SIP-GAN: Generative Adversarial Networks for SIP traffic generation

Amar Meddahi[1,2], Hassen Drira[2,3], Ahmed Meddahi[2]
[1]University of Toulouse, INP-ENSEEIHT, Toulouse, France
[2]IMT Nord Europe, Institut Mines-Télécom, Center for Digital Systems, Lille, France
[3]Univ. Lille, CNRS, UMR 9189 – CRIStAL – Lille, France
amar.meddahi@etu.inp-n7.fr, hassen.drira@imt-nord-europe.fr, ahmed.meddahi@imt-nord-europe.fr

***Abstract*—Generative adversarial networks (GANs) are one of the major ML techniques for data augmentation and classification, in the field of image processing, computer vision and natural language processing. However, in the field of data networks and protocols the use of GANs for data generation and classification (at packet level) is very limited or relatively new. Although, GANs specific properties and characteristics can be highly relevant in this context (unsupervised technique). This limitation, is even more critical if we consider network protocols or communication oriented protocols such as SIP VoIP. To address this problem, we propose "SIP-GAN" an extension and adaptation of GANs model for SIP, aiming to process and generate SIP traffic at packet level. The proposed generic model includes an encoder, a generator, and a decoder. The encoder extracts information from pcap data, associates and converts these SIP data into a GAN image representation. The generator is based on a DCGAN model, that generates new SIP dataset from each extracted image. The decoder combines the generated images and reconstruct a valid pcap file (SIP file). A specific testbed, with a formal and practical analysis, demonstrate the validity of the generated data, from the SIP-GAN model. Also, the experimental and performance results are globally satisfactory, showing the relevance of our proposed SIP-GAN based traffic generator in this context.**

***Index Terms*—generative adversarial networks, SIP, traffic generation, data augmentation, networks security.**

I. Introduction

GANs have been studied and applied to various fields such as image, audio and videos. The advantage of GAN lies in learning the intricate data distribution from the real data and carefully reproducing similar but subtle variants of the real data. For example, applied to images, a GAN learns complex patterns and relationships between pixels and generates news ones with similar properties but including a variability compared to training images.

To the best of our knowledge, the existing techniques based on GAN, to generate network data, in order to test the various security devices implemented in networks, are very limited. In fact, generating realistic network traffic, is a critical issue, for developing and testing network security techniques (ex. Intrusion Detection/Prevention Systems).

We should also point out the main weak points regarding the typical generators, such as:

- Synthetic traffic generators are often unrealistic or suffer from a data generation bias.
- Commercial test generators can be effective in certain conditions, but are generally costly in terms of CAPEX and OPEX.
- Most of the typical generators are not compatible with some well-known traffic analyzers (ex. Wireshark) for a fine-grained analysis [3].

The construction of a real database could be relevant, but building a dataset is a time-consuming and costly task. Besides, working on customer databases leads to privacy concerns or issues. It is therefore essential to define new methods or approaches to generate applications oriented or specific databases, to increase the robustness of the dataset and thus, the ad hoc security mechanisms. GANs are particularly a promising technique, to solve the main current problems, regarding network data generation. This contribution addresses the main following research challenge, that is: to produce a large corpus of realistic and labeled SIP traffic data, based on cost-effective methods.

A. Related Works

Before describing in details our proposition, we first conduct a literature review of existing work related to GANs from a network security perspective. The use of GANs for network security constitutes a new paradigm while research works in this domain were first published in 2014. [4] paved the way to many sub-research topics, especially in the computer vision domain, but very few in the communications networks domain. It is only in 2018 that the first contributions to the application of GANs in computer network security appear. Following by some specific contributions such as botnet detection [11], attack generation [2], malware [8] and network data augmentation [3].

1) Network Flow Level: [3] proposes to use GANs to generate network traffic at the flow parameter level. Generally, flow-based network traffic contains several typical attributes such as IP address, time interval, port number, layer structure sequence and options. The discrete nature of network traffic, makes difficult to build adversarial network traffic. This is mainly due to the main characteristics of GAN, that can effectively generate continuous data, such as image synthesis. However, GANs based on sequential data [12] allow to use this type of data, but are complex in terms of implementation.

978-0-7381-1316-6/21/$31.00 ©2021 IEEE

The proposed approach in [3] is structured following tree steps: encoder, generator and decoder. In order to use the discrete network traffic, the encoder is used. It encodes IP address with a graph (source IP to destination IP) to exploit a GraphGan [10]. It also encodes time intervals to an image, to exploit an ImageGan [5]. And then, a GANs based on sequential data is used to manage the layer structure sequence, which is a sequential data. After the encoding process, data are generated and then decoding process is activated to be able to exploit the generated data.

2) Packet-Level: [1] proposes to exploit GANs to generate traffic at the packet level. Excepted the checksums, all packet fields are generated directly by a GAN. The GAN generates three different types of traffic: ICMP PING, DNS queries and HTTP Get requests. As opposed to the previous approach, the author uses another method to generate traffic.

GANs are the best candidate to generate realistic images for different computer vision applications. The idea is to focus on this effective technique, but from a network traffic generation perspective. For this, the author proposes to encode network traffic data as $n \times n$ pixel images. Then, a traditional GAN training is used and the generated network data is decoded.

B. Proposed Approach

Existing approaches consider network protocols as homogeneous or traffic generation at a network flow level only. As opposed to the existing works, we proposed to exploit and adapt Deep Convolutional Generative Adversarial Networks (DCGAN) to generate SIP traffic, while taking into consideration its specific characteristics. Indeed, the proposed approaches should be application oriented, to address the specific characteristics or constraints for each type of applications and services, such as VoIP (QoS/QoE metrics). Based on the structure of DCGAN, we propose SIP-GAN for adversarial SIP traffic generation at the packet level. SIP-GAN can be exploited to generate adversarial SIP data traffic, that is compliant with the standards (RFCs) in terms of packet format, content and state machine... Each byte of a network packet can be mapped to a pixel and by extend, the SIP packet to a grayscale image. So, the generation of adversarial SIP traffic can be formulated as a GAN problem for image processing.

We also conduct tests and experiments with a real dataset, which is defined and developed specifically to assess the performance of the proposed scheme, while considering different metrics. The experimental results show that the proposed scheme can be used to generate adversarial SIP data traffic at packet level. Our main contributions are the followings:

- A model, called SIP-GAN, based on the DCGAN structure, to generate adversarial SIP traffic at packet level by formulating and mapping a SIP message (call-flow) to a grayscale image.
- A practical method, with a specific python library, to extract and process the bytes from different types of packets. The proposed method and the library are generic and are applicable to other types of network protocols.
- The experimental testbed, shows that the adversarial SIP traffic generated by SIP-GAN gives, globally, satisfactory results.

The rest of this paper is organized as follows. Section II presents preliminaries related to GANs and SIP protocol. Section III presents and describes the SIP-GAN architecture and process. In section IV we describe the methodology to extract and process the real dataset, we define and assess different metrics to evaluate the performance of our proposed approach. Finally, in section V we conclude and discuss some potential perspectives and future works.

II. Preliminaries

In this section, we focus on concepts and methods that help to understand the proposed approach, from a network perspective. We give a focus on GANs also from a network perspective, with a SIP protocol overview.

A. GANs

GANs [4] can be compared to a game, where both players are modeled by a neural network: the generative model G and the discriminator model D. The generative model captures the data distribution in order, to generate samples, while, the discriminator model estimates the probability a sample belongs to the training data, rather than in the generative model. The two models compete with each other, in a minimax game and finally reach Nash equilibrium.

The generator takes as input, a noise parameter z from a normal distribution. Then, z parameter is mapped to a real data space as $G(z, \theta_g)$ where θ_g are the parameters of the generator's neural network. Similarly, $D(x, \theta_d)$ is defined by the parameters θ_d. $D(x)$ denotes the probability that x belongs to the real data, rather than to the generator's distribution p_g. The discriminator is intended to maximize the probability of giving the correct labels of the real samples and the generated ones. Also, the generator is trained to minimize $log(1 - D(G(z)))$. Thus, this is a typical minimax game and the objective function is described by equation 1.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

The objective function is related to Jensen-Shannon divergence that minimizes the divergence between the generator's distribution p_g and the real data distribution p_{data}. This neural network architecture is still under consideration from a scientific point of view, arousing a great interest from the research community. The application of GANs is relatively broad, as it covers or can be applied to different domains, and a large part of these applications or use cases, still need to be explored.

DCGAN [7] and [9] are key contributions, as they introduce a series of recommendations for the architecture of the generator and the discriminator.

So, we propose to take benefit from the specific properties and characteristics of DCGAN, to apply and combine DCGAN in the context of SIP architectures and protocols.

B. SIP Overview

Session Invitation Protocol (SIP) [6] is a signaling protocol used for establishing, controlling and terminating call sessions (voice, video or data oriented). On behalf SIP signaling, Real Time Protocol and Real Time Control Protocol (respectively RTP, RTCP) are used for the transfer and control of the media flow (voice, video or data).

SIP is a "rendezvous" protocol standardized by the IETF, specified in RFCs 3261 and supplemented by RFC 3265, with a number of other related RFCs, for supporting different types of SIP services (Presence, IM...). SIP is a text based signaling protocol, intended to establish, modify and terminate multimedia sessions. SIP is independent of the media communication protocol used during the session. Therefore, a multitude of ad hoc protocols can be combined with SIP (ex. RTP/RTCP, HTTP, SMTP...). As SIP is based on a client/server request model, a SIP session is characterized by a succession of well-defined transactions (ex. INVITE, BYE, CANCEL...). SIP based communications are generally "real time" oriented and so, are sensitive to QoS metrics such as: delay, jitter or packet loss in the media flow and connection delay, pre/post-selection delays for the signaling flow. This end to end "real time" characteristic must be taken into consideration, when exploiting GANs for SIP.

The next section describes, step by step, the application, combination and adaptation of GAN in the field of network protocols, for generating SIP packets.

III. SIP-GAN

SIP-GAN[1] architecture and model is composed of different blocks: an encoder, a generator, and a decoder. The encoder extracts information from pcap data, and converts this pcap trace into an image. Then, a DCGAN model is used to generate new SIP data packets, for each extracted image (data augmentation step). Finally, the decoder translates the generated images, to retrieve the bytes, corresponding to the new generated packets. From the generated SIP packets, a valid pcap file is re-built, analyzed and validated, with a SIP network analyzer tool (Wireshark). The overall process of SIP-GAN is summarized and illustrated in figure 1.

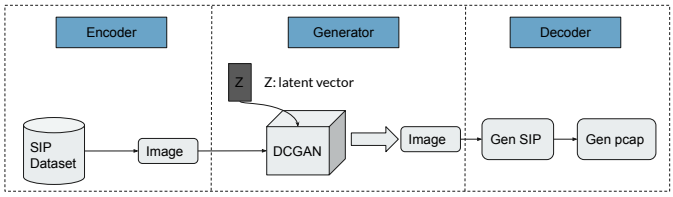


Fig. 1. SIP-GAN based generator architecture

A. Encoder

The main principle of the encoder, is to build a grayscale image from each byte of a SIP packet. Based on this encoder, an image data set is defined and used to train the generator, in order to generate "fake" SIP packets. Let $S = \{s_1, ..., s_p\}$ be a vector where $\{s_1, ..., s_p\}$ represent the value of each byte in the packet. If the SIP packet size is 500 bytes, so the length of S will be of 1000 bytes. As an example, the byte $3F$ is represented in S with $s_{2k} = 3$ and $s_{2k+1} = 15$, where k represents the byte position (index) in the SIP packet. We note l the multi-mapping parameter such as, each s_i in S is encoded and duplicated $l \times l$ times, in the corresponding grayscale image. Let $I \in M_{m \times n}(\mathbb{R})$ be a matrix, where each value corresponds to a pixel from a grayscale image, such as $mn = pl^2$. Let f be a bijective function (equation 2). Algorithm 1 illustrates the proposed method for encoding SIP packets.

$$\begin{array}{rccc} f & : & [0,15] & \rightarrow & [0,255] \\ & & x & \mapsto & 16x+8 \end{array} \quad (2)$$

Algorithm 1 SIP-GAN encoder
Input: S: SIP packet to encode, l: multi-mapping parameter
Output: I: grayscale image corresponding to S
Initialization : $I \in M_{m \times n}(\mathbb{R})$, $index = 0$
for $i = 0$ to n and $step = l$ **do**
  for $j = 0$ to m and $step = l$ **do**
    $I[i : (i + l), j : (j + l)] = f(S[index])$
    $index = index + 1$
  end for
end for

The hexadecimal code takes 16 possible values $(0_{10}, ..., 15_{10})$. Once mapped with f, each value of the hexadecimal code corresponds to a pixel value (i.e. $\in [0, 255]$) associated to one and predefined sub range:

- $0_{10} \rightarrow_f 8 \in [0, 15]$
- $1_{10} \rightarrow_f 24 \in [16, 31]$
- ...
- $15_{10} \rightarrow_f 248 \in [240, 255]$

In addition to the mapping process, and regarding each entry of the matrix that represents the "SIP" image, the value of the associated pixel, is duplicated $l \times l$ times (redundancy). The goal and benefit of this multi-mapping is to create clusters of pixels, in order to improve the learning process of the generator. The goal is also to reduce the byte error rates, for the critical fields of the SIP packets (must be detected as a valid pattern by the generator). Figure 2 gives an example of SIP packets encoded by the algorithm 1.

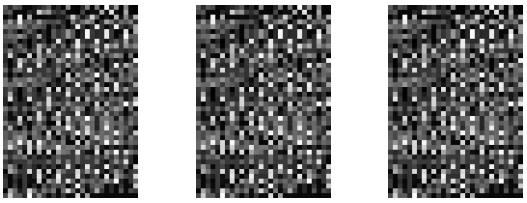


Fig. 2. INVITE SIP Requests generated by the SIP-GAN encoder

[1]Code available at `https://github.com/amarmeddahi/sip-gan`

B. Generator

The SIP-GAN generator is based on a modified and adapted DCGAN model. The SIP protocol, is a communication oriented and constrained protocol (with critical header fields for SIP routing, real time characteristics...) that force the generator to reconstitute these critical packet fields, rigorously (0 byte error). These specific and critical fields are predefined and characterized by a few bytes (ex. destination address, port number 5060, "via" field...). Figure 6 illustrates and gives an example of such critical fields (ex. REQUEST Line and other SIP header fields)

GANs, by nature do not consider this type of SIP communication oriented protocol with specific content or characteristics. Indeed, some specific pixels of the generated "SIP" image, may be inconsistent with the rest, without affecting or impacting the global quality of the generation. In SIP context, a false generated pixel in the Request Line field, implies that the packet will not be recognized or detected as a SIP packet and routed (by the SIP Proxy). For that, we propose to adapt the training of the generator by introducing a quality factor γ related to the critical fields. The principle is as follows: at each iteration of the generator training, the quality factor γ is computed on the generated samples by assessing the quality of the generator for this iteration. We set an ϵ and a maximum number of iterations K for the training stop condition to be satisfied ($\gamma \leq \epsilon \vee k \geq K$). The SIP-GAN generator training process is illustrated in figure 3.

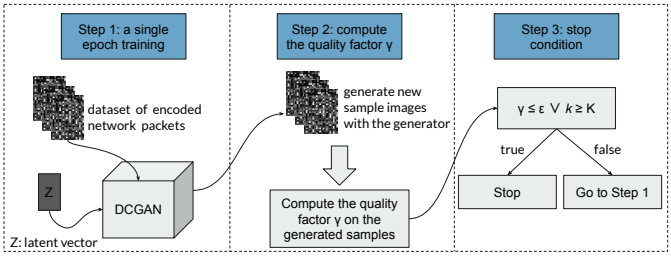


Fig. 3. The SIP-GAN generator training process

In our SIP case, this factor corresponds to the average byte error rate on the SIP Request Line of the generated packets, on a given iteration of the generator training phase. The training process of the proposed generator can be extended to other types of network protocols with such specific critical fields by adapting the ad hoc parameters. The approach is generic and gives a better control over the performance of the generator learning process.

C. Decoder

SIP-GAN decoder creates a pcap file from the generated images. The reverse mapping function f^{-1} (3) and the multi-mapping parameter l are used to convert pixels into bytes.

$$\begin{array}{rcccl} f^{-1} & : & [0,255] & \rightarrow & [0,15] \\ & & x & \mapsto & \left\lfloor \dfrac{x-8}{16} + \dfrac{1}{2} \right\rfloor \end{array} \quad (3)$$

The generator produces grayscale images with values in $[0, 255]$. The sub-intervals defined by the encoder allow to decode the correct hexadecimal code or value, even if this value is not "perfectly" reproduced or generated by the generator. For example, for all values in $[32, 47]$ the reverse mapping function f^{-1} (3) returns the correct value of 2. Algorithm 2 illustrates the proposed method for decoding the generated images.

Algorithm 2 SIP-GAN decoder
Input: I: grayscale image corresponding to a SIP packet, l: multi-mapping parameter
Output: S: a vector corresponding to a SIP packet
Initialization : $S \in \mathbb{R}^p$, $index = 0$
for $i = 0$ to n and $step = l$ **do**
  for $j = 0$ to m and $step = l$ **do**
    $S[index] = f^{-1}(\frac{1}{l^2} \sum_{a=i}^{i+l} \sum_{b=j}^{j+l} I[a, b])$
    $index = index + 1$
  end for
end for
return S

IV. Evaluation

We evaluate the similarity and thevalidity of the SIP-GAN and provide experimental results. Namely, we evaluate similarity between the generated data and the original data, and their compliance with the SIP syntax format or standards (RFC 3261, 4566).

A. Dataset creation

To the best of our knowledge, the number of datasets that are available or relevant in the network field and more particular in SIP context, are very limited or does not exist for SIP. Moreover, these datasets need to be adapted, in order to be combined with a SIP oriented protocol, to be compliant with SIP specific constraints (at protocol level but also at the implementation level). Therefore, we set up a protocol to create an adapted dataset. For this, we first consider the SIP message (INVITE packet) to validate our approach before considering its extension, to support other types of SIP messages or scenarios.

To create this dataset, we first collect and analyze the different bytes, corresponding to a set of different INVITE type of requests (20 000). Then, we exploit the raw data to perform data processing and calculations. The dataset creation process is summarized and illustrated figure 4.

1) Data Acquisition: The data acquisition is based on a different technique, in order to get a SIP compatible format (text, ASCII) that can be processed and delivered by the SIP-GAN modules or software agents. For this, the different bytes of the various SIP requests are collected and analyzed. SIPp and TCPDUMP tool kits and software are used respectively, to generate numerous SIP calls (20000 requests in a first step) and to collect the dataset as a pcap file (network level analysis). This type of file is suitable for data related to a network

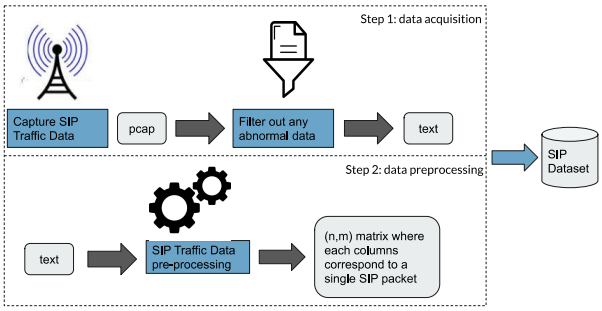


Fig. 4. SIP-GAN dataset creation and process

communication. Ethernet, IP and UDP headers are kept for technical and practical considerations.

The pcap file is analyzed (Wireshark) in order to filter out any abnormal data and then exported to a text file. This text file contains the information associated with each packet (hexadecimal format).

2) Data Preprocessing: After data acquisition, the preprocessing is done via a Python software agent to convert the raw data into a matrix $n \times m$ where each column corresponds to one SIP request (INVITE). As the length of a SIP packet, or a network packet size are variable, a padding is done to obtain a fixed size for that each vector, that contains the data packet.

B. Performance Metrics and Experimental Conditions

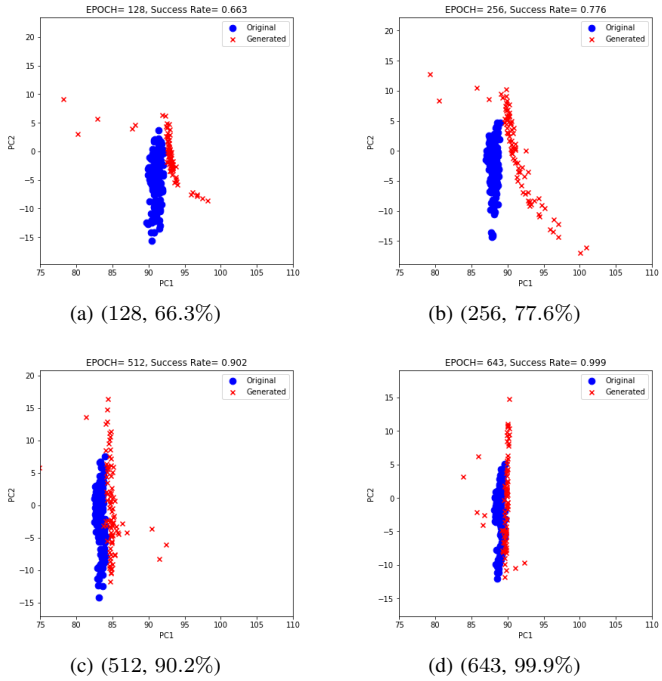
There are a number of well-known metrics, for determining the performance of a GAN in computer vision in the literature such as: Frechet Inception Distance (FID), Annealed Importance Sampling... However, in the field of data networks, these metrics are not adapted and specific metrics need to be defined for each approach, to assess the validity of the generated data. In this context, we consider and define three types of metric:

- Principal Component Analysis (PCA) between the real data and the generated data.
- SIP success rate, which is the number of packets that are successfully recognized as SIP messages by a packet analyzer divided by the total number of packets generated.
- SIP byte error rate noted γ, which is the number of byte values or fields in a packet, that is not "correctly" generated (i.e. non-compliant to SIP network standards).

Our simulations were conducted on a PC with an AMD Ryzen 9 16-Core Processor 3.40 GHz and 128 GB RAM. We used tensorflow GPU on a NVIDIA GeForce RTX 3090. Wireshark version 3.4.7 is used to analyze and verify the number of request messages that are compliant with SIP standards.

C. Experimental Results

For the evaluation of SIP-GAN and in a first step, in order to prove the similarity of the original data and the generated ones, we used PCA (each point represents a SIP packet projected in a 2-D data space) to verify and represent similarity, visually. Figure 5 shows that during training iterations, the similarity between the original and the generated data is increasing for each iteration. Also, when the stop condition is met ($\gamma \leq \epsilon$ with $\epsilon = 10^{-4}$) at iteration 643 the original data and generated ones match perfectly.



(a) (128, 66.3%) (b) (256, 77.6%)

(c) (512, 90.2%) (d) (643, 99.9%)

Fig. 5. PCA(n=2) for original vs generated SIP(training iteration,success rate)

For the second step of the evaluation, we compare an original SIP packet coming from the SIP dataset, with a generated one. Figure 6 shows that all of the critical fields (in bold type) in the SIP request line and the SIP header, are perfectly generated by SIP-GAN. The figure also shows, that SIP-GAN is able to guarantee the uniqueness of other critical fields (e.g. branch, tag, Call-ID...) which is an essential property to provide a SIP generator that simulates "real" sip communications, in practical conditions or environments.

INVITE sip:service@10.0.0.1:5060 SIP/2.0	**INVITE sip:service@10.0.0.1:5060 SIP/2.0**	SIP Request Line
Via: SIP/2.0/UDP 10.0.2.15:5060;branch=z9hG4bK-3091-1-0 From: sipp <sip:sipp@10.0.2.15:5060>;tag=3091SIPpTag001 To: service <sip:service@10.0.0.1:5060> Call-ID: 1-3091@10.0.2.15 CSeq: 1 INVITE Contact: sip:sipp@10.0.2.15:5060 Max-Forwards: 70 Subject: Performance Test Content-Type: application/sdp Content-Length: 129	**Via: SIP/2.0/UDP10.0.2.15:5060;** branch=z9hG4bK-3580-6881-0 **From: sipp<sip:sipp@10.0.2.15:5060>;** tag=3580SIPpTag001359 **To: service <sip:service@10.0.0.1:5060>** Call-ID: 1449-3580@10.0.2.15 CSeq: 1 INVITE **Contact: sip:sipp@10.0.2.15:5060** Max-Forwards: 70 Subject: Performance Test **Content-Type: application/sdp** Content-Length: 128	SIP Header
v=0 o=user1 53655765 2353687637 IN IP4 10.0.2.15 s=- c=IN IP4 10.0.2.15 t=0 0 m=audio 6000 RTP/AVP 0 a=rtpmap:0 PCMU/8000	v=0 o=user1 53655765 2353687637 IN IP4 10.0.2.15 s=- c=IN IP4 10.0.2.15 t=0 0 m=audio 6000 RTP/AVP 0 a=rtpmap:0 PCMU/8000	SIP Body

Fig. 6. SIP content comparison: original vs generated SIP packet

In the last step of our evaluation, we study the evolution of the quality factor γ and the SIP success rate, as a function of the number of learning iterations (e.g. EPOCH). We notice that the gamma quality factor (i.e. the byte error rate on the

SIP Request Line) converges to zero but with a significant fluctuation. This instability results in the SIP success rate, which also shows a fluctuation but, as the quality factor converges to zero, the SIP success rate converges to 1 (i.e. all generated SIP packets are detected as SIP compliant packets). Figure 7 shows, for example, that the SIP byte error for the SIP Request Line is greater at iteration 230 than at iteration 220 whereas, intuitively, one might think that the greater the number of iterations of training is important, the better the quality of the generated packets is. Also, when we define an ϵ threshold, we see that the stop condition is met, only at iteration 643. Our training method provides a stop condition at this iteration while giving a success rate of 99.9% for the generated SIP packets.

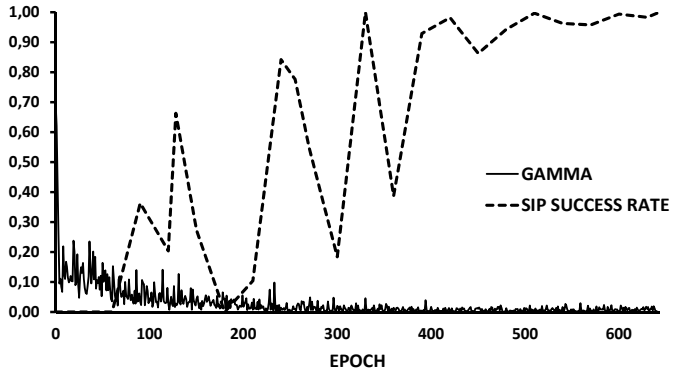


Fig. 7. γ and SIP success rate as function of EPOCHS

V. Conclusion

We propose and prototype a GAN based model to generate realistic SIP network traffic at the packet level for data augmentation. The use of GANs to generate SIP packets for data augmentation is relatively new. Also, when we compare to existing approaches, our contribution can be extended to other communication or transaction oriented protocols (generic approach) and consider traffic generation at both packet level and network level. We introduce an original approach for extracting, pre-processing and decoding network data specifically based on GANs model, that addresses the specific constraints of the network packets and protocols. Considering SIP INVITE packets generation, our experimental testbed and performance results, show that SIP-GAN provides a consistent, relevant and promising solution, for SIP traffic generation in certain conditions and constrained environments (resources, scaling, network traffic...). This contribution paved the way for future development of our GAN-based SIP traffic generators. The SIP-GAN model, addresses in a first step, the generation of individual non-sequential network packets. As future work and perspective, we plan to generate more complex SIP scenarios or attacks (ex. SIP fake register, fake bye, SIP DOS...) that take into account the temporal aspect of the communication or call flow, with different distributions. Extending SIP-GAN not only to generate different SIP attack scenarios, but also to identify and classify the SIP traffic behavior (normal and abnormal) constitutes also a relevant perspective. Indeed, we envision to exploit GAN in network security context, not only to generate and detect the existing, typical SIP attacks, but also to address the future potential ones, particularly in constrained network environments.

References

[1] Adriel 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)*. 2019.

[2] Qiumei Cheng et al. *Packet-Level Adversarial Network Traffic Crafting using Sequence Generative Adversarial Networks*. 2021.

[3] Baik Dowoo, Yujin Jung, and Changhee Choi. "Pcap-GAN: Packet Capture File Generator by Style-Based Generative Adversarial Networks". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2019.

[4] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. 2014.

[5] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[6] Ahmed Meddahi and Gilles Vanwormhoudt. *Téléphonie SIP concepts, usages et programmation en Java*. Hermès - Lavoisier, 2012.

[7] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations*. 2016.

[8] Maria Rigaki and Sebastian Garcia. "Bringing a GAN to a Knife-Fight: Adapting Malware Communication to Avoid Detection". In: *2018 IEEE Security and Privacy Workshops (SPW)*. 2018.

[9] Tim Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*. 2016.

[10] Hongwei Wang et al. "GraphGAN: Graph Representation Learning With Generative Adversarial Nets". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[11] Chuanlong Yin et al. "An enhancing framework for botnet detection using generative adversarial networks". In: *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. 2018.

[12] Lantao Yu et al. "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2017.