Evading Encrypted Traffic Classifiers by Transferable Adversarial Traffic

Hanwu Sun[1,2], Chengwei Peng[3], Yafei Sang[1,2(✉)], Shuhao Li[1,2], Yongzheng Zhang[4], and Yujia Zhu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{sunhanwu,sangyafei,lishuhao,zhuyujia}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China
pengchengwei@cert.org.cn
[4] China Assets Cybersecurity Technology Company, Beijing, China
zhangyongzheng@iie.ac.cn

Abstract. Machine learning algorithms have been widely leveraged in traffic classification tasks to overcome the challenges brought by the enormous encrypted traffic. On the contrary, ML-based classifiers introduce adversarial example attacks, which can fool the classifiers into giving wrong outputs with elaborately designed examples. Some adversarial attacks have been proposed to evaluate and improve the robustness of ML-based traffic classifiers. Unfortunately, it is impractical for these attacks to assume that the adversary can run the target classifiers locally (white-box). Even some GAN-based black-box attacks still require the target classifiers to act as discriminators. We fill the gap by proposing FAT (We use FAT rather than TAT to imporove readability.), a novel black-box adversarial traffic attack framework, which generates the trans**F**erable **A**dversarial **T**raffic to evade ML-based encrypted traffic classifiers. The key novelty of FAT is two-fold: i) FAT does not assume that the adversary can obtain the target classifier. Specifically, FAT builds proxy classifiers to mimic the target classifiers and generates transferable adversarial traffic to misclassify the target classifiers. ii) FAT makes adversarial traffic attacks more practical by translating adversarial features into traffic. We use two datasets, CICIDS-2017 and MTA, to evaluate the effectiveness of FAT against seven common ML-based classifiers. The experimental results show that FAT achieves an average evasion detection rate (EDR) of 86.7%, which is higher than the state-of-the-art black-box attack by 34.4%.

Keywords: Transferable adversarial traffic · Encrypted traffic classifiers · Adversarial example attack · Black-box attack

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 461, pp. 153–173, 2022.
https://doi.org/10.1007/978-3-031-24386-8_9

1 Introduction

Network traffic classification is a critical and fundamental task in network management and cyberspace security [1]. With the widespread use of encryption technology, the volume of encrypted traffic is ballooning (Google reports that 95% of its products use encrypted communication [2]). Unfortunately, encrypted network traffic brings enormous challenges to traditional payload-based traffic classifiers [3]. To overcome these challenges, Machine Learning (ML) algorithms have been widely leveraged in encrypted traffic classification tasks and have become the mainstream method in many applications, such as quality of service (QoS) [4], application classification [5] and malicious traffic detection [6], etc.

While excellent performances are shown, ML-based encrypted traffic classifiers introduce new vulnerabilities named *adversarial example attacks* [7,8]. An adversarial example is formed by adding a well-designed and tiny perturbation to the normal example, which can fool the ML-based classifier into giving a wrong classification output with high confidence. To evaluate and improve the robustness of ML-based classifiers, the computer vision community proposes several adversarial example attacks, such as FGSM [8] and C&W [9]. Nevertheless, these attacks are specifically designed for images and are not well suited for disrupting network traffic. Due to the limitations of the complex TCP/IP protocols, not all bytes in the encrypted traffic can be modified at will. Additionally, extracting features from traffic is non-differentiable, making most gradient-based adversarial example generation methods unsuitable for two-step ML classifiers.

Some works have attempted to remove gaps between the data and exploit the adversarial vulnerability of encrypted traffic classifiers [10–12]. Unfortunately, the assumptions are too strong to perform these adversarial attacks. Firstly, most existing adversarial example attacks assume that the adversary can obtain copies of the target classifiers and run them locally. Even some methods [11, 12] use the target classifier as the discriminator of the Generative Adversarial Network(GAN) [13] architecture to generate adversarial examples and do not require the target classifier to backpropagate gradient. The target classifiers are still needed to participate in the entire calculation process. This requirement still seems harsh for the adversary. Secondly, some works choose to directly perform adversarial perturbations on feature vectors to avoid the non-differentiability problem of feature extraction. It is impractical to assume that the adversary can manipulate the feature vectors directly. Hence, a more practical adversarial attack is required to evaluate and improve ML-based encrypted traffic classifiers.

In this paper, we fill the gap by proposing FAT with more reasonable assumptions. FAT is a novel and practical black-box adversarial traffic attack framework, which is capable of generating the trans**F**erable **A**dversarial **T**raffic to evade the ML-based encrypted traffic classifiers. The key novelty of FAT consists of two parts. Firstly, FAT does not assume that the adversary is able to obtain the copies of the target classifiers and run them locally. FAT barely requires any knowledge of the target encrypted classifiers except the prediction results. It is not harsh for the adversary to obtain the prediction result from a commercial encrypted traffic classifier. Proxy classifiers are built locally in FAT to mimic the target classifiers, and transferable adversarial traffic is generated to mis-

classify the target classifiers. Secondly, FAT makes the adversarial traffic attack more practical by translating the time-related features and packet-length-related features into traffic.

Our contributions can be briefly summarized as follows:

(1) We propose FAT, a novel and practical black-box adversarial traffic attack framework, which is able to effectively misclassify the encrypted traffic classifiers without controlling or obtaining the target classifiers. FAT is the first adversarial traffic attack assuming that the adversary does not obtain the copies of the target classifier to the best of our knowledge.
(2) We transform the time-related and packet-length-related adversarial features into real adversarial traffic, which can run on existing network infrastructure normally. It is more practical to perform an adversarial attack by the adversarial traffic than the adversarial feature vectors.
(3) We evaluate the effectiveness of FAT on two popular datasets(CICIDS-2017 and MTA). The results show that FAT can achieve an average evasion detection rate(EDR) of 86.7%, which is 51.6% and 34.4% higher than the state-of-the-art white-box and black-box attacks, respectively.
(4) We demonstrate that the architecture of the proxy classifier and the dataset on which the proxy classifier is trained are two key factors affecting the transferability of the adversarial traffic.

2 Related Work

Encrypted Traffic Classifiers. Sun *et al.* [14] propose a hybrid approach combining signature and statistical features, using the signature method to identify TLS traffic and then using the statistical feature-based method to identify different applications. Liu *et al.* [15] designed a set of packet-level statistic features and fed them into a semi-supervised algorithm to distinguish different encrypted flows. Okada *et al.* proposed and evaluated 49 statistical features and analyzed how these features differentiated between unencrypted and encrypted traffic. Liu *et al.* [16] proposed an end-to-end model that leverages the powerful representational capabilities of the DL algorithm to learn useful features from packet sequences automatically. Moreover, they use autoencoder to strengthen this feature learning process. Finally they use several layers of MLP as a classifier to classify the learned features and achieve very high classification accuracy. As described, methods using ML and DL techniques have become the mainstream.

Adversarial Examples Attack. The concept of adversarial examples was first proposed by Szegedy *et al.* [7]. They design L-BFGS to maliciously add imperceptible subtle perturbations to the input samples, which cause the target model to output incorrect classification results with high confidence. Goodfellow *et al.* [8] proposed a Fast Gradient Sign Method(FGSM) to generate adversarial examples efficiently and quickly. FGSM solved the problem of the low efficiency of L-BFGS.

In the field of traffic, the most primitive attack is traffic obfuscation against a traffic classifier. Stinson *et al.* [17] escaped the detection of the classifiers by making some random perturbations to the traffic. But such attacks didn't exploit the

adversarial vulnerabilities of machine learning. Hashemi *et al.* [11] introduced the adversarial example technique to the field of anomaly detection and designed an adversarial attack against traffic classifiers. Nonetheless, such attacks based on the white-box assumption are not very practical. Lin *et al.* [10] designed an adversarial traffic features generation method based on GAN architecture. However, this method can only generate adversarial feature sets but cannot be converted into adversarial traffic and has limited practicality. While Novo *et al.* [18] implemented adversarial examples in the traffic space by adding adversarial proxy at both hosts of the C&C communication link, it is still a while-box attack. Han *et al.* [12] proposed a sophisticated black-box adversarial traffic attack also using GAN. Unfortunately, these GAN-based methods still require the target model to act as the discriminator, which is still harsh for the adversary to some extent.

3 Preliminaries

In this section, we sequentially introduce the adversarial examples attack, our motivation and define our problem and goals.

Table 1. List of the notations

Notations	Meaning	Notations	Meaning
$\mathbb{X}$	Traffic flow set	ψ_t	Target classifier
$X_{m/b}$	A malicious/benign flow in $\mathbb{X}$	ψ_p	Proxy classifier
$X^{adv/c}$	A adversarial/clean flow in $\mathbb{X}$	α	Gradient scaling factor
x_i	The i-th feature of flow X	θ	Parameters of a classifier ψ
$Y_{m/b}$	Actual malicious/benign label	η	Adversarial perturbation
$Z_{m/b}$	Prediction of target classifiers	$\mathcal{L}$	Cross-Entropy loss
$A_{m/b}$	Prediction of proxy classifiers	ε	Perturbation constraint

3.1 Adversarial Examples

An adversarial example is a maliciously modified input, which causes the victim ML&DL-based classifier ψ to make a wrong prediction. Usually, adversarial examples are also generated by minimizing the loss function $\mathcal{L}$. However, unlike training a neural network, when generating adversarial examples, the parameters θ of the target classifier are kept fixed and modify the input samples.

$$X^{adv} = \arg \min_{d(X, X^{adv}) \leq \varepsilon} \mathcal{L}(X, Y_{targeted}, Y_{true}; \theta) \tag{1}$$

In a targeted adversarial attack, the loss function usually consists of two cross-entropy losses $\mathcal{C}$. Minimizing the first cross-entropy loss to keep the predicted

label Z far away from the actual label Y_{true}, and minimizing the second cross-entropy loss to ensure the predicted labels are as close as possible to the targeted label $Y_{targeted}$.

$$\mathcal{L}(X, Y_{targeted}, Y_{true}; \theta) = -\mathcal{C}(\psi(X;\theta), Y_{true}) + \mathcal{C}(\psi(X;\theta), Y_{targeted}) \quad (2)$$

Given the classifier prediction distribution Z and label Y, the cross-entropy loss is as follows: $I(Y = c)$ is one if $Y = c$; otherwise, it is zero.

$$\mathcal{C}(Z, Y) = -\frac{1}{N}\sum_{i}^{N}\sum_{c=1}^{C} I(Y = c)\, log\, Z_c^i \quad (3)$$

Additionally, we use the L-infinity norm distance function to measure the degree of perturbation.

$$d(X, X^{adv}) = ||X^{adv} - X||_\infty \quad (4)$$

3.2 Motivation

Adversarial attacks essentially move normal examples across the classifier's decision boundary by perturbing. Consequently, we need the parameters of the target classifier to calculate the direction in which the examples are moving. Unfortunately, it seems impractical for the adversary to get the complete parameters of a black-box classifier.

Inspired by this limitation, we attempt to locally build a proxy classifier with a similar decision boundary as the target classifier. Thus the proxy classifier guides a similar moving direction for the adversarial examples as the target classifier. Additionally, the architecture and training data are two key factors affecting the decision boundary of a classifier. Classifiers with similar architectures trained on data which have similar distributions have closer decision boundaries. Therefore, we need to search or a suitable architecture for the proxy classifier and collect data with the same or similar distributions as the target classifier used.

As shown in Fig. 1, when we build a proxy classifier that meets the requirements, the remaining work is to move the examples to cross the proxy classifier's decision boundary. By controlling the moving distance, the samples will also cross the decision boundary of the target classifier. We consider such adversarial examples to be transferable.

3.3 Problem Definition

This paper focus on the task of malicious encrypted traffic detection. Therefore, the adversarial example attack is to generate adversarial traffic to misclassify the target malicious traffic classifier. A raw traffic flow can be represented as a feature vector. In a specific attack scenario, given a malicious but clean(not be perturbed) flow $X_m^c = [x_1, x_2, \cdots, x_n], X_m^c \in \mathbb{X}$, where x_i is the i-th feature

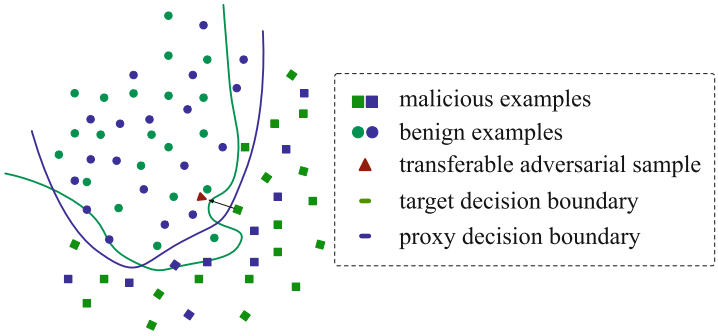

Fig. 1. The motivation of the transferable adversarial examples

of the malicious flow, and $\mathbb{X}$ is the traffic flow set. The target malicious traffic classifier ψ_t classifies the malicious flow as malicious label Z_m.

Our approach is divided into two steps. First, we need to train a proxy classifier ψ_p to mimic the target classifier. And the proxy classifier can also classify the malicious flow normally $\psi^p(X_m^c) \rightarrow Z_m$. The second step is to generate adversarial traffic flow X_m^{adv}. And the adversarial traffic flow not only can misclassify the proxy classifier $\psi_p(X_m^{adv}) \rightarrow Z_b$, but also can be transferred to misclassify the target classifier $\psi_t(X_m^{adv}) \rightarrow Z_b$ to output the benign label. Table 1 list the notations used in this paper.

4 Proposed Method

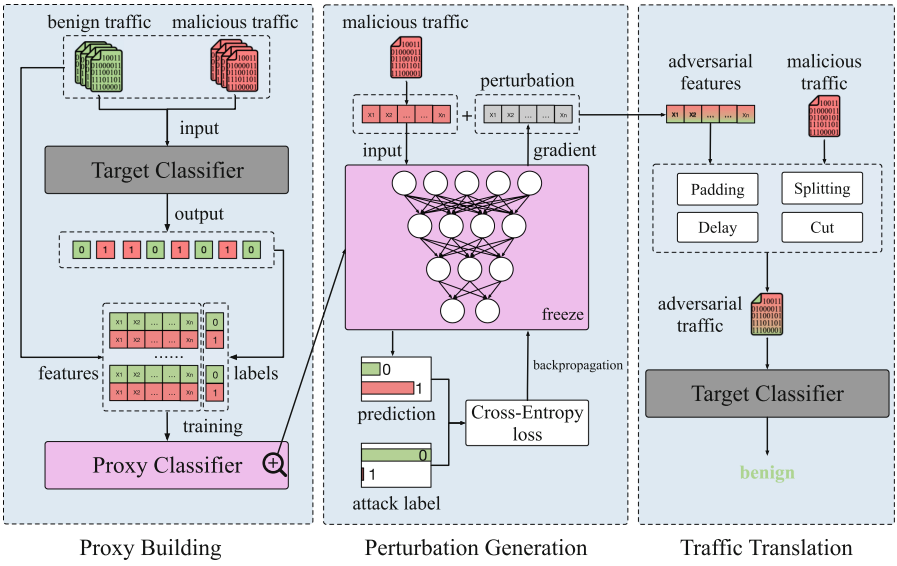

Fig. 2. The general framework of our approach.

This section first discuss the threat models in adversarial traffic attack scenarios, which contains some assumptions about the adversary and the detectors. And then, our approach will be introduced. The approach mainly consists of three steps, as shown in Fig. 2. Firstly we feed a batch of traffic samples into the target classifier and collect the output. Afterwards we use the output as the labels of these traffic samples to train a proxy classifier. Secondly, we conduct a gradient-based adversarial attack on the proxy classifier and generate the adversarial perturbation. Finally, we translate the adversarial features into the original malicious traffic and perform adversarial attack on the target encrypted traffic classifier.

4.1 Thread Model

The thread model for adversarial traffic attack is shown in Fig. 3. Entities related to it are as follows.

1) *The victim.* A victim is a host that is infected with malware or controlled by malware, also known as a zombie host, leaking information to the outside world or receiving instructions through encrypted command and control(C&C) channel.
2) *The detector*: The detectors are malicious encrypted traffic classifiers which are located in some communication nodes between the victim hosts and the hackers. They are usually viewed as black-box models and are passive in capturing the packets from the link.
3) *The adversary*: The adversaries are usually the ones who control the victim host, so they can manipulate the traffic generated by the malware and access the communication link. Furthermore, we assume that the adversaries can obtain the detection result of the detectors. It isn't a strong assumption to obtain the results of the detector compared to obtaining the copy of the detector or requiring the detector to participate in the calculation. Many commercial detectors, such as MTD [19] and GuardDuty [20], provide the service of querying detection results.

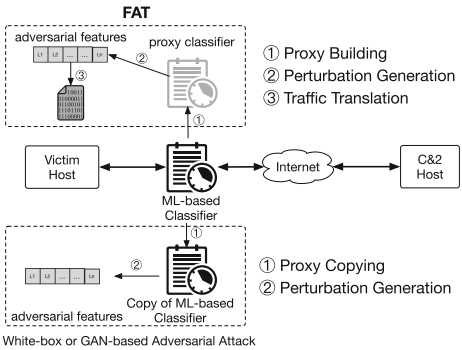

Fig. 3. Threat model for adversarial traffic attack scenarios

4.2 Proxy Building

The first and critical step of our approach is the proxy model building. In this step, we try to build a proxy classifier that has a similar decision boundary with the target classifier.

First of all, we prepare a batch of data X containing benign traffic and malicious traffic and its actual label Y. For the k-th traffic sample X_k, feed it to the target classifier ψ_t and collect its output Z_k. Note that Z_k is a category of X_k, not the probability distribution, which make it is more practical to implement. Next, we focus on training the proxy classifier ψ_p. For the j-th traffic sample X_j, we feed it into the proxy classifier and obtain the output a_j. And then calculate the cross-entropy loss(as shown in Eq. 3) of a_j and Z_j. Finally, we update the parameters θ of the proxy classifier by backpropagating. The entire process of transfer training is shown as Algorithm 1.

The adversary could control the victim host to run the malware and generate corresponding malicious traffic in a practical attack scenario. It should be noted that most online detectors have a proper delay, so there needs to be a time interval between running the malware twice on the victim host. Thus, it is time-consuming and laborious work. In the experimental environment, we directly input the prepared traffic to the target classifier.

Algorithm 1. Proxy Building

Input: Encrypted traffic samples: $X \in \mathbb{X}$; Target detector: ψ_t; Learning rate: lr; Train epoch size: *epoch*; Samples number: N

Output: proxy model: $\psi_p^* = \arg\min_{\theta \in \psi_p} \mathcal{J}(\psi_p(X), \psi_t(X))$

```
for k ← 1 to N do
    Z_k ← ψ_t(X_k)
end for
for i ← 1 to epoch do
    L ← 0
    for j ← 1 to N do
        a_j ← ψ_p(X_j)
        loss ← −(Z_j · log(a_j) + (1 − Z_j) · log(1 − a_j))
        L ← L + loss
    end for
    L ← L/N
    θ_{i+1} = θ_i − lr · ∂L/∂θ_i
end for
```

4.3 Perturbation Generation

Since the proxy model ψ_p is trained locally, we can obtain all the parameters of the proxy model. Therefore, we can generate adversarial perturbations with gradient-based methods.

Given a malicious traffic sample X_m, we need to turn it into adversarial malicious traffic X_m^{adv}. Firstly, we obtain the output A of the proxy classifier for X_m. And then calculate the cross-entropy loss of A and attack target label Y_b(means that the adversary wants the proxy classifier to misclassify X as benign). Finally, we update X_m with the gradient obtained by backpropagation. We cyclically repeat the above steps until the proxy classifier predicts X_m as benign, at which point we get the adversarial traffic X_m^{adv}. The generation of X_m^{adv} is shown in Algorithm 2.

There are two points to pay special attention to. On the one hand, unlike the training of ψ_p, we need to fix the parameters θ of the proxy classifier ψ_p and modify the input samples guided by the gradient. On the other hand, our perturbation to the traffic samples needs to be limited under ε, and the distance function d we use here is the L-infinity norm to control the cost of translating adversarial traffic.

Algorithm 2. Generate Adversarial Example

Input: proxy detector: ψ_p; Malicious but clean traffic: X_m^c; Scaling factor: α; Perturbation constraint: ϵ; Max epoch size: *epoch*;
Output: Adversarial malicious traffic: X_m^{adv}

$X_m^{adv} \leftarrow X_m$
$A \leftarrow \psi_p(X_m^{adv})$
$num \leftarrow 0$
while $A \neq Y_b$ and $num \leq epoch$ **do**
  $loss \leftarrow -(Y_b \cdot log(A) + (1 - Y_b) \cdot log(1 - A))$
  $\eta \leftarrow \frac{\partial loss}{\partial X_m^{adv}}$
  $X_m^{adv} \leftarrow X_m^{adv} + \alpha \cdot \eta$
  $X_m^{adv} \leftarrow clip(X_m^{adv}, \varepsilon)$ **s.t.** $d(X_m^c, X_m^{adv}) \leq \varepsilon$
  $A \leftarrow \psi_p(X_m^{adv})$
  $num \leftarrow num + 1$
end while

4.4 Traffic Translation

The adversarial traffic samples X^{adv} we generate so far are feature vectors. As mentioned before, an adversarial attack on features is meaningless to the adversary. So our main task in this part is to translate the adversarial features sequences X_m^{adv} into real adversarial traffic.

There are two kinds of features commonly used in ML&DL-based encrypted traffic classifiers. Traditional ML classifiers generally use manually extracted statistical features as input, while DL classifiers automatically learns useful information from sequential features. We used *CICFlowMeter* [21] to extract 77-dimensional flow-based statistical features[1], and use two sequential features:

[1] The 77-dimensional feature list extracted by CICFlowMeter can be obtained from https://github.com/ahlashkari/CICFlowMeter.

packet-length sequence and time-interval sequence. In this paper, we divide existing features into three categories according to the operations of translating adversarial feature into traffic. Features used in this paper are shown in Table 2

Packet Length-Related Features. Packet length-related features include various packet length-related statistical features and sequential features, such as mean packet length, max packet length, packet length sequence and so on. Time-related features such as packet interval are greatly affected by hardware, so they can generally be artificially increased but not reduced.

- ***Padding***. The adversary would pad meaningless data, such as 0x00, to the end of the packet to increase the packet's length. When the C&C server receives the data, it will remove the padding data at the end, so this operation will not affect the regular communication of the malware.
- ***Splitting***. The adversary would split a packet into two smaller packets to reduce the length of some large packets. For the C&C server, it is equivalent to receiving two smaller packets.

Time-Related Features. Time-related features include various time-related statistical features and time-related sequential features, such as flow duration, mean packet interval, time interval sequence and so on. For translating time-related features into traffic, the following operations are used. Note that we only operate the application layer data in order to ensure the normal transmission of the traffic.

- ***Delay.*** The adversary would delay a packet until the interval time between two packets meets the requirements.

5 Experimental Evaluation

In this section, we first introduce our experimental settings in detail, including the datasets, proxy&target classifier architectures, evaluation metrics and experimental environment. Then we search for the optimal hyperparameters and the optimal proxy classifier architectures for each target classifier. Based on optimal hyperparameters and proxy classifier, we evaluate the effectiveness of FAT and compare it with the state-of-the-art black-box and white-box adversarial attacks. Finally, we visualize the adversarial traffic and explain why it works.

5.1 Experimental Settings

Datasets. We use two popular traffic datasets(CICIDS-2017 and MTA) to evaluate the effectiveness of FAT. *CICIDS-2017* [22] dataset contains benign and the most up-to-date common attack traffic, which resembles true real-world data. We selected five common attack traffic as positive samples and the same amount of

Table 2. Summary of features

	Statistical features				Sequence features
	No.	Name	Attributes	Direction	Name
Time-related features	0	Flow duration	-	Both	Time interval sequence
	15–28	Time between two packets	Avg, Std, Max, Min	Both	
			Sum, Avg, Std, Max, Min	Fwd	
			Sum, Avg, Std, Max, Min	Bwd	
	69–76	Active time	Avg, Std, Max, Min	Both	
		Idle time	Avg, Std, Max, Min	Both	
Packet- length-related features	1–12	Packet count	–	Both	Packet length sequence
		Total length	–	Fwd, Bwd	
		Packet length	Avg, Std, Max, Min	Fwd, Bwd	
	37–41	Packet length	Avg, Std, Max, Min, Var	Both	
Others	13–14 50–68	Rate	–	–	–
	29–36 42–49	TCP flag	–	–	

benign traffic in the same period as negative samples. *MTA* [23] dataset provides a large amount of C&C communication traffic captured from sandbox environments for up to 10 years. We collected 1621 pcap files provided by MTA from 2013 to 2022. Then we filtered out all encrypted flows and labelled each SSL flow by comparing its SHA1 fingerprint of the certificate and the *SSL Blacklist project* [24]. Finally, we selected the traffic of five malware families with enough flows as positive samples and the same number of benign traffic as negative samples. Table 3 describes the malicious type and the number of flows of the datasets we used.

In each dataset, we split the dataset into a target training set (40%), a proxy training set (40%) and a test set (20%). The target training set is used to train the target classifiers, and the proxy training set contains only features and no labels. The test set is used to generate adversarial traffic and test the performance of the target classifiers. We use 5-fold cross-validation to reduce overfitting.

Target Classifiers. To evaluate the effectiveness of FAT against ML-based malicious encrypted traffic classifiers. We use the seven most common ML-based classifiers with different architectures as the target classifiers. Logistics Regression (LR), Decision Tree (DT) and Support Vector Machine (SVM) represent common supervised learning classifiers. Multi-Layer Perceptron (MLP) represents neural network-based classifiers. Isolation Forest (IF) is the representative of unsupervised classifiers. Additionally, Long Short-Term Memory (LSTM) [25] and FS-Net [16] are representatives of end-to-end classifiers that exploit the sequential features of traffic.

Table 3. Summary of datasets in evaluation

Datasets	Features type	Malicious type	# Flows
CICIDS-2017	Statistical features	Botnet	1,966
		Fuzzing	231,073
		PortScan	127,537
		BruteForce	13,835
		DDoS	97,718
MTA	Statistical features sequence features	Tofsee	22,947
		Dridex	8,165
		Quakbot	706
		TrickBot	657
		Gozi	585

Proxy Classifiers. To explore the effect of proxy classifiers using different algorithmic architectures and features on the effectiveness of FAT, we selected three proxy classifiers with different architectures. MLP is representative of traffic classifiers using statistical features, and LSTM and FS-Net are representatives of classifiers using sequential features. At the same time, they also represent common traffic classifiers with DNN and RNN structures. Note that the adversarial traffic is formed by modifying the original malicious traffic according to the back-propagated gradient of the proxy classifier, so the proxy classifier is limited to neural network-based classifiers.

Table 4. The classification result confusion matrix

True label	Predicted label	
	Z_m	Z_b
Y_m	True Positive (TP)	False Negative (FN)
Y_b	False Positive (FP)	True Negative (TN)

Metrics. For each category, according to the true label Y and predicted label Z of each sample, samples can be divided into four categories, as shown in Table 4. For each target classifier, we use macro *Precision*, *Recall*, *F1-Score* to evaluate its classification performance.

We focus on whether the target classifier can correctly identify malicious samples, so we use the macro *Evasion Detection Rate*(EDR) as the metric to evaluate our attack. EDR, aka False Negative Rate (FNR), refers to the ratio of the number of samples that are actually malicious but predicted to be benign to the total number of actual malicious samples.

$$EDR_i = \frac{FN_i}{TP_i + FN_i}, EDR = \frac{1}{N}\sum_{i=1}^{N} EDR_i \tag{5}$$

Experimental Environment. The core part of the proxy/target model and adversarial perturbing are implemented with the Sciket-learn and PyTorch frameworks. We used a machine that has a Xeon processor with 32 cores running at 2.3 GHz and 128 GB RAM.

5.2 Proxy Classifier and Hyperparameter Selection

Impact of Perturbation Limit(ε). Like other traffic obfuscation methods, adversarial traffic attacks also have a trade-off between the degree of perturbation and the evasive effectiveness. In order to search for the optimal ε, we designed the following experiments.

Firstly, when both the target classifier and the proxy classifier are FS-Net, the evasive effectiveness of FAT with different ε is shown in Fig 4(a). For all malware families in the MTA dataset, FAT's evasion detection rate(EDR) increases with the perturbation degree ε. When $\varepsilon = 0.03$, the EDR of all malware families are above 90%. What's more, Fig. 4(b) shows the evasive effectiveness of adversarial traffic for different target classifiers with different ε. Taking Trickbot as an example, when $\psi_p = LSTM$ and $\varepsilon = 0.10$, the adversarial traffic has excellent evasive effectiveness on DL-based classifiers such as FS-Net and LSTM but has weak evasive effectiveness on traditional ML-based classifiers such as SVM and LR. The reason is that the architecture of the proxy classifier and the target classifier are too different. The influence of the algorithm architecture between the proxy classifier and the target classifier will be explored later. However, even if the classifier architectures are quite different, when $\varepsilon = 0.3$, the adversarial traffic can achieve acceptable evasive effectiveness.

Additionally, it can be seen from Fig. 4(a) that the variations of the evasive effectiveness on the proxy model and the target model are similar. So the adversary can fine-tune ε based on the feedback from the proxy model until the best one is found.

Impact of Training Data. For an ML&DL-based classifier, the distribution of training data dramatically affects the shape of its decision boundary or decision hyperplane. By collecting the output of the target classifier, we try to build a training dataset with a similar distribution to the target. The more data we collect, the more similar its distribution is to the original distribution. We use different number of collected samples to train the proxy classifier ($\psi_p = LSTM$) separately and then generate adversarial traffic. The evasive effectiveness of FAT for FS-Net is shown in Fig. 5. From our experimental results, it can be seen that the more data we collect, the higher the probability of adversarial traffic evading the target classifier.

Proxy Classifier Selection. There are two key factors to guide us in choosing an appropriate proxy classifier. One is the architecture of the target classifier, and the other is the type of features used by the target classifier.

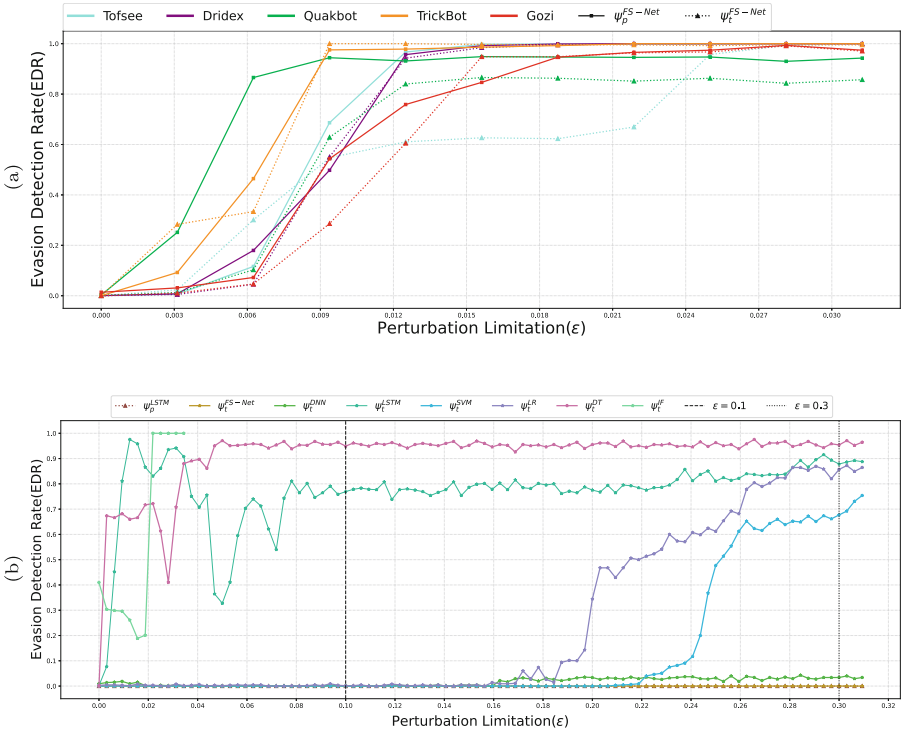

Fig. 4. The evasion effectiveness of the adversarial traffic with different ε. (a) shows different malware families and (b) shows different target architectures.

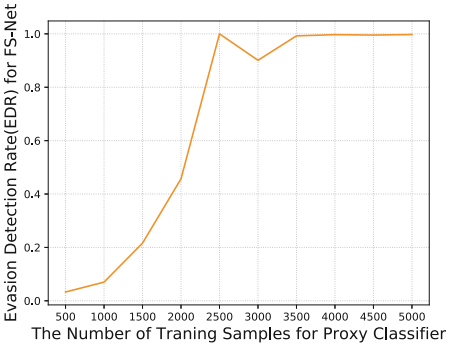

Fig. 5. The evasive effectiveness of adversarial traffic generated by LSTM classifier trained with different number of training samples.

To figure out the transferability of FAT between classifiers with different architectures, we chose three proxy classifiers (MLP, LSTM, FS-Net) and seven target classifiers and conducted 21 sets of experiments. At the same time, in

order to explore the impact of different types of features on FAT, we extract two type of features from MTA dataset for experiment. Note that FS-Net is not included in the experiments of statistical features since FS-Net cannot handle statistical features.

When the $\varepsilon = 0.1$, the experimental result is shown in Table 5. When the proxy and target classifiers use the same architecture, FAT achieves the best EDR even if their parameters are different. When the proxy and target classifiers use different but similar architecture, for example, FS-Net and LSTM, the evasive effectiveness is acceptable for the adversary. On the contrary, if the proxy classifier uses a significantly different architecture from the target classifier, such as LSTM and IF, the evasive effectiveness is poor. Thus we can conclude that **adversarial traffic transfer better between the classifier with more similar architectures**.

From the experimental results, we also find that FAT can be transferred well between classifiers using the same type of features. For example, if MLP is used as the proxy classifier, the EDR of FAT against classifiers using sequential features is significantly higher than those using sequential features. This is because such RNN-based classifiers are designed to analyse sequential data, which are different from traditional ML algorithms. In contrast, MLP is closer to traditional ML algorithms and can better fit statistical features.

We attempt to explain from the perspective of the decision boundary. There are two main factors that determine the decision boundary of a classifier, one is the algorithm architecture, and the other is the distribution of the training data. Classifiers with similar architectures contain more similar mapping functions and are more likely to learn similar decision boundaries. For example, both FS-Net and LSTM contain RNN structure, which is very different from DNN, so the adversarial traffic generated based on LSTM has good evasive effectiveness for FS-Net, but is poor for DNN classifier. What's more, some ML algorithms like DT and IF do not apply gradients to update parameters, so the gradient-based adversarial traffic seems to be less effective for evading these classifiers.

Table 5. Transferability of adversarial traffic between classifiers with different architectures ($\varepsilon = 0.1$).

Feature type	Proxy classifiers (ψ_p)	EDR against target classifiers (ψ_t)						
		MLP	FS-Net	LSTM	LR	DT	SVM	IF
Sequence features	MLP	**65.62**	32.84	27.43	20.63	15.1	0.46	**49.06**
	FS-Net	8.9	**99.94**	**96.19**	30.03	**100**	0.46	21.36
	LSTM	7.06	96.68	82.58	**42.26**	71.63	0.55	24.136
Statistical features	MLP	**69.05**	–	41.67	**72.96**	**63.69**	**51.03**	**21.74**
	LSTM	34.62	–	**75.97**	38.52	50.48	34.84	0.89

5.3 Attack Effectiveness for Different Target Classifiers

Based on the above observations and findings, the attack effectiveness experiments of FAT are set up as follows. For the sequential feature, we choose FS-Net as the proxy classifier, and for the statistical feature, we use MLP as the proxy classifier. We calculate the detection performance of four ML-based classifiers and two DL-based classifiers, respectively, and the EDR of FAT against the above seven target classifiers.

When the perturbation limitation $\varepsilon = 0.1$, the experimental results are shown in Table 6. When the target classifiers use sequential features, the adversarial traffic generated by FS-Net achieves more than 95% evasion detection rate on LSTM and FS-Net while having poor evasive effectiveness on MLP, LR, SVM and IF. DT seems odd, mainly because the tree-related classifiers can change the classification result as long as one suitable feature is modified.

When the target classifiers use statistical features, the adversarial traffic generated by MLP has more than 50% EDR for most ML classifiers except IF. The reason is that IF is different from other classifiers and does not use indicators such as distance or density to describe the difference between a sample and other samples, so modifying features in a small range may not work well for IF.

Table 6. Detection effectiveness of target classifiers and evasing effectiveness of adversarial traffic.

Target classifier (ψ_t)	Sequence features (ψ_p = FS-Net)				Statistical features (ψ_p = MLP)			
	Detection			Evasion	Detection			Evasion
	P	R	F1	EDR	P	R	F1	EDR
MLP	97.78	98.43	98.11	8.90	94.84	95.00	94.80	**69.05**
FS-Net	99.98	99.98	99.98	**99.94**	–	–	–	–
LSTM	96.88	97.09	96.85	**96.19**	94.40	94.47	94.35	41.67
LR	99.75	99.87	99.81	30.03	89.75	99.06	94.09	**72.96**
DT	100.00	100.00	100.00	**100.00**	99.61	99.53	99.57	63.69
SVM	99.85	98.99	99.42	0.46	96.43	99.24	97.77	**51.03**
IF	82.84	52.71	64.35	21.36	76.54	48.22	59.16	**21.74**

5.4 Comparison Experiments

Comparison Attack. To fully evaluate the effectiveness of FAT, we compare it with the state-of-the-art white-box and black-box adversarial traffic work. The baseline and SOTA works are summarized as follows.

– **Baseline.** Applying random perturbations to the traffic can also misclassifies the target classifiers [17], which is also adversarial to the target classifier to some extent. *R-T* randomly inserts interval time between packets; *R-D* randomly duplicates partial original traffic.

- **White-box.** Hashemi *et al.* [11] introduced adversarial attacks into network intrusion detection systems for the first time. They design a white-box adversarial attack assuming that the adversary has all knowledge about the target classifiers.
- **Black-box.** Han *et al.* [12] designed a black-box adversarial attack based on the GAN architecture and used the same CICIDS-2017 dataset as ours for evaluation.

Furthermore, to evaluate the effectiveness of our method in feature-constrained scenarios, we divide FAT into four groups according to the features. All statistical features in the first group can be modified(FAT-A). The second and third groups only modify time-related features(FAT-T) and packet length-related features(FAT-L), respectively. The last group modifies both time-related features and packet-length-related features.

Table 7. Comparative experimental results with $\psi_p = MLP$

Target	Detection			Comparison attack				Our FAT			
	P	R	F1	EDR				EDR			
				R-D	R-T	White-box	Black-box	A	T	L	T+L
MLP	99.62	98.52	99.16	12.96	7.15	27.03	47.89	94.89	73.99	58.37	**95.02**
LR	99.46	94.54	96.92	13.88	15.73	57.54	62.67	84.83	59.41	74.39	**80.38**
DT	99.88	99.86	99.87	14.02	15.90	56.70	65.11	99.99	100.00	100.00	**99.99**
SVM	99.41	96.47	97.71	15.39	13.65	33.89	59.93	99.63	81.86	65.59	**92.78**
IF	90.72	57.45	70.46	0.38	6.58	0.26	25.67	54.19	44.11	10.32	**55.33**
AVG	97.82	89.37	92.82	11.33	11.80	35.08	52.25	86.71	71.35	61.73	**82.12**

Comparison Results. The comparison results are shown in Table 7. As shown in the experimental results, when all features can be modified arbitrarily, our attack achieves an average EDR of 86.7%, which is higher than the state-of-the-art black-box attack by 34.4%. Even when only time-related features or packet length-related features are modified, our attack can achieve an average EDR of 71.3% and 61.7%, and if both types of features are used, the average EDR can reach 82.1%.

Comparing our second and third groups of attacks, we find that the performance of MLP, SVM, and IF is more dependent on time-related features, while LR is more dependent on packet-length-related features.

5.5 Adversarial Traffic Visualization

To intuitively explain why FAT can evade encrypted traffic classifiers and can be transferred between classifiers with similar architecture, we conduct the following visualization experiments.

Effectiveness of FAT. For benign traffic and malicious traffic of each attack in the CICIDS-2017 dataset, we extract their 77-dimensional statistical features and generate adversarial features for malicious traffic under the guidance of the MLP proxy classifier. Finally, we use PCA to reduce the three sets of statistical features to 2 dimensions and draw their corresponding scatter plots.

The result is shown in Fig. 6. For each malicious attack, the distribution of the adversarial traffic is far different from its original malicious traffic and is closer to the benign traffic. In other words, FAT looks more like benign traffic to the classifiers based on spatial distance. This phenomenon is more obvious on Botnet, BruteForce and PortScan. Adversarial perturbation essentially changes the distribution of traffic in space, and once it crosses the decision boundary of the classifier, the adversarial traffic can misclassify the classifiers.

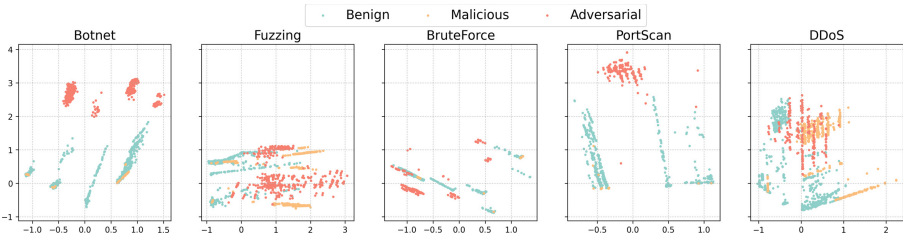

Fig. 6. Adversarial traffic visualization ($\psi_p = MLP$ and $\varepsilon = 0.1$).

Transferability of FAT. To demonstrate the transferability of FAT between classifiers with different architecture, we conduct the following experiments. For each target classifier, we take the prediction of the target classifier as the ground truth label, and combine the prediction of the proxy classifier to get the confusion matrix.

The results is shown in the Fig. 7 Fewer FPs and FNs indicate that the prediction results of the proxy classifier are closer to the target classifier. The experimental results show that the prediction results of MLP are closer to LR, SVM and DT, but quite different from IF. This means that the decision boundary of MLP is more similar with LR, SVM and DT, and FAT can be better transferred among them. Combined with the results in Table 7, it can be seen that the FAT generated by MLP has better evasion effectiveness against LR, SVM and DT, but the evasion effectiveness against IF is poor.

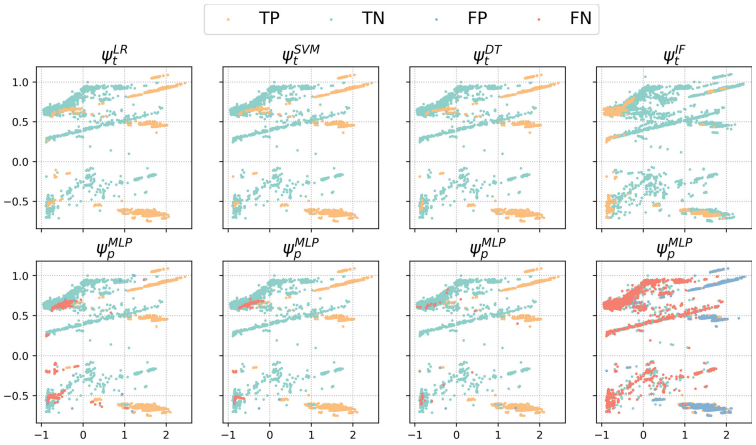

Fig. 7. Prediction results of classifiers with different architectures.

6 Conclusions and Future Work

This paper proposed a novel adversarial traffic attack framework named FAT, which can effectively misclassify the ML-based traffic classifiers without controlling or obtaining the target classifier. Firstly, we built a proxy classifier with a similar decision boundary to the target classifier. And then, we generate adversarial features via the proxy classifier to misclassify the target classifier. Finally, we translated the adversarial features into real adversarial traffic. Experimental results showed that under appropriate settings, transferable adversarial traffic can achieve an average EDR of 86.7% for unknown target classifiers, which is higher than the state-of-the-art black-box attack by 34.4%.

For future work, we will try to generate transferable adversarial traffic based on more complex features and attempt to propose a practical approach to defense the adversarial traffic attacks.

Acknowledgements. We thank the anonymous reviewers for their insightful comments. This work was supported by the National Key Research and Development Program of China (No. 2019YFB1005201, No. 2019YFB1005203 and No. 2019YFB1005205).

References

1. Tahaei, H., Afifi, F., Asemi, A., Zaki, F., Anuar, N. B.: The rise of traffic classification in IoT networks: a survey. J. Netw. Comput. Appl. **154**, 102538 (2020)
2. Google Transparency Report. https://transparencyreport.google.com/https/overview. Accessed 20 Mar 2022

3. Papadogiannaki, E., Ioannidis, S.: A survey on encrypted network traffic analysis applications, techniques, and countermeasures. ACM Comput. Surv. (CSUR) **54**(6), 1–35 (2021)
4. Wang, P., Lin, S. C., Luo, M.: A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs. In: 2016 IEEE International Conference on Services Computing (SCC), pp. 760–765. IEEE (2016)
5. Chen, Y., Zang, T., Zhang, Y., Zhou, Y. Wang, Y.: Rethinking encrypted traffic classification: a multi-attribute associated fingerprint approach. In: 2019 IEEE 27th International Conference on Network Protocols (ICNP), pp. 1–11. IEEE (2019)
6. Wang, Z., Fok, K.W., Thing, V.L.: Machine learning for encrypted malicious traffic detection: approaches, datasets and comparative study. Comput. Secur. **113**, 102542 (2022)
7. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
9. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE (2017)
10. Lin, Z., Shi, Y., Xue, Z.: IdsGAN: generative adversarial networks for attack generation against intrusion detection. arXiv preprint arXiv:1809.02077 (2018)
11. Hashemi, M.J., Cusack, G., Keller, E.: Towards evaluation of NIDSS in adversarial setting. In: Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks, pp. 14–21 (2019)
12. Han, D., et al.: Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. IEEE J. Sel. Areas Commun. **39**(8), 2632–2647 (2021)
13. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
14. Sun, G.L., Xue, Y., Dong, Y., Wang, D., Li, C.: An novel hybrid method for effectively classifying encrypted traffic. In: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, pp. 1–5. IEEE (2010)
15. Liu, H., Wang, Z., Wang, Y.: Semi-supervised encrypted traffic classification using composite features set. J. Netw. **7**(8), 1195 (2012)
16. Liu, C., He, L., Xiong, G., Cao, Z., Li, Z.: Fs-net: a flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1171–1179. IEEE (2019)
17. Stinson, E., Mitchell, J.C.: Towards systematic evaluation of the evadability of bot/botnet detection methods. WOOT **8**, 1–9 (2008)
18. Novo, C., Morla, R.: Flow-based detection and proxy-based evasion of encrypted malware c2 traffic. In Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security, pp. 83–91 (2020)
19. Managed Threat Detection. https://www.huaweicloud.com/product/mtd.html. Accessed 11 Aug 2022
20. Amazon GuardDuty. https://aws.amazon.com/cn/guardduty/. Accessed 11 Aug 2022
21. Davis, J.J., Clark, A.J.: Data preprocessing for anomaly based network intrusion detection: a review. Comput. Secur. **30**(6–7), 353–375 (2011)

22. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
23. Malware Traffic Analysis. https://malware-traffic-analysis.net/. Accessed 20 Mar 2022
24. SSL Blacklist Project. https://sslbl.abuse.ch/. Accessed 20 Mar 2022
25. Zhu, X., Sobihani, P., Guo, H.: Long short-term memory over recursive structures. In: International Conference on Machine Learning, pp. 1604–1612 (2015)