Adversarial Robustness of Phishing Email Detection Models

Parisa Mehdi Gholampour
pmehdigh@cougarnet.uh.edu
University of Houston
Houston, Texas, USA

Rakesh M. Verma
rmverma2@central.uh.edu
University of Houston
Houston, Texas, USA

ABSTRACT

Developing robust detection models against phishing emails has long been the main concern of the cyber defense community. Currently, public phishing/legitimate datasets lack adversarial email examples which keeps the detection models vulnerable. To address this problem, we developed an augmented phishing/legitimate email dataset, utilizing different adversarial text attack techniques. Next, the models were retrained with the adversarial dataset. Results showed that accuracy and F1 score of the models improved under subsequent attacks.

In another experiment, synthetic phishing emails were generated using a fine-tuned GPT-2 model. The detection model was retrained with a newly formed synthetic dataset. Subsequently, we observed that the accuracy and robustness of the model did not improve significantly under black box attack methods.

In the last experiment, we proposed a defensive technique to classify adversarial examples to their true labels using a K-Nearest Neighbor approach with 94% accuracy in our prediction.

CCS CONCEPTS

• **Security and privacy → Phishing; Social aspects of security**; • **Computing methodologies → Machine learning; Natural language processing; Natural language generation**.

KEYWORDS

phishing/legitimate dataset, adversarial attacks, data augmentation, model robustness, transformer models, deep learning, machine learning, generative AI, GPT-2

ACM Reference Format:
Parisa Mehdi Gholampour and Rakesh M. Verma. 2023. Adversarial Robustness of Phishing Email Detection Models. In *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics (IWSPA '23), April 26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3579987.3586567

1 INTRODUCTION

Phishing is defined as a social engineering cyber attack in which the attacker sends a deceptive or fake message to mislead the victim into revealing personal information or deploying malware [6, 35]. Emails are still one of the most popular vectors to launch phishing attacks. According to the recent APWG [1](Anti-Phishing Working Group), phishing activity trends report for the 2^{nd} quarter of 2022 that more than one million phishing attacks have been observed. The number of reported attacks on APWG has quadrupled since early 2020. Due to considerable damage to personal privacy and potential financial losses resulting from these attacks, designing more robust phishing detection models has been one of the main goals of the cyber security community [10, 36].

Creating a robust phishing email detector is challenging for a variety of reasons [6, 34] including the fact that class ratios are imbalanced in a realistic scenario. Email users are likely to receive fewer phishing attacks than legitimate emails. Existing datasets reflect this scenario as well.

In addition to this concern, is the active attacker, who is in a constant arms race with defenders. Popular deep learning models, including transformer-based models, are susceptible to adversarial attacks [38]. Proposing new attack techniques is currently one of the popular areas of Natural Language Processing research.

To address the above issues, in this study, we created two phishing/legitimate email datasets to address the above issues. The first contained adversarial examples and the second included synthetic ones. We showed that adversarial examples increased the robustness of detection models against possible perturbation in the model input. Additionally, synthetic examples balanced the original dataset, which in turn leads to more accurate detection models. However, the augmented models did not show improvement in the robustness and accuracy under these text attack methods.

We summarize our contributions as follows:

- We developed an adversarial phishing/legitimate dataset utilizing different attack techniques. The phishing detection models' robustness increased after retraining with the new dataset under the black box adversarial attacks.
- We augmented our phishing/legitimate dataset [2] by employing the OpenAI GPT-2 model [30] (a popular language model) to generate synthetic phishing emails. Detection models trained with the augmented dataset showed superior detection performance. Subsequently, we investigated the robustness of fine-tuned models with augmented datasets under text attack techniques and concluded that augmented dataset with synthetic phishing emails do not guarantee enhancement in the robustness of detection models. This shows the limitations of using language models like GPT-2 to protect against adversarial attacks.
- We proposed a defensive technique to recognize the true label of adversarial examples by measuring the cosine similarity

IWSPA '23, April 26, 2023, Charlotte, NC, USA

ACM ISBN 979-8-4007-0099-6/23/04…$15.00
https://doi.org/10.1145/3579987.3586567

[1] https://apwg.org
[2] The datasets are available at https://github.com/ReDASers/IWSPA-2023-Adversarial-Synthetic-Dataset.git

between their sentence embeddings and sentence embeddings of examples from the training set to find the K-nearest neighbors in their embedding space. By taking the majority vote on the neighbors' labels, we could classify adversarial examples correctly with 94% accuracy.

The rest of this paper is organized as follows: In Section 2, we review various adversarial attack techniques, phishing email generation approaches and dataset augmentation previous works in the cyber security domain. We then discuss the selected methodologies to generate adversarial and synthetic emails to augment our dataset in Section 3. In Section 4, the experiments are explained, and the results are discussed. Section 5 concludes the paper.

2 RELATED WORKS

In our literature review, initially we focus on adversarial and synthetic text generation. Later, we review data augmentation works in cyber security domain and explain the motivation for creating our datasets containing adversarial attack examples and synthetic emails.

2.1 Adversarial Attack Techniques in Text

Since the rise of transformer models, researchers have studied their robustness by proposing a variety of attack techniques at character, word, sentence levels or a combination of these in white- and black-box attack scenarios. In a *white-box* attack setting, it is assumed that attacker has complete knowledge of the model, the training dataset, and the model hyper-parameters. On the other hand, in a *black-box* attack setting, the only available information from the model is the confidence score and the prediction on the input data. The latter attack is closer to realistic attack scenarios, and therefore we mainly focus on reviewing the black-box attacks.

One of the strong baselines of character level attacks named DeepWordBug was proposed in [12]. A combination of swap, substitution, deletion and insertion of a random letter on the most important words in the sequence that had an effective role in the model prediction output were applied to create adversarial examples. Li et al. introduced TextBugger [21] for both white-box and black-box settings. In their black-box approach, they first composed the text sequence in sentences and selected the most important one in changing the output probability score. Later, they ranked the influential words and performed the perturbation both at character and word levels. Textfooler, another strong baseline was presented in [17]. Synonym replacement by considering cosine similarity between the original word and the possible candidates was performed. Additionally Universal Sentence Encoder was applied to measure the similarity between the original and new sentence and must remain above a specific threshold. In [32], a greedy algorithm named probability weighted word saliency (PWWS) was applied for generating adversarial text. Word replacement in input text with synonyms was done using WordNet. Other works including [13, 20, 22] leveraged Masked Language Models to predict the top word perturbation candidates for word replacement, insertion, or a combination of both.

2.2 Phishing Email Generation

Generating synthetic phishing emails and URLs to improve the performance of current phishing detection models has been the focus of many papers in recent years. In [2, 3], the authors use semi-automatic techniques for masquerade attack and scam augmentation. Das et al. [7] leveraged Recurrent Neural Network to automatically generate phishing emails and evaluated them on Naive Bayes, Logistic regression and SVM models. In the follow up work [8], the authors performed a quantitative evaluation by using perplexity, coherence, and human evaluation to examine the quality of the generated emails. In another paper [24] a phishing email generator system was developed by employing a graph database. Their approach consisted of five steps: Email fragmentation, inserting semantic fragments in graph database, selecting data segments with the highest weights, combining and generating a new email and adding the generated email into the database.

Over the last few years, GANs and Transformers have become powerful tools for researchers to generate text with higher quality. In phishing domain, the work is divided into two categories: generating URLs and email's body. Karman et al. [18] proposed a GAN network for real-time phishing URL detection. They employed game theoretic perspective to design two games between attacker and defender in their training setting. Another paper [1] focused on using GANs to create adversarial phishing URLs to bypass phishing detectors. GPT-2 models (Transformer based text generator model) have become popular models to generate Fake cyber threats including generating phishing emails in [31] and [9].

2.3 Data Augmentation in Cybersecurity

Datasets in cyber security studies are mostly imbalanced because the number of attacks is generally smaller than the number of normal events, or because the attacks are not made publicly available. To overcome this problem, dataset augmentation has been employed in different cyber security domains such as intrusion detection [23, 28], malware detection [27], insider attack detection [14, 15, 40], cyber threat detection [4] and phishing emails detection [29, 33].

To enhance an intrusion detection system, the authors in [28], developed a method named Sort-Augment-Combine (SAC) to augment their network traffic dataset, named Bot-IoT [19]. Moreover, in a recent work [23] for improving the performance of an intrusion detection system, a data balancing scheme based on conditional variational autoencoder was proposed to augment the minority classes of their network traffic dataset.

To develop more robust anomaly behavior detection models, in [14], a Conditional GAN was used to augment the extracted malicious behavior features from the user access log files to detect malicious insider activities. Additionally, the authors of [40] utilized an LSTM-autoencoder to encode anomalous user behavior to a continuous hidden space and then used this representation as an input to a GAN model to generate synthetic representations. Furthermore, a combination of transfer learning, data augmentation, and few-shot learning was applied in [4] to develop a model to detect cyber threat detection and intelligence model. The dataset was created by collecting tweets related to the Microsoft Exchange Server data breach of 2021 and augmented with the GPT-3 language

model. Moreover, Nazari et al. in their work [27] utilized Conditional Generative Anomaly Detection (CGAN) to generate synthetic data to balance out a wide range of cybersecurity datasets such as benign and malicious DoH (DNS over HTTPS) traffic, TOR [3] traffic, android malware/benign software and malicious/benign URLs (Uniform Resource Locator).

The focus of the cyber security research community for developing more accurate models has been on just augmenting the minority class to balance out the dataset. One of the missing points in their augmentation process was not considering possible adversarial examples that can enhance the model's robustness. In our work, we focused on developing a phishing/legitimate dataset that included these examples and filled the current gap in improving the quality of the dataset.

3 DATASET ENHANCEMENT APPROACH

We begin with the framework and the specific attack methods for generating adversarial phishing emails. Then, we discuss synthetic email generation.

3.1 Adversarial Phishing Email Generation

In natural language processing, deep neural networks are vulnerable to adversarial attack examples. They are defined as inputs that contain small perturbations and can fool the model to produce the wrong prediction. Here, in the text domain, we defined f as a trained classification model $f(f : X \Rightarrow Y)$ that maps input samples to correct labels, given the input samples $X = \{x_1, ..., x_n\}$ and corresponding labels, $Y = \{y_1, ..., y_n\}$. An adversarial example x' with $y' = f(x')$ was generated by performing minor perturbations on x with $y = f(x)$ in a way that $y \neq y'$.

To generate adversarial examples for our dataset, we used the TextAttack framework [25] and its github[4] repository. TextAttack includes 17 well known text attack techniques in white-box and black-box settings. To build our adversarial dataset, we examined the compatibility of available attack techniques with our phishing detection models and dataset and reached several conclusions. Some of the attack techniques, such as Alzantot, Fast-alzantot, Iga, and PSO, used genetic algorithms and particle swarm optimization to find attack examples; hence, they were computationally too expensive for our purposes. Additionally, a few attack methods such as Hotflip and Kuleshov were white-box attack scenarios, and they were not in the scope of our work. Moreover, some attack methods were not compatible with our imported fined-tuned transformer Huggingface[5] phishing detection models such as Bert-Attack and CLARE.

Finally, we selected Textfooler, PWWS and BAE as word-level attack techniques and DeepWordBug as a character-level attack technique to develop our dataset. These attack methods work well with our imported phishing detection models and dataset.

3.1.1 Textfooler. Textfooler [17] is one of the most popular text attack technique baselines. The first step in the attack algorithm was sorting the words in the sequence by their importance score. The score was calculated by removing each word in the sentence and calculating the prediction score before and after the removal process and finding the difference. In the second step, the original word candidate was replaced by its synonym following a set of rules. The new word should be semantically similar, fit in contextually in the surrounding text, and change the prediction of the model. Cosine similarity between the original word and the possible candidates was used to select the top N candidates using [26] vector space. After replacing the synonym in the text, Universal Sentence Encoder [5] was used to encode original and new sentences and the cosine similarity of them was measured. This value must be above a certain threshold ϵ for a word to be selected in the final candidate pool

3.1.2 PWWS. Probability weighted word saliency, PWWS, is a greedy algorithm that applies word replacement with synonyms using WordNet[6] with a unique scoring mechanism [32]. For every word in the selected text, all the word's synonyms were extracted first. Then a synonym was chosen as a replacement candidate that caused the greatest change in classification probability. Furthermore, each word's saliency factor was determined. The saliency factor is the change in the output probability if one of the words in the document is unknown. By applying this measure, the word with the most significant effect on the output probability was discovered. A score function was generated by multiplying the change in output probability upon replacement with a synonym word and the saliency factor. Words were sorted after applying the score function to every word in the document. Finally, the algorithm greedily iterated through all the words' synonym candidates until it found a word that made a change to the classification result and proceeded to replace it.

3.1.3 BAE. BAE Uses BERT's Masked language model to build attack examples [13]. By masking the words in the text sequence, replacement or insertion candidates were collected. This attack technique was proposed in four different modes. (1) BAE-R mode, contained only the replacement of the original word. (2) BAE-I mode, inserted a token to the left or right of the candidate word. (3) BAE-R/I mode, performed either replacement or insertion of a token to generate an adversarial example. (4) BAE-R+I, applied both replacement and insertion of the word to create an adversarial example, which was reported as the best approach to create more successful attacks compared to the other proposed attack methods.

3.1.4 DeepWordBug. In this baseline for character-level adversarial examples, four different approaches to score and find the important tokens in the input text [12] were proposed. (1) Replace score, for each token was calculated by subtracting the model prediction output for the sequence with the token and the sequence in which the token was replaced by the unknown token. (2) Temporal Head Score for i^{th} token was computed as the difference between model prediction for the sequence before the i^{th} token and including the token. (3) Temporal Tail Score was the difference between model prediction for the sequence from i^{th} token to the end and the remaining sequence after the i^{th} token. (4) Combined score, which was the combination of Temporal Head and Temporal Tail scores. After sorting tokens by descending scores, swap, substitution, deletion, and insertion of a random letter for each

[3] https://www.torproject.org
[4] https://github.com/QData/TextAttack
[5] https://huggingface.co
[6] https://wordnet.princeton.edu

token were performed until the output prediction changed from the target label. The perturbation process was constrained by defining a threshold of the Levenshtein distance[7] of the original sequence and the perturbed one.

3.2 Synthetic Phishing Email Generation

In this work, we fine-tuned a GPT-2 model with our phishing email from the original dataset to generate synthetic phishing emails. GPT-2 [30] as a transformer language model was pre-trained on a large corpus of web-text data. Therefore, it is a great candidate to fine-tune with our phishing-email dataset for synthetic phishing emails generation. Moreover, it uses an embedding dimensional state of 768 and a context size of 1024 that specifies the maximum number of tokens it can have for positional encoding. While training, the targeted output was considered as an input sequence that was shifted one token to the right. During this process, GPT-2 tries to predict the token i knowing the inputs from 1 to $i-1$ by employing masking, to learn the inner representation of the whole input data.

After fine-tuning the GPT-2 model, a decoding method must be applied to generate text such as Greedy, Beam, Top-K and Top-P methods to find the sequences with the high probability in the language model.

- Greedy search selects a word with highest probability as its next word to generate the word sequence.
- Beam search keeps the sequences of most probable words and selects the sequence that has the highest probability at the end.
- Top-K sampling selects the K highest probable next words and samples the next word in the word sequence from this word selection [11].
- Top-P sampling chooses the set of words that together have the probability of P and samples the next word in the word sequence from this word selection [16].

4 EXPERIMENTS AND RESULTS

We now present the experiments starting with the base dataset and its preprocessing followed by the models and their performance in several settings and conclude the section with the results of the defensive technique.

4.1 Dataset and Preprocessing

In this work, we used the public IWSPA 2.0 with no header phishing/legitimate email imbalanced dataset, collected by Verma et al. [37]. It contains 2 classes of emails, 629 phishing and 5092 legitimate emails. Since our main purpose was to augment the text body of the available emails, therefore we carefully preproccessed the dataset to remove additional web links, extra space, extra tags, attachments, phone numbers, URLs, HTML, and CSS scripts presented in the emails using Python's Regular Expression Package. Furthermore, during the preprocessing, we noticed that some of the emails contained non-English characters. Therefore, they were removed from the dataset using langdetect[8] Python module since non-English emails were an anomaly in the dataset with mainly English emails. Finally, we developed our experiments with 4606 legitimate and 568 phishing emails. Table 1 shows the average and standard deviation of the number of words in the emails for each class.

Table 1: Dataset Text Length Statistics

Email Type	Train set		Test set	
	Mean	STD	Mean	STD
Phishing	71.52	70.57	69.86	70.48
Legitimate	231.44	493.90	232.86	456.55

4.2 Phishing Detection Models

We fine-tuned a variety of transformer models from the Huggingface hub using the 'Trainer' function from the transformers python module on the email classification task as our phishing detection models with 80%, 10%, and 10% of data for training, validating, and testing. We used 'TrainingArguments('test_trainer')' function to load training hyper-parameters arguments for all the transformer models. In this function, the number of epochs was 3, the batch size was 8, and the learning rate was 5e-05. Additionally, the loss function was Cross Entropy Loss. We defined phishing emails as the positive class and legitimate emails as the negative class in our experiments.

Environment. We ran our experiments on our university cluster that hosts a total of 5704 CPU cores in 169 compute and 12 GPU nodes. CPU type in this cluster is 'Intel Xeon E5-2680v4' and GPU type is 'Nvidia P100'. In our experiments, we allocated one GPU node, one compute node, one processor core with 32GB memory to run our jobs.

Metrics. We use Accuracy, F1 score metrics to evaluate phishing detection models' performance and robustness that are presented in the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Four distinct types of text attack techniques (Textfooler, PWWS, DeepWordBug, and BAE) from the TextAttack framework were applied to the testing set and fed to the detection models to measure their robustness against these attacks. The attack results on models are shown in Tables 2, 3, 4 and 5 respectively (DBERT and SQ represent DISTILBERT and SQUEEZBERT in these tables, respectively). Additionally, the word average perturbation across all the attack methods in the TextAttack framework was calculated using the following Equation, where N is the number of successful attacks. Average Word Perturbation (AWP) is given by:

$$AWP\ \% = \frac{1}{N} \sum \frac{number\ of\ perturbed\ words\ in\ text}{number\ of\ words\ in\ text} \times 100$$

[7] https://en.wikipedia.org/wiki/Levenshtein_distance
[8] https://pypi.org/project/langdetect/

Table 2: Textfooler attack results on test set. UA stands for Under Attack, AWP stands for Average Word Perturbation in successful attack examples and ANQ stands for Average Number of Queries.

Model	Acc	F1	UA Acc	UA F1	AWP%	ANQ
ALBERT	0.99	0.95	**0.78**	0.10	20.68	2101.99
ROBERTA	0.99	0.96	0.71	0.13	31.6	1997.65
BERT	0.99	0.97	0.64	0.09	29	2023
DEBERTA	0.99	0.97	0.64	0.07	23.94	1895.49
DBERT	0.99	0.97	0.70	0.10	19.77	1961.28
SQ	0.98	0.93	0.54	0.02	27.8	1889.64
YOSO	0.98	0.92	0.36	0.01	26	1622.65

Table 3: PWWS attack results on test set. UA stands for Under Attack, AWP stands for Average Word Perturbation in successful attack examples and ANQ stands for Average Number of Queries.

Model	Acc	F1	UA Acc	UA F1	AWP%	ANQ
ALBERT	0.99	0.95	**0.86**	0.12	13.26	1642.35
ROBERTA	0.99	0.96	0.84	0.31	17.84	1648.55
BERT	0.99	0.97	0.84	0.25	20.26	1610.68
DEBERTA	0.99	0.97	0.80	0.12	18.05	1592.08
DBERT	0.99	0.97	0.77	0.14	15.36	1597.02
SQ	0.98	0.93	0.75	0.10	18.62	1594.61
YOSO	0.98	0.92	0.73	0.10	21.28	1573.77

ANQ denotes the average number of queries made by the attack method to the model on successful attacks.

As shown in the tables, Textfooler and BAE had higher average word perturbation percentages compared to DeepWordBug and PWWS across all the phishing detection models.

In the attack experiments, the ALBERT model showed higher accuracy among all the models in most attack scenarios. Furthermore, we observed that the F1 score for all attacked models was very low. Although, the DEBERTA model was considered one of the most robust models under different attack scenarios [39], it did not perform well in our experiments since the accuracy and F1 score of the model dropped significantly under text attack methods. Additionally, Table 5 displays that BAE was not very successful in attacking phishing detection models compared to the rest of the attacking methods; we suspect that this could be due to its tighter constraints (e.g., syntax preservation) in the default setting to generate attack examples.

Table 6 presents the average running time in minutes for each attack method across all the models. From this table we observed that Textfooler attack execution takes more time and is computationally expensive compared to the rest of the attack methods. Moreover, BAE algorithm has the least execution time and computational cost.

Table 4: DeepWordBug attack results on test set. UA stands for Under Attack, AWP stands for Average Word Perturbation in successful attack examples and ANQ stands for Average Number of Queries.

Model	Acc	F1	UA Acc	UA F1	AWP%	ANQ
ALBERT	0.99	0.95	0.87	0.13	9.01	384.89
ROBERTA	0.99	0.96	**0.89**	0.33	13.97	440.27
BERT	0.99	0.97	0.85	0.19	20.24	445.99
DEBERTA	0.99	0.97	0.86	0.14	13.7	520.01
DBERT	0.99	0.97	0.87	0.12	11.52	451.3
SQ	0.98	0.93	0.87	0.17	18.92	400.74
YOSO	0.98	0.92	0.86	0.10	17.1	333.8

Table 5: BAE attack results on test set. UA stands for Under Attack, AWP stands for Average Word Perturbation in successful attack examples and ANQ stands for Average Number of Queries.

Model	Acc	F1	UA Acc	UA F1	AWP%	ANQ
ALBERT	0.99	0.95	**0.95**	0.80	19.18	418.62
ROBERTA	0.99	0.96	0.91	0.70	23.53	498.78
BERT	0.99	0.97	0.95	0.76	34.58	462.26
DEBERTA	0.99	0.97	0.94	0.77	24.33	735.96
DBERT	0.99	0.97	0.95	0.80	11.52	451.3
SQ	0.98	0.93	0.93	0.65	42.51	427.46
YOSO	0.98	0.92	0.88	0.61	27.08	378.99

Table 6: Attack methods' average running time in minutes across all models

	Textfooler	PWWS	DeepWordBug	BAE
Time	743	654	487	384

4.3 Characteristics of Successful Attacks

In our second set of experiments, we derive insights on what types of original emails result in successful attacks. For this, we collected the adversarial examples of the four indicated text attack techniques on 2500 data samples (2000 train examples and 500 test samples) of original train and test separately on ALBERT model.

As shown in Table 7 PWWS and DeepWordBug attack techniques are more successful to perturb phishing examples compared to legitimate emails. However, Textfooler does not differentiate between phishing and legitimate emails to create adversarial samples. Additionally, BAE generated fewer attack examples compare to other attack techniques because of tighter constrains in the default version of the attack. Furthermore, by employing the Textstat[9] python module, we examined the statistical analysis of successful and failed attack examples. Table 8 demonstrates that attacks are successful on examples with fewer words, sentences and difficult

[9]https://pypi.org/project/textstat/

Table 7: Phishing and Legitimate Successful Attack examples on data set

Text Attack	Train set		**Test set**	
	Phishing	legitimate	Phishing	legitimate
Textfooler	256	259	52	60
PWWS	254	118	52	18
DeepWordBug	251	58	52	14
BAE	41	53	7	20

Table 8: Statistical Analysis of Textfooler Attack Result

Statistical Analysis	Succeed Examples		Failed Examples	
	Mean	STD	Mean	STD
Lexicon Count	59.1	42.25	194.82	412.4
Sentence Count	1.25	0.43	2.51	3.06
Mono Syllabus Count	40.75	31.94	132.33	259.16
Poly Syllabus Count	6	5.5	26.76	80.38
Difficult Words Count	10.95	7.35	32.75	59.227

words and proves that shorter examples in our dataset are more susceptible to become successful attack examples.

Table 9 shows the intersection among test set attack examples for different attack methods. According to the table Textfooler, PWWS and DeepwordBug methods shared the most successful attack examples. Specifically, they shared more phishing successful examples compared to legitimate emails. In contrast, BAE shared more legitimate successful attack examples with Textfooler. Overall, all the attack methods shared 11 successful attack examples with seven phishing and four legitimate emails in the testing set. In our experiment, BAE did not have any unique phishing emails that only it was successful on. However, an analysis shows that it was the only one successful on seven legitimate emails. PWWS had only that unique legitimate email and TextFooler and PWWS were successful on the same exact set of phishing emails, which was surprising and requires more analysis/experiments to confirm. DeepWordBug had one unique phishing email that only it was successful on.

After initial observation of generated adversarial examples, we created two different datasets. The first one, included the combination of the original training dataset and adversarial examples and, the second one, contained the original training dataset and half of adversarial examples. In the second dataset, the adversarial examples were selected randomly. Later, these two datasets were used for retraining the ALBERT model. Moreover, there were no changes to the original testing set. Then, these models were attacked with Textfooler, PWWS, and DeepWordBug to observe their robustness. Table 10 reports the results of the experiment. Both models that were retrained with training dataset and adversarial examples showed a considerable increase in their accuracy and F1 score. The model that was retrained with half of the adversarial examples displayed a higher F1 score when it was under attack by Textfooler, PWWS, and DeepWordBug. From this experiment we learned, some of examples could contribute more to the robustness of the model in the retraining process. On contrary, not all the generated adversarial examples can increase the robustness of the model compared to a subset of them. Additionally, we saw just a slight improvement in models under BAE attack.

In another experiment, we created two more new datasets. In the first one, we combined the original dataset and the adversarial examples that were collected from train, valid, and test set. Then we generated a new train, valid, and test set with 80%, 10%, and 10% of data, respectively. In the second dataset, we used half of the adversarial examples and the original dataset. After fine-tuning ALBERT model with new datasets, we measured the accuracy and F1 score of the model under four text attack methods. Table 11 displays the results of this experiment. With the new settings, the accuracy of the model did not improve compared to the accuracy of the model that just used the original dataset without adversarial examples. But the F1 score did show some improvement with the new dataset combinations. After analyzing the confusion matrices, we concluded that the model was more robust in detecting phishing emails under text attack methods but failed in classifying legitimate emails, correctly.

4.4 Data Augmentation Results

To generate the synthetic data, we fine-tuned a GPT-2 language model using only the phishing emails to provide enough phishing emails to overcome the imbalanced issue. AdamW optimizer with learning rate 5e-4 was used during the fine tuning process. To our knowledge AdamW optimizer yields better training loss and helps model to converge faster and has been used widely in fine-tuning GPT-2 models. We set up a scheduler with $get-linear-schedule-with-warmup$ function. The learning rate started from zero in warm up steps and increases linearly to the regular learning rate. Moreover, the loss for the GPT-2 model can be obtained from the model output itself. The first instance of output, output[0], is Cross Entropy Loss and it's been used in training process as the loss value. We chose batch size 2 for our experiment because of GPU memory limitations for batch sizes more than 2. Finally, to have a better generation model with limited number of phishing email training data point, we performed training and validation for 8 epochs and stop training before model over fitting happens (the validation loss rises).

To generate examples, we chose the first three words of each training example as a seed and fed it to the generative model. Three words was the minimum number of words, found empirically, using which we could generate more realistic phishing emails compared to the phishing emails generated with one and two seed words. Then we used three different decoding techniques including Beam, Top-K, and Top-P respectively to decode the generated embedding. In total, 5448 phishing emails were generated.

To confirm that the augmented samples were phishing emails, first, we only kept the data that had BERT Score[10] 0.85 and above with the reference data. Later, we utilized the fine-tuned ALBERT phishing detection model on the original dataset to predict the labels of augmented phishing data. Figure 1 and Figure 2 show 2D

[10]https://github.com/Tiiiger/bert_score, this is an automatic evaluation score that tries to measure the quality of text generation. It uses similarity in the contextual embedding space of words rather than syntactically matching the words.

Table 9: Intersection among ALBERT test set attack examples for different attack methods

Text Attack Method	Textfooler		PWWS		DeepWordBug		BAE	
	Phish	Legit	Phish	Legit	Phish	Legit	Phish	Legit
Textfooler ∩ PWWS	52/52	17/60	52/52	17/18				
Textfooler ∩ DeepWordBug	51/52	13/60			51/52	13/14		
Textfooler ∩ BAE	7/52	13/60					7/7	13/20
PWWS ∩ DeepWordBug			51/52	8/18	51/52	8/14		
PWWS ∩ BAE			7/52	6/18			7/7	6/20
DeepWordBug ∩ BAE					7/52	5/14	7/7	5/20
Textfooler ∩ PWWS ∩ DeepWordBug	51/52	5/60	51/52	5/18	51/52	5/14		
Textfooler ∩ PWWS ∩ BAE	7/52	6/60	7/52	6/18			7/7	6/20
Textfooler ∩ DeepWordBug ∩ BAE	7/52	5/60			7/52	5/14	7/7	5/20
PWWS ∩ DeepWordBug ∩ BAE			7/52	4/18	7/52	4/14	7/7	4/20
Textfooler ∩ PWWS ∩ DeepWordBug ∩ BAE	7/52	4/60	7/52	4/18	7/52	4/14	7/7	4/20

Table 10: ALBERT model performance under different attack methods after retraining with adversarial datasets. UA stands for Under Attack. Bold numbers show the model's under attack accuracy and F1 score improvement compared to original model without retraining with adversarial data

Attack Method	Train Dataset	Original Acc	Original F1	UA Acc	UA F1
Textfooler	Original Train Set	0.99	0.95	0.78	0.10
	Original Train Set + 1/2 Adversarial	0.99	0.95	**0.85**	**0.28**
	Original Train Set + Adversarial	0.99	0.96	**0.88**	**0.14**
PWWS	Original Train Set	0.99	0.95	0.86	0.12
	Original Train Set + 1/2 Adversarial	0.99	0.95	**0.89**	**0.39**
	Original Train Set + Adversarial	0.99	0.96	**0.89**	**0.22**
DeepWordBug	Original Train Set	0.99	0.95	0.87	0.13
	Original Train Set + 1/2 Adversarial	0.99	0.95	**0.91**	**0.48**
	Original Train Set + Adversarial	0.99	0.96	**0.88**	**0.16**
BAE	Original Train Set	0.99	0.95	0.95	0.80
	Original Train Set + 1/2 Adversarial	0.99	0.95	**0.96**	**0.86**
	Original Train Set + Adversarial	0.99	0.96	**0.96**	**0.86**

Table 11: ALBERT model performance under different attack methods after retraining with new dataset combinations. UA stands for Under Attack. Bold numbers show the model's under attack F1 score improvement compared to original model without using adversarial data in forming the dataset

Attack Method	Dataset	Original Acc	Original F1	UA Acc	UA F1
Textfooler	Original Dataset + 1/2 Adversarial	0.97	0.94	0.65	**0.24**
	Original Dataset + Adversarial	0.98	0.96	0.75	**0.30**
PWWS	Original Dataset + 1/2 Adversarial	0.97	0.94	0.79	**0.43**
	Original Dataset + Adversarial	0.98	0.96	0.79	**0.39**
DeepWordBug	Original Dataset + 1/2 Adversarial	0.97	0.94	0.79	**0.40**
	Original Dataset + Adversarial	0.98	0.96	0.77	**0.32**
BAE	Original Dataset + 1/2 Adversarial	0.97	0.94	0.88	0.73
	Original Dataset + Adversarial	0.98	0.96	0.90	0.78

sentence embedding representation of original and augmented dataset employing t-SNE function. Orange dots present phishing emails embedding and the blue ones display legitimate emails. In figure 2, phishing emails embedding cover more area in the embedding space of the dataset.

Since the original ratio of phishing to legitimate emails is 1 to 8, We sampled the augmented phishing data and derived three datasets with 1 to 4, 1 to 2 and 1 to 1 ratios. Next, the ALBERT model was fine-tuned 3 times with the newly derived datasets to ensure the consistency of the performance results. The results of

this experiment are reflected in Table 12. As we increased the number of phishing emails in the datasets, the models mostly displayed better accuracy, F1, precision, recall, and Matthew's Correlation Coefficient (MCC) scores. This shows that data augmentation did help improve model performance.

Figure 1: 2D sentence embedding representation of original dataset employing t-SNE function

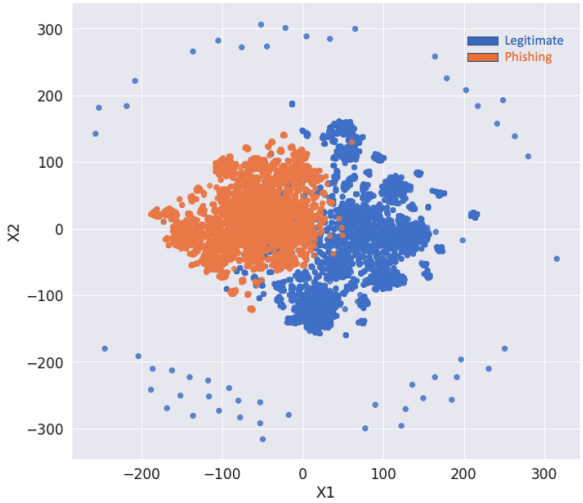

Figure 2: 2D sentence embedding representation of augmented dataset employing t-SNE function

4.4.1 Analysis of Models Robustness Trained with Augmented Data. After fine-tuning the phishing detection models with augmented data with different ratios, we attacked them with Textfooler, PWWS, DeepWordBug, and BAE attack techniques to investigate their robustness against the attacks.

Table 12: ALBERT Performance with Data Augmentation

Data Ratio	Acc	F1	Precision	Recall	MCC
1/8	0.988	0.94	0.98	0.91	0.93
1/4	0.990	0.95	0.94	0.96	0.95
1/2	0.991	0.95	0.95	0.96	0.95
1/1	0.994	0.97	0.98	0.96	0.97

Table 13: ALBERT model performance with data augmentation under different attack methods. UA stands for Under Attack. Bold numbers show the highest model's under attack accuracy trained with different data ratios

Attack Method	Data Ratio	UA Acc	UA F1
Textfooler	1/8	0.78	0.10
	1/4	0.68	0.11
	1/2	**0.83**	0.08
	1/1	0.73	0.06
PWWS	1/8	0.86	0.12
	1/4	0.83	0.25
	1/2	**0.87**	0.13
	1/1	0.85	0.13
DeepWordBug	1/8	**0.87**	0.13
	1/4	0.80	0.20
	1/2	**0.87**	0.11
	1/1	0.86	0.14
BAE	1/8	**0.95**	0.80
	1/4	0.92	0.74
	1/2	0.94	0.78
	1/1	0.89	0.64

Table 13 presents the result of augmented models' performance under Textfooler, PWWS, DeepWordBug and BAE attack methods. It is obvious that the model under Textfooler that was trained by 1/2 data ratio reached better accuracy but with lower F1 score compared to the base model. Moreover, models trained with 1/4 and 1/1 data ratios did not have better accuracy compared to the base model. This result shows that selected examples in 1/2 data ratio had higher quality in making model more robust compared to 1/4 and 1/1 data ratios. Additionally, we concluded that the phishing detection models with different ratios of augmented phishing data with our current dataset did not display any improvement in model robustness under the Textfooler attack.

Furthermore, the model under PWWS attack accuracy, trained with 1/2 phishing to legitimate data ratio increased compared to 1/8 and 1/4 data ratios. In addition, the model trained with 1/4 data ratio had a better F1 score compared to the other data ratio cases. However, we learned that these models did not show considerable improvement against PWWS attack method.

Moreover, models trained with 1/2 and 1/1 data ratio of phishing to legitimate data showed similar under DeepWordBug attack accuracy to initial model trained with 1/8 data ratio. Nonetheless, models trained with 1/4 and 1/1 had an increase in their F1 scores.

However, we learned that these models did not display a considerable improvement against the DeepWordBug attack method as well.

Lastly, models trained with 1/2, 1/4, and 1/1 data ratios of phishing to legitimate data showed less under BAE attack accuracy and F1 score compared to the initial model trained with 1/8 data ratio. In general, we learned that the augmented models did not display any improvement against the BAE attack method. After careful observation, we concluded that not just adding more synthetic data to the training data can improve the models' robustness against different attack methods. However, some of the sampled synthetic data have better quality to improve the robustness of phishing detection models.

4.5 Defensive Technique Results

In this section, we present the proposed defensive technique to recognize the true label for adversarial examples before they were fed to phishing detection model. First adversarial examples were collected by feeding the test set to the ALBERT phishing detection model under four attack methods, Textfooler, PWWS, DeepWordBug, and BAE. Later, we created a dataframe from these adversarial examples and their true label from the original test set. Moreover, we encoded these examples and original training examples using Universal Sentence Encoder to create sentence embeddings. For every adversarial input, we found the K-Nearest Neighbors (KNN) in the original training dataset by measuring the cosine similarity between their sentence embedding and then taking the majority vote of their labels. K was varied from 1, 3, 5, 7, and 9. At last, we calculated how many adversarial inputs can be correctly classified in this way and measured the defensive technique's performance. Tables 14 and 15 show the performance and the confusion matrix of different neighbor sizes of the KNN algorithm in this experiment. The overall results showed that KNNs with different sizes of neighbors classified adversarial examples in their true class with accuracies above 90 percent. The best accuracy (94%) was achieved with K = 5 and 7. Therefore, we concluded our defensive method to assign the correct label to adversarial examples was effective and the KNN classifier performed as a defensive shield for the vulnerable transformer phishing detection model under text attack methods.

In conclusion, our defensive mechanism works as a shield against adversarial examples for vulnerable deep learning phishing detection models. The KNN shield model can classify the adversarial examples to their true class labels before they reach the deep learning model. Furthermore, we can use the shield's decision in an ensemble manner in the final decision of the deep learning detection model.

Table 14: KNN performance for classifying adversarial examples to their true label with different neighbor sizes (K). Bold numbers show the highest accuracy and F1 score

K	Acc	F1
1	0.90	0.91
3	0.93	0.94
5	**0.94**	**0.95**
7	**0.94**	**0.95**
9	0.93	0.94

Table 15: KNN classifier confusion matrix for classifying the adversarial examples to their true label with different neighbor sizes (K)

K	Acc
1	$\begin{bmatrix} 100 & 12 \\ 14 & 149 \end{bmatrix}$
3	$\begin{bmatrix} 98 & 14 \\ 5 & 158 \end{bmatrix}$
5	$\begin{bmatrix} 99 & 13 \\ 3 & 160 \end{bmatrix}$
7	$\begin{bmatrix} 101 & 11 \\ 4 & 159 \end{bmatrix}$
9	$\begin{bmatrix} 100 & 12 \\ 6 & 157 \end{bmatrix}$

5 CONCLUSIONS AND FUTURE WORK

In this work, we developed an adversarial phishing/legitimate dataset utilizing different attack techniques. The phishing detection models' robustness increased after retraining with the new dataset under the black box adversarial attacks. Furthermore, we complemented our phishing/legitimate dataset by employing GPT-2 model to generate synthetic phishing emails. The detection models trained with the augmented dataset showed great accuracy. We also investigated the robustness of fine-tuned models with augmented datasets under text attack techniques and concluded that augmenting with synthetic GPT-2 phishing emails, did not guarantee increased robustness of the detection models. Lastly, we proposed a defensive technique to recognize the true label of adversarial examples. By measuring the cosine similarity between the sentence embedding of adversarial attacks and training set sentence embeddings, we found the K-Nearest Neighbors in their embedding space and took the majority vote on the neighbors' labels. We could classify adversarial examples correctly with 94% accuracy by applying this method.

There were some interesting patterns among the four attack methods studied and among the emails on which the attack methods were successful. However, they require more experiments to confirm and are left for future work. Additionally, comparing our defensive method to other techniques is left for the future.

6 ACKNOWLEDGMENTS

This research is based upon work supported by National Science Foundation grant DGE 1433817. We thank the reviewers for their constructive comments. Verma's research was also partially supported by NSF grant 2210198, ARO grant W911NF-20-1-0254, and ONR contract N000142112270. He is the founder of Everest Cyber Security and Analytics, Inc.

REFERENCES

[1] Trinh Nguyen Bac, Phan The Duy, and Van-Hau Pham. 2021. PWDGAN: Generating Adversarial Malicious URL Examples for Deceiving Black-Box Phishing Website Detector using GANs. In *2021 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*. IEEE, 1–4.

[2] Shahryar Baki, Rakesh Verma, Arjun Mukherjee, and Omprakash Gnawali. 2017. Scaling and effectiveness of email masquerade attacks: Exploiting natural language generation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 469–482.

[3] Shahryar Baki, Rakesh M Verma, and Omprakash Gnawali. 2020. Scam Augmentation and Customization: Identifying Vulnerable Users and Arming Defenders. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 236–247.

[4] Markus Bayer, Tobias Frey, and Christian Reuter. 2022. Multi-Level Fine-Tuning, Data Augmentation, and Few-Shot Learning for Specialized Cyber Threat Intelligence. *arXiv preprint arXiv:2207.11076* (2022).

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[6] Avisha Das, Shahryar Baki, Ayman El Aassal, Rakesh Verma, and Arthur Dunbar. 2019. SoK: a comprehensive reexamination of phishing research from the security perspective. *IEEE Communications Surveys & Tutorials* 22, 1 (2019), 671–708.

[7] Avisha Das and Rakesh Verma. 2019. Automated email generation for targeted attacks using natural language. *arXiv preprint arXiv:1908.06893* (2019).

[8] Avisha Das and Rakesh M Verma. 2020. Modeling Coherency in Generated Emails by Leveraging Deep Neural Learners. *arXiv preprint arXiv:2007.07403* (2020).

[9] Kayla Duskin, Shivam Sharma, Ji Young Yun, Emily Saldanha, and Dustin Arendt. 2021. Evaluating and Explaining Natural Language Generation with GenX. In *Proceedings of the Second Workshop on Data Science with Human in the Loop: Language Advances*. 70–78.

[10] Ayman El Aassal, Shahryar Baki, Avisha Das, and Rakesh M Verma. 2020. An in-depth benchmarking and evaluation of phishing detection research for security needs. *IEEE Access* 8 (2020), 22170–22192.

[11] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833* (2018).

[12] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 50–56.

[13] Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970* (2020).

[14] RG Gayathri, Atul Sajjanhar, Yong Xiang, and Xingjun Ma. 2021. Anomaly detection for scenario-based insider activities using cgan augmented data. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 718–725.

[15] Joshua Glasser and Brian Lindauer. 2013. Bridging the gap: A pragmatic approach to generating insider threat data. In *2013 IEEE Security and Privacy Workshops*. 98–104.

[16] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* (2019).

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

[18] Sharif Amit Kamran, Shamik Sengupta, and Alireza Tavakkoli. 2021. Semi-supervised Conditional GAN for Simultaneous Generation and Detection of Phishing URLs: A Game theoretic Perspective. *arXiv preprint arXiv:2108.01852* (2021).

[19] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems* (2019), 779–796.

[20] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502* (2020).

[21] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271* (2018).

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

[23] Chang Liu, Ruslan Antypenko, Iryna Sushko, and Oksana Zakharchenko. 2022. Intrusion Detection System After Data Augmentation Schemes Based on the VAE and CVAE. *IEEE Transactions on Reliability* 71 (2022), 1000–1010.

[24] Nasim Maleki and Ali A Ghorbani. 2019. Generating phishing emails using graph database. In *International Conference on Information Security Practice and Experience*. Springer, 434–449.

[25] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. 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*. 119–126.

[26] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892* (2016).

[27] Ehsan Nazari, Paula Branco, and Guy-Vincent Jourdan. 2021. Using CGAN to Deal with Class Imbalance and Small Sample Size in Cybersecurity Problems. In *2021 18th International Conference on Privacy, Security and Trust (PST)*. 1–10.

[28] Uneneibotejit Otokwala, Andrei Petrovski, and Harsha Kalutarage. 2021. Improving Intrusion Detection Through Training Data Augmentation. In *2021 14th International Conference on Security of Information and Networks (SIN)*, Vol. 1. 1–8.

[29] Fatima Zahra Qachfar, Rakesh M Verma, and Arjun Mukherjee. 2022. Leveraging Synthetic Data and PU Learning For Phishing Email Detection. In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy*. 29–40.

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[31] Priyanka Ranade, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tim Finin. 2021. Generating fake cyber threat intelligence using transformer-based models. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–9.

[32] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 1085–1097.

[33] Lee Joon Sern, Yam Gui Peng David, and Chan Jin Hao. 2020. PhishGAN: Data Augmentation and Identification of Homoglyph Attacks. In *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. 1–6.

[34] Rakesh Verma, Murat Kantarcioglu, David Marchette, Ernst Leiss, and Thamar Solorio. 2015. Security analytics: essential data analytics knowledge for cybersecurity professionals and students. *IEEE Security & Privacy* 13, 6 (2015), 60–65.

[35] Rakesh M Verma and David J Marchette. 2019. *Cybersecurity analytics*. CRC Press.

[36] Rakesh M Verma and David J Marchette. 2021. How to catch a phish with statistics. *Significance* 18, 5 (2021), 26–29.

[37] Rakesh M Verma, Victor Zeng, and Houtan Faridi. 2019. Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2605–2607.

[38] Yevgeniy Vorobeychik and Murat Kantarcioglu. 2018. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12, 3 (2018), 1–169.

[39] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840* (2021).

[40] Fangfang Yuan, Yanmin Shang, Yanbing Liu, Yanan Cao, and Jianlong Tan. 2020. Data augmentation for insider threat detection with GAN. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. 632–638.