Adversarial Attacks on Pre-trained Deep Learning Models for Encrypted Traffic Analysis

Byoungjin Seok[1] and Kiwook Sohn[2,*]

[1]*Korea University, Korea*
[2]*Seoul National University of Science and Technology, Korea*
E-mail: bjseok@korea.ac.kr; kiwook@seoultech.ac.kr
[*]*Corresponding Author*

Received 18 April 2024; Accepted 16 June 2024

Abstract

For web security, it's essential to accurately classify traffic across various web applications to detect malicious activities lurking within network traffic. However, the encryption protocols for privacy protection, such as TLS 1.3 and IPSec, make it difficult to apply traditional traffic classification methods like deep packet inspection (DPI). Recently, the advent of deep learning has significantly advanced the field of encrypted traffic analysis (ETA), outperforming traditional traffic analysis approaches. Notably, pre-trained deep learning based ETA models have demonstrated superior analytical capabilities. However, the security aspects of these deep learning models are often overlooked during the design and development process. In this paper, we conducted adversarial attacks to evaluate the security of pre-trained ETA models. We targeted ET-BERT, a state-of-the-art model demonstrating superior performance, to generate adversarial traffic examples. To carry out the adversarial example generation, we drew inspiration from adversarial

Journal of Web Engineering, Vol. 23_6, 749–768.
doi: 10.13052/jwe1540-9589.2361

attacks on discrete data, such as natural language, defining fluency from a network traffic perspective and proposing a new attack algorithm that can preserve this fluency. Finally, in our experiments, we showed our target model is vulnerable to the proposed adversarial attacks.

Keywords: Encrypted traffic analysis, adversarial attacks, pre-trained deep learning models, bert, web security.

1 Introduction

Network traffic analysis, which identifies and classifies various web applications or services, is an essential web security technology for detecting malicious activities and managing services on networks [14, 23]. Traditional models rely on inspecting traffic packet ports and payload information for flow and content classification [1, 24]. However, the adoption of encryption technologies to strengthen security and privacy has rendered these conventional techniques less effective, as attackers can also encrypt malicious traffic, thus evading traditional traffic analysis methods. As a countermeasure, extensive research has been conducted on encrypted traffic analysis (ETA) models that do not rely on port or payload information [17]. With the recent advancements in deep learning, the primary approach now involves training models on the statistical features of encrypted traffic for classification tasks. These models have significantly outperformed their traditional counterparts.

Among various deep learning-based ETA models, those derived from image recognition, such as convolutional neural networks (CNNs), and natural language processing, like recurrent neural networks (RNNs), are well-established [19]. Pre-trained deep learning models for ETA, inspired by the success of pre-trained AI models such the large language model (LLM), have exhibited even better performance and robustness across diverse types of encrypted traffic due to their extensive pre-training. For the convenience of explanation, the pre-trained ETA model will be referred to as the *large encrypted traffic model* (LETM). Despite the powerful performance of deep learning-based ETA models, the reliability of these models, including the LETMs, remains a concern, primarily due to their *black box* nature, which obscures their inner workings and decision-making processes. Their performance is highly dependent on the training data, raising concerns that malicious inputs or training could impair their functionality. Thus, the security of the models themselves is a critical consideration in the perspective of web security.

In this paper, we explore the security of the LETMs against adversarial attacks. We target encrypted traffic bidirectional encoder representations from transformer (ET-BERT) [12], a state-of-the-art model demonstrating superior performance, to generate adversarial traffic examples. Inspired by adversarial strategies applied to discrete datasets in natural language processing, we proposed new adversarial attack to preserve the fluency from a network traffic perspective (i.e., grammar correctness and fluent lexical capacity). We applied our proposed attack to models that had been fine-tuned for ET-BERT's downstream tasks, assessing their performance under adversarial conditions. Our experimental results showed that the proposed attack successfully generate adversarial examples for the fine-tuned ET-BERT models by altering less than 10% of the original traffic data, while preserving the fluency of network traffic.

2 Related Works

Traditionally, encrypted traffic analysis utilized algorithms constructed or trained with machine learning techniques on statistical characteristics for network security tasks such as application identification, classification, and malicious activity detection. However, the rapid advancement in deep learning within machine learning has spurred active research into constructing deep learning-based ETA models. This section discusses the latest approaches directly relevant to our study, focusing on deep learning-based ETA techniques and adversarial attacks against ETA models. (For a comprehensive survey of statistical characteristic-based and machine-learning-based ETA, see [17].)

2.1 Deep-learning-based Encrypted Traffic Analysis

Existing ETA works based on traditional deep-learning models. The advent of deep learning has significantly improved the analysis and classification of encrypted network traffic. Deep learning models trained on the structural and statistical features of encrypted traffic have achieved notable accuracy in classification. Mohammad et al. [13] explored using CNN models to analyze encrypted traffic, generating 1D grayscale images of packets for effective CNN training. This innovative method utilizes CNN's robust feature extraction capabilities, adapted to encrypted traffic's unique challenges. Auwal Sani et al. [8] employed deep convolutional generative adversarial networks (DCGAN) for data augmentation in training on encrypted traffic.

They created a pseudo image matrix incorporating features like packet inter-arrival times, direction, length, and TCP window size, showcasing generative models' potential in enriching training datasets. Additionally, Shen et al. [22] and Diao et al. [3] used graph neural network (GNN) models to capture packet interactions, generating graph data representing these interactions. This approach introduces a novel representation of network traffic through graph data, proving highly effective for encrypted traffic analysis.

Existing ETA works based on pre-training approaches. The success of pre-trained LLM models such as bidirectional encoder representations from transformers (BERTs) and generative pre-trained transformers (GPTs) in natural language processing has inspired efforts to extend their application, including encrypted traffic classification. He et al. [6] extracted network packets, tokenizing the payload in 2 Byte units for BERT model pre-training and classification. This approach applies NLP methodologies to network traffic analysis, especially for encrypted data. Hu et al. [7] combined CNN embeddings of packets and flows with BERT model pre-training, showcasing an innovative blend of CNN and transformer models for network traffic understanding. Lin et al. [12] developed ET-BERT, segmenting network traffic into BURST units and tokenizing these for BERT model pre-training and classification, establishing a state-of-the-art ETA model with superior performance.

2.2 Adversarial Attacks on ETA Models

Zhao et al. [26] and Sadeghzadeh et al. [20] proposed GAN-based techniques for generating adversarial examples against ML/DL-based NIDS, demonstrating the potential for detection evasion or misclassification. McCarthy et al. [26] targeted Kitsune, an IoT NIDS built on auto encoders (AE), with attacks extracting crucial features through saliency maps to generate adversarial examples, successfully bypassing detection or triggering false alarms.

Reviewing deep learning-based ETA and adversarial attack technologies, we note that while various models like CNNs, GANs, and GNNs have been used, recent studies focus on deep learning models with pre-training approaches (i.e., development of LETM). However, research on adversarial attacks specifically targeting LETMs remains scarce. Existing adversarial attack techniques, typically involving GANs or gradient-based methods, are not specialized for natural language processing. Therefore, similar to NLP, these techniques are difficult to apply to LETM, which handles discrete data. This gap led us to explore adversarial attacks on LETMs.

3 Adversarial Attacks on LETMs

3.1 Brief Description of Target Model: ET-BERT [12]

ET-BERT is a framework for analyzing encrypted traffic based on the bidirectional encoder representation from transformers (BERT) [2], proposed at the WWW conference in 2021. The structure of the ET-BERT framework is as follows: It consists of a Datagram2Token pre-processor, which pre-processes network traffic into a token format that can be fed into the model, and a pre-training part that trains a deep learning network composed of bidirectional transformer blocks using a large volume of unlabeled encrypted traffic data. Additionally, it features a fine-tuning part that conducts specialized training for detailed tasks from a downstream task perspective. A brief description of each component is as follows:

* **Datagram2Token pre-processor**: This component takes raw network traffic as input and pre-processes it into flows based on the same IP, port, and protocol information according to protocol rules. It then arranges the traffic data in time units and performs tokenization to generate a format suitable for the pre-training input of the ET-BERT model, a process called BURST2Token. In BURST2Token, the payload part of a packet is bisected, and from each part, 256 bytes are extracted to form sub-BURST^A and sub-BURST^B, thus constructing a BURST $=$ sub-BURST^A||sub-BURST^B. Then, bigram encoding is applied to the BURST to create an encrypted traffic corpus. Finally, byte-pair encoding [5] is performed on the encrypted traffic corpus to construct the vocabulary used by the ET-BERT model.
* **Pre-training part**: This part involves pre-training the bidirectional transformer blocks with tokenized encrypted traffic information, utilizing two learning strategies – the masked BURST model and same-origin BURST Ppediction – on unlabeled traffic data. The learning strategies are defined as follows:
  - Masked BURST model: A strategy where specific tokens in the traffic data are probabilistically selected and replaced with [MASK] tokens, and the model is trained to infer the original tokens. The loss function for this strategy is defined as $L_{\mathrm{MBM}} = -\Sigma_{i=1}^{k} \log(P(\mathrm{MASK}_i = token_i|\bar{X};\theta))$.
  - Same-origin BURST prediction: A strategy aimed at predicting whether two sub-BURSTs originate from the same source. The loss function for this strategy is defined as $L_{\mathrm{SBP}} =$

$-\Sigma_{j=1}^{n} \log(P(y_j|B_j;\theta))$, where the label $y_i \in [0,1]$ (0 represents the same-origin BURST and 1 represents the different-origin BURST).

The final pre-training loss function combines the losses from these two strategies.; $L = L_{\mathrm{MBM}} + L_{\mathrm{SBP}}$. Pre-training utilized a total of 30 GB of unlabeled encrypted traffic data, combining 15 GB from publicly available datasets (ISCX-VPN [4], CICIDS2017 [21]) and 15 GB from a dataset collected by the authors, CSTNET.[1]

* **Fine-tuning part**: This part involves training optimized for downstream tasks, differing from the pre-training part as it learns classification on a packet or flow unit datagram basis rather than on BURST units. Here, datagrams at the packet or flow unit level are pre-processed into flow units based on the same IP, port, and protocol information, similar to the Datagram2Token process, and then tokenization is performed. The structure of the model is fundamentally the same as in the pre-trained model, with the output of the node corresponding to the original [CLS] token fed into a multi-class classifier, and the loss function is defined as $L_{CE} = -\Sigma_{c=1}^{N} y_i \cdot \log(\mathrm{softmax}(z_i;\theta))$, where N is the total number of classes, y_i is the one-hot encoded target label vector, and z_i represents the model's predicted probability for the classes.
Theoretically, fine-tuning can be applied to any network traffic analysis task, but the authors optimized the training for the following five specific downstream tasks:
  - General encrypted application classification (GEAC): Classification on encrypted application traffic under standard encryption protocols.
  - Encrypted malware classification (EMC): Classification on encrypted traffic consisting of malware and benign applications.
  - Encrypted traffic classification on VPN (ETCV): Classification on encrypted traffic that uses virtual private networks (VPNs).
  - Encrypted application classification on Tor (EACT): Classification on encrypted traffic that uses Onion Router (Tor).
  - Encrypted application classification on TLS 1.3 (EAC-1.3): Classification on encrypted traffic over new encryption protocol TLS 1.3.

  The training results of ET-BERT demonstrated an accuracy of over 97% across all five downstream tasks, showcasing superior performance

[1] The GitHub repository for the datasets, including CSTNET, used in the training of ET-BERT is available at https://github.com/linwhitehat/ET-BERT/tree/main/datasets.

Table 1 Performance of ET-BERT on five downstream tasks

Tasks	Datasets	Traffic level	Accuracy	Precision	Recall	F1-score
GEAC	Cross-platform (iOS) [18]	Flow	0.9844	0.9701	0.9632	0.9643
		Packet	0.9810	0.9757	0.9772	0.9754
	Cross-platform (Android) [18]	Flow	0.9865	0.9323	0.9266	0.9246
		Packet	0.9728	0.9439	0.9119	0.9206
EMC	USTC-TFC [25]	Flow	0.9929	0.9930	0.9930	0.9930
		Packet	0.9915	0.9915	0.9916	0.9916
ETCV	ISCX-VPN-Service [4]	Flow	0.9729	0.9756	0.9731	0.9733
		Packet	0.9890	0.9891	0.9890	0.9890
	ISCX-VPN-App [4]	Flow	0.8519	0.7508	0.7294	0.7306
		Packet	0.9962	0.9936	0.9938	0.9937
EACT	ISCX-Tor [10]	Flow	0.8311	0.5564	0.6448	0.5886
		Packet	0.9921	0.9923	0.9921	0.9921
EAC-1.3	CSTNET-TLS 1.3 [12]	Flow	0.9510	0.9460	0.9419	0.9426
		Packet	0.9737	0.9742	0.9742	0.9741

compared to existing deep learning-based ETA models (10 types based on CNN and 1 type based on LLM). The performance of the fine-tuned ET-BERT for each task is presented in the Table 1.

3.2 Attack Methodology

In this paper, we perform adversarial attacks targeting ET-BERT trained to identify and classify applications within encrypted traffic. Our proposed attack generates adversarial examples that lead to misclassification during the execution of downstream tasks, following the model's training. These adversarial examples are designed to:

1. Mirror the characteristics of genuine network traffic, in terms of patterns and statistical properties.
2. Conform to the requisite structure and rules for input into the ETA model.

We established these characteristics based on the properties typical of adversarial examples in natural language processing models. Adversarial examples typically involve data alterations that are imperceptible to humans but deleterious to the performance of the trained model. Unlike continuous data, whose significance may be influenced by quantity, the data utilized in natural language processing is discrete and categorical, bearing semantic value influenced by its inherent meaning.

To attain the adversarial example generation goal, we create semantically identical and grammatically correct examples that can adversely affect the results of natural language processing models while preserving linguistic

fluency. In this context, we reconceptualize fluency in terms of network traffic, aiming to generate adversarial examples that induce misclassification in application classification tasks by the ETA model.

Fluency in terms of network traffic. Network traffic data consists of binary sequences constructed in accordance with communication protocols, forming a language not readily comprehensible to humans. Thus, it lacks explicit semantic content accessible to human readers. However, the ability to enact networking functions defined by network packet data implies that the data harbors implicit semantic content intelligible to machines (i.e., network devices and applications). In this light, *vocabulary* can be likened to commands within TCP/IP, TLS protocols, and so on. Furthermore, to transmit commands effectively over a network, source, destination, and content (payload) must adhere to stringent rules akin to *grammatical structures* in human languages.

With the aforementioned conception of network traffic fluency, we define a target LLM-based model, analyze it in terms of semantic and syntactic elements peculiar to network traffic, and undertake adversarial attacks predicated on these analyses. The subsequent section provides a detailed account of this process.

3.3 Generation of Adversarial Examples Against ET-BERT

The adversarial attacks on ET-BERT described in the previous section target the model trained for downstream tasks in the fine-tuning part, aiming to generate traffic examples that preserve fluency from a network traffic perspective. We have designed an adversarial attack algorithm that maintains network traffic fluency in terms of *grammar correctness* and *fluent lexical capacity*, based on an analysis of the dataset pre-processing and ET-BERT's learning process. A detailed explanation is as follows.

Grammarly correctness. To figure out the impact of arbitrary modifications on the data input to the fine-tuned model, let's first examine the generating steps for the ET-BERT fine-tuning dataset. The generation steps involves the following detailed tasks to convert the raw traffic data into token form.

1. Raw traffic files are segregated into individual pcap files based on the flow, defined by having the same IP, port, and protocol. During this process, packets irrelevant to specific content transmission, such as those from the address resolution protocol (ARP) and the dynamic host configuration protocol (DHCP), are excluded.

2. For each packet in the saved flow-specific pcap files, the header information comprising a total of 38 bytes, including the Ethernet header (14 bytes), IP header (20 bytes), and the source and destination port in TCP header (4 bytes), is removed. Subsequently, if the model for fine-tuning operates at the flow-level, data is extracted from 5 consecutive pre-processed packets within the flow, taking 128 bytes from each and concatenating them to form a single datagram. If the model for fine-tuning operates at the packet-level, 128 bytes of data are extracted from a single pre-processed packet to form a datagram.
3. Finally, Bigram encoding is performed on the constructed datagram to form the datagram used for training, and each datagram is labeled to create the dataset.

The first step in the fine-tuning dataset creation process simply sorts packets by flow, thus not altering the shape of data within the packet. Hence, from a network grammar perspective, the first step need not be considered. Subsequent to this, the second step involves packet header delimitation, where headers below the TCP layer, including the source and destination port information, are removed. Therefore, any arbitrary network routing format and source/destination information can be written within the Ethernet and IP headers below the TCP layer without issue. Furthermore, if arbitrary modifications occur in the header information following the TCP header's sequence number or in the payload, it is considered a TCP payload data error, which is handled at the transport layer through error detection or retransmission or is resolved at the application layer through error processing. In other words, they are considered manageable errors that do not cause problems in transmission or reception by network devices and can be processed normally. The final third step involves applying Bigram encoding to the datagrams to create the dataset, transforming them into a format suitable for ET-BERT input. When creating adversarial examples, it is possible to generate such examples without preserving the Bigram encoding format; however, from the perspective of the ET-BERT model, this could be considered grammatically incorrect data. Therefore, based on the analysis of the fine-tuning dataset creation process, we construct attacks that generate adversarial traffic examples through payload modifications while preserving the Bigram encoding format.

Fluent lexical capacity. In addition to the grammar correctness characteristics previously discussed, the fluency of network traffic should also satisfy semantic similarity. While adversarial examples in natural language processing can be created using semantically similar synonyms or antonyms, generating semantically identical traffic in network traffic analysis

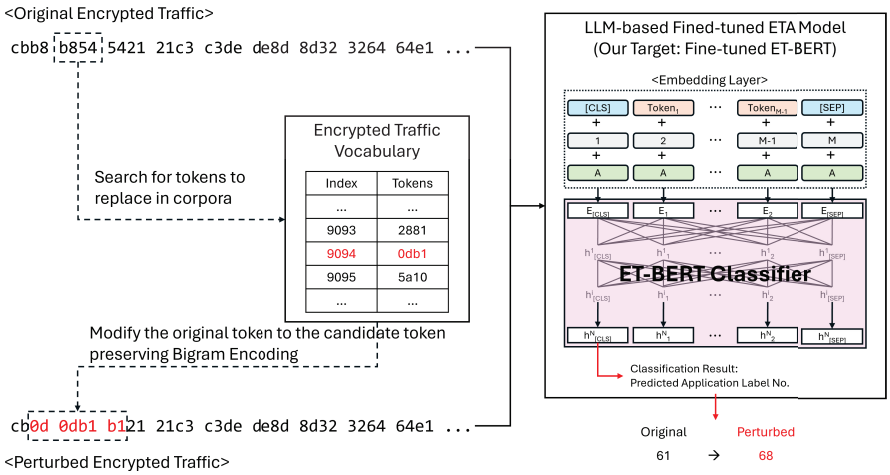

Figure 1 The proposed adversarial attack against the fine-tuned ET-BERT.

is challenging since humans cannot directly discern meaning from traffic data. In other words, constructing packets that differ in data values but still enable the same operation to be performed is a difficult problem, especially with encrypted traffic. As an alternative solution to this problem, we devised an approach to generate adversarial examples based on patterns frequently appearing in actual network traffic, utilizing the vocabulary of the pre-trained ET-BERT, which forms the basis for the fine-tuning of ET-BERT. As explained in Section 3.1, the pre-trained ET-BERT employs a vocabulary composed of patterns that frequently occur in traffic for its training on the encrypted traffic corpus. Thus, this vocabulary can be considered ET-BERT's lexical capacity. By using patterns existing in the vocabulary to generate adversarial examples, we aimed to produce network traffic that bears similar patterns to actual network traffic.

Considering the grammar correctness and fluent lexical capacity we have discussed so far, we propose an attack that generates adversarial examples by preserving the Bigram encoding format and finding replacement keywords within ET-BERT's vocabulary, as illustrated in Figure 1. The detailed attack algorithm is as Algorithm 1. The proposed attack has been modified to reflect the grammatical characteristics of ET-BERT based on the adversarial attack algorithm for the BERT model presented in [9].

Given a datagram sentence $X = x_1, x_2, ..., x_n$ and a target model F, the problem of our adversarial attack algorithm is to find a modified X' from parts of X such that $F(X) \neq F(X')$. Here, X' must be contextually similar to X such that the similarity of the results performed by word embedding is

Algorithm 1 Proposed adversarial encrypted traffic example generation algorithm for fine-tuned ET-BERT

Input: A burst sample $X = \{x_1, x_2, ..., x_n\}$, the corresponding ground truth label Y, target model F', word embedding similarity function $WordSim$, sentence similarity threshold ϵ, word embeddings database over the vocabulary $Vocab$.
Output: Adversarial example X_{adv}

1: **for** each word x_i in X **do**
2: Compute the importance score $I_{x_i} = F(X) - F(X_{\setminus x_i})$
3: **end for**
4:
5: Create a set W of all words $x_i \in X$ sorted by the descending order of their importance score I_{x_i}.
6: Initiate the set of candidates C by extracting the full tokens in *Vocab* such that each token t matches the regular expression pattern for 4-digit hexadecimal numbers.
7: *BigramC* $\leftarrow \{\}$
8: **for** $c_k \in C$ do **do** ▷ Transformation with Bigram Encoding Rule
9: $X' \leftarrow$ Replace w_i with c_k in X
10: **if** $i > 0$ **then**
11: $c_{k^{pre}} \leftarrow$ the first two digits of x_{i-1}||the first two digits of c_k
12: Replace x_{i-1} with $c_{k^{pre}}$ in X'
13: **end if**
14: **if** $i < len(X) - 1$ **then**
15: $c_{k^{post}} \leftarrow$ the last two digits of c_k||the last two digits of x_{i+1}
16: Replace x_{i+1} with $c_{k^{post}}$ in X'
17: **end if**
18: **if** $WordSim(X', X) > \epsilon$ **then**
19: Add the sequence of tokens $\{c_{k^{pre}}, c_k, c_{k^{post}}\}$ to the set *BigramC*.
20: (Note that $c_{k^{pre}}$ or $c_{k^{pre}}$ may not exist according to the index i.)
21: $Y_k \leftarrow F(X')$
22: **end if**
23: **end for**
24: **if** there exists c_k whose prediction result $Y_k \neq Y$ **then**
25: In *BigramC*, only keep the candidate sequences $\{c_{k^{pre}}, c_k, c_{k^{post}}\}$ whose prediction result $Y_k \neq Y$
26: $\{c^*_{k^{pre}}, c^*_k, c^*_{k^{post}}\} \leftarrow \text{argmax}_{c \in BigramC} Sim(X, X_{\{x_{i-1}, x_i, x_{i+1}\} \rightarrow \{c_{k^{pre}}, c_k, c_{k^{post}}\}})$
27: $X_{adv} \leftarrow$ Replace $\{x_{i-1}, x_i, x_{i+1}\}$ with $\{c_{k^{pre}}, c_k, c_{k^{post}}\}$ in X_{adv}
28: **return** X_{adv}
29: **end if**
30: **return** None

greater than ϵ (i.e., $WordSim(X, X') > \epsilon$). To solve the defined problem, we calculate the word importance score I_{x_i} for each word in X and store the words in W in descending order of their influence on the model's output when modified (lines 1–5). Here, I_{x_i} [11] is defined by the difference in F's

output for X and for the sentence $X_{\setminus x_i}$, which x_i has been deleted, such that $I_{x_i} = F(X) - F(X_{\setminus x_i})$. Subsequently, we construct the set of candidate words C by extracting tokens from ET-BERT's vocabulary that are composed of four hexadecimal digits (line 6). Using the tokens in the candidate set, we modify words in X to form X' and calculate the model output $Y_k = F(X')$ for X' that satisfies $WordSim(X, X') > \epsilon$ (lines 7–23). Here, to satisfy the characteristics of Bigram Encoding, the first two digits of the candidate token c_k are reflected in the word preceding the target word, and the last two digits are reflected in the word following the target word (lines 10–17). Among the results of the preceding processes, the one with the highest similarity and satisfying $Y_k \neq Y$ is returned as the adversarial example X_{adv}.

4 Experiments

4.1 Implementation

We have practically implemented the proposed adversarial attack using the TextAttack library. TextAttack is a Python library framework introduced by Morris et al. [16] that allows for the implementation of adversarial attacks on trained models. To implement an adversarial attack with TextAttack, one must write an attack recipe that describes the attack's objective and method. An attack recipe consists of a `(search method)` for finding target words for replacement, a `(goal function)` that serves as the attack's objective function, a `(transformation)` method for word replacement, and `(constraints)` that the outcome of the replacement must satisfy. The attack recipe for Algorithm 1 is as follows.

```
Attack(
    (search_method): GreedyWordSwapWIR(delete)
    (goal_function):  UntargetedClassification
    (transformation):  CustomWordSwapEmbedding
    (constraints):
        (0): WordEmbeddingDistance(
            (embedding):  WordEmbedding
            (min_cos_sim):  0.9
            (cased):  False
            (include_unknown_words):  True
            (compare_against_original):  True)
        (1): RepeatModification
        (2): StopwordModification
)
```

Here, `CustomWordSwapEmbedding` is implemented to replace target words according to the bigram encoding rules using four-digit hexadecimal numbers from the encrypted traffic vocabulary (Algorithm 1 lines 8–23). The implementation can be found in our Github repository.[2]

4.2 Experimental Settings

We constructed fine-tuned ET-BERT models for three tasks (EMC, ETCV, and EAC-1.3) and conducted adversarial attacks on them. We could not perform the same fine-tuning for the remaining two tasks (GEAC and EACT) among the five tasks proposed in [12] because the datasets required for fine-tuning these two downstream tasks were not disclosed in the authors' Github repository. Attempts to compile datasets for these tasks with data from other sources failed due to the pre-processing code released by the authors not handling numerous errors effectively, thus constructing completely identical datasets was challenging. Therefore, we resorted to using the pre-processed downstream datasets (USTC-TFC, ISCX-VPN-Service, ISCX-VPN-App) provided by the authors to build the fine-tuned ET-BERT models for our adversarial attack experiments.

For the experiments, adversarial examples were generated from 100 samples extracted from each downstream dataset, with the threshold ϵ for $WordSim$ set to 0.9. Moreover, the experiments were performed in an environment featuring an Intel Xeon Gold 6230R CPU @ 2.10GHz and a single NVIDIA GeForce RTX 3090 GPU.

4.3 Experimental Results

The results of the proposed adversarial attacks on the EMC, ETCV, and EAC-1.3 tasks are shown in Tables 2 and 3. Table 2 presents the generated adversarial examples (perturbed traffic data) and the corresponding classification results. The generated adversarial examples cause the model's output to shift from the original labels to different ones. For instance, in the EAC-1.3 Task, altering the first two words in the first traffic example changes the classification from class 68, which was predicted with 100% probability, to class 52 with a 49% probability. This indicates that although the attack model could clearly determine the original data, it failed to classify the adversarial examples with high certainty. In the case of ISCX-VPN-App in the ETCV

[2]The GitHub repository for our implementations of the proposed adversarial attacks is available at https://github.com/BJSeok/AdversarialAttacksET-BERT.

Table 2 Examples of perturbed traffic data of three downstream tasks (EMC, ETCV, and EAC)

Tasks	Datasets	Original data	Label (prob.)	Perturbed data	Label (prob.)
EMC	USTC-TFC	e240 4067 67c0 c032 32e9 e967 67bf bfe4 e4f4 f480 8018 1821 21f0 f077 77a0 a000 0000 0001 0101 0108 080a 0a06 06c5 c561 61bc bcf5 f573 73aa aada da01 0100 0059 597d 7d12 1266 667b 7b40 4053 5306 0647 4745 451e 1e65 654c 4c05 056a 6a50 5074 744a 4a7e 7e79 7901 0100 0000 0000 0000 0000 0000 0000 0000 004a 4a9c 9cc1	17 (100%)	e240 405b 5bc0 c032 32e9 e967 67bf bfe4 e4f4 f480 80d7 d721 21f0 f077 77a0 a01b 1b00 0001 0101 0108 0827 2706 06c5 c561 61bc bcf5 f573 73aa aada da01 0100 0059 597d 7d12 1266 667b 7bec ec53 5306 0671 7145 451e 1e65 654c 4c05 056a 6a50 5074 7452 527e 7e79 7901 0100 0000 0000 0000 0000 0000 0000 004a 4a9c 9cc1	8 (93%)
ETCV	ISCX-VPN-Service	12d4 d400 0000 0001 0101 0108 080a 0a6e 6e29 29be bed0 d000 0015 1506 0632 3217 1703 0303 0300 002a 2a9f 9f0a 0a03 0392 921f 1fc4 c457 57a4 a414 14c4 c47e 7e9e 9eb1 b10e 0e59 59ae aeae aef4 f405 050f 0fed edc0 c019 1963 63cc ccec ec8d 8d1c 1cdc dc23 2324 24b0 b022 220a 0a71 7103 030e 0e94 9407 0733 3399 9925	6 (100%)	12d4 d400 0000 0001 0101 0108 080a 0a1a 1a29 29be bed0 d000 0015 1506 0632 3217 1703 0303 0300 002a 2a9f 9f0a 0a03 0392 921f 1fc4 c457 57a4 a414 14c4 c47e 7e9e 9eb1 b10e 0e59 59ae aeae aef4 f405 050f 0fed edc0 c019 1963 63cc ccec ec8d 8d1c 1cdc dc23 2324 24b0 b022 220a 0a71 7103 030e 0e94 9407 0733 3399 9925	8 (100%)
	ISCX-VPN-App	1191 9155 5530 303e 3e31 316a 6a78 78d7 d798 9850 5010 1000 00ed edf4 f471 7100 0000 00cc cc24 245f 5f16 166b 6b52 525e 5e2c 2cd2 d2df dfdc dc7d 7de7 e712 12bf bf02 020d 0d18 1834 3442 4202 02a9 a93d 3dd0 d070 70ff ff0e 0e18 1819 1934 34e0 e0b0 b0f4 f4cb cb1f 1fa5 a5de de3e 3e13 1366 6637 378b 8bb9 b928 28dd dd9e 9e81	11 (100%)	1191 9155 5530 303e 3e31 316a 6a78 78d7 d798 9850 5010 1014 14ed edf4 f471 7100 0000 00cc cc24 245f 5f16 166b 6b52 525e 5e2c 2cd2 d2df dfdc dc7d 7de7 e712 12bf bf02 020d 0d18 1834 3442 4202 02a9 a93d 3dd0 d070 70ff ff0e 0e18 1819 1934 34e0 e0b0 b0f4 f4cb cb1f 1fa5 a5de de3e 3e13 1366 6637 378b 8bb9 b928 28dd dd9e 9e81	2 (96%)
EAC-1.3	CSTNET-TLS 1.3	e57a 7af9 f9d8 d828 2874 74fb fb4c 4ca4 a4af af80 8018 1801 01f5 f5f1 f112 1200 0000 0001 0101 0108 080a 0aad ad90 9064 6433 334d 4d8d 8d2b 2b08 0835 35eb eb46 462a 2a9b 9b2a 2a90 90a9 a97e 7e96 968d 8d4c 4c82 826a 6a6b 6b5a 5a88 888b 8b70 708a 8ad4 d4dd dd16 1697 97dc dc6c 6c88 8843 43a3 a3d1 d1f9 f929 29d6 d6ee eeb7	68 (100%)	e501 01f9 f9d8 d828 2874 74fb fb4c 4ca4 a4af af80 8018 1801 01f5 f5f1 f112 1200 0000 0001 0101 0108 080a 0aad ad90 9064 6433 334d 4d8d 8d2b 2b08 0835 35eb eb46 462a 2a9b 9b2a 2a90 90a9 a97e 7e96 968d 8d4c 4c82 826a 6a6b 6b5a 5a88 888b 8b70 708a 8ad4 d4dd dd16 1697 97dc dc6c 6c88 8843 43a3 a3d1 d1f9 f929 29d6 d6ee eeb7	52 (49%)

Table 3 Adversarial attack results on different tasks

Tasks	Datasets	Success rate	Avg perturbed word %	Avg # of queries
EMC	USTC-TFC	96%	7.5%	767.32
ETCV	ISCX-VPN-Service	99%	8.14%	964.8
	ISCX-VPN-App	100%	5.28%	535.49
EAC-1.3	CSTNET-TLS 1.3	97%	3.43%	317.1

Task, there were even adversarial examples that presented a 100% probability for the changed class.

Table 3 provides a summary of the adversarial attacks on 100 samples across four datasets corresponding to the three tasks. As can be seen, all three tasks were susceptible to adversarial attacks with a success rate exceeding 96%, and notably, the adversarial attacks on the ISCX-VPN-App samples achieved a 100% success rate. In terms of attack difficulty, the proportion of words perturbed in the datagram sentence was mostly less than 9%, indicating that attacks could lead to misclassification with only minor alterations to the

original sentence, with the average number of queries being less than 965. Specifically, for the CSTNET-TLS 1.3 dataset, a very slight word change of 3.43% was sufficient for misclassification, with the average number of queries to generate adversarial examples being the lowest at 317.1.

5 Conclusion

In this paper, we performed adversarial attacks on encrypted traffic analysis models based on large language models, and the target model is the ET-BERT model, a state-of-the-art encrypted traffic model founded on the BERT model. To carry out adversarial attacks on the fine-tuned ET-BERT model that processes network traffic in text form, this paper defines fluency from the perspective of network traffic. The fluency means having patterns similar to actual network traffic while preserving network protocol rules and the data processing form of the ET-BERT model. To generate adversarial examples that preserve such network traffic fluency, we proposes an algorithm that performs sentence manipulation according to the rules of Bigram encoding, which is the data processing form handled by the fine-tuned ET-BERT model, by finding replacement words within the encrypted traffic vocabulary of the pre-trained ET-BERT model. The proposed algorithm is designed to preserve both the data processing form and lexical capacity of ET-BERT, thus maintaining grammar correctness and fluent lexical capacity from the perspective of network traffic. The proposed attack was experimentally evaluated on the fine-tuned ET-BERT models trained with datasets USTC-TFC, ISCX-VPN-App, ISCX-VPN-Service, and CSTNET-TLS 1.3, and the results indicated that it was possible to generate adversarial examples with a success rate of over 96%. As demonstrated by the results of this paper, LLM-based ETA models are vulnerable to adversarial attacks. This indicates a potential vulnerability to zero-day attacks, malicious data transmission, and other security concerns from a network security perspective, necessitating immediate countermeasures. In this regard, we plan to conduct future research on enhancing the robustness of LLM-based ETA models through additional training against adversarial examples or employing GAN-based training strategies.

Acknowledgement

This study was supported by the Research Program funded by the SeoulTech(Seoul National University of Science and Technology

References

[1] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. Independent comparison of popular dpi tools for traffic classification. *Computer Networks*, 76:75–89, 2015.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Zulong Diao, Gaogang Xie, Xin Wang, Rui Ren, Xuying Meng, Guangxing Zhang, Kun Xie, and Mingyu Qiao. Ec-gcn: A encrypted traffic classification framework based on multi-scale graph convolution networks. *Computer Networks*, 224:109614, 2023.

[4] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414, 2016.

[5] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

[6] Hong Ye He, Zhi Guo Yang, and Xiang Ning Chen. Pert: Payload encoding representation from transformer for encrypted traffic classification. In *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*, pages 1–8. IEEE, 2020.

[7] Xinyi Hu, Chunxiang Gu, Yihang Chen, and Fushan Wei. Cbd: A deep-learning-based scheme for encrypted traffic classification with a general pre-training method. *Sensors*, 21(24):8231, 2021.

[8] Auwal Sani Iliyasu and Huifang Deng. Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks. *Ieee Access*, 8:118–126, 2019.

[9] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[10] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of tor traffic using time based features. In *International Conference on Information Systems Security and Privacy*, volume 2, pages 253–262. SciTePress, 2017.

[11] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*, 2020.

[12] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022*, pages 633–642, 2022.

[13] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, 2020.

[14] Zineb Maasaoui, Anfal Hathah, Hasnae Bilil, Van Sy Mai, Abdella Battou, and Ahmed Lbath. Network security traffic analysis platform-design and validation. In *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–5. IEEE, 2022.

[15] Andrew McCarthy, Essam Ghadafi, Panagiotis Andriotis, and Phil Legg. Defending against adversarial machine learning attacks using hierarchical learning: A case study on network traffic attack classification. *Journal of Information Security and Applications*, 72:103398, 2023.

[16] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020.

[17] Eva Papadogiannaki and Sotiris Ioannidis. A survey on encrypted network traffic analysis applications, techniques, and countermeasures. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[18] Jingjing Ren, DJ Dubois, and D Choffnes. An international view of privacy risks for mobile apps, 2019.

[19] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine*, 57(5):76–81, 2019.

[20] Amir Mahdi Sadeghzadeh, Saeed Shiravi, and Rasool Jalili. Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification. *IEEE Transactions on Network and Service Management*, 18(2):1962–1976, 2021.

[21] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[22] Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, and Xiaojiang Du. Accurate decentralized application identification via encrypted

traffic analysis using graph neural networks. *IEEE Transactions on Information Forensics and Security*, 16:2367–2380, 2021.

[23] Hongtao Shi, Hongping Li, Dan Zhang, Chaqiu Cheng, and Xuanxuan Cao. An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. *Computer Networks*, 132:81–98, 2018.

[24] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25(5):355–374, 2015.

[25] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International conference on information networking (ICOIN)*, pages 712–717. IEEE, 2017.

[26] Shuang Zhao, Jing Li, Jianmin Wang, Zhao Zhang, Lin Zhu, and Yong Zhang. attackgan: Adversarial attack against black-box ids using generative adversarial networks. *Procedia Computer Science*, 187:128–133, 2021.

Biographies

Byoungjin Seok is working as a Research Professor at Korea University. Dr. Seok received his Ph.D. degree from the Graduate School of the Department of Computer Science and Engineering at Seoul National University of Science and Technology (SeoulTech), Korea, in February 2023. After receiving his Ph.D. degree, he worked as a senior researcher at the Research Center of Electrical and Information Technology at SeoulTech in 2023.

Kiwook Sohn is working as a Professor at the Department of Computer Science and Engineering, Seoul National University of Science and Technology (SeoulTech), Korea. Professor Kiwook Sohn received his Ph.D. degree at the Graduate School of the Department of Electrical and Computer Engineering, Sungkyunkwan University, Korea. In 1992–1999, he worked at the Electronics and Telecommunications Research Institute (ETRI), Korea. In 2000–2023, he held a position at the National Security Research Institute (NSR), Korea.