URLTran: Improving Phishing URL Detection Using Transformers

Pranav Maneriker[⋆§^], Jack W. Stokes[†§], Edir Garcia Lazo[†], Diana Carutasu[†],
Farid Tajaddodianfar[‡^], Arun Gururajan[†]
[⋆]The Ohio State University, Columbus, OH 43210 USA
[†]Microsoft, Redmond, WA 98052 USA
[‡]Amazon, Seattle, WA 98109 USA

Abstract—**Browsers often include security features to detect phishing web pages. In the past, some browsers evaluated an unknown URL for inclusion in a list of known phishing pages. However, as the number of URLs and known phishing pages continued to increase at a rapid pace, browsers started to include one or more machine learning classifiers as part of their security services that aim to better protect end users from harm. While additional information could be used, browsers typically evaluate every unknown URL using some classifier in order to quickly detect these phishing pages. Early phishing detection used standard machine learning classifiers, but recent research has instead proposed the use of deep learning models for the phishing URL detection task. Concurrently, text embedding research using transformers has led to state-of-the-art results in many natural language processing tasks. In this work, we perform a comprehensive analysis of transformer models on the phishing URL detection task. We consider standard masked language modeling and additional domain-specific pre-training tasks, and compare these models to fine-tuned BERT and RoBERTa models. Combining the insights from these experiments, we propose URLTran which uses transformers to significantly improve the performance of phishing URL detection over a wide range of very low false positive rates (FPRs) compared to other deep learning-based methods. For example, URLTran yields a true positive rate (TPR) of 86.80% compared to 71.20% for the next best baseline at an FPR of 0.01%, resulting in a relative improvement of over 21.9%. Further, we consider some classical adversarial black-box phishing attacks such as those based on homoglyphs and compound word splits to improve the robustness of URLTran. We consider additional fine-tuning with these adversarial samples and demonstrate that URLTran can maintain low FPRs under these scenarios.**

Index Terms—**Phishing Detection, Neural Networks, BERT, Adversarial Robustness**

I. Introduction

Phishing occurs when a malicious web page is created to mimic the legitimate login page used to access a popular online service for the purpose of harvesting the user's credentials or a web page whose purpose is to input credit card or other payment information. Typical phishing targets include online banking services, web-based email portals, and social media web sites. Attackers use several different methods to direct the victim to the phishing site in order to launch the attack. In some cases, they may send the user a phishing email containing the URL (Uniform Resource Locator) of a phishing page. Attackers may also use search engine optimization techniques to rank phishing pages high in a search result query. Modern email platforms use various machine learning models to detect phishing web page attacks. In this work, we propose a new deep learning model that analyzes URLs and is based on transformers which have shown state-of-the-art performance in many important natural language processing tasks.

To prevent users from inadvertently uploading personal information to the attackers, web browsers provide additional security services which identify and block or warn a user against visiting a known phishing page. For example, Google's Chrome browser utilizes their Safe Browsing technology [1] and Microsoft's Edge browser includes Windows Defender SmartScreen [2]. In a related attack which is also addressed by these services, malicious URLs may point to a web page hosted by a misconfigured or unpatched server with the goal of exploiting browser vulnerabilities in order to infect the user's computer with malware (i.e., malicious software).

Successful phishing web page detection includes a number of significant challenges. First, there is a huge class imbalance associated with this problem. The number of phishing pages on the internet is very small compared to the total number of web pages available to users. Second, phishing campaigns are often short-lived. In order to avoid detection, attackers may move the login page from one site to another multiple times per day. Third, phishing attacks continue to be a persistent problem. The number of known phishing sites continues to increase over time. Therefore, blocking phishing attacks only using a continuously growing list of known phishing sites often fails to protect users in practice.

Popular web browsers may render hundreds of millions or even billions of web pages each day. In order to be effective, any phishing or malicious web page detection must be fast. For this reason, several researchers [3]–[5] have proposed detecting both phishing and malcious web pages based solely on analyzing the URL itself.

With the proliferation and ease of access to phishing kits sold on the black market as well as phishing as a service offerings, it has become easy for attackers with little expertise to deploy phishing sites and initiate such attacks. Consequently, phishing is currently on the rise and costing over $57 million

[§] The authors contributed equally to this work.

[^] This work was done while the author was employed at Microsoft.

from more than 114,000 victims in the US last year according to a recent FBI report [6]. The number of phishing attacks rose in Q3 of 2019 to a high level not seen since late 2016 [7]. As phishing is proving to be more and more fruitful, the attacks have become increasingly sophisticated. At the same time, the lifespan of phishing URLs has continued to drop dramatically – from 10+ hours to minutes [8].

Given the significant repercussions of visting a phishing or malicious web page, the detection of these URLs has been an active area of research [9]. In some cases, researchers have proposed the use of classic natural language processing methods to detect malicious URLs [3]. Other recent work has begun to use deep learning models to detect these URLs. URLNet [4] is a deep convolutional neural network (CNN) and includes separate character and word-level models for the malicious URL detection task. The Texception [5] model, which is used to detect phishing URLs, extends some of the ideas in URLNet by including small kernels which can be deployed in a wide variety of configurations in terms of width, depth or both.

Recently, semi-supervised machine learning methods have been used to create text embeddings that offer state-of-the-art results in many natural language processing tasks. The key idea in these approaches is the inclusion of a transformer model [10]. BERT [11], [12] utilizes transformers to offer significant improvements in several natural language processing (NLP) tasks. GPT [13], GPT-2 [14], and GPT-3 [15] have also followed a similar approach. The semantics and syntax of natural language are more complex than URLs, which must follow a strict syntax specification [16]. However, recent work using transformers has also demonstrated that these models can be applied to tasks involving data with more strict syntactic structures. These include tabular data [17], python source code [18] and SQL queries [19]. The success of these approaches further motivates us to apply transformers on URLs.

In this paper, we compare two settings: 1) we pre-train and fine-tune an existing transformer architecture using only URL data, and 2) we fine-tune publicly available pre-trained transformer models. In the first approach, we apply the commonly used Cloze-style masked language modeling objective [20] on the BERT architecture. In the second approach, we fine-tune BERT [11] and RoBERTa [21] on the URL classification task. Each of these systems forms an example of a URLTran model of which URLTran_BERT is the best. Finally, we simulate two common black-box phishing attacks by perturbing URLs in our data using unicode-based homoglyph substitutions [22] and inserting '-' characters between sub-words in a compound URL (e.g., 'bankofamerica.com' → 'bank-of-america.com'), along with a perturbation scenario under which the parameters are reordered and the URL label remains unchanged to improve the robustness of URLTran.

Results on a large corpus of phishing and benign URLs show that transformers are able to significantly outperform recent state-of-the-art phishing URL detection models (URLNet, Texception) over a wide range of low false positive rates where such a phishing URL detector must operate. At a false positive rate of 0.01%, URLTran increases the true positive rate from 71.20% for the next best baseline (URLNet) to 86.80% (21.9% relative increase). Thus, browser safety services, such Google's Safe Browsing and Microsoft's SmartScreen, may potentially benefit using the proposed URLTran system for the detection of phishing web pages.

This paper offers the following contributions:

- We propose the use of transformers to improve the detection of phishing URLs.
- We build URLTran, a large-scale system with production data and labels and demonstrate that transformers do offer a significant performance improvement compared to previous recent deep learning solutions over a wide range of very low false positive rates.
- We analyze the impact of various design choices in terms of hyperparameters, pre-training tasks, and tokenizers to contribute to an improved model.
- We analyze adversarially generated URLs from the system to understand the limitations of URLTran.

II. Phishing URL Data

The datasets used for training, validation and testing were collected from Microsoft's Edge and Internet Explorer production browsing telemetry during the summer of 2019. The schema for all three datasets is similar and consists of the browsing URL and a boolean determination of whether the URL has been identified as phishing or benign. Due to the highly unbalanced nature of the datasets (roughly 1 in 50 thousand URLs is a phishing URL), down-sampling of the benign set was necessary for both the training and validation sets. The resulting training dataset had the total size of 1,039,413 records with 77,870 phishing URLs and 961,543 benign URLs. Of the 259,854 URLs in the validation set, 19,468 corresponded to phishing sites and 240,386 to benign sites. The test set used for evaluating the models consists of 1,784,155 records, of which 8,742 are phishing URLs and the remaining 1,775,413 are benign. The labels included in this study correspond to those used to train production classifiers. Phishing URLs are manually confirmed by analysts including those which have been reported as suspicious by end user feedback. Other manually confirmed URLs are also labeled as phishing when they are included and manually verified in known phishing URL lists including Phishtank.

Benign URLs are those which correspond to web pages which are known to not be involved with a phishing attack. In this case, these sites have been verified by analysts using manual analysis. In other cases, benign URLs can be confirmed by thorough (i.e., production grade) off-line automated analysis which is not an option for real-time detection required by the browser. None of the benign URLs have been included in known phishing lists or have been reported as phishing pages by users and later verified by analysts. Although these last two criteria are not sufficient to add an unknown URL to the benign list, it is important to note that all URLs labeled as benign correspond to web pages that have been validated.

They are not simply a collection of unknown URLs, i.e., ones which have not been previously detected as phishing sites.

III. Methodology

URLTran seeks to use recent advances in natural language processing to improve the task of detecting phishing URLs by employing a two-pronged approach towards adapting transformers for the task of phishing URL detection. First, state-of-the-art transformer models, BERT [11] and RoBERTa [21], are fine-tuned, starting from publicly available vocabularies and weights and across different hyperparameter settings and resulting in URLTran_BERT and URLTran_RoBERTa, respectively. Second, domain-specific vocabularies are built using different tokenization approaches, and a domain specific transformer (URLTran_CustVoc) is first pre-trained and then fine-tuned on the task.

The general architecture of all the explored models takes a three stage approach for inference shown in Figure 1. It first uses a subword tokenizer to extract tokens from a URL. Next, a transformer model generates an embedding vector for the unknown URL. Finally, a classifier predicts a score indicating whether or not the unknown URL corresponds to a phishing web page.

In the following sections, we first provide briefly summarize the transformer model architecture, followed by the training tasks used to train the model, next with a description of the adversarial settings under which the best URLTran model is evaluated and then trained with adversarial examples to improve its robustness, and end with the threat model.

A. Architecture

We describe the tokenization schemes and overall architecture for classification in this section, skipping a detailed description of transformer models for brevity. Interested readers can review the transformer [10], BERT [11], or RoBERTa [21] papers for details of the internal structure of transformer layers.

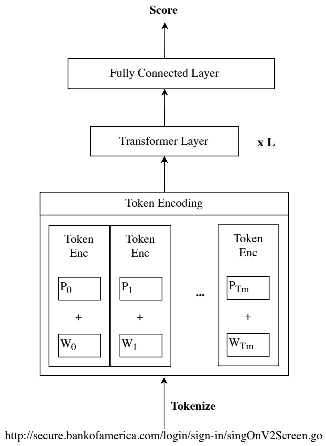


Fig. 1: URLTran phishing URL detection model.

URL (u_m)	secure.bankofamerica.com/login/sign-in/signOnV2Screen.go
Tokens ($\mathbf{TOK_m}$)	secure, ., bank, ##of, ##ame, ##rica, ., com, /, log, ##in, /, sign, -, in, /, sign, ##on, ##v, ##2, ##screen, ., go

TABLE I: Example of the (comma-separated) wordpiece token sequence extraction from a popular banking web page.

1) Tokenization: The raw input to the URLTran model is the URL, which can be viewed as a text sequence. The first step in the phishing URL detection task involves converting this input URL into a numerical vector which can be further processed by a classical machine learning or deep learning model.

Previous approaches have split URLs into sparse, binary features using important delimiters (e.g., ‘=’, ‘/’, ‘?’, ‘.’, ‘ ’) [3] and word/character-level CNNs of varying spans [4], [5]. Instead of these approaches, we experiment with multiple subword tokenization schemes in URLTran. While using full-length words reduces the input representation length (number of tokens) allowing more input to be processed by a fixed-length model, using a subword model can provide morphological insights to improve inference. For example, a full-length model would consider ‘bankofamerica’ and ‘bankofcanada’ as completely unrelated tokens, whereas a subword model can recognize the shared subword ‘bank’ to correlate URLs belonging to the two banks. Important character subsequences, including prefixes and suffixes, can also provide relevant information while being more robust to polymorphic attacks. Subword models attempt to find a balance of using both character subsequences and full words.

In particular, for URLTran_BERT and URLTran_RoBERTa, we use the existing word piece [11], [23] and Byte Pair Encoding (BPE) models [14], [21] , respectively. In addition to these, custom character-level and byte-level BPE vocabularies are created using the training URL data to have a domain specific vocabulary for URLTran_CustVoc with two different vocabulary sizes, 1K and 10K.

The BPE models first break the m^{th} URL, u_m, into a sequence of text tokens, $\mathbf{TOK}_m$, where the individual tokens may represent entire words or subwords. Following the notation in [11], the token sequence is formed as:

$$\mathbf{TOK}_m = \text{Tokenizer}(u_m) \tag{1}$$

where $\mathbf{TOK_m}$ is of length T_m positions and consists of individual tokens Tok_t at each position index t. For example, the BERT wordpiece token sequence generated from the URL of a popular banking login pageis shown in Table I. The wordpiece model includes special text tokens specified by (##) which build upon the previous token in the sequence. In the example in Table I, ‘##of’ means that it occurs after a previous token (‘bank’), and it is distinguished from the more common, separate token ‘of’.

2) Classifier: We use the transformer embeddings for two tasks: pre-training masked language models and fine-tuning for classification of phishing URLs. Both of these tasks require a final classification layer which can be applied to multiple tokens for masked token prediction and a pooled representation for classification. The transformer models that we train use a single, dense classification layer, which is applied to a special pooled token (`[CLS]`') for classification. For pre-training, a dense layer having `vocab_size` classes is used for predicting the masked token for the masked language modeling task. We use two-class classification for the fine-tuning model where the two classes are 1 for a phishing URL and 0 if the URL is benign. In both scenarios, the classification layer is:

$$\mathbf{s}_m = \mathbf{W}\mathbf{x}_m + \mathbf{b}. \tag{2}$$

In (2), $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector, respectively, for the final dense linear layer. $\mathbf{s_m}$ is the score which predicts if the URL $\mathbf{u_m}$ corresponds to a phishing web page when performing classification and is the sequence of masked token probability score vectors when performing masked language modeling for input token $\mathbf{x_m}$.

B. Training

1) Masked Language Modeling (MLM): The MLM task is commonly used to perform pre-training for transformers where a random subset of tokens is replaced by a special '`[MASK]`' token. The training objective for the task is the cross-entropy loss corresponding to prediction of the correct tokens at masked positions. The intuition for using this task for URLs is that specific query parameters and paths are generally associated with non-phishing URLs and therefore predicting masked tokens would help to uncover these associations. Similar intuitions derived from the cloze task [20] motivate the usage of MLMs for pre-training natural language models. Following the MLM hyperparameter settings for BERT, 15% of the tokens were uniformly selected for masking, of which 80% are replaced, 10% were left unchanged, and 10% were replaced by a random vocabulary token at each iteration. Dynamic masking [21] was used, i.e., different tokens masked from the same sequence across iterations. The training subset of the full dataset was used for pre-training to prevent any data leakage.

2) Fine-Tuning: The initial parameters for URLTran_BERT and URLTran_RoBERTa are derived using a large natural language corpus generated by their respective authors, were used. For URLTran_CustVoc, the final learned weights from the MLM pre-training step were used as initialization values. Next, URLTran's model parameters were further improved using a second "fine-tuning" training process which utilizes the error signal from the URL classification task and gradients based on gradient descent using the Adam [24] optimizer with the cross-entropy loss.

C. Adversarial Attacks and Data Augmentation

Phishing URL attacks can occur on short-lived domains and URLs which have small differences from existing, legitimate domains. We simulate two attack scenarios by constructing examples of such adversaries based on modifying benign URLs. Note that these generated domains do not actually exist in the pre-existing training and testing data, but are based upon frequently observed phishing attack patterns. We also utilize a reordering-based augmentation, which is used to generate benign perturbations for evaluating adversarial attacks.

1) Homoglyph Attack: We generate domains that appear nearly identical to legitimate URLs by substituting characters with other unicode characters that are similar in appearance. This attack strategy is commonly referred to as a *homoglyph attack* [25], [26], and we implement this strategy using the python library `homoglyphs`[1]. In particular, given a URL, we first extract the domain. For a randomly selected character in the domain, we check for one unicode (utf-8) Latin or Cyrillic character that is a homoglyph for it. We only perturb one character to minimize the probability that such a URL would we be identified as phishing by the user. We then replace the character by its homoglyph to construct a new URL.

2) Compound Attack: An alternative way to construct new phishing URLs is by splitting domains into sub-words (restricted to English) and then concatenating the sub-words with an intermediate hyphen. For example, 'bankofamerica.com' → 'bank-of-america.com'. To implement this, we leverage the `enchant` dictionary[2]. Consider a URL with domain d having $|d| = n$ characters. Let $\mathscr{D}$ denote the `enchant` English dictionary. Let $C(d, i, j)$ denote the function that returns True if $d[i \ldots j]$ can be split into one or more parts, each of which is a word in the dictionary $\mathscr{D}$. The compound word problem can be formulated recursively as

$$C(d,i,j) = \begin{cases} \text{True}, & d[i \ldots j] \in \mathscr{D} \\ \text{True} & \exists k, C(d,i,k) \text{ and } C(d,k+1,j) \\ \text{False} & \text{otherwise} \end{cases} \tag{3}$$

Using this recursive definition, we implement a dynamic programming algorithm that can compute whether a domain can be split and the corresponding splits. These splits are then concatenated with hyphens between the discovered words. Note that the base case check $d[i \ldots j] \in \mathscr{D}$ is performed in a case insensitive manner to ensure that the dictionary checks do not miss proper nouns.

3) Parameter Reordering: We extend text-aumentation approaches [27] for URL augmentation. As the query parameters of a URL are interpreted as a key-value dictionary, this augmentation incorporates permutation invariance. An example of a URL and permutation is provided in Figure 2. We use this approach to generate benign examples. Reordering the parameters still results in a valid URL, i.e., parameter reordering does not represent a phishing attack, and therefore we do not modify the URL's label.

4) Adversarial Attack Data: The approach we use for generating data for an adversarial attack includes generating separate augmented training, validation and test datasets based on their original dataset [28]. For each URL processed in these datasets, we generate a random number. If it is less than 0.5, we augment the URL, or otherwise, we include it

[1] https://pypi.org/project/homoglyphs/

[2] https://pypi.org/project/pyenchant/

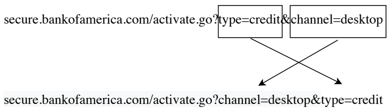

secure.bankofamerica.com/activate.go?type=credit&channel=desktop
secure.bankofamerica.com/activate.go?channel=desktop&type=credit

Fig. 2: An example of parameter reordering

in its original form. For URLs which are to be augmented, we modify it using either a homoglyph attack, a compound attack, or parameter reordering with equal probability. If a URL has been augmented, we also include the original URL in the augmented dataset.

D. Threat Model

The threat model for URLTran allows for the attacker to create any phishing URL including those which employ domain squatting techniques. In its current form, URLTran is protected against homoglyph and compound word attacks through dataset augmentation. However, any domain squatting attacks can also be simulated and included in the augmented adversarial training, validation, and test sets. In addition, a larger number of adversarial training examples can be directed at more popular domains such as https://www.bankofamerica.com that may be a target of attackers.

We assume that inference can be executed by the countermeasure system prior to the user visiting the unknown page. This can be done by the email system at scale by evaluating multiple URLs in parallel. In our evaluation, we found that URLTran requires 0.36096 milliseconds per URL on average which is a reasonable amount of latency.

IV. Numerical Evaluation

In this section, we evaluate and compare URLTran to several recently proposed baselines. We also report the model's training and inference times. Finally, we analyze the robustness of the model to generated phishing URLs.

Setup. We set the hyperparameters for previously published models according to the relevant published values. For evaluating URLTran_CustVoc, we vary the number of layers between $\{3, 6, 12\}$, number of tokens per input URL sequence between $\{128, 256\}$, and {byte, char}-level BPE tokenizer with $\{1K, 10K\}$-sized vocabularies. We randomly pick 15 hyperparameter combinations among these settings and present the results for these. The Adam optimizer [29] is used in both pre-training and fine-tuning, with the triangular scheduler [30] used for fine-tuning. All training and inference experiments were conducted using PyTorch [31] version 1.2 with NVIDIA Cuda 10.0 and Python 3.6. The experiments were performed by extending the Hugging Face and Fairseq PyTorch implementations found on GitHub [32], [33]. As the large class imbalance makes accuracy a poor metric of model performance, we evaluated all the models using the true positive rate (TPR) at low false positive rate (FPR) thresholds. We used the receiver operating characteristics (ROC) curve to compute this metric.

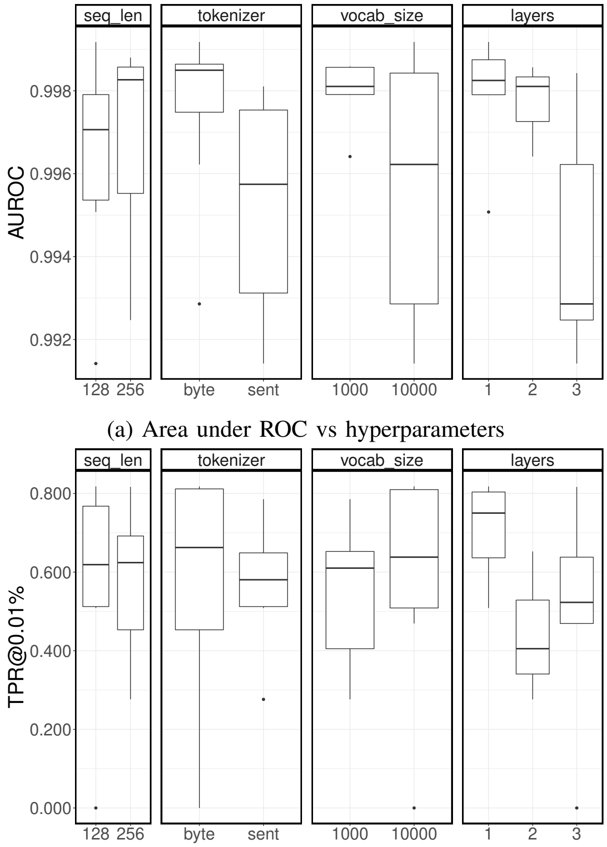
Baselines. To evaluate the performance of our models, we compared them to two baseline URL detection models: URLNet and Texception. URLNet [4] is a CNN-based model which was recently proposed for the task of detecting URLs associated with malicious web sites. In our baseline, we have completely trained and tested the URLNet model for the detection of phishing URLs. Texception [5] is another deep learning URL detection model which has been proposed for the task of identifying phishing URLs. As noted by Tajaddodianfar et al [5], Texception offered better performance than logistic regression, thus, we omit this comparison.

URLTran_CustVoc. Transformers typically require large amounts of pre-training data (e.g., BERT [11] used a corpus of $\approx$ 3.3 B tokens). However, this data is derived from text articles, which are structured differently from URLs. We also trained the URLTran_CustVoc model based soley on the URL data found in our datasets to compare the results of fine-tuning using standard BERT and RoBERTa pretrained models to models pretrained from the URL data. The difference in dataset size and data domain make it important to understand the impact of different hyperparameters used when training transformers from scratch. We compare runs across different hyperparameters on the basis of area under ROC (AUROC) and TPR@0.01% FPR. Figure 3 demonstrates that the training is not very sensitive to sequence length. Smaller byte-level vocabularies tend to be better overall, but at low FPR, the difference is not significant. Finally, we found that the three layer model generalized the best. We hypothesize that the better performance of the model with fewer layers is because of limited pre-training data and epochs. In the next few sections, we validate this hypothesis by evaluating models that have longer pre-training (URLTran_BERT, URLTran_RoBERTa) and tuned on a larger, adversarial dataset.

Model Performance. We next analyze the performance of the best parameters of all the proposed transformer variants. To understand how these models compare at *very low FPRs* where detection thresholds must be set to operate in a production environment, we first plot the ROC curves on a linear x-axis zoomed into a 2% maximum FPR in Figure 4. We also re-plot these ROC curves on a log x-axis in the semilog plot in Figure 5. These results indicate that all variants of URLTran offer a significantly better true positive rate over a wide range of extremely low FPRs. In particular, URLTran matches or exceeds the TPR of URLNet for the FPR range of 0.001% - 0.75%. The result is significant as phishing URL detection models must operate at very low FPRs (e.g., 0.01%) in order to minimize the number of times the security service predicts that a benign URL is a phishing site (i.e., a false positive). In practice, the browser manufacturer selects the desired FPR and tries to develop new models which can increase the TPR for the selected FPR value. Note that TPR@FPR is the **standard metric** commonly used both in production settings and in prior art such as Texception and URLNet. In addition to the ROC curve analysis, we also summarize a number of key performance metrics in Table II. In the table, 'F1' is the F1 score, and 'AUC' is the area under the model's ROC curve.

Model	Accuracy (%)	Precision (%)	Recall (%)	TPR@FPR=0.01%	F1	AUC
Texception	99.6594	99.7562	99.6594	52.1505	0.9969	0.9977
URLNet	99.4512	99.7157	99.4512	71.1965	0.9954	0.9988
URLTran_CustVoc	99.5983	99.7615	99.5983	81.8577	0.9965	0.9992
URLTran_RoBERTa	99.6384	99.7688	99.6384	82.0636	0.9968	0.9992
URLTran_BERT	99.6721	99.7845	99.6721	86.7994	0.9971	0.9993

TABLE II: Comparison of different performance metrics for URLTran and the two baseline models



(a) Area under ROC vs hyperparameters

(b) TPR@FPR = 0.01% vs hyperparmeters

Fig. 3: Variance in URLTran_CustVoc performance

The proposed URLTran model outperforms both Texception and URLNet for all of these metrics. In particular, we note that at an FPR of 0.01%, URLTran_BERT has a TPR of 86.80% compared to 71.20% for URLNet and 52.15% for Texception.

Training and Inference Times. The time required for training the best URLTran_BERT model was 4:57:11 and inference was 0:10:44 for an average of ≈ 0.361 ms per sample.

Adversarial Evaluation. To understand URLTran's robustness to adversarial attacks, we first compared the low FPR regions of the ROC curve of the unprotected model tested with the original test set to the test set which includes adversarial samples (AdvAttack) generated through the methods described in Section III-C (Figure 6). There is a significant drop in performance of URLTran_BERT when attacked with adversarial URLs. Next, we consider the scenario where attack strategies are incorporated into the training data (AdvTraining). On the

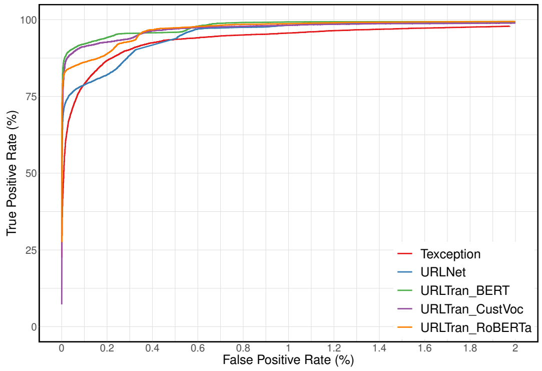


Fig. 4: Receiver operating characteristic curve indicating the performance of the URLTran and several baseline models zoomed into a maximum of 2% false positive rate.

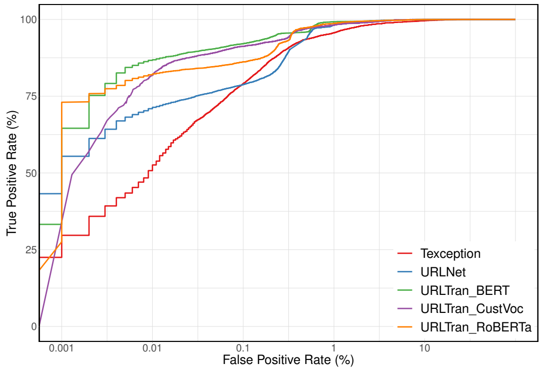


Fig. 5: Zoomed in receiver operating characteristic curve with a log x-axis.

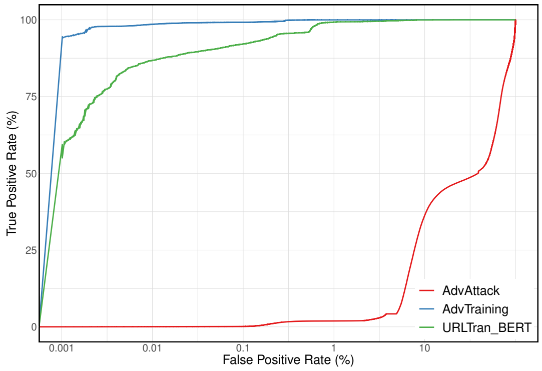


Fig. 6: ROC curve for URLTran_BERT when under adversarial attack, and adversarial robustness after augmented training

addition of adversarial attack patterns to the training, the model is able to the adapt to novel attacks, and even outperform the unprotected version of URLTran. These results demonstrate that URLTran can adapt to novel attacks. Further, as new attack strategies are recognized (e.g., homoglyph), a robust version of URLTran can be trained to recognize similar patterns in unseen test data.

V. Related Work

The URLTran system is most closely related to phishing and malicious URL detection models which have been previously proposed in the literature. In this section, we describe related work for deep learning-based text embeddings in general. We then review related work in phishing and malicious web page detection using its URL which builds upon models proposed in the NLP domain, in particular, URLNet and Texception, which helped to inspire this work.

Text Embeddings. Deep learning models for text embeddings have been an active area of research. One form of models called a character-level CNN learns a text embedding from individual characters, and these embeddings are then processed using a sequential CNN and one or more dense layers depending on the task. Recent examples of character-level CNNs include [34], [35]. In particular, Conneau et al. [34] investigated very deep architectures for the purpose of classifying natural language text. Typically, these models are trained in an end-to-end fashion instead of from manually engineered features. Transformers for text embedding were introduced by Vaswani et al. [10] in the context of neural machine translation. A number of models used transformers for other natural language processing tasks including BERT [11], [12], GPT [13], GPT-2 [14], and GPT-3 [15]. RoBERTa [36] used careful optimization of the BERT parameters and training methodology to offer further improvements.

Adversarial Attacks on Text. Adversarial example generation has been a focus of some recent work on understanding the robustness of various text classification tasks. The examples generated using these approaches aim to impose certain semantic constraints without modifying the label of the underlying text. White-box attacks (e.g., Hotflip [37]) require access to the internals of the classification model used, such as the gradient on specific examples. The attack framework proposed in our work is more in line with black-box attack frameworks such as DeepWordBug [26] and TextAttack [38] where the construction of adversarial data is motivated by a threat model but independent of the classifier used. We specialize this attack scheme to apply in the URL context.

URL-Based Phishing and Malicious Web Page Detection. We next review some recent systems for phishing and malicious web page detection using its URL in chronological order. Early phishing page detection based on URLs followed conventional deep learning approaches. A summary of these methods is included in [9]. Blum et al. [3] proposed using confidence weighted, online learning using a set of lexical features which are extracted from the URL. To extract these features, the URL is first split using the following delimiters: '?', '=', '/', '.', and ' '. Next, individual features are determined based on the path, domain, and protocol.

Le et al. [4] proposed the URLNet model whose task is to detect URLs which are references to malicious web pages found on the Internet. URLNet processes a URL using a character-level Convolutional Neural Network (CNN) and a word-level CNN. For the character-level CNN, the URL is first tokenized by each of the characters.

Inspired by the Xception deep object recognition model for images, Texception [5] also uses separate character-level and word-level CNNs like URLNet. However, Texception's CNN kernels form different size text windows in both the character and word levels. Multiple Texception blocks and Adaptive Max Pooling layers can be combined in different model configurations in terms of both depth and width. In addition, Texception utilizes contextual word embeddings in the form of either FastText or Word2Vec to convert the URL into the input embedding vector. Another CNN-based phishing detection model was proposed by Yerima and Alzaylaee [25]. Using the page's content, the authors create a 31-dimensional feature vector for each web page in their dataset and train a CNN based on this feature vector. URLTran differs from this work because it only processes the URL instead of extracting the page content which will be much slower for inference. Other work has proposed using LSTMs (i.e., recurrent sequential models) for phishing and malicious URL detection including [39], [40]. Processing LSTMs is expensive in terms of computation and memory for long URLs which makes them impractical for large-scale production. Huang et al. [41], also investigated capsule networks for detecting phishing URLs.

VI. Conclusion

We have proposed a new transformer-based system called URLTran whose goal is to predict the label of an unknown URL as either one which references a phishing or a benign web page. Transformers have demonstrated state-of-the-art performance in many natural language processing tasks, and this paper seeks to understand if these methods can also work well in the cybersecurity domain. In this work, we demonstrate that transformers which are fine-tuned using the standard BERT tasks also work remarkably well for the task of predicting phishing URLs. Instead of extracting lexical features or using CNN kernels that span multiple characters and words, which are both common in previously proposed URL detection models, our system uses the BPE tokenizers for this task. Next, transformers convert the token sequence to an embedding vector which can then be used as input to a standard, dense linear layer. Results indicate that URLTran is able to significantly outperform recent baselines, particularly over a wide range of very low false positive rates. We also demonstrate that transformers can be made robust to novel attacks under specific threat models when we adversarially augment the training data used for training them.

References

[1] Google, "Making the world's information safely accessible." [Online]. Available: https://safebrowsing.google.com/

[2] Microsoft, "Microsoft defender smartscreen." [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-smartscreen/microsoft-defender-smartscreen-overview

[3] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," *Proceedings of the Workshop on Artificial Intelligence and Security*, vol. 1, no. 1, pp. 1–37, 2010.

[4] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection," Tech. Rep., 2018. [Online]. Available: https://doi.org/10.475/123{_}4

[5] F. Tajaddodianfar, J. W. Stokes, and A. Gururajan, "Texception: A character/word-level deep learning model for phishing url detection," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2857–2861, 2020.

[6] F. B. of Investigation, "2019 internet crime report." [Online]. Available: https://pdf.ic3.gov/2019_IC3Report.pdf

[7] HelpNetSecurity, "Phishing attacks at highest level in three years." [Online]. Available: https://www.helpnetsecurity.com/2019/11/07/phishing-attacks-levels-rise/

[8] zvelo, "The rise of single-use phishing urls and the need for zero-second detection." [Online]. Available: https://zvelo.com/single-use-phishing-urls-need-zero-second-detection/

[9] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL Detection using Machine Learning: A Survey," vol. 1, no. 1, pp. 1–37, 2017. [Online]. Available: http://arxiv.org/abs/1701.07179

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[12] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in bertology: What we know about how bert works," *arXiv preprint arXiv:2002.12327*, 2020.

[13] R. Alec, N. Karthik, S. Tim, and S. Ilya, "Improving language understanding with unsupervised learning," Tech. Rep., Technical report, OpenAI, Tech. Rep., 2018.

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," Tech. Rep., Technical report, OpenAI, Tech. Rep., 2019.

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[16] T. Berners-Lee, R. T. Fielding, and L. M. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986, Jan. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc3986.txt

[17] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "TaBERT: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8413–8426. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.745

[18] A. Kanade, P. Maniatis, G. Balakrishnan, and K. Shi, "Learning and evaluating contextual embedding of source code," in *International Conference on Machine Learning*, 2020.

[19] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7567–7578. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.677

[20] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[22] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Detecting homoglyph attacks with a siamese neural network," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 22–28.

[23] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. S. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *ArXiv*, vol. abs/1609.08144, 2016.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] S. Y. Yerima and M. K. Alzaylaee, "High accuracy phishing detection based on convolutional neural networks," in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2020, pp. 1–6.

[26] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, May 2018, pp. 50–56.

[27] S. Kobayashi, "Contextual augmentation: Data augmentation by words with paradigmatic relations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 452–457.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," dec 2014. [Online]. Available: http://arxiv.org/abs/1412.6572

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[30] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 464–472.

[31] Facebook, "Pytorch - from research to production." [Online]. Available: https://pytorch.org/

[32] HuggingFace, "Transformers - state-of-the-art natural language processing for pytorch and tensorflow 2.0." [Online]. Available: https://github.com/huggingface/transformers

[33] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[34] A. Conneau, H. Schwenk, Y. Le Cun, and L. Barrault, "Very Deep Convolutional Networks for Text Classification," Tech. Rep., 2017. [Online]. Available: https://arxiv.org/pdf/1606.01781.pdf

[35] X. Zhang, J. Zhao, and Y. Lecun, "Character-level Convolutional Networks for Text," pp. 1–9, 2015.

[36] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[37] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 31–36.

[38] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 119–126.

[39] F. Ren, Z. Jiang, and J. Liu, "A bi-directional lstm model with attention for malicious url detection," in *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1, 2019, pp. 300–305.

[40] Y. Peng, S. Tian, L. Yu, Y. Lv, and R. Wang, "A joint approach to detect malicious url based on attention mechanism," *International Journal of Computational Intelligence and Applications*, vol. 18, no. 03, p. 1950021, 2019.

[41] Y. Huang, J. Qin, and W. Wen, "Phishing url detection via capsule-based neural network," in *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*. IEEE, 2019, pp. 22–26.