Efficient Character Edit Method - A Black-box Textual Adversarial Attack on Machine Learning Models in Web Security

Yixiao FEI
SPEIT
Shanghai Jiao Tong University
Shanghai, China
targaryen@sjtu.edu.cn

Lei WANG
SPEIT
Shanghai Jiao Tong University
Shanghai, China
thorray@sjtu.edu.cn

Qinming LIU
SPEIT
Shanghai Jiao Tong University
Shanghai, China
liuqingminlaurent@gmail.com

Nan ZONG
SPEIT
Shanghai Jiao Tong University
Shanghai, China
cisnazong@sjtu.edu.cn

Xinzhi MU
SPEIT
Shanghai Jiao Tong University
Shanghai, China
draconidsmxz@gmail.com

Jialiang LU
SPEIT
Shanghai Jiao Tong University
Shanghai, China
jialiang.lu@sjtu.edu.cn

Ruan HE
Network Security
Tencent
Shanghai, China
ruanhe@tencent.com

***Abstract*—Researchers are making efforts on detection and prevention in web security by deploying machine learning models, while many machine learning models are vulnerable to samples which carry minor perturbation. So far, it is unclear how much risk adversarial perturbation counts for the safety of the detection of web application since in most cases, attackers generate such adversarial examples by having knowledge of detailed model information.**

The objective of the work is to address this problem and propose a textual adversarial attack which requires only detection labels and is applicable to real-world black-box detection models such as intrusion detection systems. Here we introduces Efficient Character Edit Method (ECEM) that begins with a large adversarial perturbation and seeks to minimize the perturbation while staying adversarial. This adversarial attack outperforms Fast Gradient Sign Method (FGSM), a gradient-based setting, and Boundary Attack, a decision-based attack from which our inspiration comes. The comparison is done on various machine learning models with CSIC 2010 HTTP data set, and two open data sets from Kaggle.com. We include experiments to show that the generated adversaries are natural, legible to humans, and useful in evaluating and analyzing black-box classifiers. ECEM in particular and adversarial attacks in general in web security pave a novel way to the further research in the safety of machine learning models with textual inputs.

***Index Terms*—adversarial example, machine learning, black-box, web security, intrusion detection**

I. Introduction

Instead of human reasoning, machine learning is the foundation that builds smart systems of detection and prevention models in web security, but they have been shown vulnerable to adversarial examples which are maliciously perturbed examples much similar to original samples in human perception, but cause models to make incorrect decisions. [1] The vulnerability of machine learning models to adversarial examples implies a security risk in applications with real-world consequences, such as intrusion detection system, web application firewall and comment toxicity analyzer. The study of adversarial examples is thus necessary to identify the limitation of current machine learning algorithms, provide a metric for robustness, investigate the potential risk, and suggest ways to improve the robustness of models. [2]

Although image adversarial attack is largely studied in computer visions, recent research of textual adversarial attack in web security or in natural language processing seems insufficient. The concept of adversarial attack comes originally from models in computer vision [1] and both FGSM and Boundary Attack mentioned above are the fruits of continuous research. However, applying image adversarial attack to models for textual inputs requires having access to the embedding layer, which is impractical. Studies on textual adversarial attack is a necessity for countless applications in real word. Major difficulties of textual adversarial attacks are reflected on three inherent proprieties: discreteness, perceivability and semantics. Textual inputs are discrete features while image inputs are continuous. Little modification of pixels is hard for a human to distinguish but change of characters or words are obvious, which is also the reason of unchanged semantics of images and possible change of semantics of textual inputs. Methods for textual adversarial attack entail mainly searching in underlying semantic space, character flipping and word substitution.

Important criteria for an adversarial example are mainly the similarity metric, the attack goal and the threat model. There are three widely-used similarity metrics to quantify similarity between perturbed and original images, all of which are L_p distance, with $p = \infty, 2, 0$. They have been proposed as an approximation of human perceptual distance. Cosine similarity and Levenshtein distance are two common metrics to assess the difference between adversarial and target examples for words and texts. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. [3]

The attack goal would be either targeted or untargeted. An untargeted adversarial attack would be successful when the prediction of the input after perturbation is flipped to any one but the original class, while a targeted adversarial attack has to ensure that the prediction class of the adversarial example is a specific target. Most of the adversarial attacks is capable of switching between two types of attacks via a change of loss function. [4]

Threat models fall into two categories: white-box and black-box. Unlike the white-box setting, the black-box setting can only have access to the outputs of the victim model without any knowledge of the model architecture or hyper-parameter information. Based on whether the attacker is able to obtain the full probability or the label of a given input, black-box attacks are further divided into score-based and decision-based. FGSM is a white-box attack since the generation of the perturbation requires the gradient information [5], while Boundary Attack is a black-box and more precisely, decision-based attack. [6]

This paper emphasizes the consequences of adversarial examples and proposes a black-box textual adversarial attack for machine learning models in web security. The generation of adversarial examples adopts a greedy search which starts with a large perturbation and approaches the original example as close as possible. The attack algorithm is referred as Efficient Character Edit Method because each iteration involves one character edit and the total number of edits is bounded by a small value. Figure 1 shows the overview of the ECEM. ECEM could be either targeted or untargeted, depending on the attack's willing. Our experiments demonstrate that ECEM requires significantly fewer model queries than Boundary Attack and possesses the advantage of generating much more similar adversarial examples than FGSM. Furthermore, ECEM shows its universality to various machine learning models and different data sets as long as inputs are textual, while Boundary Attack needs to procure the values after the embedding layer and FGSM requires in addition the decision function derivable. The fact that both FGSM and Boundary Attack treat values after embedding layer as images might cause unexpected results. In summary, contributions of this paper are:

- Consequences of adversarial examples are analyzed in real-world applications.
- ECEM, a black-box textual adversarial attack is proposed which requires no machine learning model structure and weights information.
- Performance of ECEM, FGSM and Boundary Attack is evaluated on various data sets.
- Practical applicability of ECEM to general web applications is shown.

The paper proceeds as follows. Section II presents background on techniques in web security and problems of machine learning, followed by definitions of some basic knowledge. Section III sketches our approach to generate adversarial examples. In Section IV, experiments of three adversarial attacks are evaluated in different conditions. Section V demonstrates application examples of our adversarial attack. In Section VI we discuss related work, and conclusion and future work are included in Section VII.

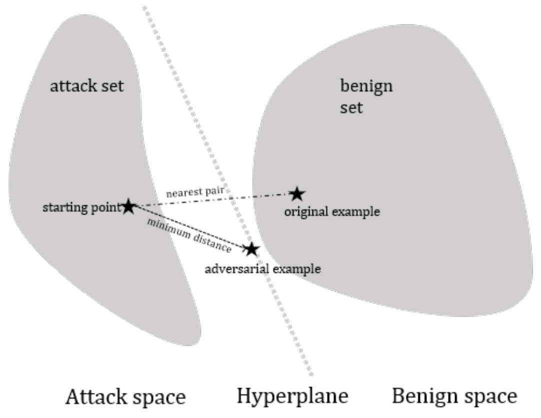


Fig. 1. Overview of Efficient Character Edit Method as a textual adversarial attack. For an original example, a nearest starting point in the opposite set is selected and approaches to the original example as much as possible. The last example would be the nearest adversarial example as the output.

II. Background

A. Techniques in Web Security

A variety of techniques have been developed for detection models in web security. There are two basic detection methods currently being adopted in research, anomaly-based and signature-based. [7]

- Anomaly-based: Anomaly-based techniques detects unknown attacks learning the existing patterns for benign access. Regrettably, performance and accuracy suffer with high false positive.
- Signature-based: Signature-based techniques rely on predefined rules of attack signatures, which achieves high accuracy for known attacks and is less prone to false positives. Unfortunately, Zero-day attack is hard to be detected.

Machine learning is the ability to automate solving problems and tasks by learning immanent patterns from large amounts of data. Detection models based on machine learning could be either anomaly-based or signature-based, or a mixture which combines advantages of two modes. [8]

B. Adversarial Attack

Adversarial examples aim at causing target model to make a mistake on prediction. The existence of adversarial examples arises worries about the safety of machine learning models. No matter it is an intentional or unintentional adversarial attack, evaluating adversarial examples has become a trend of building a robust deep learning model and understanding the shortcomings of models.

In machine learning, systems which employ offline learning alter no longer their prediction function when the initial training phase has been completed. In web security, injecting tons of adversarial examples into such systems may significantly increase false positive rate and drop the prediction accuracy. The worst case is for example that some malicious accesses finally disguise themselves and successfully cheat the intrusion detection systems. In [9], a backdoor could be implemented for neural networks to make them perform well during training step while poorly during evaluation step.

On the other hand, online learning is a method in which the prediction model is trained by some data and should be updated to a better predictor with more data in the future. By flooding with adversarial examples, machine learning models could be totally changed or dysfunctional. Figure 2 demonstrate how a machine learning model changes its prediction function after learning new data. In [10], the error rates of systems from Amazon and Google are respectively up to 96.19% and 88.94% after adversarial attack. Also in 2016, Microsoft's AI chatbot, Tay, became extremely racist in less than 24 hours after learning comments on Twitter. [11]

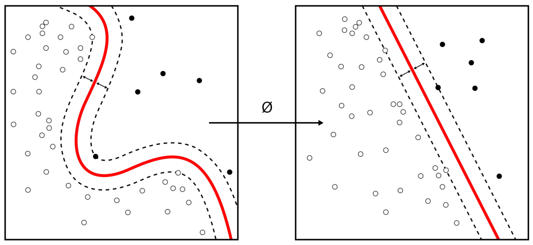


Fig. 2. The prediction function changes fundamentally after learning new data. Online learning is supposed to eliminate bias and overfitting but models could also be poisoned by malicious intention.

C. Similarity Metrics

Out of generality, words, texts, logs, URLs and all the textual inputs could be considered as strings in computer science. We have applied Ratcliff-Obershelp pattern recognition for strings [12] as an essential similarity metric throughout the paper, explained as follows.

Definition 1. Ratcliff-Obershelp pattern recognition: Compute the similarity of two strings as the number of matching characters divided by the total number of characters in the two strings. Matching characters are those in the longest common subsequence plus, recursively, matching characters in the unmatched region on either side of the longest common subsequence. [12]

If $\rho(\cdot,\cdot)$ represents the similarity between 2 strings, $|\cdot|$ calculates the number of characters of a string, $\cdot \bigcap \cdot$ gives the matching characters of two strings, and S_a and S_b are 2 strings. The formula is shown as equation (1)

$$\rho(S_a, S_b) = \frac{2 \times |S_a \bigcap S_b|}{|S_a| + |S_b|} \tag{1}$$

D. Levenshtein Distance

Levenshtein distance is a string metric for measuring the difference between two sequences, named after the Soviet mathematician Vladimir Levenshtein. [3]

Definition 2. Levenshtein Distance: Levenshtein Distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

Given two strings S_a and S_b, denote $\psi_{S_a,S_b}(i,j)$ as the Levenshtein Distance between the first i characters in S_a and the first j characters of S_b, thus Levenshtein Distance equals $\psi_{S_a,S_b}(|S_a|,|S_b|)$, where $|\cdot|$ is the length of a string. Its formula is defined in equation (2)

$$\psi_{S_a,S_b}(i,j) = \begin{cases} max(i,j) & ,if\ min(i,j) = 0 \\ min \begin{cases} \psi_{S_a,S_b}(i-1,j)+1 \\ \psi_{S_a,S_b}(i,j-1)+1 \\ \psi_{S_a,S_b}(i-1,j-1)+\mathbb{1}_{S_a[i]\neq S_b[j]} \end{cases} & ,otherwise \end{cases} \tag{2}$$

where $S_a\,[i]$ represents i-th character in S_a and $\mathbb{1}_{S_a[i]\neq S_b[j]}$ is an indicator function such that it is 1 if $S_a\,[i] \neq S_b\,[j]$, otherwise it is 0.

III. Efficient Character Edit Method

A. Victim Model

We assume a classification model which is prediction function F as our victim model, accepting an input S as a string and producing an output y which is treated as a probability distribution on classes. The input S will be embedded into a numeric expression x since textual inputs could not be directly used for numerical calculation and the model produces in addition a classifier C that assigns an input S with the class of maximum probability: $C(S) = \arg\max_c(F(S))_c$. Since our attack is decision-based, only the classifier C is accessible, but not the underlying model F.

Training data is crucial for a machine learning model. Without knowledge of training data, a synthetic training data set could be generated using a random feature vector method [13] or more sophisticated techniques, such as hill-climbing [14]. In addition, [15] and [16] have shown the possibility of extracting the training data from a machine learning model. With the training data set, attackers could benefit from more information for their attack techniques.

B. Optimization Framework

As mentioned in Section I, the attacked goal could be either targeted or untargeted. ECEM begins with a large adversarial perturbation and seeks to minimize the perturbation while staying adversarial – i.e., the label should never be the original class. For a targeted attack, the starting point is supposed to be chosen as the targeted class and stays the same during the iteration, while for an untargeted setting, choose any starting point whose label is different from the original one and it is apparently unnecessary to stay the same as long as it is not the original class. We are going to present the targeted attack by default in the following parts and the untargeted setting would also be mentioned for the slight difference of configuration.

ECEM is an iterative algorithm based on rejective sampling, initialized at an example that lies in the target class. At each step, a character edit is chosen from a proposal set, which reduces the Levenshtein Distance of the adversarial towards the original example. If the adversarial example still lies in the target class, the character edit is kept. Otherwise, the edit is dropped. The road map for generating an adversarial example is depicted as follows and Figure 3 presents the idea of generation:

1 Pick an original example.
2 Choose a starting point of the targeted class (or any class different from that of the original example for an untargeted goal).
3 Calculate the difference between the starting point and the original example.
4 Iterate the starting point by editing characters and approach to the original example.
5 Stop when no more character edit could be done while staying adversarial.

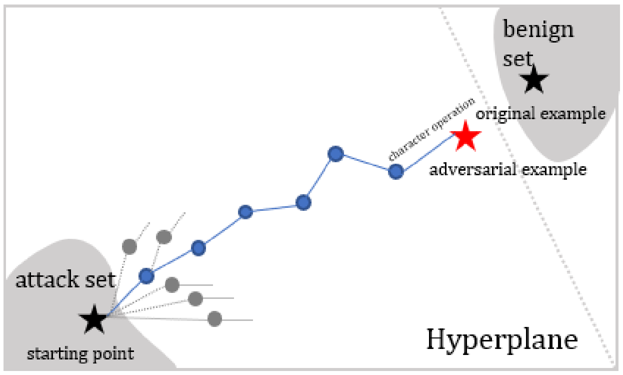


Fig. 3. The generation of an adversarial example begins with a starting point with a large perturbation and approaches to the original example. Each iteration involves a character edit until no more edit could be done while staying adversarial. The result would be the nearest adversarial example. In most cases, it is not unique. Therefore a greedy search in a specific direction is adopted.

If the victim model is confirmed with a classifier C, the generation of an adversarial example could be formulated as an optimization problem as follows:

$$\begin{aligned}
&\min_{S'} d(S, S') \\
&s.t. \quad \exists\ (S_i)_n \in \mathbb{S},\ (\phi_i)_{n-1} \in \Phi, \\
&\forall i \in [0, n-1]\,,\ S_{i+1} = \phi_i(S_i), \\
&\forall (i,j) \in [0,n]^2,\ C(S_i) = C(S_j),\ C(S_i) \neq C(S), \\
&and\ S_n = S'
\end{aligned} \tag{3}$$

where S is the original string, S' is the adversarial example, S_0 is the chosen starting point, $\mathbb{S}$ is the space of all strings, and Φ is the set of character operation – i.e., insertion, deletion and substitution of a character at a given position. d is the Levenshtein Distance function that $d(S_a, S_b) = \psi_{S_a,S_b}(|S_a|, |S_b|)$, mentioned in Section II-D.

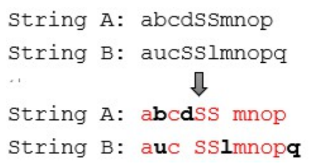
The complexity of such problem is $O(n!)$, because of the combination of possible orders of character operations, with n as the Levenshtein Distance between the starting point and the original example. If the original example is extremely long and complex, like a comment of a hundred words, and if the starting point is not wisely chosen, the distance could be several thousand. We know that $1000! \approx 10^{249}$. Hence the choice of the starting point is essential and a greedy search is adopted to accelerate the calculation process, which reduces the complexity significantly to $O(n)$. Details would be presented in the following parts.

C. Generating Adversarial Examples

Initialisation: To choose properly a starting point, training data set could be helpful, obtained either by training data extraction or a synthetic generation as mentioned in Section III-A. Using the training data set, we could separate the data set by their classes. For a targeted adversarial attack, choose the example as the starting point, with the target class and has the minimum Levenshtein Distance to the original example. While for an untargeted attack, the example with any class other than that of the original example and also has the minimum Levenshtein Distance would be the starting point. The reason for choosing such starting point is to minimize at the begining the Levenshtein Distance which determines the total complexity.

Operations: According the Levenshtein Distance, a minimum number of character operations required for transforming one string to the other could be inferred. In fact, these operations are already known when calculating the Levenshtein Distance. As shown in Figure 4, minimum operations for transforming String A to String B are as follows, where index begins with 0:

- Replace ‘b’ with ‘u’ at 1st position
- Delete ‘d’ at 3rd position
- Insert ‘l’ at 6th position
- Insert ‘q’ at 10th position



```
String A: abcdSSmnop
String B: aucSSlmnopq
              ⇓
String A: abcdSS mnop
String B: auc SSlmnopq
```

Fig. 4. This demonstrates the minimum operations for transforming String A to String B. Red parts are matching characters in two strings while black parts and spaces indicate character edits.

Greedy search: Instead of searching all the combinations of character operations, a greedy setting is adopted. As a rejective sampling, during each iteration, an operation would be chosen randomly from the possible operation set according to the Levenshtein Distance. If this operation could not make the example stay adversarial, we could safely drop this edit and choose another one. Once this character edit is done, the operation would be removed from the set. The searching ends when none of the operations in the set could keep the class label unchanged. Hence the approximately nearest adversarial example would be the last example during the search.

Each iteration reduces the Levenshtein Distance by 1 and when the distance is large, the number of trying possible operations would be exactly 1 because the example is surrounded by other examples with the same class label, which would be further supported in Section IV. The greedy setting successfully reduces the calculation complexity from $O(n!)$ to $O(n)$, with n as the distance between the starting point and the original example.

Attack Goal: As mentioned before, the attack goal could be either targeted or untargated. We could choose a corresponding starting point according to the rules in the initialisation. For the greedy search, the class label should always be maintained during the searching, which is the same as the target class, if the user is about to launch a targeted attack. While for the untargeted setting, the class label could be any class other than that of the original example. The class label is not required to be strictly maintained as the same all the time.

The detailed algorithm of targeted adversarial attack is presented in Algorithm 1, while for the untargeted setting, there would be only a tiny change of Line 12 to '**if** $C(S_{new}) \neq L_S$ **then**'.

Algorithm 1: Efficient Character Edit Method

```
Input: Starting point $S_0$; Original string $S$; Classifier $C$ of the victim model
Output: Approximately nearest adversarial example $S'$
 1: Let $L_S$ be the class label of $S$ passing $C$
 2: Let $L_{S'}$ be the class label of $S'$ passing $C$
 3: Verify $L_S \neq L_{S'}$
 4: Let $d$ be the Levenshtein Distance function
 5: $N \leftarrow d(S, S_0)$
 6: Generate $\Phi$ as the operation set of minimum required
    number of operations for transforming $S_0$ to $S$
 7: for $i = 0$ to $N - 1$ do
 8:     Shuffle $\Phi$
 9:     $S_{i+1} \leftarrow None$
10:     for $\phi$ in $\Phi$ do
11:         $S_{new} \leftarrow \phi(S_i)$
12:         if $C(S_{new}) = L_{S'}$ then
13:             $S_{i+1} = S_{new}$
14:             Remove $\phi$ from $\Phi$
15:             Break
16:         end if
17:     end for
18:     if $S_{i+1} = None$ then
19:         $S' = S_i$
20:         Break
21:     end if
22: end for
23: return $\bar{S}'$
```

D. Relation between Similarity and Distance

Each iteration decreases the Levenshtein Distance by 1. We are going to prove that such iteration also increases the string similarity, which would be an insertion, a deletion or a substitution of a character.

Recall the string similarity defined in Section II-C and suppose the similarity before the character edit equals as follows:

$$\rho(S, S_i) = \frac{2 \times |S \bigcap S_i|}{|S| + |S_i|}$$

Insertion: This edit would insert a character from S into S_i, so the length of S_i and the number of matching characters would both increase 1.

$$\rho(S, S_{i+1}) = \frac{2 \times (|S \bigcap S_i| + 1)}{|S| + |S_i| + 1} \geq \rho(S, S_i)$$

Deletion: This edit would delete an unnecessary character of S_i, so the length of S_i would decrease 1 while the number of matching characters remains the same.

$$\rho(S, S_{i+1}) = \frac{2 \times |S \bigcap S_i|}{|S| + |S_i| - 1} \geq \rho(S, S_i)$$

Substitution: This edit would replace an error character of S_i by the exact character of S at the same position, so the length of S_i remains the same while the number of matching characters would increase 1.

$$\rho(S, S_{i+1}) = \frac{2 \times (|S \bigcap S_i| + 1)}{|S| + |S_i|} \geq \rho(S, S_i)$$

IV. Experiment and Evaluation

A. Data sets

Three data sets are explored for our experiment: HTTP data set CSIC 2010, Malicious & Non-Malicious URL and Toxic Comment Classification.

The HTTP data set CSIC 2010 includes thousands of web requests, widely used for the testing of web attack protection systems. It was developed at the "Information Security Institute" of CSIC (Spanish Research National Council). [17]

Malicious & Non-Malicious URL contains 411247 URLs with labels indicating whether they are malicious for accessing. This is an open data set from Kaggle.com. [18]

Toxic Comment Classification is a data set of comments from Wikipedia's talk page edits. These comments contains different types of of toxicity like threats, obscenity, insults, and identity-based hate. Building accurate detection models based on this data set is a challenge from Kaggle.com. [19]

B. Comparison of three Adversarial Attack

In this part, ECEM, FGSM and Boundary Attack are evaluated on the HTTP data set CSIC 2010, according to the following criteria:

- String similarity
- Levenshtein Distance
- Score
- Success rate

We have generated adversarial examples for those whose original labels are benign, which could be used to fool the model and poison an online-learning model. In this case, the false positive rate would be largely increased. A CNN-based detection model is acting as our victim model. [20]

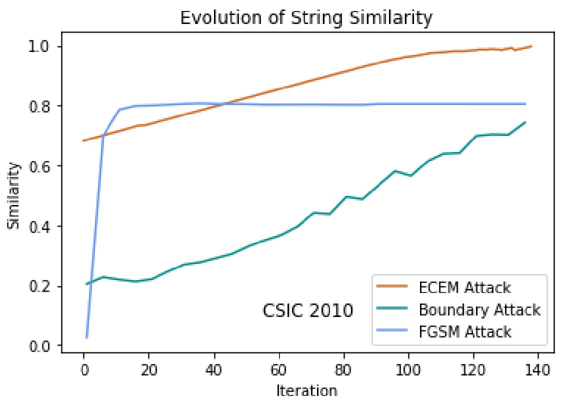


Fig. 5. The graph presents the evolution of average string similarity between adversarial and original example. The red line is for ECEM, which starts from a high value and approaches steadily to 1. The green line is for Boundary Attack which starts from a low value and also approaches to 1. The blue line is for FGSM which starts from a low value, increases surprisingly rapidly and converges approximately around 0.82.

The evolution of average string similarity between adversarial and original example is presented in Figure 5. We can observe from the graph that all three attacks start from a low value and increase with the iteration. Since ECEM chooses a nearest example in the "attack" set as the starting point, the string similarity begins with a higher value than FGSM and Boundary Attack. String similarity of ECEM converges to 1, which can be reached only when two strings are exactly the same, while FGSM is bounded by approximately 0.82. The advantage of FGSM is that within 20 iterations, the string similarity is able to achieve a surprisingly high value, but further effort is worthless to render it any larger. Boundary Attack also increases the string similarity step by step, but apparently the query number is much larger than ECEM to achieve the same quality of result.

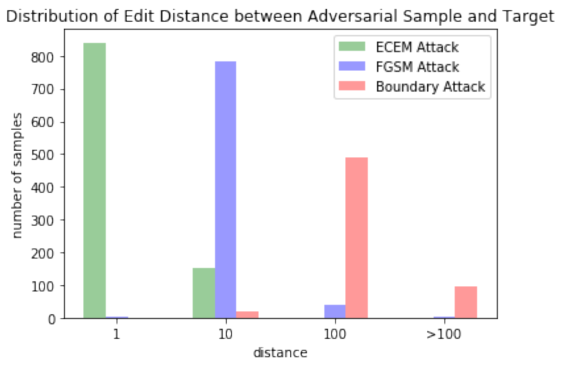


Fig. 6. The graph presents the distribution of Levenshtein Distance between the final adversarial and original example.

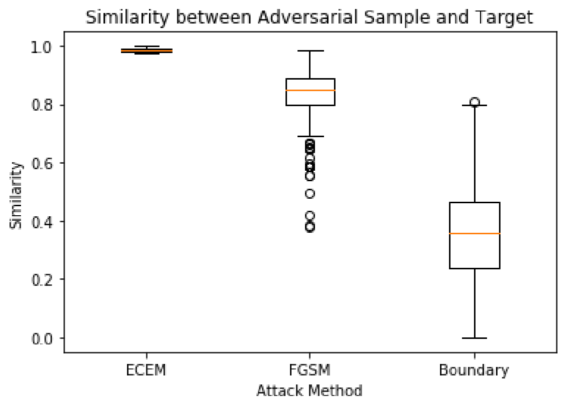


Fig. 7. The graph presents the distribution of string similarity between the final adversarial and original example.

Figure 6 further validates that ECEM outperforms FGSM and Boundary Attack. Here we presents the distribution of Levenshtein Distance between the best adversarial and original example – e.g., the number of iterations could be as much as the attack wants to generate the most similar adversarial example. Surprisingly, almost all the adversarial examples of ECEM have at most 10 characters different than their original examples, and a large part of them have only one character difference. FGSM can achieve a similar performance, but unfortunately, few adversarial examples could have only one character difference. Character difference is crucial for textual inputs since the less it is, the more possible the semantic meaning remains unchanged. Figure 7 shows that from the string similarity perspective, the variance of ECEM is much less than FGSM or Boundary Attack.

The prediction of a class is always related to the score of the last layer of the machine learning model. A classifier often needs to calculate the probability of the different classes based on the score. For a binary classifier for example, the prediction will be an "attack" label only if the probability is higher than 50%.

ECEM is a decision-based adversarial attack which has access only to the result of the classification. For the research, we are interested in how the probability of the "attack" label evolves during the iteration. To be clear, this probability is not used for the generation of adversarial example.

The evolution of average probability of "attack" label is presented in Figure 8. We can observe from the graph that all three attacks start from a large value and decrease with the iteration, because they all start from an example labelled "attack". Since the adversarial example is situated much closely to the hyper-plane of the prediction function, the probability of the "attack" label should be also close to 50%. Attack-label probability of ECEM indeed converges to 1, according to the graph, while FGSM is bounded by approximately 0.8. Again, the advantage of FGSM is that within 20 iterations, the probability is able to achieve a low value, but further effort is worthless to render it any lower. The deadly problem of Boundary Attack is that it always treats inputs as images and

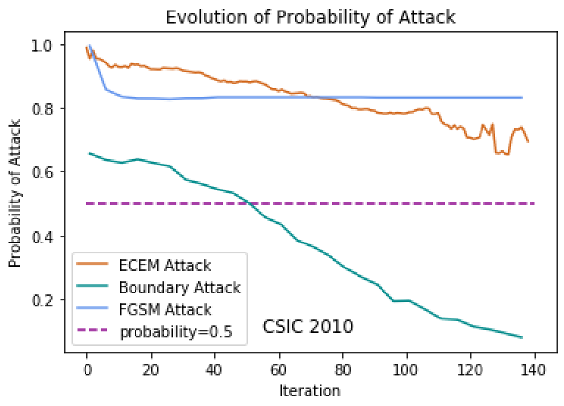


Fig. 8. The graph presents how the average probability of "attack" label evolves step by step. The red line is for ECEM which starts from a high value and goes down to 0.5. The green line is for Boundary Attack which starts from a low value and goes down to 0. The blue line is for FGSM which starts from a high value and converges approximately around 0.8

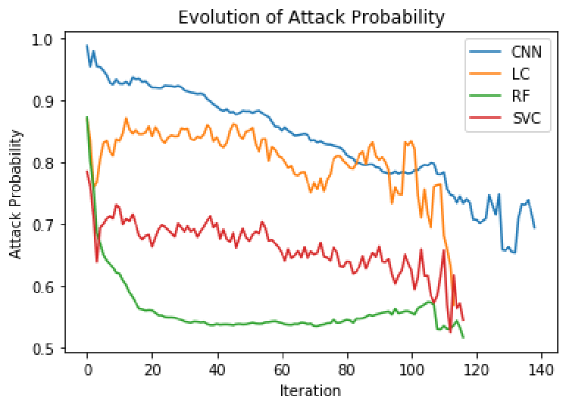


Fig. 9. The graph shows the evolution of the probability of "attack" label with models of CNN-based, logistic regression, random forest and linear support vector, attacked by ECEM. All line start form a high value and goes down to 0.5.

with further iteration, the example stays no longer adversarial.

Another important factor for an adversarial attack will be its success rate. As we mentioned before, the boundary attack is not capable of assuring the success of generating an adversarial sample for a text with the evolution of the iteration. Table I shows the success rates of ECEM and Boundary Attack launched on 1000 samples and 10000 samples respectively with 25, 50 and 75 iterations. The success rate of ECEM is always assured as 100%.

TABLE I
SUCCESS RATE

# iterations	1000 samples B-A[a]	1000 samples ECEM	10000 samples B-A	10000 samples ECEM
25	60.73%	100%	60.40%	100%
50	50.25%	100%	50.67%	100%
75	33.23%	100%	32.55%	100%

- [a]Boundary Attack

In this part, we have compared ECEM with FGSM (a white-box setting) and Boundary Attack (a black-box setting). The result of the experiment has shown following advantages:

- low query times requirement
- high similarity between original and adversarial examples
- 100% success rate
- no hyper-parameter tuning
- no knowledge of model architecture

C. More models

In order to further validate the performance of ECEM, we test it on three more machine learning models, namely, logistic regression, linear support vector machine and random forest. The models on CSIC achieve the following accuracy: LC 92.66%, Linear SVC 92.85% and RF 98.47%.

The similarity curve and the probability curve for four detection methods on HTTP data set CSIC 2010 are given in Figure 9 and 10.

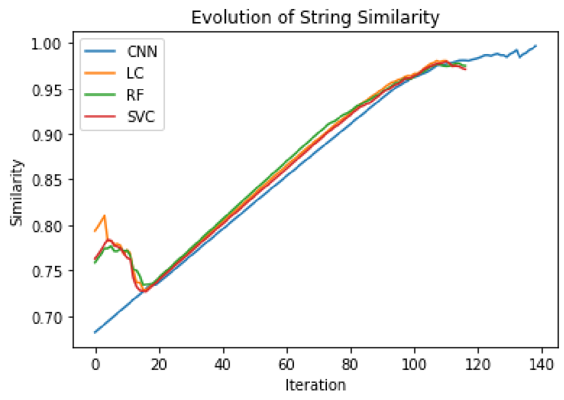


Fig. 10. The graph shows the evolution of the string similarity between adversarial and original examples with models of CNN-based, logistic regression, random forest and linear support vector, attacked by ECEM. All lines start from a high value and approach to 1.

We observe that our method consistently performs well under different machine learning detection models. The string similarity finally converges to 1, and the probability of "attack" label decreases until a value above 50%.

Although these models have a good performance on predicting attacks, LC and linear SVC have some tiny fluctuation on predicting attack probability. Besides, RF decreases the probability rapidly within several iterations. Hence, we hypothesize that the performance may be influenced by models' architectures.

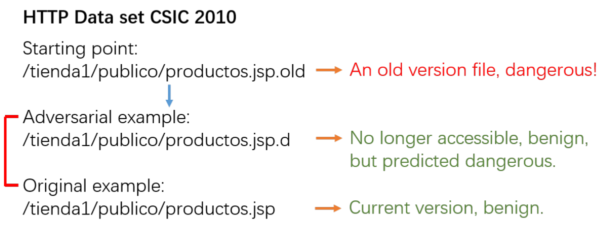
In this part, we have validated ECEM on different victim models. The result of the experiment has shown following advantages:

- universally applicable
- high similarity between original and adversarial examples

V. Real-world Applications

In many real-world machine learning applications, the attacker has no access to the architecture or the training data but can only observe the final decision. This is true for security systems (e.g. face identification), autonomous cars or speech recognition systems like Alexa or Cortana.

In this section we apply ECEM to two models respectively of two data sets. The first model classifies URLs into 2 classes, dangerous or benign. Dangerous URLs could be files in critical sections such as the old version or backup files. The second model could be used as an online comment filter which identifies toxic languages. Attention that the example for the second model might contains vulgar words.



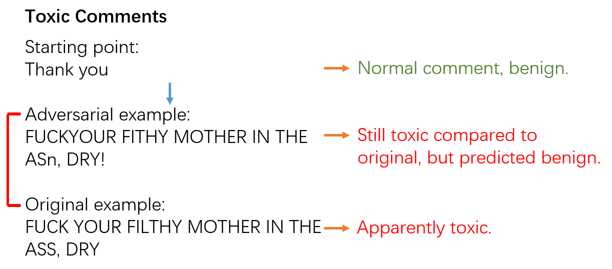
HTTP Data set CSIC 2010

Starting point:
/tienda1/publico/productos.jsp.old → An old version file, dangerous!

Adversarial example:
/tienda1/publico/productos.jsp.d → No longer accessible, benign, but predicted dangerous.

Original example:
/tienda1/publico/productos.jsp → Current version, benign.

Fig. 11. An adversarial example predicted as dangerous while in fact benign.



Toxic Comments

Starting point:
Thank you → Normal comment, benign.

Adversarial example:
FUCKYOUR FITHY MOTHER IN THE ASn, DRY! → Still toxic compared to original, but predicted benign.

Original example:
FUCK YOUR FILTHY MOTHER IN THE ASS, DRY → Apparently toxic.

Fig. 12. An adversarial example predicted as benign while in fact toxic.

Figure 11 shows an adversarial example which is predicted dangerous. However, since this adversarial URL can no longer have access to an old version file, it is in fact benign.

Figure 12 presents us a case in online comments toxicity detection. The iteration starts from a benign sentence "Thank you" and generates an adversarial example. Though the adversarial sentence has some spell mistakes, it should be considered as toxic. If this is an online comments filter, the consequence would be imaginable that the website fails to block vulgar and rude comments.

VI. Related Work

Recent research has shown interest in the deployment of machine learning techniques in web security problems. [21]–[23] Islam et al. [24] investigated the possibilities of modeling spammer behavioral patterns by Naive Bayesian classifier, Decision Tree Induction and Support Vector Machines. Calzavara et al. [25] proposed a detection method based on supervised learning for web authentication. Clincy et al. [26] leveraged Genetic Algorithm to generate attack signatures, building Intrusion Detection System for web services.

The existence of adversarial examples for neural networks (Szegedy et al. [1]; Biggio et al. [27]) was initially a theoretical concern. Recent research has demonstrated the applicability of adversarial examples in real world. Adversarial examples on a printed page remain adversarial when captured using a cell phone camera in an approximately axis-aligned setting. [28] The research of adversarial examples continues. Engstrom et al. argue that the existence of adversarial examples is tied to the unrobust features. [29]

There are numbers of image adversarial attacks. Methods for transforming ordinary two-dimensional images into adversarial examples, including techniques such as the L-BFGS attack (Szegedy et al. [1]), FGSM (Goodfel- low et al. [5]), and the CW attack (Carlini Wagner [4]), are well-known. Later, Brendel et al. [6] introduced the first decision-based adversarial attack, Boundary Attack, which requires no information about model architecture or weights. More research for black-box adversarial attack is proposed such as HopSkipJumpAttack (Chen et al. [2]) and Spatial Attack (Engstrom et al. [30]) Adversarial examples generated by these techniques could be also transferred into real world. [28] Athalye et al. [31] also shows the possibility to render adversarial examples robust to noise in real world.

Compared with images, adversarial attack on text is more challenging, composed of discrete words, hard to adopt gradient-based methods to perturb it. Various methods are proposed to tackle the challenges, such as back-translation (Iyyer et al. [32]), searching in underlying semantic space (Zhao et al. [33]), character flipping (Ebrahimi et al. [34]) and word substitution (Ribeiro et al. [35]; Alzantot et al. [36]; Ren et al. [37]). Zang et al. [38] proposed an attack model which comprises a sememe-based word substitution strategy and the particle swarm optimization algorithm, to tackle the existing problems.

One common observation from the development of security-related research is that attack and defense often come hand-in-hand, and one's improvement depends on the other's progress. Defenses against adversarial examples could be roughly categorized into three types: detection-based defense, gradient and representation masking, and adversarial training. Detection-based approaches aim to differentiate an adversarial example from a set of benign examples using statistical tests or out-of-sample analysis. Interested readers can refer to recent works in [39]–[44] Papernot et al. [45] proposed a defense distillation with gradient masking. Representation masking aims to replace the internal representations in DNNs with robust representations to alleviate adversarial attacks. For example, Bradshaw et al. [46] proposed to integrate DNNs with Gaussian processes and RBF kernels for enhanced robustness. Adversarial training, also known as the data augmentation method, makes DNNs less sensitive to perturbations to the examples by jointly training with adversarial examples. Interested readers can refer to the recent works in [47]–[51] and the references therein.

VII. Conclusion and Future Work

In this paper, we have proposed a black-box textual adversarial attack on machine learning models in web security. Importantly, it requires no knowledge of model architecture or weights, even the word or character embedding method. Efficient Character Edit Method is proved to be more competitive than FGSM and Boundary Attack and we actually believe that it outperforms most image adversarial attack.

We foresee many avenues for further research in the domain of machine learning techniques in web security. Adversarial examples could be used for studying properties of models. Furthermore, defense mechanism could be one promising research direction for the robustness and safety of models.

Based on our foundation, we can transfer this textual adversarial attack to numerous applications in network security: intrusion detection system, firewall and so on. In addition, real-world applications as long as inputs are textual now become vulnerable to our attack.

The largest open question is how to formally verify the general security of machine learning techniques. Machine learning techniques facilitate common life in all kinds of areas, but we should be careful about the deployment of machine learning models since they suffer vulnerabilities, including adversarial examples.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," dec 2013.

[2] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A Query-Efficient Decision-Based Attack," pp. 1–23, 2019.

[3] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb 1966.

[4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, May 2017.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–11, 2015.

[6] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–12, 2018.

[7] T. Kokkonen, "Anomaly-based online intrusion detection system as a sensor for cyber security situational awareness system," *Jyväskylä studies in computing*, no. 251, 2016.

[8] P. Y. Simard, "Using machine learning to break visual human interaction proofs (hips," in *Advances in Neural Information Processing Systems 17, Neural Information Processing Systems (NIPS'2004*, pp. 265–272, MIT Press, 2004.

[9] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[10] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, (New York, NY, USA), p. 506–519, Association for Computing Machinery, 2017.

[11] G. C. Amy Tennery, "Microsoft's ai twitter bot goes dark after racist, sexist tweets." https://www.reuters.com/article/us-microsoft-twitter-bot-idUSKCN0WQ2LA, 2016.

[12] J. Ratcliff and D. Metzener, "Ratcliff-obershelp pattern recognition." https://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html, 1998.

[13] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 601–618, USENIX Association, Aug. 2016.

[14] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, May 2017.

[15] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani, and D. Vitali, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," 2013.

[16] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," 2016.

[17] I. S. Institute, "Http dataset csic 2010." https://www.isi.csic.es/dataset/, Jan. 2012.

[18] Kaggle.com, "Malicious & non-malicious url." https://www.kaggle.com/antonyj453/urldataset, Nov. 2017.

[19] Kaggle.com, "Toxic comment classification challenge." https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview, Mar. 2018.

[20] X. Gong, H. Zhu, R. Deng, F. Wang, and J. Lu, "Crafting and detecting adversarial web requests," in *The 4th IEEE International Conference on Smart Cloud 2019, December 2019*, 2019.

[21] M. Babiker, E. Karaarslan, and Y. Hoscan, "Web application attack detection and forensics: A survey," *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, vol. 2018-Janua, pp. 1–6, 2018.

[22] B. W. Masduki, K. Ramli, F. A. Saputra, and D. Sugiarto, "Study on implementation of machine learning methods combination for improving attacks detection accuracy on intrusion detection system (ids)," in *2015 International Conference on Quality in Research (QiR)*, pp. 56–64, 2015.

[23] G. Tsudik and M. Almishari, "Machine-learning based security and privacy techniques for the modern web," 2012.

[24] M. S. Islam, A. A. Mahmud, and M. R. Islam, "Machine learning approaches for modeling spammer behavior," *asia information retrieval symposium*, vol. 6458, pp. 251–260, 2010.

[25] S. Calzavara, G. Tolomei, M. Bugliesi, and S. Orlando, "Quite a mess in my cookie jar!: leveraging machine learning to protect web authentication," in *Proceedings of the 23rd international conference on World wide web*, pp. 189–200, 2014.

[26] V. Clincy and H. Shahriar, "Web service injection attack detection," *2017 12th International Conference for Internet Technology and Secured Transactions, ICITST 2017*, no. Line 7, pp. 173–178, 2018.

[27] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," *Lecture Notes in Computer Science*, p. 387–402, 2013.

[28] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016.

[29] L. Engstrom, J. Gilmer, G. Goh, D. Hendrycks, A. Ilyas, A. Madry, R. Nakano, P. Nakkiran, S. Santurkar, B. Tran, D. Tsipras, and E. Wallace, "A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features'," *Distill*, vol. 4, no. 8, 2019.

[30] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," 2017.

[31] A. Athalye, L. Engstrom, A. Ilyas, and K. Kevin, "Synthesizing robust adversarial examples," *35th International Conference on Machine Learning, ICML 2018*, vol. 1, pp. 449–468, 2018.

[32] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 1875–1885, Association for Computational Linguistics, June 2018.

[33] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2017.

[34] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," 2017.

[35] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging NLP models," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 856–865, Association for Computational Linguistics, July 2018.

[36] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," 2018.

[37] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 1085–1097, Association for Computational Linguistics, July 2019.

[38] Y. Zang, C. Yang, F. Qi, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Textual adversarial attack as combinatorial optimization," 2019.

[39] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017.

[40] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," 2017.

[41] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," 2016.

[42] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017.

[43] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.

[44] W. Xu, D. Evans, and Y. Qi, "Feature squeezing mitigates and detects carlini/wagner adversarial examples," 2017.

[45] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," 2015.

[46] J. Bradshaw, A. G. de G. Matthews, and Z. Ghahramani, "Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks," 2017.

[47] J. Jin, A. Dundar, and E. Culurciello, "Robust convolutional neural networks under adversarial noise," 2015.

[48] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017.

[49] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," 2017.

[50] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," 2017.

[51] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," 2016.