SMAUG: Streaming Media Augmentation Using CGANs as a Defence Against Video Fingerprinting

Alexander Vaskevich,* Thilini Dahanayaka,* Guillaume Jourjon,† Suranga Seneviratne*

* University of Sydney, Australia, † CSIRO, Space & Astronomy,
Email: {first name.last name}@sydney.edu.au, {firstname.lastname}@csiro.au

***Abstract*—Traffic fingerprinting and developing defenses against it has always been an arms race between the attackers and the defenders. The rapid evolution of deep learning methods makes developing stronger traffic fingerprinting models much easier, while overhead, latency, and deployment constraints restrict the abilities of the defenses. As such, there is always the need of coming up with novel defenses against traffic fingerprinting. In this paper, we propose SMAUG, a novel CGAN-based (Conditional Generative Adversarial Network) defense to protect video streaming traffic against fingerprinting. We first assess the performance of various GANs in video streaming traffic synthesis using multiple GAN quality metrics and show that CGAN outperforms other types of GANs such as basic GANs and WGANs (Wasserstein GAN). Our proposed defense, SMAUG, uses CGANs to synthesize video traffic flows and use those synthesized flows to camouflage the original traffic that needs protection. We compare SMAUG with other state-of-the-art defenses - FPA and d*-private methods, as well as a kernel density estimation-based baseline and show that SMAUG provides better privacy with lower overhead and delay.**

I. Introduction

Around 90% of the global web traffic is currently end-to-end encrypted using HTTPS as opposed to the 50% in 2017 [1], implying a rapid growth. There has also been increased user interest in utilizing end-to-end encrypted communications not only for web browsing but also for other related services that were known to leak information such as DNS [2]. Nonetheless, various internet protocols are known to leak information through side-channels, despite being end-to-end encrypted. In fact, multiple papers demonstrated the possibility of using packet sizes, packet direction, and inter-packet times to infer the underlying content of encrypted communications such as websites visited [3], videos being watched [4], specific messenger app activities [5], and even detecting specific words in encrypted VoIP traffic [6].

To mitigate the effects of side-channel attacks, various papers proposed defenses that attempt to obfuscate distinguishable statistical patterns of original content appearing in encrypted traffic by adding dummy data and/or packet delays to the original content before encryption, at the cost of data and timing overhead. For example, defenses such as BuFLO [7], Walkie-Talkie [8], and FPA method [9] add both dummy data and packet delays while defenses such as WTF-PAD [10] and FRONT [11] add only dummy data. Nonetheless, side-channel attacks and defenses has always been an arms race between the attackers and defenders, where a more powerful traffic fingerprinting method always beats the state-of-the-art defense. With the advances in deep learning, we observe more and more fingerprinting methods being proposed [3], [12]. As such, it is required to assess and revisit existing defenses and propose new ones to circumvent the threats of traffic fingerprinting.

978-1-6654-9550-9/21/$31.00 ©2021 IEEE

To this end, we propose **SMAUG** - **S**treaming **M**edia **A**ugmentation **U**sing C**G**ANs, a novel defense mechanism to defend video streaming traffic against traffic fingerprinting, using Generative Adversarial Networks (GANs) [13]. While GANs have been used extensively in computer vision tasks [14], their applications in network traffic have been explored only in limited settings [15], [16]. We leverage a Conditional Generative Adversarial Network (CGAN) [17] to generate synthetic video streaming traffic flows and use them to morph original traffic into different patterns so that an attacker cannot build a traffic classifier. To the best of our knowledge, our paper is the first to use CGANs as a defense mechanism against traffic fingerprinting. More specifically, we make the following contributions.

- We investigate the use of different GANs in network traffic synthesis and show that Conditional GANs produce more realistic results and outperform other GAN types according to several GAN quality measurement metrics.
- We propose SMAUG, a novel CGAN-based traffic morphing method that allows protecting video streaming from traffic fingerprinting attacks. We evaluate SMAUG's effectiveness in protecting video traffic using two real-world datasets. Our results show that using CGAN synthesized traces in the morphing process drops the attacker accuracy by approximately 4%–8%.
- We compare SMAUG with a Kernel Density Estimation (KDE)-based baseline and two state-of-the-art methods, FPA and d*-privacy, and show that SMAUG provides a better balance between privacy, overhead, and delay.

The rest of the paper is organized as follows. In Section II, we present related work and in Section III we present the background information, datasets, and the overall design of our solution. We compare the performance of various GANs in traffic synthesis in Section IV. Our proposed defense and its performance results are presented in Section V. In Section VI we discuss implications, limitations, and possible future extensions of our work. Section VII concludes the paper.

II. Related Work

We first present prior work on traffic fingerprinting attacks and then discuss key defenses proposed against such attacks highlighting those on video streaming. Finally we present work that used GANs in the context of network traffic classification.

A. *Traffic fingerprinting attacks*

As discussed earlier, statistical properties of encrypted flows can be used to conduct traffic fingerprinting attacks undermining the privacy expectations of end-to-end encryption.

i) Website fingerprinting and inferring user activities: Initial papers on website fingerprinting used traditional machine learning techniques such as Naïve-Bayes [18] and SVMs [19]. Recent papers [3], [12] adopted deep learning methods for website fingerprinting with results over 90% accuracy in identifying the websites users are visiting. Apart from website fingerprinting, attacks have also been demonstrated for messenger apps [5], smart assistants [20], and VoIP traffic [21].

ii) Video streaming fingerprinting: Schuster et al. [4] used a CNN to fingerprint video streaming traffic from YouTube, Amazon, Vimeo, and Netflix and achieved $\sim$95% accuracy. Li et al. [22] showed the same possibility, yet over encrypted WiFi traffic using a Multi Layer Perceptron (MLP) model. Reed and Kranch [23] identified Netflix videos with over 99.99% accuracy only using the TCP/IP header information.

B. *Defenses against traffic fingerprinting attacks*

Several papers focused on developing defenses against traffic fingerprinting attacks. The key idea behind all such defenses is to obfuscate statistical properties of traffic flows by adding dummy packets and/or delaying packets before encryption.

i) Defenses against website fingerprinting: BuFLO [7], one of the very first defenses against website fingerprinting enforces fixed packet lengths and fixed inter packet intervals and provides high privacy but incurs significantly high data and timing overhead. CS-BuFLO [24] and Tamaraw [25] are two extensions of BuFLO. In Walkie-Talkie [8], the browser communicates with servers in half duplex mode and adds dummy packets and delays to create confusions between traces. Nonetheless, it still has high overhead and requires significant changes to the existing infrastructure. In contrast, WTF-PAD [10] uses adaptive padding and incurs zero timing overhead and relatively low data overhead. Nonetheless, later it was shown to be ineffective against deep learning based classifiers [3]. The current state-of-the-art defense FRONT [11] focuses on obfuscating trace fronts by using a Rayleigh distribution to determine the timing for dummy packets and provides acceptable privacy with much less data overhead compared to previous defenses. More recent papers defended Tor traffic in real-time by leveraging *universal adversarial perturbations* [26] and *input agnostic adversarial patches* [27] to pre-compute dummy packet sequences, incurring relatively low data overhead compared to earlier defenses.

ii) Defenses against video streaming fingerprinting: Zhang et al. [9] explored how ideas from differential privacy could be used to defend against video streaming fingerprinting attacks. The authors used two ϵ-differentially private mechanisms, Fourier Perturbation Algorithm (FPA) and d*-private mechanism to add dummy packets to streaming traffic. While the defense based on the FPA algorithm significantly reduced attacker's accuracy, it requires the entire trace to be known in advance. On the other hand, d*-private defense incurs significantly high overhead despite providing reasonable privacy.

C. *GANs in traffic classification and defenses*

With respect to traffic classification attacks, GANs can be especially helpful when there is a lack of training data. For example, GANDalf [15] used GANs to generate synthesized training data for website fingerprinting such that classifier training can be done with as few as 20 real samples per class (total of 100 classes) and still achieve 87% accuracy, surpassing *Triplet Fingerprinting* [28] which used N-shot learning for the same problem. Similarly, GANs have also been used to improve the performance of intrusion detection systems [29], [30] and malware detection systems [31], [32] by augmenting the training sets with adversarial samples.

GANs have also been used to defend against traffic fingerprinting by using them to generate adversarial samples. Unlike most defenses that leveraged adversarial samples using gradient based and optimization based methods, only a few such as [33] and [34] that defend against website fingerprinting and [16] and [35] that defend against malware detection systems and intrusion detection systems, respectively used GANs. *In contrast, our work is focused on defending streaming video traffic, rather than web traffic, using GANs.*

III. Background and Solution Overview

A. *Generative Adversarial Networks (GANs)*

While some of the fundamental ideas of GANs were there for some time, the first key paper in GANs was the work of Goodfellow et al. [36]. The authors simultaneously trained two neural networks; a *generator model (G)* that captures the distribution of the input data and a *discriminator model (D)* that can determine whether a given sample is real or synthesized by G. During training, the minimax GAN loss encourages G to generate samples that D cannot distinguish as coming from original input data or from G while at the same time, encouraging D to correctly distinguish samples generated by G. Currently, GANs are mainly used in computer vision, such as in image synthesis, super resolution, and de-noising. Only very recently did papers start emerging in using GANs in other domains such as time series data generation [37].

Multiple subsequent papers proposed changes to the base GAN to address some of its limitations. Radford et al. [38] proposed *Deep Convolutional Generative Adversarial Network (DCGAN)* which uses deep CNN architectures for both G and D and found that it avoids the *mode collapse* problem (i.e., G producing the same or a small set of outputs without diversity).

The *Wasserstein Generative Adversarial Network (WGAN)* extends regular GANs by altering the training procedure so that D is updated many more times than G for each iteration. Furthermore, D was updated to output a real-value (linear activation) instead of a binary prediction with a sigmoid activation while both G and D were trained using *Wasserstein loss* which is the average of the product of real and predicted values from D in order to provide linear gradients that are useful for updating the model. By using the Wasserstein Distance in the loss function, WGANs provide more stability during training, while avoiding mode collapse.

Conditional Generative Adversarial Networks (CGAN), another variant of GANs, use additional information other than just the input noise vector to generate new data. One possible additional information is class labels that can be used to condition G and D to generate samples of a specific class, unlike other GAN variants where the class of the sample generated cannot be predicted in advance.

B. Threat Model and Solution

Our threat model assumes an attacker who tries to predict the exact video streamed by a target user based on passively observed encrypted traffic only as illustrated in Fig. 1-(a). First, the attacker collects data of themselves playing target videos multiple times and uses them to train a classifier model. Next, the attacker uses the trained model on network traffic of a target user to predict the streamed video. Such attacks have been demonstrated in [4] and [22]. While the attacker can use any machine learning model as the classifier, for clarity we assume the attacker is using a CNN as the classifier. In the Appendix, we show that our proposed defense is equally effective against other classifier models, such as Long Short Term Memory (LSTM) networks, AdaBoost, and MLPs.

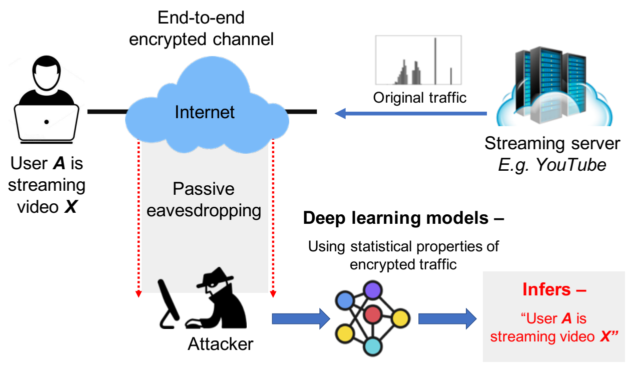
In this paper, we propose a defense against such attacks where our key idea is to use a GAN synthesized trace to hide the original trace from the attacker as illustrated in Fig. 1-(b). The SMAUG algorithm takes the trace that needs to be defended and a GAN synthesized trace from a different class as inputs and morphs the original trace to look like the synthesized trace, ensuring that no data is lost and the receiver can still recover the original message, and without adding significant delays or overhead to the traffic flow.

We assess four GAN archetypes for traffic synthesis: the original GAN [36], DCGAN [38], WGAN [39], and CGAN [17]. As we show later, CGAN outperforms the others and thus we use CGANs for the SMAUG algorithm.

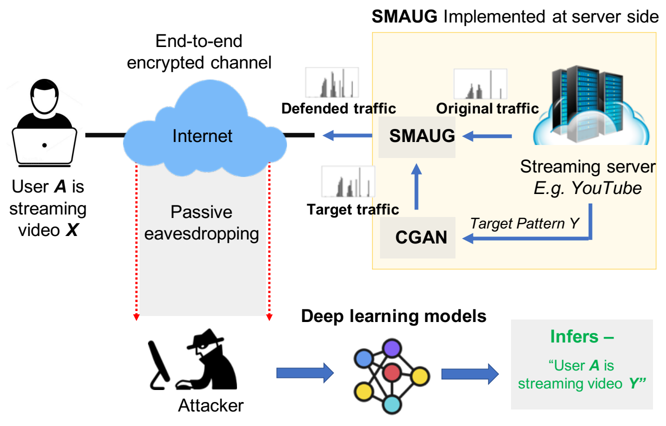
C. Datasets

Finally, we need encrypted video streaming traffic datasets to assess the performance of our defense. To this end, we use two publicly available datasets as described below.

i) Deep Content dataset (DC) [22] comprises of traffic traces for the first three minutes of 10 *YouTube* videos collected at the data-link layer (WiFi). Each video has 320 traces corresponding to different replays. For each trace, the three minute interval is binned into 500 time slots (0.36s each) and



(a) Undefended scenario



(b) Defended scenario

Fig. 1: Threat model and defense in operation

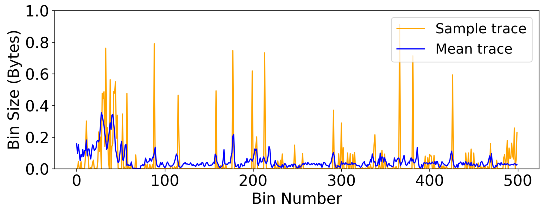
each time slot is represented by summary statistics of packets observed during that time. Out of the 24 features given in the original dataset, we use the features *Number of bytes in uplink* and *Number of bytes in downlink* in our experiments.

ii) SETA: Scalable Encrypted Traffic Analytics dataset [40] comprises of traffic traces containing the first 64 bytes of each outgoing and incoming packet sequences for 20 selected *Netflix* videos. It contains 100 traces per video in the same binned format as the DC dataset.

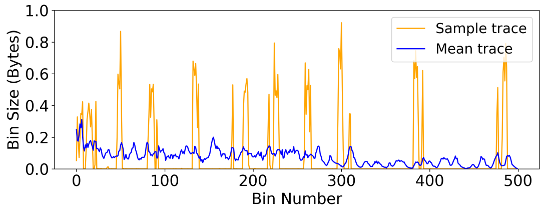
Since we use two features corresponding to uplink and downlink for each dataset, we effectively ended up with 4 datasets: DC Up, DC Down, SETA Up, and SETA Down. In Fig. 2 we visualize a sample trace and the mean trace for each dataset. Note that the DC dataset on average is more "active" than SETA (i.e. the blue curve in Fig. 2 goes to zero more frequently for the SETA dataset while it is rare in the DC dataset). Due to this, DC dataset has more frequent peaks which are also larger in magnitude than those in SETA. This is an observation we will be drawing upon later to aid with the explanation of some results.

IV. GANs for Traffic Synthesis

We first investigate which GAN architecture is the most suitable for synthesizing network traffic. Previous papers in



(a) Mean and sample traces from the SETA Down dataset



(b) Mean and sample traces from the DC Down dataset

Fig. 2: Mean and sample traces from DC and SETA datasets

network traffic synthesis [41], [42] rarely used any evaluation metrics directly for their GANs. Instead, performance was measured through methods specific to the experiment, such as the percentage of fake packets that generated a successful web response [41], and, when a performance metric specific for the GAN was utilized, it was only a variation of the GAN Quality Index (i.e., Train-on-Synthetic data and Test-on-Real data) [42]. In contrast, we leverage GAN performance metrics such as Inception Score [43] and Fréchet Inception Distance [44] in addition to the GAN Quality Index [45]. To the best of our knowledge, our paper is the first to conduct a methodical and rigorous approach to measure the GAN performance for network traffic synthesis. We next describe the GAN performance metrics we used.

A. GAN Performance Metrics

i) Inception Score (IS) - The Inception Score (IS) [43] as defined in (1) is a metric proposed in response to the demand for an objective GAN performance metric. It measures two key characteristics of GAN-synthesized samples; classifier confidence and the diversity of synthesized samples.

$$IS(G) = exp(\mathbb{E}_x D_{KL}(p(y|x)||p(y))) \tag{1}$$

1) *Classifier confidence* - $p(y|x)$ in (1) defines the probability that a sample belongs to a given class (i.e. the "quality" of the sample). This can be calculated by using a state-of-the-art classifier to make predictions on synthesized data. For instance, in [43] the authors used the then state-of-the-art Inception v3 model to classify synthesized samples and recorded the prediction probabilities as $p(y|x)$. In our case, instead of Inception v3 we use the custom CNNs used for video traffic fingerprinting proposed in [22], [40].
2) *Diversity* - $p(y)$ in (1) defines the diversity of the synthesized samples. A GAN that produces samples only from a single class will have a lower diversity, whereas a GAN producing similar number of samples from all the classes in the training data will have a higher diversity.

Finally, the expected KL divergence is calculated over these two metrics to obtain the final score, which ranges from 1.0 (lowest) to n, where n is the number of classes in the dataset.

ii) Fréchet Inception Distance (FID) - The Fréchet Inception Distance defined in (2), was proposed by Heusel et al. [44] as an improvement to the Inception Score.

$$FID = ||\mu - \mu_g||^2 + tr(\Sigma + \Sigma_g - 2(\Sigma\Sigma_g)^{1/2}) \tag{2}$$

where tr(A) is the trace of matrix A.

Unlike IS, FID compares the statistical properties of synthesized samples to those of the real data distribution by measuring the following two properties.

- The squared sum of the difference between the mean values of real and generated features calculated over all the features.
- The square root of the dot product between the covariance matrices for the features of the real and synthesized data samples. This factor compares the original feature distribution to the synthesised one - it can be thought of as a measurement testing the extent to which the GAN is experiencing mode collapse.

Here, a lower FID indicates a better GAN performance.

iii) GAN Quality Index (GQI) - The idea behind GAN Quality Index [45] is that if the "quality" of the samples (as per some arbitrary definition of quality as observed by a classifier such as a CNN) generated by the GAN is high, and there are enough samples being generated from each class, then the classifier trained on the synthesized samples should be able to perform well when tasked with classifying real data. Accordingly, the GQI is calculated as follows.

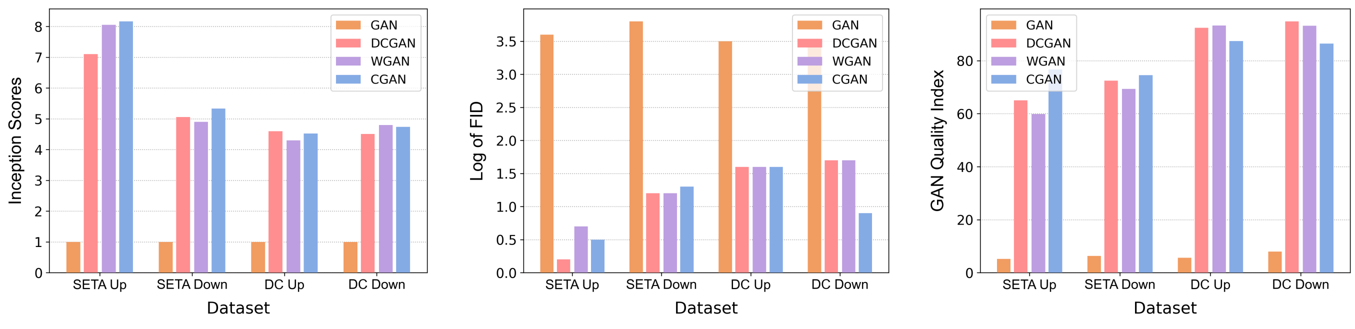
$$GQI = \frac{ACC(C_{S_g})}{ACC(C_{S_r})} * 100 \tag{3}$$

where $ACC(C_{S_g})$ is the accuracy of a classifier trained on a synthesized set of samples S_g and tested on a real set of samples S_r, and $ACC(C_{S_r})$ is the accuracy of a classifier trained and tested on subsets of S_r. The closer the GQI is to 100, the more comparable the performance of a classifier trained on synthetic data is to that of a classifier trained on real data and thus, the GAN can be considered as having a better performance.

B. Performance of Various GANs in Traffic Synthesis

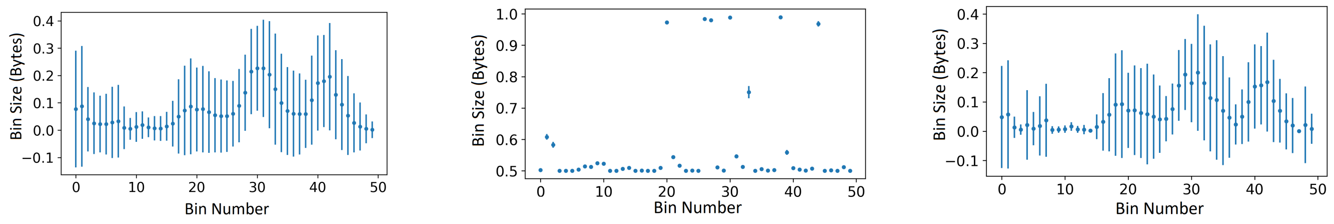
We trained the GAN types mentioned in Section III-B using all available samples from each dataset. Afterwards, we generated the same number of traces for each dataset as was used during training, both for the purpose of simplifying the calculation of the FID score, as well as for fairness in terms of the GQI score, as the original CNN (whose accuracy comprises the $ACC(C_{s_r})$ part of GQI) have been trained on 2,000 and 3,200 samples respectively.

In Fig. 3-(a), (b), and (c) we show the three GAN performance metrics for each GAN variant, grouped by the dataset.



(a) Inception Score (b) Fréchet Inception Distance (c) GAN Quality Index

Fig. 3: Inception, FID and GQI Scores grouped by dataset.



(a) Original distribution (b) GAN synthesized distribution (c) DCGAN synthesized distribution

Fig. 4: Means and standard deviations of the original, GAN, and DCGAN sample distributions.

Results show that the base GAN performs worse compared to other models across all datasets and metrics. We note that as mentioned in Section IV-A, a lower value for IS and GQI or a higher value for FID indicates a poor performance. Other GAN types have approximately same performances across all metrics and datasets.

The poor performance of base GANs can be attributed to its susceptibility to *mode collapse*. Mode collapse is a phenomenon observed in GANs wherein the Generator fails to produce a diverse range of outputs, rather creating outputs that are very similar or even identical to one another. In most cases, this is due to the Generator being able to fool the Discriminator by perfecting the look of a single sample, thus "winning" the game at the cost of diversity. In other scenarios, the Discriminator might get so adept at discerning the synthesized samples provided by the GAN from real ones, that the Generator is no longer able to receive any meaningful feedback from the Discriminator and as a result cannot improve. This problem is common with the base GAN in image generation, and is one of the things that architectures such as the DCGAN and the WGAN have attempted to resolve [38], [39].

In Fig. 4, we show example evidence of mode collapse of base GANs. The figure shows the box plots of the first 50 bins of 2,000 samples each from the SETA UP dataset, synthesized from the base GAN, and synthesized from a DCGAN, respectively. From Fig. 4-(b) it is clear that the GAN is generating traces that look almost identical to each other, as evidenced by the almost non-existent inter-quartile range for the vast majority of the 50 bins. In contrast, the DCGAN (i.e., Fig. 4-(c)) have significant inter-quartile ranges similar to the real data (i.e., Fig. 4-(a)), indicating synthesized samples are sufficiently different from each other.

C. Selecting a GAN type for SMAUG

Though other GAN types show better performance than the base GAN in Fig. 3, they are not without their own issues. For example, we used the DCGAN and the WGAN to synthesize 1,000 samples each from the SETA Up dataset, and conducted following two experiments:

1) We trained a CNN on the synthesized samples and used it to classify our real dataset. In this case, we noticed that the CNN achieved a satisfactory accuracy (as reflected by the GQI score in Fig. 3). Nonetheless, the confusion matrix revealed that one of the classes, *class 8*, was misclassified almost all the time.
2) We trained a CNN on real data and had it classify 10,000 synthesized samples. Here, we noticed that both DCGAN and WGAN synthesized significantly less number of samples from class 8 as seen in Fig. 5, where only a small fraction of classified samples belong to *class 8*.

A closer look at the dataset revealed that *class 8* is too statistically dissimilar to the rest of the classes, to the point where the Generators could not learn a representation of it that would trick the Discriminator, resulted in the abandonment of *class 8* altogether. We observed the same with both the DCGAN and WGAN. Between DCGAN and WGAN, while they have

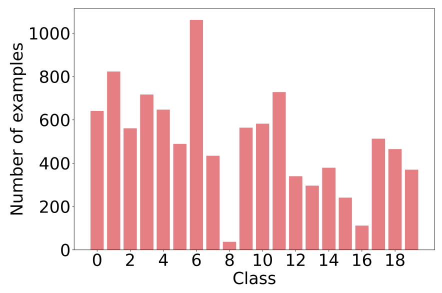


Fig. 5: DCGAN synthesized samples class distribution.

comparable performances, WGAN takes a significantly longer time to train. As a result, we decided against using WGAN.

Referring to back Fig. 3, we observe that the CGAN comes out on top in terms of performance more frequently than others, and in some cases such as the GQI graph for SETA Up, show significantly higher performance. Moreover, the CGAN trivially does not face the class distribution problems observed in other models as the class that the CGAN produces a sample of is determined by the target class input that is fed into it. Hence, for the design of SMAUG defense, we used CGANs.

V. Proposed Defence & Results

As mentioned in Section III-B, our end goal is to build a defense strategy to provide privacy in video streaming under the attack model described in Section III-B.

There are a few design goals to consider when designing a defense against traffic fingerprinting. First, when the defense is applied on real traffic it must significantly drop the accuracy of the attacker classifier. This must be true even if the attacker trains a new model by collecting large volumes of newly defended traffic. Second, is the trade-offs between overhead and delay, and the level of privacy provided by the defense. For example, a naive defense of making all the packets the same size will consume unnecessarily large bandwidth despite providing perfect privacy. Equally, the defense must not delay packets too much because that will adversely affect the end user's quality of experience.Having these design goals in mind, we propose **SMAUG** - **S**treaming **M**edia **A**ugmentation **U**sing C**G**ANs to defend streaming traffic based on three key ideas.

1) **CGAN synthesized target traffic template -** We utilize a CGAN to generate a "target" video traffic trace, that is then used by the defense algorithm to gauge which packets to delay and which packets to pad.
2) **Aligning activity peaks -** We delay bursts of activity from the original trace (within a certain limit) so that they coincide with activity in the target trace. Any remaining peak in the target trace, we add them to the resulting defended trace as "dummy" peaks.
3) **Adding random noise -** Optionally, noise can be added to target traces. This allows the algorithm to more effectively hide remnants of the original trace.

A. SMAUG

We present the detailed algorithm in Algorithm 1. Given a trace T_o with label L_o that we would like to defend, we first generate a target trace T_t using the CGAN with label L_t such that $L_o \neq L_t$. Optionally, the algorithm iterates through T_t and add some random noise between 0 and z to each of T_t's bins. Next, the algorithm redistributes the contents of the bins of T_o, moving in chronological order and only looking b bins ahead so as to make T_o resemble T_t, resulting in the defended trace T_r.

One of the core steps of the algorithm involves "aligning" the peaks of T_o with those of T_t (i.e., lines 7-19). This is an iterative process in which the contents of the bins of T_o are delayed in such a way that their destination time slots correspond to time slots in T_t that also have some level of activity as follows.

1) In a situation where a peak of T_o is smaller in size than a corresponding peak in T_t, we pad the bins of T_o up to the level of T_t. Fig. 6-(a) illustrates this idea. The original trace peak, in blue, is delayed in order to be hidden behind the target trace peak, in orange (the maximum allowed delay is b bins).
2) If the peak in T_o is greater in size, the parameter m comes into play. m is the maximum percentage by which a peak in T_t may be expanded in order to accommodate a peak from T_o. Several trials helped us conclude that a value of m that worked well was 0.1, and this was kept static across all datasets and experiments. If there is still leftover traffic in the T_o peak after this peak expansion, the remaining traffic is left undefended and unmoved. An extreme case of this, where T_t contains no peaks whatsoever, is illustrated in Fig. 6-(b). Here, the peak is left untouched and thus undefended.

After this process there will still be some peaks in T_t that are left unused. Those peaks are still added to T_r as "dummy peaks", to mislead the attacker as illustrated in Fig. 6-(c).

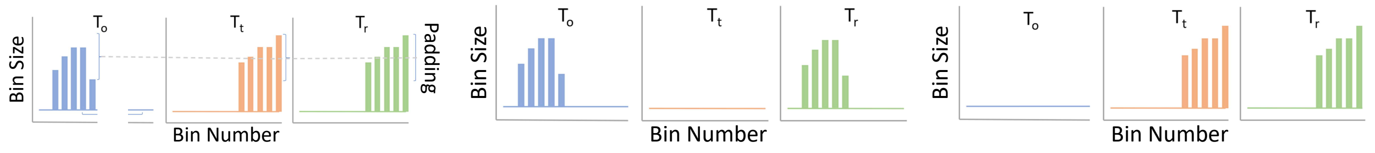
In contrast to previous papers [46], SMAUG does not assume knowledge of the entire trace that needs to be defended in advance, since this is not practical in real-world implementations. Instead, we assume that we only know b bins in advance, and follow a greedy approach for delaying and padding our packets. This approach is more practical and suits well for defending video streaming traffic. Finally, we highlight that SMAUG runs in $O(n)$ time, where n is the length of a trace and as such is efficient. The hyper-parameters of the algorithm, b and m were selected empirically.

B. Evaluation of SMAUG's Effectiveness

We evaluate the effectiveness of SMAUG in defending video streaming traffic under two attack scenarios.

1) A scenario in which the attacker is unaware of the defense being employed, and as such continues using their current CNN to classify user traffic.
2) A scenario where the attacker realizes that the traffic is being protected, and accordingly builds a new CNN-based classifier as an attempted counter-measure.

In scenario 1, we found that the attacker always achieved no better than a random-guess accuracy on defended data. For



(a) Aligning peaks (b) Peak left undefended (c) Dummy peak added

Fig. 6: An illustration of the key ideas of the algorithm.

Algorithm 1 SMAUG

```
Require: $b \geq 1, T_o, T_t, m, z$    ▷ $b$ is the max delay, $T_o$ and $T_t$ are the original and target traces, $m$ is the max allowed peak magnitude increase, $z$ is the optional noise parameter.
 1: $T_{r_i} \leftarrow 0,\ i = 0, ..., n$
 2: for $i \leftarrow 0$ to $n$ do    ▷ Optional step: add noise to target
 3:     $T_t[i] \leftarrow T_t[i] + random\ float\ between\ 0\ and\ z$
 4: for $i \leftarrow 0$ to $n$ do
 5:     $C \leftarrow \max(T_o[i] - T_t[i] + T_r[i], 0)$
 6:     $T_r[i] \leftarrow T_r[i] + T_o[i] - C$
 7:     if $i + b + 1 \geq (n)$ then    ▷ End of trace approaching
 8:         $b \leftarrow b - 1$
 9:     if $C > 0$ then
10:         if $i + 1 < n$ then
11:             $targ\_slice \leftarrow T_t[i+1 : i+1+b]$
12:             $res\_slice \leftarrow T_r[i+1 : i+1+b]$
13:             $avail \leftarrow [\max(m * x + x - y, 0)$ for $(x, y)$ in zip$(targ\_slice, res\_slice)]$
14:             $avail\_total \leftarrow$sum$(avail)$
15:             $dump \leftarrow$max$(C - avail\_total, 0)$
16:             $T_r[i] \leftarrow T_r[i] + dump$
17:             for $j \leftarrow 0$ to $b$ do
18:                 if $C \leq 0$ then
19:                     break
20:                 $T_r[i+j] \leftarrow T_r[i+j]+$ min$(avail[j], C)$
21:                 $C \leftarrow C-$ min$(avail[j], C)$
22:         else
23:             $T_r[i] \leftarrow T_r[i] + C$    ▷ Ran out of trace
24:     $T_r[i] \leftarrow \max(T_r[i], T_t[i])$
```

example, the accuracy against defended SETA Up traces was ∼5%–6%. The rest of the accuracies were similar. Therefore, for the rest of the paper we consider only scenario 2.

Additionally, when constructing the defended training dataset for the attacker, we defended each trace 10 times separately, resulting in a dataset of 20,000 samples for SETA and 32,000 for DC. This was done to impose the maximum possible disadvantage on the defender, by making the assumption that the attacker will collect and train their model on a significant volume of defended data. We used 90% of data for training and the rest of 10% of data for testing and we report the attacker's accuracy on the test set as the metric to measure the effectiveness of the defense.

TABLE I: Performance of real and synthetic targets.

Dataset	Undef.	Real	Real + Noise	Synth.	Synth. + Noise
SETA Up	98.0%	72.9%	70.9%	67.0%	55.4%
SETA Down	94.3%	50.3%	42.9%	37.1%	29.6%
DC Up	97.7%	74.2%	43.3%	67.8%	45.5%
DC Down	96.5%	74.5%	48.1%	71.9%	48.4%

1) Real vs. CGAN Generated Target Traces: SMAUG can take both CGAN-synthesized *and* real traces as targets, and can attempt to make the original trace resemble the target to the best of its ability. As such, we check whether using CGAN indeed affect the attacker accuracy. We consider both cases of with and without the optional noise addition steps. We show the results in Table I.

The results in Table I show that our algorithm is effective in thwarting attacker's attempts to compromise user privacy, whether the defense utilizes real or synthesized targets. Results further show that, when traces were protected using our algorithm with *synthesized targets*, the attacker's new model did worse than when *real targets* were used.

Further investigations showed that CGAN synthesized traces were generally around 20% larger in size than real traces. We believe that this is the key to explaining why synthesized traces help more in the defense process than real traces. We found that CGAN-generated traces have more peaks of activity and noise in general, which the algorithm can use to more effectively hide the original trace.

Finally, using CGANs to generate target traffic compared to real traces also have the extra advantage of the possibility of generating as many samples as required. If real traces are used, after some replays the defense algorithm has to repeat using traces that have been used in the past, which may assist the attacker. With CGANs, the defender can draw as many samples as required without two samples being identical.

2) SMAUG vs. SOTA: Next, we compare the performance of SMAUG with two state-of-the-art defenses; the d*-private mechanism and the Fourier Perturbation Algorithm (FPA) method, both of which were proposed by Zhang et al. [47] with the intention of enforcing differential privacy on streaming traffic. As a standard baseline we also assess the Kernel Density Estimation (KDE) for target generation than CGANs.

d-private mechanism* calculates a noise value to be added per each packet in real time using an algorithm proposed by Xiao et al. [48]. This results in the trace becoming "noisy" and hence a very high overhead in the vast majority of cases. The *FPA* method based on the paper by Rastogi et al. [46]

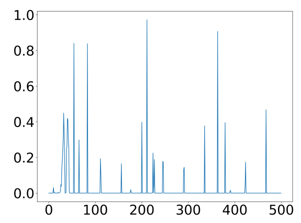
TABLE II: Comparison of various defence mechanisms.

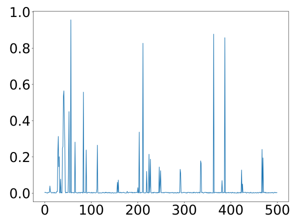
Dataset	Defence	Accuracy	Delay (bins)	Overhead
	Undefended	98.0%	0	0%
	CGAN	67.0%	2.64	36%
	CGAN + Noise	55.4%	2.29	66%
SETA Up	KDE	85.5%	2.28	45%
	KDE + Noise	79.4%	2.04	65%
	FPA	8.7%	Full trace	61%
	d*	36.7%	0	960%
	Undefended	94.3%	0	0%
	CGAN	37.1%	4.92	17%
	CGAN + Noise	29.6%	3.23	56%
SETA Down	KDE	59.9%	4.73	47%
	KDE + Noise	46.5%	4.19	60%
	FPA	7.2%	Full trace	20%
	d*	42.3%	0	395%
	Undefended	97.7%	0	0%
	CGAN	67.8%	12.9	23%
	CGAN + Noise	45.5%	11.8	58%
DC Up	KDE	89.8%	9.0	29%
	KDE + Noise	48.7%	8.2	61%
	FPA	21.3%	Full trace	32%
	d*	94.1%	0	279%
	Undefended	96.5%	0	0%
	CGAN	71.9%	9.5	14%
	CGAN + Noise	48.4%	8.7	56%
DC Down	KDE	91.5%	8.2	21%
	KDE + Noise	75.4%	7.9	63%
	FPA	26.7%	Full trace	36%
	d*	91.4%	0	234%

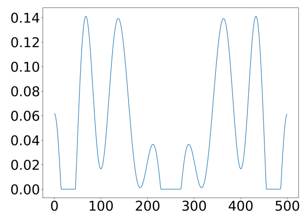
observes the entire packet sequence and then converts it to the frequency domain and retaining only the first k fourier coefficients, adds Laplacian noise to each coefficient before converting it back to the time domain. Finally, *KDE* is simply a method of estimating the probability density function of a given distribution.

We report the results in Table II. In addition to the attacker accuracy, we also report *delay* and *overhead*. *Delay* is the median number of bins that each bin gets delayed by as a result of a given defense mechanism. *Overhead* for this experiment is calculated by taking the difference between the sum of all of the defended bin sizes and the undefended bin sizes, divided by the sum of all undefended bin sizes. For overhead costs across experiments, the number we report is also the median. Note that when deciding the level of noise to add in the noised entries, we tried to add as much noise as necessary to reach an arbitrary overhead between 55-65%, and accuracies within this overhead are what we report in the table.

According to the results, FPA appears to be the best algorithm consistently giving the lowest accuracy for all the datasets. Nonetheless, FPA's assumption of knowing the entire trace in advance, limits its ability to deploy in real-world settings. KDE on the other hand performs worst. It has highest accuracy for all four datasets. Though KDE targets has comparable delays as CGAN targets, it has slightly higher overhead. This can be attributed to KDE not being as proficient as the CGAN in learning the distribution of the dataset. Finally, despite d* algorithm producing lowest accuracy in SETA Up dataset, it has significantly higher overhead in all the datasets. We show the effect of SMAUG, the d* algorithm and the FPA


(a) Original trace from SETA Up


(b) Trace defended using SMAUG


(c) Trace defended using FPA

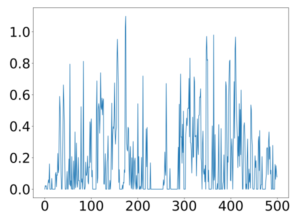
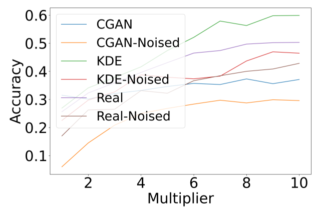
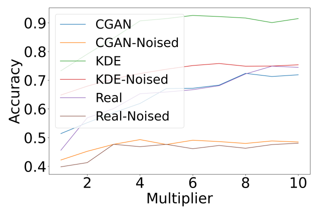

(d) Trace defended using d*

Fig. 7: Example defended traces.



(a) SETA Down



(b) DC Down

Fig. 8: Effect of attacker accumulating defended samples.

method on the original traces in Fig. 7. The figure shows that FPA distorts the signal significantly while d* method adds a significant amount of overhead (Note that the y-axis scale is different in the FPA graph). *Overall, CGAN targets provide a better trade off between defenses, delays, and overhead compared to all the other alternatives we investigated. Moreover, the optional addition of noise further improves the defense in all cases.*

3) *Longer observation periods:* Our previous results were generated assuming the attacker has access to 20,000 and 32,000 defended samples from SETA and DC datasets respectively. Next, we assess how the attacker's accuracy may increase as they get access to more defended data. We progressively create a larger training dataset by defending a single trace multiple times. For example, for the SETA Down dataset, we first defend all of the 2,000 examples from the dataset using CGAN targets and train a new attacker model. Next we progressively increase the training set size defending each training sample multiple times, thus creating new training samples. When we defend the entire dataset once, we refer to that as a "1" multiplier. When we defend it twice (for example, resulting in 2,000 * 2 = 4,000 training samples for

SETA Down), we refer to that as a "2" multiplier, and so on. We show the results in Fig. 8.

Results show that the attacker's accuracy indeed goes up when they observe more and more training samples. However, that plateaus out at 50% and the attacker can not achieve more than that even if they keep collecting defended traffic. The initial increase in accuracy is likely due to the fact that the examples generated by the CGAN are not entirely independent of each other - a property necessary to achieve true randomness. Nonetheless, despite this phenomenon our defense algorithm was still able to sufficiently obstruct the original trace from the attacker so as to make reliable classification not possible with the current attacker architecture.

VI. Discussion

In this section, we discuss the practical aspects, limitations and possible future improvements of our work.

Practical aspects - Our results show that SMAUG has less overhead and delay compared to state-of-the-art methods. Unlike other prior papers (e.g. FPA) SMAUG does not need to observe the entire trace in advance, rather requires the knowledge of next b bins. As a result, it is better suited to be deployed over DASH. Future work could explore how to further minimize this delay, for instance whether it is possible to predict the values of next b bins with reasonable accuracy to ensure near real-time defense.

Next, in our experiments, we show the effectiveness of SMAUG in separately defending both uplink and downlink traffic, yet did not explain how the solution can be practically deployed. In practice streaming server will have access to traffic patterns of its closed video catalog (e.g, Netflix) and is in a position to train the CGAN. As such, at the start of a new client session the server can generate a new CGAN sample and initiate a defense for the downlink.

Defending uplink requires some extra steps. Once the CGAN is trained the server can share it with the client. This can be done apriori and can be periodically updated to highlight the changes in the catalog. However, our design does not require the CGAN to cover the entire video catalog of the server - what is required is CGAN's ability to generate a diverse range of target traces. When the client wants to stream a new video, during the handshaking process the server can indicate which class label to give as the input to the CGAN and the client can generate a target pattern for uplink at their end. The rest of the process will be same as defending the downlink.

GAN Training - Overfitting the CGAN generating target traces on the training set may cause the generated traces to be exactly similar to training set samples of the target class. This might undermine the use of the CGAN for target generation as opposed to picking a real trace from the training set. While the target trace needs to be close to the behavior of the target class, including a certain amount of noise in the target trace will help the defending process while reducing attacker accuracy. Thus, future work need to explore methods for early stopping of CGAN training, striking a suitable balance between the similarity between actual and generated samples and the amount of noise in the generated sample.

Further efficient defenses - Other approaches that can further reduce attacker accuracy with reasonable overhead needs to be explored. For instance, future work can explore how approaches such as universal adversarial perturbations [49] and master faces [50] can be used in relation to network traffic to pre-compute perturbations that can be added to streaming traffic with minimum delays.

VII. Conclusion

In this paper, we introduced SMAUG, a novel defense mechanism to defend video streaming traffic against traffic fingerprinting using GANs. In order to construct our defense mechanism, we conducted the first comprehensive performance evaluation of different types of GANs in the context of network traffic synthesizing using state-of-the-art metrics from the machine learning community. We demonstrated that Conditional GANs (CGANs) outperform any other GANs when it comes to traffic synthesis.

We then presented our defense framework SMAUG, leveraging CGANs, and evaluated its performance against an attacker unaware and aware of the defense. We also compared SMAUG against comparable state-of-the-art defenses. Overall, we demonstrated that our solution provides better privacy as well as better delays and overhead compared to all the other alternatives we investigated.

References

[1] "HTTPS encryption on the web," 2021. [Online]. Available: https://transparencyreport.google.com/https/overview

[2] J. Bushart and C. Rossow, "Padding ain't enough: Assessing the privacy guarantees of encrypted {DNS}," in *10th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 20)*, 2020.

[3] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proc. of the 2018 ACM SIGSAC CCS*, 2018.

[4] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the Burst: Remote identification of encrypted video streams," in *26th USENIX Security Symposium*, 2017, pp. 1357–1374.

[5] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple iMessage and beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[6] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted VOIP conversations," in *2008 IEEE Symposium on Security and Privacy*.

[7] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 332–346.

[8] T. Wang and I. Goldberg, "Walkie-Talkie: An efficient defense against passive website fingerprinting attacks," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1375–1390.

[9] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang, "Statistical privacy for streaming traffic," in *Proceedings of the 26th ISOC Symposium on Network and Distributed System Security*, 2019.

[10] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 27–46.

[11] J. Gong and T. Wang, "Zero-delay lightweight defenses against website fingerprinting," in *29th USENIX Security Symposium*, 2020.

[12] V. Rimmer, D. Preuveneers, M. Juárez, T. van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *25th Annual Network and Distributed System Security Symposium*, 2018.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[14] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[15] S. E. Oh, N. Mathews, M. S. Rahman, M. Wright, and N. Hopper, "GANDaLF: GAN for data-limited fingerprinting," *Proceedings of Privacy Enhancing Technologies*, vol. 2021, no. 2, pp. 305–322, 2021.

[16] 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)*. IEEE, 2018, pp. 70–75.

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

[18] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the Multinomial Naïve-Bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud Computing Security*. ACM, 2009, pp. 31–42.

[19] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *23rd Annual Network and Distributed System Security Symposium*, 2016.

[20] C. Wang, S. Kennedy, H. Li, K. Hudson, G. Atluri, X. Wei, W. Sun, and B. Wang, "Fingerprinting encrypted voice traffic on smart speakers with deep learning," in *Proc. of the 13th ACM WiSec*, 2020.

[21] Y. Zhu and H. Fu, "Traffic analysis attacks on Skype VoIP calls," *Computer Communications*, vol. 34, no. 10, pp. 1202–1212, 2011.

[22] Y. Li, Y. Huang, R. Xu, S. Seneviratne, K. Thilakarathna, A. Cheng, D. Webb, and G. Jourjon, "Deep Content: Unveiling video streaming content from encrypted WiFi traffic," in *17th IEEe NCA*, 2018.

[23] A. Reed and M. Kranch, "Identifying HTTPS-protected Netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. ACM, 2017, pp. 361–368.

[24] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A congestion sensitive website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014.

[25] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proc. of the 2014 ACM SIGSAC CCS*, 2014.

[26] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations," in *30th USENIX Security Symposium*, 2021.

[27] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "A real-time defense against website fingerprinting attacks," *arXiv:2102.04291*, 2021.

[28] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with N-shot learning," in *Proc. of the 2019 ACM SIGSAC CCS*.

[29] J. Lee and K. Park, "GAN-based imbalanced data intrusion detection system," *Personal and Ubiquitous Computing*, pp. 121–128, 2021.

[30] H. Chen and L. Jiang, "Efficient GAN-based method for cyber-intrusion detection," *arXiv preprint arXiv:1904.02426*, 2019.

[31] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, 2016, pp. 13–21.

[32] R. Burks, K. A. Islam, Y. Lu, and J. Li, "Data augmentation with generative models for improved malware detection: A comparative study," in *2019 IEEE 10th UEMCON*.

[33] M. S. Rahman, M. Imani, N. Mathews, and M. Wright, "Mockingbird: Defending against deep-learning-based website fingerprinting attacks with adversarial traces," *IEEE TIFS*, vol. 16, pp. 1594–1609, 2020.

[34] C. Hou, G. Gou, J. Shi, P. Fu, and G. Xiong, "WF-GAN: Fighting back against website fingerprinting attack using adversarial learning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*.

[35] D. Shu, N. O. Leslie, C. A. Kamhoua, and C. S. Tucker, "Generative adversarial attacks against intrusion detection systems using active learning," in *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, 2020, pp. 1–6.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[37] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv preprint arXiv:1706.02633*, 2017.

[38] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[39] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[40] K. N. Choi, A. Wijesinghe, C. M. M. Kattadige, K. Thilakarathna, S. Seneviratne, and G. Jourjon, "SETA: Scalable encrypted traffic analytics in multi-Gbps networks," in *2020 IEEE 45th LCN*, 2020.

[41] P. Zingo and A. Novocin, "Can GAN-generated network traffic be used to train traffic anomaly classifiers?" in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference*.

[42] 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*.

[43] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[44] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," *NeurIPS*, 2017.

[45] Y. Ye, L. Wang, Y. Wu, Y. Chen, Y. Tian, Z. Liu, and Z. Zhang, "GAN quality index (GQI) by GAN-induced classifier," 2018.

[46] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*.

[47] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang, "Statistical privacy for streaming traffic," in *Proceedings of the 26th ISOC Symposium on Network and Distributed System Security*, 2019.

[48] Q. Xiao, M. K. Reiter, and Y. Zhang, "Mitigating storage side channels using statistical privacy mechanisms," in *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[49] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE CVPR*, 2017.

[50] R. Shmelkin, T. Friedlander, and L. Wolf, "Generating master faces for dictionary attacks with a network-assisted latent space evolution," *arXiv preprint arXiv:2108.01077*, 2021.

VIII. Appendix

Table III shows the performance of LSTM, AdaBoost, and MLP on defended traces. As with the other experiments, the original dataset has been defended 10 times before these models were trained and tested on it. We see that the MLP performs surprisingly well on traces defended without using the optional noise step. However adding noise brings the MLP's accuracy in line with the earlier CNN results. Overall, the results show that SMAUG is generalizable and is able to reliably thwart model architectures other than CNNs.

TABLE III: SMAUG's effectiveness on other classifiers.

Dataset	Target	AdaBoost	MLP	LSTM
SETA Up	Undefended	83.0%	87.5%	83.1%
	CGAN	21.6%	61.1%	5.5%
	CGAN + Noise	18.4%	57.2%	5.0%
SETA Down	Undefended	61.2%	79.5%	56.5%
	CGAN	12.7%	59.4%	22.9%
	CGAN + Noise	5.0%	30.4%	7.3%
DC Up	Undefended	95.5%	97.0%	94.7%
	CGAN	41.7%	69.3%	56.3%
	CGAN Noise	20.8%	45.3%	45.1%
DC Down	Undefended	96.9%	94.5%	88.5%
	CGAN	35.7%	71.5%	64.9%
	CGAN + Noise	23.3%	46.6%	46.4%