.6%. That same CCGAN augmentation test also shows significant improvement in the holdout classes category with 67.1% accuracy versus a 55.1% baseline accuracy, and only lost 0.2% accuracy in the non-holdout classes (98.7% vs 98.9%). This makes that version of CCGAN a strong candidate for the best overall result, depending on how the categories are weighted.

The best result on holdout classes was obtained by training the CCGAN with the UNSW dataset, tanh activation function, seven layers, and RMSProp optimizer. The augmented NIDS was able to score a 91.5% accuracy against a baseline of 55.1% accuracy, but at a trade-off of only scoring 69.7% versus 98.9% baseline on the non-holdout classes. Surprisingly, one augmentation test performed slightly better than baseline (< 0.1% difference) on the non-holdout classes, which is where we expected the baseline to be the clear winner. It did so by being worse at classifying the holdout classes than baseline, i.e. 46.4% vs 55.1% baseline.
(a) Baseline NIDS without augmentation

(b) NIDS augmented with CCGAN trained with UNSW-NB15 dataset, ReLU activation function, 28 layers, and RMSProp optimizer.

(c) NIDS augmented with CCGAN trained with CIC-IDS 2017 dataset, ReLU activation function, 7 layers, and Adam optimizer.

(d) NIDS augmented with CCGAN trained with UNSW-NB15 dataset, tanh activation function, 7 layers, and RMSProp optimizer.

Figure 3: Accuracy on malicious holdout classes, an ideal model would have 100 malicious in all four classes

The holdout classes are not uniform in quantity, which skews the holdout accuracy towards the models that perform the best on generic as the largest class. Table 1 highlights the holdout classes. Generic consists of 87% of the total holdout packets. Followed by DoS slowloris at 7%, backdoor at 4%, and Web Attack - XSS at 2%.
Figure 3 contains the confusion matrix for each holdout class for four experiments. Figure 3a is the baseline which is particularly good at identifying backdoor packets at 99% accuracy but close to random guessing on generic. The UNSW, ReLU, 28, RMS result in Figure 3b shows one experiment that was able to substantially improve on the DoS slowloris holdout class, going from 79% baseline to 97% augmented accuracy. Similarly, the UNSW, tanh, seven RMS result below in Figure 3d had the highest scoring holdout class accuracy because it was able to raise the generic accuracy to 94%. The bottom left matrix for CICIDS, ReLU, seven, Adam in Figure 3c shows the all classes accuracy winner, which is primarily because it was able to raise the generic accuracy from 52% to 66% without significantly compromising the accuracy on the non-holdout classes (98.9% to 98.7%).

6. CONCLUSION

CCGAN shows significant promise in generating synthetic data to augment training a ML model for a NIDS to identify novel attacks that are not present in the training data. Several versions of CCGAN were able to raise accuracy significantly on the holdout classes, and often without compromising accuracy on the non-holdout classes. Using tanh as the final output of CCGAN to constrain the values to [-1, 1] may be the cause of most values learning to be near zero because tanh is not a linear function. A different final function such as a clamp which allows values between 0-255 but forces anything below 0 to be 0 and above 255 to be 255 may produce results with a wider range of outputs. A transformer may be a better model than a GAN for producing payloads, if we can identify how to make the transformer creative to produce payloads that do not match the known payload distributions. CCGAN’s loss function pushes it to find payloads that are outside of the known distribution. Adding an ensemble or meta-learning approach could potentially allow multiple versions of CCGAN to be combined to get the best results of different models. For example, a combined model that uses the NIDS from Figure 3b to identify DoS slowloris packets but the NIDS from Figure 3d to identify generic packets.

ACKNOWLEDGEMENTS

This work was supported in part by the U.S. Army Combat Capabilities Development Command (DEVCOM) C5ISR Center under Support Agreement No. USMA21056. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Military Academy, the United States Army, the Department of Defense, or the United States Government.

REFERENCES


