2021 IEEE Conference on Communications and Network Security (CNS) | 978-1-6654-4496-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CNS53000.2021.9705052

"X-Phish: Days of Future Past"‡: Adaptive & Privacy Preserving Phishing Detection

Shalin Kumar Deval* Meenakshi Tripathi* Bruhadeshwar Bezawada† Indrakshi Ray‡
*Computer Science & Engineering Department, Malaviya National Institute of Technology,
Jaipur, Rajasthan, 302017, India. Email: {2019pcp5280, mtripathi}@mnit.ac.in
†Computer Science & Engineering Department, Indian Institute of Technology Jammu,
Jammu and Kashmir, 181221, India. Email: bez.bru@iitjammu.ac.in
‡Computer Science & Engineering Department, Colorado State University,
Fort Collins, Colorado 80526, USA. Email: Indrakshi.Ray@colostate.edu

***Abstract*—Website phishing continues to persist as one of the most important security threats of the modern Internet era. A major concern has been that machine learning based approaches, which have been the cornerstones of deployed phishing detection solutions, have not been able to adapt to the evolving nature of the phishing attacks. To create updated machine learning models, the collection of a sufficient corpus of real-time phishing data has always been a challenging problem as most phishing websites are short-lived. In this work, for the first time, we address these important concerns and describe an adaptive phishing detection solution that is able to adapt to changes in phishing attacks. Our solution has two major contributions. First, our solution allows for multiple organizations to collaborate in a privacy preserving manner and generate a robust machine learning model for phishing detection. Second, our solution is designed to be flexible in order to adapt to the novel phishing features introduced by attackers. Our solution not only allows for incorporating novel features into the existing machine learning model, but also can help, to a certain extent, the "unlearning" of existing features that have become obsolete in current phishing attacks. We evaluated our approach on a large real-world data collected over a period of six months. Our results achieve a high true positive rate of 97%, which is on par with existing state-of-the art centralized solutions. Importantly, our results demonstrate that, a machine learning model can incorporate new features while selectively "unlearning" the older obsolete features.**

***Index Terms*—Phishing detection, Privacy Preserving, Machine Learning, Adaptive, Collaborative Learning**

I. Introduction

A. Motivation

Phishing attacks on the world-wide web continue to contribute to massive financial losses and sensitive information leakage [1]. A phishing website clones a legitimate website and lures users into divulging sensitive information such as passwords, identity, and credit card numbers, among others. But, more than the financial losses, personal information leakage has far-reaching consequences for the victim of the phishing attack. Given the impact of these attacks on the safety and security of users, there is a critical need to deploy robust defenses against phishing attacks.

Machine learning approaches [1]–[16] have been quite effective in detecting phishing websites with minimal overhead on the user web-browsing experience. In general, a machine learning model works as a classifier to distinguish between a phishing website and a legitimate website. Therefore, these approaches require a substantial amount of training data, of both phishing and legitimate websites, and a set of well formulated features that help to construct an effective classifier. However, over the years, phishing attacks have adapted to the machine learning based defenses by targeting the building blocks of machine learning defenses, *i.e.*, data and feature manipulation. Therefore, there is a critical need for designing machine learning based phishing defenses that are resilient to such adaptive strategies of the attackers.

B. Problem Statement

The problem of website phishing detection is to determine if a website is a phishing or a legitimate site based on standard definitions in literature [17], [18]. The general template of a phishing detection solution is to first identify a good set of features that can discriminate between a phishing and legitimate website and, train a machine learning classifier on these features using the best possible sample data set available at that time period. We focus on the problem of phishing detection wherein the phishing websites show a *progressive adaptation* to defenses by masking, adding, perturbing, or removing features of interest in successive generations of phishing websites. Specifically, we address the problem of phishing detection wherein some new important features are not known upfront at the time of training the machine learning classifier and/or some existing features are no longer useful.

C. Limitations of Prior Art

Content-based approaches [1]–[16], [19] perform in-depth analysis of content to classify phishing websites. Uniform Resource Locator (URL)-based approaches [11]–[16] analyze various features based on the target URL such as length of the URL, page rank of the URL, presence of special characters

‡A reference to the movie *"X-Men: Days of the Future Past" (2014, Marvel) where the X-Men glimpse into the past to save their future.*

The work of Indrakshi Ray was supported in part by funds from NSF under award number CNS 1822118 and from NIST, Statnett, Cyber Risk Research, American Megatrends Inc., and Army Research Laboratory.

in the URL, IP address instead of host name, DNS features *etc.* 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. Despite their great success, we identified three key shortcomings of prior work:

- *Insufficient Data*. Most phishing websites are taken down within 48-72 hours, making it difficult for any single security analyst to collect a significant amount of data and multiple analysts may not be willing to share data. Also, data from public resources could be noisy.
- *New Phishing Features.* Attackers keep changing the style of phishing websites to adapt to the previous generation of defenses. This creates new features of interest for phishing detection and need to be considered.
- *Dynamic Updates to Machine Learning Models.* Once a machine learning model is deployed, it is difficult to update it without changing the entire model. Addressing this challenge is critical to adapt to the strategies of the phishing attackers.

As a consequence, we emphasize that the key shortcoming of prior art is the inability to adapt to feature modifications by the attackers. Research [7], [20] has shown that most existing classifiers can be ineffective if some or all of the features originally used for training are made obsolete or redundant.

D. Key Contributions

First contribution is that, our approach is *adaptive* to changes in phishing strategies. Our adaptive approach allows for feature addition and removal from deployed machine learning models. We illustrate that some phishing features in the past become obsolete as time passes on data spanning the last three years.

Second contribution is that, we describe an approach to generate machine learning models in a *privacy preserving* manner from disparate data sets, thereby, overcoming the challenges in data collection. Our approach ensures that multiple parties have access to the knowledge of multiple data sets without actually sharing the data sets and are able to build a robust machine learning model. We show that our approach achieves a high true-positive rate, for detecting phishing websites, of 97%, which is the state-of-the-art performance for many existing centralized solutions.

Our third and final contribution is that, during our investigation for adaptive features, we identified and engineered several new features that are not reported in literature thus far. This makes our approach robust and resilient against known and possibly future attacks.

II. Related Work

A. Phishing Detection with Standard Data

Cui *et al.* [7] 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 similar HTML Document Object Model (DOM) structure and over 90% of these attacks were actually replicas or variations of other attacks in the database. Hong *et al.* [19] created a data set to make use of the well-known term frequency inverse document frequency (TF-IDF) algorithm to find the top-5 important words in a web page and cross-checked using the Google® search engine. Zhang *et al.* [3] 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 legitimate web page and a suspicious web page. Miyamoto *et al.* [21] provide an overview of nine different machine learning techniques and analyzed the accuracy of each classifier on the CANTINA data set [19], reporting a maximum accuracy of 91.34% using AdaBoost. Xiang *et al.* [4] proposed a layered anti-phishing solution with a rich set of features based on the HTML DOM structure, search engine capabilities, and third-party services. Marchal *et al.* [6], [8] propose a client-side detection approach using proprietary data sets from Intel security. However, their approach uses over $200+$ features for classification, a factor that needs to be considered when deploying phishing detection solutions in a client browser. Hossein *et al.* [22], used domain-name based features to classify phishing websites with a positive detection rate of 97%. This model uses less number of features, but cannot adapt to newer features as standard statistical classifiers are utilized. In 2015, Verma *et al.* [13] described an approach based on textual similarity and frequency distribution of text characters in URLs. Recently, Rao *et al.* [23] proposed a heuristic URL classification technique where the input to their algorithm is constructed by domain + title or domain only of the given URL. They achieved an average positive detection rate of 99.77% for phishing sites. Some studies ventured into the nature of malicious content in online social networks. Al-Janabi *et al.* [14] described a supervised machine learning classification model to detect the distribution of malicious content in online social networks (OSNs). A good survey of phishing detection approaches can be found in [18].

B. Phishing Detection with Evolving Attackers

The attackers have constantly evolved their strategies to evade phishing detection mechanisms. An illustration is the use of the "HTTPS" protocol, which was not found in most phishing websites as shown by Hossein *et al.* [20], but has become more prevalent in modern phishing websites. Fernando *et al.* [24] have shown that how educating about the good old URL obfuscation techniques is not as effective an anti-phishing measure as it was against new URL obfuscation techniques like "Obfuscating with HTTPS schema" and "Obfuscating with Internationalized Domain names". Abuzuraiq *et al.* [25] implemented a fuzzy logic algorithm to achieve higher accuracy but the model becomes less and less accurate as more features are incorporated in the data set. There has been a rise in *extreme* phishing attacks [1] on financial institutions where the phishing website mimics the legitimate website to an alarming degree. The high level of noise in such websites is likely to defeat most content-based machine learning approaches. Mahdi *et al.* [26] proposed a framework for dynamic retraining of spam tweets detection model consisting of two kinds of machine learning models. The first one is a supervised model that classifies

the tweets as spam or not-spam. The second model is an unsupervised one that collects new tweets, annotates them using clustering algorithms, and prepares a new feature vector with a predefined set of 17 features. Once they have a sufficient collection of new tweets, the first model is retrained on the newly generated feature vector from the second model. This approach is semi-adaptive, in that, a model is able to learn new patterns within the same 17 features, but it cannot learn any *new* feature as the number and the name of the features are fixed. With time attackers learn to evade existing phishing detection models [7], [20] as certain features become obsolete over time and do not contribute to the classification.

Addressing the evolution of attacker strategies is an important challenge and offers a new direction of research. But, we make a cautious note that, it may not be feasible to arrive at a solution that can adapt to any kind of attacker strategy. We propose a first time approach that attempts to adapt to attacker strategies by learning and *unlearning* machine learning features over time.

III. Proposed Approach

A. Overview of Proposed Approach

The main goal of our approach is to construct a machine learning model that can detect a phishing website based on certain features while having the ability to adapt to newer features as time progresses. Our approach consists of three key steps. In the first step of feature engineering, we analyzed some interesting features of phishing websites over the last two years and discovered some changes in the phishing strategies. Our analysis has identified new features that are useful for phishing detection. In the second step of machine learning, we apply the collaborative learning approach described in [27] to generate the machine learning model. In this approach, multiple parties holding different data sets participate in a protocol for constructing a common machine learning model. Our choice of collaborative learning addresses the key challenge in phishing detection, *i.e.*, *the lack of sufficient data*, as multiple parties contribute to the learning process with their disparate data sets. The final step of our approach is the design of a feature vector that allows for "addition" or "removal" of features. The addition or removal of a feature is followed by retraining phase where the machine learning model is updated with new data from the new feature. The combination of collaborative learning and feature addition and removal, provide the important ability to adapt to evolving phishing strategies. This is the spirit of our $XPhish$ framework wherein the machine learning model attempts to retain the past while attempting to view the future.

B. Feature Engineering and Validation

As much as feasible, 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. 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. To validate the intuition behind each non-binary feature, we tested the empirical cumulative distribution function (ECDF) of the feature for 2000 phishing websites against 2000 legitimate websites. For binary features, we use our entire collected data of 40000 phishing and 40000 legitimate websites to show the distribution of the features. We also indicate if the features are "New", meaning designed by us, or "Existing", meaning that other researchers [3], [4], [6], [22], [28], [29] have designed it. In the following, we describe the existing and new features, identified by us, in this work.

Table I
Binary Feature Distribution

Feature	Legitimate	Phishing
HTTPS Present	0.97	0.55
Non-alphabetical Characters	0.02	0.13
Copyright Symbol present	1	0.19
Copyright year (2020/2021)	0.85	0.10
SSL Name and Domain Match	0.79	0.40
SSL Name and Copyright Match	0.65	0.01
Copyright-Domain Match	0.78	0.02
Suspicious Action attribute	0.006	0.18
Suspicious URL	0.06	0.35

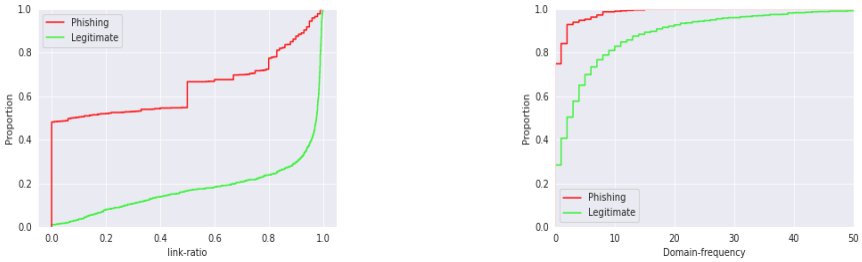
1) Feature 1 (Existing): Link Ratio in BODY: As described in [4], [22], this feature is defined as the ratio of the number of hyper-links pointing to the same domain to the total number of hyper-links on the web page. This feature is content-agnostic as the ratio can computed for any phishing website that exhibits this behavior. Figure 1(b), shows the ECDF of this feature, of the raw ratios, with sufficient separation between the two distributions.

2) Feature 2 (Existing): Frequency of Domain Name: As described in [22], this feature counts the number of times the domain name appears as a word in the visible text of the web page. This is a key feature that captures the visual relationship of the domain name to the web page. Note that, for classification purpose, we converted this feature into a binary feature, *i.e.*, if the domain name does not appear in the web page, we set it to 0 and if it appears, we set it to 1.

3) Feature 3 (Existing): HTTPS Present: Most legitimate websites use SSL certificates and operate over HTTPS protocol. Therefore, if a website uses HTTPS, the feature value is 1 and if not, it is 0. Table I summarizes the percentage distribution of the binary features in the sample data set. Recently, phishing websites are using HTTPS as well and this explains the relatively high distribution.

4) Feature 4 (Existing): Non-alphabetical Characters in Domain Name: Attackers use non-alphabetical characters, like numbers or hyphen, 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.

5) Feature 5 (Existing): Presence of Copyright Symbol: The copyright symbol is a mark of trust for the end user. Many legitimate websites use the copyright logo to indicate the trademark ownership of their organization name. If a copyright symbol is present in the web page, then this feature evaluates to 1, otherwise 0.



(a) Link Ratio in BODY (b) Domain Name Frequency

Figure 1. ECDF Plots of Link Ratio in BODY and Domain Name Frequency

6) Feature 6 (New): The year along with Copyright Symbol: For many, the copyright date is seen as a way to determine if the site is maintained or not. Some websites have just the most recent year mentioned alongside the copyright symbol while some others have a range of years from when they got first published. The intuition behind this is that attackers may use same old templates to create a fake web page or even if they do modify the look of the page, they will not bother to check the copyright information for a temporal feature. This feature evaluates to 1 if the copyright year is either 2020 or 2021, otherwise 0.

7) Feature 7 (Existing): Domain Name with Copyright Logo: Usually, the domain name is placed before or after the copyright logo for such websites. To generate this feature, we considered all the characters before and after the copyright logo, removed the white spaces, and checked for the presence of the domain name in the resulting string. We found that most of the phishing websites did not place their actual domain names along with the copyright logo. As shown in Table I, our intuition proved right, only 2% of the phishing websites were using this feature, but over 78% of legitimate websites had this feature.

8) Feature 8 (New): SSL Name and Copyright Name Match: Secure-Sockets Layer (SSL) Name is the value present in the *Issued to* or *Common name* field of the SSL certificate issued to the website owner. Many legitimate websites' SSL name matched with the organization's name present alongside the copyright symbol. The value of this feature is set to 1 if the SSL name and the name at the copyright symbol matches, otherwise, 0. As shown in Table I, our intuition proved right, only 1% of the phishing websites use this feature, but over 65% of legitimate websites have this feature.

9) Feature 9 (New): SSL Name and Domain Name Match: The SSL certificates include the domain name in the "issued to" field of the SSL/TLS certificate to which the certificate is issued to. SSL Common name mismatch arises when the SSL name does not match with the domain of the website in the address bar of the browser. This feature is set to 1 if there is a match, otherwise, 0.

10) Feature 10 (New): Suspicious Action attribute: Usually, phishing websites try to steal the user credentials through a form on the web page. These forms are then submitted to the desired location, generally, through *action* attribute. Many of phishing websites that had the form element, their *action* attribute had the following pattern: *filename.extension*. For example: *login.php*. Whereas, legitimate websites usually link to the URL where the form is submitted for processing. The value for this feature is set to 1 if there is a suspicious action attribute, otherwise 0.

11) Feature 11 (New): Suspicious URL: In our data set, a large number of phishing websites' URLs end with the page extension like $.html$ or $.php$. If any URL has such pattern, then this feature is set to 1, otherwise, it is set to 0. Table I shows that only 0.6% of legitimate websites have this feature while over 35% of phishing websites have suspicious URLs.

C. Collaborative Training Framework

Shokri *et al.* [27] proposed a distributed training technique, based on selective stochastic gradient descent and differential privacy. This framework form the basis of our machine learning model for phishing detection. The collaborative training framework assumes that there are N-participants in the training process. Each participant can be referred to as a local client or just participant. A common neural network architecture is agreed in advance by all of the participants. The parameter server is responsible for managing a list of global parameters which essentially represents the common model trained in collaboration by all the local clients.

Initially, each local participant i connects to the global parameter server and receives the structure of a neural network model, which will be the local model for this client. Each client maintains a list of local parameters, *i.e.*, weight-gradients and bias-gradients, which it can either initialize or leave them as it is. For participant i, this list is named as $w^{(i)}$.

Then, the local training begins where each participant trains the neural network on its own private data using an optimization algorithm such as Stochastic Gradient Descent (SGD) [27]. The training continues for many epochs until the required condition to halt the training is met. The local training is independent of any other participant in the process. They do not impact each other's models directly, rather indirectly via the parameter server. Following is the algorithmic process of local training on participant's side.

1) Initialize the model parameters and the learning rate α.
2) Repeat until the set number of epochs or the minimum error is achieved:

- After each epoch, compute the gradient vector $\Delta w^{(i)}$ on all the weights of the neural network as: $d\Delta w^{(i)} = w_j - w_i$, where w_j are the weights after training and w_i are the weights before the training.
- Select and upload Θ_u X $\left|w^{(i)}\right|$ most significant gradients to the server. These are the Θ_u largest values of the weight-gradients, from each layer of the neural network.
- Download the parameters from the server and replace the corresponding local parameters in the client's model.
- Run the next epoch of training on the model with the replaced parameters on the local data set and update the local parameters $w^{(i)}$ accordingly.

Each participant sorts the values inside the weight-gradient vector. Then, exactly Θ_u most significant values are picked from each layer, which contribute more towards the gradient descent, and are shared with the other participants.

The parameter server manages the global parameter vector Δw^g, of the common global model being trained by the participants, using the process shown here.

1) Initialize the global parameters $w^{(g)}$.
2) When a participant uploads the weight-gradients $w^{(i)}$: Update the global parameters by adding the corresponding values of $w^{(i)}$ and $w^{(g)}$ as: $\Delta w^g = \Delta w^g + \Delta w_j$, where the value of uploaded gradient for j^{th} parameter is Δw^g.
3) When a participant downloads the parameters $w^{(i)}$: Send all the parameters in the vector $w^{(g)}$ to the participant.

D. Adaptive Learning: The Null Feature Vector

Our approach for adaptive learning is to make the feature vector flexible in a way to incorporate the feature addition and removal without changing the architecture of the learning model. Let the feature vector be denoted as: $F = \{f_1, f_2, \cdots, f_{max}\}$ where max is the maximum length of F. Now, out of these features, at any give time only a few features might be relevant. Therefore, during the training process, the relevant features will have non-zero values and the remaining features will have zero values (or "null") values. Going further, if over time, a feature becomes less effective or useless for the classification purpose, instead of designing and training a new model, we render that feature column in the feature vector, as the *Null* feature.

Feature Removal We show an illustrative usage of null feature vector in the learning process in Figure 2. This figure shows how features can be removed. In the first phase, a model is trained on with a feature vector containing all relevant features, which can be less than Max features. Then, before the second phase, two *normal features* are converted to *null features* by replacing their original value with '0', the cells shown in red. Next, the existing model is trained on this updated feature set.

Feature Addition. Now, if any new features are required to be added into the feature vector, we replace any of the *null* feature with the new feature values shown in the dark green cell. We note that, while our approach of feature removal or addition may not be formally sound, we have been able to validate experimentally that it achieves the desired results as required for an adaptive phishing detection scheme.

IV. Performance Evaluation

A. Experimental Methodology

We implemented our approach using the PyTorch library in Python 3.8 on a desktop running Mint OS with Intel core® i5-5200U CPU© 2.7 GHz processor with 8 GB RAM. We have used multi-layer perceptron (MLP) as the common neural network architecture for every participant. MLPs are feed-forward neural network architectures in which neurons in each layer are fully connected to the neurons in the next layer. The neural network has 3 hidden layers and 1 output layer. The number of input neurons is 11, corresponding to the number of features. Number of neurons in 1st, 2nd, and 3rd hidden layers are 8, 10, and 32, respectively. The rectified linear activation function (ReLU) has been used on the hidden layers and the Sigmoid activation function on the output layer. In all the experiments, the batch size is 256 and learning rate is 0.01 with the Binary Cross Entropy (BCELoss) as the loss function and SGD as the optimization algorithm. The weights are initialized randomly but any initialization scheme can be used by participants. During the local training, the participants communicate asynchronously with the global parameter server. Once the participant's local model is updated with new weights downloaded from the server, the next training epoch starts. In all the experiments, the value of θ is set to 0.5.

We conducted four sets of experiments to assess the collaborative and adaptive performance of our model. The first set of experiments were conducted by training the model on all 11 features directly. The second set of experiments were conducted by incrementally adding a new feature in the data set. The third set of experiments were conducted by removing a feature from the data set. The fourth set of experiments included simultaneous addition and removal of a feature from the data set. During classification, we denote the phishing websites correctly classified by, true positive (TP) and incorrectly classified as legitimate sites by, false negatives (FN), and the legitimate sites correctly classified by, true negatives (TN) and incorrectly classified as phishing websites by, false positive (FP). We report standard classification metrics such as, positive predictive value, $PPV = \frac{TP}{TP+FP}$; true positive rate, $TPR = \frac{TP}{TP+FN}$; accuracy, $ACC = \frac{TP+FN}{TP+FP+FN+TN}$ and F-score, $F - score = \frac{2TP}{2TP+FP+FN}$.

B. Data Sets

For the list of legitimate websites, we obtained a total of 38500 websites from the `majestic.com` and assumed them as legitimate. For the phishing websites, we got a total 40000 phishing websites from `PhishTank.com`, `OpenPhish.com` and `isitphishing.com`.

C. Experiment 1: Performance on Collaborative Learning

In this set of experiments, we compare the performance of collaboratively trained model in a 4-participant system with a model trained in centralized way where whole data is pooled

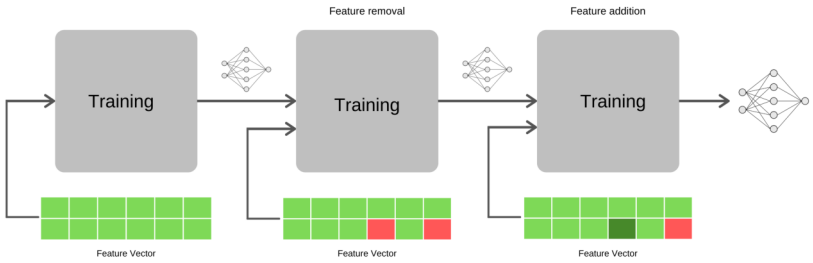


Figure 2. Training with null features

(a) Centralized training (b) Collaborative training

Figure 3. Comparison of Collaborative learning with Centralized learning

(a) 50% sharing (b) 100% sharing

Figure 4. Performance on varying the fraction of shared weights

at one location for training. The results are shown in Figure 3. We observe that even with 50% of weight-gradients sharing, our model's average performance is really close to the other one. For example, the average TPR of collaboratively trained model is 96.8% while the other model achieved a TPR of 97.25%.

In the above experiments, we kept the value of Θ_u as 0.5. Figure 4 shows the performance comparison for a 4-client system in two scenarios: first, when Θ_u is set to 0.5, and second, when Θ_u is set to 1, *i.e.*, all the weight-gradients are shared among the participants. We observe a slight increase in the average performance when all the weights are shared. For example, the average value of TPR jumped from 96.8% to 97.03%. This shows that even when only a fraction of parameters are shared, we get high-performing models as most significant parameters from each layer are getting shared among participants.

D. Experiment 2: Performance on adding new features

We designed four different experiments to evaluate the performance of the model when new features are added to the data set. Each one was carried out in 3 different collaborative environments; first with 2 participants, second with 4 participants, and third with 8 participants. Data was divided uniformly, *i.e.*, equal sharing among different participants. We have performed experiments with unbalanced data sets as well with similar results, In the first experiment, we selected 6 features to train the model. The values for remaining 5 features were kept as 0 (*value*). Once the training is completed, each client can disconnect from the server and test the model on their own private data set. In the second experiment, to show the adaptive nature of training, we included a new feature in the data set by restoring a previously *null feature's* original values. Then, the trained model from the first experiment was collaboratively trained again on this new data set. Similarly, in the third experiment, we included two more features in the data set and trained the model from second experiment, on this new data set. In the fourth experiment, we collaboratively trained the previous model on all the 11 features. All experiments were repeated in 5 trials for each collaborative scenario, *i.e.*, 2, 4,

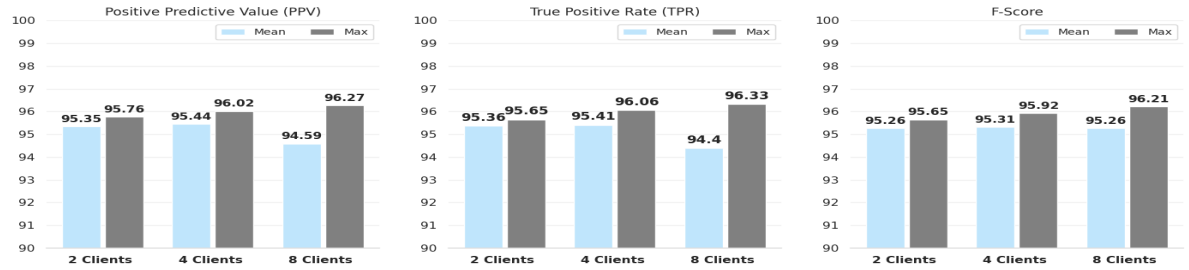


Figure 5. Performance of collaborative model with 6 normal features and 5 *null features*

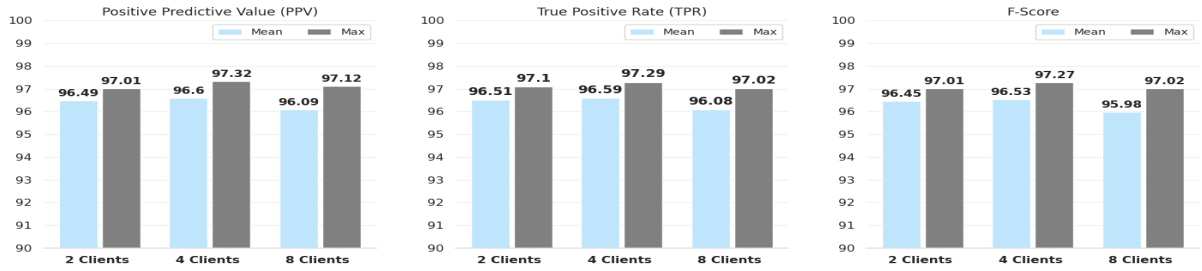


Figure 6. A feature is added by replacing one *null feature*; 7 normal and 4 *null* features.

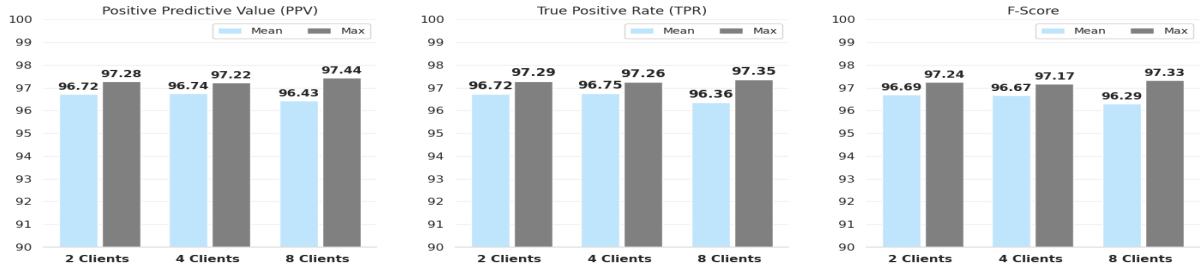


Figure 7. Two more features are added; 9 normal and 2 *null* features

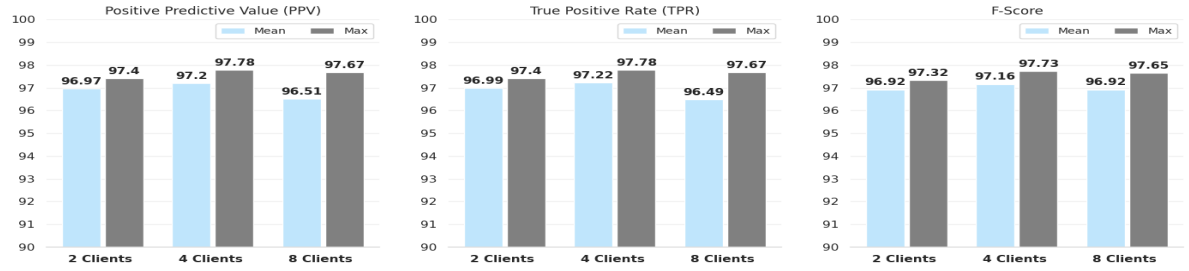


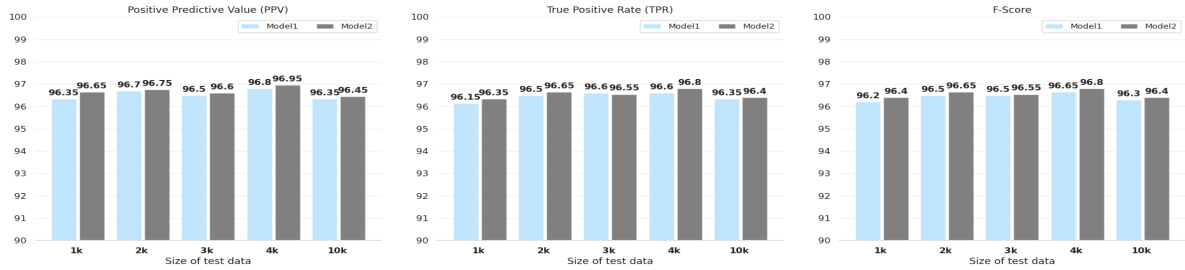
Figure 8. PPV, TPR and F-Score on training on all 11 features.

and 8 participants. To assess the overall performance, we took average and maximum of the aforementioned metrics across individual client's model performance in each trial.

Results when new features are added: The results show that our model is able to learn the new features using our *null features* approach as the performance gets better when a new feature is added to the data set. We show the results in Figures 5, 6, 7, and 8. In all the experiments, all three metrics, PPV, TPR and F-score (accuracy results were similar), were averaging around $96 - 97\%$, and the maximum values were over 97%. These values demonstrate the proof-of-concept of the proposed adaptive feature vector notion and show that the phishing detection model can be updated periodically, subject to the maximum length of the feature vector The results are for epochs up to 50; beyond this, we can still manage to get a slightly higher performance parameters.

E. Experiment 3: Performance for Feature Removal

The aim for this experiment is to show that our model can adapt its learning on a data set in which a feature, which was previously present, is now removed. For this purpose, we first trained our model with all the features in a 2-participant environment where each participant had 20000 samples of



(a) PPV (b) TPR (c) F-Score

Figure 9. *Unlearning* performance of collaboratively trained model

training data. Then we replaced the values of the feature *HTTPS* with 0 (thus, making it a *null feature*). After that, we collaboratively trained the previous model on this new and modified data set. This experiment corresponds to the scenario where a feature becomes obsolete after some time, and no longer helps in learning. The decision to remove *HTTPS* feature was taken mainly due to following reasons:

- Statistics [30] show that *HTTPS is no longer a strong classification feature for phishing websites.*
- *In our data set as well, a lot of phishing websites have* HTTPS in the URL.

To assess the adaptive performance in this set of experiments, we compared the results of the first model that was trained on data set without the HTTPS feature, to that of the second model, which was first trained on all the features and then, trained again on data set without "HTTPS" feature (10 features only).

Results when a feature is removed: The results show that our model is able to unlearn the feature that was removed from the data set. The results were collected for test data consisting of 1000, 2000, 3000, 4000, and 10000 data samples. We show the results in Figure 9. For each of the performance parameters, we compared the average results of aforementioned models for varying sized test data. Furthermore, it shows that there isn't any significant drop in the performance of the model when it unlearns a not-so-helpful feature. Instead, we noticed a slight improvement in PPV, TPR, and F-score after the *unlearning*, in some cases. The maximum values were above 97% and the average remained around $96 - 97\%$.

F. Experiment 4: Performance on simultaneous addition and removal of a feature

Finally, both addition and removal of a feature was performed simultaneously in the data set. For this purpose, we first trained a model on 10 features only, in 2-participant environment with 20k training data each. When the training was completed, we made the following changes in the data set. Added the feature "copyright year" and removed the "https" feature due to the reasons stated in the above section. After that, the previous model was collaboratively trained on this modified data set. To assess the performance of our model on simultaneous addition and removal of features, we compared the results of our final model (Model2) with the model (Model1) that was trained with sequential modification of data set, *i.e.*, the model was first trained on 10 features data set, then on data set with "copyright year" feature added, and finally, on the data set with the "https" feature removed.

Results when simultaneous addition and removal of features is performed. The results were collected for test data consisting of 1000, 2000, 3000, 4000, and 10000 samples. They show that the performance of Model1 and Model2 is almost similar in every case as it should be. We show the results in Figure 10. For each of the performance parameters, we compared the average results of Model1 and Model2 for varying sized test data. The metrics averaged between $95 - 97\%$ for both models, which demonstrates that the model is able to add and remove features without noticeable difference in performance. Some outliers and corner-cases might exist and this will require deeper exploration of newer data sets in the future.

V. Conclusion

In this work, for the first time, we describe a collaborative phishing detection approach that has the ability to adapt to evolving phishing strategies. Our approach uses a combination of collaborative learning with a flexible feature vector design to achieve this goal. We validated our experiments on data collected over the last 6 months and dating back up to two years. We also discover and engineer new phishing features that were hitherto unexplored in this domain. Our model achieves a high TPR of 97%, which is comparable to the state-of-the-art centralized phishing detection approaches. However, as we mentioned in a cautionary note, our approach is only the first attempt and adapting to phishing strategies remains a non-trivial challenge that needs further research and exploration.

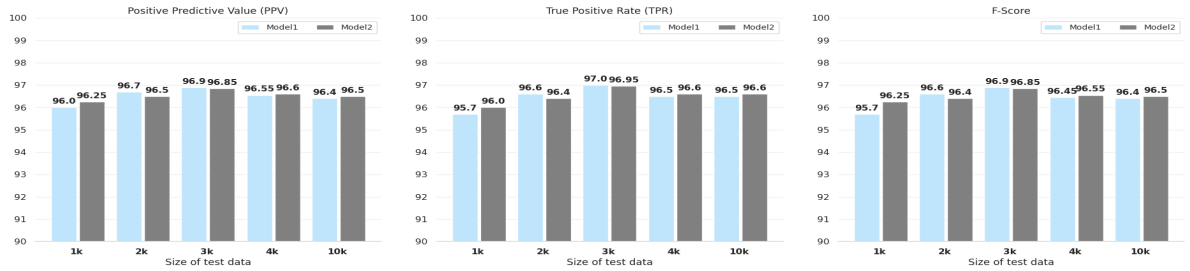
References

[1] Rui Zhao, Samantha John, Stacy Karas, Cara Bussell, Jennifer Roberts, Daniel Six, Brandon Gavett, and Chuan Yue. Design and evaluation of the highly insidious extreme phishing attacks. *Computers & Security*, 70:634 – 647, 2017.

[2] Ram B. Basnet, Srinivas Mukkamala, and Andrew H. Sung. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry. Studies in Fuzziness and Soft Computing*, volume 226, pages 373–383. Springer, 2008.

[3] Haijun Zhang, Gang Liu, Tommy W. S. Chow, and Wenyin Liu. Textual and visual content-based anti-phishing: A bayesian approach. *IEEE Trans. on Neural Networks*, 22(10):1532–1546, 2011.

[4] Guang Xiang, Jason Hong, Carolyn P. Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Information and Systems Security (TISSEC)*, 14(2):1–28, September 2011.



(a) PPV (b) TPR (c) F-Score

Figure 10. Simultaneous addition and removal of features.

[5] R. Gowtham and Ilango Krishnamurthi. A comprehensive and efficacious architecture for detecting phishing webpages. *Computers and Security*, 40:23–37, February 2014.

[6] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. In *Proc. of IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, pages 323–333. IEEE, 2016.

[7] Qian Cui, Guy-Vincent Jourdan, Gregor V Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In *Proc. of the Int. World Wide Web (WWW) Conf.*, pages 667–676, 2017.

[8] Samuel Marchal, Giovanni Armano, Tommi Grondahl, Kalle Saari, Nidhi Singh, and N. Asokan. Off-the-hook: An efficient and usable client-side phishing prevention application. *IEEE Trans. on Computers*, 66(10):1717–1733, 2017.

[9] Ankit Kumar Jain and B. B. Gupta. Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, pages 1–14, December 2017.

[10] Routhu Srinivasa Rao and Alwyn Roshan Pais. Detection of phishing websites using an efficient feature-based machine learning framework. *Neural Computing and Applications*, January 2018.

[11] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. A framework for detection and measurement of phishing attacks. In *Proc. of the ACM Workshop on Recurring Malcode (WORM)*, pages 1–8. ACM, 2007.

[12] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious urls. In *Proc. of the ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 1245–1254. ACM, 2009.

[13] Rakesh Verma and Keith Dyer. On the character of phishing urls: Accurate and robust statistical learning classifiers. In *Proc. of ACM Conf. on Data and Applications Security and Privacy (CODASPY)*, pages 111–122, 2015.

[14] Mohammed Al-Janabi, Ed de Quincey, and Peter Andras. Using supervised machine learning algorithms to detect suspicious urls in online social networks. In *Proc. of the IEEE/ACM Int. Conf. on Advances in Social Network Analysis and Mining (ASONAM)*, pages 1104–1111, 2017.

[15] Choon Lin Tan, Kang Leng Chiew, KokSheik Wong, and San Nah Sze. Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems*, 88(C):18–27, August 2016.

[16] Wei Zhang, Qingshan Jiang, Lifei Chen, and Chengming Li. Two-stage elm for phishing web pages detection using hybrid features. *World Wide Web*, 20(4):797–813, July 2017.

[17] 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*, 19(4):2797–2819, 2017.

[18] Neda Abdelhamid, Fadi A. Thabtah, and Hussein Abdel-jaber. Phishing detection: A recent intelligent machine learning comparison based on models content and features. In *Proc. of the IEEE Int. Conf. on Intelligence and Security Informatics (ISI)*, pages 72–77, 2017.

[19] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: A content-based approach to detecting phishing web sites. In *Proc. of the World Wide Web (WWW) Conf.*, pages 639–648. ACM, 2007.

[20] Hossein Shirazi, Bruhadeshwar Bezawada, Indrakshi Ray, and Chuck Anderson. Directed adversarial sampling attacks on phishing detection. *J. Comput. Secur.*, 29(1):1–23, 2021.

[21] Daisuke Miyamoto, Hiroaki Hazeyama, and Youki Kadobayashi. An evaluation of machine learning-based methods for detection of phishing sites. In *Proc. of the Int. Conf. on Neural Information Processing (ICONIP)*, pages 539–546. Springer, 2008.

[22] Hossein Shirazi, Bruhadeshwar Bezawada, and Indrakshi Ray. *Kn0w Thy Doma1n Name*: Unbiased phishing detection using domain name based features. In *23nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2018*, pages 69–75. ACM, 2018.

[23] Routhu Srinivasa Rao and Alwyn Roshan Pais. Jail-phish: An improved search engine based phishing detection system. *Computers and Security*, 83:246–267, 2019.

[24] Matheesha Fernando and Nalin Asanka Gamagedara Arachchilage. Why johnny can't rely on anti-phishing educational interventions to protect himself against contemporary phishing attacks?, 2020.

[25] A. Abuzuraiq, M. Alkasassbeh, and M. Almseidin. Intelligent methods for accurately detecting phishing websites. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 085–090, 2020.

[26] Mahdi Washha, Aziz Qaroush, Manel Mezghani, and Florence Sedes. Unsupervised collective-based framework for dynamic retraining of supervised real-time spam tweets detection model. *Expert Systems with Applications*, 135:129–152, 2019.

[27] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 909–910, 2015.

[28] Routhu Srinivasa Rao and Syed Taqi Ali. A computer vision technique to detect phishing attacks. In *Proc. of Int. Conf. on Communication Systems and Network Technologies (CSNT)*, pages 596–601. IEEE, 2015.

[29] D. Brites and M. Wei. Phishfry - a proactive approach to classify phishing sites using scikit learn. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2019.

[30] Apwg phishing activity trends reports, https://apwg.org/trendsreports/.