DISSERTATION

PHISHING DETECTION USING MACHINE LEARNING

Submitted by

Hossein Shirazi

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2021

Doctoral Committee:

Advisor: Indrakshi Ray
Co-Advisor: Chuck Anderson

Yashwant K. Malaiya
Haonan Wang

ABSTRACT

PHISHING DETECTION USING MACHINE LEARNING

Our society, economy, education, critical infrastructure, and other aspects of our life have become largely dependent on cyber technology. Thus, cyber threats now endanger various aspects of our daily life. Phishing attacks, even with sophisticated detection algorithms, are still the top Internet crime by victim count in 2020. Adversaries learn from their previous attempts to (i) improve attacks and lure more victims and (ii) bypass existing detection algorithms to steal user's identities and sensitive information to increase their financial gain.

Machine learning appears to be a promising approach for phishing detection and, classification algorithms distinguish between legitimate and phishing websites. While machine learning algorithms have shown promising results, we observe multiple limitations in existing algorithms. Current algorithms do not preserve the privacy of end-users due to inquiring third-party services. There is a lack of enough phishing samples for training machine learning algorithms and, over-represented targets have a bias in existing datasets. Finally, adversarial sampling attacks degrade the performance of detection models.

We propose four sets of solutions to address the aforementioned challenges. We first propose a domain-name-based phishing detection solution that focuses solely on the domain name of websites to distinguish phishing websites from legitimate ones. This approach does not use any third-party services and preserves the privacy of end-users. We then propose a fingerprinting algorithm that consists of finding similarities (using both visual and textual characteristics) between a legitimate targeted website and a given suspicious website. This approach addresses the issue of bias towards over-represented samples in the datasets. Finally, we explore the effect of adversarial sampling attacks on phishing detection algorithms in-depth, starting with feature manipulation strategies. Results degrade the performance of the classification algorithm significantly. In the

next step, we focus on two goals of improving the performance of classification algorithms by increasing the size of used datasets and making the detection algorithm robust against adversarial sampling attacks using an adversarial autoencoder.

ACKNOWLEDGEMENTS

First, and foremost, I am extremely grateful to my supervisor, Prof. Indrakshi Ray, for her invaluable advice, continuous support, and patience during my Ph.D. study. Her immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank Dr. Chuck Anderson, Dr. Yashwant Malaiya, Dr. Haonan Wang, Dr. Indrajit Ray, Dr. Bruhadeshwar Bezawada, and Dr. Ritwik Banerjee for their technical support on my study. I would like to thank all the members of the RaysCyberResearchLab and the Department of Computer Science at Colorado State University. It is their kind help and support that have made my study and life in the U.S. a wonderful time. Finally, I would like to express my gratitude to my parents and my wife. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

DEDICATION

To Hajar and Hamoon

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction to Phishing

Our society, economy, education, critical infrastructure, and other aspects of our life have become largely dependent on cyber and information technologies. Simultaneously, cyber-attacks are becoming more attractive for adversaries [1]. The world is forecasted to spend $133.7 billion in 2022 on cyber-security [2]. 62% of businesses experienced social engineering attacks, including phishing attacks in 2018, and 68% of business leaders feel their risks related to cybercrimes are increasing [3]. However, only 5% of companies' folders on average are appropriately protected [4]. 4.1 billion data records have been breached only in the first half of 2019, 71% were financially motivated, and 25% were motivated by espionage [5]. 52% of breaches have featured hacking as an attack vector, 28% involved malwares, and 32–33% included phishing or social engineering as attack vector [6]. These numbers demonstrate the importance of cyber-attacks.

Phishing, defined as *the attempt to obtain sensitive information such as usernames, passwords, and credit card details, often for malicious reasons, by masquerading as a trustworthy entity in an electronic communication* [7], is a problem that is as old as the Internet itself. Trying to get unsuspecting users to give up their money, credentials, or privacy is a particularly insidious form of social engineering that can negatively affect people's lives.

Phishing attacks, even with sophisticated detection algorithms, are still dominant cyber-crimes. FBI's Internet Crime Complaint Center (IC3) reports phishing (including other similar types of attacks like vishing, smishing, and pharming) to be the most prevalent crime type by number in 2019 with an estimated 12.5 billion USD in financial losses worldwide between 2013-2018 [8, 9]. Adversaries learn from their previous attempts to (i) improve attacks and lure more victims and (ii) bypass existing detecting algorithms to steal user's identity and sensitive information [10, 11] to increase their financial gain.

1.1 Limitations of current machine learning-based solutions

Social engineering attacks in general, and phishing attacks specifically, are not successful because of the vulnerability in systems, but due to misjudgment of humans in distinguishing legitimate entities from fake ones. Consequently, a wide range of techniques has been studied in the literature to counter such attacks having different levels of sophistication.

Machine learning aims at automating the learning processes from existing examples and experiences without being explicitly programmed [12]. Machine learning algorithms have shown promising results [13–16]. This technique requires prior real-world data that has been classified or marked to carry out the training [17]. However, we have faced the following limitations using machine learning-based techniques to detect phishing websites in existing approaches.

1.1.1 Not privacy preserving

Supervised deep learning appears to be a promising approach for phishing detection [18–20]. Machine learning requires a large volume of training data, such data extraction violates the privacy of end-users. For extracting feature values of suspicious websites, third-party services like a search engine has to be used. This reveals the browsing history of end-users and violates privacy.

1.1.2 Inadequate phishing samples

The next limitation is the lack of having real attack data or inadequate data samples. In cyber-security systems, threats are rare events, so datasets are biased towards normal events. Such datasets have much more normal instances than minority events and learning from a biased dataset is challenging. The interest arises in the minority samples where rare instances belong to phishing attacks. Besides, researchers barely share their datasets in cyber-security problems for reasons of confidentiality and privacy. Only 10% of researchers shared their dataset in a similar security networking problem [21]. That makes creating a ground truth dataset impossible.

With a low volume of existing phishing datasets [22], the learning classifier may not converge, and the performance will be inconsistent. In short, the training model may be imperfect in the absence of adequate data.

1.1.3 Over-represented targets

Zou and Schiebinger [23] revealed multiple examples of Artificial Intelligence (AI) where some of the samples are over-represented, and others are under-represented. These biased datasets will lead to a learning model that is racist, sexist, or unfair to the group of minorities. For example, a medical machine learning algorithm that was trained for skin cancer from photographs was not tested on dark-skinned people due to the lack of enough samples of that group, only 5% of images were of dark-skinned individuals in the dataset [23].

Our observation here is if the phishing samples in a training dataset are biased towards the most targeted websites, the detection rate of phishing instances for the more popular sites would be higher than groups with fewer numbers. Although the overall results for the detection algorithm are high, the detection results vary widely among the different target websites depending on their popularity. In this situation, the algorithm is biased towards *pro-big-techs*, meaning it is more successful for detecting attacks against popular websites compared to relatively unknown ones.

1.1.4 Adversarial attacks

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. One crucial assumption in using machine learning approaches is that the training data collection process is independent of the attackers' actions [24]. However, in adversarial contexts, *e.g.* phishing, 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.

1.2 Our proposed solutions

For the aforementioned challenges, we proposed a set of solutions. Each solution addresses one or more challenges.

1.2.1 Privacy preserving phishing detection using domain-name based features

In this proposed approach, we check whether a given suspicious website is phishing or not by considering only the domain-name of the website. Typically, the content of a phishing website is textually and visually similar to some legitimate website. Based on this, the problem statement we examine is, to determine the features that quantify the attacker strategies in terms of the content found in the phishing website. Our approach is based on the intuition that the domain name of the phishing websites is a key indicator of a phishing attack. We design several features that are based solely on the domain name and train a machine learning classifier based on sample data. The trained classifier is used to test a suspicious website against these features.

The primary challenge is to justify the use of domain name-based features. A phisher has much control over the formation and structure of the URL and therefore, can generate noisy URLs that can bypass most machine learning approaches. On the other hand, the phisher has limited control over the domain name, *i.e.*, the adversary can generate several types of URLs within the same domain, but the domain name remains fixed throughout. Second, domain name-based features are likely to be more independent of the content in the phishing pages. The structure of the page layout, the HTML tags, and the dynamic content will no longer be a major part of the detection algorithm. Third, a phishing domain name typically can contain additional characters or numbers to give the

illusion of a legitimate website, *e.g.*, *go0ogle.com*. These variations are subtle and are likely to provide sufficient statistical distinctions between legitimate and phishing websites. Hence, based on these arguments, we claim that domain name-based features are likely to exhibit more regularity than URL-based features.

The penultimate challenge concerns the validity of the features. We performed a statistical validation against a small sample of the data to verify the utility of the features across phishing and legitimate websites. We were able to eliminate several features and our final classifier consists of only seven features.

The final challenge is testing the resiliency of the domain-based features to detect unknown or zero-day phishing attacks. To address this, we tested the classifier against a blacklist of URLs taken from the latest updates on `OpenPhish.com`.

In this approach, we are not using any third-party services and feature extraction so the privacy of users is preserved and addresses the challenge mentioned earlier. In Chapter 3, we explain the details of our approach.

1.2.2 Fingerprinting-based phishing detection

Current approaches [25], [26], [27], and [28] perform an in-depth analysis to find characteristics that are common across phishing websites but help distinguish them from genuine ones. These characteristics form the basis of features that are used by machine learning algorithms. This approach appears counter-intuitive as adversaries use different techniques to make a phishing website similar to a genuine target website, not other phishing instances. The choice of features and their representation often depends on the skill of the model designer and the types of attacks that can be detected by the algorithm. Adversaries are always looking for alternate attack vectors to bypass current learning models and make existing features obsolete. As new attacks emerge, the current models must be upgraded.

Our proposed approach consists of finding similarities between a legitimate website that is targeted and all of the phishing websites that mimic the legitimate website. We define features to

compare a phishing sample to a target website. We propose the idea of fingerprinting a legitimate website using its visual and textual characteristics which will uniquely represent it. We also suggest using screenshots of websites instead of relying on the HTML code of websites; this makes bypassing the learning model extremely hard for the attacker. The fingerprint will be compared with the given samples to detect phishing instances.

Each machine learning vector will represent the similarity of a phishing website to a specific target, not the similarity to other phishing instances. In this case, the machine learning algorithm will not answer the critical question of phishing detection as *"if the given website is phishing or not"* but it will answer *"if a given website is attacking a specific target or not"*. The learning algorithm in this model improves learning scores based on the similarity of a phishing instance to the target website and do not depend on other samples.

Thus, the model will not skew towards targets with more individuals, as each target has its own learning model and dataset. Accordingly, each site is being judged independently and, it guarantees there is not any bias toward groups of sites with a high volume of samples, and oversampling and undersampling cannot affect the learning scores.

The model is looking for the visual and textual similarity between a given suspicious website and a targeted genuine website. Thus, new attack vectors will not change the learning model, so there is no need to update the model over time unless the target website has been changed.

This approach addresses the issue of bias towards over-represented samples. Chapter 4 explains details of our approach.

1.2.3 Adversarial sampling attacks

In this proposed approach, we first 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.

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.

In the next step, we focus on two goals of improving the performance of classification algorithms by increasing the size of the dataset and making the detection algorithm robust against adversarial sampling attacks. Regarding the first goal, we propose a deep-learning approach to synthesize new samples that preserve the characteristics of existing data but without doing actual data collection. These samples will be added to the training datasets. Such an approach is essential when data is unavailable, or the collection process is laborious and infeasible. In addition, we use new synthesized instances as an adversarial attack against the detection model and achieve the second goal. We leverage the datasets with synthesized samples to make them prone to such attacks.

We develop an adversarial autoencoder (AAE) network to mimic websites that are in tune with the capabilities and characteristics of actual attackers as phishing samples as well as synthesizing new legitimate samples. Our proposed AAE has been used to extend both phishing and legitimate instances in the dataset and is compatible with the adversary we modeled. We inspect the similarity between synthesized samples via AAE and original samples to guarantee the synthesized samples follow the same characteristics as the original ones. That proves the validity of our synthesized samples.

Chapter 5, we first show the details of adversarial attacks against phishing detection algorithms. In Chapter 6, we address two issues of the low volume of data and adversarial attacks in phishing detection.

1.3 Key Contribution

Our key contribution in this dissertation as follows:

- In Chapter 3, we describe a machine learning-based approach for phishing detection that relies entirely on domain name-based features and preserves the privacy of end-users.

- Our approach achieves a 97% accuracy on a set of 2000 URLs with five-fold cross-validation. In addition, our approach achieves a 97-99.7% detection rate on live blacklist data from *OpenPhish.com*.

- In Chapter 4, we define a new fingerprinting approach based on the visual and textual traits of legitimate websites. Our algorithm is able to detect whether a given suspicious website attacks a specific target. We implemented our approach on 14 legitimate websites and tested against 1446 unique samples. Our model reported an accuracy of at least 98% and it is not biased towards any website. This is in contrast to the current machine learning models that may be biased towards groups of over-represented samples and lead to more false-negative errors for less popular websites.

- In Chapter 5, 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. Our experiments reveal that the phishing detection mechanisms are vulnerable to adversarial learning techniques. Specifically, the identification rate for phishing websites dropped to 70% by manipulating a single feature. When four features are manipulated, the identification rate dropped to zero percent. This result means that any phishing sample, which would have been detected correctly by a classifier model, can bypass the classifier by changing at most four feature values

- We define phishing instances' vulnerability level, which quantifies and optimizes the attackers' efforts to generate adversarial samples. In addition, 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.

- In Chapter 6, we present an AAE model to synthesize phishing and legitimate data that mimic original ones to augment the training dataset.

- We quantify the improvement of the accuracy of models by using synthesized data. We also discuss how to design robust classifiers using synthetic data that is resistant to adversarial attacks. We exemplify the widespread applicability of our approach for a range of datasets and different classification algorithms.

1.4 Dissertation organization

In Chapter 2 of this dissertation, we first study the phishing detection background by giving a classification framework to compare existing approaches. In Chapter 3, we define machine learning algorithms to classify between legitimate and phishing instances based on the domain name. In Chapter 4, we propose a model to calculate the visioned similarity between a phishing website and the target. We then study the vulnerability of existing models against adversarial sampling attacks in Chapter 5. We synthesize new phishing instances through a feature manipulation process. These samples bypass the existing learning model demonstrating the current phishing detection model's vulnerability. Finally, in Chapter 6, we propose an Adversarial Auto-Encoder (AAE) algorithm that synthesizes new phishing and legitimate adversarial samples that mimic real samples. We evaluated whether these new synthesized samples can bypass the existing model or not. Chapter 7 summarises our findings and concludes the dissertation.

Chapter 2

Related Work

Phishing attacks are classified as social engineering attacks. In this kind of attack, the adversary does not necessarily look for a vulnerability in the system but looks for unaware users to lure them. For example, an attacker creates a web page similar to a login page of a well-known email provider, sends the links to the users, and asks them to log in. In this example, there is not any security concern related to the email provider. If the end-user is not aware of the potential threats, they may be fooled by the attacker. During the last decade, different researchers tried to come up with different approaches. From a broader perspective, we categorize all these efforts into two major categories. In the first category, we discuss the approaches that try to address the problem in a human-based manner. The approaches in this category increase the knowledge of end-users and help them to make good decisions when they face suspicious websites. In the second category, we study software-based approaches. In this approach, different techniques aim to distinguish between legitimate websites and phishing ones. The result of this category may also be fed to the first category to help end-users.

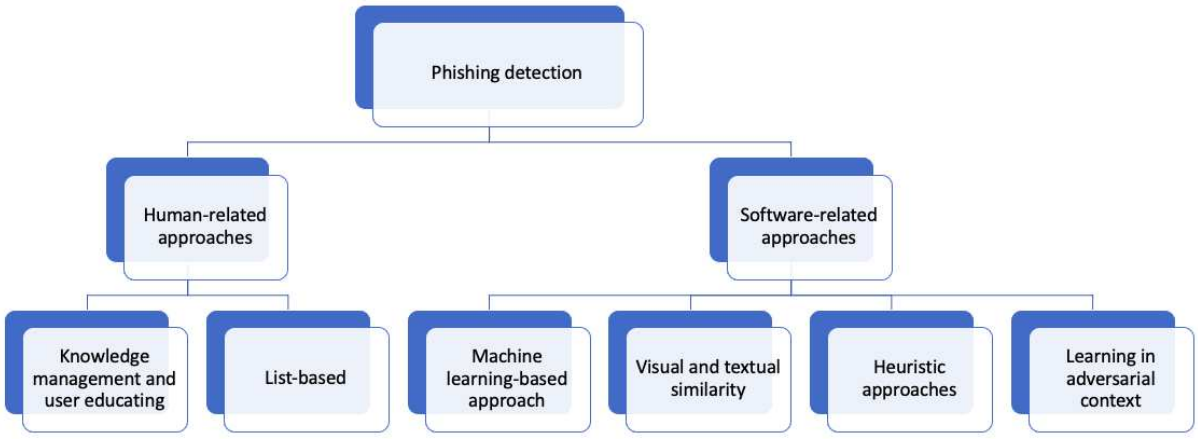


Figure 2.1: Phishing detection approaches in the literature. Human related approaches focus on the role of human and software related approaches focus to improve software-based solutions.

2.1 Human-related approaches

The strategy of phishing attackers is based on taking advantage of unaware or inexperienced users. The users who do not know about these attacks are in more danger. Figure 2.1 shows existing phishing detection algorithms in the literature. Knowledge management helps to increase user's information about the attacks and educate them when faced with it, however, the list-based approach shows a warning to prevent the user from being fooled by the phishing website.

2.1.1 Knowledge management and user educating

The users are the ones who are at risk so it is beneficial to educate them and increase their ability to protect themselves against these attacks. Jensen *et al.* [29] explored how an organization can utilize its employees to combat phishing attacks collectively through coordinating their activities to create a human firewall. They utilize knowledge management research on knowledge sharing to guide the design of an experiment that explores a central reporting and dissemination platform for phishing attacks. Results demonstrate that knowledge management techniques are transferable to organizational security which can benefit from insights gained from combating phishing. Specifically, they highlight the need to both publicly acknowledge the contribution to a knowledge management system and provide validation of the contribution by the security team. They reported that doing only one or the other does not improve outcomes for correct phishing reports.

Sheng *et al.* [30] design an online game that teaches users good habits to help them avoid phishing attacks and use learning science principles to design and iteratively refine the game. The participants were tested on their ability to detect phishing websites from the legitimate ones before and after playing the designed game and reading an article about phishing. The results show that playing the game can increase the ability to find the phishing website of the participant. Asanka *et al.* [31] create a mobile version of a game aimed to enhance avoidance behavior through the motivation of home computer users to protect against phishing threats [32].

To explore the effectiveness of embedded training, researchers conducted a large-scale experiment that tracked workers' reactions to a series of carefully crafted spear-phishing emails and

a variety of immediate training and awareness activities [33]. Based on behavioral science findings, the experiment included four different training conditions, each of which used a different type of message framing. The results from three trials showed that framing had no significant effect on the likelihood that a participant would click a subsequent spear-phishing email and that many participants either clicked all links or none regardless of whether they received training. The study was unable to determine whether the embedded training materials created framing changes in susceptibility to spear-phishing attacks because employees failed to read the training materials.

2.1.2 List-based

List-based solutions have fast access time, but they suffer from a low detection rate especially for the zero-day attacks, which exploit potentially serious software security weaknesses that the vendors or developers may be unaware of. Afroz *et al.* [34] build profiles of trusted websites based on fuzzy hashing techniques. This approach combines white-listing with black-listing and heuristic approaches to warn users of attacks. Jain *et al.* [35] used an auto-updated white-list of legitimate sites accessed by the individual user. When users try to open a website, which is not available in the white-list, the browser warns users not to disclose their sensitive information. However, all list-based approaches suffer from the problem of dynamic updates and scalability, which makes them impractical for client-side detection.

Modern browsers use a list-based approach in an embedded manner and update the list regularly. The browser checks every single website that users want to visit against that list and if the webpage is listed there, gives a warning to the user. Figure 2.2 shows an example of that warning shown to the users by Google Chrome and Microsoft Edge.

Firefox checks each website that a user visits against reported phishing, unwanted software, and malware lists. These lists are automatically downloaded and updated every 30 minutes by default when the "Phishing and Malware Protection" feature is enabled [36].

Microsoft SmartScreen, used in Windows 10 and both Internet Explorer 11 and Microsoft Edge, helps to defend against phishing by performing reputation checks on visited sites and block-

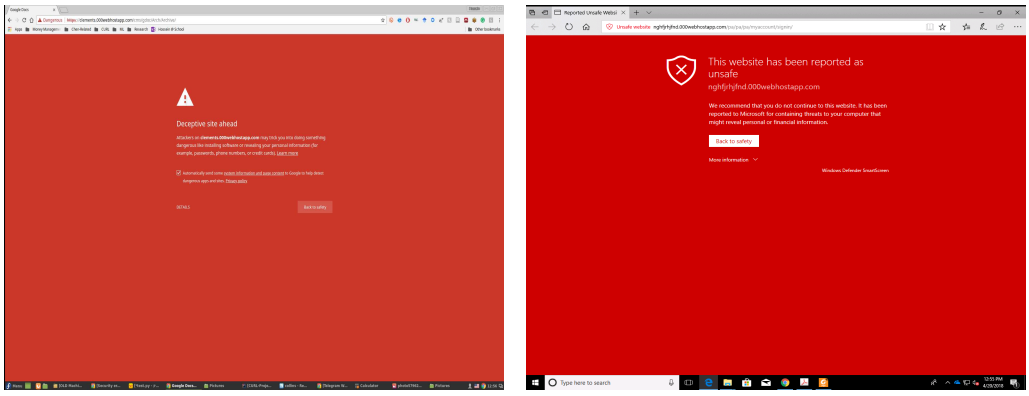


Figure 2.2: Warning of phishing attacks in two different browsers; Left: Google Chrome - Right: Microsoft Edge

ing any sites that are thought to be phishing sites. SmartScreen also helps to defend people against being tricked into installing malicious applications. Google's Safe Browsing infrastructure displays warning messages in Google Chrome, Android, and Gmail if the user tries to access a potentially malicious site or download malware and viruses [37].

2.2 Software-related approaches

Relying solely on the end-user in the fight against phishing attacks is inadequate. The end-users are prone to make incorrect decisions, even with education and awareness. Addressing this problem needs the help of software-related techniques to prevent, detect and mitigate phishing. In this section, we will discuss different software-related techniques to fight against them, namely, visual and textual similarity, machine learning, heuristics, and learning in adversarial contexts.

2.2.1 Machine learning-based approach

Machine learning algorithms have been proven to have the ability to discover complex correlations among different data items of similar nature. Many algorithms consist of two steps: *learning* and *testing*. In the *learning* step, the algorithms try to learn from supporting examples, and in the *testing* phase, the researchers evaluate the accuracy of the algorithms.

Attackers often use email to send out phishing URLs to the victim. Consequently, detecting potentially dangerous emails helps to protect users from phishing websites. There is a wide literature on automating detection for phishing emails by looking at the context of the email. For example, Basnet *et al.* [38] used 16 features to detect phishing emails. While they use email messages as a source to extract the features, we only focus on the website itself rather than how the attacker tries to tempt the users.

Ma *et al.* [39] described an approach based on URL classification using statistical methods to discover the lexical and host-based properties of malicious website URLs. They use lexical properties of URLs and registration, hosting, and geographical information of the corresponding hosts to classify malicious web pages at a larger scale. These methods are able to learn highly predictive models by extracting and automatically analyzing tens of thousands of features potentially indicative of suspicious URLs. The resulting classifiers obtain 95-99% accuracy, detecting large numbers of malicious websites by just using their URLs. However, their approach requires a large feature set and extracts host information with the help of third-party servers. In Section 2.3, we discussed why using URL-based features and third-party services leads to a biased dataset.

Miyamoto *et al.* [40] provided an overview of nine different machine learning techniques, including Support Vector Machine, Random Forests, Neural Networks, AdaBoost, Naive Bayes, and Bayesian Additive Regression Trees. They analyzed the accuracy of each classifier on the CANTINA dataset [41], a state of the art dataset, and achieved a maximum accuracy of 91.34% using AdaBoost. They used a wide range of classifiers but due to the adaptive nature of these attacks and not having the capability of updating training dataset, they cannot guarantee the resiliency of the solution.

Aburrous *et al.* [42] proposed association data mining algorithms to characterize and identify the rules to classify phishing websites. They implemented six different classification algorithms and techniques to mine the phishing training datasets. They used a phishing case study that was applied to illustrate the website phishing process. The rules generated from their associative classification model showed the relationship between some important characteristics like URL and

domain identity, security, and encryption criteria. The experimental results demonstrated the feasibility of using Associative Classification techniques in real applications and its better performance as compared to other traditional classifications algorithms, *e.g.* Multi-class Classification based on Association Rule algorithm which has an error rate of Rate 12.6%

Xiang *et al.* [43] proposed a layered anti-phishing solution with a rich set of features. They proposed 15 features that exploit the Document Object Model (DOM) of webpages including using search engine capabilities, and third-party services, with machine learning techniques to detect phishing attacks. Also, they designed two filters to help reduce False Positive Rate (FPR) and achieve runtime speedup. The first is a near-duplicate phishing detector that uses hashing to catch highly similar a fake website. The second is a login form filter, which directly classifies web pages with no identified login form as legitimate. The key shortcoming of this approach is that the experiments were conducted with biased datasets. The `Alexa.com` website, which provided most of the legitimate websites in this dataset, only gives the domain name of legitimate websites. While the phishing websites are taken from `PhishTank.com` are mostly complete URLs of phishing web pages. So the types of data instances are different. Also, using third-party services to extract some features may endanger the privacy of users by revealing their browsing history.

In 2015, Verma *et al.* [44] described an approach based on textual similarity and frequency distribution of text characters in URLs. For instance, they examined the character frequencies in phishing URLs and the presence of suspicious words as features. However, this approach is entirely based on URLs and is likely to be biased in the modern-day context. Some of their features, like *presence of suspicious words*, will need to be updated frequently as newer phishing attack surfaces emerge.

Jain *et al.* [45] described a machine learning-based approach that extracts the features from the client-side only. Their approach examined the various attributes of phishing and legitimate websites in-depth and identified 19 features to distinguish phishing websites from legitimate ones. Their approach has a relatively high accuracy in the detection of phishing websites as it achieved a 99.39% true positive rate and 99.09% of overall detection accuracy. While their approach relied

only on the client-side feature and did not use any third-party features, there are some drawbacks to this approach. For example, their method of dataset creation is flawed. For phishing websites, they used `PhishTank.com` as a source of phishing websites. For legitimate websites, they used mostly `Alexa.com`, which ranks the most top-ranked domain names in the world. While `PhishTank.com` generated the phishing pages, `Alexa.com` gives only domain names and not the internal pages of the domain. As a result, their features are biased with respect to the dataset. This factor was not considered in the feature extraction process. For example, one feature in their approach is the number of dots in the given URL. In the training phase, while all given legitimate instances consist of only domain names, the phishing instances consist of entire URLs. Another feature looks for suspicious words in the URL, but many legitimate websites also have these words.

Al-Janabi *et al.* [46] described a supervised machine learning classification model to detect the distribution of malicious content in online social networks (OSNs). Multisource features have been used to detect social network posts that contain malicious URLs. These URLs direct users to websites that contain malicious content, drive-by download attacks, phishing, spam, and scams. For the data collection stage, the Twitter streaming API was used. They just focused only on one OSN network (Twitter) and applied their approach. Their features can neither be extracted locally nor guarantee the security of users outside of the network during regular browsing.

Marchal *et al.* [47,48] proposed a client-side detection approach complete it with a browser extension [49] using custom datasets from *Intel Security* and tried to eliminate bias in datasets. They developed a target identification component that can identify the target website that a phishing web page is attempting to mimic. However, their approach uses over $200+$ features for classification, which complicates the feature extraction part in comparison with approaches with a fewer number of features. Moreover, not much is known about the exact design of their features and the dataset used is not available to replicate their results.

Rao *et al.* [50] proposed a classification model based on an ensemble of features that are extracted from URL, source code, and third-party services. Their approach is inefficient and suffers

from the same problems as other techniques using URL-based features. Furthermore, this approach uses third-party servers and reveals a user's browsing history to untrusted servers.

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

Niakanlahiji *et al.* [13] 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. The authors reported an accuracy of 95% on their datasets.

According to a Symantec report [51], 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, adding sensitive keywords like "login", "secure", "https", etc. Sahinguz *et al.* [14] proposed a real-time detection mechanism based on Natural Language Processing (NLP) of URLs. The technique used a large dataset without requiring third-party services and focused on features derived from URL obfuscation and achieved an accuracy of 95%.

Verma *et al.* [52] defined lexical, distance, and length-related features for the detection of phishing URLs. They employed the two-sample Kolmogorov-Smirnov statistical test along with other features to detect phishing websites. They conducted a series of experiments on four large proprietary datasets and reported an accuracy of 99.3% with a false positive rate of less than 0.4%.

Jiang *et al.* [53] merged information from DNS and the URL to develop a Deep Neural Network (DNN) with the help of NLP to detect phishing attacks. While other approaches need to specify

features explicitly, this method extracts hidden features automatically. The approach relies on the information from DNS and, thus, requires third-party services.

Attackers use Domain Generation Algorithms (DGA) to dynamically generate a large number of random domain names for adversarial purposes, including phishing attacks. Pereira *et al.* [54] introduced an approach for detecting such domains. These domains are considered legitimate for detection mechanisms and human analysis. The authors used a graph-based algorithm to extract the dictionaries that have been used by attackers to detect malicious domains.

While these proposed approaches are promising, they often do not consider the page content. Attackers have full control over the URL and thus, they can create any URL to bypass the classifier. Also, the content of the website is the most critical factor in luring the end-users rather than the URL or domain name themselves. Therefore, any solution not considering the website content would not be useful in the real world.

Tian *et al.* [55] studied five types of domain squatting; the practice of actors registering and using domains that impersonate companies, organizations, brands, or even people without having the right to do so. The authors studied a large DNS dataset of over 224 million registered domains. They identified 657 thousand domains that potentially targeted 702 popular websites. Using visual and Optical Character Recognition (OCR) analysis, they created a highly accurate classifier and found more than one thousand new phishing instances of which 90% of them successfully evaded well-known blacklists even after one month. The authors combined two powerful techniques: domain squatting and OCR analyses on a large dataset. The advantage of this approach is in finding new instances that evaded the current classifiers. However, there is a significant cost in keeping this information current.

Recently, Li *et al.* [56] proposed an approach to extract the features from both URL and web page content and ran multiple machine learning techniques, including GBDT, XGBoost, and LightGBM, in multiple layers, referred to as stacking approaches. The URL-based feature set includes eight features in total such as *using IP address*, *suspicious symbols*, *sensitive vocabulary*. The HTML-based category includes features like *Alarm Window*, *Login Form*, *Length of HTML Con-*

tent. The dataset has 20 features in total. 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 [57].

2.2.2 Visual and textual similarity

Since over 90% of users rely on the website appearance to verify its authenticity [58], the adversaries try to create the visual appearance of phishing websites nearly identical to that of legitimate ones. Consequently, the researchers try to use the similarity between websites as a key feature to discriminate between legitimate and phishing websites. Some approaches use visual similarity between websites while others use textual similarity.

Chen *et al.* [59] proposed an approach for detecting visual similarity between two web pages. They tested their system using the most popular web pages to examine its real-world applicability. Accuracy in the case of true positive and false positive rates reached 100 and 80 percent, respectively.

Fu *et al.* [60] used Earth Mover's Distance (EMD) to measure webpage visual similarity. They first converted the involved web pages into low-resolution images and then used color and coordinate features to represent the image signatures. Then they used EMD to calculate the signature distances of the images of the web pages. They employed an EMD threshold vector for classifying a web page as a phishing or a normal one. Also, they built up a real system that is already used online and it has caught many real phishing cases.

Routhu Srinivasa Rao *et al.* [61] proposed a combination of white list and visual similarity-based techniques. They used a computer vision technique called SURF detector to extract discriminative key point features from both suspicious and targeted websites followed by computing similarity degree between the legitimate and suspicious pages.

All these approaches need a target website to compare the similarity between two web pages and detect one of them as phishing. Zhang *et al.* [62] created a framework using a Bayesian

approach for content-based phishing web page detection. The model takes into account textual and visual contents to measure the similarity between the protected web page and suspicious web pages. A text classifier, an image classifier, and an algorithm fusing the results from classifiers are described. But, this process is expensive and often results in false positives.

Recently, there has been a rise in *extreme* phishing attacks [63, 64], a form of fine-grained content mimicking phishing, on financial institutions where the phishing website mimics the legitimate website to an alarming degree. Typically, these websites are meant to defeat visual and textual similarity analysis. The high level of noise introduced in such websites is likely to defeat most content-based machine learning approaches in the past.

2.2.3 Heuristic approaches

Neil *et al.* [65] implemented SpoofGuard, a plugin for Internet Explorer, that detects phishing attempts on the client side. It assigns weights to different anomalies found in the HTML page of websites and assigns a score. If the assigned score crosses a certain threshold, it will label the website as a phishing website and send a warning to the user. This tool runs on the client-side and can detect phishing websites based on those anomalies on the page.

Cui *et al.* [66] tried to find similarities between different attacks during a 10-month study by monitoring around 19000 websites. The study showed that 90% of phishing websites have a similar DOM structure and over 90% of these attacks were actually replicas or variations of other attacks in the database.

Bulakh *et al.* [67] use a different approach to detect a phishing website. They proposed an approach where each branded company can define its phishing detection mechanism and protect their customers. Phishing website may link their materials to the spoofed website and interact with it *e.g.* using the images or scripts, or links on spoofed pages directly from the targeted pages. They created the dataset based on those features and achieved an accuracy of 96.34% and a false positive rate of 3.39% with the Random Forest (RF) algorithm. While this can be used as an

excellent complimentary service besides other detection approaches, especially by highly targeted websites to protect their customers, it suffers from a lack of generality.

Han *et al.* [68] studied the entire life cycle of phishing campaigns in the wild. The previous researchers have studied the phishing kits after anti-phishing services had detected them, and the researchers did not observe the real way that victims interact with phishing kits, apparently because of ethical reasons. In this study, the authors presented a sandbox that protects the privacy of the victims thoroughly to address those two dilemmas. They draw the first comprehensive picture of the phishing attack with precise timing.

Anti-Phishing Working Group (APWG) reports in the third quarter of 2019, more than two-thirds of all phishing websites were using SSL certificates [69], the highest rate since 2015 when they started tracking this parameter. However, it has become clear that the usage of the HTTPS protocol alone is not a credible sign of a secure website anymore. This is being called the *HTTPS paradox* [70]. There are few researchers in the literature to detect malicious SSL certificates. Drury *et al.* [70] used SSL certificate meta-data and used machine learning algorithms to discriminate benign websites and phishing ones, but they were unsuccessful. Torroledo *et al.* [71] defined and extracted features from SSL certificates and used a deep neural network. Their results showed an accuracy of 88.6% for phishing websites, which is significant, but this method does consider SSL packet fields and traffic flow, so it cannot detect encrypted malicious streams [72].

Cui *et al.* [66] monitored more than 19000 phishing attacks for ten months and found over 90% of attacks were a replication or variation of other attacks in the database. In a subsequent work, Cui *et al.* [73] have done a more in-depth analysis.

Ho *et al.* [8] created a large-scale dataset of emails from 92 enterprise organizations and created a detection algorithm to discover spear-phishing emails. The model found hundreds of real spear-phishing emails with a very low false-alarm rate: four per every one million. Gutierrez *et al.* [74] observed that current machine learning-based detection algorithms are vulnerable to structural or semantic change in the message. They implemented machine learning on a large corpus of phishing and legitimate emails and employed under-sampling boost algorithms to handle the class imbalance

problem of phishing datasets. But they did not study the problem of the imbalanced datasets in phishing datasets.

Van Der Heijden *et al.* [75] developed an automated and fully quantitative method based on machine learning and econometrics to measure cognitive vulnerability triggers in a phishing email to predict the degree of success of an attack. Instead of selecting the best features from a machine learning point of view, this study is based on the human cognitive method. The study shows how adversaries convince end-users to give up their sensitive information. These detected metrics can improve learning algorithms and help response teams to prioritize their effort in case of a real attack.

Marchal *et al.* [76] focused on detecting phishing domains and created a proactive mechanism instead of reactive approaches like blacklisting. The second-level domain of the URL and a Markov chain have been used to detect suspicious domain names. They leveraged natural language modeling to create a blacklist based on phishing-related vocabulary.

2.2.4 Learning in adversarial context

The proposed defense mechanisms 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 an attack where an adversary generates a phishing data sample that appears to the phishing detection classifier as a legitimate data sample and thereby, avoids 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 [77], 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 in other application contexts like image recognition. In the following, we briefly explore these efforts.

Dalvi *et al.* [24] 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. This assumption, however, does not hold in many situations. For example, in the phishing detection system, the adversary does not know the training set or the actual classification algorithm used. The attackers may directly or indirectly target the vulnerabilities in the feature selection procedure. Although the attackers might target the trained classification system, it still is an indirect attack on the chosen features.

Xiao *et al.* [78] explored the vulnerabilities of feature selection algorithms under adversarial sampling attacks. They extended a previous framework [79] 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.* [80] investigated a defense method that can improve the security of linear classifiers 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.* [81] theoretically guaranteed the robustness of the k-nearest neighbors 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.

Finally, there are some tools for benchmarking and standardizing the performance of machine learning classifiers against adversarial attacks in the literature. *Cleverhans* [82] is an open-source library that provides an implementation of adversarial sample construction techniques and adversarial training for image datasets. Given the lack of such benchmarking tools for the phishing problem, we tested our approach with our own attack strategies and implementation.

2.3 Limitations of past work

The studies in the existing literature emphasize feature definition or enhancing the statistical learning models to discriminate between phishing and legitimate websites. The state-of-the-art

solutions for phishing detection [13, 53, 54, 56, 57] use engineered features based on observations made by the research experts in this domain on publicly available datasets. One crucial assumption, in existing machine learning approaches, is that the training data collection process is independent of the attackers' actions [24]. However, in adversarial contexts, *e.g.* phishing or spam filtering, this is far from reality as attackers either generate noisy data samples or generate new attack samples by manipulating features of existing ones. Furthermore, the manipulation of 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 the choice of 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.

Current content-based approaches [25–28,57] performed an in-depth analysis of the contents of the phishing websites to extract similarities among them and discriminate them from genuine ones. This analysis has been used to render sets of features and create training datasets to be used by machine learning algorithms. This approach appears counter-intuitive as adversaries use different techniques to make a phishing website similar to a genuine target website, not other phishing instances. The same argument holds for the set of phishing websites. In addition, the strength of a detection model is based on the expertness and aptitudes of the model designer; for example, how many current attack vectors have been rendered as machine learning features into the dataset. Adversaries are continuously looking for alternate attack vectors to bypass current learning models and make features obsolete. The current models are vulnerable to these new coming attacks and, thus, need to be updated for new attacks.

In addition, another major problem in detecting phishing attacks is the adaptive nature of strategies used by the phishers. Generating a phishing website has not only become trivial but also the attackers are able to bypass most defense strategies with relative ease. For instance, the evolution of *extreme phishing*, a complex form of phishing that targets the identity of users shows the severity and intensity of phishing attacks. Phishers are constantly improving phishing toolkits to generate websites that can evade nearly all forms of defenses available. Therefore, there is a need for developing phishing detection approaches that demonstrate robustness and resiliency against the adaptive strategies being used by the phishers.

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.

Chapter 3

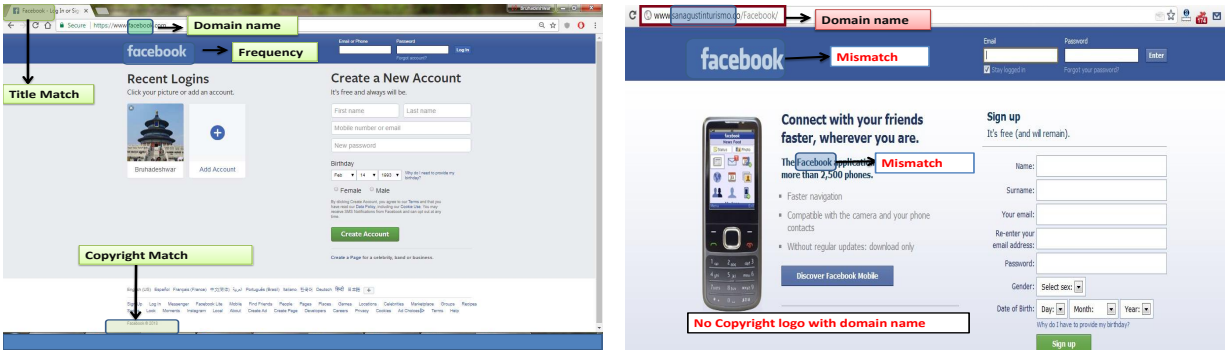
Privacy-preserving Phishing Detection using Domain-Name Based Features

3.1 Domain-name-based features

We focus on the general problem of determining if a target website is a phishing website or not, based on the standard definitions of a phishing website from literature [83, 84]. Typically, the content of a phishing website is textually and visually similar to some legitimate website. Based on this, the problem statement we examine is, to determine the features that quantify the attacker strategies in terms of the content found in the phishing website. Such features will be used to train a machine learning model to classify between phishing and legitimate websites.

The URL-based approaches [39, 44, 46, 85] analyze various features based on the target URL such as length of the URL, page rank of the URL, number of dots in the URL, presence of special characters, hostname features like IP address, domain age, DNS properties, and geographic properties, among other features. While the intuition in these approaches is sound, *i.e.*, the URL is a good indicator of phishing attacks, the structural changes of modern-day URLs negate several lexical features identified by these approaches. For instance, these days, the URLs generated by websites like Google and Amazon, are long and contain many non-alphabetic characters, which dilute the lexical similarity of legitimate URLs. For this reason, the URL-based approaches inadvertently tend to be biased towards the datasets being used and are likely to be ineffective in the future. A few hybrid detection mechanisms [86, 87] combine content and URL features, but suffer from the same problems.

In Figure 3.1, we demonstrate some of the distinguishing domain-name-based features of legitimate and phishing websites.



(a) Legitimate site

(b) Phishing site

Figure 3.1: Domain-name features for legitimate and phishing websites. (a) depicts a legitimate Facebook website, and (b) shows a phishing website that targets Facebook.

3.1.1 Feature engineering and validation

As far as possible, our feature design attempts to be content-agnostic, *i.e.*, the feature design attempts to model the principles of phishing attacks and reduce the dependence of the features on specific data values. Our feature set consists of two types of features: binary, *i.e.*, the feature value is 0 or 1, and non-binary, *i.e.*, the feature is real-valued. In summary, the key principle of our feature engineering is that all features depend on the domain-name of the website and the relationships, visual and statistical, of the domain-name with the content of the website. These aspects ensure that our features are not affected by biased datasets and are robust to noise.

To validate the intuition behind each feature, we tested the empirical cumulative distribution function (ECDF) of the feature for 1000 phishing websites against 1000 legitimate websites. We show sample ECDF plots for a few features. We also indicate if the features are "New", meaning designed by us, or "Existing", meaning that other researchers have designed it.

3.1.2 Non-binary features

We defined the following non-binary features in our proposed approach.

Feature 1 (New): Domain Length. The attackers who want to register a domain for phishing have to choose a longer domain-name in comparison with the legitimate website. The length of the domain-name is the number of characters in the domain-name string. As shown in Figure 3.2

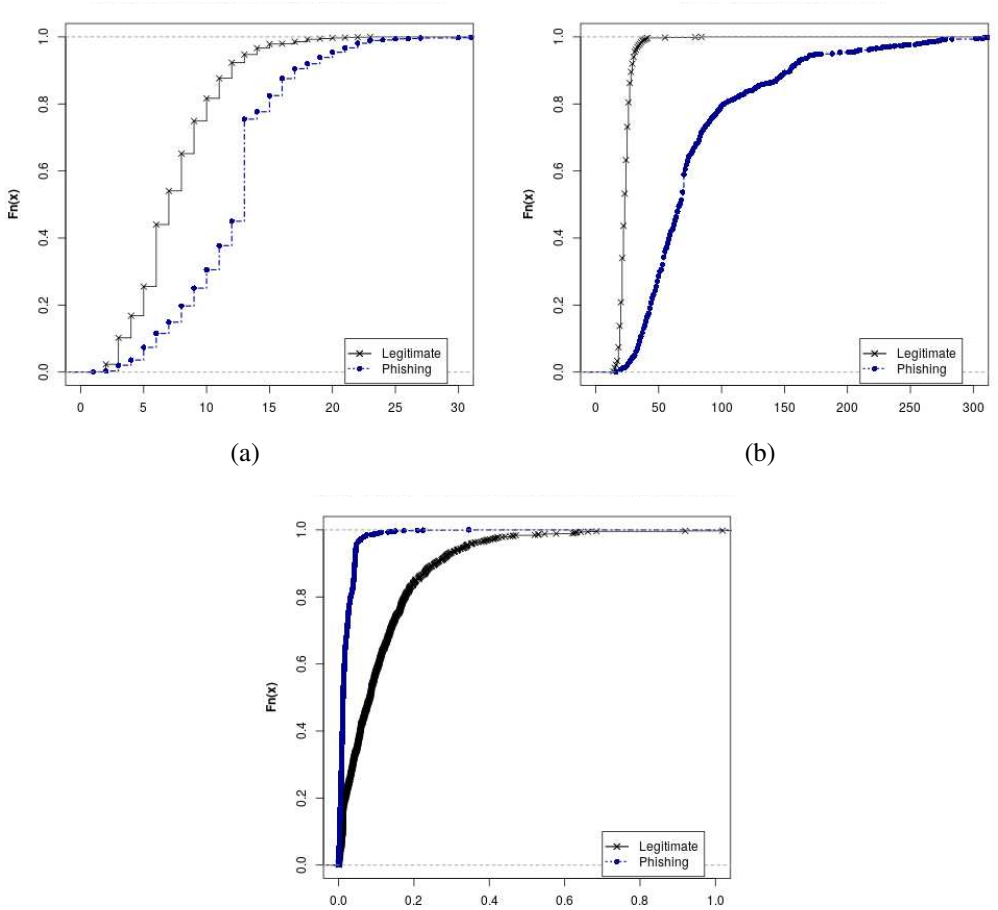


Figure 3.2: ECDF plots for (a) Domain-Name Length, (b) URL Length and, (c) Link Ratio in BODY

(a), the ECDF of this feature shows sufficient distinction between the legitimate websites and the phishing websites.

Feature 2 (Existing): URL Length. The URL length is a popular feature among all known phishing detection approaches and is based on the intuition that phishing URLs are longer than legitimate URLs. We describe this feature here primarily to highlight the issue of dataset bias discussed in Section 1.1. In Figure 3.2 (b), we show the ECDF of this feature. On the surface, it seems an excellent feature, however, it is completely data-dependent, and most existing works have generated results that are likely to be heavily influenced by the distribution of this feature in the phishing and legitimate datasets. We generated two sets of classification results: with and without the URL length, to demonstrate the impact of classification due to this feature. The average

accuracy of classification increases by 2% because of this feature and reaches 99%, which matches the state-of-the-art result when only accuracy is considered. Furthermore, if the feature extraction time is also considered, we show that our results are better than the state-of-the-art work.

Feature 3 (Existing): Link Ratio in BODY. This feature is defined as the ratio of the number of hyperlinks pointing to the same domain to the total number of hyperlinks on the web page. The intuition is that, in the process of making a phishing website similar to the legitimate website, the attackers refer the hyperlinks on the landing page to a legitimate domain-name, which is different from the domain-name displayed in the address bar of the browser. This feature is content-agnostic as the ratio can be computed for any phishing website that exhibits this behavior. For example, the phishers create a phishing page to mimic a well-known payment service where all links on the page are to a legitimate website except the login-form in which the users need to enter their information. Accordingly, the ratio of the links referring to the current domain compared to all links found on the website will be different when compared between a phishing website and a legitimate website. To evaluate this feature, we find all the links on the page and the ratio of links referring to the current page over the number of all links found on the page. However, some legitimate websites also exhibited this behavior, and therefore, we used a scaling process to derive the final value of the feature. Figure 3.2 (c), shows the ECDF of this feature, of the raw ratios, with sufficient separation between the two distributions.

Feature 4 (New): Frequency of Domain-Name. This feature counts the number of times the domain-name appears as a word in the visible text of the web page. The intuition is that many web pages repeat the domain-name several times in their web page, as part of disclaimers, privacy terms, and so on. Therefore, if the domain-name does not appear at all on the web page, then there is something suspicious about such a web page. This is a key feature that captures the visual relationship of the domain-name to the web page. In practice, we find this feature to be very indicative and useful in detecting phishing websites. Note that, for classification purpose, we converted this feature into a binary feature, *i.e.*, if the domain-name does not appear on the web page, we set it to 0 and if it appears more than once, we set it to 1.

3.1.3 Binary valued features

Table 3.1 summarizes the percentage distribution of the binary features in the sample dataset.

Feature 5 (Existing): HTTPS Present. An SSL certificate is issued for a particular domain-name. Most legitimate websites used SSL certificates and operated over HTTPS protocol. Therefore, if a website uses HTTPS, the feature value is 1 and if not, it is 0. Recently, phishing websites are using HTTPS as well and this explains the relatively high distribution.

Table 3.1: Binary Feature Distribution

Feature	Legitimate	Phishing
HTTPS Present	0.92	0.23
Non-alphabetical Characters	0.05	0.36
Copyright Logo Match	0.26	0.0
Page Title Match	0.87	0.03

Feature 6 (New): Non-alphabetical Characters in Domain-Name. Attackers use non-alphabetical characters, like numbers or hyphens, to generate newer phishing domain-names, which are very similar to legitimate domain-names. If the domain-name has any non-alphabetic character, this feature is set to 1 and 0, otherwise. Past works [44, 45] have considered a variant of this feature, *i.e.*, they examined the number of special characters in the entire URL. However, as discussed earlier, generating customized noisy URLs is a relatively easy task for the attackers.

Feature 7 (New): Domain-Name with Copyright Logo. Many legitimate websites use the copy-right logo to indicate the trademark ownership on their organization name. Usually, the domain-name is placed before or after the copy-right logo for such websites. To generate this feature, we considered up to 50 characters before and after the copyright logo, removed the white spaces, and checked for the presence of the domain-name in the resulting string. Surprisingly, we found that none of the phishing websites placed their actual domain-names along with the copyright logo. That has aroused the suspicion of any web user and therefore, we found this feature to be an excellent distinguisher.

Feature 8 (New): Page Title and Domain-Name Match. Many legitimate websites repeat the domain-name in the title of the web page. We found that many phishing websites used this feature to deceive users into believing that they were visiting legitimate websites. But, clearly, a phishing website would not use the phishing domain-name on the title page as it would be clearly visible to the user. As shown in Table 3.1, our intuition proved right and we found that less than 3% of the phishing websites were using this feature, but over 87% of legitimate websites had this feature.

A Comparison with [48]. In [48], although the authors have alluded to the use of the domain-name as *one* of the factors and described several features, they did not base their approach entirely on this aspect as we have done in our work. Some of the features common with our work are Feature 4, the frequency of occurrence of a domain-name, and Feature 8, the match of a domain-name with the title along with some more domain-name-based features. Furthermore, the approach in [48] uses many other features, over 200, to perform the final classification and even ignored some domain-name-based features. For instance, they ignored Feature 7, domain-name match with copyright logo, which we found very useful in detecting phishing websites.

3.2 Experimental evaluation

We conducted two sets of experiments to assess the performance of our model trained with various machine learning classifiers. The first set of experiments were conducted on a prepared dataset and the second set of experiments were conducted on a live unknown phishing dataset from `OpenPhish.com`. Only one past work [44] demonstrated a similar result on unknown datasets with a detection rate of 95%. In contrast, our approach achieves much higher detection accuracy, close to 99.7%. We show the time taken to extract the feature values for each website, the training time for each classifier, and the time taken by the classifier to predict whether a website is phishing or not. We implemented our approach using the Sci-kit [88] library in Python 2.7 on a desktop running Fedora 24 OS with Intel Core® 2 Duo CPU E8300© 2.4 GHz processor with 6 GB RAM.

3.2.1 Datasets

For the list of legitimate websites, we obtained the 1000 top-ranked websites from `Alexa.com` and assumed them as legitimate. For the phishing websites, we got 1000 phishing websites from `PhishTank.com` and 2013 phishing websites from `OpenPhish.com`.

DS-1. This set includes 1000 legitimate websites from `Alexa.com` and 1000 phishing websites from `PhishTank.com`. In the experiments, we trained and tested on this dataset with 80% data for training and 20% data for testing using five-fold cross validation.

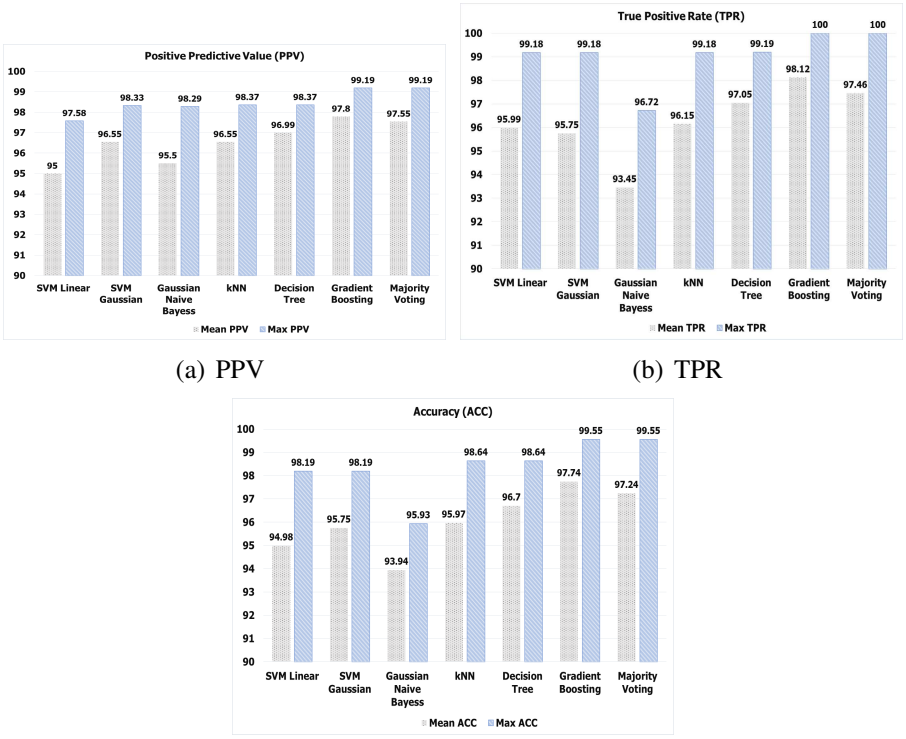
DS-2. This dataset includes 1000 legitimate websites from `Alexa.com` and 3013 phishing websites from `PhishTank.com` and `OpenPhish.com`. For this dataset, we considered 1000 legitimate and 1000 phishing websites for training without cross-validation. The remaining 2013 websites were used for testing.

3.2.2 Experiment 1: performance on DS-1

We designed two different experiments to evaluate the accuracy of classifiers on DS-1. In the first experiment, we used all the features described in Section 3.1 except for the URL length. In the second experiment, to show the bias of URL-based features, we included URL length and demonstrated the increase in classification accuracy. The URL length feature is one such biased feature that exhibits significantly different distribution for phishing and legitimate URLs, as phishing URLs are typically longer in publicly available datasets.

Results without URL Length Feature. Our domain-name-based approach achieves 97% accuracy and validates our basic hypothesis. We show the results in Figure 3.3. For each of the parameters, we show the maximum value achieved and the average value across all the validations. Gradient Boosting performed the best with a maximum accuracy of 99.55% percentage and an average accuracy of 97.74%. For Gradient Boosting and Majority Voting, the TPR is very high, 98.12% and 97.46%, respectively, and so is the PPV, 97.8% and 97.55%, respectively, showing the high phishing detection capability of the classifiers. We note that our average accuracy of 97.74%

is very high when compared to several existing works that used a rather large and diverse set of features.

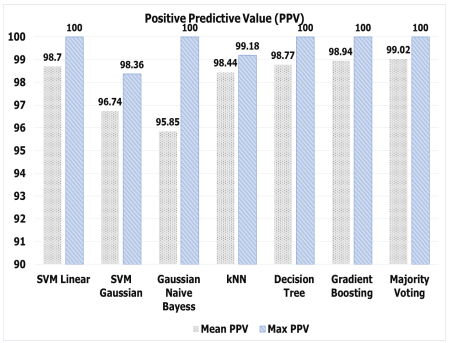


(a) PPV (b) TPR

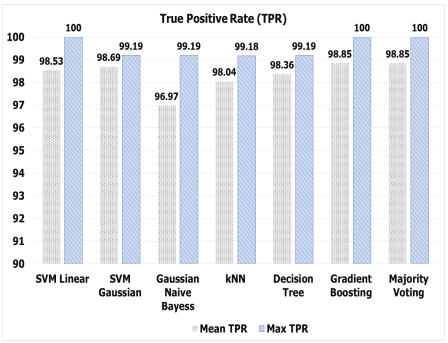
(c) ACC

Figure 3.3: PPV, TPR and ACC on DS-1 without URL Length Feature

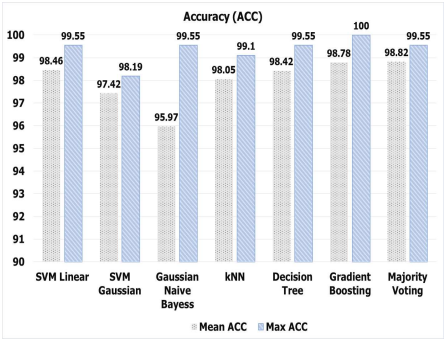
Results with URL Length Feature. This feature results in higher accuracy and clearly demonstrates the bias due to the dataset. We show the results of these experiments in Figure 3.4. There is an increasing trend across all the classifiers for all the parameters considered. There is a clear increase in PPV where four classifiers reported an average of 98% and above with Majority Voting reporting 99%. Excepting Gaussian Naive Bayes, all other classifiers recorded an average TPR of 98% and above, with a maximum of 100% for three classifiers. The accuracy also showed an increasing trend with the average accuracy increasing to 98.8% for Gradient Boosting, and the maximum accuracy of 99.55% for several other classifiers. This experiment clearly shows that features like URL length tend to impact classification accuracy depending on the dataset.



(a) PPV



(b) TPR



(c) ACC

Figure 3.4: PPV, TPR and ACC on DS-1 with URL Length Feature

3.2.3 Time analysis for DS-1

Feature Extraction Timings. Our feature extraction time is very low, of the order of few milliseconds, and demonstrates the efficiency of our feature set. Table 3.2 shows the results of our feature extraction. The total time for extracting features of a legitimate website is about 0.117 seconds and for a phishing website is 0.02 seconds, which indicates the real-time nature of our approach. This is extremely low compared to the state-of-the-art approach in [48] where the extraction time was in the order of a few seconds. We emphasize that the average loading time of a web page like `msn.com`, is around 1 second and our feature extraction adds only a few milliseconds overhead to this process.

Training and Classification Timings. Our classifier training and classification times, that are shown in Table 3.3, are very low, of the order of a few micro-seconds, and again demonstrates the efficiency of our approach. The testing times reported are the average across the five-fold cross-validation and do not include the feature extraction time. The training can be done offline and the testing takes a few micro-seconds to perform, after the feature extraction. Given that cumulative

Table 3.2: Feature Extraction Timings

Feature	**Legitimate** (μs)	**Phishing** (μs)
HTTPS Present	4.12	3.87
Domain Length	63.45	66.45
Page Title Match	26.9	32.3
Frequency Domain-Name	333.8	33.09
Non-alphabetic Characters	32.64	13.68
Copyright Logo Match	2737.56	450.48
Link Ratio in Body	114482.87	19445.67
URL Length	0.3576	0.5066
Total Time (s)	0.117	0.02

Table 3.3: Training/Testing Timings

Classifier	**Train** (in ms)	**Test** (in μs)
SVM Linear	1339.85	6.74
SVM Gaussian	703.62	38.32
Gaussian Naive Bayes	2.28	1.47
kNN	7.36	14.85
Decision tree	2.49	0.80
Gradient Boosting	2737.56	450.48
Majority Voting	177.73	3.25

time for feature extraction and testing is less than 2 milliseconds, we claim that our approach can be deployed in practice as a client-side browser plug-in.

3.2.4 Experiment 2: performance on DS-2

In this experiment, we examine the robustness of our learning approach on unknown and unseen data. We obtained a list of 2013 live phishing websites from `OpenPhish.com`. Although a higher number of sites were listed, many sites were unavailable and few were blocked by the corresponding ISPs. We trained the classifier in two modes: without including the URL length feature and with the URL length feature included. Finally, we tested the classifiers on the 2013 data instances and show the results in Table 3.4. These results show the remarkable performance of our approach. Unlike the previous approach [44], which attempted a similar experiment, for many of our classifiers, the TPR largely remains *unchanged* across both the experiments and even

Table 3.4: True positive rate of testing phase for DS-2. Best values are shown in bold.

Classifier	Without URL Length	With URL Length
SVM Linear	94.09	94.24
SVM Gaussian	92.75	90.81
Gaussian Naive Bayes	91.06	92.75
kNN	93.74	**99.7**
Decision tree	97.91	97.27
Gradient Boosting	**98.21**	**99.75**
Majority Voting	95.33	97.67

shows a slight increase for Decision tree and Gradient Boosting classifiers. Furthermore, when including URL length, the TPR even reaches 99.7% for kNN and Gradient Boosting. This result also confirms our hypothesis that domain-name-based features can accurately capture the nature of a phishing website.

3.2.5 Comparison with previous work

We compare our results empirically with existing state-of-the-art solutions in Table 3.5. Our basis for comparison is the number of features, the accuracy, and whether client-side features only are used or third-party features are included. We did not include the run-times of the approaches as that is a system-specific metric. However, we note that our scheme reports micro-second level feature extraction and classification time, even when run on a relatively low-performance laptop with Core 2 Duo processor. From a different perspective, while Verma *et al.* [44] and Marchal *et al.* [48] used large number of features with 35 and 210 respectively, our proposed approach uses fewer number of features (only 7 and 8 features) and performance is comparable.

3.3 Conclusion

In this section, we described the first approach towards the design of *only* domain-name-based features for the detection of phishing websites using machine learning. Our feature design emphasized the elimination of the possible bias in classification due to differently chosen datasets of phishing and legitimate pages. Our approach differs from all previous works in this space as it

Table 3.5: Comparison with State-of-the-art Approaches. #Leg. indicates number of legitimate instances in the dataset, and #Phi. indicates number of phishing instances.

Approach	# Leg.	# Phi.	# Features	Acc.	Client Side
Cantina [41]	2100	19	7	96.97	No
Cantina+ [43]	1868	940	15	97	No
Verma *et al.* [44]	13274	11271	35	99.3	Partial
Off-the-Hook [48]	20000	2000	210	99.9	Yes
Our approach without URL Length	1000	3013	7	97.7	Yes
Our approach with URL Length	1000	3013	8	98.8	Yes

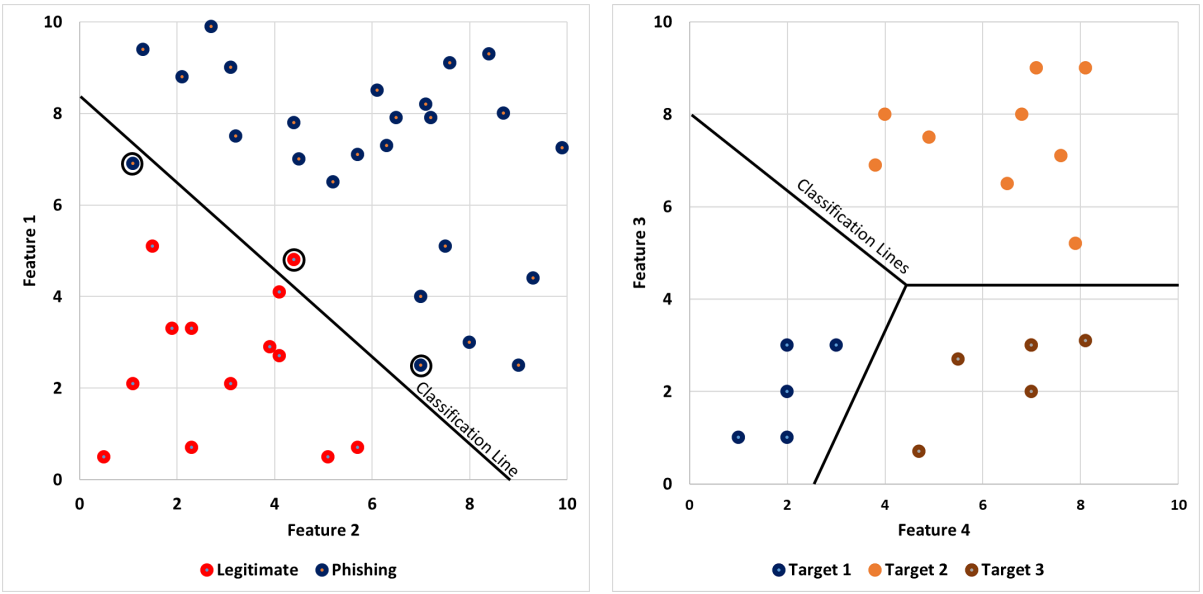
models the relationship of the domain-name to the intent of phishing. With only seven features we are able to achieve a classification rate of 97%. Furthermore, we were able to show a detection rate of 97-99.7% for live black-listed URLs from `OpenPhish.com`. This shows that our approach is able to adapt to the complex strategies used by phishers to evade such detection mechanisms. As our features explore the content found in the visible space of the web page, an attacker will need to put a huge effort to bypass our classification. In trying to bypass our approach, an adversary may end up designing a page that will make any user suspicious. Furthermore, we demonstrated the shortcoming of using URL features such as URL lengths, which seem to give higher accuracy but may not do so in the near future. Our feature extraction and classification times are very low and show that our approach is suitable for real-time deployment.

Chapter 4

Fingerprinting-Based Approach for Overcoming Bias in Phishing Detection

4.1 Proposed approach

Phishing campaigns usually work based on a three-part scheme. The first part is using email or some form of communication to lure users and redirect them to a phishing page. The fake page closely mimics a trustworthy site. Finally, the user enters their information, which is captured by the adversary. Phishing websites must target at least one genuine site, which we define *target website*, and should be similar in terms of visual and textual traits to the target website to earn the end-user's trust.



(a) Existing Approach

(b) Proposed Approach

Figure 4.1: Two different ways of modeling the phishing problem with regards to feature definition. (a) image shows features that are defined based on similarity among phishing websites and that among legitimate websites. (b) image shows features that have been defined based on their similarity to target websites. Samples that are attacking a target are clustered together.

In our proposed approach, instead of defining features that group the phishing websites together, we relate a suspicious phishing website to its target and define features based on the similarity of a given suspicious website and its target. Figures 4.1 (a) and (b) hypothetically explain this issue in detail. Figure 4.1(a) is a scatterplot of phishing and legitimate samples with two features: *feature 1* in the y-axis and *feature 2* in the x-axis in existing approaches. The green dots represent the position of phishing instances, and the blue dots represent legitimate websites. This image clearly shows, based on feature definition, phishing and legitimate websites are distinguishable with two misclassified samples. In addition, it shows that phishing samples are not related to any legitimate website.

Figure 4.1(b) shows our proposed approach. There are still two features, namely *feature 3* and *feature 4* in the y-axis and the x-axis respectively. There are three target websites in these graphs named *Target 1*, *Target 2*, and *Target 3*. All dots in the graph indicate phishing samples. Since features are defined to show the similarity of phishing instances to the target website, phishing websites in each group are close to each other, and this group represents individuals attacking a target website in the form of a cluster.

4.1.1 Fingerprint definition

We define the fingerprint of a legitimate website as a mathematical representation of that website, which can uniquely distinguish a legitimate website from other legitimate websites. Moreover, comparing suspicious websites with this fingerprint would determine if it is similar to the target website or not, a vital sign when a phishing website attacks a target. If a suspicious website's similarity to a target website exceeds a given threshold, then this would be assumed as a phishing website.

For each given target site, we would consider both the visual and textual characteristics of the target site. The process of extracting a fingerprint and then matching it with suspicious websites is independent of other legitimate websites. Consequently, the phishing detection process for each target would be independent of other targets. Thus, we have no issues with biased data.

We define the fingerprinting for any given legitimate website as follows. For each given legitimate website, there would be a set of features that:

- Uniquely represent a website so that it can be distinguished from other legitimate websites which we call the *fingerprint* of the website, and
- Comparing the fingerprint of a website to any other given website will return the level of similarity between them.

Each legitimate website has at least one screenshot. We use f_i for feature i and denote F_j^A as all fingerprint features of j^{th} screenshot of legitimate website A. Thus, if screenshot j has L_j number of features, F_j^A would be:

$$F_j^A = \{\ f_0\ \cup\ f_1\ \cup ... f_{L_j}\} \tag{4.1}$$

If $|A|$ is the total number of screenshots for legitimate website A, then the fingerprint of legitimate website A would be called as F^A and calculated as follows:

$$F^A = \{\ F_0^A\ \cup\ F_1^A\ \cup ...\ F_{|A|}^A\ \} \tag{4.2}$$

Screenshots are captured images that have been shown on the end-users display. These captures are taken from login pages of legitimate websites or login pages of older versions of a legitimate website. If visual or textual traits of a legitimate website change over time, we need to add new screenshots and update the fingerprint to capture these changes to detect new phishing attacks.

4.2 Methodology of proposed approach

Our approach consists of the following steps. We are given the *benign set* that consists of the set of legitimate websites that we are trying to protect from phishing attacks. For each legitimate website in the benign set, we perform the following three steps.

- **Step 1: Extracting Fingerprints from Legitimate Websites.** We create a fingerprint using textual and visual features.
- **Step 2: Creating and Labeling Dataset.** We create a labeled dataset. We assign a label of 1 if the data is a phishing sample that is targeting the legitimate website. Otherwise, we assign it a label of 0.
- **Step 3: Create a Machine Learning Model.** We create a machine learning model corresponding to the legitimate website.

4.2.1 Extracting fingerprints

For each given legitimate website, we prepare one or more screenshots of the target website. In some cases, if the website has multiple login pages that are not visually identical, we prepare more than one screenshot, then extract textual and visual features. The visual and textual characteristics of each page define the unique identity of each legitimate website.

Textual Sector. Textual elements are the text that is visible to the end-user. Examples include text that asks users to enter their username and password or terms and conditions of using the services. Graphical designers use these characteristics to create a uniquely distinguishable webpage. Thus, we use those characteristics to create fingerprints in this study.

For the textual feature gathering, we use an Optical Character Recognition (OCR) algorithm, which uses machine learning to extract text word by word as an object which we can use in programming. For our OCR algorithm, we used the web-based Google Cloud Vision API which is one of the best algorithms available. Google OCR hides technical details from end-users, but it includes five steps: *Text Detection*, *Direction Identification*, *Script Identification*, *Text Recognition*, and *Layout Analysis*.

Text Detection uses a Conventional Neural Network (CNN)-based model, to detect and localize the line of text that generates a set of bounding boxes. *Direction Identification* classifies direction per bounding box and *Script Identification* identifies script per bounding box. *Text Recognition* recognizes the text from the image. It includes a character-based language model, an inception-

style optical model, and a custom decoding algorithm. Finally, *Layout Analysis* determines reading order and distinguishes titles, header, *etc.* [89].

The extracted text provided by Google OCR is cleaned. We ignore punctuation and stop words and make all texts lower-case. We fix misspelled words if there are any. Misspelled words are relatively rare in legitimate websites but common for fake websites. The cleaned list of extracted words creates the *textual sector* of the fingerprint.

Visual Sector. Visual elements include, but are not limited to, brand logo and other graphical iconic elements related to the website. These are elements that make a login page unique compared to other legitimate webpages. We used a segmenting algorithm to detect, localize, and save many of these segments found in legitimate web pages. The segments generated were then manually scrutinized to eliminate the ones that were considered not relevant to the site's fingerprint.

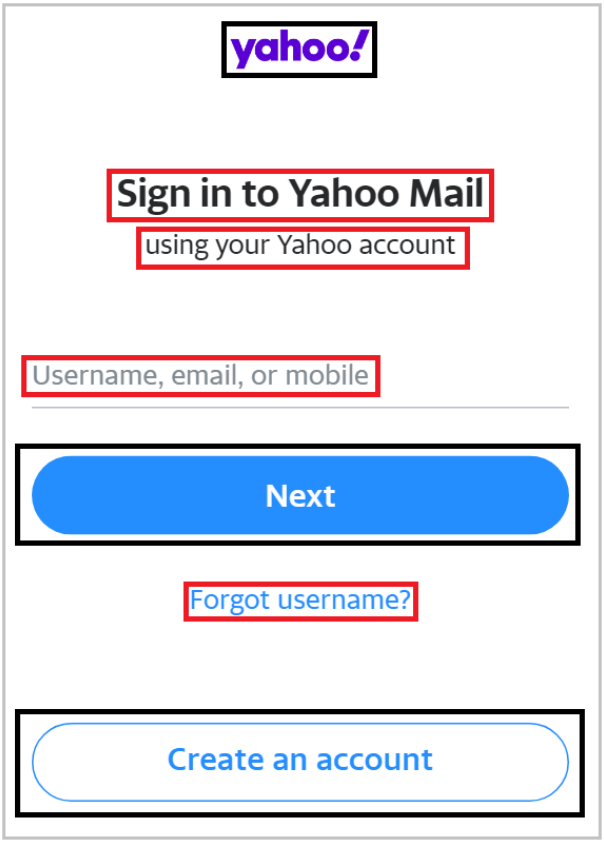


Figure 4.2: Legitimate screenshot from Yahoo.com. Black colored boundaries specify visual segments and red-colored boundaries specify parts with texts returned by the OCR algorithm.

Figure 4.2 shows a screenshot of *Yahoo*, captured from the legitimate website. The parts in black rectangles are the visual segments and the parts in red rectangles are the textual segments that constitute the fingerprint.

4.2.2 Creating dataset

We create a dataset for each legitimate website. The phishing samples that target the legitimate website are labeled as 1 and anything else is labeled as 0.

Textual Sector Matching. For each given input website, we use the previously discussed OCR algorithm to get all the text out of the screenshot. Then we match each word in the fingerprint with words of the given input. If the word in the fingerprint does not match any word in the website, we assign that feature as -1. If the word in the fingerprint does match that of the word in the input, then that corresponding feature is assigned a value that reflects the importance of the word in the input website. The TF-IDF algorithm statistically reflects the importance of a word in a corpus. The corpus consists of all of the words in the legitimate websites. Thus, for each word of a given website, we use TF-IDF to evaluate its importance in the context of the website, if it matches with a word in the fingerprint. In such a case, the TF-IDF score is assigned to the corresponding feature. For example, the company's name will have a higher value than a word like *login* in the learning vector and makes it more meaningful for the machine learning algorithm.

Visual Sector Matching. In order to determine if the visual characteristics of an input image match that of our target website, we need to check if the legitimate website's segments exist in the input image. This fact introduced many complications, as image comparison is often times challenging. We also had to consider what an adversary might do to bypass a comparison algorithm. We decided to leverage the concept of homography to counter simple rotations and deformations. This was achievable by using the key-points of both the segments and the input image. In the next step, we determined the quality of the projective transformation of the input image for each legitimate segment. If no homography transformation is found, we consider that segment to be absent from the input image. Furthermore, if a homography does exist, we compare the resulting image with

the segment to determine the validity of this homography. For the comparison algorithm, we implemented a custom algorithm called *BFMatcher*, which compares the key points using brute force key-point matching and the dot product between these matches. If the comparison value is above a threshold, we consider the segment to exist in the input image. The feature value of this part of the fingerprint matching is the result of the comparison algorithm.

For every given input, we will have a vector that specifies the similarity between the given website and the fingerprint of the target site. We have created our dataset after calculating these features.

4.3 Experiments and results

We discuss the datasets used and machine learning metrics and then elaborate on the two experiments we have conducted and the results in this section.

4.3.1 Created dataset

Dalgic *et al.* [90] gathered screenshots of phishing attacks and made them publicly available. This dataset includes labeled phishing samples of 14 brands. While authors [90] only has the phishing samples, we need screenshots of the legitimate websites to create a fingerprint for them. Thus, we captured screenshots of these legitimates websites and added them to our dataset. We also manually double-checked all of the phishing samples and their relationship to the claimed target website to find discrepancies and fixed a few of them. In the next step, we run our fingerprint extraction algorithm to create a fingerprint for each legitimate website. We evaluated the learning vector based on the similarity to the fingerprint extracted for the targeted website.

For each target website, we created a separate dataset and for each given screenshot, we evaluated whether the text or visual segments exist in the fingerprint or not which was used to encode the feature vector. If the instance is attacking this target website, we label it as 1, otherwise we label it as 0.

Table 4.1: List of used target websites and number of textual, visual, and, total features used for fingerprint, and total number of phishing samples for each target. (* Bank of America)

	#Fingerprint Segment			
Website	**Tex.**	**Vis.**	**Tot.**	**Samples**
Adobe	26	26	52	70
Alibaba	82	22	104	76
Amazon	24	14	38	29
Apple	34	31	65	64
BOA*	182	39	221	111
Chase	127	99	226	111
DHL	48	33	81	109
Dropbox	45	16	61	115
Facebook	84	34	118	144
Linkedin	77	28	105	38
Microsoft	15	10	24	117
Paypal	27	14	41	214
Wellsfargo	166	42	208	134
Yahoo	14	12	26	114

Table 4.1 lists all target sites we used in our experiments with the number of items in both textual and visual segments of the fingerprint and the number of phishing instances for each target. We created 14 separate datasets for the 14 respective target websites. The machine learning classifier will be trained on the corresponding dataset. This helps to relate the phishing samples to a specific target site, instead of relating all phishing websites to all legitimate websites, as we discussed it in [91].

4.3.2 Machine learning and scores

In our experiments, we used five different classifiers available in Sckikit-learn toolkit [88] and then we selected the best one. We used Random Forest (RF), Gradient Boosting (GB), and Support Vector Machine (SVM) with two different kernels: Linear (l) and Gaussian Radial-basis function kernel (g). In addition, we used a Majority Voting (MV) as another classifier that acts as a voter among all of the fitted classifiers. We ran each experiment 10 times and reported the average of the results with five-fold cross-validation to avoid issues of over-fitting. We tested the performance of each learning model against unseen samples.

Evaluating the effectiveness of an algorithm only by relying on accuracy in imbalanced datasets may be misleading. Because a majority group with a large margin can dominate the accuracy result. We have an imbalanced dataset and thus reporting accuracy may be misleading. To address this limitation, F1 score, which is a harmonic average of precision and recall, has been proposed and widely accepted. We report F1 score to show how effective is our algorithm. We also reported accuracy score for further comparison with other studies.

4.3.3 Performance of classifiers

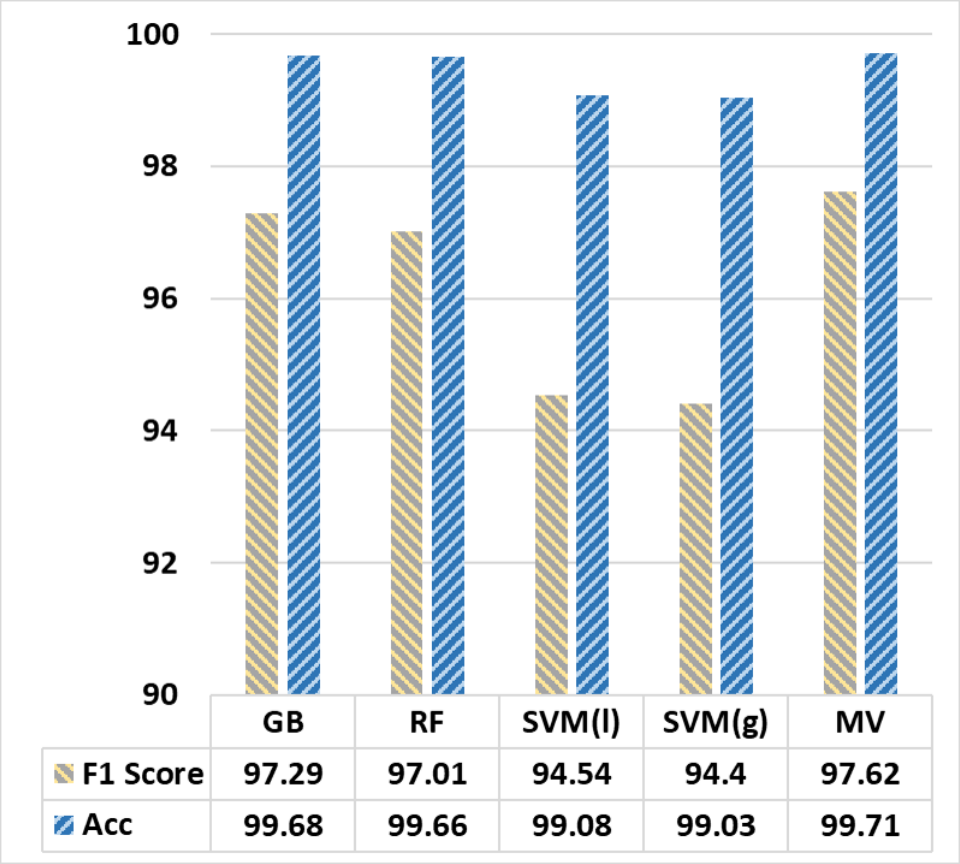
For each dataset of the target site, we split the dataset into three sub-sets: one set with only textual features, the other set with entirely visual features, and the last one with both textual and visual features. We then trained and tested all five classifiers for each sub-set ten times and reported the average.

Figure 4.3 highlights these results. It shows MV with the highest F1 score of 97.62% among all of the classifiers; thus, we selected this classifier for further experiments. Furthermore, it gives the best accuracy as well.

The next best classifier after MV is GB. It has an average F1 score of 97.29% and an accuracy of 99.68%, which is slightly more than RF. These results show that both GB and RF were able to significantly detect phishing attacks against all 14 websites.

4.3.4 Effectiveness of model

Figure 4.4 reports the accuracy of the model for all 14 targeted websites that we ran the experiments for, and figure 4.5 reports the F1 scores with separation based on textual and visual sections of the dataset. For this experiment, we used GB as it gave the best results in the previous experiment. Figure 4.4 shows the results of three sub-sets: only textual features, only visual features, and both together. The accuracy was over 98% for all cases when we used both the visual and textual features. It also shows that the visual sector alone does not give good results for the following websites: *Adobe*, *Amazon*, *Microsoft*, and *Yahoo*.



	GB	RF	SVM(l)	SVM(g)	MV
F1 Score	97.29	97.01	94.54	94.4	97.62
Acc	99.68	99.66	99.08	99.03	99.71

Figure 4.3: F1 score of trained model for different classifiers.

Our dataset is a highly imbalanced dataset, and we reported the F1 score to balance between *precision* and *recall*. Having a high F1 score guarantees both precision and recall have high values. In this case, the classifier does not ignore one class with a lower volume of data to increase total accuracy. Figure 4.5 clearly shows the model gives a high F1 score for all of the websites. *Chase* website with the F1 score of 99%75 has the highest score, and *Microsoft* with 88.42% has the lowest score when we used both visual and textual features. The reason that *Microsoft* has the lowest score among the targeted website is that the fingerprint process could not find enough segments to create the fingerprint.

In addition, our model gives exceptionally high scores among all of the websites regardless of their popularity. While *Amazon* and *Yahoo* are among the top twenty most visited websites [91] but they have the same accuracy or F1 score as *DHL*, which has a popularity rank of 1248 [91].

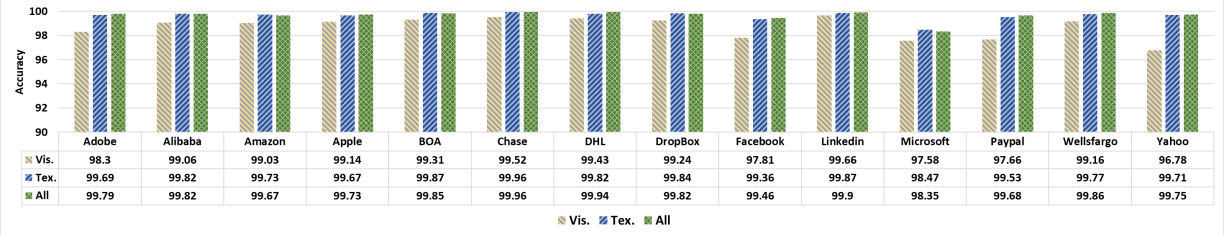


Figure 4.4: Accuracy of trained model for targeted websites.

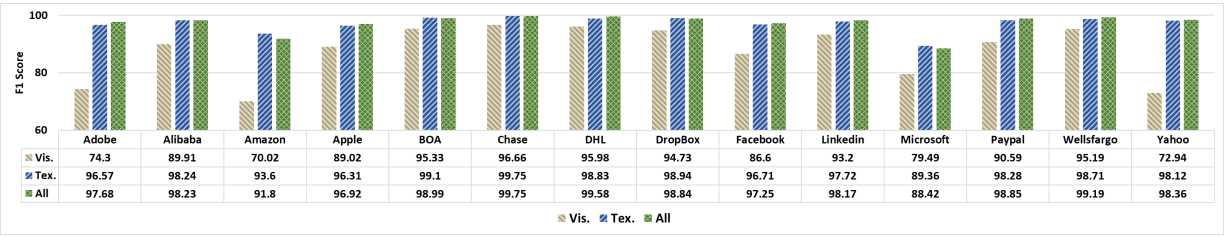


Figure 4.5: F1 score for trained model for targeted websites.

This demonstrates that our experiments were free from the bias stemming from the popularity of the website in contrast to current approaches.

4.4 Conclusion

In this chapter, we propose an approach that detects whether a phishing website is attacking a target legitimate website. We generate fingerprints for legitimate websites using visual and textual characteristics and detect phishing websites based on how closely their features match these fingerprints. Our approach is not biased towards more popular websites and can be adapted for new attacks. We demonstrate our approach on 14 different target websites with varying popularity. Our model achieve an accuracy of 99% for all of them with cross-validated data. Furthermore, we employ a one-vs-all technique and create an imbalanced dataset; we report an accuracy of more than 98% among all of the websites, which is surprisingly high. It may be possible that through adversarial machine learning attackers generate phishing samples that match the fingerprint of legitimate websites. Our future work will investigate how to protect against such attacks.

Chapter 5

Effects of Adversarial Sampling Attacks in Phishing Detection

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 [13, 53, 54, 56, 57] 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 [24]. However, in adversarial contexts, *e.g.* phishing or spam filtering, this is far from 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.

5.1 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.

5.1.1 Attacker's goal

Biggioa *et al.* [92] 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 to a specific type of error in the system and degrades the classifier's scores.

5.1.2 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* [78, 92] 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.

5.1.3 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 an *evasion* attack, the attacker does not have access to the training set and intentionally and smartly manipulates 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.

5.1.4 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. The adversary generates new phishing samples that are labeled incorrectly by the 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*. The 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.

5.2 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 5.1.2.

5.2.1 Defining dataset

We use a similar notation to that used in [78]. 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:

$$x_i = [x_i^1, \cdots x_i^d] \in \mathcal{X} \tag{5.1}$$

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.

5.2.2 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 costs. In Table 5.1, we defined the notation used for describing our approach.

Table 5.1: Table of notations

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

5.2.3 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 combinations 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, 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 does 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.

5.2.4 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 5.4. 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 5.2:

$$d(x_i, x_i^{'}) = \sqrt{\sum_{k=1}^{n} (x_i^k - x_i^{'k})^2} \tag{5.2}$$

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.

5.2.5 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.

5.3 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.

5.3.1 Outline of clustering approach

In general, the clustering of data samples using standard approaches like the k-means algorithm [93] 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.

5.3.2 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 a *transferred sample*. This definition captures a key notion that an adversarial sample is likely to belong to a different cluster due to 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 5.4.

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 5.3 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.

$$P(Ay \mid Tr) = \frac{P(Ay \cap Tr)}{P(Tr)} \tag{5.3}$$

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.

5.4 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.

5.4.1 Used datasets

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

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.* [57] reported an accuracy of 97-98% in the experiments, which is significantly high.

DS-2: Rami *et al.* [26] created this dataset in 2012 and shared it with the UCI machine learning repository [94]. 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.

DS-3: In 2014, Abdelhamid *et al.* [25] shared their dataset on the UCI machine learning repository [94]. 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.

DS-4: This dataset is the most recent, from the year 2018, that is publicly available. It has been created by Tan *et al.* [27] and was published on Mendeley [1] 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 5.2 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,

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

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.

5.4.2 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 $+1$ and legitimate websites as -1. We used five-fold cross-validation to avoid over-fitting 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 5.3 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 an 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 5.4 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 the best results among all of the experiments, so we selected these two classifiers for the next experiments.

5.4.3 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 5.2. 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 5.2.4. We call the phishing samples with the lowest cost of generating adversarial samples as *optimized samples*.

5.4.4 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 5.4. For the datasets DS-1 and DS-3, we selected GB, and for DS-2 and DS-4, we chose RF.

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

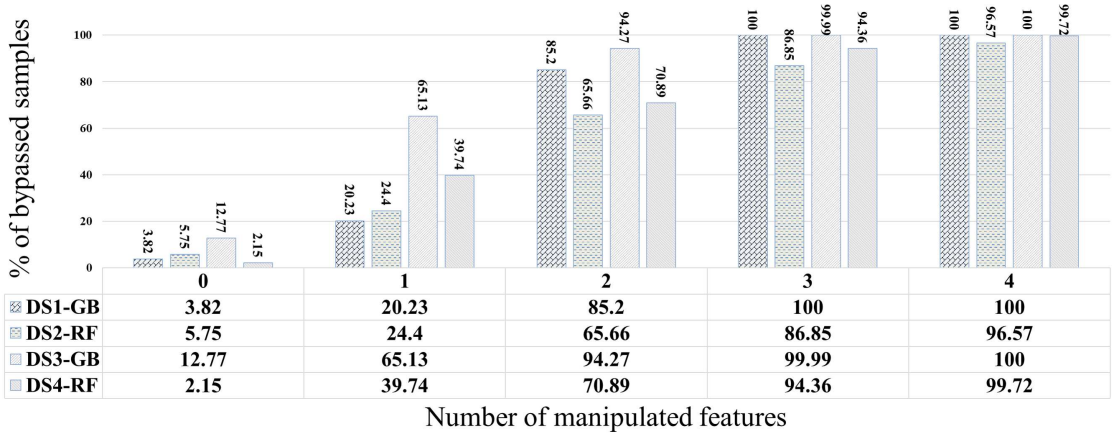


Figure 5.1: Robustness of datasets against adversarial samples.

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 5.1 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 5.1 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.

5.4.5 Dataset vulnerability level

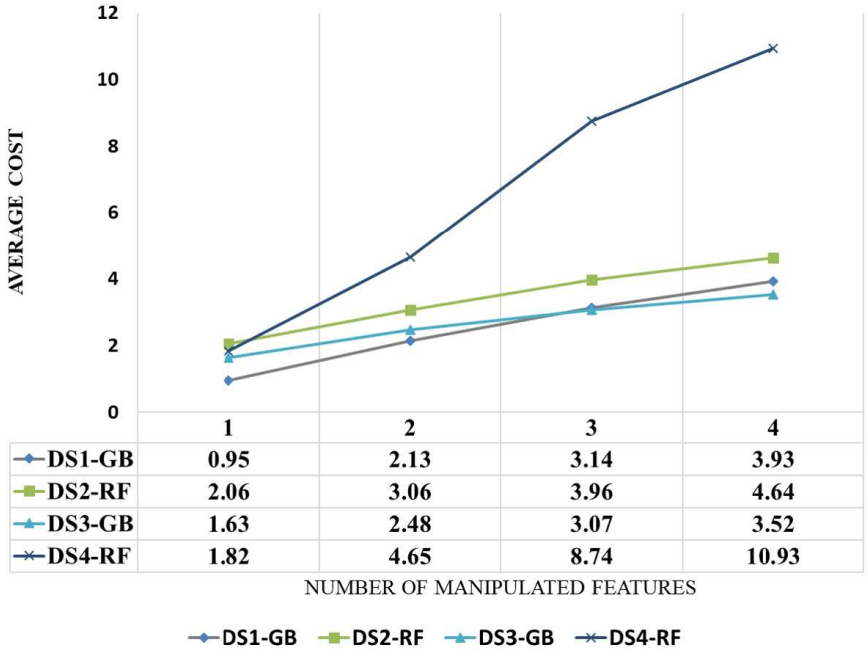
In this experiment, we studied the cost that an adversary has to pay to bypass a classifier. From an adversary perspective, it is not inexpensive to manipulate an instance with new feature values to create an adversarial sample. In Section 5.2.4, we assessed the cost and in Section 5.2.5, we defined 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 5.4. 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 5.2 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.

5.4.6 Cluster directed adversarial sampling

We discuss using the clustering approach described in Section 5.3 and present the results. In this experiment, we calculated the probability of *transferred samples* and *adversarial samples* as we discussed in Section 5.3. For each dataset, we calculated the probability of generating adversarial samples and the probability of such a sample being transferred to a new cluster from the original cluster.



	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

Figure 5.2: The manipulation cost for adversarial samples based on the number of manipulated features.

Figure 5.3 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 5.3 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 5.4 depicts how these probabilities varied among different clusters. Figure 5.4(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 5.4(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 5.4(c) for DS-3. The same pattern can be seen for transferring as well. There is a

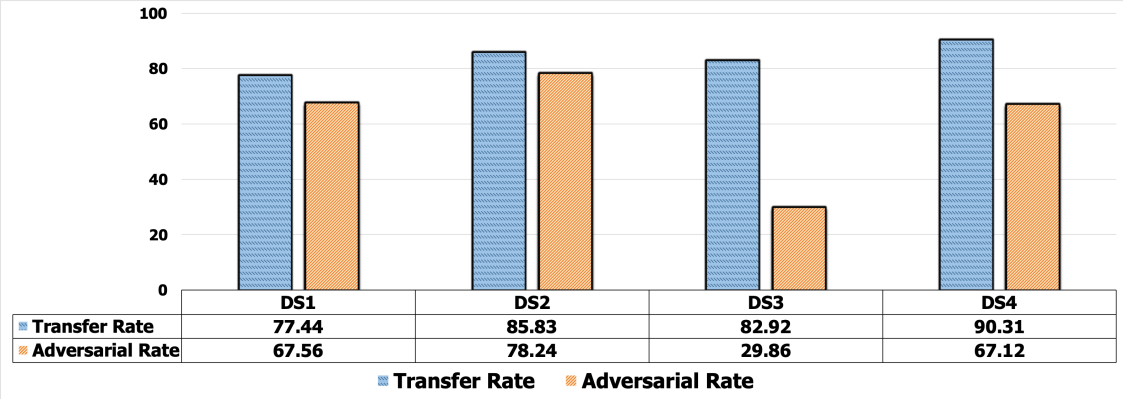


Figure 5.3: Ratio of bypassing and transferring adversarial samples in tested datasets

big gap between the probability of generating adversarial samples and transferring in this dataset, something that has not been seen in other datasets.

Figure 5.4(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.

5.4.7 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 5.5 depicts these results.

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.

This knowledge for an adversary is compatible with the threat model defined in Section 5.1. In our proposed model, an attacker has access to the predict function and phishing website.

Figure 5.5 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

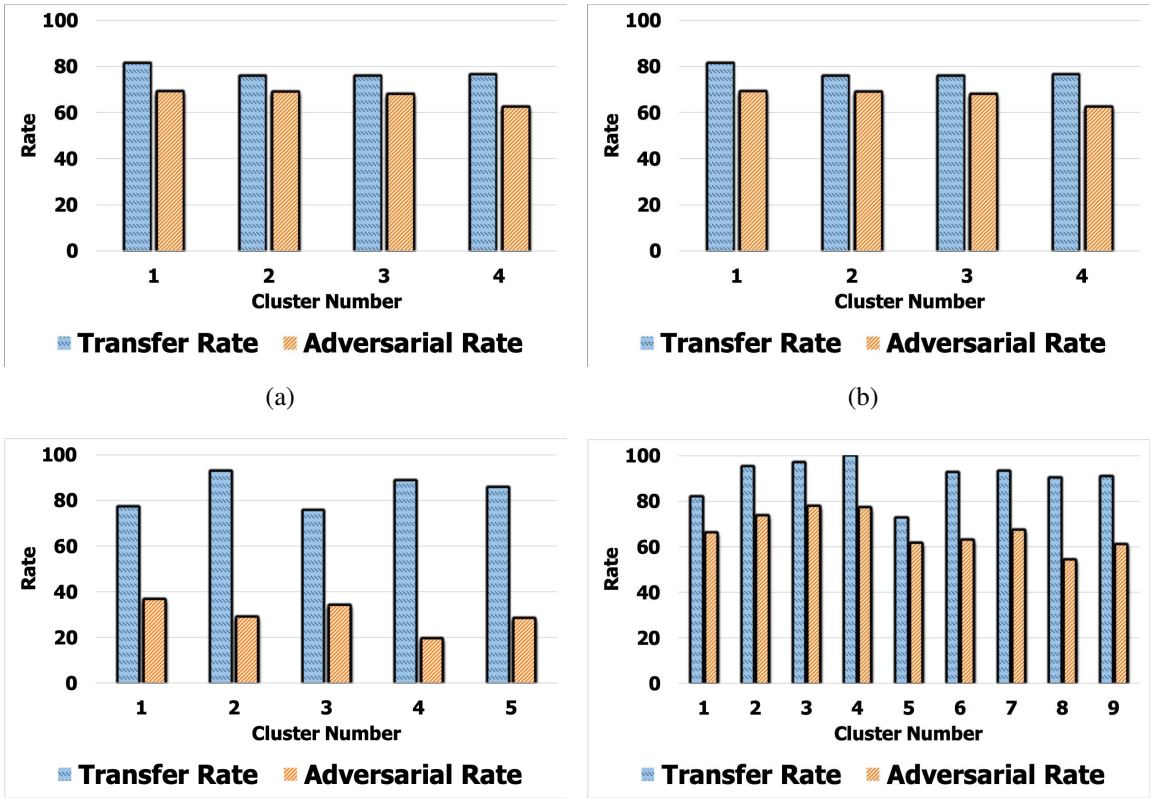


Figure 5.4: Distribution of bypassed and transferring samples for each cluster in all of tested datasets: (a) DS-1; (b) DS-2; (c) DS-3; and, (d) DS-4

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 not happening. Based on the results, there is not a significant correlation between these two probabilities.

5.4.8 Selecting best cluster

As discussed earlier in Section 5.2.4, 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

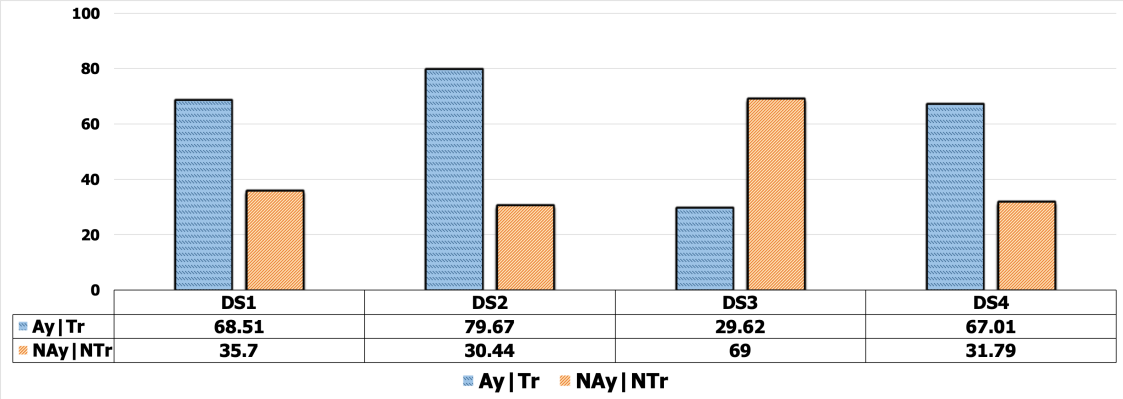


Figure 5.5: Conditional probability of adversarial samples

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 the set of instances in cluster i, and ins is the set of all phishing instances. $P(c_i)$ will be calculated as follow:

$$P(c_i) = \frac{|in_i|}{|ins|} \tag{5.4}$$

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 the 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 the initial cluster is i and a newly generated sample is transferred to cluster j, we denote this probability as $P(tr_{i,j})$. This probability will be calculated as follow:

$$P(tr_{i,j}) = \frac{|mem\_trans_{i,j}|}{|ins_i|} \tag{5.5}$$

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 the 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 $P(Ay_i)$.

$$P(Ay_i) = \frac{|adv\_sam_i|}{|gen\_sam_i|} \tag{5.6}$$

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 normalized this probability with cost in Equation 5.7, which can be viewed as the *likelihood* of this transfer with a given cost from an original cluster of i to a transferred cluster of j.

$$\mathcal{L}(i,j) = \frac{P(c_i) \ * \ P(tr_{i,j}) * \ P(Ay_j)}{cst_{i,j}} \tag{5.7}$$

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 5.6 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.

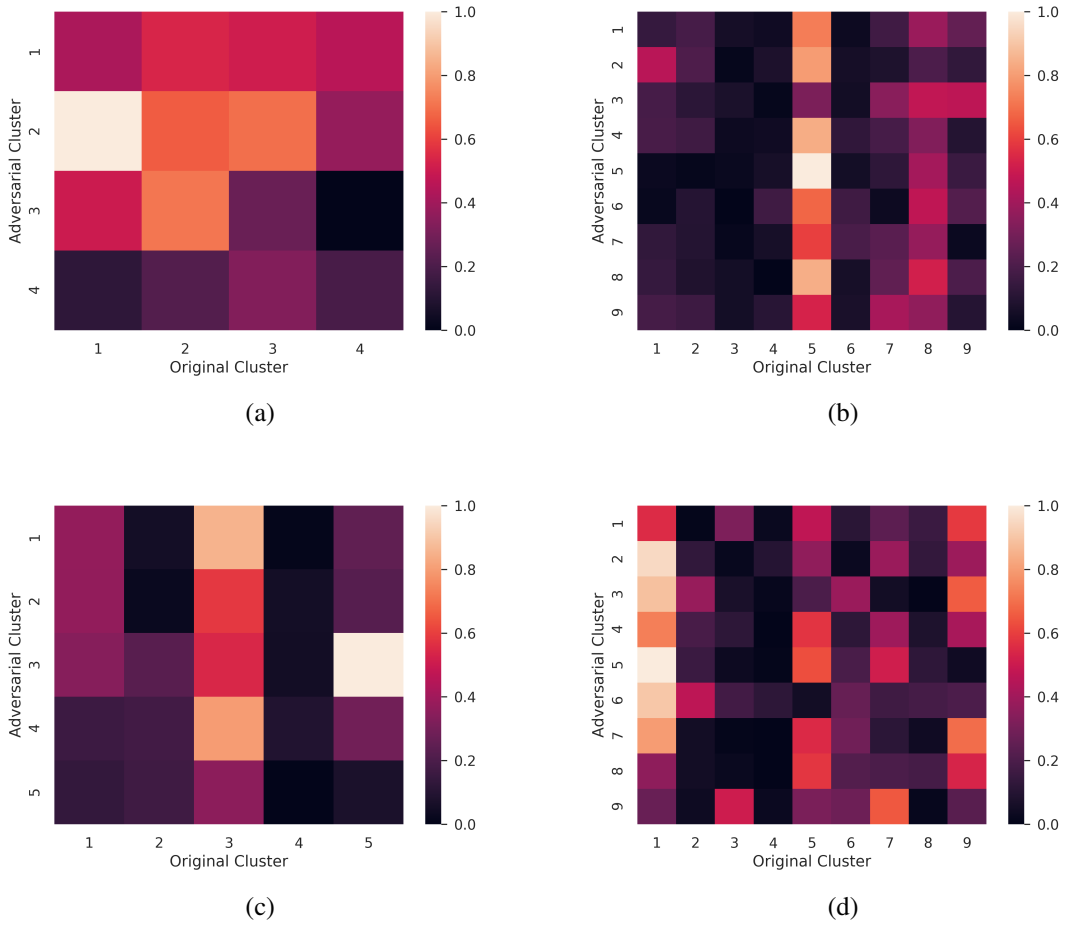


Figure 5.6: The relation between an original cluster of an instance with adversarial samples: (a) describes the DS-1; (b) describes the DS-2; (c) describes the DS-3; and, (d) describes the DS-4.

For example, Figure 5.6(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 5.6(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 a low cost. A similar pattern is seen in Figure 5.6(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 5.6(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.

5.4.9 Comparing the results with prior research

In this section, we compare our approach with some of the previous research in this field. Table 5.5 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 solely focused on the URL itself [14, 55], but others look at both the URL, and the content of the page [57, 67]. 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 [57, 67], there are also datasets with millions of instances [53, 55]. 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 [52, 54] achieved an accuracy of 99%, which is significantly high. Tian *et al.* [55] 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 5.5. 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.

5.5 Conclusion

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 at 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 manipu-

lated. 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.

Algorithm 1: Algorithm for generating adversarial samples. It manipulates feature values and if the new sample bypasses the classifier, it will return them as new adversarial samples.

$x = [x^1, x^2, \ldots, x^d]$;
$X_P^i, \forall i \in 1 \ldots d$; /* Phishing instance x used to generate samples, feature value i obtained from all phishing instances for all features 1 to d. */
$genSamples$; /* Array of generated samples */
$advSamples$; /* Array of generated adversarial samples */
/* Refer to 5.1 for the notation. */
/* Initialization */
$advSamples \leftarrow \{\}$; /* Initialising set of generated adversarial samples */
$genSamples \leftarrow \{\}$; /* Initialising set of generated samples */
$PhVal \leftarrow 1$; /* *Cartesian Product* generates all possible feature values of current phishing instances. */
for $k \leftarrow 1$ *to* d **do**
    $PhVal \leftarrow PhVal \times X_P^k$
for $\pi \in \Pi$ **do**
    for $n\_fea \in PhVal$ **do**
        $nw\_smp \leftarrow (\pi * n\_fea) + (\neg\pi * x)$; /* 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. */
        /* Check if the generated sample differs from the input sample. */
        if $x \neq nw\_smp$ **then**
            $genSamples \leftarrow genSamples \cup nw\_swp$ **if** $xislabeledaslegitimate$ **then**
                $advSamples \leftarrow advSamples \cup nw\_smp$

Table 5.2: Number of instances, features, and a portion of legitimate and phishing websites in each dataset.

	Data shape (#)		Labels (%)	
Dataset	**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

Table 5.3: 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 5.4: The classifier that holds the 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

Table 5.5: Comparison of different approaches in the literature including our proposed approach. * indicates current work.

Author	Description	Size	ACC	Adv.
[13]	-Scalable feature-rich framework with a series of new and existing features -Not using third-party services, Language agnostic	22.3K	95%	No
[14]	-Real-time detection mechanism based on NLP of URLs, Language independent -Tested on a large dataset, Not using third-party service	73K	97%	No
[52]	-Features based on lexical-, distance-, and length-related features of the URL -Using four large datasets	115K	99.3%	No
[53]	-Combined the URL and DNS information, Used a deep neural network with the help of NLP, Automatically extracts hidden features	7M	96%	No
[55]	-Studied five types of domain squatting, Using a dataset of over 224 million registered domains, Using visual and OCR analysis, Found new phishing instances that evaded common blacklist	234M	N/A	No
[54]	-Detecting algorithmically generated domain, Graph-based algorithm to extract the dictionaries that are being used to generate algorithmically domains	80K	99%	No
[57]	-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
[56]	-Extract the features from both URL and HTML of the page -Not using third-party services	50K	97%	No
[67]	-Companies can define their phishing detection mechanism and protect the customers -Can be used as a complimentary service besides other detection approaches	1.3K	96.34%	No
*	-Evaluate the performance of existing datasets including [25–27, 57] -Using multiple classifiers and comparing the results	2-10K	81-95%	No
*	- 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

Chapter 6

Using Adversarial Autoencoder for Generating Samples in Phishing Detection

6.1 Proposed approach

In this section, we begin by modeling an attacker and then discuss how to get data that mimics the one that may be produced by him taking into account his capabilities. We then propose our AAE to generate synthesized samples of phishing and legitimate instances.

6.1.1 AAE for synthesized data generation

We utilize the adversarial autoencoder (AAE) for synthesizing both phishing and legitimate samples that mimic respective websites. The AAE is capable of generating both continuous and discrete data distributions. Therefore, AAE is a perfect fit for generating discrete feature sets of datasets described in Subsection 6.2.1. The high-level architecture of the AAE is depicted in Figure 6.1. The autoencoder derives a compressed knowledge representation of the original input, which reconstructs the same data distribution.

$$q(z) = \int_x q(z|x)p_d(x)dx \tag{6.1}$$

An aggregated posterior distribution of $q(z)$ on the latent code is defined with the encoding function $q(z|x)$ and the data distribution $p_d(x)$ as shown in Eq. 6.1 where x denotes the real phishing dataset. In this work, we synthesize phishing and legitimate samples separately, where we train two different AAEs for each dataset.

The operating principle of AAE is that the autoencoder seeks to minimize the reconstruction error while the adversarial network attempts to minimize the adversarial cost. The *Reconstruction phase* and *regularization phase* are two simultaneous phases that arise during training. In the

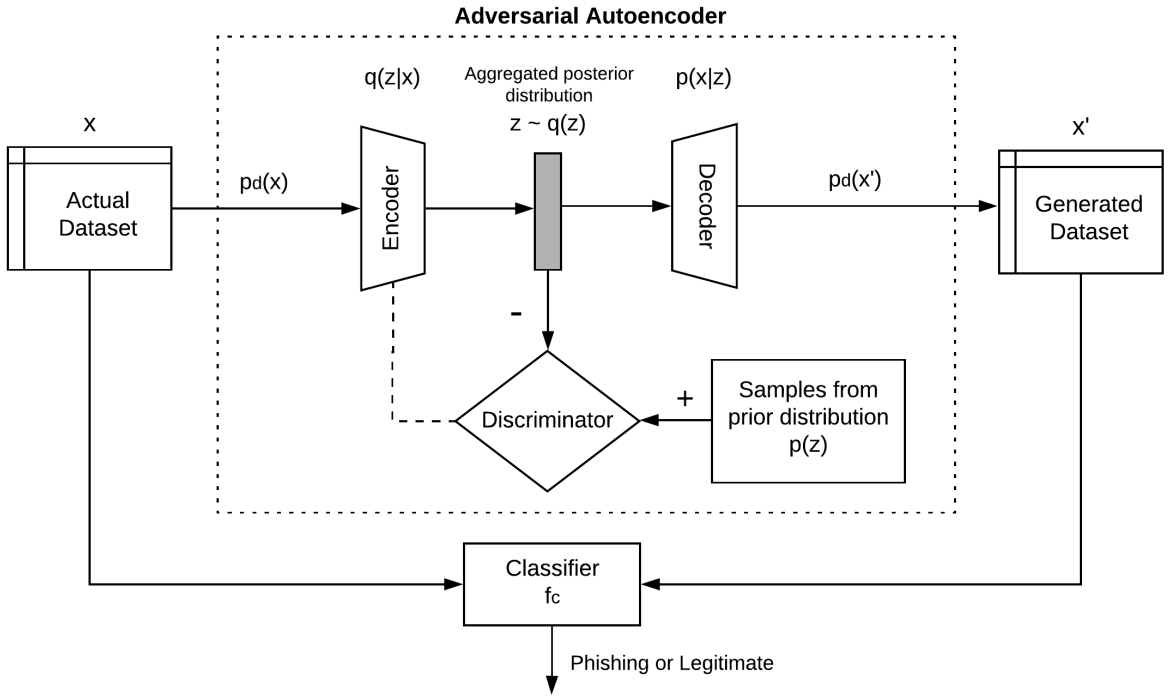


Figure 6.1: The high-level architecture of our proposed approach. It includes an adversarial autoencoder (AAE) that generates synthesized data using both phishing and legitimate data. The top row represents the standard autoencoder that reconstructs the data from the latent code z. The next row shows the discriminative network that predicts whether the samples emerge from the hidden code of the autoencoder $q(z)$ or the user-defined prior distribution $p(z)$ [95]. $p_d(x)$ denotes the data distribution. $q(z|x)$ and $p(x|z)$ denote the encoding and decoding distributions respectively. After the data generation, a machine learning classifier (f_c) described in Subsection 6.1.3 is applied for different classification tasks.

reconstruction phase, the autoencoder's data reconstruction error is minimized, often referred to as the loss. The regularization phase relates to the adversarial component of the network. It minimizes the adversarial cost to fool the discriminator by maximally regularizing an aggregated posterior distribution $q(z)$ to the prior $p(z)$ distribution.

The simultaneous training process allows the discriminative adversarial network into believing that the samples from hidden code $q(z)$ come from the prior distribution $p(z)$ [95]. A normal distribution is exploited as the arbitrary previous $p(z)$ in this work. After the training process, the adversarial network synthesizes samples similar to the actual samples through the prior distribution $p(z)$.

In this work, we synthesize phishing and legitimate samples separately, where we train two different AAEs for each dataset because each has different sets of distinct features. The feature values are varied in many value ranges. Thus, the values are normalized between -1 and 1 before feeding to the encoder and are denormalized after data generation from the decoder.

After the generation of both phishing and legitimate samples separately, we integrate them into a single dataset. Therefore, in the end, we have two datasets: *Original Dataset*, which has both phishing and legitimate samples and has been used to generate adversarial samples and a new *Synthesized Dataset* that consists of new synthesized both phishing and legitimate samples.

The synthesized dataset has the characteristics of phishing samples generated by real-world attackers and the characteristics of legitimate samples collected from real websites. We feed them into a classification algorithm that can recognize phishing samples from legitimate ones. This classifier is unaware of whether the samples are synthesized, which means generated by adversarial, or actual, which means we got them from an existing dataset. The instances are labeled as legitimate or phishing, and the classifier will predict them subsequently.

The use of synthesized samples solves multiple purposes at the same time. We increase the dataset size and alleviate the problem of data unavailability and data collection. The various areas like phishing detection and other investigative domains that are difficult to gather data then suffer from insufficient data for further analysis. The samples that have been generated by that AAE will be injected into our training set with respective labels. That helps to make the existing learning algorithm resilient against adversarial attacks. In this work, we defined five pairs of hypotheses scores. We verified our hypotheses in Section 6.2 in the experimental study.

6.1.2 Experimental methodology

We define five hypotheses on the goals we introduced in Section 1.2.3. We define five scores, one for each hypothesis, that measuring those scores will empirically evaluate each hypothesis and prove them.

Hypothesis-1. It is essential to make sure our classification algorithm holds acceptable performance on the dataset without considering any synthesized samples. Hypothesis-1 states if classification algorithms can reproduce the performance close to the performance reported by original authors of datasets. In other words, we evaluate if our classification algorithms outperform, or at least be as good as, the original author's report. In case the authors do not report any results, the performance needs to be in an acceptable range. We train and test our classifiers without considering synthesized samples and compare results with the authors' results to prove this hypothesis. Δ^1 evaluates Hypothesis-1 as defined as follows.

Δ^1 is the difference in performance between the performance reported by the original authors of the dataset and the best performance we got in our experiments. Positive values or close to zero are desired as it proves the performance of our model is better or close to what the original authors reported.

Hypothesis-2. The second hypothesis states that synthesized samples (generated by AAE) should be similar to original samples. Thus, it is unlikely that a machine-learning-based algorithm can distinguish synthesized samples from original samples. For this purpose, a classification algorithm is targeting to detect synthesized samples from original samples on both phishing and legitimate samples without regard if samples are phishing or legitimate.

Δ^2 is the performance when classifiers are trained on original and synthesized samples. We calculate Δ^2 in both cases the best performance, Δ^2_{Max} and average performance, Δ^2_{Avg}. Average performance on different classifiers is important as a classifier could be successful on few datasets but fails on some others.

Hypothesis-3. The third hypothesis states that synthesized samples are more likely to evade the classifier and being mislabeled. In other words, synthesized phishing samples will be incorrectly labeled as legitimate more than original phishing samples. While this could happen for synthesized legitimate samples as well. This is one of the most critical goals of our research as we want to show current classification algorithms are vulnerable to this adversarial sampling attack.

For evaluating this hypothesis, we will test synthesized samples with classification algorithms that have been trained only with original samples. This guarantees algorithms do not have information about synthesized samples. We will then compare these results with the performance of when the algorithms were tested with original samples as Δ^3 score.

Δ^3 specifies the different performance of a model when it is tested on synthesized with original samples. The model is trained only with original samples, composing both phishing and legitimate. The lower values for Δ^3, the more the classifier and dataset are vulnerable against synthesized samples and proves Hypothesis-3.

Hypothesis-4. The fourth hypothesis is referring to our mitigation approach. Our initial hypothesis is re-training classifiers on datasets that have been injected with synthesized samples will recover the performance of models from Hypothesis-3. For this purpose, we defined the following score of Δ^4.

Δ^4 calculates the difference between the performance of a classifier when it was tested on synthesized samples, but it was trained on only original samples with models that have been trained on a combination of original and synthesized samples. Higher values on Δ^4 are desired.

Hypothesis-5. In Section 1.2.3 we explained data gathering is a difficult task in an adversarial context like phishing detection. One of the usefulness of our proposed approach is to increase the size of the training set without needing to gather real data. This hypothesis states that injecting synthesized samples into the training set will increase the performance of the classifiers on original samples. That helps to improve the performance of the models without doing the data gathering. Δ^5 has been defined for this purpose.

Δ^5 defines the improvement of the performance over original samples. For this purpose, we calculate the difference between the performance of two classifiers when it is tested with only original samples, but once it is trained on only original samples and once it is trained on both original and synthesized samples. This score helps to understand if adding synthesized samples.

Table 6.1 summarizes hypothesis and scores we defined in this section.

Table 6.1: Summary of the hypothesises and scores

Hypothesis		Score					
Name	Description	#	Op_1 Trn	Op_1 Tst	Op_2 Trn	Op_2 Tst	Trg
Hyp-1	Reproducing datasets authors performance	Δ^1	Org	Org	Org	Org	Phi vs Leg
Hyp-2	Distinguishing synthesized and original samples	Δ^2	Syn/Org	Syn/Org	–	–	Syn vs Org
Hyp-3	Performance of synthesized samples	Δ^3	Org	Org	Org	Syn	Leg vs Phi
Hyp-4	Recovery performance for synthesized samples	Δ^4	Org	Syn	Org/Syn	Syn	Leg vs Phi
Hyp-5	Increase performance of original samples	Δ^5	Org/Syn	Org	Org	Org	Leg vs Phi

6.1.3 Machine learning classifier

For training purposes, we use eight different classifiers available in the Scikit-learn tool [88]. The classifiers that we use are *Decision Tree* (DT), *Gradient Boosting* (GB), *k-Nearest Neighbors* (KNN), *Random Forest* (RF), Gaussian Naive Bayes (GNB), and *Support Vector Machine* with two kernels: *Linear* (SVM(l)) and *Gaussian* (SVM(r)) kernel. For each experiment, we optimized specific parameters of each classifier to get the best results and prevent overfitting.

6.2 Experiments and evaluation

In this section, we conduct a set of experiments to empirically prove the hypothesizes we defined in Section 6.1. We start with introducing datasets we have used, followed by experiments we have conducted for each hypothesis and the results we got in our experiments.

6.2.1 Summary of phishing datasets

In our experiments, we use nine phishing datasets publicly available on the Internet. For each dataset, we summarized the number of each group of labels (phishing or legitimate) and the total number of instances in the dataset. In addition, we explained the number and types of features in each dataset. Types of features are important as it explains in what types of datasets our algorithms can be run. In addition, we reported the highest performance reported by the original authors of the dataset, when it was available, for our comparison in the next step.

DS-1: Shirazi *et al.* [57] published their unbiased phishing dataset that focuses on a subset of domain-name-based features without third-party inquires in 2018. The reported accuracy varies between 0.97-0.98 on the validation set and unknown live phishing URLs.

DS-2: Rami *et al.* [26] created this dataset in 2012 and shared it with the UCI machine learning repository [94]. Authors detected characteristics that distinguish phishing websites from legitimate ones, like long URL, IP address in URL, adding prefix and suffix to domain and request URL, *etc.* In the next steps, the authors defined a set of 30 features that are divided into five categories: *URL-based*, *abnormal-based*, *HTML-based*, *JavaScript-based*, and *domain-name-based* features. Finally, the authors analyzed what features are the most significant ones regarding the detection algorithm. However, we used all 30 features in our experiments, regardless of the importance of features. The authors reported 0.922 for accuracy on this dataset.

DS-3: In 2014, Abdelhamid *et al.* [25] shared their dataset on the UCI machine learning repository [94]. 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. The best accuracy reported for this dataset is 0.97.

DS-4: This dataset has been created by Tan *et al.* [27] and was published on Mendeley [2] dataset library. This dataset includes 48 features, a combination of URL-based and HTML-based features. The authors integrated a feature selection phase with a training phase and chose the 10 best features with random forest classifiers. However, similar to DS-2, we used all features in our experiments. The best performance achieved in this model is 0.96.

DS-5: Hannousse *et al.* [96] released this dataset that includes more than 11 thousand phishing and legitimate URLs with 87 extracted features from three different categories: 56 extracted from the structure and syntax of URLs, 24 extracted from the page contents, and 7 are extracted by querying external services. The dataset is balanced, and 50% of phishing and the other 50% of legitimate instances. The best accuracy reported by the authors is 0.966.

DS-6: Vrbancic *et al.* [97, 98] generated this dataset with 111 URL-based features without considering the contents of webpages or using third-party services. This dataset includes more than 146 thousand instances of phishing and legitimate websites. Unfortunately, we could not find any reported accuracy for this dataset.

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

Table 6.2: Summary of datasets we used in our experiments including released year, author's reported performance, dataset size (number legitimate, phishing, and total instances), number of features, and type of features including URL-based, Page-content-based, and inquiring third-party services.

Author			Size (K)			Features			
Name	**Year**	**Prf.**	**Leg.**	**Phi.**	**Tot.**	**No.**	**URL**	**Pg. Cnt.**	**3rd. Pt.**
DS-1	2018	0.98	1	1.2	2.2	7	✓	✓	
DS-2	2012	0.922	6.2	4.9	11.1	30	✓	✓	
DS-3	2014	0.97	0.6	0.7	1.3	9		✓	✓
DS-4	2018	0.96	5.0	5.0	10.0	48	✓	✓	
DS-5	2020	0.966	5.7	5.7	11.4	87	✓	✓	✓
DS-6	2020	-	58.0	30.7	88.7	111	✓		
DS-7	2020	0.983	5.9	7.2	13.1	35		✓	
DS-8	2020	0.970	2.2	1.8	4.0	76	✓	✓	
DS-9	2020	-	7.8	7.6	15.8	79	✓		

DS-7: Moruf *et al.* [99,100] released a dataset of phishing and legitimate websites with 35 features. The authors added features related to image identity, page style, layout identity, and text identity. There are more than 13 thousand instances, 7200 instances labeled as phishing, and 5800 labeled as legitimate. The authors reported an accuracy of 0.983.

DS-8: Mahmodi *et al.* [101, 102] gathered a dataset of legitimate and phishing websites with more than 8000 instances. 75 URL-, and content-based features compromised this dataset. For this dataset, the authors reported an accuracy of 0.970.

DS-9: Muhammad *et al.* [103] created a dataset of phishing and legitimate instances with 15000 instances. All of the 79 features are URL-based features. Similar to DS6, we could not find any reported performance for this dataset as well.

Table 6.2 summarizes the number of instances, features, and the portion of legitimate vs. phishing instances in each dataset. In addition, we specified each of these datasets have URL-based, page-content-based, and third-party-based features.

6.2.2 Hyp-1: reproducing author's results

In this experiment, we evaluate Hypothesis-1 as it questions if our proposed method can reproduce the performance close to the performance reported by original authors of datasets without considering any synthesized samples. We used 80% of data for training purposes and 20% for testing in five-fold cross-validation. We ran all experiments ten times and reported the mean accuracy. The results are reported in Table 6.3.

Table 6.3: Evaluation of the model against different classifiers with two metrics of ACC.

	DT	GNB	GB	KNN	RF	SVC(l)	SVM(r)	Auth.	Δ^1_{Acc}
DS1	0.959	0.946	0.968	0.962	**0.971**	0.95	0.955	0.98	-0.009
DS2	0.967	0.597	0.972	0.948	**0.973**	0.932	0.972	0.922	$+0.051$
DS3	0.92	0.896	0.924	0.932	0.932	0.900	**0.94**	0.97	-0.03
DS4	0.973	0.85	0.984	0.873	**0.986**	0.946	0.951	0.96	$+0.026$
DS5	0.942	0.72	0.961	0.843	**0.967**	0.937	0.871	0.966	$+0.001$
DS6	0.953	0.839	0.957	0.877	**0.971**	0.917	0.756	-	-
DS7	0.991	0.911	0.991	**0.993**	0.991	0.939	0.963	0.983	$+0.01$
DS8	0.99	0.92	**0.999**	0.979	0.996	0.988	0.956	0.970	$+0.029$
DS9	0.972	0.821	0.983	0.97	**0.987**	0.963	0.956	-	-
Avg.	0.963	0.833	0.971	0.931	**0.975**	0.941	0.924	0.964	$+0.011$

Table 6.3 summarizes the accuracy scores we achieved for all 9 datasets and 7 classification algorithms. In addition, it expresses reported accuracy by authors, except for datasets DS6, and DS9, which we could not find reported accuracy by original authors. For calculating Δ^1_{Acc}, we selected the maximum accuracy we got among 7 classifiers (declared in bold font) and then subtracted it from the reported performance by authors.

Positive values for Δ^1_{Acc} show our model outperformed the accuracy of original authors. For 5 out of 7 datasets, we are reporting positive Δ^1_{Acc} values. While for two datasets of DS1 and DS3, we are reporting negative values for Δ^1_{Acc}, those are not statistically significant. In addition, the best average performance we got among classifiers, belonging to RF classifiers, is higher than the average performance reported by original authors by 0.011. These results prove Hypothesis-1.

6.2.3 Hyp-2: distinguishing synthesized samples

In this experiment, we evaluate to see if synthesized samples are distinguishable from original samples, based on Hypothesis-2. For each dataset, our AAE has generated 10K phishing and 10K legitimate samples. We trained our set of classifiers on original samples, as positive labels, and synthesized samples, as negative labels. Similar to the previous experiment, we used 80% of data for training purposes and 20% for testing in five-fold cross-validation. While some of the datasets are imbalanced, we are best performing F1 score as $\Delta^2_{Max\_F1}$ in Table 6.4.

Table 6.4: Performance of models to evaluate Hypothesis-2. Each model is trained on original and synthesized data to distinguish between those samples. F1 score has been reported.

	DT	**GNB**	**GB**	**KNN**	**RF**	**SVC(l)**	**SVM(r)**	$\Delta^2_{Max\_F1}$
DS1	0.53	0.0	**0.543**	0.462	0.51	0.0	0.47	0.543
DS2	0.829	0.303	0.881	0.725	**0.907**	0.098	0.899	**0.907**
DS3	0.127	0.0	0.142	0.144	**0.153**	0.0	0.139	0.153
DS4	0.942	0.334	0.962	0.89	**0.975**	0.542	0.937	0.975
DS5	0.999	0.237	0.998	0.745	**0.999**	0.744	0.096	0.999
DS6	0.993	0.317	0.99	0.903	**0.998**	0.827	0.515	0.998
DS7	0.902	0.452	0.931	0.857	**0.972**	0.171	0.787	0.972
DS8	0.991	0.257	0.992	0.738	**0.996**	0.284	0.753	0.996
DS9	0.998	0.309	0.999	0.924	**0.999**	0.887	0.931	0.999
Avg.	0.812	0.245	0.826	0.71	**0.834**	0.395	0.614	

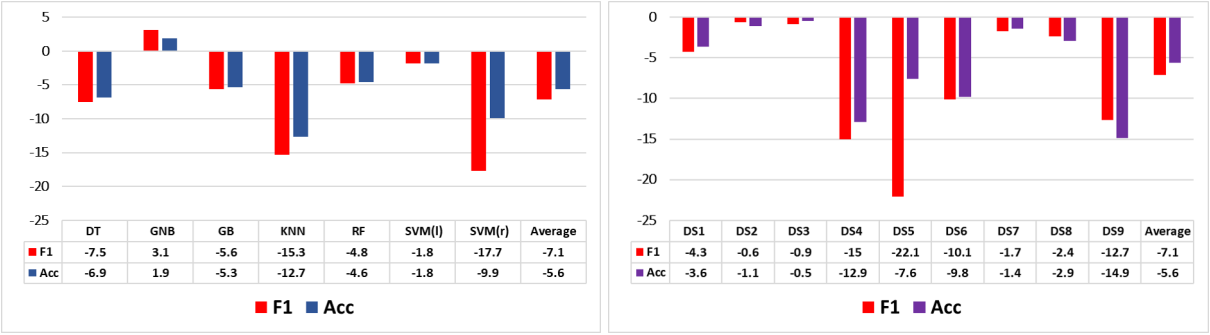
We fine-tuned all classifiers over datasets in this experiment.

The lower $\Delta^2_{Max\_F1}$ scores demonstrate the lack of ability to distinguish synthesized samples from legitimate and support our Hypothesis-2. Table 6.4 summarizes $\Delta^2_{Max\_F1}$ scores as we defined in Section 6.1. For each pair of classifiers and dataset, we report an F1 score. In addition, we report the average and best F1 scores for each classifier and dataset respectively. As Table 6.4 shows, the best F1 score for DS1 and DS3, declared by $\Delta^2_{Max\_F1}$, are very low. For other datasets, the $\Delta^2_{Max\_F1}$ are reasonably high, with the lowest value of 0.907 belonging to DS2. While Δ^2 holds significant results, the average of different classifies over all of our datasets are very low. The highest average

score belongs to the RF classifier with 0.834. These results show that while classifiers may be successful in discriminating synthesized samples from original in some datasets, the average results are not acceptable and proves our hypothesis that synthesized samples are difficult to be detected from original ones on average.

6.2.4 Hyp-3: performance against synthesized samples

To prove Hypothesis-3, we checked the performance of classifiers against synthesized samples, when algorithms are trained only with original samples. That shows if synthesized samples can by-pass the models and exploit vulnerabilities. We then calculated Δ^3, as the different performance of classifiers trained only on original samples and tested on synthesized samples and original samples. For reporting results, we averaged the difference between the performance of models in Section 6.2.2 and when models were tested against synthesized samples on different datasets. That number shows how performance between these two testing set changes, either increasing or decreasing on average. We have tested our models with 2000 synthesized phishing samples and 2000 legitimate samples, 4000 samples in total. For this experiment, we injected 80% of our synthesized sample (both phishing and legitimate samples) into the training set and re-trained.



(a) (b)

Figure 6.2: (a) Performance of classifiers against synthesized samples when algorithms are only trained on original samples. (b) Difference of performance when models were tested on original and synthesized samples are calculated and depicted in this graph.

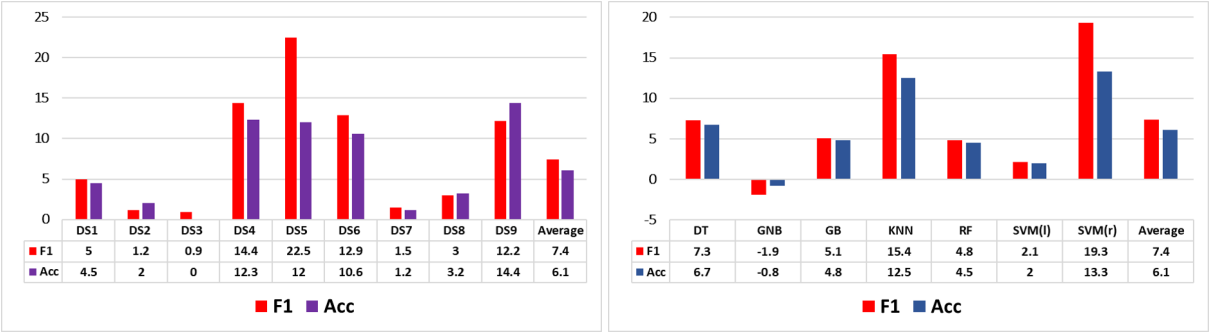
Figure 6.2 depicts Δ^4_{Acc} and Δ^4_{F1} scores for different classifiers and datasets. On average, the Δ^4_{F1} score has been decreased by 7.1% and Δ^4_{Acc} by 5.6%; a clear sign of synthesized samples can evade the classifier. Among different datasets, DS4, DS5, and DS9 have more decrease in performance, with the heights downgrade for DS5 with 22% and 7% for Δ^4_{F1} and Δ^4_{Acc} respectively. For different classifiers, SVM with a linear kernel and KNN has a significant downgrade in performance. Δ^4_{F1} score has been downgraded 17.7% and 15.3% for linear SVM and KNN respectively.

These results demonstrate our synthesized phishing and legitimate samples were able to evade trained classifiers; a clear sign of vulnerability for models for both classification algorithms and datasets we experimented. In addition, it proves the strength of our algorithm is not limited to a specific algorithm or dataset and our algorithm is successful in wide ranges of datasets and classification algorithms.

6.2.5 Hyp-4: mitigating against adversarial samples

In this experiment, we evaluated our mitigation approach. We injected synthesized samples to measure if the performance can increase. That proves Hypothesis-4. For that, we defined Δ^4, which is the difference of performance when algorithms were trained only with original samples and when was trained with both original and synthesized samples. For each dataset, we injected 80% synthesized samples into the training set and reserved 20% for testing. Similar to the previous experiment, Figure 6.3 explains the results of experiments for Δ^4_{F1} and Δ^4_{Acc} based on classifier and datasets.

Figure 6.3 depicts improvements in performance that happened after injecting synthesized samples into the training set. That helped the performance to recover significantly. We made the following observations. Δ^4_{F1} and Δ^4_{Acc} are significantly high when it was averaged on datasets and classifiers. Δ^4_{F1} and Δ^4_{Acc} scores, for all cases, have been improved but the GNB classifier. In addition, the recovery is similar to the performance downgrade in the previous experiment that means the downgrade of performance that was because synthesized samples have been fully recovered.



(a) (b)

Figure 6.3: This experiment evaluates recovery from synthesized attacks. The results show the performance of testing with synthesized samples when the model was trained with synthesized samples and were not trained with those samples when datasets were tested (a) and when classifiers were tested (b).

The average of Δ^4_{F1} and Δ^4_{Acc} are 7.4% and 6.1% respectively, while the decrease in performance in the previous experiment was 7.1% and 5.6%. That shows performance has been recovered to the same amount it was degraded in the previous experiment. These results prove Hypothesis-4 as the proposed approach was able to recover the performance to the level before using synthesized samples.

6.2.6 Hyp-5: improving performance for original Samples

In previous experiments, we showed injecting synthesized samples into the training set mitigates vulnerability against synthesized samples. However, this was not the only benefit of using our proposed approach. As data gathering is a laborious activity, we want to analyze if extending the size of the dataset with synthesized samples improves the performance of models when those are tested with original samples, in comparison with algorithms when those were trained only with original samples. We calculate Δ^5 as this difference. Tables 6.6 and 6.5 summarize results for two scores of Δ^5_{F1} and Δ^5_{Acc}.

As Tables 6.6 and 6.5 summarize, the average performance for two scores of Δ^5_{Acc} and Δ^5_{F1} have been slightly improved, 0.089 and 0.156 respectively. As original performance for these models is significantly high, even a small improvement is considerable. Besides, the improvement

Table 6.5: Improving performance of each classifier and dataset on original samples with Δ^5_{Acc}.

Dataset	DT	GNB	GB	KNN	RF	SVC(l)	SVM(r)	Avg.
DS1	1.1	0.5	0.5	1.6	0.3	0.9	1.1	0.9
DS2	-0.9	7.1	-1.1	0.7	0.2	0	0.2	0.9
DS3	0.4	-0.8	0	-1.2	-1	0.4	-1.6	-0.5
DS4	-1.7	-0.7	-1.3	0.9	-0.4	-0.9	-0.4	-0.6
DS5	-0.2	1.1	-0.6	-2.9	0.2	ERR	0	-0.4
DS6	0.1	2.2	-0.4	0.3	0.1	ERR	2	0.7
DS7	0	-0.9	-0.2	0	-0.1	-0.2	-0.4	-0.3
DS8	-0.1	0.7	-0.4	-0.2	0.1	0.9	1.2	0.3
DS9	-0.1	-1.3	-0.2	0	-0.2	-0.9	1.4	-0.2
Avg.	-0.156	0.878	-0.411	-0.089	-0.089	0.029	0.389	0.089

Table 6.6: Improving performance of each classifier and dataset on original samples with Δ^5_{F1}.

Dataset	DT	GNB	GB	KNN	RF	SVC(l)	SVM(r)	Avg.
DS1	0.7	0.4	0.4	1.4	0.3	0.8	1	0.7
DS2	-0.9	4.5	-1.2	1	0.3	0.1	0.3	0.6
DS3	0.9	-0.2	0.4	-0.7	-0.5	0.9	-1.1	0
DS4	-1.5	-0.2	-1.2	1	-0.4	-0.8	-0.3	-0.5
DS5	0	0.9	-0.5	-1.5	0.3	ERR	-7.8	-1.4
DS6	0.2	5.6	-1.3	-0.1	0.2	ERR	10	2.4
DS7	0	-0.6	-0.2	0	-0.1	-0.2	-0.3	-0.2
DS8	0	1.6	-0.4	-0.1	0.2	1	1.5	0.5
DS9	0	-2.5	-0.7	-0.3	0	-0.9	-0.8	-0.7
Avg.	-0.067	1.056	-0.522	0.078	0.033	0.129	0.278	0.156

for specific algorithms is significant. For example, GNB accuracy has been increased by 7% when it was trained on DS2 and by 2.2% on DS6.

Having said that, the performance improvement that has been demonstrated by Δ^5_{Acc} and Δ^5_{F1} scores proves Hypothesis-5. In other words, extending the dataset with synthesized samples helped to improve the performance of the system for both synthesized and original samples.

6.3 Conclusion

Supervised machine learning is a promising approach for phishing detection. However, sufficient volumes of data regarding phishing websites are unavailable and often infeasible to obtain. Towards this end, we demonstrated how Adversarial Autoencoders can be used for synthesizing samples that mimic data of real phishing websites. We compared the similarity of the features and instances of the generated data to ensure that the generated data may be realistically generated by the attacker. We used nine publicly available datasets for our experiments. Our experiments revealed that the learning algorithms work better when they are trained with larger volumes of data. Injecting synthesized data in the training set improved the performance of the learning algorithms. Moreover, the learning algorithms that included some synthesized data also were significantly more robust to exploratory attacks.

Chapter 7

Conclusion

In this dissertation, we described the approach towards the design of *only* domain-name-based features for the detection of phishing websites using machine learning. Our feature design emphasized the elimination of the possible bias in classification due to differently chosen datasets of phishing and legitimate pages. Our approach differs from all previous works in this space as it models the relationship of the domain name to the intent of phishing.

We then proposed a fingerprinting approach that detects whether a phishing website is attacking a target legitimate website. Our approach considers both visual and textual characteristics and detects phishing websites based on how closely their features match these fingerprints. That addressed two issues of bias towards more popular websites and adaptability to new attacks in existing algorithms.

In the next step, we studied the vulnerability of existing phishing detection approaches against adversarial sampling attacks. We showed that newly generated samples with a small number of perturbed features can bypass existing models. In addition, we observed 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 at the adversary to select more efficient feature manipulation to evade the classifiers.

Finally, we demonstrated how Adversarial Autoencoders can be used for synthesizing phishing samples. Our experiments revealed that the learning algorithms work better when they are trained with larger volumes of data. Injecting synthesized data in the training set improved the performance of the learning algorithms. Moreover, the learning algorithms that included some synthesized data also were significantly more robust to exploratory attacks.

Bibliography

[1] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973–993, 2014.

[2] S. Moore, *Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019*, Gartner, accessed July 13, 2020. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to\-exceed-124-billion-in-2019

[3] D. Milkovich, *15 Alarming Cyber Security Facts and Stats*, Cybint, accessed July 13, 2020. [Online]. Available: https://www.cybintsolutions.com/cyber-security-facts-stats/

[4] L. P. Kelly Bissell, *The Cost of Cybercrime*, Accenture, accessed July 13, 2020. [Online]. Available: https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf#zoom=50

[5] *Cyber Risk Analitics*, Risk Based Security, accessed July 13, 2020. [Online]. Available: https://pages.riskbasedsecurity.com/2019-midyear-data-breach-quickview-report

[6] G. Bassett, C. D. Hylender, P. Langlois, A. Pinto, and S. Widup, *Data Breach Investigations Report*, Verizon, accessed July 13, 2020. [Online]. Available: https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf

[7] Wikipedia, "Phishing — Wikipedia, the free encyclopedia," 2016, [Online; accessed 21-October-2016]. [Online]. Available: https://en.wikipedia.org/wiki/Phishing#cite_note-APWG-10

[8] G. Ho, A. Cidon, L. Gavish, M. Schweighauser, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner, "Detecting and characterizing lateral phishing at scale," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1273–1290.

[9] F. B. of Investigation (FBI), "Business e-mail compromise 12 billion dollar scam," https://www.ic3.gov/media/2018/180712.aspx, (accessed July 2, 2020).

[10] S. Singh, A. K. Sarje, and M. Misra, "Client-side counter phishing application using adaptive neuro-fuzzy inference system," in *International Conference on Computational Intelligence and Communication Networks*, 2012.

[11] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Conference on Human Factors in Computing Systems*, 2006.

[12] A. Handa, A. Sharma, and S. K. Shukla, "Machine learning in cybersecurity: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1306, 2019.

[13] A. Niakanlahiji, B.-T. Chu, and E. Al-Shaer, "Phishmon: A machine learning framework for detecting phishing webpages," in *IEEE International Conference on Intelligence and Security Informatics*, 2018, pp. 220–225.

[14] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from urls," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.

[15] J. Mao, J. Bian, W. Tian, S. Zhu, T. Wei, A. Li, and Z. Liang, "Phishing page detection via learning classifiers from page layout feature," *Journal on Wireless Communications and Networking*, 2019.

[16] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommunication Systems*, vol. 68, no. 4, pp. 687–700, 2018.

[17] J. Kirchner, A. Heberle, and W. Löwe, "Classification vs. regression-machine learning approaches for service recommendation based on measured consumer experiences," in *IEEE World Congress on Services*, 2015.

[18] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing url detection via cnn and attention-based hierarchical rnn," in *IEEE International Conference On Trust, Security And Privacy In Computing And Communications*. IEEE, 2019, pp. 112–119.

[19] P. Saravanan and S. Subramanian, "A framework for detecting phishing websites using ga based feature selection and artmap based website classification," *Procedia Computer Science*, vol. 171, pp. 1083–1092, 2020.

[20] E. Zhu, Y. Ju, Z. Chen, F. Liu, and X. Fang, "Dtof-ann: An artificial neural network phishing detection model based on decision tree and optimal features," *Applied Soft Computing*, vol. 95, p. 106505, 2020.

[21] S. Abt and H. Baier, "Are we missing labels? a study of the availability of ground-truth in network security research," in *International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. IEEE, 2014, pp. 40–55.

[22] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (sok): A systematic review of software-based web phishing detection," *IEEE Communications Surveys Tutorials*, 2017.

[23] J. Zou and L. Schiebinger, "Ai can be sexist and racist—it's time to make it fair," 2018.

[24] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial classification," in *International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, pp. 99–108.

[25] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948–5959, 2014.

[26] R. M. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *International Conference for Internet Technology and Secured Transactions*. IEEE, 2012, pp. 492–497.

[27] C. L. Tan, "Phishing dataset for machine learning: Feature evaluation," 2018, https://data.mendeley.com/datasets/h3cgnj8hft/1 (Accessed 2019-05-12).

[28] H. Shirazi, B. Bezawada, I. Ray, and C. Anderson, "Adversarial sampling attacks against phishing detection," in *Data and Applications Security and Privacy*. Springer International Publishing, 2019.

[29] M. Jensen, A. Durcikova, and R. Wright, "Combating phishing attacks: A knowledge management approach," in *50th Hawaii International Conference on System Sciences*, 2017.

[30] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Anti-Phishing Phil: The Design and Evaluation of a Game That Teaches People Not to Fall for Phish," in *3rd Symposium on Usable Privacy and Security*, ser. SOUPS '07. New York, NY, USA: ACM, 2007, pp. 88–99.

[31] N. A. G. Arachchilage, S. Love, and K. Beznosov, "Phishing threat avoidance behaviour: An empirical investigation," *Computers in Human Behavior*, vol. 60, pp. 185–197, 2016.

[32] D. D. Caputo, S. L. Pfleeger, J. D. Freeman, and M. E. Johnson, "Going spear phishing: Exploring embedded training and awareness," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 28–38, 2014.

[33] ——, "Going spear phishing: Exploring embedded training and awareness," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 28–38, 2013.

[34] S. Afroz and R. Greenstadt, "Phishzoo: Detecting phishing websites by looking at them," in *IEEE International Conference on Semantic Computing*. IEEE, 2011, pp. 368–375.

[35] A. K. Jain and B. Gupta, "A novel approach to protect against phishing attacks at client-side using auto-updated white-list," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 9, 2016.

[36] Wikipedia, "Mozilla. phishing protection," 2017, [Online; accessed 04-April-2017]. [Online]. Available: https://support.mozilla.org/t5/Protect-your-privacy/How-does-built-in-Phishing-and-Malware-Protection-work/ta-p/9395

[37] A. L. H. N. L. H. N. Security, "Inside Google's Global Campaign to Shut Down Phishing." [Online]. Available: https://www.wired.com/2017/05/inside-googles-global-campaign-shut-phishing/

[38] R. B. Basnet, S. Mukkamala, and A. H. Sung, "Detection of phishing attacks: A machine learning approach," in *Soft Computing Applications in Industry. Studies in Fuzziness and Soft Computing*. Springer, 2008, vol. 226, pp. 373–383.

[39] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious urls," in *ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 1245–1254.

[40] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "An evaluation of machine learning-based methods for detection of phishing sites," in *International Conference on Neural Information Processing*. Springer, 2008, pp. 539–546.

[41] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *ACM International Conference on World Wide Web*. ACM, 2007, pp. 639–648.

[42] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah, "Predicting phishing websites using classification mining techniques with experimental case studies," in *International Conference on Information Technology: New Generations*. IEEE, 2010, pp. 176–181.

[43] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Transactions on Information and System Security*, vol. 14, no. 2, pp. 1–28, 2011.

[44] R. Verma and K. Dyer, "On the character of phishing urls: Accurate and robust statistical learning classifiers," in *ACM Conference on Data and Applications Security and Privacy (CODASPY)*, 2015, pp. 111–122.

[45] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommunication Systems*, pp. 1–14, December 2017.

[46] M. Al-Janabi, E. d. Quincey, and P. Andras, "Using supervised machine learning algorithms to detect suspicious urls in online social networks," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2017, pp. 1104–1111.

[47] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know your phish: Novel techniques for detecting phishing sites and their targets," in *IEEE International Conference on Distributed Computing Systems*. IEEE, 2016, pp. 323–333.

[48] S. Marchal, G. Armano, T. Grondahl, K. Saari, N. Singh, and N. Asokan, "Off-the-hook: An efficient and usable client-side phishing prevention application," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1717–1733, 2017.

[49] G. Armano, S. Marchal, and N. Asokan, "Real-time client-side phishing prevention add-on," in *IEEE International Conference on Distributed Computing Systems*. IEEE, 2016, pp. 777–778.

[50] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Computing and Applications*, January 2018.

[51] S. R. Team, "ISTR Internet Security Threat Report Volume 23," 2018. [Online]. Available: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/istr-23-cyber-security-threat-landscape

[52] R. Verma and K. Dyer, "On the character of phishing urls: Accurate and robust statistical learning classifiers," in *Data and Application Security and Privacy*, 2015, pp. 111–122.

[53] J. Jiang, J. Chen, K.-K. R. Choo, C. Liu, K. Liu, M. Yu, and Y. Wang, “A deep learning based online malicious url and dns detection scheme,” in *Security and Privacy in Communication Systems*, 2017, pp. 438–448.

[54] M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, “Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic,” in *Research in Attacks, Intrusions, and Defenses*, 2018, pp. 295–314.

[55] K. Tian, S. T. Jan, H. Hu, D. Yao, and G. Wang, “Needle in a haystack: Tracking down elite phishing domains in the wild,” in *Internet Measurement Conference*, 2018, pp. 429–442.

[56] Y. Li, Z. Yang, X. Chen, H. Yuan, and W. Liu, “A stacking model using url and html features for phishing webpage detection,” *Future Generation Computer Systems*, vol. 94, pp. 27–39, 2019.

[57] H. Shirazi, B. Bezawada, and I. Ray, “"kn0w thy doma1n name": Unbiased phishing detection using domain name based features,” in *Access Control Models and Technologies*, 2018, pp. 69–75.

[58] S. Afroz and R. Greenstadt, “PhishZoo: Detecting Phishing Websites by Looking at Them,” in *IEEE Fifth International Conference on Semantic Computing*, Sep. 2011, pp. 368–375.

[59] T.-C. Chen, S. Dick, and J. Miller, “Detecting visually similar web pages: Application to phishing detection,” *ACM Transactions on Internet Technology*, Jun. 2010.

[60] A. Y. Fu, L. Wenyin, and X. Deng, “Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover’s Distance (EMD),” *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 301–311, Oct. 2006.

[61] R. S. Rao and S. T. Ali, “A computer vision technique to detect phishing attacks,” in *International Conference on Communication Systems and Network Technologies*, 2015, pp. 596–601.

[62] H. Zhang, G. Liu, T. W. S. Chow, and W. Liu, "Textual and visual content-based anti-phishing: A bayesian approach," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1532–1546, 2011.

[63] R. Zhao, S. John, S. Karas, C. Bussell, J. Roberts, D. Six, B. Gavett, and C. Yue, "The highly insidious extreme phishing attacks," in *International Conference on Computer Communication and Networks*. IEEE, 2016, pp. 1–10.

[64] ——, "Design and evaluation of the highly insidious extreme phishing attacks," *Computers & Security*, vol. 70, pp. 634 – 647, 2017.

[65] N. Chou, R. Ledesma, Y. Teraguchi, J. C. Mitchell *et al.*, "Client-side defense against web-based identity theft," in *Computer Science Department, Stanford University*, 2004.

[66] Q. Cui, G.-V. Jourdan, G. V. Bochmann, R. Couturier, and I.-V. Onut, "Tracking phishing attacks over time," in *International Conference on World Wide Web*, 2017, pp. 667–676.

[67] V. Bulakh and M. Gupta, "Countering phishing from brands' vantage point," in *International Workshop on Security And Privacy Analytics*, 2016, pp. 17–24.

[68] X. Han, N. Kheir, and D. Balzarotti, "Phisheye: Live monitoring of sandboxed phishing kits," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1402–1413.

[69] APWG, "Global phishing survey: Trends and domain name use in 2016," 2017, [Online; accessed 21-October-2016]. [Online]. Available: http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf

[70] V. Drury and U. Meyer, "Certified phishing: taking a look at public key certificates of phishing websites," in *Symposium on Usable Privacy and Security*, 2019, pp. 211–223.

[71] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, "Hunting malicious tls certificates with deep neural networks," in *Proceedings of ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 64–73.

[72] R. Dai, C. Gao, B. Lang, L. Yang, H. Liu, and S. Chen, "Ssl malicious traffic detection based on multi-view features," in *Proceedings International Conference on Communication and Network Security*, 2019, pp. 40–46.

[73] Q. Cui, G.-V. Jourdan, G. V. Bochmann, I.-V. Onut, and J. Flood, "Phishing attacks modifications and evolutions," in *European Symposium on Research in Computer Security*. Springer, 2018, pp. 243–262.

[74] C. N. Gutierrez, T. Kim, R. Della Corte, J. Avery, D. Goldwasser, M. Cinque, and S. Bagchi, "Learning from the ones that got away: Detecting new forms of phishing attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 988–1001, 2018.

[75] A. Van Der Heijden and L. Allodi, "Cognitive triaging of phishing attacks," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1309–1326.

[76] S. Marchal, J. François, T. Engel *et al.*, "Proactive discovery of phishing related domain names," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2012, pp. 190–209.

[77] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *ACM Workshop on Security and Artificial Intelligence*, 2011, pp. 43–58.

[78] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning*, 2015, pp. 1689–1698.

[79] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, pp. 984–996, 2014.

[80] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[81] Y. Wang, S. Jha, and K. Chaudhuri, "Analyzing the robustness of nearest neighbors to adversarial examples," in *International Conference on Machine Learning*, 2018, pp. 5120–5129.

[82] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "cleverhans v1. 0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, vol. 10, 2016.

[83] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (sok): A systematic review of software-based web phishing detection," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2797–2819, 2017.

[84] N. Abdelhamid, F. A. Thabtah, and H. Abdel-jaber, "Phishing detection: A recent intelligent machine learning comparison based on models content and features," in *IEEE International Conference on Intelligence and Security Informatics*, 2017, pp. 72–77.

[85] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *ACM Workshop on Recurring Malcode*. ACM, 2007, pp. 1–8.

[86] C. L. Tan, K. L. Chiew, K. Wong, and S. N. Sze, "Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder," *Decision Support Systems*, vol. 88, no. C, pp. 18–27, August 2016.

[87] W. Zhang, Q. Jiang, L. Chen, and C. Li, "Two-stage elm for phishing web pages detection using hybrid features," *World Wide Web*, vol. 20, no. 4, pp. 797–813, July 2017.

[88] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine Learning research*, 2011.

[89] E. Ma, "Secret of google web-based ocr service," Jan 2019. [Online]. Available: https://towardsdatascience.com/secret-of-google-web-based-ocr-service-fe30eecedd01

[90] F. C. Dalgic, A. S. Bozkir, and M. Aydos, "Phish-iris: A new approach for vision based brand prediction of phishing web pages via compact visual descriptors," in *International Symposium on Multidisciplinary Studies and Innovative Technologies*. IEEE, 2018, pp. 1–8.

[91] Alexa, "amazon.com competitive analysis, marketing mix and traffic," 2020. [Online]. Available: https://www.alexa.com/siteinfo/amazon.com

[92] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *arXiv preprint arXiv:1712.03141*, 2017.

[93] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[94] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[95] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[96] S. Hannousse, Abdelhakim; Yahiouche, "Web page phishing detection," 2020, https://data.mendeley.com/datasets/c2gw7fy2j4/2(Accessed2020-09-12).

[97] G. Vrbančič, "Phishing dataset for machine learning: Feature evaluation," 2020, http://dx.doi.org/10.17632/72ptz43s9v.1 (Accessed 2020-10-12).

[98] G. Vrbančič, I. Fister, and V. Podgorelec, "Datasets for phishing websites detection," *Data in Brief*, vol. 33, p. 106438, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352340920313202

[99] M. Adebowale, “Phishing dataset for machine learning: Feature evaluation,” 2019, https://data.mendeley.com/datasets/gt7xdbs3kt/2(Accessed2020-09-12).

[100] M. Adebowale, K. Lwin, E. Sánchez, and M. Hossain, “Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text,” *Expert Systems with Applications*, vol. 115, pp. 300 – 313, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418304925

[101] H. Sadoghi Yazdi, e. mahmodi, and A. Ghaemi Bafghi, “Data for: An online minimal uncertainty drift-aware method for anomaly detection in social networking,” 2020, https://data.mendeley.com/datasets/zw7knrxpy5/1(Accessed2020-09-12).

[102] H. S. Yazdi, A. G. Bafghi *et al.*, “A drift aware adaptive method based on minimum uncertainty for anomaly detection in social networking,” *Expert Systems with Applications*, vol. 162, p. 113881, 2020.

[103] M. HANIF, “Malware webpages data,” 2020, https://data.mendeley.com/datasets/zsj5pgrsg9/1(Accessed2020-09-12).