Directed Adversarial Sampling Attacks on Phishing Detection

Hossein Shirazi[1], Bruhadeshwar Bezawada[2],
Indrakshi Ray[1], and Chuck Anderson[1]

[1] Colorado State University, Fort Collins CO 80523, USA
[2] Mahindra Ècole Centrale, Hyderabad, Telangana, India

Abstract. Phishing websites trick honest users into believing that they interact with a legitimate website and capture sensitive information, such as user names, passwords, credit card numbers, and other personal information. Machine learning is a promising technique to distinguish between phishing and legitimate websites. However, machine learning approaches are susceptible to *adversarial learning* attacks where a phishing sample can bypass classifiers. Our experiments on publicly available datasets reveal that the phishing detection mechanisms are vulnerable to adversarial learning attacks. We investigate the robustness of machine learning-based phishing detection in the face of adversarial learning attacks.

We propose a practical approach to simulate such attacks by generating adversarial samples through direct feature manipulation. To enhance the sample's success probability, we describe a clustering approach that guides an attacker to select the best possible phishing samples that can bypass the classifier by appearing as legitimate samples. We define the notion of *vulnerability level* for each dataset that measures the number of features that can be manipulated and the cost for such manipulation. Further, we clustered phishing samples and showed that some clusters of samples are more likely to exhibit higher vulnerability levels than others. This helps an adversary identify the best candidates of phishing samples to generate adversarial samples at a lower cost. Our finding can be used to refine the dataset and develop better learning models to compensate for the weak samples in the training dataset.

Keywords: Phishing, Machine Learning, Adversarial Sampling, Classifiers

1 Introduction

1.1 Motivation

Phishing, as defined in [1], is an attempt to obtain sensitive information such as user-names, passwords, and credit card details by masquerading as a trustworthy entity in an electronic communication. The first recorded mention of the term is found in the hacking tool against American Online (AOL) users in

1995 named *AOHell*. The technique was elaborated in a presentation by Felix *et al.* as early as 1987 [2]. Phishing attacks have shown remarkable resilience against a multitude of defensive efforts, and attackers continue to generate sophisticated phishing websites that closely mimic legitimate websites. There were 328,000 unique attacks reported in 2007, and this number almost quadrupled by 2017 [3].

When viewed as a social-engineering attack, phishing cannot be solved solely by educating the end-users, and hence, automatic detection techniques are essential. Several defenses were proposed against phishing attacks, such as URL blacklisting, keyword-based filtering, IP address filtering, and machine learning-based techniques. Solutions like URL-blacklisting are no longer effective as attackers can evade such techniques through simple URL manipulation or by hosting websites on popular free hosting services on the Internet. Machine learning-based techniques appear to be a promising direction.

1.2 Problem Statement

Phishing websites are the ones that mimic legitimate websites to defraud honest Internet users and steal personal information. Machine learning-based techniques are effective in detecting patterns among different types of websites, *i.e.* phishing and legitimate. However, phishing and legitimate websites should be represented as a set of features for use in machine learning algorithms. A feature is a measurable property of a characteristic of a website. Researchers define a set of features and measure feature values for each given website. Features could be defined at certain levels *i.e.* contextual characteristics of the websites or URLs of the websites. A labeled phishing dataset is comprised of a set of instances of phishing and legitimate websites where each instance is represented by its feature values and has a label that indicates whether it is phishing or legitimate. A classification algorithm trains on a part of labeled data to make predictions about the label of the other parts which have not been used for training.

Most works emphasize feature definition and aim to improve the statistical learning models to discriminate between phishing and legitimate websites. The state-of-the-art solutions for phishing detection [4–8] use engineered features based on observations made by the research experts in this domain on publicly available datasets. One crucial assumption in using machine learning approaches is that the training data collection process is independent of the attackers' actions [9]. However, in adversarial contexts, *e.g.* phishing or spam filtering, this is far from the reality as attackers either generate noisy data samples or generate new attack samples by manipulating features of existing phishing instances. Furthermore, manipulating features results in a dangerous scenario wherein, an attacker can bypass the generated classifier without much effort. A carefully crafted phishing data sample that appears to a machine learning classifier as a legitimate sample is called an *adversarial sample*. The immediate impact of adversarial samples is to degrade the accuracy of a machine learning classifier. A key problem for the attacker to consider would be choosing the features that need to be manipulated and the associated cost for such manipulation. Ideally,

the attacker would like to bypass the classifier with the lowest cost of manipulating the data sample features. In this work, we explore and study the effect of adversarial sampling on phishing detection algorithms in-depth, starting with some simple feature manipulation strategies, and show some surprising results that demonstrate impact on the classification accuracy with trivial feature manipulation.

1.3 Proposed Approach and Key Contributions

We gathered four separate, publicly available phishing datasets developed by other researchers and applied adversarial sampling techniques to evaluate the robustness of the trained model against artificially generated samples. Although we do not show any solution to address this current threat, we demonstrate the vulnerability of the existing approaches and explore the datasets' robustness against the engineered features and the learning models. Our key contributions are as follows:

- We survey a full range of phishing detection techniques focusing on machine learning-based approaches and model the threat against them by the attackers' access, knowledge, and goal, which the attackers utilize to attack any given trained classifier model.
- We show the weakness of some well-known machine learning approaches and emphasize how a phisher can generate new phishing website instances, i.e., adversarial samples, to evade the machine learning classifier in each of these approaches.
- We define phishing instances' vulnerability level, which quantifies the attackers' efforts and optimize the attacker's effort to generate adversarial samples.
- Finally, we describe a clustering approach to direct the attacker in generating better adversarial samples with a higher likelihood of success to bypass the classifier. We show that the clustering approach identifies data samples with higher vulnerability levels.
- We built an experimental setup, conducted a wide range of experiments and analyzed how vulnerable the datasets and learning models are by testing against the generated samples.

The rest of this paper is organized as follows. In Section 2, we describe a wide range of defense mechanisms against phishing attacks in the literature. Also, we describe the various adversarial attacks against the machine learning classifiers in non-phishing domains. In Section 3, we model the threat from three points of view: attackers' *goal*, *knowledge*, and *influence*. In Section 4, we simulated an adversarial sampling attack followed by assessing the vulnerability level and quantifying the cost of the attack. In Section 5, we describe the clustering-based approach for generating samples with a higher chance of bypassing the classifier at a lower cost. In Section 6, we explain the results of our experiments to prove the robustness of the classifiers and datasets against these attacks. In Section 7, we conclude the paper and discuss some future work.

2 Related Work

2.1 Machine Learning for Phishing Detection

Researchers engineered novel sets of features from diverse perspectives based on public datasets of phishing and legitimate websites in prior machine learning approaches for phishing detection. Machine learning algorithms are well suited to assimilate common attack patterns such as hidden fields, keywords, and page layouts across multiple phishing data instances and train learning models that are resilient to small variations in future unknown phishing data instances.

According to a Symantec report [10], the number of URL obfuscation based phishing attacks was up by 182.6% in 2017. Some URL obfuscation techniques used by attackers are the misspelling of the targeted domain name, using the targeted domain name in other parts of the URL like the sub-domain, and adding sensitive keywords like 'login', 'secure', or 'https'. Researchers defined machine learning features based on the URLs to capture these techniques and trained learning models. For example, Jiang *et al.* [5] merged information from DNS and the URL to develop a Deep Neural Network (DNN) with the help of Natural Language Processing (NLP) to detect phishing attacks. While other approaches need to specify features explicitly, this method extracts features automatically. However, this approach relies on information from third-party services like search or DNS queries to leverage the feature set and make the feature set more reliable; it also endangers users' privacy. Third-party inquiries to fetch the feature value reveals the browsing history of the end-users.

Sahinguz *et al.* [11] addressed this issue and proposed a real-time detection mechanism based on Natural Language Processing (NLP) of URLs on a large dataset of features derived from URL obfuscation without requiring third-party inquiry, and achieved an accuracy of more than 95%. While URL based phishing detection approaches are promising but have two limitations of (i) having full control over URLs by attackers that can create any URL and (ii) not considering pages' contents. The website's content is the most critical factor in luring the end-users rather than the URL or domain name themselves. Therefore, any solution distinct from the websites' content would not be useful in the real world.

Niakanlahiji *et al.* [4] introduced PhishMon, a scalable feature-rich framework with a series of new and existing features derived from HTTP responses, SSL certificates, HTML documents, and JavaScript files and reported accuracy of 95%.

Shirazi *et al.* [7] observed two concerns with existing machine learning approaches: a large number of training features and bias in the type of datasets used. Their study focused on the features derived from the domain name usage in phishing and legitimate websites, not the URL, and reported an accuracy of $97 - 98\%$ on the chosen datasets.

Li *et al.* [8] proposed an approach to extract the features from both URL and webpage content and ran multiple machine learning techniques, including GBDT, XGBoost, and LightGBM, in numerous layers, referred to as stacking approaches. The experiment has been conducted on three datasets, of which two

are large ones with 50K instances, and the accuracy is more than 97% in all cases. Although this approach is similar to recent machine learning approaches and does not use third-party services, it is similar to other previous work [7].

While these approaches have demonstrated excellent results for detecting phishing websites, they also suffer from severe disadvantages due to adversarial sampling, as we show in the following discussion.

2.2 Learning in Adversarial Context

Defense mechanisms have been proposed in the literature, widely employed machine learning techniques to counter phishing attacks. However, adversarial sampling attacks can threaten current defense mechanisms. An adversarial sampling attack is when an adversary generates a phishing data sample based on existing phishing samples to avoid detection by the classifier. In general, such a sample is called an *adversarial sample*. While there is some general analysis of the vulnerabilities of classification algorithms and the corresponding attacks [12], to the best of our knowledge, there is no other study on adversarial sampling in the context of the phishing attacks. Thus far, researchers have studied and formulated these threats in a general manner or other application contexts like image recognition. In the following, we briefly explore these efforts.

Dalvi *et al.* [9] studied the problem of adversary learning as a game between two active agents: data miner and adversary. The goal of each agent is to minimize its cost and maximize the cost to the other agent. The classifier adapts to the environment and its settings either manually or automatically in this approach. The authors assumed that both sides, including data miners and adversaries, have perfect knowledge about a problem. However, this assumption does not hold in many situations as we modeled our adversary in Section 3, and elaborate on why the adversary cannot have perfect knowledge.

Xiao *et al.* [13] explored the vulnerabilities of feature selection algorithms under adversarial sampling attacks. They extended a previous framework [14] to investigate the robustness of three well-known feature selection algorithms.

There are a few approaches that create more secure machine learning models. Designing a secure learning algorithm is one way to build a more robust classifier against these attacks. Demontis *et al.* [15] investigated a defense method that can improve the security of linear classifier by learning more evenly-distributed feature weights. They presented a secure SVM called Sec-SVM to defend against evasion attacks with feature manipulation. Wang *et al.* [16] theoretically guaranteed the robustness of the k-nearest neighbor algorithm in the context of adversarial examples. They introduced a modified version of the k-nearest neighbor classifier where k is equal to 1 and theoretically guaranteed its robustness in a large dataset.

Shirazi *et al.* [17] used an adversarial autoencoder to generate synthesized phishing samples and tested these samples against models trained with real-world data. It is shown that a portion of generated samples was able to evade existing detection models. Some synthesized samples have been used for training and showed the new learning models are more robust against adversarial attacks

and hold higher accuracy. In other words, real-world phishing site data augmented with synthesized data used for training the model provides more robust classifiers which are more effective for phishing detection.

Finally, there are some tools for benchmarking and standardizing the performance of machine learning classifiers against adversarial attacks in the literature. *Cleverhans* [18] is an open-source library that provides an implementation of adversarial sample construction techniques and adversarial training for image datasets. Given the lack of benchmarking tools for the phishing problem, we tested our approach with our attack strategies and implementation.

3 Threat Model

In this section, we model the adversarial sampling attack against machine learning-based phishing detection approaches. We start with the attacker's *goal*, *knowledge*, and *influence* in general machine learning solutions, and then we explain them in the context of our phishing problem. We model the adversarial sample generation for existing phishing instances based on the attackers' abilities and then evaluate the cost that the adversary has to pay for the successful execution of this attack. Finally, we define the vulnerability level for the dataset.

3.1 Attacker's Goal

Biggioa *et al.* [19] explored three different goals for attackers in reactive arms race namely *security violation*, *attack specificity*, and *error specificity*. An attacker's goal in the *security violation* is to evade well-known security metrics, including confidentiality, availability, and integrity. The attacker may violate the availability of the system by a denial-of-service attack. In this case, if the system cannot accomplish the desired task due to the attacker's behavior, the availability of the service would be affected. The attacker needs to obtain users' sensitive and private information with approaches like reverse-engineering to violate the user's confidentiality.

In a phishing context, the adversary will attack the integrity of the system. The integrity is violated if the attack does not permit the regular system behavior; however, the attacker violates the accuracy of the classifier *e.g.* by making the classifier label the maliciously crafted phishing instances as legitimate to evade the classifier. The attack *specificity* depends on whether a specific set of samples (like phishing) being incorrectly classified for any given sample. The error *specificity* relates a specific type of error in the system and degrades the classifier's scores.

3.2 Attacker's Knowledge

An attacker may have different levels of knowledge about the machine learning model. An attacker might have detailed knowledge, *i.e.*, *white-box* or *perfect knowledge*, minimal knowledge about the model called *zero knowledge* [13,

19] and limited knowledge about the model known as *gray-box*. If the adversary knows everything about the learning model, parameters, and the training dataset, including the classifier parameters, the attacker has *perfect knowledge*. In the case of *zero knowledge*, the adversary can probe the model by sending instances and observing the results. The adversary infers information about the model by choosing appropriate data samples. In the case of *limited knowledge*, it is assumed that the adversary knows about features and their representation and the learning algorithm. However, the adversary does not know about the training set or the algorithm's parameters.

From the dataset point of view, the attacker may have partial or full access to the training dataset. The attacker may also have partial or full knowledge about the feature representation or feature selection algorithm and its criteria. In the worst-case scenario, an attacker may know about the subset of selected features.

3.3 Attacker Influence

Two major types of attacker influence have been defined in the literature, namely *poisoning* and *evasion* attacks. In a *poisoning* attack, the adversary generates and injects adversarial instances in the training phase. Adversarial instances are the ones with manipulated labels. For example, email providers use spam detection services and give the users the ability to override the email's label *e.g.* re-labeling a spam email as non-spam to deal with false-positive detection cases. The system benefits from the user's labeling to improve accuracy by updating the training set. However, in a *poisoning* attack, an attacker, with an authorized email account in the system, can re-label the correctly detected spam emails as non-spam to poison the training set which results in a poor learning model that is easy to bypass even with slightly manipulated phishing instances.

In *evasion* attack, the attacker does not have access to the training set and intentionally and smartly manipulate features to avoid samples being labeled correctly by the classifier at the testing phase. Similar to the previous example on the spam detection system, a phisher may send an email with intentionally misspelled words to evade the classifier.

3.4 Our Assumption

In this subsection, we define the threat model that we assumed in this work.
Attacker's Goal. We consider that the adversary attacks the *specificity* of the learning model. Adversary generates new phishing samples that are labeled incorrectly by classifier as legitimate. Thereby, these samples will bypass the learning model and deceive the end-users. Also, with respect to error *specificity*, the adversary wants to decrease the system's ability to detect phishing instances and increase false-negative rate of the system.
Attacker's Knowledge. We assume that the adversary has *limited knowledge*. Adversary only knows about the feature set. However, it does not know about

the training set, the learning algorithm that has been used, or the classifier's training parameters.

Attacker Influence. We assume that the adversary can test as many instances as needed and get the results. Under this assumption, an adversary can create a large number of new samples and test them to see if they can bypass the model.

In the next section, we describe our sample generation approach and outline our method for measuring the effectiveness of the samples in lowering the classifier's accuracy.

4 Adversarial Sampling for Phishing

We simulate the attacker's approach to generate new adversarial samples based on the existing phishing samples. The adversary generates new samples by manipulating phishing samples' features and then checks whether the generated samples evade the classifier. A phishing sample evades the classifier if it is labeled as a legitimate sample. All such generated samples that bypass the machine learning classifier are adversarial samples. The motivation for using features from existing phishing samples is to guarantee that the generated samples are guaranteed to possess some key phishing characteristics. We assume that the attacker has full control over the URL and phishing page content except for the domain name part. The attacker has *limited knowledge* about the classifier and features, as discussed in Section 3.2.

4.1 Defining the Dataset

We use similar notation to that used in [13]. The dataset has been generated by a procedure $\mathcal{P} : \mathcal{X} \mapsto \mathcal{Y}$. We denote a set D with n samples as $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$. Each instance in the dataset has d features that are represented as a d-dimensional vector:

$$x_i = [x_i^1, \cdots x_i^d] \in \mathcal{X} \tag{1}$$

Each instance x_i is tagged to a target label $y_i \in \mathcal{Y}$. There are two types of labels for instances: legitimate (L) instances labeled as 0 and phishing (P) instances labeled as 1, which implies that $\mathcal{Y} = \{0, 1\}$. A learning algorithm trains on this dataset and will be expected to predict the label of an unknown website instance correctly, i.e., 0 for legitimate, or 1 for phishing.

4.2 Selecting Features for Manipulation

To specify a subset of features, we introduce the notation $\pi = \{0, 1\}^d$ to denote a d-bit value. If the value in the i^{th} bit of π is 0, then that feature is not selected for manipulation, and if the value is 1 then it is chosen for manipulation. We use Π_s^d to denote the set of all possible combinations of s features that have been selected out of total d features. The size of this set is given by $\binom{d}{s}$.

To illustrate, $\pi \in \Pi_s^d$ denotes an element π when there are s numbers of features selected for manipulation out of total d features. For example, $\Pi_1^3 = \{100, 010, 001\}$ means all possible subsets of 3 features when only 1 feature has been selected for manipulation. In addition, $\pi_1 \in \Pi_1^3$ is 100, $\pi_2 \in \Pi_1^3$ is 010, and $\pi_3 \in \Pi_1^3$ is 001.

The first step towards generating samples is to select one or more features for manipulation. The generative algorithm can be represented in terms of function $h(x_i)$ that selects a feature subset π by minimizing the number of features and cost. In Table 1, we defined the notation used for describing our approach.

Table 1. Table of Notation

Notation	Meaning
$\mathcal{D}$	Dataset
$\mathcal{X}$	Set of instances in the dataset
$\mathcal{Y}$	Set of labels in the dataset
n	Number of instances in the dataset
d	Number of feature vectors for each instance
x_i	i^{th} instance in the dataset
x_i^j	j^{th} feature value of i^{th} instance
$\neg$	Not operand
π	d-bit string indicating chosen features for manipulation
Π	Set of all possible feature combination
Π_i^j	Set of i feature selected out of total j features
$\neg\pi$	negation of π; if $\pi = 001$, $\neg\pi = 110$
$h(x_i)$	Select a feature subset π for a given x_i by minimizing number of features and cost
X^i	Set of all values of feature i
X_P^i	Set of all values of feature i for phishing instances
$\times$	Cartesian product
$*$	Cross vector product

4.3 Assigning New Values to Selected Features

After defining the features that will be manipulated, we must assign new values to them. We assume that each value will be replaced only by values that appeared in existing phishing instances. The intuition is that if the value has been found to be assigned to that feature previously for a phishing instance, then that feature value is more likely to be found in another phishing instance. In Algorithm 1, in lines 4 to 5, we used *Cartesian Product* to generate all possible combination for each feature, taking the values from phishing instances.

For the features that have been selected for manipulation, the corresponding bit in π will be 1. In this case, the $(\pi * n\_fea)$ term will select new feature values and assign them. For the features that have not been selected for manipulation,

Algorithm 1: Generating Samples

Input: $x = [x^1, x^2, \ldots, x^d]$; $\mathrm{X}_P^i, \forall i \in 1 \ldots d$

/* Phishing instance x used to generate samples, feature value i obtained from all phishing instances for all features 1 to d. */

Output: genSamples; advSamples

/* Array of generated samples; array of generated adversarial samples */

/* Refer to Table 1 for the notation. */

/* Initialization */

1 $advSamples \leftarrow \{\}$

2 $genSamples \leftarrow \{\}$

3 $PhVal \leftarrow 1$

/* *Cartesian Product* generates all possible feature values of current phishing instances. */

4 **for** $k \leftarrow 1$ *to* d **do**

5 $\quad PhVal \leftarrow PhVal \times X_P^k$

6 **forall** $\pi \in \Pi$ **do**

7 $\quad$ **forall** $n\_fea \in PhVal$ **do**

8 $\quad\quad nw\_smp \leftarrow (\pi * n\_fea) + (\neg\pi * x)$

$\quad\quad$ /* For each feature, if the feature position is selected for manipulation, the feature value will be replaced by a corresponding feature value from some phishing instance; otherwise, the feature value remains unchanged. */

$\quad\quad$ /* Check if the generated sample differs from the input sample. */

9 $\quad\quad$ **if** $x \neq nw\_smp$ **then**

10 $\quad\quad\quad genSamples \leftarrow genSamples \cup nw\_swp$

11 $\quad\quad\quad$ **if** *x is labeled as legitimate by the classifier* **then**

12 $\quad\quad\quad\quad advSamples \leftarrow advSamples \cup\ nw\_smp$

the corresponding bit in π will be 0. In this case, $(\neg\pi * x)$ will be used, and it will keep original input instance values.

If the newly generated sample is equal to the given input, we discard it as it doe not hold any changes; Otherwise, we include it in the set of *genSamples*. We test the generated sample with a classifier to check the label. We add generated phishing samples that are labeled as legitimate to *advSamples*. These are samples that have been classified incorrectly. These are samples that are able to evade the classifier. This is defined in lines 9-12 in Algorithm 1.

4.4 Adversary Cost

Attackers have to handle two challenges for generating adversarial samples. From a machine learning point of view, the dataset includes feature vectors. Still, the attacker has to change the phishing website to generate the desired vector similar to adversarial samples. For example, if a feature is URL length, the adversary can generate a new URL with the desired length based on adversarial samples.

This is not a trivial process, and it has a considerable cost for the attacker. Whereas adversarial samples may have a higher chance of evading the classifier, they may not be visually or functionally similar to the targeted websites. This increases the chance of being detected by the end-user. Thus, the adversary wants to minimize two parameters: the number of manipulated features and the assigned feature values. We consider this as a cost function for the adversary.

In the previous section, we discussed how the attacker controls the number of manipulated features, but it is not the only parameter. If the manipulated feature values differ much from the original values, it will increase the classifier's chance of evading. We study this hypothesis in Section 6. This will also change the website's visual appearance or behavioral functionality from the targeted website, thereby increasing the chance of phishing websites being detected by the end-user.

In this work, we used the *Euclidean distance* between the original phishing sample and newly generated sample to estimate the cost; a higher distance indicates a higher cost. Consider x_i to be a phishing instance and $x_i^{'}$ a manipulated one based on the original x_i instance. Both are vectors of size n. The *Euclidean distance* between x_i and $x_i^{'}$ will be calculated by Equation 2:

$$d(x_i, x_i^{'}) = \sqrt{\sum_{k=1}^{n} (x_i^k - x_i^{'k})^2} \tag{2}$$

If l is the number of manipulated features to generate $x_i^{'}$ from x_i, and d is *Euclidean distance* between them, the total cost c will be derived from this equation: $\mathcal{C}(x_i, x_i^{'}) = (l, d)$. This tuple will be used to evaluate the total cost for generating the adversarial samples.

4.5 Vulnerability Level

A phishing instance is vulnerable at the level l with the cost d if there is at least one adversarial instance generated from this phishing instance that can bypass the machine learning classifier with l manipulated features and a distance d from the original instance. In other words, we call this instance vulnerable if manipulating l features of the original phishing instance with a distance of d allows it to bypass the classifier. The attacker's goal here is optimizing the l and d, a multi-objective optimization problem for the attacker. For example, suppose we have a phishing instance, and we are able to generate an adversarial sample by manipulating 3 features with *Euclidean distance* of 2.7. In that case, we say that the original phishing sample is vulnerable at the level of 3, with a cost of 2.7.

5 Directed Adversarial Sampling

In the approach described so far, the adversary needs to adopt a trial-and-error with a given phishing sample, *i.e.*, the attacker is not sure whether a given phishing sample can be used to generate an adversarial sample that can bypass the

classifier. This process is further constrained if the attacker attempts to minimize the cost of generating such adversarial samples. As a result, the attacker's effort is increased significantly as the attacker needs to experiment with each sample and try various feature manipulation combinations to generate an adversarial sample. To address these problems, we describe a clustering-based pre-processing approach that *directs* the attacker towards selecting the best possible phishing samples that are likely to bypass the classifier. Simultaneously, this approach also helps the defender identify those features that are more likely to be useful to adversaries and refining the existing machine learning model.

5.1 Outline of Clustering Approach

In general, the clustering of data samples using standard approaches like the k-means algorithm [20] generates groups of samples that share a significant number of common features or have similarities in a few dimensions. This feature of clustering algorithms is the key intuition for our improved adversarial sample generation technique.

Concisely stated, our approach first clusters the phishing samples using a standard clustering algorithm such as k-means and initializes a per-cluster counter to zero. Next, we select one random sample from each of the clusters to generate adversarial samples using Algorithm 1. If the generated sample is adversarial, we increment the per-cluster counter of the respective cluster. Next, we repeat the experiment with a few more samples by progressively selecting more samples from successful clusters, *i.e.*, the cluster with higher per-cluster counter values, after the initial testing period.

We note that, based on the properties of clustering, *i.e.*, similar data samples are placed in the same cluster; we surmise that a cluster that has contributed to adversarial samples is more likely to contribute to many other adversarial samples. Our experimental results show that this is indeed the case and demonstrate that the clustering approach significantly improves the success rate of generating adversarial samples.

5.2 Correlating Cluster Membership and Adversarial Sampling

From our experimental results, we make a few important observations and state them here. We clustered adversarial samples using the existing clusters of the data. If an adversarial sample belongs to a different cluster than the cluster to which the original phishing sample belonged, we denote such an adversarial sample as *transferred sample*. This definition captures a key notion that an adversarial sample is likely to belong to a different cluster due to the feature manipulation. When viewed from a different perspective, this indicates that a generated sample is likely to be an adversarial sample if the generated sample's cluster membership is different from the original phishing sample from which it was generated. We demonstrate this characteristic using our experimental results in Section 6.

Using this notion of transferred samples, we define the correlation between adversarial samples and cluster membership transfer. For this purpose, we calculate the probability of an adversarial sample being transferred to a new cluster. Formula 3 articulates the probability of success in generating an adversarial sample when the generated sample is transferred to a new cluster. In this formula, Ay denotes adversarial samples, and Tr represents transferred samples in a given experiment.

$$P(Ay \mid Tr) = \frac{P(Ay \cap Tr)}{P(Tr)} \tag{3}$$

In the next section, we evaluate our clustering approach, validating the basic approach, and in the process, demonstrate some important results that enable an adversary to generate effective adversarial samples.

6 Experiments and Results

This section shows the effectiveness of our proposed adversarial sampling attack that degrades existing learning models' accuracy and efficacy. First, we discuss the datasets utilized, and we elaborate on three different experiments we have conducted and the results.

6.1 Used Datasets

We obtained four publicly available phishing datasets on the Internet, and the details of these datasets are given below.

Dataset 1**: DS-1**: This set includes 1000 legitimate websites from *Alexa.com* and 1200 phishing websites from *PhishTank.com*; 2200 in total. Each instance in this dataset has eight features, and all are related to the websites' domain name. The features used are domain length, presence of a non-alphabetic character in the domain name, the ratio of hyperlinks referring to the domain name, the presence of HTTPS protocol, matching domain name with copyright logo, and matching domain name with the page title. With these features, Shirazi *et al.* [7] reported an accuracy of 97-98% in the experiments, which is significantly high.

Dataset 2**: DS-2**: Rami *et al.* [21] created this dataset in 2012 and shared it with UCI machine learning repository [22]. This set includes 30 features that are divided into five categories: *URL based*, *abnormal based*, *HTML-based*, *JavaScript based*, and *domain-name based* features. This dataset includes 4898 legitimate instances from *Alexa.com* merged with 6158 phishing instances from *PhishTank.com*; more than 11000 in total, making it the most extensive dataset that we have used in this study.

Dataset 3**: DS-3**: In 2014, Abdelhamid *et al.* [23] shared their dataset on UCI machine learning repository [22]. This dataset includes 651 legitimate websites and 701 phishing websites; 1352 instances in total. The features include HTML content-based features and some features that require third-party service inquiries, such as DNS servers that perform domain-name age lookup and

so on. The authors report a detection accuracy in the range of 90%-95% in their experiments.

Dataset 4: DS-4: This dataset is the most recent, from the year 2018, that is publicly available. It has been created by Tan *et al.* [24] and was published on Mendeley [3] dataset library. This set contains 5000 websites from *Alexa.com* and as well as those obtained by web crawling, labeled as legitimate, and 5000 phishing websites from *PhishTank.com* and *OpenPhish.com*. The authors collected this data from January to May 2015 and from May to June 2017. This dataset includes 48 features, a combination of URL-based and HTML-based features.

Table 2 summarizes the number of instances, features, and the portion of legitimate vs. phishing instances in each dataset. We have datasets with a large number of instances, DS-2 and DS-4, with 11000 and 10000 instances, respectively. We also have a small dataset DS-3 with 1250 instances. With respect to the number of features, DS-1 has just seven features, whereas DS-4 has 48 features. Besides, each dataset's features are selected from different points of view, such as URL-based features in DS-2, DS-3, and DS-4, or domain-related features in DS-1, and HTML-based features in DS-2 and DS-4. These variations validate our hypothesis in a stronger and more general sense. Also, it shows that adversarial sampling is a severe problem that may manipulate different types of features to evade the classifier.

Table 2. Number of instances, features, and portion of legitimate and phishing websites in each dataset

Dataset	**Data shape (#)**		**Labels (%)**	
	Size	**Features**	**Legitimate**	**Phishing**
DS-1	2210	7	44.71	55.29
DS-2	11055	30	55.69	44.31
DS-3	1250	9	43.84	56.16
DS-4	10000	48	50.0	50.0

6.2 Machine Learning Metrics

For evaluating the robustness of the classifier against the adversarial samples, we used standard machine learning metrics. We calculated: True Positive Rate (TPR), Positive Predictive Value (PPV), F1, and Accuracy (ACC) to evaluate the performance of our proposed approach.

6.3 Phishing Detection Accuracy without Adversarial Sampling

In the first experiment, we tested each dataset's performance against a wide range of standard classifiers. We labeled phishing websites in all datasets as +1

[3] https://data.mendeley.com/

Table 3. Data Definitions

Metric	Definition
N_L	Total number of legitimate websites
N_P	Total number of phishing websites
$N_L \rightarrow L$	Number of legitimate classified as legitimate
$N_L \rightarrow P$	Number of legitimate classified as phishing
$N_P \rightarrow P$	Number of phishing classified as phishing
$N_P \rightarrow L$	Number of phishing classified as legitimate

Table 4. Definition of performance metrics

Score	Formula	Description
TPR	$\frac{N_P \rightarrow P}{N_P}$	*Correctly* classified phishing
PPV	$\frac{N_P \rightarrow P}{N_P \rightarrow P + N_L \rightarrow P}$	*Correctly* over *total* predicted phishing
F1	$2 * \frac{TPR * PPV}{TPR + PPV}$	Harmonic average of TPR and PPV
ACC	$\frac{N_L \rightarrow L + N_P \rightarrow P}{N_L + N_P}$	Classified *correctly* in the dataset

and legitimate websites as -1. We used five-fold cross-validation to avoid overfitting issues and test the learning model's performance against unseen data instance classification. We used six different classification algorithms namely Decision Tree (DT), Gradient Boosting (GB), Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) with two different kernels: Linear (lin) and Gaussian Radial-basis function (rbf) to make different algorithms comparable. We repeated each experiment 10 times and reported the average and standard deviation of the results. Table 5 explains the achieved results in this experiment.

For DS-1, RF and GB both generate the highest ACCs and the TPRs for both classifiers are comparable.. Also, DS-1 has the best average of TPR among all datasets. RF gives the best TPR (94.25%) and ACC (95.76%) on DS-2. Interestingly, the DT does not generate a good TPR (86.77%). The DS-3 dataset experiments did not yield a high TPR or the ACC. Both GB and SVM with Gaussian kernel have the TPRs close to 87%, which is not that good. The best ACC, for this dataset, is from GB, with 83%. The experiment on DS-4 gave excellent results. Both GB and RF gave a TPR over 97% and accuracy of 97%, which are very high. This dataset has the best average of ACC among different classifiers meaning this dataset performs very well with different types of classifiers. With six different classifiers, the experiments on both DS-1 and DS-4 show an average ACC of more than 94%, which is significantly high.

We used a single metric of F1 to compare all classifiers and datasets together. Table 6 shows the best F1 score for each dataset with the classifier that has produced that result. It is evident from this table that both GB and RF generate

Table 5. Evaluation of model against different classifiers with two metrics.

(a) TPR

Cls.	DS-1	DS-2	DS-3	DS-4	Avg.
DT	95.25	86.77	84.97	96.14	**95.25**
GB	96.18	92.25	87.23	97.65	**96.18**
KNN	95.93	90.61	84.95	93.97	**95.93**
RF	96.25	94.25	85.84	97.85	**96.25**
SVM(l)	95	89.62	86.71	94.93	**95**
SVM(r)	93.67	91.88	87.88	95.69	**93.67**
Best	**96.25**	**94.25**	**87.88**	**97.85**	

(b) ACC

Cls.	DS-1	DS-2	DS-3	DS-4	Avg.
DT	94.8	92.1	82.51	95.73	**91.29**
GB	95.49	94.32	83.76	97.52	**92.77**
KNN	94.82	92.21	81.16	93.76	**90.49**
RF	95.35	95.76	82.89	97.8	**92.95**
SVM(l)	93.96	92.4	79.16	94.38	**89.98**
SVM(r)	93.96	94.14	82.4	95.2	**91.43**
Best	**95.49**	**95.76**	**83.76**	**97.8**	

Table 6. The classifier that holds best F1 on each dataset has been selected. TPR and ACC are also reported for comparison

Metric	DS-1	DS-2	DS-3	DS-4
Best Classifier	**GB**	**RF**	**GB**	**RF**
Best F1	**95.94**	**95.17**	**85.83**	**97.8**
TPR	96.18	94.25	87.23	97.85
ACC	95.49	95.76	83.76	97.8

the best results among all of the experiments, so we selected these two classifiers for the next experiments.

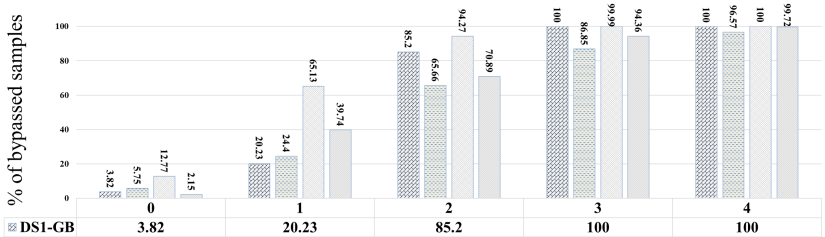
6.4 Adversarial Sample Generation

We reserved 200 random phishing instances in each dataset and then trained the model without the 200 random reserved phishing instances. The generated adversarial samples need to be similar to the phishing examples; otherwise, those cannot be assumed to be phishing instances. We used previously seen values in the phishing instances to assign new values to the features and generate new instances. With this strategy, it is guaranteed that the newly assigned value is valid and has already been seen in other phishing instances in the dataset. We discussed this process earlier in Section 4. We randomly selected features, up to four different features, and changed each feature's values with all possible feature values. If an adversarial sample is generated, we consider the original phishing instance to be vulnerable. A given phishing instance can generate several adversarial samples with varying costs, as defined in Section 4.4. We call the phishing samples with the lowest cost of generating adversarial samples as *optimized samples*.

6.5 Robustness of the Learning Model

This experiment studies the robustness of datasets and learning models against generated adversarial samples. We selected one classifier that performs best for each dataset based on the F1 score from Table 6. For the datasets DS-1 and DS-3, we selected GB, and for DS-2 and DS-4, we chose RF.

Fig. 1. Robustness of datasets against adversarial samples



	0	1	2	3	4
DS1-GB	3.82	20.23	85.2	100	100
DS2-RF	5.75	24.4	65.66	86.85	96.57
DS3-GB	12.77	65.13	94.27	99.99	100
DS4-RF	2.15	39.74	70.89	94.36	99.72

Number of manipulated features

In this experiment, we counted the number of reserved phishing instances that are vulnerable. This means that there should be at least one adversarial sample with the lowest cost based on the original sample. With small perturbation in these instances, they can bypass the classifier and elude the users to release their critical information. Based on our hypothesis, these are vulnerable instances and can be assumed as a threat to the learning model. We repeated each experiment ten times and reported the average of the results.

Figure 1 shows the results of our experiment. The x-axis shows the number of manipulated features; zero manipulated feature means that the test happened with the original phishing instances detected correctly by the classifier. The trend of results reveals that increasing the number of perturbations results in an increase in the number of evaded samples proportionally. We continued increasing the perturbed features for up to four different features at a time. We observed that with four features, almost all manipulated phishing instances bypass the classifier model.

For example, Figure 1 shows that less than 4% of phishing instances in DS-1 can bypass the classifier without any perturbation. With only one manipulated feature, more than 20% of phishing instances can bypass the classifier. With two manipulated features, almost all instances can bypass the GB. The results are almost the same for other datasets. In another case, while just 12% of original phishing instances (the instances without any changes) have been misclassified in DS-3, the results significantly go up to 65% with only one perturbed feature.

This experiment shows how vulnerable the machine learning models are to the phishing problem. Small perturbation on features can bypass the classifier and degrade the accuracy significantly.

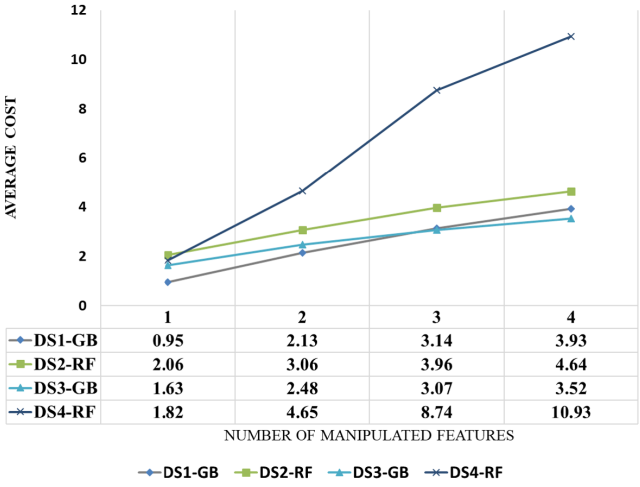
6.6 Dataset Vulnerability Level

In this experiment, we studied the cost that an adversary has to pay to bypass a classifier. From an adversary perspective, it is not inexpensive to manipulate an instance with new feature values to create an adversarial sample. In Section 4.4, we assessed the cost and in Section 4.5, we defined the term *vulnerability level* for one instance. Once again, we reserved 200 random phishing instances

from each dataset and chose the classifier for each dataset based on Table 6. For datasets DS-1 and DS-3, we chose GB while we chose RF for both DS-2 and DS-4 datasets. Averaging the *vulnerability level* for each of the 200 selected instances and repeating the experiment ten times, we assessed the whole dataset's vulnerability level.

Figure 2 presents the results of this experiment for all datasets for two parameters: the number of manipulated features and the average cost of adversarial instances. It is evident that, by increasing the number of manipulated features, the cost also increases steadily. For example, for the dataset DS-1, the average cost, for adversarial samples, with one manipulated feature is 0.95, and with four manipulated features, the cost is 3.93.

Furthermore, the average cost for some datasets is more than that of other datasets. For example, in the DS-4, the adversary has to pay more cost, particularly when the number of features increases to three and four compared to the other datasets. This shows that this dataset is more robust against these attacks and has a lower vulnerability level.



	1	2	3	4
DS1-GB	0.95	2.13	3.14	3.93
DS2-RF	2.06	3.06	3.96	4.64
DS3-GB	1.63	2.48	3.07	3.52
DS4-RF	1.82	4.65	8.74	10.93

Fig. 2. The manipulation cost for adversarial samples based on number of manipulated features

6.7 Cluster Directed Adversarial Sampling

We discuss using the clustering approach described in Section 5 and present the results. In this experiment, we calculated the probability of *transferred samples* and *adversarial samples* as we discussed in Section 5. For each dataset, we calculated the probability of generating adversarial samples and also the probability of such a sample being transferred to a new cluster from the original cluster.

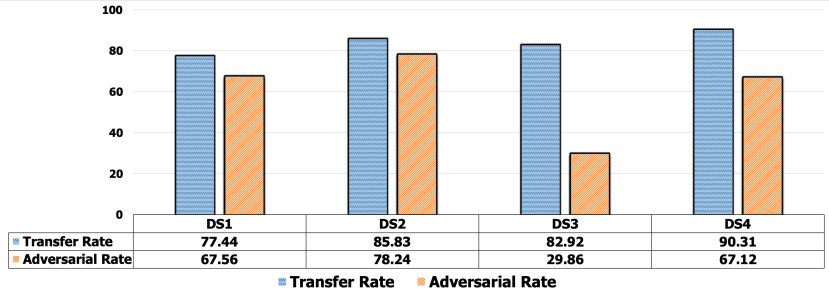


Fig. 3. Ratio of bypassing and transferring adversarial samples in tested datasets

Figure 3 shows the adversarial sampling probability and transferred samples for each dataset. On average, more than 60% of all adversarial samples in DS-1, DS-2, and DS-4 datasets were able to bypass the classifier. For the DS-3 dataset, the bypassing rate is around 30%.

Another measure in Figure 3 is the transferring rate in which new adversarial samples are categorized in a new cluster. In all datasets, we see an average of at least 75% or more. This reveals that the majority of adversarial samples belong to a different cluster rather than the original cluster. This is the first significant finding related to the clustering approach.

This experiment investigated adversarial sampling and transferring probability based on each cluster. Figure 4 depicts how these probabilities varied among different clusters. Figure 4(a) for DS-1 shows adversarial samples are uniform, bypass classifiers, and transferred among clusters, and it is not significantly different among different clusters. Figure 4(b) for DS-2 shows some variations among different clusters. Clusters 3 and 8 have the highest chance of generating adversarial samples. Cluster 5 has a significantly low chance of transferring an adversarial sample.

The chance of an adversarial sample generation does not vary among different classifiers, as shown in Figure 4(c) for DS-3. The same pattern can be seen for transferring as well. There is a big gap between the probability of generating adversarial samples and transferring in this dataset, something that has not been seen in other datasets.

Figure 4(d) shows results for DS-4. Cluster 8 has the lowest chance of generating adversarial samples, and clusters 3 and 4 have the highest one. Cluster 4 also has the highest chance of transferring to other clusters.

6.8 Conditional Probability of Transferred Samples

This experiment used conditional probability to show how adversarial samples and transferred samples are co-related to each other. For this purpose, we calculate the probability that an adversarial sample is transferred to another cluster. It shows how likely an adversarial sample would be transferred to a new cluster. Figure 5 depicts these results.

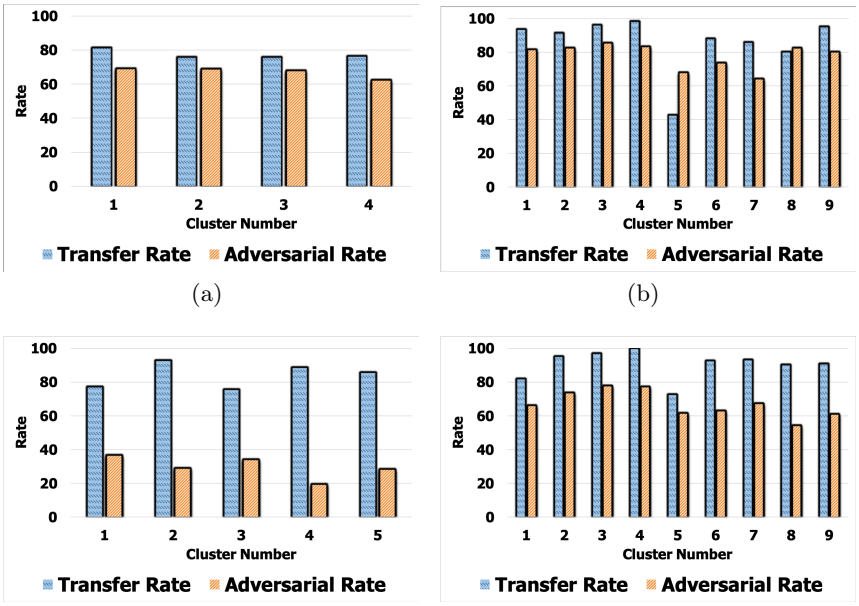


Fig. 4. Distribution of bypassed and transferring samples for each cluster in all of tested datasets: (a) DS-1; (b) DS-2; (c) DS-3; and, (d) DS-4

In this Figure, $Ay|Tr$ shows the probability of a manipulated sample being an adversarial sample (Ay) given that the sample is transferred (Tr) to a new cluster. In the same way, $NAy|NTr$ shows the probability of not being an adversarial sample (NAy) given that the sample is not transferred (NTr) to a different cluster.

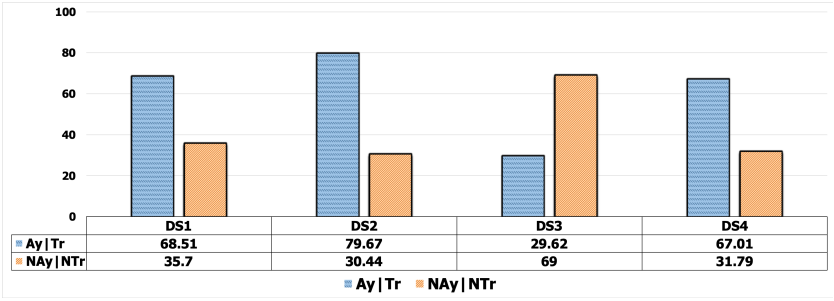


Fig. 5. Conditional Probability of Adversarial Samples

This knowledge for an adversary is compatible with the threat model defined in Section 3. In our proposed model, an attacker has access to the predict function and phishing website.

Figure 5 shows that in DS-1, DS-2, and DS-4, the probability of generating an adversarial sample when a manipulated sample is transferred to a new cluster rather than its original cluster is at least 65%. It gives hints to the attackers to target features that, with manipulation, the instance would transfer to another cluster. We also calculated when conditional of these two parameters were are not happening. Based on the results, there is not a significant correlation between these two probabilities.

6.9 Selecting Best Cluster

As discussed earlier in Section 4.4, generating adversarial samples is not an inexpensive process, and an adversary would like to optimize this effort. This section defines the probability of generating adversarial samples and identifying clusters with a lower cost to optimize an adversary's efforts. To achieve this goal, we considered the following parameters.

Probability of Cluster Membership. Using the clustering algorithm, each data instance belongs to one cluster, and not all clusters have the same number of instances. In this case, the probability of a data instance belonging to different clusters changes across different clusters. We calculate the probability of an instance is a member of each cluster when considered over the universe of all instances.

Probability of samples belonging to cluster i is denoted as $P(c_i)$. Also, in_i is set of instances in cluster i, and ins is set of all phishing instances. $P(c_i)$ will be calculated as follow:

$$P(c_i) = \frac{|in_i|}{|ins|} \tag{4}$$

Probability of Membership Transfer For generating new samples, we manipulate the feature values of each instance. In the next step, using the clustering algorithm, we find cluster membership of each new instance. The cluster to which a generated sample belongs may or may not be the same as the cluster of the original sample used to generate the sample. Furthermore, a generated sample's membership may be transferred to any other cluster, but with differing probabilities. We calculate the probability of an instance transferring from a given cluster to all other clusters for each such membership transfer. If initial cluster is i and newly generated sample is transferred to cluster j, we denote this probability as $P(tr_{i,j})$. This probability will be calculated as follow:

$$P(tr_{i,j}) = \frac{|mem\_trans_{i,j}|}{|ins_i|} \tag{5}$$

In this formula, $mem\_trans_{i,j}$ is the set of instances of cluster i that transferred their membership to cluster j, and ins_i is set of all instances in cluster i.

Adversarial Sample Probability of Cluster When a generated sample is found to be transferred to a different cluster, such a sample may or may not be an adversarial sample. According to the definition, only those generated samples that can bypass the classifier are adversarial samples. Such adversarial samples may not be equally distributed among all the clusters, and hence, the probability of a cluster containing adversarial samples varies from cluster to cluster. Specifically, we calculate the probability of a generated sample getting its membership transferred to a specific cluster and being an adversarial sample at the same time. We denote the probability of an adversarial sample belonging to a cluster i as $P(Ay_i)$.

$$P(Ay_i) = \frac{|adv\_sam_i|}{|gen\_sam_i|} \tag{6}$$

In this case, adv_sam_i is the set of adversarial samples that belong to cluster i, and gen_sam_i is the number of generated samples in cluster i.

With these parameters, we are in a position to calculate the probability of generating adversarial samples based on instances chosen from a specific cluster while focusing on the membership transfer of such adversarial samples to another chosen cluster.

Probability Normalised with Cost

We calculated the average cost of each transfer between different pairs of clusters. We call cost cst_i as the cost of manipulating features for instances in cluster i. To consider this cost as well as the probability, we normalised this probability with cost in the Equation 7, which can be viewed as the *likelihood* of this transfer with a given cost from original cluster of i to a transferred cluster of j

$$\mathcal{L}(i,j) = \frac{P(c_i) \ * \ P(tr_{i,j}) * \ P(Ay_j)}{cst_{i,j}} \tag{7}$$

One cluster membership transfer of a generated adversarial sample with high probability and high cost is not desirable and will get a lower total score than a transfer with high probability and low cost. The desired transfer from an adversary perspective is one with the highest probability and lowest cost.

We visualize these probability and cost metrics in Figure 6 to show the best transfer that can be made. The X axis shows the initial cluster of phishing samples, and Y axis shows the cluster of generated adversarial samples. Darker colors show lower probability and higher cost. Lighter colors show higher probability and lower cost. In essence, the heat map shows what transfer has the highest probability of adversarial samples with the lowest cost.

For example, Figure 6(a) for DS-1 shows that if the generated sample's membership is transferred from cluster 1 to cluster 2, then there is a higher probability of this sample being an adversarial sample. In other words, this is a better choice for the adversary. Furthermore, Figure 6(b), for DS-2, shows that cluster number 5 is a vulnerable cluster and generates adversarial samples whose membership is transferred to a different cluster with low cost. A similar pattern is seen in

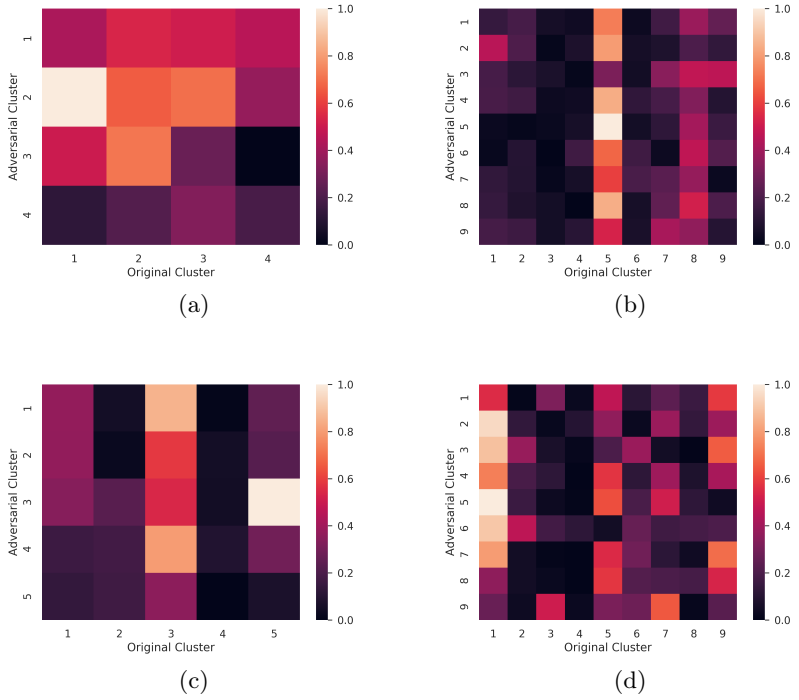


Fig. 6. The relation between an original cluster of an instance with adversarial samples: (a) describes the DS-1; (b) describes the DS-2; (c) describes the DS-3; and, (d) describes the DS-4.

Figure 6(d) for DS-4 in cluster 1. In other words, if manipulating a sample in cluster 1 transfers its membership to a different cluster, then there is a higher likelihood that this sample is adversarial. Similarly, Figure 6(c), for DS-3, shows that the most vulnerable cluster is three on average.

In this experiment, we used the previous discussion to form a probabilistic model used by both the adversary and the defender. The adversary can find the best suitable transformation among different cluster samples that generate a higher number of adversarial samples. A defender can find the specific vulnerability of the learning model and the clusters contributing to a higher number of adversarial samples, thereby enabling a specific corrective action.

6.10 Comparing the Results with Prior Research

In this section, we compare our approach with some of the previous research in this field. Table 7 compared nine different approaches in the literature. We summarized each approach's advantages and disadvantages and showed the dataset size and best accuracy results of each approach. We studied a wide range of previous efforts by focusing on machine learning techniques. Some of the techniques

Table 7. Comparisons of different approaches in the literature including our proposed approach

Author	Description	Size	ACC	Adv.
Niakanlahiji *et al.* [4]	-Scalable feature-rich framework with a series of new and existing features -Not using third-party services, Language agnostic	22.3K	95%	No
Sahinguz *et al.* [11]	-Real-time detection mechanism based on NLP of URLs, Language independent -Tested on a large dataset, Not using third-party service	73K	97%	No
Verma *et al.* [27]	-Features based on lexical-, distance-, and length-related features of the URL -Using four large datasets	115K	99.3%	No
Jiang *et al.* [5]	-Combined the URL and DNS information, Used a deep neural network with the help of NLP, Automatically extracts hidden features	7M	96%	No
Tian *et al.* [25]	-Studied five types of domain squatting, Using dataset of over 224 million registered domains, Using visual and OCR analysis, Found new phishing instances that evaded common blacklist	234M	N/A	No
Pereira *et al.* [6]	-Detecting algorithmically generated domain, Graph-based algorithm to extract the dictionaries that are being used to generate algoritmically domains	80K	99%	No
Shirazi *et al.* [7]	-Studying limitation current approaches: large number of features and bias in the datasets , Focused on the domain name, Running at the client-side -Not using third-party services	2.2 K	97-98%	No
Li *et al.* [8]	-Extract the features from both URL and HTML of the page -Not using third-party services	50K	97%	No
Bulakh *et al.* [26]	-Companies can define their phishing detection mechanism and protect the customers -Can be used as an complimentary service besides other detection approaches	1.3K	96.34%	No
Our work	-Evaluate the performance of existing datasets including [7, 21, 23, 24] -Using multiple classifiers and comparing the results	2-10K	81-95%	No
Our work	- Proposing adversarial sampling attack against the learning model, Showing the feasibility of the attack, Prove the vulnerability of current model, Modeling the vulnerability level and cost	2-10K	0%	Yes

solely focused on the URL itself [11, 25], but others look at both URL, and the content of the page [7, 26]. The use of third-party services is another difference between approaches that possess privacy risks. The previous studies have been done on variable sizes of datasets. While some of the datasets have less than 5 thousand records [7, 26], there are also datasets with millions of instances [5, 25]. Also, for approaches analyzing just the URL without the webpage content, creating massive datasets are easier. Most of the approaches achieved high accuracy of over 95%. Both [6, 27] achieved accuracy of 99%, which is significantly high. Tian *et al.* [25] found new phishing samples that were not detected by common phishing detection mechanisms even after one month. We also added the results of this study to Table 7. We trained the classifier on the four public datasets and achieved very high accuracy. When we added the manipulated features in the testing phase, the accuracy degraded significantly and finally became zero. These experiments prove that our proposed attack is sufficient to evade existing classifiers for phishing detection.

7 Conclusion and Future work

In this work, we explained the limitation of machine learning techniques when adversarial samples are considered. We introduced the notion of vulnerability level for data instances and datasets based on the adversarial attacks and quantified it. We achieved high accuracy in the absence of this attack using seven different well-studied classifiers in the literature: more than 95% for all classifiers except one that had 82%. However, when we evaluated the best-performing classifier against the adversarial samples, the classifier's performance degraded significantly. With only one feature perturbation, the TPR falls from 82-97% to

79%-45% and, increasing the number of perturbed features to four, the TPR fell to 0%, meaning that all of the phishing instances were able to bypass the classifier. Subsequently, we continued our experiments by factoring in the adversary cost. We showed that both the number of manipulated features and the total manipulation cost, which can be derived from the difference between the original phishing sample and the adversarial sample, are essential. This means that from an attacker's point of view, changing the minimum number of instances is desired, but the adversarial sample must have the minimum cost. This shows the weakness of well-known defense mechanisms against phishing attacks. To increase the success rate for adversarial sampling, we devised a clustering approach that directs the adversary towards identifying the best possible phishing samples for manipulation. We showed that our clustering approach allows an adversary to pick adversarial samples from a specific cluster and achieve a high-rate of success close to 75%. Adversarial samples transferred from the original cluster to a new cluster have a higher chance of bypassing the model. Our clustering approach allows an attacker to identify better samples and allows analysts to identify better defenses. It hints the adversary to select more efficient feature manipulation to evade the classifiers. Our future work is to develop robust learning models in the face of such organized adversarial sampling strategies. Specifically, our adversarial sampling approach gives indications of the features that are more likely to be manipulated. Defenders can focus on these features to make it infeasible to generate adversarial samples. The complex correlation between the features and the nature of phishing attacks is a topic for future exploration.

Acknowledgements

This work is supported in part by funds from NSF Awards CNS 1650573, CNS 1822118 and funding from CableLabs, Furuno Electric Company, SecureNok, Statnett, Cyber Risk Research, and AFRL. Research findings and opinions expressed are solely those of the authors and in no way reflect the opinions of the NSF or any other federal agencies.

References

1. Yanping Zhang, Yang Xiao, Kaveh Ghaboosi, Jingyuan Zhang, and Hongmei Deng. A survey of cyber crimes. *Security and Communication Networks*, 5:422–437, 2012.
2. Jerry Felix and Chris Hauck. System security: a hacker's perspective. *Interex Proceedings*, 1:6–6, 1987.
3. APWG. Phishing attack trends report - 3q 2018, 2018. [Online; accessed 24-Jan-2019].
4. Amirreza Niakanlahiji, Bei-Tseng Chu, and Ehab Al-Shaer. Phishmon: A machine learning framework for detecting phishing webpages. In *Intelligence and Security Informatics*, pages 220–225, 2018.
5. Jianguo Jiang, Jiuming Chen, Kim-Kwang Raymond Choo, Chao Liu, Kunying Liu, Min Yu, and Yongjian Wang. A deep learning based online malicious url and dns detection scheme. In *Security and Privacy in Communication Systems*, pages 438–448, 2017.

6. Mayana Pereira, Shaun Coleman, Bin Yu, Martine DeCock, and Anderson Nascimento. Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic. In *Research in Attacks, Intrusions, and Defenses*, pages 295–314, 2018.
7. Hossein Shirazi, Bruhadeshwar Bezawada, and Indrakshi Ray. "kn0w thy doma1n name": Unbiased phishing detection using domain name based features. In *Access Control Models and Technologies*, pages 69–75, 2018.
8. Yukun Li, Zhenguo Yang, Xu Chen, Huaping Yuan, and Wenyin Liu. A stacking model using url and html features for phishing webpage detection. *Future Generation Computer Systems*, 94:27–39, 2019.
9. Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *International Conference on Knowledge Discovery and Data Mining*, pages 99–108, 2004.
10. ISTR Internet Security Threat Report Volume 23. Technical report.
11. Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2019.
12. Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *ACM Workshop on Security and Artificial Intelligence*, pages 43–58, 2011.
13. Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698, 2015.
14. Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26:984–996, 2014.
15. Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable and Secure Computing*, 2017.
16. Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, pages 5120–5129, 2018.
17. Hossein Shirazi, Shashika Muramudalige, Indrakshi Ray, and Anura Jayasumana. Improved phishing detection algorithms using adversarial autoencoder synthesized data. In *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020.
18. Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. cleverhans v1. 0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 10, 2016.
19. Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *arXiv preprint arXiv:1712.03141*, 2017.
20. James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
21. Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. An assessment of features related to phishing websites using an automated technique. In *Internet Technology And Secured Transactions*, pages 492–497, 2012.
22. Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.
23. Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. Phishing detection based associative classification data mining. *Expert Systems with Applications*, 41(13):5948–5959, 2014.

24. Choon Lin Tan. Phishing dataset for machine learning: Feature evaluation, 2018.
25. Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Internet Measurement Conference*, pages 429–442, 2018.
26. Vlad Bulakh and Minaxi Gupta. Countering phishing from brands' vantage point. In *International Workshop on Security And Privacy Analytics*, pages 17–24, 2016.
27. Rakesh Verma and Keith Dyer. On the character of phishing urls: Accurate and robust statistical learning classifiers. In *Data and Application Security and Privacy*, pages 111–122, 2015.