PHISHSIM: Aiding Phishing Website Detection With a Feature-Free Tool

Rizka Widyarini Purwanto, *Graduate Student Member, IEEE*, Arindam Pal, *Senior Member, IEEE*,
Alan Blair, and Sanjay Jha, *Senior Member, IEEE*

Abstract— In this paper, we propose a feature-free method for detecting phishing websites using the Normalized Compression Distance (NCD), a parameter-free similarity measure which computes the similarity of two websites by compressing them, thus eliminating the need to perform any feature extraction. It also removes any dependence on a specific set of website features. This method examines the HTML of webpages and computes their similarity with known phishing websites, in order to classify them. We use the Furthest Point First algorithm to perform phishing prototype extractions, in order to select instances that are representative of a cluster of phishing web-pages. We also introduce the use of an incremental learning algorithm as a framework for continuous and adaptive detection without extracting new features when concept drift occurs. On a large dataset, our proposed method significantly outperforms previous methods in detecting phishing websites, with an AUC score of 98.68%, a high true positive rate (TPR) of around 90%, while maintaining a low false positive rate (FPR) of 0.58%. Our approach uses prototypes, eliminating the need to retain long term data in the future, and is feasible to deploy in real systems with a processing time of roughly 0.3 seconds.

Index Terms— Phishing detection, webpage, incremental learning, feature-free methods.

I. INTRODUCTION

PHISHING is defined as a cyber-attack which uses social engineering via digital means to persuade victims to disclose their personal information, such as their password or credit card number [13]. The strategies used in phishing

attacks exploit human vulnerabilities in distinguishing between authentic and phishing messages or websites [29]. Phishing is a low-cost, yet essential tool to aid various cyber-attacks, as it is often used as the key step in advanced persistent threats. With our ever-increasing reliance on various digital platforms, phishing has become a versatile weapon in the attacker's arsenal. Despite the general definition of phishing, the term itself has been commonly associated specifically with Web phishing attacks that use emails or SMS as the attack vector to lure victims into submitting personal information via phishing websites or by downloading malicious software. These websites are typically crafted to look professional and convincing as if they are legitimate.

As mentioned in their report, the Anti-Phishing Working Group (APWG) recorded a considerable increase in unique phishing attacks from 2014 to 2016 [13], which caused significant financial losses, estimated to be between $60 million and $3 billion per year in the United States [24]. In another report, the APWG detected around 65,400 phishing websites per month in 2018 [14], while PhishLabs reported that phishing volume grew by 40.9% in 2018 compared to the previous years [32]. PhishLabs also reported how attack volumes continue to increase as actors' methods evolve and adapt to changes in the digital landscape. Furthermore, the use of free hosting providers has resulted in an increase in phishing attacks during the past four years, from 3.0% in 2015 to 13.8% in 2018. It is also relatively easy to set up phishing websites using phishing toolkits. The availability of these toolkits enables a single actor to create a large number of professional-looking phishing websites in a short period of time. As an example, PhishLabs reported a sharp increase in the number of attacks in August 2018. This phishing campaign used at least 2,000 freely-hosted phishing websites, all created using the same kit [32]. With the availability of these phishing kits provided by organized crime groups, it is likely that the number of phishing attacks will increase further in the future. The use of free hosts, phishing kits, and SSL certificates show that there has been a persistent effort by actors to capitalize upon new opportunities, resulting in the continuous growth of the number of phishing attacks from year to year. This makes it challenging to develop a reliable phishing detection method that can deal with the dynamic nature of the attacks [28].

To prevent the negative impacts of phishing, researchers have studied various methods in recent years to build an automated phishing website detection system by investigating the website content, appearance, URL, and other web-related features [21], [36]–[39], [41]. In general, these approaches

could be classified into two categories. The first approach finds intrinsic features of phishing websites and attempts to detect these attacks based on these specific characteristics. Many studies in recent years go along with this type of approach, using machine learning and deep learning methods. While these methods have shown to perform well in detecting phishing, their detection are less robust towards concept drift since they are heavily based on certain characteristics assumed to be related to phishing websites (e.g., specific types of web forms or unusual structures in the URLs) which may change and become irrelevant in the future. Meanwhile, the second approach tries to detect phishing by measuring similarities between phishing websites and the targeted legitimate website. This method is less robust to zero-day phishing attacks compared to the first approach. However, the similarity-based methods are effective for quickly filtering a large amount of phishing websites, before being fed into machine learning based methods which typically take more time to perform classification. A number of previous studies have proposed the use of various similarity metrics and models to detect near similar phishing websites. These similarity-based techniques usually require the websites to be modeled into a certain representation space, e.g., using DOM trees, bag-of-words, or Doc2Vec models [20].

In contrast, we attempt to propose a feature-free method for detecting phishing websites which uses the Normalized Compression Distance [17] to compute website HTML similarity. The rationale of the study is based on the work of Cui *et al.* [18], which reveals the commonalities between phishing websites, demonstrating that 90% of the 19,066 confirmed phishing websites from PhishTank [9] are replicas or variations of other already known phishing websites. In their study, Cui *et al.* proposes a new distance metric, the proportional distance, to assess website similarities for detecting phishing. This distance metric takes into account the number of occurrences of a predefined set of HTML tags, which are found to provide significant information whether a website is related to phishing attacks. On the other hand, the use of prototypes and the Normalized Compression Distance in our proposed method would eliminate the need of the predefined HTML tags, and uses compression algorithms to universally measure similarities between two streams of data based on the amount of information they have in common. Our proposed method is not limited to a specific kind of phishing attack or phishing campaign, nor a certain type of the phishing email. However, our method is limited to variations of phishing websites which have occurred at least once, and unable to identify a new phishing website with a completely different and distinctive HTML structure.

To summarize, this paper makes the following contributions:

- We introduce a systematic method to perform website similarity measurements for detecting similar phishing websites using *Normalized Compression Distance* (NCD).
- We provide an analysis on the similarities and differences between phishing and legitimate website contents and visual appearances, and how content-based methods

would effectively detect phishing attacks better than visual-based methods.

- We propose PhishSim as a tool to effectively detect slightly modified or near-similar phishing websites using prototype-based learning algorithms and the *Normalized Compression Distance*, which is a parameter-free and application independent distance metric to measure similarities between websites' HTML content. This tool works by measuring the pairwise similarity between websites in the dataset, clustering these websites, and performing phishing classifications based on whether a website is grouped in the same cluster with a known phishing website.
- We introduce a feature-free phishing detection system architecture that can be deployed within intranet servers or in the cloud.

The paper is organized as follows. Section II provides an overview of past studies which are related to our work in phishing detection systems. In section III, we introduce the normalized compression distance (NCD) and some background mathematical concepts which are used in this paper. Section IV describes the overview of our system. Section V provides results on the website similarity analysis to observe the characteristics between a phishing website and its legitimate target website based on the pairwise NCD value. Section VI provides results on the optimal distance threshold selection process. We describe the experimental setup in Section VII, then provide the performance evaluation results in Section VIII and further analysis on these results in Section IX. At the end of the paper, in Section X, we wrap up with conclusions.

II. RELATED WORKS

There is significant research focusing on phishing detection. Some studies focus on the use of blacklists and whitelists in anti-phishing systems. Blacklist-based methods keep a list of domain names or links to known phishing websites and alert users if they try to visit those sites. However, phishing websites are highly dynamic, and the average lifetime of a phishing webpage is only a few hours [12]. In many cases, zero-hour phishing attacks easily bypass blacklist-based methods. Meanwhile, whitelist-based approaches allow users to browse only those webpages that are deemed safe, which is often impractical.

Other studies in phishing detection use similarity-based methods to measure similarities between websites by analyzing a website's textual content or screenshot. Text similarity-based methods analyze the semantics of the textual content of emails and webpages to decide whether they classify as a phishing attempt. This method is likely to fail in the future with the increased use of code obfuscation techniques [25]. A past study by Chen *et al.* [16] introduced the use of the normalized compression distance (NCD) to measure visual similarities between webpages using the website screenshots. Their work is based on the assumption that phishing websites usually look almost identical to the legitimate website they are targeting. However, this method might not perform well when detecting phishing websites which

have a distinct appearance to the target website (Figure 6 and Figure 7). Cui *et al.* [18] proposed the proportional distance metric, which computes similarities between websites based on the number of occurrences of a predefined set of HTML tags. A more recent work by Feng *et al.* [20] studied the use of Doc2Vec, which is a deep learning based method to create a numerical representation of phishing websites, and performed phishing website detection based on the similarity of the website's numerical representation.

Several other studies have attempted to find intrinsic characteristics of phishing websites and use machine learning algorithms to perform classification based on these features. Whittaker *et al.* [36] introduced a classification system to maintain Google's phishing blacklist automatically based on features extracted from the website's URL, page hosting information, and page content. Furthermore, Zhang *et al.* [41] developed a framework, called CANTINA, to identify phishing targets using TF-IDF (Term Frequency-Inverse Document Frequency) analysis and seven other content-based heuristics, including domain age, logo image and domain name inconsistency, as well as suspicious links in the HTML. Purwanto *et al.* [33] designed a new compression-based algorithm PHISHZIP to detect phishing websites. Xiang *et al.* [37] improved CANTINA by proposing a more comprehensive framework to detect phishing websites named CANTINA+, which makes use of URL-based, HTML-based, and web-based features. Xiang and Hong [38] also proposed a hybrid phishing detection approach by extracting keywords and performing identity discovery with named entity recognition. Araujo and Martinez-Rico proposed a phishing detection system which combines the use of website link and language-model (LM)-based features [15]. Meanwhile, Zhang *et al.* [39] introduced a phishing website detection system that analyses the website's textual and visual content, and assesses the similarities. In a recent study, Quinkert *et al.* [34] attempted to identify scams and phishing by looking for homograph domains that use visually similar Unicode characters to create URLs that look identical.

Several past studies focused on using natural language processing (NLP) based techniques to perform phishing website classification. Zhang *et al.* [40] proposed the use of text semantic features together with statistical features from the website to improve Chinese phishing website detection. Meanwhile, Opara *et al.* [30] proposed HTMLPhish which uses convolutional neural networks to learn the feature representation of websites based on the semantic dependencies of the website's textual content. The use of semantic analysis has shown to perform well in identifying phishing and legitimate websites, since specific words has shown to occur more frequently in phishing websites [33], e.g. email, sign, account, password. However, the use of NLP based techniques limits the phishing detection to a specific language.

III. CONCEPTS AND DEFINITION

In this section, we introduce and discuss the use of normalized compression distance (NCD) to perform phishing website detection, and prototype-based learning algorithms for clustering and classifying websites.

A. NCD for Measuring Phishing Website Similarity

NCD is a parameter-free distance measure, which is universal such that it attempts to approximately measure the similarity of dominant features in all pairwise file or object comparisons. The aim of NCD is to capture each effective distance, including the effective versions of Hamming distance, Euclidean distance, and edit distance. Further details on NCD is provided in Appendix A. The generic characteristics of NCD make it applicable to various kinds of applications [17]. Based on these characteristics, we have set out to explore whether the use of NCD as a non-feature similarity metric is suitable in the study of phishing detection systems. With the dynamics of phishing, a detection system which relies on a specific and static set of features would potentially fail to detect phishing once the attack behavior changes.

Besides the selection of NCD as the similarity metric, another aspect to consider is the selection of object or data to be compared in the context of phishing detection. Chen *et al.* [16] use NCD to measure visual similarities between websites and detect phishing websites by calculating the NCD between two website screenshot images. Their study attempts to detect deceptive phishing attacks where the phishing sites are visually very similar to their target legitimate website. Chen *et al.* argue that while there are some small differences between phishing and legitimate websites, attackers must design the phishing page to be similar to the legitimate page in order to convince users to believe that the website is legitimate. Based on this assumption, they performed phishing website detection by computing the NCD between a suspicious website and the legitimate website. An NCD value below a certain threshold indicated that the website was indeed imitating the legitimate website, therefore categorizing it as a phishing website. While this assumption is true for some phishing websites, we found that in many other cases, phishing websites are not necessarily identical to their target website.

On the other hand, Cui *et al.* [18] demonstrated that 90% of the phishing data collected in a 10-month period in 2016 were variations or replications of other previous attacks in the database, which indicates that there are repetitions and similarities among the phishing websites themselves. This is also understandable as there has been an increase in the use of phishing kits which increase the possibility of similar HTML contents from new phishing websites. Thus, we take a new approach where we perform phishing detection by detecting similarities in the website HTML content, as phishing websites are often developed from a certain template or kit. Therefore, in this study, we perform pairwise NCD calculations between website HTML datasets to measure website similarities and detect phishing websites. Further details regarding the system design will be discussed in the next section.

B. Prototype-Based Learning

Using NCD metrics to measure the similarities between two websites, we perform *clustering* to divide phishing websites with similar HTML contents into a number of groups and *classification* to assign a website to the previously generated

group with the closest similarity. The aim of our clustering is to group similar websites, indicated by the relatively small number of pairwise NCD values.

When deciding which approach to take when performing clustering and classification, we take into consideration the main goals or characteristics which are important to have for the detection to be accurate and sustainable. Firstly, we aim for the system to be feature-free, meaning that it can learn directly from the data without the need to perform manual feature extraction. Furthermore, one of the advantages of using a feature-free approach is that the system can adapt to changes in phishing behavior or data representation. Secondly, we attempt to build a detection system that has the ability to continuously learn and gain knowledge from the previous learning process, and incorporate this with new knowledge obtained while processing new data, resulting in a continuously learning system. With the ability to incrementally learn, the goal is that the system will be able to improve its detection over time as it meets new phishing examples.

For the reasons mentioned above, a prototype-based clustering approach has been selected, where clusters are represented by actual samples in the dataset instead of centroids. By using actual website samples as cluster representations, we are able to implement an incremental feature-free learning method as the prototype-based method does not necessarily require the data to be transformed into a specific feature representation. The prototype-based method fits our purpose of designing a fast, efficient, and feature-free algorithm. Using a greedy clustering algorithm [18], the overall time-complexity for n data samples would be $O(n^2)$, whereas it would only be $O(nk)$ using the prototype-based approach, where k is the number of clusters [22].

By definition, a *prototype* is a data point, which represents all the data points in a cluster. Each data point in a given cluster can be assigned to a specific prototype, where the distance from this prototype to the data instance is less than a certain threshold distance d. Each prototype in our detection system would be one or several phishing websites which represent a cluster of similar phishing websites, potentially generated using the same template or phishing kit. However, we are not focusing on which cluster or class of phishing websites each dataset belongs to. Instead, we aim to detect whether a website would be categorized as one of the phishing classes or whether it is not similar to any class at all. Therefore, our detection approach attempts to check whether a website is similar to any phishing cluster/class, by measuring the pairwise NCD values between the website and each prototype. Further details on the algorithms will be given in the following sections.

There is no specific requirement as to what distance metrics to choose (e.g., Hamming distance, Euclidean distance, and edit distance). However, in our case, we have selected NCD as the distance measure for reasons that were mentioned in the earlier sections. The concept of prototypes is illustrated in Figure 1. In this figure, roughly 60 data instances are grouped in three classes, which are represented using five prototypes. In some cases, a cluster can also be represented by more than one prototype, e.g., the clusters in the left-bottom and right parts of Figure 1. Using the prototypes, we are
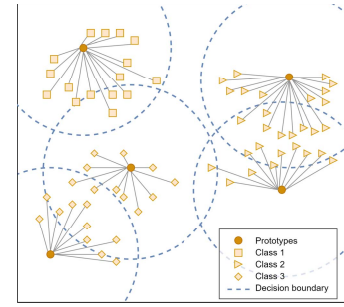

Fig. 1. Classification using Prototypes.

(a) First prototype
(b) Second prototype (furthest point from $prototype_1$)
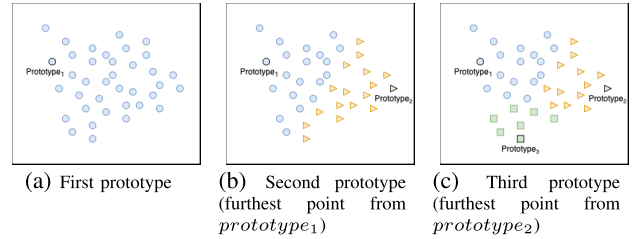(c) Third prototype (furthest point from $prototype_2$)

Fig. 2. Furthest Point First Algorithm.

also able to perform classification on a new data point x by checking if the data is similar to one of the prototypes, i.e., $NCD(x, z) < d_{threshold}$ for each $z \in prototypes$ (the set of prototypes). As seen in Figure 1, it is possible that the new data sample may be close to two or more prototypes, or located in the regions where the prototype decision boundaries overlap with one another. For simplicity, we have assigned such data samples to the prototype which is the closest (i.e., with the smallest NCD). Note that the clusters are circular in nature, with a prototype at the center. This is in contrast with Lloyd's algorithm for the k-means clustering problem, which produces Voronoi cells, with the cluster center as the centroid.

1) Prototype Extraction: Extracting representative prototypes from the data is a non-trivial task. The algorithms used to perform prototype extraction is inspired by [22], with changes made to use NCD as the distance function. The algorithm adopted an $O(nk)$-time algorithm originally proposed by Gonzàlez [22], in which extracting k prototypes from n data points takes an objective function value at most twice the size of the optimal solution. Gonzàlez's Furthest Point First (FPF) algorithm is illustrated in Figure 2. The main idea behind this algorithm is to select a data instance as a prototype, assign all data to the closest prototype, then choose the next instance furthest from the current prototype as the next prototype. This process is repeated until all data is assigned to a prototype with NCD values less than $d_{threshold}$. After performing prototype extraction by running this algorithm, we will have a set of prototypes representative of the phishing data.

2) Classification: With the list of prototypes, we are able to classify whether a website is similar to one of the known phishing websites. This is done by computing the NCD between the website x and each phishing website prototype z, and checking whether a prototype exists such that the NCD value is less than a certain maximum distance value or $NCD(x, z) < d_{threshold}$. The algorithm to perform

Algorithm 1 Prototype Extraction

1: $prototypes \leftarrow \emptyset$
2: **for all** $x \in data$ **do**
3: $\quad distance[x] \leftarrow \infty$
4: $\quad cluster[x] \leftarrow \emptyset$
5: **end for**
6: **while** $max(distance) > d_{threshold}$ **do**
7: $\quad z \leftarrow \arg\max_{x \in data} distance[x]$
8: $\quad$ **for** $x \in data$ **do**
9: $\quad\quad$ **if** $distance[x] > NCD(x, z)$ **then**
10: $\quad\quad\quad distance[x] \leftarrow NCD(x, z)$
11: $\quad\quad\quad cluster[x] \leftarrow z$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad prototypes \leftarrow prototypes \cup \{z\}$
15: $\quad data \leftarrow data \setminus \{z\}$
16: **end while**

classification for this work is adapted from the method proposed by Rieck at al. [35], as shown in Algorithm 2.

Algorithm 2 NCD-Based Classification

1: **for** $x \in data_{test}$ **do**
2: $\quad z \leftarrow \arg\min_{p \in prototypes} NCD(x, p)$
3: $\quad$ **if** $NCD(x, z) < d_{threshold}$ **then**
4: $\quad\quad$ classify x as phishing
5: $\quad$ **else**
6: $\quad\quad$ classify x as non-phishing
7: $\quad$ **end if**
8: **end for**

3) Incremental Learning: For continuous phishing website detection, the use of an incremental learning algorithm is proposed, which is adapted from the incremental learning algorithm introduced by Rieck *et al.* [35]. The main idea is to perform NCD-based classification in each iteration, and update the set of prototypes with new prototypes extracted from misclassified phishing samples belonging to new data in every iteration.

Algorithm 3 Incremental Learning

1: **for** $data \leftarrow source$ **do**
2: $\quad$ **for** $x \in data$ **do**
3: $\quad\quad$ classify x using $prototypes$ $\quad\quad \triangleright$ Algorithm 2
4: $\quad$ **end for**
5: $\quad rejected \leftarrow$ samples in $data$ rejected as phishing
6: $\quad prototypes_{new} \leftarrow$ prototypes in $rejected$ $\quad \triangleright$ Algorithm 1
7: $\quad prototypes \leftarrow prototypes \cup prototypes_{new}$
8: **end for**

C. Optimal Distance Threshold

When performing prototype extraction, we need to select the maximum distance threshold that defines the size of each prototype's covering, which in the end will affect the size of

each cluster and the classification's performance. A common and intuitive characteristic of a good clustering threshold is one that produces compact or dense clusters, which are far away from each other [18]. Based on these characteristics, we define a *quality of clustering (QC)* metric, which is obtained by calculating the ratio between the average cluster compactness and the minimum distance between the prototypes, as shown in Equation 3. The *compactness* of each cluster is computed by taking the average of pairwise NCD values between each data instance in a cluster with the corresponding prototype (Equation 1). Note that the more compact the cluster, the smaller this value will be. *Minimum inter-cluster distance (MICD)* is the minimum NCD value between every combination of prototype pairs (Equation 2). The further apart the clusters are, the larger this value will be. Therefore, a smaller QC value represents a better quality of clustering.

$$Compactness(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} NCD(x, prototype_i) \quad (1)$$

$$MICD = \min_{x, y \in prototypes} NCD(x, y) \quad (2)$$

$$QC = \frac{\frac{1}{|C|} \sum_{i=1}^{|C|} Compactness(C_i)}{MICD} \quad (3)$$

During the threshold selection process, our aim is to select the distance threshold $d_{threshold}$ which leads to the best clustering quality. As smaller QC implies better cluster quality, the threshold selection process is done by performing optimization to obtain the minimum QC parameter.

IV. PHISHSIM SYSTEM OVERVIEW

We propose a server-based phishing detection system *PhishSim* that can be deployed by enterprises in their corporate intranets, Internet Service Providers (ISP), and cloud providers like Amazon, Microsoft, and Google, to defend against phishing attacks. Figure 3 shows a high-level overview of how the system works and how it can be implemented. The system receives website URLs requested by users as input. Then, it provides recommendations about whether the webpages are safe or malicious. The recommendation output is generated from the NCD-based classifier using prototypes stored in its *Prototype DB* database. This system also has an ability to update its prototype database by receiving a new phishing website list and generating new prototypes. The new phishing websites can be obtained from user reports and feedback, and from phishing blacklist providers, such as PhishTank [9].

In the following subsections, we describe the two main components in our proposed feature-free phishing detection system: *phishing website classification* and *phishing prototype database update*.

A. Phishing Website Classification

To perform phishing website detection, our proposed system receives the website URL when a user is about to open the website's HTML Document Object Model (DOM), hence simulating how the page is rendered by a web browser. To obtain the website's HTML DOM, we used Chromium [10], an open-source software project, which is the basis of several web browsers, including Google Chrome and Microsoft Edge.
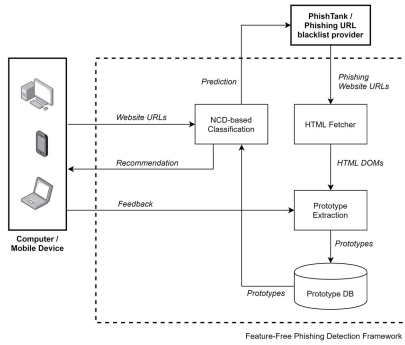
Fig. 3.   System Diagram of the Feature-free Phishing Detection Framework *PhishSim*.

Fig. 4.   Netflix dendrogram (HTML DOM).

Fig. 5.   Netflix Dendrogram (Website Screenshot Image).

After obtaining the website's HTML DOM, the system performs classification by measuring NCD-based similarities between the website and known phishing prototypes in the database. The classification process returns a prediction on whether the website is phishing or legitimate. If the website is suspected to be a phishing website, this result will be added to the prototype database or blacklist provider, and users will be redirected to a warning page (with a recommendation to not open the page) when accessing the webpage. Our system can be used in conjunction with Google Safe Browsing [36] that is based on the list of URLs for websites that contain phishing content.

To enhance the robustness and prevent attackers from evading detection, we removed the text and HTML comments in the content prior to performing compression, leaving only the HTML tags which are rendered and visually shown on the browser. Thus, the addition of invisible elements in the HTML will not affect the performance.

B. Phishing Prototype Database Update

To keep the system prediction accurate, our system has a mechanism to frequently update the phishing prototype database. The system is able to update its prototype database by periodically receiving data from phishing blacklist providers (e.g., daily or weekly) and users' reports. After obtaining the websites' HTML DOMs, the system performs prototype extraction to extract representative prototypes from these new data. Afterwards, the extracted prototypes are stored in the prototype database used for NCD-based classification.

V. SIMILARITY ANALYSIS

To validate our method, we conducted two experiments to observe the characteristics and similarities between a phishing website and its legitimate target website by performing pairwise NCD computations. In the first experiment, we performed pairwise NCD calculations and clustering on data targeting a specific brand, and we focused on observing the similarities between phishing and legitimate websites. The aim of this experiment was to observe the relationship between each website, specifically the relationship between websites (phishing and legitimate) of different brands. In the second experiment, we applied the same approach as in the second experiment to phishing and legitimate website data of multiple brands.
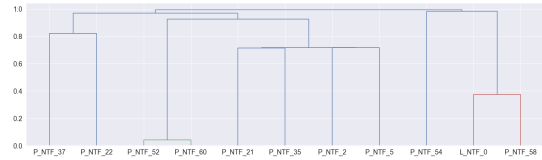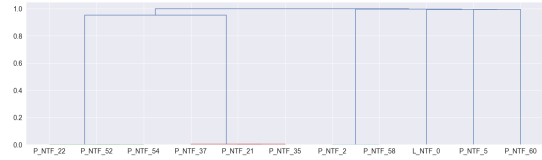
The phishing data used in this experiment is from websites reported by Internet users to *PhishTank* [9] and collected between March and April 2020. To identify the target brand of each phishing website, we utilized the target brand information, which was provided by PhishTank. However, we noticed that many phishing websites reported to PhishTank did not provide target brand information (i.e., identified as 'Others'). Thus, we also made use of Google's Cloud Vision API to estimate the target brand by detecting the logos that appeared in website screenshots. Lastly, we performed manual checking after grouping the data based on its brand to ensure that the target brand was correct.

A. Brand Specific Similarity Analysis

For comparison with Chen *et al.* [16], we also performed website similarity analyses using the website screenshots. In this experiment, we focus on three brands, which are Microsoft, PayPal, and Netflix, as we found that these brands dominate our phishing website data. Furthermore, VadeSecure reported that these brands were among the top five phishing targets in 2019 and the top three phishing targets in Q1 and Q3 2019 [5]–[8]. For each brand, we included the legitimate website data and ten corresponding phishing websites selected at random.

We first performed pairwise NCD computation using HTML DOM files to observe whether similarities exist among the phishing websites, and whether they are also similar to the legitimate target website. To gain a better understanding of the relationships between the websites, we performed data clustering on the pairwise NCD data and visualized the data as dendrograms. In the dendrograms provided in the following subsections, we followed a strict naming convention with three label codes (separated by an underscore). The first code (prefix), indicates whether it is a legitimate or phishing website ('L' for legitimate, 'P' for phishing). The second code indicates the brand ('MST' for Microsoft, 'NTF' for Netflix, and 'PYL' for PayPal). The third code (suffix) indicates the website number. In the following subsections, we will focus our discussion on the findings for each brand.

In this section, we will only include similarity analyses on Netflix phishing and legitimate websites, as we found that the design of the Netflix phishing websites has more variations. These variations are potentially due to frequent changes of the design of the target legitimate website itself.

(a) P_NTF_52   (b) P_NTF_60

Fig. 6.   Phishing Websites with Similar HTML Contents (Cluster 1).

Fig. 7.   Netflix Legitimate Website.

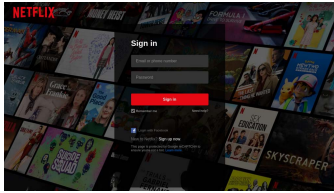Fig. 8.   Multi-Brand Website Data (HTML DOM).

Fig. 9.   Multi-Brand Website Data (Website Screenshot Image).

To observe the relationship between these websites based on their content, we compute the pairwise NCD values and perform data clustering on the HTML DOM files. The pairwise NCD values between the HTML DOM files are visualized as a cluster dendrogram as shown in Figure 4. Cutting the dendrogram on the highest level gives two clusters. The legitimate website is clustered together with two of the phishing websites, while the remaining phishing websites are clustered together. As shown in the dendrogram, the phishing websites that are most similar to each other are P_NTF_52 and P_NTF_60, with a NCD of 0.04. Interestingly, these two websites have a very distinctive design, as shown in Figure 6. By looking into the HTML DOM files, however, these websites' HTML DOMs are almost identical. The only difference is in a numeric variable set when loading the CSS stylesheet. It is very likely that these websites are built using the same phishing kit, indicated by the similarities in the HTML DOM structure. We also found that this is the case in the other cluster of websites, which are (P_NTF_21, P_NTF_35, P_NTF_2, P_NTF_5), and (P_NTF_37, P_NTF_22). Similar to the first case of P_NTF_52 and P_NTF_60, the websites which are clustered together have an identical HTML DOM structure, but with different website designs. We also found that in some cases, the websites are similar in style but have different background images. Meanwhile, we also see that the legitimate website has a close relationship with one of the phishing websites, P_NTF_58. While this might introduce false positives in the detection system for detecting phishing using NCD-based similarity measurements, we argue that this would be a rare case and it can be avoided by whitelisting legitimate websites. The aim of the whitelist method is to filter legitimate websites by comparing whether the content and domain of the website is identical to one of the whitelisted websites. Using this method, the phishing website will fail to satisfy the properties needed to escape filtering, and would be processed further with the NCD-based similarity assessment.

To observe the relationship between the websites based on their screenshots, we also analyzed the pairwise NCD values between the screenshots. Using the same approach as the Netflix HTML DOM files, we performed clustering with th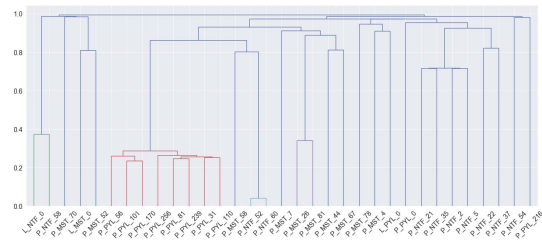e screenshot imag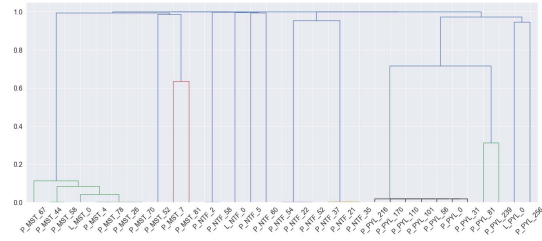e data. The cluster dendrogram is shown in Figure 5. Based on these results, it is demonstrated that websites with different designs have pairwise NCD values between 0.95 and 1. Meanwhile, the websites that are detected as having high similarity are clustered together, as shown in the dendrogram. These websites have identical designs, such as (P_NTF_22, P_NTF_52, P_NTF_54), (P_NTF_37, P_NTF_21, P_NTF_35), and (P_NTF_2, P_NTF_58). Interestingly, the legitimate website (Figure 7) was not detected as similar to any of the phishing websites, despite having similar element styles. The only difference is the background image, which may be updated frequently with images from the latest movie or TV series. Based on this experiment, we were unable to detect website similarities by computing the pairwise NCD values on the website screenshots.

B. Multi-Brand Similarity Analysis

To observe the relationship between various brands' phishing and legitimate websites, we attempted to compute the pairwise NCD values between the websites. We did this in terms of the HTML DOM and website screenshot images. The brands that we included in this analysis were Microsoft, Netflix, and PayPal, which are the three brands that consistently ranked among the top five phishing targets in every quarter in 2019 [5]–[8]. Having these pairwise NCD values, we were able to cluster the websites and observe the relationship by visualizing the clusters as a dendrogram. The dendrogram of websites from the pairwise NCD values between the HTML DOM files is shown in Figure 8, while the dendrogram constructed using the screenshot image files is shown in Figure 9.

By comparing the dendrograms in Figure 8 and Figure 9, we are able to capture similarities and differences between the website clusters constructed using HTML DOM files and website screenshot image files. In terms of similarities, phishing websites targeting similar brands are shown to be clustered together and have a close relationship in both dendrograms. However, the first dendrogram (constructed using the HTML DOM files) demonstrates a greater separation between the cluster of phishing websites targeting a certain brand and the cluster where the legitimate target website belongs.
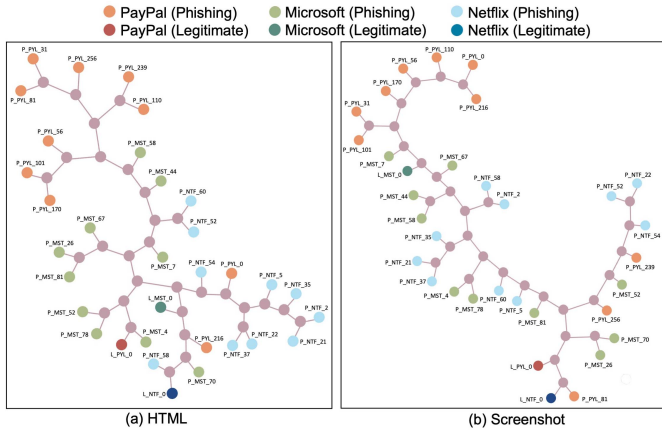
Fig. 10. Website Clusters.

Meanwhile, in the second dendrogram (constructed using the website screenshot images), clusters of the same brands tend to be alongside each other. Furthermore, there seems to be very limited separation between the legitimate and phishing websites.

The relationship between each website can also be observed by plotting the clusters as a quartet tree (Figure 10), which visualizes the closeness of the websites based on their distances to each other. To generate the quartet trees, we used the CompLearn toolkit,[1] which uses the Minimum Cost Quartet Tree Reconstruction method to generate the optimal quartet trees that represent the clusters. As shown in Figure 10, the majority of PayPal phishing websites occupy a single branch on the tree when clustering using HTML and screenshot data. Using the HTML data, it is shown that the Microsoft phishing websites are clustered more closely compared to clustering using screenshot data. The Netflix phishing websites are grouped together in the lower right part of the tree when clustering using HTML data. Meanwhile, using screenshot data, the Netflix phishing websites with identical designs are clustered together (i.e., (P_NTF_22, P_NTF_52, P_NTF_54), (P_NTF_37, P_NTF_21, P_NTF_35), and (P_NTF_2, P_NTF_58)) as also shown in the dendrogram in Figure 5. Based on these quartet trees, the phishing websites of the same brands are relatively closer to each other when clustered using its HTML data compared to the screenshot data, which indicates high similarity between phishing websites' HTML content despite its visual appearance.

VI. Optimal Distance Threshold Selection

We attempted to vary the maximum distance value to obtain the minimum Quality of Clustering (QC) value as defined in Section III-C. Note that a smaller QC value indicates superior cluster quality. After calculating the Quality of Clustering (QC) values, we plot the QC against the maximum distance threshold and fit an eighth-degree polynomial to the points, then take the distance which gives the minimum value of the fitted curve as the distance threshold.

For the threshold selection process, we used the phishing data shared by Cui *et al.* [18], and selected the ones which are
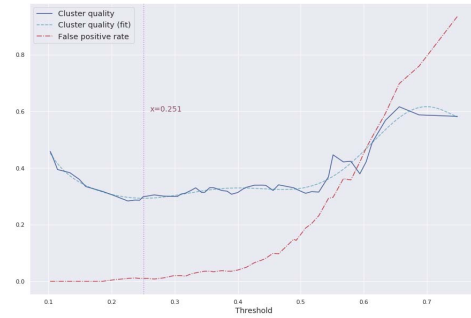
---

[1]https://complearn.org/index.html

Fig. 11. Threshold Selection.

reported between 1 and 31 January 2016. The cluster quality and false positive rate for a corresponding selected threshold is shown in Figure 11. As shown in the graph, we obtained the best cluster quality or minimum QC by selecting 0.251 as the distance threshold. At this point, the false positive rate is also very low, which is close to zero.

VII. Experimental Setup

In this section, we briefly describe the experiment and methodology to evaluate our proposed method, followed by further details on the phishing and legitimate website dataset used in our study.

A. Evaluation Methodology

There are three experiments in this study. The first experiment aims to evaluate the performance of our prototype-based method in detecting phishing websites and compare the result to other methods, i.e., the proportional distance based method proposed by Cui *et al.* [18] and the use of Doc2Vec models and the Manhattan distance as introduced by Feng *et al.* [20]. In this experiment, we also evaluated the performance of detection in various legitimate-to-phishing class ratios, and the detection performance when using the combination of NCD with other distance measures. In the second experiment, we simulate the incremental detection of phishing websites and assess the performance of NCD and prototype-based incremental learning. Finally, in the third experiment, we perform an analysis on the memory requirements and run-time performance for evaluating the feasibility of PhishSim implementation. In this study, we used the LZMA algorithm when computing the NCD values at it is shown to approximate the NID better which leads to better prototype extraction and superior phishing detection performances as shown in Appendix B.

B. Dataset

To evaluate the performances, we used our own dataset which is completely different from the one used in the similarity analysis (Section V) and the optimal distance threshold selection process (Section VI). We plan to share the dataset for the experiments for future work studies.

To build the phishing dataset, we created a script which fetched the latest phishing URL list from PhishTank and collected phishing pages every 3 hours to maximize the number of live phishing pages. The phishing URLs that are provided by PhishTank are typically fake sign-in or login pages, or pages

TABLE I
DETECTION METHOD COMPARISON

	PhishSim	Cui et al. [18]	Feng et al. [20]
Website Representation	Raw data	Tag vector (number of occurrences of specific HTML tags)	Doc2Vec vector
Distance Metric	Normalized compression distance	Proportional distance	Manhattan distance
Phishing Sample Database	Extracted phishing prototypes from prior submissions	All phishing websites from prior submissions	All phishing websites from prior submissions
Clustering Algorithm	Prototype-based clustering	Agglomerative clustering	Agglomerative clustering

with input forms to steal account information. The total number of collected phishing websites are roughly 13,300 websites. To fetch the website content, we used Selenium and Chromium [10], instead of fetching the HTML source code using built-in *wget* or *curl* tools, which gave us the HTML as rendered in a browser. After performing sanitation and removing error/empty pages, we have 9,034 phishing websites which were reported by users to PhishTank [9] between 28 April 2020 and 22 February 2021. We also removed cloaked phishing websites, by inspecting the URL of the landing page, then removed pages whose domain is included in the top 500 domains, which indicates that there is a redirection to a legitimate page.

Meanwhile, to obtain a representative legitimate website dataset, we compiled legitimate pages in the Common Crawl database [3]. The legitimacy of the website is assessed by the website's popularity based on the Tranco page ranking list by Le Pochat *et al.* [26]. This page ranking is selected because it is claimed to be more robust against manipulation by adversaries. We use the Tranco list[2] generated on 14 March 2021, and selected the top 4,000 domains. For each domain, we collected 100 pages in Common Crawl's February/March 2021 crawl archive.[3] After removing empty pages and error pages, we have a collection of 180,302 websites.

VIII. RESULTS

In this section, we provide further details regarding the results of the experiments. We study the performance of the phishing detection algorithm, evaluate the incremental learning framework, and analyze the memory requirements and the runtime measurement.

A. PhishSim Performance

Based on the results we obtained during the threshold selection process, we selected the maximum distance threshold value as 0.251 (Section VI). We evaluated the performance of our proposed phishing website detection method using prototype-based learning algorithms, with NCD as the distance metric. We applied a temporal split to the phishing dataset based on the website submission date, when allocating the data for prototype extraction (model training) and performance evaluation (testing). This is recommended in past studies over the cross-validation evaluation as cross-validations could introduce performance overestimation due to the risk of training on future data and testing on past data [23], [31]. We use the 7,746 phishing data submitted prior to 18 February 2021 when extracting the phishing prototypes, while the remaining 1,288 phishing data are allocated for testing. Meanwhile, we use
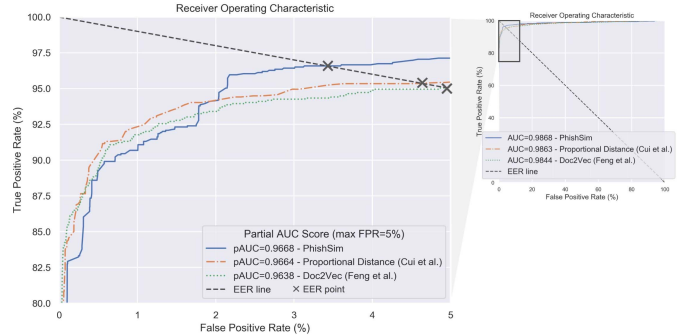
Fig. 12.   Receiver Operating Characteristic (ROC) Curve.

TABLE II
PHISHING DETECTION PERFORMANCE COMPARISON
(DEFAULT DISTANCE THRESHOLD)

Performance Metrics	PhishSim	Proportional distance [18]	Doc2Vec [20]
TPR	89.75%	84.08%	95.65%
TNR	99.42%	99.90%	93.05%
FPR	0.58%	0.10%	6.95%
Accuracy	99.36%	99.79%	93.07%
G-mean	94.47%	91.65%	94.34%

the entire legitimate website dataset for testing. The testing dataset has a phishing to legitimate class ratio of 1 to 140. We intentionally craft the testing dataset class ratio to resemble phishing detection in the real scenarios, where typically there is one phishing page for every 100 legitimate pages [19], [36].

We compare the performance of PhishSim with other similarity-based phishing detection approaches introduced in past studies by Cui *et al.* [18] and Feng *et al.* [20]. Cui *et al.* [18] proposed a distance metric for detecting similarity among phishing websites, the *proportional distance*, and made use of hierarchical clustering to group similar phishing websites and detect whether an unknown website is clustered together with a known phishing website. Meanwhile, Feng *et al.* [20] proposed a method to detect phishing websites using clustering algorithms and Doc2Vec models to represent phishing and legitimate websites. To measure the website similarities, Manhattan distance is used as proposed in their study. Details on each technique, including the distance metrics and clustering algorithms, are summarized in Table I. As the code implementation of the approaches proposed by Cui *et al.* [18] and Feng *et al.* [20] are not publicly shared, we replicated these approaches using the same algorithms and model hyperparameters as mentioned in the research papers.

To compare the detection performances and characteristics, we plot the receiver operating characteristic (ROC) curve based on the distance of each instance to the closest phishing sample or phishing prototype. The ROC curve is shown in Figure 12. The AUC score could be interpreted as the probability of the model assigning a phishing website with a

TABLE III
PHISHING DETECTION PERFORMANCE COMPARISON (AT EER POINT)

Performance Metrics	PhishSim	Proportional distance [18]	Doc2Vec [20]
TPR	96.58%	95.34%	95.03%
TNR	96.59%	95.50%	95.03%
FPR	3.41%	4.50%	4.97%
Accuracy	96.59%	95.50%	95.03%
G-mean	96.59%	95.42%	95.03%

higher similarity score than the legitimate website's similarity score. The AUC score is an aggregate measure that model's classification performance across all possible thresholds which provides a method to compare binary classifier performances regardless of the selection of distance threshold or cut-off. Based on the AUC score, PhishSim slightly outperforms other methods with an AUC score of 0.9868, indicating that it performs better in classifying phishing and legitimate websites in general. In the case of detecting phishing attacks, we would be more interested in selecting an operating point with a low false positive rate (FPR). To observe the performance in this region, we also provide a ROC curve which focuses on the low-FPR region and the partial AUC scores of each method (maximum FPR of 5%). As shown in Figure 12, the partial AUC score also shows that PhishSim outperforms other methods with a partial AUC score of 0.9668.

We also compare the detection performance when choosing the default optimal distance thresholds. As discussed in Section III-C, we found that PhishSim works best when selecting 0.251 as the distance threshold. Meanwhile, we selected the best distance threshold of the baseline methods using the suggested value or settings as mentioned in the past study [18], [20]. As shown in Table II, PhishSim achieved an excellent TPR of almost 90% with a low FPR of 0.58%. While Doc2Vec achieves a higher TPR, the FPR at this operating point is also significantly higher, which would not be ideal for phishing detection. We also measured the G-mean, or the geometric mean of the TPR and true negative rate (TNR), which provides a better way to evaluate a model in a highly imbalanced classification problem, which is similar to the ratio between phishing to legitimate websites in real scenarios. As shown in Table II, PhishSim outperformed other methods with a G-mean of 94.47%, indicating a better overall performance in detecting phishing with the real imbalanced class distribution. Furthermore, we also compared the performance of each method when selecting the EER point as the operating point, which indicates the operating point when the false negative rate and the false positive rate have the same value. As shown in Table III, PhishSim has superior performance compared to other methods, by achieving TPR, TNR, and accuracy, with the lowest FPR. It is also shown that PhishSim gives the best G-mean score, which shows the ability to provide the best compromise between achieving a high TPR while still having a high TNR.

B. PhishSim for Improving Existing Methods

We also observed if the use of PhishSim in combination with other methods would improve the phishing detection performance. In this experiment, we combined the two best
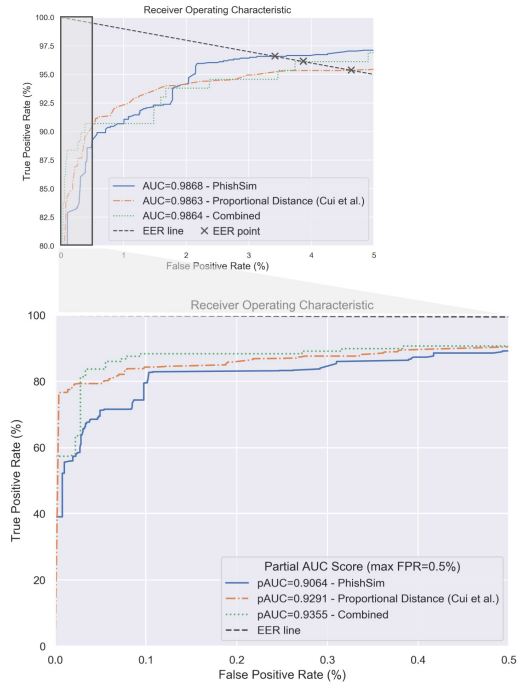

Fig. 13. Combining NCD and Proportional Distance.

performing methods in terms of AUC score, which are PhishSim and the proportional distance based method by Cui *et al.* [18] and trained a logistic regression model to perform a weighted sum on the NCD and proportional distance values.

For this experiment, we used the dataset that we collected, as mentioned in Section VII-B. We allocated 90% of the phishing and legitimate website data for training the logistic regression model and used the remaining 10% of the phishing and legitimate website data for testing the model's performance, while keeping the phishing to legitimate class ratio to 1:140 in both training and testing dataset. To improve the trained logistic regression model's performance, we applied the SMOTE oversampling method on the training dataset to generate synthetic phishing data to provide more samples.

The performance of the logistic regression model is shown in Figure 13. The AUC score shows a similar performance in general. However, in a very low FPR region (maximum FPR of 0.5%), the combined method was able to achieve a higher partial AUC score compared to PhishSim and proportional distance alone. To compare the performance, we select an operating point where the FPR is around 0.1% which is considered an acceptable false positive rate for phishing detection in real scenarios, similar to some past studies [18], [36]. The TPR, FPR, and accuracy of the combined method at this operating point is shown in Table IV. It is shown that combining these methods resulted in a great improvement in terms of TPR, with a 5% difference from using proportional distance alone, while having a slightly lower FPR of 0.09% and a higher accuracy of 99.82%.

C. Incremental Learning

With the selected threshold (Section VI), we also evaluated the performance using the incremental learning method.

TABLE IV

PERFORMANCE OF PHISHSIM WITH
PROPORTIONAL DISTANCE (FPR <0.1%)

Performance Metrics	PhishSim	Proportional distance [18]	Combined
TPR	79.58%	84.32%	**88.37**%
TNR	99.90%	99.90%	**99.91**%
FPR	0.10%	0.10%	**0.09**%
Accuracy	99.76%	99.79%	**99.82**%
G-mean	89.16%	91.78%	**93.96**%

TABLE V

PHISHING DETECTION PERFORMANCE COMPARISON FOR INCREMENTAL
LEARNING (DEFAULT DISTANCE THRESHOLD)

Performance Metrics	PhishSim	Proportional distance [18]	Doc2Vec [20]
TPR	89.52%	84.08%	94.25%
TNR	99.22%	99.91%	95.74%
FPR	0.78%	0.09%	4.26%
Accuracy	97.11%	96.47%	95.42%
G-mean	94.25%	91.66%	95.00%

TABLE VI

PHISHING DETECTION PERFORMANCE COMPARISON FOR INCREMENTAL
LEARNING (AT EER POINT)

Performance Metrics	PhishSim	Proportional distance [18]	Doc2Vec [20]
TPR	96.66%	95.34%	95.11%
TNR	96.74%	95.57%	95.10%
FPR	3.26%	4.43%	4.90%
Accuracy	96.72%	95.52%	95.10%
G-mean	96.70%	95.46%	95.10%

Similar to the previous experimentation, we also applied a temporal split to the dataset. The phishing dataset was sorted and divided by its week of submission for each iteration of the incremental learning process. In each iteration, we treated the last week of data as the *testing data* and performed classifications to evaluate the detection performance using the prototypes extracted in earlier months. Following this, we performed prototype extraction on the testing data at each iteration to learn new phishing prototypes, simulating the process of weekly detection and database updates of anti-phishing systems. Using the testing data from 28 May 2020 to 22 February 2021, we performed evaluations for roughly 39 weeks. As a comparison, we also implemented the hierarchical clustering method proposed by Cui *et al.* [18] and Doc2Vec based method by Feng *et al.* [20] in an incremental learning experimental setup. Note that instead of updating the clustering model, we performed clustering from scratch in every iteration when implementing the method by Cui *et al.* [18] and Feng *et al.* [20], since we are unable to find an approach to perform the clustering incrementally using this method. The performance of PhishSim compared to these methods at the last iteration of the incremental learning experiment is shown in Table V, which is consistent with the results from the non-incremental learning setting in Table II. We also included the ROC curve and performance at the EER point in Table VI.

Furthermore, we also observed the true positive rate and false positive rate at each iteration, which are provided in
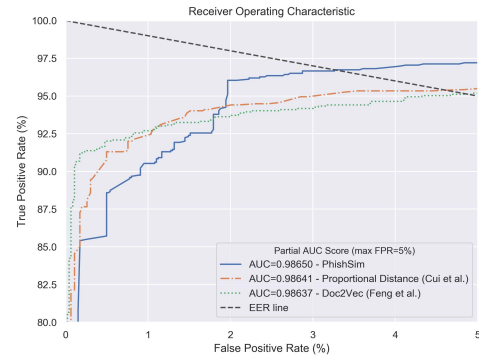

Fig. 14. Receiver Operating Characteristic (ROC) Curve (in Incremental Learning Experiment).
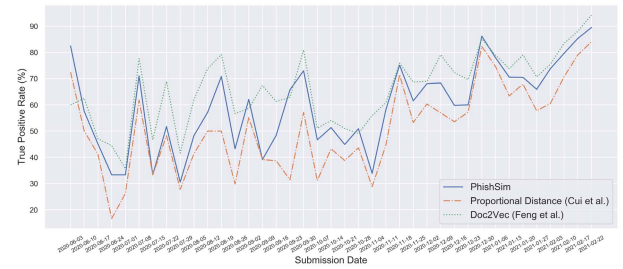

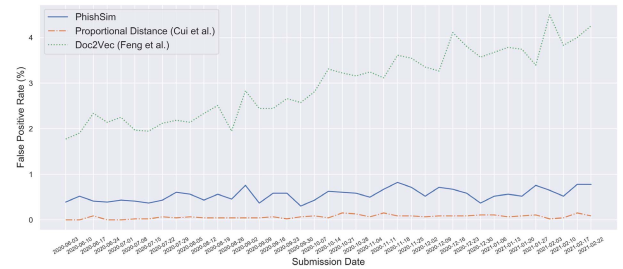Fig. 15. TPR in Incremental Learning Setting.

Fig. 16. FPR in Incremental Learning Setting.

Figure 15 and Figure 16. As shown in Figure 15, the Doc2Vec based method by Feng *et al.* [20] outperforms the other methods in most iterations. However, it is also shown in Figure 16 that the false positive rate of the Doc2Vec-based method is relatively high and seems to increase over time, which would not be suitable for phishing detection. On the other hand, PhishSim consistently outperforms the proportional distance based method by Cui *et al.* [18] in terms of true positive rate with a TPR of nearly 90%. While having a high TPR, PhishSim maintains a low and stable FPR of under 0.8% in each iteration.

D. Memory Requirements and Run-Time Analysis

Besides evaluating the detection performance, we also analyzed the memory requirements and run-time performance of PhishSim to evaluate the feasibility of deployment in real scenarios. To observe the memory requirements, we analyze the number of prototypes extracted at each iteration in the incremental learning experiment in comparison to the number of phishing websites these prototypes represent. We also computed the amount of time needed to process every website using a 6-core 3.70 GHz Intel(R) Xeon(R) W-2135 CPU.

Figure 17 depicts the compression ratios at each iteration in the incremental learning experiment (as discussed in
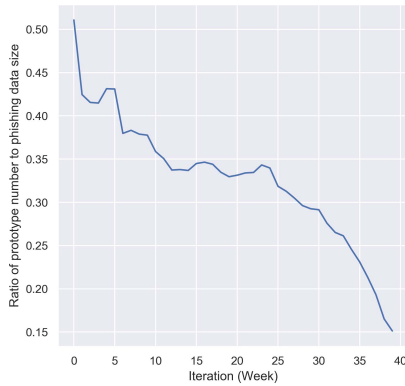
Fig. 17.　Ratio of prototype number to phishing data size.

Section VIII-C). Compression ratio in this context is defined as the ratio between the number of prototypes extracted relative to the total number of phishing websites they represent. As the detection system incrementally learns, the number of prototypes increases as new website samples are learned. However, compression ratio gradually decreases over time, showing that the system was able to extract meaningful phishing prototype representations. At the last iteration, around 1,366 prototypes were extracted which represents 9,034 phishing websites, giving a compression ratio of 0.15. The phishing HTML DOM data size average is around 2.02 KB with a standard deviation of 3.81 KB. A larger standard deviation relative to the mean would indicate that the data distribution is heavily right-skewed with a number of data outliers. Therefore, we use the median of the phishing website HTML DOM size in this analysis to predict the data storage consumption. Based on our experiment, the number of prototypes is 1,366 at the last iteration. With a median of around 727 B, storing 1,366 phishing prototypes would take roughly 0.947 MB of data storage.

To evaluate the run-time performance, we analyzed the time needed to perform classification on each website. To decide whether a website is legitimate or phishing, we perform pairwise NCD calculation between the website and each phishing prototype. The length of the compressed phishing prototype can be precomputed before classification, while the length of the compressed website only needs to be computed once. Thus, the process which would take time the most is compressing the concatenation of the website and each phishing prototype, which cannot be precomputed and should be performed individually for each prototype. The total amount of time to process a website would depend on the number of phishing prototypes. With the assumption that the length of each compressed prototype file is precomputed and the number of prototypes in the database is 1,366, we found that it would take around 0.3 seconds to process a single website.

In the incremental learning experiment (Section VIII-C), we also measured the total amount of time taken by PhishSim and other methods in past studies to perform detection and any other processes to prepare the detection model, which include prototype extraction (PhishSim), HTML tag computation and distance matrix construction (Proportional Distance [18]), Doc2Vec model training, vector inference, and distance matrix construction (Doc2Vec [20]). The total process duration at

the first five iteration is shown in Figure 18. As shown in Figure 18, PhishSim takes less than 0.3 seconds to perform detection, which is relatively similar to the time taken by the other methods. While PhishSim has the highest overhead on the first iteration, it takes the least amount of time to update the model on the next iterations compared to other methods, indicating the PhishSim's detection model can be updated incrementally faster.

IX. DISCUSSION

A. Detection Method and Performance

This study proposes a feature-free method for detecting phishing websites, which we argue would be suitable with the dynamic characteristics of phishing attacks. As reported by PhishLabs [32], phishing attackers have been persistently adapting to new opportunities and changing the manner in which they adapt. Furthermore, a study by Cui et al. [18] found that around 90% of phishing websites are variations or replicas of other phishing websites. There has also been an increased use of phishing kits, which ease the process of setting up phishing websites, enabling attackers to launch a large number of phishing attacks in a short period of time. As discussed in Section V-A, we found that these kits make the process of changing the phishing website layout and style more effortless.

Based on the experiments in Section VIII-A and Section VIII-C, our proposed method is able to outperform the hierarchical-clustering based method by Cui et al. [18] and Doc2Vec based method by Feng et al. [20] based on its AUC score. PhishSim achieves a high TPR of around 90% while maintaining a low FPR of 0.58%. We can also apply this method to build a model which can incrementally learn using one week of data to perform phishing detection with relative success. Evaluating our detection method in comparison with state-of-the-art methods, it has also been shown that PhishSim is able to outperform past studies with significant improvements in terms of true positive rate and with a relatively low FPR which is suitable for phishing detection. It has also been shown that we are able to improve the phishing detection performance when combining PhishSim with proportional distance, as shown in Table IV. Using this combined method, we could operate at an even lower FPR of 0.09% with a TPR of 88.37% and accuracy of 99.82%.

The method that we proposed is a feature-free method, which means that our method is not tied to a fixed website representation (Table I) and is more robust compared to feature-based methods which rely on specific features. On the other hand, the method by Cui et al. [18] uses the proportional distance which measures website similarity based on the number of occurrences of a set of predefined HTML tags and assumes that similar websites have a nearly identical number of HTML tag counts. While phishing websites that use custom HTML tags are not very common currently, the use of WebComponents [11] provide a method to perform novel website obfuscation technique using custom HTML tags to avoid detection from content-based methods which heavily relies on the analysis of the HTML content. While adding HTML tag definition and model training are possible,
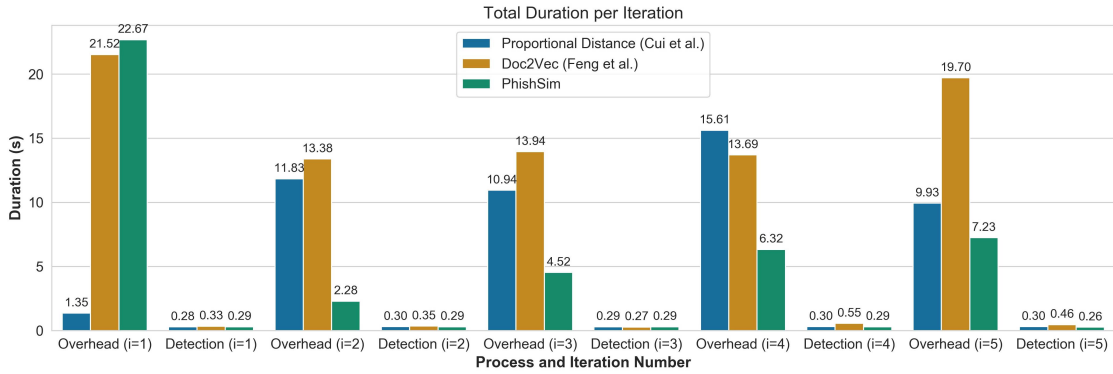
Fig. 18.   Total Process Duration at 1st to 5th Iteration.

phishing attacks are very dynamic and aggressive with changes in evasive phishing campaign methods every 37 days in average [2]. Having the flexibility to perform detection based on similarities without reliance on a preset corpus would be a great benefit to counteract the dynamics of phishing. Similarly, the approach by Feng *et al.* [20] uses a trained Doc2Vec model to generate vector representation for each website. It is not guaranteed that the generated vectors are always representative over time. To have an accurate detection, these methods rely on the assumption that the website representation would always remain relevant, which might not always be the case. Some features that were previously used to detect phishing websites accurately (e.g., bad forms, bad action fields, and non-matching URLs [37]) may no longer be relevant to phishing attacks. A past study investigated the relevance of these features in a phishing dataset collected in 2020 and has shown that only 8.90% of the phishing websites contain bad forms, while only 19.95% and 47.92% of them contain bad action fields and non-matching URLs respectively [33].

On the other hand, NCD as a similarity measure is universal and is purely based on the shared information between measured websites. Every phishing cluster is represented by a data instance, while the NCD value is used for measuring the relationship between all pairs of websites. Furthermore, the use of the Furthest Point First (FPF) algorithm provides a systematic way for prototype learning such that it does not require the data to be represented in a fixed vector representation or data structure. This is unlike most clustering and classification algorithms which require the data to be represented as a vector of representative feature values. Once the selected feature values are no longer relevant for phishing attacks, there is a risk of missing malicious phishing attacks. Meanwhile, our feature-free method would still be able to continuously identify these attacks with a maintained up-to-date phishing data and without the reliance on a specific feature set.

Robustness against code obfuscation is generally a concern when the detection approach evaluates the website's HTML source code. However, as stated in Section IV, PhishSim evaluates the HTML of the rendered page instead of the HTML source code. A web browser will produce the same HTML code regardless of the HTML obfuscation or hidden alterations in the CSS or JavaScript files. Furthermore, we also removed the text and HTML comments in the content prior to

performing compression, leaving only the HTML tags which are rendered and visually shown on the browser. Thus, the performance will also be unaffected by addition of invisible elements in the HTML.

B. Limitations and Future Work

There are some limitations to our proposed method. First, it may not be able to detect zero-day attacks or new variants of phishing websites, as our method is primarily focused on detecting variations of known attacks. However, we argue that our proposed method would improve the quality of detection done by experts, by using various machine learning algorithms, and by providing a framework to continuously learn and adapt to new cases of phishing, without the need to choose a fixed representation of phishing instances.

Moreover, there are some challenges in the practical implementation of this method. It is possible that a certain phishing prototype becomes obsolete which then makes the prototype irrelevant for detection. Therefore, ideally, there needs to be a way to effectively maintain the prototype collection so that only the useful prototypes are retained in the database. This is, however, currently beyond the scope of our paper. The proposed method can also be improved in the future by designing methods to effectively maintain useful prototypes, as well as by removing instances which are no longer relevant. The ability to remove irrelevant prototypes would be beneficial to reduce the storage size of the prototype database.

Our proposed phishing detection approach uses a prototype-based clustering and classification approach. This helps us to make the algorithm's decisions more interpretable, as it clarifies what each cluster or class corresponds to, by observing the prototypes. For future work, it would be interesting to further investigate the relationship between the set of prototypes that the algorithm finds and the phishing dataset, and how they evolve over time.

Furthermore, the study of PhishSim was mainly motivated by a previous work [18], which found that most of the phishing websites are replicas with similar HTML contents. Currently, performing a comprehensive comparison of using content similarity over visual similarity would be a challenging task, due to the unavailability of a large-scale high-quality dataset that contains both the rendered page's HTML and its screenshot. Future studies that focus on a comprehensive study of the content-based and visual-based methods using

a standard website dataset would be beneficial in improving phishing detection studies.

X. Conclusion

In this paper, we propose a feature-free method for detecting phishing websites using the Normalized Compression Distance (NCD) which measures website similarity by compressing them, eliminating the need to perform any feature extraction nor any reliance on a specific set of website features. This method examines the HTML source codes of webpages and computes their similarity with known phishing websites. We propose the use of the Furthest Point First algorithm to perform phishing prototype extractions, in order to select instances that are representative of a cluster of phishing webpages. We also introduce the use of an incremental learning algorithm as a framework for continuous and adaptive detection without re-performing new feature extraction when concept drift occurs. Evaluating the performance on a recent large-sized dataset, our proposed method is shown to outperform past studies in detecting phishing websites with an AUC score of 98.68%, a high true positive rate of around 90% while maintaining a low FPR of 0.58%.

Appendix A
Normalized Compression Distance

Normalized Compression Distance (NCD) is an application independent information theoretic method for measuring the similarities between two objects. It is a parameter-free similarity measure which uses compression algorithms to perform data clustering and classification in a broad range of applications. With this method, useful knowledge can be obtained from data without prior domain expertise as it operates on generic file objects, regardless of their format, structure, or semantics [27].

NCD computes the distance between two files by observing the result of compressing both files together and comparing this with the result of compressing each file separately. The key concept behind NCD is that two very similar files would compress much more effectively when combined prior to compression, compared with the total file size when compressed separately. On the other hand, compressing two files with little or nothing in common would not be as beneficial as compressing them separately [17].

NCD is motivated by the idea that the similarity of two objects can be measured by how easy it is to transform one object into the other. This concept is formally expressed by the information distance $E(x, y)$, which is defined as the length of the shortest binary program to compute y from x or x from y, and can be rewritten as

$$E(x, y) = max\{K(x|y), K(y|x)\} \quad (4)$$

The information distance is based on the notion of *Kolmogorov complexity*, $K(x)$, which depicts the length of the shortest program that computes x [17].

The normalized version of $E(x, y)$, called the normalized information distance (NID), is defined as

$$NID(x, y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}} \quad (5)$$

NID itself satisfies the conditions of a metric and can represent the similarity between two arbitrary entities according to the dominating shared features between both objects [27]. Unfortunately, the NID is based on the Kolmogorov complexity, which is not computable. This indicates that NID cannot be used directly. However, Cilibrasi and Vitanyi [17] demonstrate that we can approximate the Kolmogorov complexity, and introduce the notion of NCD as a practical similarity metric approximating NID using a real-world compression algorithms C, which is defined as,

$$NCD(x, y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}} \quad (6)$$

The NCD is a non-negative number between 0 and $1 + \epsilon$ which depicts how similar two objects are. Highly similar files would give smaller NCD values, while distinctive files would give NCD values closer to 1. Meanwhile, the ϵ in the upper bound represents a small error caused by imperfections in the compression algorithms which is usually a value below 0.1 for most standard algorithms [17].

The approximation of the denominator of NID in Equation 5 given a compressor C is quite obvious, giving us the denominator of NCD in Equation 6. However, the approximation of the NID numerator is not as straightforward. The numerator in Equation 5 can be expressed as

$$max\{K(x, y) - K(x), K(x, y) - K(y)\} \quad (7)$$

with

$$K(x, y) = K(xy) = K(yx) \quad (8)$$

where xy or yx depicts the concatenation of x and y, and $K(x, y)$ denotes the length of the shortest program to compute (x, y). The approximation is expressed using $K(x, y)$ since in practice, it would be easier to perform compression on the concatenation xy. Equation 7 can be best approximated by

$$min\{C(xy), C(yx)\} - min\{C(x), C(y)\} \quad (9)$$

In our experiment, $C(x, y)$ is used instead of $min\{C(xy), C(yx)\}$ as proposed in [17]. Here, we assume that C is *symmetric*, i.e., $C(xy) = C(yx)$. This is justified by previous experiments [17] using various block-coding based and stream-based compression algorithms, where it has been shown that the results are symmetric or only produce small deviations from symmetry.

Appendix B
Compression Algorithm Selection

In this section, we performed an evaluation on the performance of PhishSim using various compression algorithms (i.e. zlib, bz2, LZMA, and gzip). For this comparison, we used a different set of phishing and legitimate website data from the datasets used in our main experiment. To build the phishing dataset for this evaluation, we use the list of 9,245 phishing website URLs reported by user to PhishTank [9] between 7 November 2008 and 28 March 2020. Meanwhile, to obtain representative legitimate website dataset, we collected URLs of non-phishing pages which are associated with

TABLE VII
PERFORMANCE COMPARISON OF VARIOUS COMPRESSION ALGORITHMS

Performance Metrics	zlib	bz2	LZMA	gzip
TPR	61.717%	66.757%	**74.114%**	66.076%
TNR	**96.594%**	96.322%	94.142%	94.550%
Accuracy	79.155%	81.540%	**84.128%**	80.313%
AUC Score	**92.786%**	92.596%	91.804%	91.212%
G-mean	77.211%	80.188%	**83.530%**	79.041%

the brands or organizations that are commonly targeted by phishing attacks, especially its login page. This data collection process makes use of the Google Custom Search API [4] to perform custom search using a specific query and specifying the search results to be pages from a specific domain. To collect the legitimate webpage URLs, the phishing target list by PhishTank and Alexa top 1 million sites [1] were used. Each domain in the list was searched using the Google Custom Search Engine API and we collected the login page as well as the top 100 URLs the search. In total, our legitimate dataset consists of 10,869 legitimate websites, which were collected between February and April 2020. We applied a temporal split to the phishing dataset based on the website submission date, when allocating the data for prototype extraction (model training) and performance evaluation (testing). The training dataset consists of 8,511 phishing websites and 8,511 legitimate websites, while the testing dataset comprises 734 phishing and 734 legitimate websites.

The phishing detection performance comparison results are provided in Table VII. As shown in this table, the use of zlib algorithm leads to the best performance in terms of TNR and AUC score. However, the LZMA algorithm achieved a significantly higher TPR of 74.114% which is 13% higher than the TPR of zlib algorithm, indicating a superior ability in classifying the phishing websites. Furthermore, LZMA also outperformed other compression algorithms in terms of the accuracy and G-mean score, which computes the geometric mean of TPR (sensitivity) and TNR (specificity).

To observe the memory requirements, we also analyzed the number of prototypes extracted at each iteration in an incremental learning experiment in comparison to the number of phishing websites these prototypes represent. Fig. 19 shows the number of extracted prototypes, while Fig. 20 depicts the compression ratios at each iteration in the incremental learning experiment (Algorithm 3). Compression ratio in this context is defined as the ratio between the number of prototypes extracted relative to the total number of phishing websites they represent. As the detection system incrementally learns, the number of prototypes increases as new website samples are learned. However, compression ratio gradually decreases over time, showing that the system was able to extract meaningful phishing prototype representations. As shown in Figs. 19 and 20, the use of LZMA algorithms achieved the highest number of extracted prototypes which are representative of the online phishing data. The ability to extract more phishing prototypes consequently led to a better performance over time without the need to retain a large number of phishing
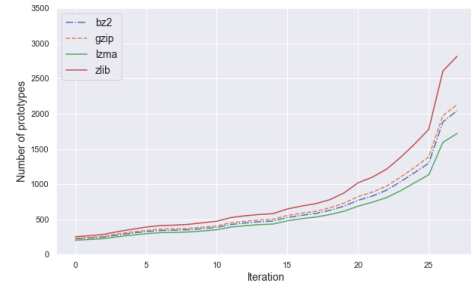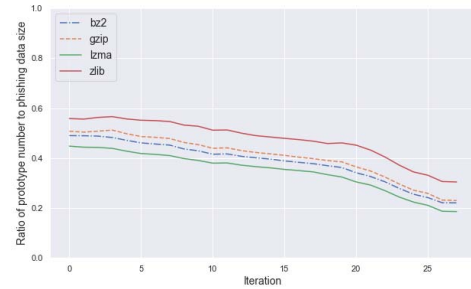

Fig. 19. Number of prototypes.

Fig. 20. Ratio of prototype number to phishing data size.

prototypes, as shown by the decreasing ratio of prototype number to phishing data size.

REFERENCES

[1] Alexa. *Top 500 Sites in Each Country*. Accessed: May 11, 2020. [Online]. Available: https://www.alexa.com/topsites/countries

[2] *Attackers Use Morse Code, Other Encryption Methods in Evasive Phishing Campaign*. Accessed: Dec. 17, 2021. [Online]. Available: https://www.microsoft.com/security/blog/2021/08/12/attackers-use-morse-code-other-encryption-methods-in-evasive-phishing-campaign/

[3] *Common Crawl*. Accessed: Mar. 25, 2021. [Online]. Available: https://commoncrawl.org/

[4] *Google Custom Search*. Accessed: Mar. 4, 2019. [Online]. Available: https://developers.google.com/custom-search/

[5] *Phishers' Favorites: After Five Quarters, Microsoft is Unseated by PayPal*. Accessed: Apr. 12, 2020. [Online]. Available: https://www.vadesecure.com/en/phishers-favorites-q3-2019/

[6] *Phishers' Favorites: It's Lonely at the Top: Microsoft Remains the #1 Impersonated Brand in Phishing Attacks*. Accessed: Jan. 28, 2020. [Online]. Available: https://www.vadesecure.com/en/phishers-favorites-q1-2019/

[7] *Phishers' Favorites: Microsoft Holds Onto the #1 Spot but Facebook Phishing Surges*. Accessed: Apr. 12, 2020. [Online]. Available: https://www.vadesecure.com/en/phishers-favorites-q2-2019/

[8] *Phishers' Favorites: PayPal Leads, Note Phishing Increases, and Smaller Banks Become Bigger Targets*. Accessed: Apr. 12, 2020. [Online]. Available: https://www.vadesecure.com/en/phishers-favorites-q4-2019/

[9] *PhishTank: An Anti-Phishing Site*. Accessed: Oct. 1, 2018. [Online]. Available: https://www.phishtank.com/

[10] *The Chromium Projects: Chromium*. Accessed: Jul. 27, 2020. [Online]. Available: https://www.chromium.org/Home

[11] *Using Custom Elements*. Accessed: Dec. 17, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_custom_elements

[12] A. Aleroud and L. Zhou, "Phishing environments, techniques, and countermeasures: A survey," *Comput. Secur.*, vol. 68, pp. 160–196, Jul. 2017.

[13] *APWG Phishing Trends Reports*, Anti Phishing Working Group, San Francisco, CA, USA, 2016.

[14] *APWG Phishing Trends Reports*, Anti Phishing Working Group, San Francisco, CA, USA, 2018.

[15] L. Araujo and J. Martinez-Romo, "Web spam detection: New classification features based on qualified link analysis and language models," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 3, pp. 581–590, Sep. 2010.

[16] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar web pages: Application to phishing detection," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 1–38, May 2010.

[17] R. Cilibrasi and P. M. B. Vitányi, "Clustering by compression," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1523–1545, Apr. 2005.

[18] Q. Cui, G.-V. Jourdan, G. V. Bochmann, R. Couturier, and I.-V. Onut, "Tracking phishing attacks over time," in *Proc. 26th Int. Conf. World Wide Web*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, Apr. 2017, pp. 667–676.

[19] 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 Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2797–2819, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8036198/

[20] J. Feng, Y. Zhang, and Y. Qiao, "A detection method for phishing web page using DOM-based Doc2Vec model," *J. Comput. Inf. Technol.*, vol. 28, no. 1, pp. 19–31, Jul. 2020.

[21] A. Y. Fu, L. Wenyin, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD)," *IEEE Trans. Depend. Sec. Comput.*, vol. 3, no. 4, pp. 301–311, Oct. 2006.

[22] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theor. Comput. Sci.*, vol. 38, pp. 293–306, 1985.

[23] G. Ho *et al.*, "Detecting and characterizing lateral phishing at scale," in *Proc. 28th USENIX Secur. Symp. (USENIX Security)*, 2019, pp. 1273–1290.

[24] J. Hong, "The state of phishing attacks," *Commun. ACM*, vol. 55, no. 1, pp. 74–81, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2063176.2063197

[25] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 4th Quart, 2013.

[26] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2019, pp. 1–15.

[27] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, "The similarity metric," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3250–3264, Dec. 2004.

[28] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, New York, NY, USA, 2009, pp. 681–688, doi: 10.1145/1553374.1553462.

[29] A. Neupane, N. Saxena, J. O. Maximo, and R. Kana, "Neural markers of cybersecurity: An fMRI study of phishing and malware warnings," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 9, pp. 1970–1983, Sep. 2016.

[30] C. Opara, B. Wei, and Y. Chen, "HTMLPhish: Enabling phishing web page detection by applying deep learning techniques on HTML analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[31] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *Proc. 28th USENIX Secur. Symp. (USENIX Security)*, 2019, pp. 729–746.

[32] *2019 Phishing Trends and Intelligence Report—The Growing Social Engineering Threat*, PhishLabs, Charleston, SC, USA, 2020.

[33] R. Purwanto, A. Pal, A. Blair, and S. Jha, "PhishZip: A new compression-based algorithm for detecting phishing websites," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Jun. 2020, pp. 1–9.

[34] F. Quinkert, T. Lauinger, W. Robertson, E. Kirda, and T. Holz, "It's not what it looks like: Measuring attacks and defensive registrations of homograph domains," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Jun. 2019, pp. 259–267.

[35] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *J. Comput. Secur.*, vol. 19, no. 4, pp. 639–668, 2011.

[36] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proc. NDSS*, 2010, pp. 1–14. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/10/pdf/08.pdf

[37] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 1–28, Sep. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2019599.2019606

[38] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 571–580.

[39] H. Zhang, G. Liu, T. W. S. Chow, and W. Liu, "Textual and visual content-based anti-phishing: A Bayesian approach," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1532–1546, Oct. 2011.

[40] X. Zhang, Y. Zeng, X.-B. Jin, Z.-W. Yan, and G.-G. Geng, "Boosting the phishing detection performance by semantic analysis," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 1063–1070.

[41] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," in *Proc. 16th Int. Conf. World Wide Web*, New York, NY, USA, 2007, pp. 639–648. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242659

Rizka Widyarini Purwanto (Graduate Student Member, IEEE) received the bachelor's degree in electrical engineering from Institut Teknologi Bandung, Indonesia, in 2013, and the master's degree from the University of New South Wales (UNSW) in 2018, concentrating on artificial intelligence and internetworking, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her research interests include cybersecurity and artificial intelligence. She was also a recipient of the University International Postgraduate Award (UIPA) Scholarship.

Arindam Pal (Senior Member, IEEE) received the Ph.D. degree in computer science from the Indian Institute of Technology Delhi, New Delhi, India. He is currently a Senior Research Scientist with Data61 in Commonwealth Scientific and Industrial Research Organization (CSIRO), and a Conjoint Senior Lecturer at UNSW Sydney. He works on business and research problems of CSIRO, and collaborates with faculty members of universities, both in Australia and abroad. He has more than 13 years of industrial research experience in software companies like Microsoft, Yahoo, and Novell. He has published academic papers in reputed conferences and journals, and filed patents in various countries, such as India, USA, and Europe. His research interests are in artificial intelligence, cyber security, and machine learning. He is a Senior Member of ACM.

Alan Blair received the B.Sc. and B.A. degrees from The University of Sydney, Sydney, Australia, and the Ph.D. degree in mathematics from the Massachusetts Institute of Technology (MIT). He went on to do research in computer science at the University of Brandeis, The University of Queensland, and The University of Melbourne before taking up his current position at the University of New South Wales (UNSW), Sydney. He is currently a Senior Lecturer with the School of Computer Science and Engineering, UNSW. His research interests include machine learning, evolutionary and emergent computation, coevolution and games, information geometry and sensor fusion, robot control and navigation, recurrent neural networks, and language processing.

Sanjay Jha (Senior Member, IEEE) is currently a Full Professor and the Director of the Cybersecurity and Privacy Laboratory, School of Computer Science and Engineering, University of New South Wales, and leads the UNSW Cybersecurity Cooperative Research Centre. His research interests include wireless mesh/sensor networks (IoT), and network security. He is the principal author of the book *Engineering Internet QoS* and a co-editor of the book *Wireless Sensor Networks: A Systems Perspective*. His editorial affiliations include the IEEE TRANSACTIONS ON MOBILE COMPUTING (TMC) and the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC).