MirageNet - Towards a GAN-based Framework for Synthetic Network Traffic Generation

Santosh Kumar Nukavarapu, Mohammed Ayyat, and Tamer Nadeem
Department of Computer Science
Virginia Commonwealth University
Richmond, VA 23220, USA
{nukavarapuskk, ayyatma, tnadeem}@vcu.edu

***Abstract*—With the emergence of machine learning technology that supports the development of synthetic models, many new use cases and challenges are emerging in the fields of computer vision and security. The main model behind this technology is Generative Adversarial Networks (GANs), with their ability to model unknown distributions accurately and perform well in generating synthetic data such as images and videos. However, the application of this technology by the networking community has been lacking. Given this motivation, we introduce MirageNet; our vision for a GAN-based synthetic network traffic generation framework, which can automatically create synthetic network models of protocols, applications, and devices. With the potential to build many applications for privacy, security, and network optimization. In this paper, we present MiragePkt; the first component of MirageNet. It is a GAN-based model to synthetically generate network packets. We describe the different challenges, limitations, and solutions for generating synthetic network packets. Finally, we validate and evaluate the performance of our framework with the synthesizing DNS packets.**

I. Introduction

The emergence of Generative Adversarial Networks (GANs) [9] - an advanced machine learning technology that can build synthetic models for audio and video - is creating new opportunities and as well as challenges in the field of computer vision and security. With more than 14,698 deep fake videos found on the internet, which is a significant increase of 84% from the last year [22], it highlights the rapid growth and excellent abilities of this technology to model very complex tasks and problems. The generated deep fakes are difficult to identify and pose a significant security risk for businesses and governments.

This emerging growth of synthetic machine learning technology, its use cases, and security challenges still needs to be explored by the systems and networking community. For example, the application of GAN technology to automatically comprehend and reverse engineer the grammar of networking protocols such as TCP and UDP, or to automatically create a network traffic model of devices and mobile applications, remains to be investigated.

The development of such synthetic network models might have several applications in privacy, security, and network optimization. In terms of privacy, synthetic network models of apps and devices, for example, can generate fake user activities in smart homes, reducing the accuracy of side-channel attacks for user activity inference. In security research, the synthesized network data from these models could augment training data to build effective intrusion detection systems. Moreover, real-world honeypots may be constructed utilizing fictitious devices and applications in order to better understand the attacker's behavior and interactions with these apps and devices. Furthermore, application and device synthetic network models might be utilized to replicate numerous testing scenarios for analyzing performance concerns such as load balancing, predicting network conditions, and diagnosing network difficulties.

Creating synthetic network models of real-world devices and applications is challenging, given the black-box nature of their proprietary protocols and application logic. Therefore, to automatically reconstruct an application or device's network model, it is necessary to automatically understand the network model's syntax and semantics and reverse engineer them without the use of manual coding. While the syntax part consists of the protocol, flow, and packet structures, the semantics consists of meaningful network activity. For example, a synthetic network model for an IoT device should first generate network traffic for the ON user event before generating the network traffic for the OFF event. Thus, an efficient synthetic network model of a device or application should learn the syntax, semantics, and interdependencies among them automatically.

Given this motivation, in this paper, we design, evaluate and discuss the challenges of a network packet generation framework *MiragePkt* that can automatically understand the syntax of the network packets from the raw network packets. This framework is the first step towards realizing our vision of creating a GAN-based synthetic Network traffic generation framework, *MirageNet*, that can automatically create synthetic network models of protocols, applications, and devices. We summarize the contributions of our paper as follows:

- We design and develop *MiragePkt*; a synthetic Network Packet Generator framework, as the initial component of *MirageNet*.
- We present a sequential byte modeling approach for network packet generation using GANs.
- We describe the challenges, limitations, and solutions towards generating synthetic network packets.

This material is based upon work partially supported by the US National Science Foundation under Grants No. CNS-1764185 & OAC-2212424.

978-1-6654-3540-6/22 © 2022 IEEE

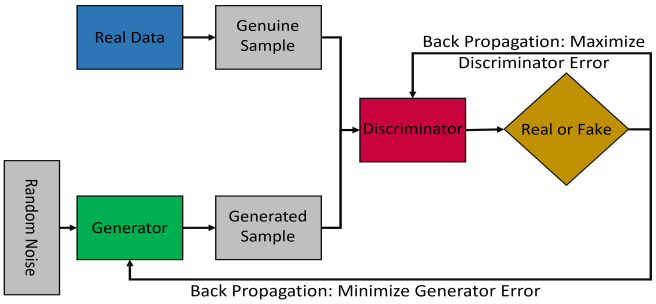

Fig. 1: Overview of the base architecture of GAN

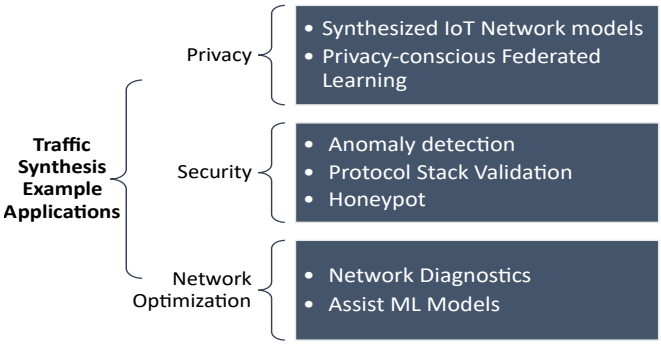

Fig. 2: Different research and engineering domains where synthetic network traffic would have significant impact

- We validate and evaluate the performance of our framework using synthesized DNS packets.

II. Background

For many complex generative tasks, such as image generation, estimating the probability distribution of the different features of images (i.e., pixel intensity values) is difficult. Generative adversarial networks (GANs) are known to model unknown complex distributions and have performed well on such complex generative tasks [9]. The GAN model consists of two neural networks, discriminator and generator, that form a two-player min-max game where the generator tries to generate fake samples and the discriminator tries to identify if the samples are from the training data (real) or generated by the Generator (fake). As shown in Figure 1, the generator model accepts as an input a random noise and then transforms this noise into a fake sample as an output. The min-max game forces the generator to approximate the true distribution. Moreover, the training process reaches an optimum solution when the min-max game reaches Nash Equilibrium, where the fake samples from the generator are predicted as fake by the discriminator with 50% probability.

The GAN training consists of simultaneously training the generator and the discriminator models on small batches of data known as mini-batch with a fixed size. Each iteration will train the GAN model with this fixed mini-batch of data points. For a training iteration, the generator is given a batch of random points from the latent space to output a batch of fake samples. This batch of fake samples is given to the discriminator to predict if they are real or fake. This process updates the weights of the discriminator. Similarly, next, a batch of samples from the training data is given as input to the discriminator, which again updates the discriminator's weights. This process forces the discriminator to learn the features of real and fake data. Moreover, the loss of discriminator for predicting fake samples is sent as feedback to the generator, which updates its weights accordingly to generate better realistic samples. The generator training is called unsupervised training as the generator model is never exposed to real data during training. After training, the generator can be detached from the GAN model and independently used for generative tasks. Recently, GAN-based generative models are used for disruptive technologies such as deep fakes to generate fake content (videos, audio, images) [1] [23] [12] that is indistinguishable from real content. Similarly, the discriminator model can be used for many image classification tasks [16] [20]. Recently, image-based anomaly detection using a GAN discriminator has been proposed [14].

III. MirageNet Applications

Synthetic data generation is an emerging research topic in the machine learning domain. As it can boost the performance of various machine learning algorithms [11], solve class imbalance issues when there is insufficient data for certain classes [21], and most importantly, the ability to model any unknown distribution [9]. Similarly, we believe synthetic network traffic data can enable new different applications in the network domain and support existing ones. Figure 2 shows the three areas of applications that we envision for synthetic network traffic: privacy, security, and network optimization.

Starting with privacy, we see multiple use cases for synthesizing network traffic such as generating traffic simulating real IoT devices and using it to counter side-channel attacks to prevent the leakage of user activities in sensitive applications such as healthcare [2]. Another use case is developing a privacy-conscious federated learning model where instead of using real data to build network models which can lead to information leakage [8], we can train the federated learning models using synthetic data with similar data distribution.However, it is to be noted tha that preserving privacy is only possible if the GAN training is stable and the generator is not overfitted to the training data as that can lead to memorization and information leakage.

Another major area that can utilize synthetic network data is the security domain. For example, data augmentation using synthetic network traffic can be used to train anomaly detection systems with a low number of labeled malware samples by generating synthetic malware samples and adding them to the training data. Additionally, traffic synthesis can be used to generate packets with a wide range of characteristics to test protocol vulnerabilities in the device network stack. For example, a malformed packet can cause certain target device functionalities to fail and cause a denial of service. Building synthesized malware is another interesting application where attack patterns can be recorded by deploying "Honeypots" to

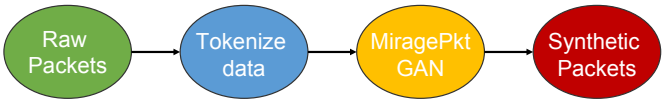

Fig. 3: Packet generation pipeline of MiragePkt

capture real malware traffic and learn its properties and use the synthesized malware to build defense mechanisms that can defend against large-scale IoT-based attacks such as Mirai [6].

Finally, we also consider the area of network optimization, where synthetic network traffic simulating certain network conditions such as high congestion can be used to measure network performance. Moreover, generating scenario-specific synthetic network data can help improve the efficiency of many ML-based network optimization tools. For example, the efficiency of reinforcement learning-based models can be improved by training them under an extensive range of network conditions using GANs [11]. Given the above interesting applications, in the below section, we will discuss our framework for generating synthetic network packets as a first step towards building a complete flow generation framework.

IV. MiragePkt - Packet Generation Framework

In this paper, our focus is to build a packet generation framework that can synthesize network packets whose protocol is unknown. We chose this objective given the significance of generating synthetic data for proprietary protocols where traditional network parsing tools such as scapy and traffic generators cannot be utilized as they fail to automatically understand the syntax of the network packet with multiple fields, values, and their corresponding correlations. Specifically, we focus on building the synthetic packet generation model using the raw data collected using tools such as TCP dump, which consists of a binary stream and does not give information about the structure of the packet. In this paper, we pre-collect only the raw binary stream of the network packets as the training data and do not consider any metadata information about the structure or field values to build our protocol-agnostic packet generation framework. With this motivation, we designed our packet generation, which is depicted in Figure 3, as a multi-component system with each component described in more detail in the following subsections

A. Raw Packet Stream Data Collection

This is the first step of our framework pipeline that performs two significant tasks, collecting the network packets and extracting raw byte streams from the network packets. The collection of network packets from online or LAN networks that follow proprietary protocols is challenging, given that traditional network parsing tools such as scapy cannot identify similar packets belonging to a specific proprietary protocol. For example, collecting DNS packets without knowing any information about the DNS protocol is difficult. Therefore, in this component, we use Natural Language Processing (NLP) and byte similarity-based calculations between packets to identify and filter the required raw packets by using the byte stream. However, for this paper, we generate DNS packets based on a scapy script and use this component only to extract the hex stream from the packets. Designing a sub-component to filter and collect data streams of network packets of a specific protocol is out of the scope of this paper.

B. Data Tokenization

Tokenization is an NLP-based text pre-processing scheme that is used to divide the text into different tokens where each token can be a word within the text, a group of words, characters, or sentences. Similarly, tokenization for packet-based byte data consists of dividing the bytes either into individual tokens or groups of bytes within different packet headers. In this paper, we transform the network byte stream data into sequences of tokens where each token represents the byte of the packet. Our intuition here is to model the packets as sequences of bytes and use GANs to understand the conditional distribution of bytes in a sequence of packets. In this paper, we first tokenize the data and perform one-hot encoding on each byte in a sequence for creating the training data for GAN. However, our goal is to later extend this in future work with different data pre-processing methods from NLP such as word-bag, word embedding, or building a word2vec model for bytes of network packets that can better understand the correlations between the bytes of network data.

C. MiragePkt - GAN Generation Model

Figure 4 shows the overall architecture of the *MiragePkt* GAN model, which consists of two components; a generator and a discriminator. The generator's objective is to generate the sequence of tokens representing the byte stream of a packet. The discriminator's objective is to extract the features from the generated sequence and identify whether the generated sequence belongs to the true distribution of the training data.

This model training process is similar to the vanilla GAN training discussed in the background section. However, one of the significant challenges in generating sequences of bytes using vanilla GAN is its inefficiency in generating discrete data. Many GAN architectures perform efficiently with continuous data and have shown excellent performance with image data. However, for the sequence generation tasks, GANs have performed poorly in domains such as music and text.

In this paper, we use a combination of 1D CNN and Softmax functions in the generator to predict each of the tokens to form the byte sequence of a network packet. Generally, CNNs are well known for extracting the raw features from the 2D image data using the convolution operation. However, they have not been that efficient with 1D dimensional data such as text. Recently, 1D CNN [13], which is very similar to the 2D CNN with convolutional operations but has one-dimensional filters to extract the features from 1D data, has shown good performance with text classification tasks. Therefore, this paper uses the 1D CNN in the discriminator to extract the features from the 1D byte sequences data to identify between real and fake packets. It is to be noted that, unlike a simple RNN or LSTM-based regression task that can predict a token based on the previous tokens, this GAN architecture can generate

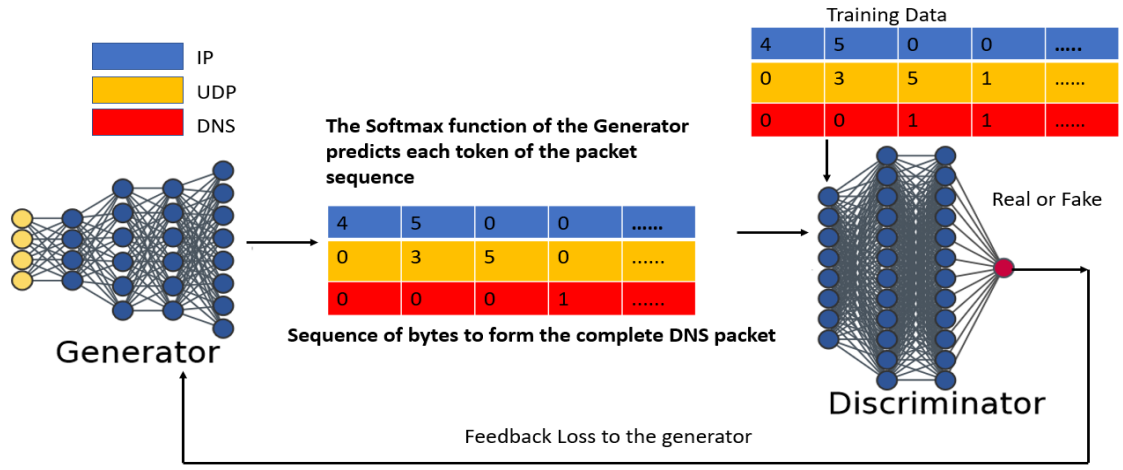

Fig. 4: MiragePKT generation framework

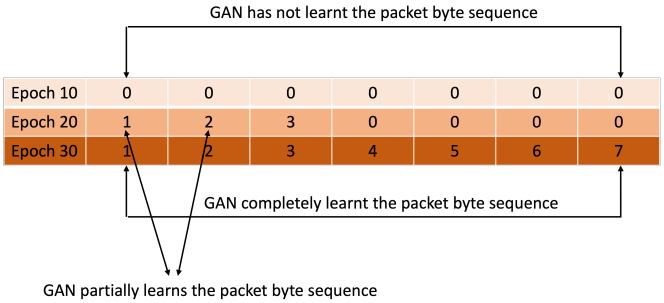

Fig. 5: Byte sequence generation during MiragePKT training

Domains
instagram.com
bing.com
google.co.uk
alibaba.com
google.com.br
google.co.in
wikipedia.org

TABLE I: Samples domains from the Alexa dataset

all tokens of the sequence at once, which is closer to the true distribution of the training data.

The generator and discriminator models consist of 1D CNN layers and are built using the improved Wassertian model [10], which has shown improved synthetic generation for images. The improved Wassertian model has also shown significant improvements in stable GAN training ,and therefore it is adapted as our base model in this work. Figure 5 shows the evolution in byte sequence generation of the *MiragePkt* where in the early epochs, the generator model is producing random sequences without any correlation. However, in the later epochs, the generator learns the structure of the packet and its order sequence and, therefore, can generate high-quality sequences at later epochs. We discuss the improvement of the packet generation during training by showing the verbose text of network packets later in the evaluation section.

D. Post Processor

Finally, this last component of the framework pipeline generates the hex stream of the packet from the output of the *MiragePkt* that consists of a numerical representation of a packet sequence and then identifies if the packet is of high or low quality before sending it to the network. Determining the quality of a network packet can be measured by looking at metrics such as measuring its similarity to the training data, and measuring the randomness of the data such that the packet does not belong to the original training data or previously generated packets. We discuss some of these metrics in our evaluation section that we discuss next.

V. Evaluation

In this paper, we pick DNS packets as our use case for modeling network packets using GANs as they follow a complex syntax grammar with multiple fields such as domain name, questions, and answers. This complexity can provide better insights into the challenges of synthetic packet generation. This complexity can provide better insights into the challenges of synthetic packet generation. Additionally, recent attacks using malformed DNS packets [7] have further motivated us to pick this use case from a security perspective, as an attacker can generate malformed DNS packets to attack devices, and thus, a GAN-based tool to generate DNS packets can further help with the security research. It is noteworthy that DNS packets contain variable length payloads because of the domain names, and in our *MiragePkt* during the tokenization step, we get the maximum length of byte streams among the training data packets to create byte sequences with the max length and append zeros to any packet with less than the max length. Building packet generation system with dynamic payloads and variable sizes or complex conditions is much more challenging than simple deterministic network packets. We discuss next some of these challenges below.

A. Datasets and Experiment Design

To train *MiragePkt*, we first generate DNS packets using scapy and then use their hex stream as our raw training

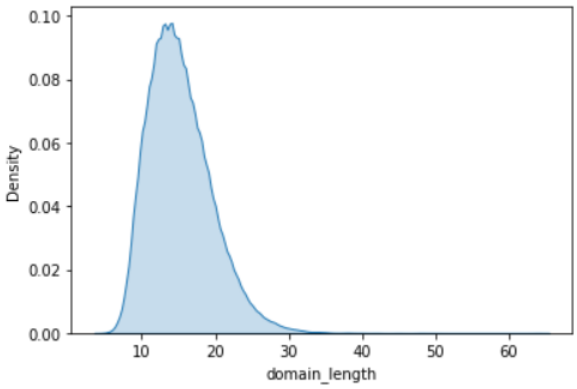

Fig. 6: Domain length distribution in *VarLen-AnyDom* dataset

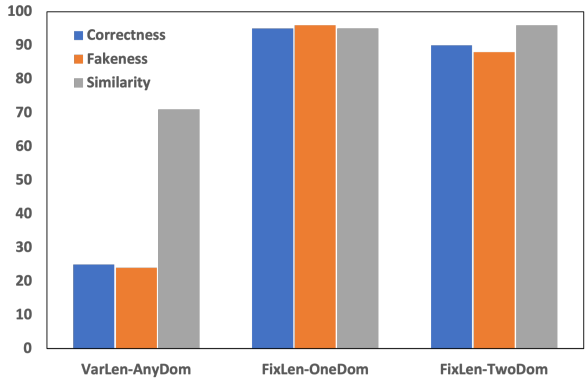

Fig. 7: Performance of *MiragePkt* across different complexity of datasets

data. We create a python scapy based script that generates DNS packets using real-world domain names as the payload. Figure 6 shows the distribution of the domains with variable length, and Table I shows some of the sample domains from the dataset. The efficiency of the machine learning model is highly dependent on data pre-processing and the final input training data. Therefore, to understand the impact of the training data on *MiragePkt* ability to reverse-engineer the DNS network packets automatically, we created three different training datasets with varying complexities, *VarLen-AnyDom*, *FixLen-OneDom*, *FixLen-TwoDom*. In the *VarLen-AnyDom*, we pick 50000 variable-length payloads with a distribution of domain name lengths as shown in Figure 7. Next, we create the second dataset *FixLen-OneDom* that consists of 50000 packets with a fixed domain length domain and a sigle domain suffix such as "*.com*". Finally, the *FixLen-TwoDom* dataset has 25000 domain names with two suffixes such as "*.com*" and "*.org*". The *FixLen-TwoDom* configuration is to understand the impact of DNS packets with different domain suffixes on *MiragePkt*. We implement *MiragePkt* using PyTorch [17] and Python and train it on a GPU-enabled machine with 32 GB of memory. In the following subsections, we analyze the training data and discuss our model evaluation with different training dataset configurations with different complexity levels.

B. Impact of Training Dataset Configurations

Using several metrics, such as correctness, fakeness, and similarity, we validate various models of *MiragePkt* trained on datasets of varying complexity, as discussed earlier, with the intuition that model performance will increase with a decrease in the mathematical dependencies of the training data. We define ***Correctness*** as a measure of how many of the synthetically generated packets are legitimate DNS network packets. We also define ***Fakeness*** as a measure of percentage of new valid DNS packets, and ***Similarity*** as a measure of how similar the byte strings of the generated packets and the training dataset packets are.

From Figure 7, we see that for *VarLen-AnyDom* configuration, *MiragePkt* performance is highly inefficient across different metrics. The *MiragePkt* major failure for this training dataset configuration is its inability to generate valid DNS packets. Table II shows the different packet errors found while generating DNS packets, where most of the generated packets have incorrect IP header length and UDP header length field values. *Therefore, MiragePkt needs to be improved to understand unknown functions among different bytes of the packet.* However, designing such a network architecture or supporting loss function is a very challenging task and needs to be researched more.

Figure 7 shows that changing configuration from *FixLen-OneDom* to *FixLen-TwoDom* improves the performance of the *MiragePkt* significantly, with an increase in the number of correct packets from 25% to 95% and an increase in fakeness from 25% to 94%. It is noteworthy that we measure the fakeness of the packets by searching for the fake byte stream string in the corresponding training dataset configuration. *Therefore, MiragePkt does not overfit the training model by merely memorizing the training data. On the contrary, it generates high-quality synthetic content*. Moreover, we observed that the number of domains generated for *FixLen-TwoDom* contains both "*.com*" and "*.org*" in the generated packets. *Therefore MiragePkt was able to capture the different modes of the training data and does not fall into the mode collapse problem where the GANs only generate one mode of the distribution.*

From Figure 7, there is no significant increase in the percentage of correctness or fakeness when switching from *FixLen-OneDom* to *FixLen-TwoDom*, where both configurations have the same fixed length of domains. *Therefore, variations in domain suffix structure do not affect MiragePkt significantly as much as the mathematical dependencies introduced by variable length domains.*

Additionally, it is interesting to note that irrespective of the different configurations, the similarity score of the bytes of the synthetic content across all configurations remains very high. *Therefore, MiragePkt can generate a synthetic packet that is very similar to the original training dataset*. As for calculating similarity, we performed a byte-level string-to-string similarity to find the maximum similarity of each generated packet across training dataset packets and then calculated the average of all the similarity scores. Figure 8 compares the first and last bytes of payloads created synthetically from *MiragePkt* versus actual

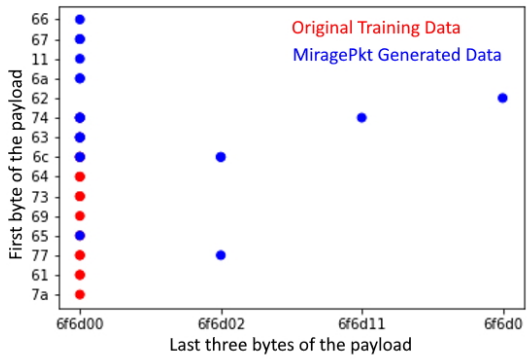

Fig. 8: The first and last three bytes of payload samples from original training packets (red dots) and *MiragePkt* generated packets (blue dots)

Packet Errors
IPV4 header "*IHL (Internet Header Length)*" field value > Total length of the IP Packet
UDP header "*Length*" field value > IP Payload length of the packet
IP header "*Total Length*" field value > IP Packet Length
Malformed Packet – due to invalid bytes for different fields

TABLE II: Examples of parsing errors in generated packets

payloads taken from the training dataset. The figure's X axis displays various combinations of the payload's last three bytes in hex, while the Y axis displays various combinations of the first byte. The scatter figure demonstrates that the generated payloads can completely recreate the last bytes of the payload, which stand in for the suffix, as well as the majority of the payload's initial bytes.

C. Content Visualization

To inspect the synthetic packets visually in a process similar to how the quality of synthetic images is assessed, we use the scapy tool to output the verbose text of the generated hex-byte stream. Figure 9 shows the improvement of packet generation during training. At epoch 20, the network packet is malformed, and scapy cannot construct a valid packet using the generated hex stream. However, by epoch 50, scapy becomes fully capable of reconstructing the network packet with all the fields. *Thus MiragePkt can learn the syntax of the DNS packets with their correct field values.*

VI. Related Work

Some researchers have recently worked on using GANs to build models that generate synthetic metadata of network traffic where the metadata is the high-level parameters of a network flow such as number of bytes, number of packets, and flow duration [18] [19]. In [19] authors used a *word2vec* embedding approach to generate the flow metadata parameters of the traffic, given an embedding approach that can capture the high-level relationships among different flow metadata attributes in high dimensional space. However, their work generates the flow metadata traffic and doesn't generate the network packets. We believe their work can be extended

Fig. 9: Packet visualization of *MiragePkt* using scapy

and augmented to our packet generation model for building an initial design of a complete network traffic generation framework.

Recently, some researchers have used GANs to generate the synthetic packets. In [4] they use CNN GANs and a special encoding mechanism to generate different types of network packets such as DNS, Ping, etc. However, our work is significantly different, given we use different data prepossessing modeling using a sequence-based approach to model the bytes of a network packet. Moreover, unlike their work which uses a GAN to generate continuous values and then map them to discrete byte values, our model uses the softmax in the generator to predict the byte sequences of the packet. Furthermore, unlike their work, where the data transformation is done by duplicating a byte value of a hex stream packet multiple times to achieve high similarity at the cost of increasing the dimensions of training data, our model only requires the tokenized data, making it more scalable for training and packets with large byte streams.

Recently authors in [5] used a sequence-based approach to generate adversarial packets, which is similar to our sequence-based modeling approach to packet generation. However, *MiragePkt* significantly differs in the model design, framework and our objective to show the challenges with packet generation using DNS packets, unlike their focus on evading IDS using adversarial packets. Moreover, *MiragePkt* is only the first component of our larger vision of building *MirageNet*.

VII. Conclusion & Future Work

In this paper, we focused on the first step towards realizing *MirageNet* by developing *MiragePkt* where we were able to demonstrate the many intricacies in the packet generation process, the impact of training data, and the limitations of GANs in comprehending some of the mathematical functions. In our future work, we intend to develop a few components that improve the packet generation model.

First, we want to investigate pre- and post-processing components to aid in the preparation of better training data and to automatically fix any malformed packets with minor

flaws. For example, as a pre-processing step, we can design scripts to identify non-deterministic regions in the overall training data, such as payload, and map them to any of their correlated fields, such as header length, to provide metadata that will support GAN in automatically understanding these mappings. Similarly, a post-processing tool might be created by employing a reinforcement learning (RL) agent [3] capable of learning to repair any malformed packet based on replies from a server or device that is processing the synthesized packets.

Another *MiragePkt* enhancement would be conditional generation employing conditional GANs [15], where instead of randomly generating any network packet ending in "*.com*" or "*.org*", the *MiragePkt* may generate packets with a certain domain suffix as requested by the user during the generation stage. The next important component we intend to develop is the *MirageNet* flow generation framework in its entirety. In this framework, the model must be capable of capturing the temporal dependencies between packets in a flow as well as grasping the protocol syntax. For example, in a TCP flow, the first three packets must be syn, syn-ack, and ack, in which case the model must automatically capture these rules. This would require combining two models; a flow metadata generator to generate information about the packets and their sequences in a flow, and the other model to generate actual packets. We plan to use an LSTM-based model for generating the metadata of the synthetic flow, which can be sent to *MiragePkt* for generating the content of the packets conditioned on this synthetic input metadata.

REFERENCES

[1] R. Aloufi, H. Haddadi, and D. Boyle. Emotion filtering at the edge, 2019.

[2] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic, 2017.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[4] A. Cheng. Pac-gan: Packet generation of network traffic using generative adversarial networks. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0728–0734, 2019.

[5] Q. Cheng, S. Zhou, Y. Shen, D. Kong, and C. Wu. Packet-level adversarial network traffic crafting using sequence generative adversarial networks. *arXiv preprint arXiv:2103.04794*, 2021.

[6] T. N. J. Cybersecurity and C. I. C. (NJCCIC). "mirai botnet". https://github.com/sdv-dev/CTGAN, December 2016. Accessed: 2020-15-08.

[7] D. dos Santos, S. Dashevskyi, A. Amri, J. Wetzels, S. Oberman, and M. Kol. Name:wreck, breaking and fixing dns implementations. https://www.forescout.com/resources/namewreck-breaking-and-fixing-dns-implementations/.

[8] J. Geng, Y. Mou, F. Li, Q. Li, O. Beyan, S. Decker, and C. Rong. Towards general deep leakage in federated learning. *arXiv preprint arXiv:2110.09074*, 2021.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[11] A. T. Z. Kasgari, W. Saad, M. Mozaffari, and H. V. Poor. Experienced deep reinforcement learning with generative adversarial networks (gans) for model-free ultra reliable low latency communication. *IEEE Transactions on Communications*, 69(2):884–899, 2020.

[12] Y. Kataoka, T. Matsubara, and K. Uehara. Image generation using generative adversarial networks and attention mechanism. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6, 2016.

[13] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

[14] F. D. Mattia, P. Galeone, M. D. Simoni, and E. Ghelfi. A survey on gans for anomaly detection, 2019.

[15] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[16] A. Odena. Semi-supervised learning with generative adversarial networks, 2016.

[17] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] M. Rigaki and S. Garcia. Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 70–75, 2018.

[19] M. Ring, D. Schlör, D. Landes, and A. Hotho. Flow-based network traffic generation using generative adversarial networks. *Computers & Security*, 82:156–172, May 2019.

[20] T. Salimans et al. Improved techniques for training gans, 2016.

[21] V. Sampath, I. Maurtua, J. J. Aguilar Martín, and A. Gutierrez. A survey on generative adversarial networks for imbalance problems in computer vision tasks. *Journal of big Data*, 8(1):1–59, 2021.

[22] D. S. T. Times. Number of deepfake videos online rises 84 percent in less than a year. https://www.techtimes.com/articles/245628/20191009/number-of-deepfake-videos-online-rises-84-percent-in-less-than-a-year.htm. Accessed: 2020-15-08.

[23] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation, 2017.