Computers & Security 136 (2024) 103545

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning

Arvind Prasad *, Shalini Chandra

Department of Computer Science, BBA University, Lucknow, India

ARTICLE INFO

Keywords:
Phishing URL detection
PhiUSIIL phishing dataset
Similarity index
Incremental learning
Cybersecurity

ABSTRACT

With the proliferation of the World Wide Web and the increasing sophistication of cyber threats, phishing attacks have emerged as a significant concern for individuals and organizations alike. Phishing attacks, commonly executed through deceptive URLs, aim to deceive users into divulging sensitive information, leading to financial loss, identity theft, or compromising sensitive data. It continues to pose a significant threat to individuals and organizations in today's digital landscape, necessitating the development of effective and efficient detection frameworks. This article presents PhiUSIIL, a **Phi**shing **U**RL detection framework based on **S**imilarity **I**ndex and **I**ncremental **L**earning. The similarity index helps effectively identify visual similarity-based attacks such as zero-width characters, homograph, punycode, homophone, bit squatting, and combosquatting attacks. The incremental learning approach allows the framework to continuously update its knowledge base with new data. Further, implementing diverse security profiles accommodates diverse security requirements of users or organizations. PhiUSIIL extracts URL features, downloads the webpage from URL to extract HTML features, and derives new features from existing information to construct a phishing URL dataset, named PhiUSIIL phishing URL dataset, encompassing 134850 legitimate and 100945 phishing URLs. The proposed phishing URL detection framework has extensively experimented with the PhiUSIIL phishing URL dataset. The constructed dataset helps to improve the detection accuracy when used during pre-training approach. PhiUSIIL achieved an accuracy of 99.24% when experimented with a fully incremental training approach and 99.79% when experimented with a pre-training approach. The experimental results show its effectiveness and ensure the framework remains effective and up-to-date against emerging and sophisticated phishing techniques.

1. Introduction

The exponential growth of the Internet and Internet-enabled devices have revolutionized the way we access online services such as education, finance, and healthcare. The continuous development of digital technologies, such as high-speed Internet, more friendly Internet access devices, and cloud services, enhances the end-user experience daily. As a result, the rapidly evolving interconnected digital landscape has made online activities an integral part of our daily lives. Furthermore, the pandemic caused by COVID-19 played a pivotal role in driving digital transformation and triggered a massive shift of businesses and services from offline to online to minimize physical interactions (Nurhas et al., 2022). The pandemic compelled all organizations, industries, governments, and individuals to rapidly adopt digital transformation to ensure continuity during uncertainty. Digital transformation presents new avenues for growth and is key to success in the increasingly interconnected digital world. However, while technological advancements offer tremendous opportunities, they also bring forth a surge in cyber threats that cannot be ignored (Kávrestad et al., 2022). The rapid proliferation of interconnected internet-enabled devices such as laptops, mobiles, smart gadgets, smart home appliances, and smart cars has created an expansive attack surface, leading to an alarming upswing in sophisticated cyberattacks (Shih et al., 2023).

Phishing, a prevalent cyberattack, poses a significant threat to individuals, organizations, and society. It typically involves deceptive tactics, such as fraudulent URLs, aimed at tricking unsuspecting users into revealing sensitive information, including passwords, financial details, or personal data. URL visual similarity-based attacks such as zero-width characters, homograph, punycode, homophone, bit squatting, and combosquatting attacks are techniques cyber criminals use to create fraudulent websites that closely resemble legitimate ones (Sahingoz et al., 2019). While each attack has its technical nuances, they all exploit

* Corresponding author.
E-mail address: arvindbitm@gmail.com (A. Prasad).

https://doi.org/10.1016/j.cose.2023.103545
Received 26 June 2023; Received in revised form 10 October 2023; Accepted 16 October 2023
Available online 20 October 2023

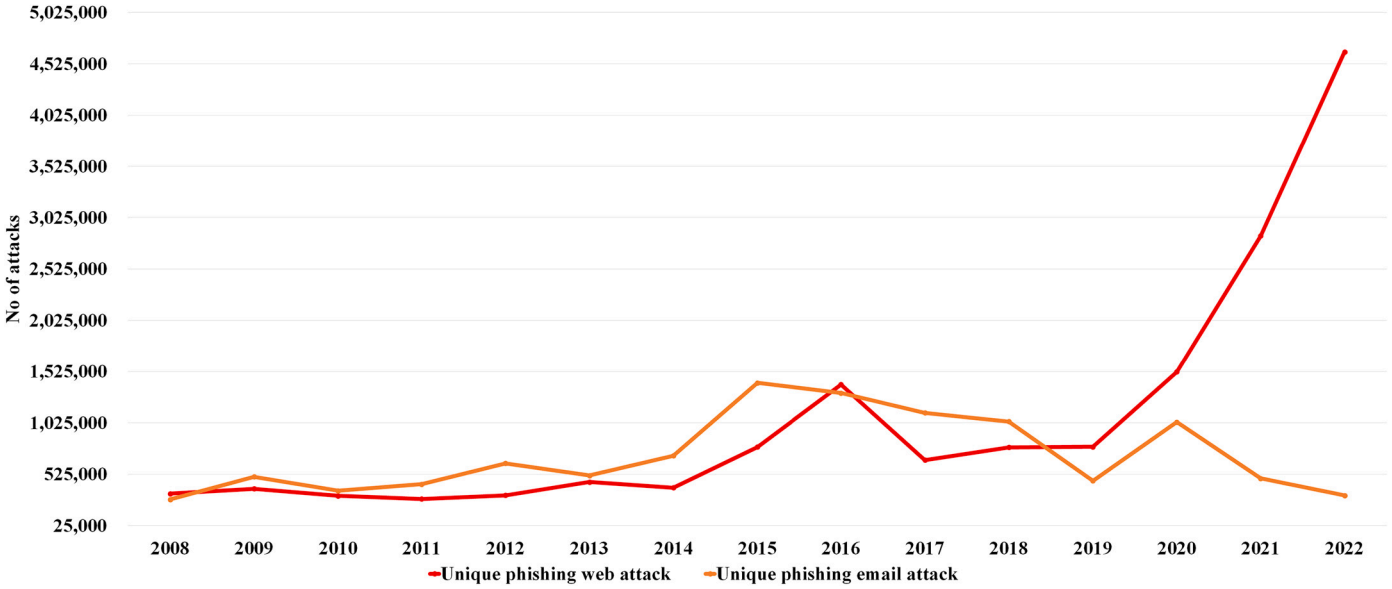

Fig. 1. Increase in phishing URL attacks post pandemic. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

human error, visual similarity, or typo mistakes to deceive users into visiting phishing websites that often mimic legitimate ones.

Cybercriminals often use subdomain spoofing techniques by creating a subdomain resembling a legitimate website to convince users they are accessing a legitimate website (Maroofi et al., 2021). Malicious actors can use transposition to swap adjacent characters in a domain name to create a visually similar but phishing website. Top-level domains (TLDs), the last segment of a domain name, are extensively used by malicious actors to create a domain using a TLD that resembles well-known and trusted extensions to trick users into visiting fake websites (Anon, 2023a). The newly released TLDs .zip and .mov have allowed cybercriminals to register a domain resembling well-known file extensions. Sending such domain names could deceive users into thinking they are downloading a harmless file when, in reality, it is a phishing URL. Within a few days after the release of .zip and .mov TLDs, cybercriminals have already registered many phishing domains (Anon, 2023b). Phishers often use TLDs that closely resemble well-known brands or trusted entities to trick users into believing they are visiting legitimate websites.

The exponential growth of URL phishing attacks has become increasingly evident in recent years. We collected the last fifteen years' phishing URLs and email data from the Anti-Phishing Working Group (APWG) (Anon, 2023c) and plotted a graph in Fig. 1. The graph shows that the outbreak of the COVID-19 pandemic has led to an immense increase in phishing URL attacks. Cybercriminals exploit the fear, heightened emotions, uncertainty, medical urgency, increased online activity, and information-seeking behavior of individuals during times of crisis. Fig. 1 shows that the growth of phishing URL attacks is at an alarming rate after the outbreak of the COVID-19 pandemic.

Ensuring online security is paramount in the digital age, where the Internet has become integral to our daily lives. The consequences of phishing attack can be dire, ranging from compromising sensitive data to financial loss (Wen et al., 2023). The exponential growth of URL phishing attacks necessitates robust and effective defense mechanisms to counter these threats. Unfortunately, traditional rule-based approaches fail to combat the evolving tactics employed by cybercriminals. Machine learning has emerged as a powerful tool in the battle against various cyberattacks (Zhang et al., 2017; Prasad and Chandra, 2022a; Unal et al., 2021; Xiao et al., 2021). It has the ability to analyze vast amounts of data and identify patterns that are indicative of phishing URLs (Prasad and Chandra, 2022b; Bountakas and Xenakis, 2023; Rupa et al., 2021).

Phishing attacks constantly evolve, and new attack techniques and patterns emerge regularly. It is crucial to keep the machine learning model updated by training them with new data to improve model adaptability, accuracy, and real-time detection. Classical machine learning algorithms require periodic retraining on the entire dataset (including existing and newly collected data) to incorporate changes. When enough new and unseen data is collected, the classical machine learning model must be retrained from scratch to prevent excessive model staleness (Prapas et al., 2021). Retraining brings scalability issues as we need to store the entire training dataset continuously. As the dataset grows, it needs more reprocessing to train the model on the growing dataset.

The proposed machine learning-based phishing URL detection approach is based on incremental learning. Incremental learning is a machine learning technique where a model can continuously learn from new data, reducing the need for full retraining and minimizing the disruption caused by the retraining process. While the limitation of batch learning is its dependence on retraining the entire dataset whenever new data is introduced, this process can be computationally expensive and time-consuming, making it less suitable for dynamic environments. Incremental learning enables a model to continuously adapt and improve its performance by continuous training whenever it receives new data rather than starting from scratch every time it gets new data (Purwanto et al., 2022; Huang and Ma, 2023). The continuous learning nature of incremental learning makes it highly scalable and more efficient in terms of time and computational resources. It enables the model to stay up-to-date, making it adaptable to changing data.

This research article aims to develop, implement, and evaluate a scalable and adaptable phishing URL detection framework that leverages similarity index and incremental learning techniques to combat phishing attacks effectively. The objectives encompass constructing a comprehensive phishing URL dataset, proposing a URL similarity index for detecting visual similarity-based attacks, designing an adaptable phishing URL detection framework with diverse security profiles, and enhancing model performance through pre-training to improve robustness and generalization.

The major contribution of this research work is as follows.

- An extensible dataset construction approach is proposed and implemented to construct a large phishing URL dataset encompassing 134850 legitimate and 100945 phishing URLs.

- A URL similarities index calculation technique is proposed that helps to detect URL visual similarity-based attacks such as zero-width characters, homograph, punycode, homophone, bit squatting, and combosquatting attacks.
- A diverse security profile empowered incremental lea-rning-based scalable phishing URL detection framework is proposed that provides better scalability and adaptability and prevents model staleness problems. The proposed model performance enhancement approach leverages readily available data to pre-train the incremental learning model to enhance its robustness and generalization capabilities.

The rest of the paper is organized as follows. Section 2 provides background on existing phishing attack detection methods and challenges that need to be addressed while developing a modern and sophisticated phishing URL detection system. Further, Section 3 explains our approach to construct the phishing URL dataset. In Section 4, we propose URL similarity index (USI) and incremental learning-based phishing URL detection technique. The experimental result is presented and discussed in Section 5. Despite its potential, the proposed framework has limitations, discussed in 6. Finally, we conclude the paper in section 7 with a summary of our contributions.

2. Related work and challenges

A thorough understanding of the existing literature and studies on machine learning-based phishing attack detection techniques is crucial for identifying the gaps, limitations, and opportunities that drive our research forward. By examining the relevant work conducted by other researchers, we gain valuable insights into the current state of the field and the challenges that have been encountered. The following subsections delve into the related work and challenges surrounding machine learning-based phishing attack detection techniques.

2.1. Related work

Various researchers have developed techniques to detect and mitigate phishing URL attacks to combat the growing menace of phishing URL attacks. This subsection will discuss the machine learning-based approaches closely related to the proposed work. Further, it summarizes the key findings and methodologies employed in each study. This exploration of the existing literature provides a foundation for our research.

Gupta et al. (2021) proposed an innovative machine-learning approach to detect phishing URLs. Their approach involves extracting and analyzing lexical features present within the URLs, presenting a powerful tool for combating fraudulent phishing URL attacks. Although the researcher extracted many features from the URL, with the help of feature selection, they selected only nine features by implementing multiple feature importance algorithms. The ML model's dependence on the reduced number of features makes it best suitable for resource-constrained devices. RF, SVM, KNN, and LR algorithms have been experimented on ISCXURL-2016, and RF achieved the highest accuracy of 99.57%. Pandey and Mishra (2023) proposed Phish-Sight, which takes a screenshot of a test webpage and matches it with the screenshot of its counterpart legitimate website based on a visual inspection similar to the human approach of distinguishing two similar sites. For visual similarity inspection, Phish-Sight implements dominant color features, eight RGB color palette features, and optical character recognition (OCR) to extract the eighteen most targeted brand names. Both are extracted from the test website screenshot. Five classical ML techniques, namely DT, LR, NB, RF, and SVM, were implemented on the extracted features to investigate the test URL. SVM outperformed other ML algorithms, achieving an accuracy of 99.13%.

To construct a dataset, Ahammad et al. (2022) collected linguistic and domain-based properties from 1500 phishing and 1500 legitimate URLs. The dataset includes statistical information from phishtank.com and DNS information from WHOIS. It also includes existence symbols such as '@' or '-' symbol, http or https information based on the position of '//,' the length of the URL, and the number of subpages. The researchers employed five machine learning models, namely RF, DT, LightGBM, LR, and SVM, respectively, to build an ML model for identifying phishing URLs. In their experiment, based on classification accuracy and analysis of the validation curve, LightGBM outperformed other models. An efficient malicious mobile webpage detection approach, APuML (Anti Phishing using Machine Learning), is proposed by Jain et al. (2022) that detects drive-by downloads, zero-day, and clickjacking attacks effectively. APuML extracts features of various webpage components, such as URL features, mobile-specific features, HTML features, JavaScript features, and website-specific features, and creates feature vectors for 2000 legitimate and 2000 phishing webpages. Further, a mutual information-based feature selection technique is implemented, and a set of eleven features that includes nine binary and two Naive Bayes probability features are identified. Finally, the researchers experimented with various machine learning algorithms, and the random forest algorithm achieved the highest detection accuracy of 93.85%.

Alani and Tawfik (2022) proposed PhishNot, a machine learning-based phishing URL detection system that relies on very few features extracted from URLs. The researcher achieved high accuracy by experimenting with several algorithms and selecting the one with the highest accuracy, easy implementability by relying on a few features extracted from URLs, high efficiency by lowering the data acquisition time and prediction time, and ease of access by deploying it as a plug-in in browsers, or any other deployment architecture. PhishNot has experimented with RF, LR, DT, GNB, and MLP machine-learning algorithms. Random forest achieved the highest accuracy of 97.5%. Ding et al. (2019) proposed Search & Heuristic Rule & Logistic Regression (SHLR) for detecting phishing websites. SHLR includes three steps to filter legitimate websites efficiently. Initially, it extracts the title tag from the website and searches it on the Baidu search engine to find its rank. If the ranking is under 10, it is considered legitimate; otherwise, the second step is involved further analysis. In the second step, seven heuristic rules are developed based on character features to detect obfuscated URLs. Then, it checks the URL against these rules. Those URLs matched with the rules and were identified as pushing; otherwise, the third step is called for further analysis. In the third step, a machine-learning model is built using logistic regression. It extracts relevant features from the webpage and trains the model. During the experiment, SHLR achieved an accuracy of 98.9%.

Sharma and Singh (2022) proposed a phishing detection technique that implements term frequency and inverse document frequency (TF-IDF) to analyze webpage and measure the quantity of information a word provides based on the word's frequency in the document. Further, the AdaBoost algorithm is employed to enhance the classification accuracy of the proposed technique. In their comprehensive research endeavor, the researchers investigated a dataset of 50000 URLs, yielding impressive results in terms of recognition accuracy when differentiating between phishing sites and legitimate ones. Furthermore, it was experimentally shown that the analysis and calculation performed before machine learning improved the proposed technique's overall performance. The technique achieved an accuracy of 98.014%. Nagunwa et al. (2022) proposed an ML approach to detect phishing webpages hosted in Fast Flux Service Networks (FFSNs). The devices frequently change in such networks, which makes the blocklisting technique less effective. The proposed approach introduces 56 highly diversified features derived from the characteristics of the hosting network. These predictive features are further analyzed using eight machine learning and three deep learning algorithms powered by binary and multi-class classification models. The binary classification approach achieved an accuracy of 98.42%, effectively distinguishing between phishing and legitimate URLs. Similarly, in multi-class classification, the researchers achieved an accuracy of 97.81%, further affirming their proposed methodology's

Table 1
Comparative summary of related work with proposed work.

Ref.	Features	Detection model	Dataset records	Accuracy	Limitations
Gupta et al. (2021)	URL features only	Depends on Random forest algorithm	11964	99.57%	Depends solely on URL features, and on single ML algorithm, experimented on small dataset
Pandey and Mishra (2023)	Dominant color features and OCR	Depends on Random forest algorithm	6200	99.13%	Depends solely on single ML algorithm, experimented on small dataset
Ahammad et al. (2022)	URL Features only	RF, DT, Light GBM, LR, and SVM	3000	89.50%	Low prediction performances, experimented on small dataset
Jain et al. (2022)	Static and site popularity features	LR, KNN, SVM, DT, and RF	4000	93.85%	Depends on third party features, low prediction performances, experimented on small dataset
Alani and Tawfik (2022)	URL and third-party features	RF, LR, DT, GNB, and MLP	88646	97.50%	Depends on third party features, low prediction performances
Ding et al. (2019)	URL, HTML and third-party features	Logistic regression	8659	98.90%	Depends on third party features, depends on single ML algorithm, small dataset
Sharma and Singh (2022)	Features from webpage HTML code	TF-IDF and AdaBoost	50000	98.01%	Depends on single ML algorithm, small dataset
Nagunwa et al. (2022)	Features derived from DNS, host, and network	Eight ML and three DL algorithms	11801	98.42%	Computationally expensive model (11 algorithms), small dataset
Sameen et al. (2020)	URL Features only	Boosting-based (2), Ten algorithms	100000	98.00%	Computationally expensive model (10 algorithms)
Rao et al. (2022)	HTML code, and domain specific features	RF, SVM, LR, DT, and XGBoost	10514	99.34%	Experimented on small dataset
Proposed	URL, HTML, and derived features	BernoulliNB, PassiveAggressive, and SGDClassifier	235795	99.79%	Limitation of classifying URLs that download executable file.

robustness and reliability in accurately classifying URLs across multiple categories.

Sameen et al. (2020) introduced PhishHaven, a cutting-edge system specifically tailored to detect both AI-generated and human-crafted phishing URLs. PhishHaven's innovative approach harnesses the power of ensemble learning, providing an effective defense against a wide range of sophisticated phishing attacks. The researcher implements lexical analysis for extracting useful features. The URL HTML Encoding is further introduced as lexical feature to boost the detection capability. PhishHaven exhibits proficiency in detecting tiny URLs through its advanced URL Hit approach. By meticulously opening each URL and comparing the resulting reply URL with the requested tiny URL, PhishHaven effectively identifies and scrutinizes these condensed URLs, providing a comprehensive defense against potential phishing threats. Finally, a voting-based ensemble learning technique is implemented to detect phishing URLs. PhishHaven has experimented on 100000 legitimate and phishing URLs and achieved an accuracy of 98%. Rao et al. (2022) proposed a phishing detection technique that does not depend on third-party information such as webpage ranking, webpage age, or search engine indexing. The proposed technique is based on plain text from webpage code and domain-specific text. The plain text is extracted from the webpage source and fed to frequency (TF-IDF and count vectorization) and prediction (Word2Vec, GloVe, and FastText) based word embedding algorithm to create a word dictionary. The domain-specific texts are copyright information, webpage title, and HTML headers. Both plain text and domain features are used to create feature vectors. Further, the feature vector is fed into both ensemble and multimodal machine learning algorithms. The multimodal approach achieved higher accuracy with 99.34%.

In conclusion, the extensive research in machine learning-based phishing URL detection has demonstrated significant progress and promising results. However, several key observations and requirements have emerged from the reviewed literature, underscoring the need for a comprehensive framework to address the challenges and limitations in this domain. The major challenges identified in the reviewed work are discussed in detail in the next subsection.

Furthermore, a comparative summary is provided in Table 1 that shows the advantages of the proposed work over the state-of-the-art.

2.2. Challenges

Despite significant development against phishing URL attacks, it continues posing serious threat to online security. The existing approaches to combat phishing attacks seem insufficient to address the ever-evolving sophisticated phishing techniques and the emergence of previously unseen phishing URLs. Based on the above discussion, we identified the following major areas that need to be addressed while developing a modern and sophisticated phishing URL detection system and where our research aims to make a significant impact.

Continuous training: Phishing attacks constantly evolve, and new attack techniques and patterns emerge regularly. By continuously training the machine learning model, it can adapt to the changing landscape of cyber threats. Continuous training of machine learning models helps improve adaptability, accuracy, early detection of zero-day attacks, and enhanced behavioral analysis. Regular updates help the model stay current and learn from new attack patterns, making it more effective in identifying and mitigating emerging phishing threats. Most existing studies implement classical machine learning algorithms to build the model. Researchers train the model once and continue using the same trained model to detect attacks. It prevents a model from gaining exposure to a wider range of legitimate and phishing URL scenarios. Classical machine learning algorithms often struggle to adapt to the evolving nature of phishing attacks, as they require periodic retraining to maintain optimal performance.

Latest and large dataset: Phishing attack techniques and strategies constantly evolve as cybercriminals find new ways to deceive end users. By incorporating the latest datasets, the machine learning model can learn from recent attack patterns and identify new phishing techniques. A large phishing URL dataset enables the model to learn from a wide range of attack vectors and improve its ability to detect phishing attacks effectively. Many researchers have trained the model on old and small datasets, which does not accurately evaluate and benchmark the machine learning model against the most recent phishing attacks.

Dependencies on third-party features: Many existing studies depend on third-party features such as age, rank, and country where the URL is registered. Relying on these features introduces a dependency on their availability and reliability. If the third-party provider experiences downtime or discontinues their service, it can significantly impact the model. Therefore, dependence on third-party features can

have potential drawbacks and negative impacts on the model's long-term sustainability.

Dependency solely on URL features: The dependency of a machine learning model solely on URL features restricts the model's ability to capture other important contextual information, such as webpage content. For example, webpage content provides huge amounts of information that can be used to detect phishing URLs more accurately.

Dependencies on a single ML algorithm: Many developed defense systems against a phishing attack depend on a single machine learning model. However, every machine learning algorithm has its strengths and weaknesses. Relying solely on a single machine learning algorithm may result in an inability to handle diverse data and limit the capability to detect the constantly evolving phishing attacks.

In summary, examining the challenges identified in the previous research highlights the critical areas that require further attention and innovation in machine learning-based phishing attack detection. Furthermore, these challenges are the impetus for developing a comprehensive framework to address these limitations and propel the field forward.

3. Proposed framework

Phishing attacks continue to pose significant threats in today's digital landscape. In response to this growing problem, this research work presents PhiUSIIL, a phishing URL detection framework combining two key components: URL similarity index and incremental learning. This combination provides a robust and adaptive approach to detecting phishing URLs in real-time. PhiUSIIL's unique contribution lies in its focus on developing a diverse security profile that accommodates diverse user requirements. We include the above discussed in PhiUSIIL's phishing URL detection module. In addition, the PhiUSIIL framework has an extensible dataset construction module to construct a phishing URL dataset called PhiUSIIL phishing URL dataset. The PhiUSIIL phishing URL dataset can be used as a benchmark dataset to pre-train the incremental learning model. We will start by explaining the PhiUSIIL phishing URL dataset construction module and then move to the next section to discuss the phishing URL detection module.

3.1. PhiUSIIL phishing URL dataset construction module

The following subsection discusses the approach of constructing an extensible phishing URL dataset. We collected legitimate URLs from Open PageRank Initiative Anon (2023g) and phishing URLs from PhishTank, OpenPhish, and MalwareWorld. We start with explaining the steps to download the webpages using the collected legitimate and phishing URLs. Further, features that are critical to detecting phishing URLs are discussed.

3.1.1. Downloading phishing webpage

Phishing websites are created to steal sensitive information from unsuspecting users by tricking them into believing that the URL they are accessing is genuine. These websites are taken down by internet service providers or hosting providers once security researchers or organizations discover their suspicious activity. Cybercriminals abandon the site quickly to create a new and more convincing phishing site. The quick discovery of phishing websites makes its lifespan incredibly short (Wei et al., 2020). Therefore, it is important to download such websites before they are blocked or removed. We decided to monitor various anti-phishing sites continuously to get the list of latest phishing websites. Anti-phishing sites such as PhishTank Anon (2023d), OpenPhish Anon (2023e), and MalwareWorld Anon (2023f) actively monitor the internet to discover new phishing websites.

Once we get a new phishing URL, we download the webpage referred to by the phishing URL. We try to download them before they disappear. Accessing malicious webpages through web browsers is risky; we set up a separate environment to download the phishing webpage programmatically and stored it locally as a .txt file. A Java program is developed that takes PhishID and PhishURL as input to read the webpage referred by the PhishURL and save it locally with the name PhishID.txt. The one-line Java code to do these tasks is given here.

```
Files.writeString(Paths.get(PhishID+".txt")
,HttpClient.newHttpClient().send(HttpRequest
.newBuilder().uri(URI.create(PhishURL)).GET()
.build(),HttpResponse.BodyHandlers.ofString())
.body(),StandardCharsets.UTF_8);
```

Where, **ID** is a unique ID for each phishing URL and **PhishURL** is the URL of the phishing webpage.

We continuously monitored anti-phishing sites and downloaded over 100000 phishing webpages from October 2022 to May 2023. The above-discussed approach is used to download over 100000 legitimate webpages used to construct the dataset.

3.1.2. Extracting URL features

URL features can be an important aspect of phishing URL detection as they provide valuable information about the legitimacy of a webpage. Machine learning models can use these features to analyze a URL to identify phishing URLs. Although the URL feature has a high potential to identify a phishing URL, we cannot depend only on these features. A cybercriminal may learn the URL-based detection mechanism and create more sophisticated URLs to invade the system. Some of the key URL features we extracted to construct the dataset are discussed here.

TLD: TLD (Top Level Domain) is the last part of the domain name, such as .com or .edu. Phishing URLs often use TLDs that are not commonly associated with the legitimate domain, such as .link or .inf.

URLLength: Phishing URLs are often longer than legitimate URLs as they contain additional digits, symbols, subdirectories, and parameters.

IsDomainIP: a URL using an IP address instead of a domain name can be a red flag for users.

NoOfSubDomain: Subdomain is part of URL that appears before domain name. Cybercriminals often use visual similarity techniques to trick users. They create subdomains that look like subdomain of legitimate websites.

NoOfObfuscatedChar: Shows a count of obfuscated characters in URL. Obfuscated URL https://abc.com%6d opens https://abc.comm and obfuscated URL https://facebook.com@%61%62%63.%43%4F%4D opens https://abc-.com and try to login https://abc.com.

IsHTTPS: Indicates if the webpage is running on unsecured HTTP (hypertext transfer protocol) or secured HTTPS. A URL using the http protocol is highly likely to be a phishing URL. Most legitimate websites, especially those that require users to input sensitive information like passwords or credit card numbers, use HTTPS to protect their users' data. If a webpage asks for sensitive information but doesn't use HTTPS, it could be a sign that the webpage is a phishing scam.

No. of digits, equal, qmark, amp: A large number of digits or symbols such as '=', '?', or '%' in a URL increases the possibility of being a phishing URL.

3.1.3. Extracting HTML features

HTML (Hypertext Markup Language) is a standard markup language used to create webpages. Inspecting the HTML code of a webpage can help detect phishing sites by identifying suspicious elements or code that may be used to collect sensitive information, trick users into making monetary transactions, or redirect users to a phishing website. Certain HTML features that can be used to identify potentially suspicious or fraudulent websites are extracted from the webpage HTML code to construct the dataset. Some of such key HTML features are discussed here.

LargestLineLength: Cybercriminals may use encrypted text to hide the actual code from the user. A longer line of code may indicate code obfuscation.

HasTitle: Most legitimate websites have page titles provided. Missing title tags may indicate a phishing scam.

HasFavicon: Most legitimate websites have their website logo included in the favicon tag. Missing a favicon tag may indicate a phishing scam.

IsResponsive: Most legitimate websites are responsive, which helps web content to be appropriately adapted across devices to give better readability and view. Fortunately, many phishing websites are not responsive, as threat actors find it challenging to ensure the responsiveness of their quickly designed websites on all major devices.

NoOfURLRedirect: Phishing sites may use redirects to direct users to a different page than they were expecting. For example, the HTML code may contain JavaScript or meta tags that redirect users to a different URL. The HTML tags such as 'http-equiv', 'refresh', 'window.location', 'window.location.replace', 'window.location-.href', 'http.open' can help identify URL redirection.

HasDescription: Legitimate websites provide page descriptions for each page using the 'description' meta name. Missing page descriptions may raise a red flag for a webpage.

NoOfPopup, NoOfiFrame: Phishing websites may use pop-ups or iframe to distract users and capture sensitive information. These pop-ups and iframe can be detected by looking for tags 'window.open' and 'iframe' in the HTML code.

HasExternal FormSubmit: Phishing sites often use HTML forms to collect user information. Form submitting to an external URL can be a red flag for users.

HasCopyrightInfo, HasSocialNet: Most legitimate websites have copyright and their social networking information. Missing such information may indicate a phishing scam.

HasPasswordField, HasSubmitButton: HTML provides a variety of form elements that allow users to input data and submit it to other URLs. For example, HTML tags such as 'passwordfield' or 'submitbutton' can be extracted to examine the HTML code of the webpage.

HasHiddenFields: Phishing websites may use hidden fields to capture sensitive information without the user's knowledge. These fields can be detected by examining the HTML code of the webpage.

Bank, Pay, Crypto: Elements such as bank, pay, or crypto may indicate that the webpage is asking for sensitive financial information from the user, which may be used to siphon money. Therefore, such websites need to be analyzed for suspicious activities.

NoOfImage: Threat actors can use screenshots of legitimate websites and design phishing websites to make them appear more legitimate. More images used in respectively small websites may indicate phishing websites.

NoOfJS: JavaScript is a programming language that can be embedded in HTML to create interactive webpages. Phishing websites may use JavaScript to create pop-up windows or other misleading elements that trick users into revealing sensitive information. A large number of JavaScript included in a webpage can make it suspicious.

NoOfSelfRef, NoOfEmptyRef, NoOfExternalRef: Hyperlinks (href) are clickable links that allow users to navigate between webpages or navigate to external webpages. Phishing websites may use hyperlinks that appear to direct to a legitimate webpage, but instead, they redirect the user to a phishing page. A large number of hyperlinks navigating to itself, navigating to empty links, or navigating to external links can be suspicious.

3.1.4. Crafting derived features

CharContinuationRate: CharContinuationRate indicates how a URL contains the alphabet, digit, or special characters. To find the CharContinuationRate of a URL, we identified the longest alphabet, digit, and special character sequences and total their length. Further, the total length of these sequences is divided by the totaled length of the URL. URLs such as www.abc.com, www.abc123.com, or www.abc_123.com can give one CharContinuationRate value. However, URLs such as www.a1b2c3.com or www.a1_b-_2.com get lower CharContinuationRate.

URLTitleMatchScore: Cybercriminals often use social engineering tactics to trick users into believing a website is legitimate. They may use a URL that looks similar to a legitimate website and create a convincing webpage title reflecting the website's content. We introduced URLTitleMatchScore to identify the discrepancy between the URL and the webpage title. A lower score can be a sign that the website is a phishing attempt because the webpage title does not match the content that is expected to be found on the website. A higher score 100 or close to 100 indicates that the website is what it claims to be. The code to calculate URLTitleMatchScore is given in Algorithm 1.

URLCharProb: While most legitimate URLs look meaningful, many phishing URLs contain random alphabet, digits, and misspelled words that do not look meaningful. Often, an attacker uses the typosquatting technique to create a URL similar to a legitimate URL but with small typographical errors. To understand the pattern of each alphabet and digit in a URL, we count the occurrence of each alphabet and digit in the 10 million legitimate URLs and divide them by the total count of all alphabets and digits of 10 million legitimate URLs. Further, to compare it with the pattern of phishing URLs, we collected 7 million phishing URLs and calculated the probability of each alphabet and digit using the same method. Finally, a comparative graph is plotted in Fig. 2. It shows that most of the characters such as a, c, e, o, r, and t are more frequently used in legitimate URLs, while alphabets such as b, f, q, v, w, x, y, z, and all the digits are more frequent in phishing URLs. The probability of each alphabet and digit calculated from the 10 million legitimate URLs, shown in Equation (1), is used to calculate the URLCharProb of a URL by combining the probability of each alphabet and digit and dividing it by the URL length. The formula to calculate the URLCharProb of each URL is given below.

$$URLCharProb = \sum_{i=0}^{n} \frac{prob(URL[char_t])}{n} \tag{1}$$

where, n is count of characters from a-z and digits from 0-9.

TLDLegitimateProb: The top-level domain (TLD) is the last part of a domain name that indicates the purpose or origin of a URL. Phishing attackers often use TLDs that are uncommon or unrelated to the purpose of the website they are trying to spoof. Legitimate websites often use specific TLDs associated with their industry or location. We extracted all the TLDs from the top 10 million websites and counted the occurrence of each TLD. Further, we calculated the ratio of each TLD by dividing the total occurrence of that TLD by the total occurrence of all the TLDs. A higher TLDLegitimateProb of a URL may indicate a legitimate URL, and a lower TLDLegitimateProb value may help identify phishing URLs.

In summary, the phishing URL dataset construction technique involves extracting and analyzing the URL, HTML, and derived features to create a comprehensive dataset for detecting phishing attacks. These features provide valuable information about the potential malicious intent of the URL.

4. PhiUSIIL's phishing URL detection module

The PhiUSIIL phishing URL detection module is divided into two phases. Phase 1 is URL similarity index(USI) based phishing URL detection, and Phase 2 is incremental learning-based phishing URL detection.

Phase 1 computes the URL similarity index for a given URL to identify phishing URLs. The given URL is blocked if its USI is $>=80$ and <100; otherwise, it activates Phase 2. When Phase 2 is activated, it extracts key features from the URL and HTML code to generate feature vector for the given URL. The feature vector is fed into the incremental leering model in the next stage to get a classification score. Further, this score is used to classify the URL as legitimate or phishing. Each phase of the PhiUSIIL phishing URL detection module is explained in detail in the following subsections. The working of the PhiUSIIL phishing URL detection is shown in Fig. 3.

Algorithm 1 Calculation of URL-Title match score.

Require: Webpage title and URL
Ensure: URL-Title matching score
1: $tSet \Leftarrow title.split()$
▷ remove 'https', 'http', 'www', and TLD from URL to get root domain
2: $txtURL \Leftarrow RootDomain(URL)$
3: **call** URLTitleMatchScore(tSet, txtURL)
4: $score \Leftarrow URLTitleMatchScore(tSet, txtURL)$
5: **function** URLTitleMatchScore(Set,Txt)
6: $score \Leftarrow 0$
7: $baseScore \Leftarrow 100/(len(Txt))$
8: **for** element in Set **do**
9: **if** $Txt.find(element) >= 0$ **then**
10: $n \Leftarrow len(element)$
11: $score \Leftarrow score + baseScore * n$
12: $Txt \Leftarrow Txt.replace(element, "")$
13: **if** $score > 99.9$ **then**
14: $score \Leftarrow 100$
15: **end if**
16: **end if**
17: **end for**
18: **return** score
19: **end function**

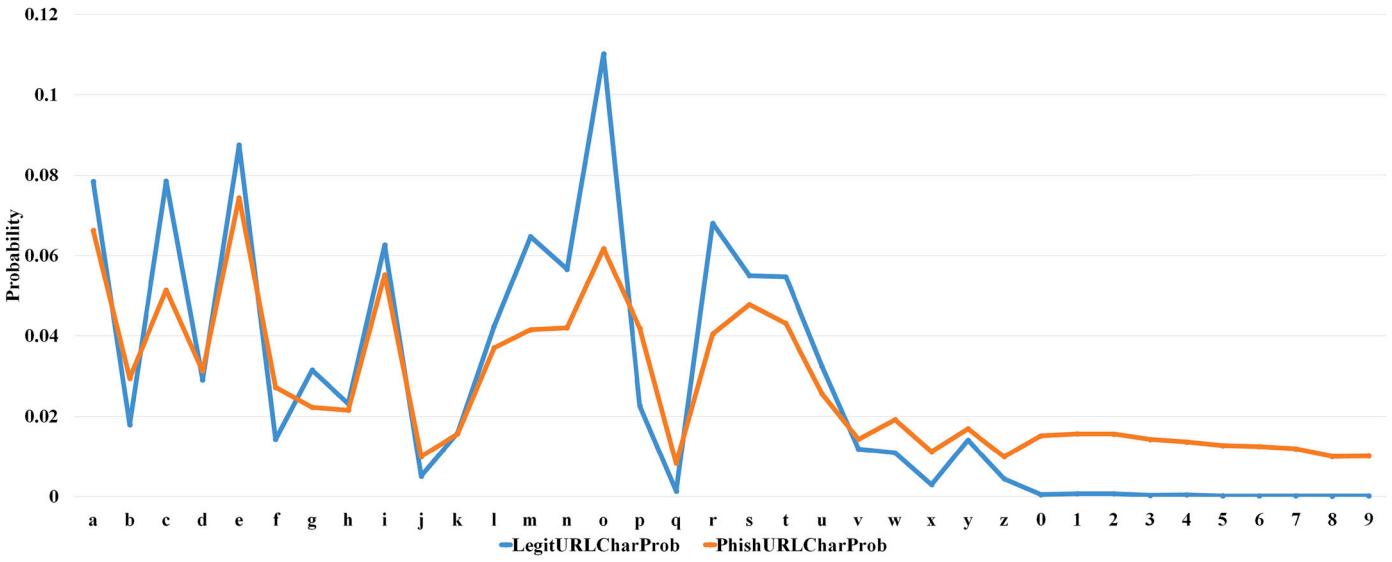

Fig. 2. Probability comparison of alphabets and digits in legitimate URLs versus phishing URLs.

4.1. Phase 1: URL similarity index based phishing URL detection

The URL similarity index is calculated between a source URL and a target URL. The source URL is the given URL for calculating USI. The target URL is a URL from the list of the top 10 million legitimate websites downloaded from Open PageRank Initiative. Our idea is to find how closely a URL from the list matches any of the URLs from the top 10 million list. A USI 100 shows a legitimate URL, and a USI close to 100 may indicate a phishing URL that looks very similar to a legitimate URL. Each character of the source and target URL is compared; for each matching character, one similarity value is calculated and assigned to it. For all unmatched characters, the i^{th} character from the longest URL is deleted, and the value of i is reduced by 1 to compare the same i^{th} position again in the next iteration.

The maximum USI is considered 100, and the minimum is 0. Initially, 100 is divided into two equal parts. Further, the first 50 is equally distributed to all the URL characters as the base similarity value. The remaining 50 is distributed based on the weighted similarity value calculated for each matching character. For each i^{th} matching character, the weighted similarity value is calculated and added to the base similarity value. Finally, the sum of similarity value of all the matching characters is calculated as USI. A URL with USI value higher than 80 is considered as phishing URL and blocked by PhiUSIIL. The machine learning mode analyzes the remaining URLs.

The formula to calculate USI between two URLs, Src and Tar, is given in Equation (2), and complete calculation steps are given in Algorithm 2.

$$URLSimilarityIndex(USI) = \sum_{i=0}^{n} \frac{50}{n} + \frac{100 * (n-i)}{n * (n+1)} \quad (2)$$

where, n is length of longest URL in Src and Tar

4.2. Phase 2: Incremental learning based phishing URL detection

If a URL is not classified by Phase 1, PhiUSIIL Phase 2 is activated. In Phase 2, we will start with the discussion of the prerequisite of the incremental learning-based phishing URL detection.

In the following subsections, we discuss the implemented feature selection techniques to get a reduced feature set. Next, we discuss the technique to select the most suitable incremental learning algorithm. Then we discuss the hyperparameter tuning to improve the model's performance. Further, the selected algorithms are employed to a diverse security profile empowered incremental learning model to classify the given URL. Each stage is discussed in the following subsections.

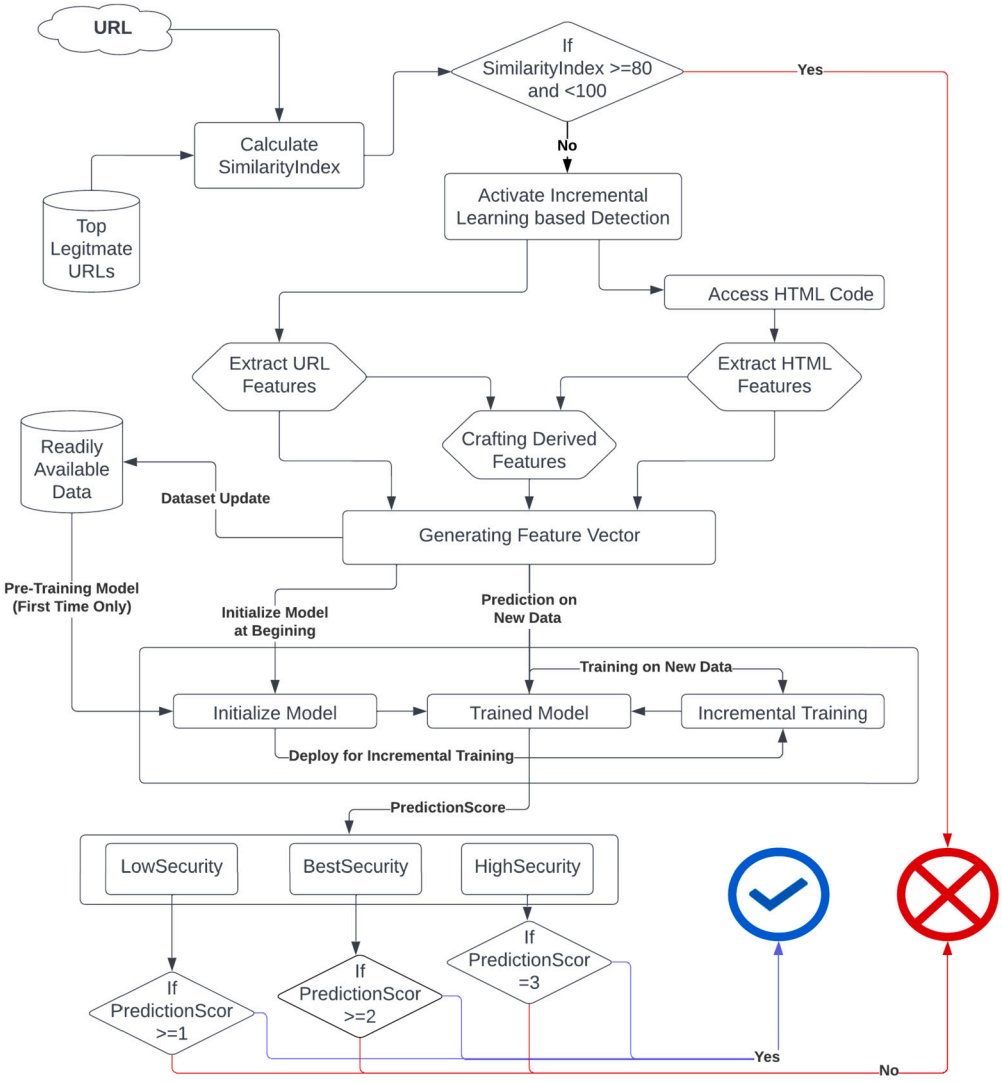

Fig. 3. Work flow of PhiUSIIL's phishing URL detection framework.

Algorithm 2 SimilaritiesIndex calculation between two URLs.

```
function GETURLSIMILARITYINDEX(Src, Tar)
    #Set X = shortest URL, Y = longest URL, and n = length of shortest URL from Src and Tar URLs
    X, Y, n = getMin(Src, Tar)
    LengthOfLongestURL(N) ⇐ max(len(Src), len(Tar))
    SimilarityIndex(SI) ⇐ 0
    BaseValue(a) ⇐ 50/N
    SumOfNaturalNumbers(nsum) ⇐ (N ∗ (N + 1))/2
    for i in range(0, n) do
        if X[i] == Y[i] then
            SI ⇐ SI + a + (50 ∗ (N − i))/nsum
        else
            #Remove i^th (unmatched) character from longest URL
            Y ⇐ Y[: i] + Y[i + 1 :]
            #Set X = shortest, Y = longest, and n = length of shortest URL
            X, Y, n ⇐ getMin(Src, Tar)
            i ⇐ i − 1
        end if
    end for
end function
function GETMIN(Src,Tar)
    return min(Src,Tar),max(Src,Tar), min(len(Src),len(Tar))
end function
```

4.2.1. Prerequisite

The prerequisite section discusses the implemented feature selection to get a reduced feature set, the technique to select the most suitable incremental learning algorithm, and the hyperparameter tuning to improve the model's performance.

Feature selection: Feature selection is an essential and crucial step in machine learning that involves identifying and choosing the most

Table 2
Experimental result of selected incremental learning algorithm on Phishing Attack dataset.

Algorithms	Accuracy	Precision	Recall	F1Score	Training Time	Count
BernoulliNB	**0.98654**	**0.97875**	**0.99812**	**0.98834**	158	**4**
GaussianNB	0.57154	0.57154	**1**	0.72736	140	1
MultinomialNB	0.69085	0.65370	0.97626	0.78307	212	0
PassiveAggressive	**0.99418**	**0.99424**	0.99557	**0.99491**	**108**	**4**
Perceptron	0.94364	0.91122	**0.99869**	0.95295	**135**	2
SGDClassifier	**0.94658**	**0.91604**	0.99800	**0.95527**	**108**	**4**

relevant and informative features from a larger set of available features (Verma and Chandra, 2023; Srijayanthi and Sethukarasi, 2023). While relevant and informative features can enhance the performance of machine learning models, irrelevant or redundant features can introduce noise and complexity, leading to poor performance of the machine learning model.

Although all the features are thoughtfully extracted or derived from the URL or HTML code, it is important to identify their relevance for machine learning. Determining and quantifying the importance of individual features helps select and discard features in machine learning. Having a high number of features can increase the complexity of the model and potentially make it harder to interpret. Feature importance helps identify less influential features and can be safely removed without significantly affecting model performance. Feature importance scores can help identify which features impact the model's predictions most and are relevant for machine learning. We followed the following steps to identify relevant features.

- We implemented RandomForestClassifier, Decision-TreeClassifier, LGBMClassifier, and XGBClassifier to find the importance of each feature. All these algorithms have an attribute called feature importances that assigns a relative importance score to each feature based on their contribution to the final prediction. Each machine learning algorithm has different techniques to calculate feature importance. Creating an ensemble for feature importance using multiple algorithms offers robustness by capturing diverse insights. We selected all the features having feature importance values higher than zero. We selected all the features having feature importance values higher than zero.
- The mutual information value is measured for all the features identified from step one using Mutual_info_cla-ssif. Mutual_info_classif measures the independent variable's dependency on the dependent variable. The top 85% of features are selected based on their higher Mutual_info_classif value.

Identify most suitable incremental learning algorithms: Different machine learning algorithms are designed to work with different types of data and have different problem-solving approaches. If unsuitable algorithms are selected for a particular dataset, it may not achieve accurate and reliable results. Unsuitable algorithms may not even converge, making it impossible to obtain useful insights from the data. Therefore, it is important to evaluate and select the most appropriate algorithms for a given dataset through experiments. The proposed model is based on an ensemble of three incremental learning algorithms. While it is not necessary to limit the number of incremental learning algorithms used in an ensemble, keeping the number of algorithms to three can simplify the implementation and management of the ensemble model. Training multiple incremental learning algorithms can be computationally expensive, especially when dealing with large datasets. Limiting the number of algorithms used in the ensemble can help reduce the computational overhead and speed up the training process.

Our experiment involved trying out different algorithms that support incremental learning, such as MultinomialNB (Multinomial Naive Bayes), BernoulliNB (Bernoulli Naive Bayes), PassiveAggressiveClassifier, Perceptron, and SGDClassifier (Stochastic Gradient Descent), comparing their classification performance. Next, we identified the top three incremental learning algorithms from Accuracy, Precision, Recall, F1Score, and Training Time. Finally, we counted the occurrence of each algorithm in the top three lists of Accuracy, Precision, Recall, F1Score, and Training Time. Based on their count, we select the best three algorithms, namely BernoulliNB, PassiveAggressiveClassifier, and SGDClassifier, to achieve the most accurate and reliable results by effectively learning the underlying patterns and relationships from the dataset. The experimental result is shown in Table 2. The highlighted value indicates the algorithm's selection in the top three lists.

Hyperparameter tuning: Hyperparameter tuning is a crucial process of finding the best set of hyperparameters for a machine learning model that can lead to significant improvements in model performance, faster convergence, reduced overfitting, and more efficient use of computational resources (Sáez-de-Cámara et al., 2023). We implemented GridSearchCV based hyperparameter tuning technique to optimize the performance of the selected incremental learning algorithms. GridSearchCV works by exhaustively searching through a specified grid of hyperparameter values and evaluating the model's performance using cross-validation. The hyperparameters for all three algorithms and their value are given below.

- **hpBNB:** 'alpha': 0.01, 'binarize': 0.0, 'class_prior': None, 'fit_prior': False, 'force_alpha': True
- **hpPA:** 'C': 5, 'fit_intercept': False, 'n_iter_no_change': 50, 'n_jobs': 10, 'shuffle': True
- **hpSGD:** 'alpha': 0.01, 'eta0': 100, 'n_iter_no_change': 50, 'n_jobs': 1

4.2.2. *Diverse security profile empowered incremental learning model*

This research article proposes a phishing URL classification model that utilizes three incremental learning algorithms: BernoulliNB, PassiveAggressive, and SGDClassifier. The ensemble of multiple machine learning algorithms has shown better classification in various studies (Taha, 2021; Prasad et al., 2023). These algorithms are identified from previous section based on their performance. The PhiUSIIL introduces three distinct security profiles, namely LowSecurity, BestSecurity, and HighSecurity. Different users or organizations may have diverse security needs and risk tolerances. By offering diverse security profiles, the model can cater to these diverse requirements. By providing different security profiles, the model can adapt and adjust its classification criteria based on the prevailing threat landscape. This flexibility allows the model to maintain optimal performance in dynamic environments without the need for frequent retraining or model updates. All three profiles are discussed below.

LowSecurity profile: In this profile, if any one of the incremental learning algorithms correctly identifies a URL as legitimate, it is classified as such. Thus, the LowSecurity profile may exhibit higher false positives but ensures minimal disruption to legitimate URLs. It avoids overblocking legitimate URLs. The LowSecurity profile, with a more lenient classification threshold, helps minimize the occurrence of false positives, ensuring that legitimate URLs are not unnecessarily blocked. False positives occur when legitimate URLs are incorrectly classified as phishing URLs. This can lead to disruptions in normal user activities and frustration.

Algorithm 3 Incremental learning-based phishing URL detection algorithm.

Require: Input for model
Models
df is PhishingURL dataset
Reduced feature set
LABEL denotes class
hpBNB, hpPA, denotes hpSGD are hyperparameters for respective algorithms

$df X \Leftarrow pd.DataFrame(df[RFS]).copy()$
$df Y \Leftarrow pd.DataFrame(df[LABEL]).copy()$
$Models.append((\text{“}BernoulliNB\text{”}, BernoulliNB(hpBNB)))$
$Models.append((\text{“}PassiveAggressive\text{”}, PassiveAggressiveClassifier(hpPA)))$
$Models.append((\text{“}SGDClassifier\text{”}, SGDClassifier(hpSGD)))$
$Handler = pd.DataFrame()$ ▷ For calculating model performance
$X \Leftarrow df X.iloc[[0]].to\_numpy(), Y \Leftarrow df Y.iloc[[0]], Y \Leftarrow np.array(Y[LABEL])$
for name, model in Models **do**
    $model.partial\_fit(X, Y, classes = [0, 1])$ ▷ Initialize algorithms, to predict first on unseen data
end for
for i in range(1,n) **do**
    $predict \Leftarrow 0, yP \Leftarrow 0, nRow \Leftarrow []$
    $nRow['FILENAME'] \Leftarrow df['FILENAME'][i], nRow['isCorrect'] \Leftarrow 0$
    $nRow['LowSecurity'] \Leftarrow 0, nRow['BestSecurity'] \Leftarrow 0, nRow['HighSecurity'] \Leftarrow 0$
    $X \Leftarrow df X.iloc[[i]].to\_numpy(), Y \Leftarrow df Y.iloc[[i]], Y \Leftarrow np.array(Y[LABEL])$
    for name, model in Models **do**
        $pr \Leftarrow model.predict(X)$ ▷ Predicting first without training on unseen data
        $predict \Leftarrow predict + pr[0]$
        $model.partial_fit(X, Y, classes = [0, 1])$ ▷ Post prediction, model training on unseen data
    end for
    if $predict == 1$ **then**
        $nRow['LowSecurity'] \Leftarrow 1$
    else if $predict == 1$ **then**
        $yP \Leftarrow 1$
        $nRow['LowSecurity'] \Leftarrow 1, nRow['BestSecurity'] \Leftarrow 1$
    else
        $yP \Leftarrow 1$
        $nRow['LowSecurity'] \Leftarrow 1, nRow['BestSecurity'] \Leftarrow 1 nRow['HighSecurity'] \Leftarrow 1$
    end if
    if $yP == Y[0]$ **then**
        $accurate = accurate + 1$
        $newRow['isCorrect'] = 1$
    end if
    $AvgLRList.append(accurate/i)$ ▷ For plotting learning rate graph
    $Handler = Handler.append(newRow, ignore\_index = True)$
end for

BestSecurity profile: In this profile, if any two incremental learning algorithms correctly identify a URL as legitimate, it is classified as legitimate. This profile aims to strike a middle ground that satisfies the security requirements of users or organizations without overly restricting legitimate URLs or impeding normal online activities, providing improved security while maintaining overall acceptable performance.

HighSecurity profile: This profile requires all three incremental learning algorithms to correctly identify a URL as legitimate to classify it as such. By incorporating the HighSecurity profile, which requires consensus among all three incremental learning algorithms, the profile achieves a higher level of security. This profile is designed to be more conservative while classifying URLs, reducing the risk of false negatives (phishing URLs misclassified as legitimate). It provides an extra layer of protection for users or organizations that prioritize stringent security measures.

The proposed model is an ensemble of incremental learning models, namely BernoulliNB, PassiveAggressive, and SGDClassifier. The incremental training approach we have outlined has a unique and thoughtful design to ensure both model training and effective prediction while minimizing the risk of exposing test data to the model. Here is the step-wise explanation of the incremental training approach we have implemented:

- The very first record of the dataset is used to train the mode that also initializes the model. This is important as an uninitialized model cannot be used for prediction.
- A two-step process is involved for each subsequent record in the dataset (from the second record onward).
  - In the first step, the model performs a prediction on the given record, generating an initial output.
  - In the second step, the same record is used for training the model, enhancing its predictive capabilities.
- This approach ensures that each record contributes to both the prediction and the continuous improvement of the model.
- By predicting before training with the same record, we prevent the risk of data exposure that can lead to overly optimistic model performance.

If the prediction of any algorithm is 1, it means that the algorithm has predicted the URL as legitimate. Prediction value 0 shows a phishing URL. The prediction total from all the algorithms is stored in the predict variable. If the value of predict is 1, the LowSecurity profile algorithms predicts it as legitimate. If the value of predict is 2, both LowSecurity and BestSecurity profiles classify it as legitimate. If the value of predict is 3, all three profiles classify it as legitimate. Post prediction, all three algorithms are partially trained with the new data. The algorithm of the proposed approach is given in Algorithm 3. The result calculation approach for one of the profiles (BestSecurity) is given in Algorithm 4. A similar approach is implemented to calculate the result of other profiles.

4.2.3. Model performance enhancement using pre-training approach

The classical training approach assumes that the entire training dataset is available upfront, and the model is trained on this fixed dataset without any incremental training approach. The traditional way of training models involves training the model once using the entire training data and is commonly used in various machine learning algorithms. Pre-training an incremental model using a classical training

Algorithm 4 Pseudo code for evaluating models performance (best security mode).

```
RES ⇐ pd.DataFrame().copy()
RES['M1Vote'] ⇐ Handler['BestSecurity']
records ⇐ len(Handler[LABEL])
CorretPred ⇐ np.where(Handler[LABEL] == RES['M1Vote'], 1, 0).sum()
TP ⇐ int((np.where(Handler[LABEL] == 1, np.where(Handler['BestSecurity'] == 1, 1, 0), 0)).sum())
TN ⇐ int((np.where(Handler[LABEL] == 0, np.where(Handler['BestSecurity'] == 0, 1, 0), 0)).sum())
FP ⇐ int((np.where(Handler[LABEL] == 0, np.where(Handler['BestSecurity'] == 1, 1, 0), 0)).sum())
FN ⇐ int((np.where(Handler[LABEL] == 1, np.where(Handler['BestSecurity'] == 0, 1, 0), 0)).sum())
MCC ⇐ int((TP * TN) − (FP * FN))/(int((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)) ** 0.5)
Accuracy ⇐ CorretPred/records
Precision ⇐ TP/(TP + FP)
Sensitivity ⇐ TP/(TP + FN)
F1Score ⇐ 2 * ((Precision * Sensitivity)/(Precision + Sensitivity))
```

approach enhances robustness and generalization capabilities. In addition, this approach can leverage large amounts of readily available data that can lead the model to faster convergence and better performance on new unseen data. In our experiment, the proposed performance enhancement approach involves training the model in two steps: pre-training and incremental training.

- In the pre-training phase, the model is trained on a portion of the records from a dataset. This initial training is typically done to provide the model with a general understanding of the underlying patterns and features present in the dataset. In addition, pre-training helps the model learn useful representations that can be fine-tuned later.
- Once the pre-training phase is complete, the remaining portion of records from the dataset is used for incremental training. This means that the model is further trained on this additional data, allowing it to gradually improve and adapt to the specific patterns and characteristics of the dataset.

The model is initially pre-trained on a small portion (10%) of the dataset, and the remaining records are used for incremental training. This process is repeated with increasing proportions for pre-training (20%, 30%, 40%, and so on) while the incremental training proportion decreases correspondingly.

5. Result & discussions

In the following subsection, we list the tools and system configurations used to perform the experiments and then discuss the performance metrics used for model evaluation. In the next subsection, we validate the newly created PhiUSIIL phishing URL dataset. Then, we evaluated the URL similarity index-based phishing URL detection phase. Further, we evaluated the incremental learning-based second phase of the phishing URL detection model. Finally, we compared the proposed model with state-of-the-art models.

5.1. System configuration and tools

All experiments are conducted on a 64-bit Windows 10 Pro, version 22H2 operating system equipped with a 2.70 GHz core i7-7500U processor, 16 GB RAM, and a 2 GB graphics card. The tools and programming languages used are Python 3.8.8 using Jupyter Notebook and Java version 17.

5.2. Performance metrics

Performance metrics are used to evaluate the effectiveness and accuracy of a machine learning model's prediction ability. These metrics provide a quantitative measure of how well a machine learning model performs, allowing comparisons between different models. We have extensively used these metrics to identify the most suitable incremental learning algorithms for the proposed ensemble model, tuning hyperparameters, getting feedback on the strengths and weaknesses of the proposed machine learning model, and comparing the proposed model performance with different state-of-the-art machine learning models. The following metrics help assess how well the proposed model is performing.

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{3}$$

$$Precision = \frac{T_p}{T_p + F_p} \tag{4}$$

$$Recall = \frac{T_p}{T_p + F_n} \tag{5}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

$$MCC = \frac{(T_p * T_n) - (F_p * F_n)}{\sqrt{(T_p + F_p) * (T_p + F_n) * (T_n + F_p) * (T_n + F_n)}} \tag{7}$$

where,

T_p is legitimate URL correctly classified as legitimate.
T_n is phishing URL correctly classified as phishing.
F_p is legitimate URL incorrectly classified as phishing.
F_n is phishing URL incorrectly classified as legitimate.

5.3. Dataset validation

For any newly constructed dataset, validation is a crucial step to ensure that the dataset used to train and evaluate a machine learning model is accurate and reliable, leading to more trustworthy and effective machine learning models. Our dataset validation involves assessing its quality and suitability for training and evaluating the machine-learning models. The following steps are followed to assess the quality of the dataset.

URL verification: All the URLs in the dataset are collected from valid sources and included in the dataset for verification.

Null value: There is no null value in the dataset.

Missing value: There is no missing value in the dataset.

Duplicate records: All the records in the dataset are unique.

Zero variance feature: There is no feature in the dataset with identical data values.

Infinite values: There is no positive or negative infinite value in the dataset.

Class imbalance: The dataset has 57% legitimate URLs and 43% phishing URLs that do not indicate a disproportionate distribution of class labels.

It is important to run the dataset through diverse machine learning algorithms and establish initial performance metrics to ensure the reliability of a dataset. It helps validate the dataset's suitability for training and provides a benchmark for future model improvements. If the dataset contains errors or biases, the model will also learn and propagate those errors, leading to unreliable and inaccurate predictions. Our experiment uses base machine learning algorithms, incremental learning algorithms, boosting algorithms, bagging algorithms, voting

Table 3
Performance of various machine learning algorithms on Phishing URL dataset.

Model	Accuracy	Precision	Recall	F1Score	MCC	Training Time
BernoulliNB	0.98644	0.97879	0.9979	0.98826	0.97247	1
PassiveAggressive	0.98021	0.97072	0.9954	0.98291	0.95983	1
SGDClassifier	0.94294	0.91113	0.9975	0.95236	0.88722	2
MultinomialNB	0.68961	0.65277	0.97669	0.78253	0.39829	1
Perceptron	0.98041	0.97039	0.99612	0.98309	0.96027	1
AdaBoost	0.99981	0.99978	0.99989	0.99983	0.99961	13
LightGBM	0.9999	0.99991	0.99993	0.99992	0.99981	1
XGBClassifier	0.99993	0.99993	0.99994	0.99994	0.99985	12
CatBoost	0.99987	0.99981	0.99996	0.99989	0.99974	35
GradientBoosting	0.99978	0.99974	0.99987	0.99981	0.99955	50
Kneighbors	0.99378	0.9933	0.99583	0.99456	0.98729	43
RandomForest	0.99982	0.9998	0.99989	0.99984	0.99963	21
DecisionTree	0.99866	0.99885	0.99881	0.99883	0.99727	2
LogisticRegression	0.99654	0.99712	0.99683	0.99698	0.99294	2
Bagging	0.99923	0.99967	0.99898	0.99932	0.99842	16
Vote_Hard	0.99874	0.99852	0.99928	0.9989	0.99742	83
Vote_Soft	0.99817	0.99802	0.99878	0.9984	0.99625	75
StackingClassifier	0.99979	0.99978	0.99985	0.99981	0.99957	870

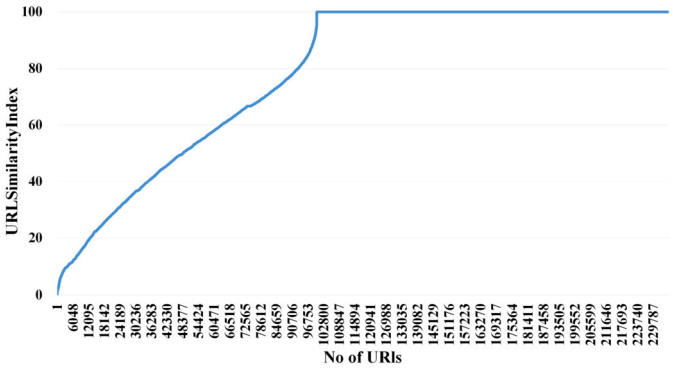

Fig. 4. URL similarity index graph.

algorithms, and stacking algorithm to evaluate the dataset. The evaluation result in Table 3 shows that most algorithms generalize well and achieve impressive results.

5.4. Evaluation of URL similarity index based phishing URL detection

Phishing URLs generally mimic legitimate URLs, deceiving users into thinking they are accessing a legitimate website. Cybercriminals create phishing URLs that look similar to legitimate URLs by replacing any similar-looking character. In addition, they swap characters, making phishing URLs appear identical to the legitimate URL. The proposed URL similarity index (USI) technique identifies such tricks by calculating USI for a URL. We calculate the USI for each URL and analyze them based on the following conditions.

- If the USI value is 100, it is identical to a legitimate URL. Hence, considered a legitimate URL.
- If USI value is $>=80$ and <100, it shows that the URL is highly identical to a legitimate URL. Hence, such URLs are identified as Phishing URLs.
- Any USI value less than 80 is considered doubtful and passed to the machine learning model for further analysis.

During our experiment, we calculated USI value for all the URLs. Fig. 4 depicts the graph plotted based on the USI value. Based on the above conditions, we were able to detect 7341 phishing URLs correctly.

The experimental result in Table 4 shows how a phishing URL looks visually similar to a legitimate URL. Cybercriminals replace a character from a legitimate URL with a visually resembled Latin character or misspell. These small modifications by a cybercriminal can trick users into believing they are visiting a legitimate website. The URL similarity index approach identifies any such misspellings, substituted characters, or slight variations in the URL.

5.5. Evaluation of diverse security profile empowered incremental learning model

In this subsection, we experimented with the proposed incremental learning model on the PhiUSIIL dataset. A comprehensive set of evaluation metrics is employed to evaluate the performance of the proposed model, including accuracy, precision, sensitivity, F1-score, and MCC, which will provide insights into the strengths and limitations of different profiles and their ability to detect phishing URLs accurately. Initially, all three algorithms, namely BernoulliNB, PassiveAggressiveClassifier, and SGDClassifier, are initialized with the first records of the PhiUSIIL dataset to prevent the model from generating a 'not fitted yet' exception. The experimental result is presented in Table 5.

Table 6 shows the overall performance of the model in terms of True Positive (TP), True Negative (TN), False Positive (FP), and False Positive (FP) organized in a confusion matrix. It represents the counts of different types of predictions made by the model. The LowSecurity profile gets the lowest Type II error (lowest FN), which shows that only one legitimate URL is classified as phishing. It ensures that legitimate URLs are not unnecessarily blocked. While LowSecurity profile archives highest accuracy of correctly classifying legitimate URL (99.9993%), it achieved lowest accuracy of correctly classifying phishing URLs (72.5732%). The HighSecurity profile gets the lowest Type I error (lowest FP), which shows that only 27 phishing URLs are classified as legitimate. It ensures an extra layer of protection that prioritizes stringent security measures. While HighSecurity profile archives highest accuracy of correctly classifying phishing URLs (99.9733%), it achieved lowest accuracy of correctly classifying legitimate URL (95.4898%). Table 6 shows that the BestSecurity profile provides improved security while maintaining overall acceptable performance.

Further, to get valuable insights into the behavior and performance of the proposed incremental learning model, we calculated model learning rate based on the model's predictions of each iteration from model training. It helps to assess the stability of model's predictions over time. The learning rate value for n_{th} record is calculated based on formula given in equation (8)

Table 4
Experimental result showing URL similarity based phishing URL detection.

Legitimate URL	Test URL	URL Similarity index	Change in URL	Result
example.com	example.com	100	Same URL	Legitimate
example.com	examplẹ.com	91.18457	Latin e with dot below symbol	Phishing
example.com	examṗle.com	90.63361	Latin p with dot above symbol	Phishing
example.com	exȧmple.com	90.08264	Latin a with dot below symbol	Phishing
example.com	ėxample.com	89.80716	Latin a with dot below symbol	Phishing
example.com	exampl.com	83.47107	Missing e	Phishing
example.com	exapmle.com	73.27824	Character m and p are swapped	Doubtful
example.com	eaxmple.com	72.17631	Character a and x are swapped	Doubtful

Table 5
Security profile wise performance of the proposed model.

Security Profile	Accuracy	Precision	Sensitivity	F1Score	MCC	Analysis
LowSecurity	0.88258	0.82966	0.99999	0.90690	0.77595	Highest sensitivity
BestSecurity	0.99239	0.98870	0.99810	0.99338	0.98449	Good overall performance
HighSecurity	0.97409	0.99979	0.95490	0.97683	0.94877	Highest precision

Table 6
Confusion matrix showing overall performance of the model.

Security Profile	TP	TN	FP	FN
LowSecurity	134849	73259	27686	1
BestSecurity	134594	99406	1539	256
HighSecurity	128768	100918	27	6082

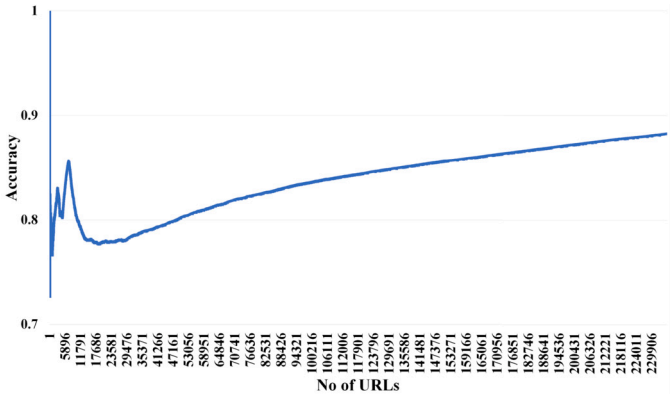

Fig. 5. LowSecurity profile learning rate.

$$Learning\,Rate = \sum_{i=0}^{n} \frac{Correct\,Prediction_{i}}{n} \tag{8}$$

A graph is plotted based on the learning rate values in Fig. 5 for LowSecurity, Fig. 6 for BestSecurity, and Fig. 7 for HighSecurity profile. While comparing the learning rates of different security profiles, we can see there are fluctuations in learning rate for low and high security profiles, but BestSecurity profile remains stable after rapid learning and adaptation from initial training. If the learning rate remains relatively constant, it indicates that the model is converging and becoming more stable. By observing the BestSecurity learning rate graph, we find there are no sudden spikes or drops in the learning rate.

5.6. Evaluating the pre-training approach of model performance enhancement

In the next experiment, we split the dataset into two parts. The first part of the split is used to train the model with a classical training approach where entire training data is used to train the model at once. The second part of the dataset is used to test and train the model with the incremental training approach discussed in Section 4.2.2. This approach analyzes the model performance when the model is pre-trained with a

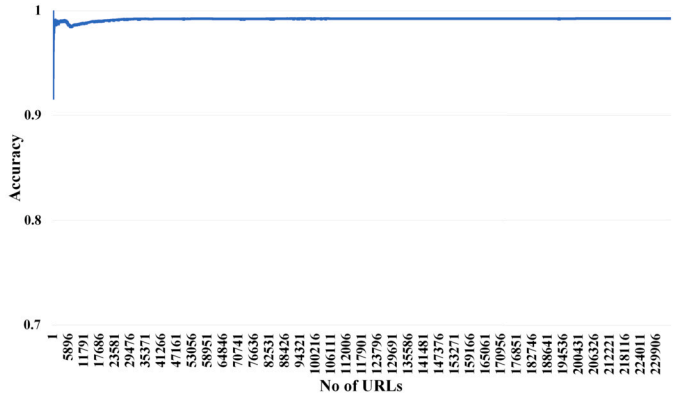

Fig. 6. BestSecurity profile learning rate.

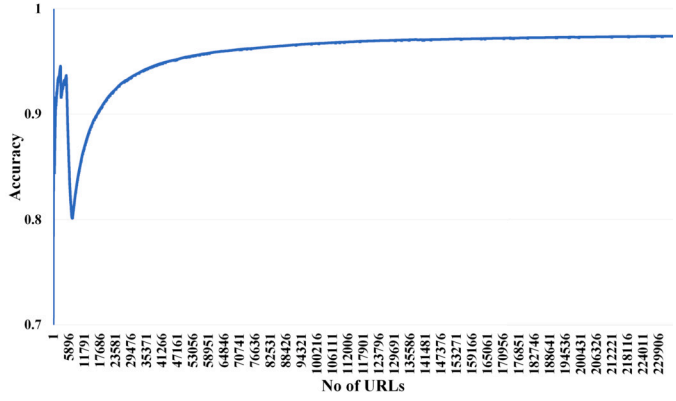

Fig. 7. HighSecurity profile learning rate.

portion of the dataset before going live with incremental training. The experimental result is shown in Table 7. Based on the experimental result, a graph is plotted in Fig. 8 showing the overall performance of pre-trained models.

From Fig. 8, it can be concluded that the incremental learning model exhibits improved performance as the amount of data used for pre-training increases. The model initially undergoes pre-training using 10% of the dataset, with the remaining 90% used for incremental training. This process is then repeated with increasing percentages for pre-training (20%, 30%, 40%, and so on). The experimental results indicate that as the record size used for pre-training increases, the model's performance also improves.

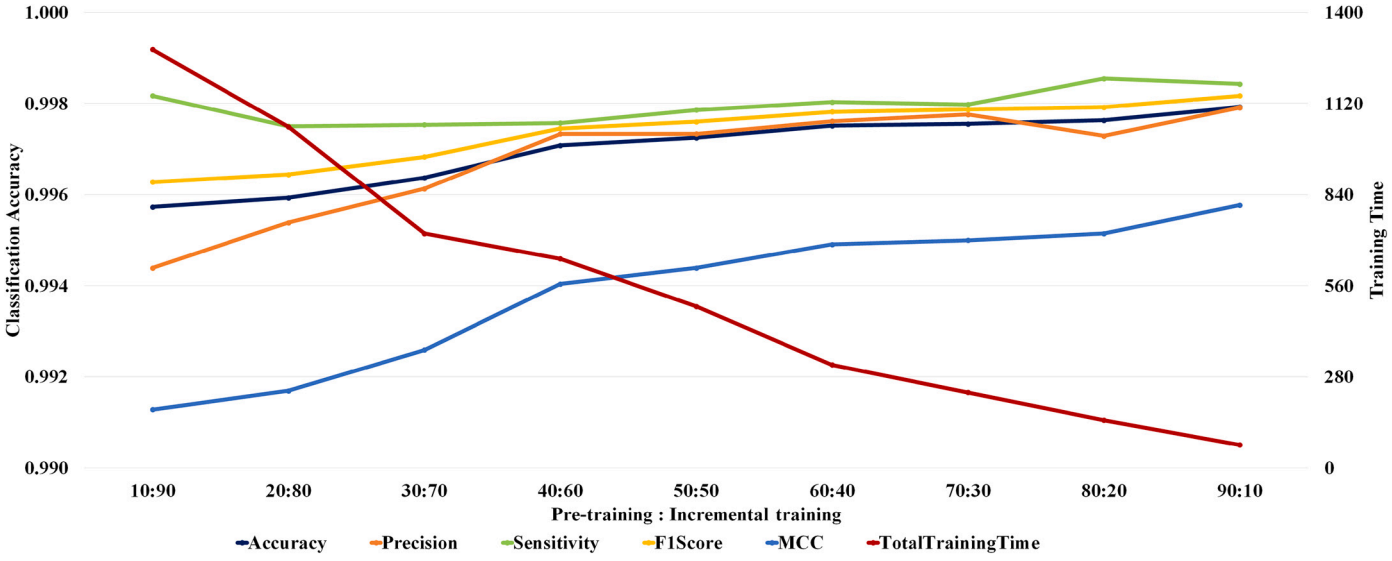

Fig. 8. Pre-trained model classification performance graph.

Table 7
Experimental result when model pre-trained on portion of the dataset.

Split ratio	Accuracy	Precision	Sensitivity	F1Score	MCC	Training Time
10:90	0.99573	0.99439	0.99817	0.99628	0.99128	1286
20:80	0.99593	0.99539	0.9975	0.99644	0.99169	1048
30:70	0.99637	0.99613	0.99753	0.99683	0.99259	721
40:60	0.99708	0.99733	0.99757	0.99745	0.99404	643
50:50	0.99725	0.99733	0.99786	0.9976	0.99439	496
60:40	0.99751	0.99761	0.99803	0.99782	0.99491	317
70:30	0.99755	0.99776	0.99798	0.99787	0.995	232
80:20	0.99763	0.99729	0.99855	0.99792	0.99515	147
90:10	0.99792	0.99791	0.99843	0.99817	0.99577	71

Table 8
Comparison of PhiUSIIL (BestSecurity profile) with state-of-the-art.

Ref.	Legitimate URLs	Phishing URLs	Total URLs	Accuracy
Gupta et al. (2021)	10000	9964	11964	0.9957
Pandey and Mishra (2023)	2700	3500	6200	0.9913
Ahammad et al. (2022)	1500	1500	3000	0.8950
Jain et al. (2022)	2000	2000	4000	0.9385
Alani and Tawfik (2022)	58000	30646	88646	0.9750
Ding et al. (2019)	5883	2776	8659	0.9890
Sharma and Singh (2022)	24870	25130	50000	0.9801
Nagunwa et al. (2022)	7530	4271	11801	0.9842
Sameen et al. (2020)	50000	50000	100000	0.9800
Rao et al. (2022)	5438	5076	10514	0.9934
PhiUSIIL	**134850**	**100945**	**235795**	Pre-trained model: **0.9979** Un-Trained model: 0.9924

We have presented the comparative results, including the accuracy and details of the dataset used, in Table 8. By comparing the accuracy of PhiUSIIL with other studies, it can be concluded that PhiUSIIL outperforms the existing state-of-the-art models in detecting phishing URLs. Furthermore, the higher accuracy of PhiUSIIL suggests that it is more effective in distinguishing between legitimate and phishing URLs, thereby reducing the risk of falling victim to phishing attacks.

5.7. Discussions

This work introduces PhiUSIIL, an advanced phishing URL detection framework with the objective of constructing a diverse dataset, implementing a URL similarity index and incremental learning approach to detect phishing URLs.

The objective of constructing a diverse dataset with 134850 legitimate and 100945 phishing URLs is pivotal for robust machine learning in phishing URL detection. This constructed dataset enhances model generalization, reduces bias, and improves detection accuracy. Mimicking real-world scenarios equips the model to handle evolving attack strategies and ensures its relevance over time, reinforcing its efficacy and contributing to cybersecurity advancements. Table 7 shows a direct correlation between the dataset size and the model's performance in phishing URL detection. As we gradually increased the total number of records for pre-training (from 10% to 20%, 30%, and beyond), the model's accuracy consistently improved.

This research's significance lies in the introduced URL similarity index technique. Visual similarity-based attacks exploit minute URL differences, such as zero-width characters, homograph, punycode, homophone, bit squatting, and combosquatting. This technique's role is vital:

it detects cybercriminal tactics where visually resembling characters deceive users into accessing malicious sites. Experimentally demonstrating the similarity between phishing and legitimate URLs underscores the potential risks. Table 4 clearly shows the effectiveness of the URL similarity index technique by identifying minute changes in the URL to detect phishing URLs. The URL similarity index approach emerges as a powerful tool to counteract these subtle yet potent attacks, enhancing cybersecurity by unearthing disguised threats and safeguarding users against deception.

The proposed diverse security profile integrated with incremental learning presents a pivotal advancement in our research. This framework addresses scalability, adaptability, and model staleness prevention challenges in phishing URL detection. Through empirical analysis, we observe that the BestSecurity profile demonstrates consistent and stable learning rates. Leveraging readily available data for pre-training further improves the model's resilience and generalization capabilities. Our experiment in Table 7 shows that the BestSecurity profile achieves enhanced security without compromising overall performance, showcasing its potential to ensure robust and reliable phishing detection while maintaining efficiency. The learning rate of BestProfile in Fig. 5 shows the model's stability after initial learning.

Additionally, this research recognizes the limitations of relying on URL's third-party features and underscores the significance of incorporating contextual information beyond URL features. Additionally, diversifying machine learning algorithms highlights a comprehensive defense approach.

Collectively, the PhiUSIIL framework exhibits robust scalability and adaptability in phishing URL detection. The framework continuously updates itself with new data through incremental learning, enabling it to adapt to emerging threats without requiring complete retraining. The framework's consistent learning rates ensure steady performance even with increased data volume, contributing to its scalability. Adaptability is further highlighted by introducing a visual similarity index technique, which detects subtle differences in URLs exploited by visually similar attacks. Moreover, pre-training with available data enhances the model's resilience and capacity to absorb new information, contributing to its adaptability.

In summary, this research introduces a substantial data-set comprising 134850 legitimate and 100945 phishing URLs, surpassing existing benchmarks. The experimentation achieved a remarkable 99.79% accuracy, outperforming state-of-the-art techniques. This underscores the dataset's efficacy and the potential of the proposed approach for advanced phishing URL detection.

6. Limitations and future ahead

Despite its potential, the PhiUSIIL framework faces limitations. Continuous training of PhiUSIIL may introduce the risk of over-training, potentially hampering efficiency and effectiveness. The extended training periods could slow down PhiUSIIL's responsiveness. Moreover, as the model continuously adapts to new data, it risks becoming overly specialized and less capable of generalization. Exploring strategies to mitigate this challenge remains a promising avenue for further research and advancement of PhiUSIIL. Further research may include techniques such as regularization, discarding redundant training data, and optimizing continuous training to mitigate this limitation.

Furthermore, the research's limitation lies in its current extent of detecting URLs that distribute executable files. Further advancement is necessary to enhance detection capability to analyze the URLs that auto download files such as .exe, .jar, .bat, .dll, .vbs, .js, .lnk, .sys, .pdf, etc. This improvement is crucial to effectively counter evolving cyber threats involving malicious URL downloads, ensuring robust cybersecurity measures.

While PhiUSIIL presents a promising approach to phishing URL detection, its vulnerability to potential attacks is another notable limitation. Integrating blockchain technology warrants exploration to enhance its security. Block-chain could fortify PhiUSIIL's defenses against adversarial threats, contributing to a more robust and reliable phishing detection framework.

7. Conclusion

The article introduces PhiUSIIL, a novel phishing URL detection framework designed to combat the ongoing threat of phishing attacks. Phishing attacks, which rely on deceptive URLs to trick users into revealing sensitive information or performing harmful actions, pose significant risks such as financial loss, identity theft, and data compromise. The PhiUSIIL framework addresses these challenges by leveraging a URL similarity index and incremental learning. The URL similarity index effectively identifies visually similar attacks, including zero-width characters, homograph, punycode, homophone, bit squatting, and combosquatting attacks. This capability enhances the framework's ability to detect sophisticated phishing techniques that rely on visual deception. Furthermore, PhiUSIIL incorporates incremental learning to adapt to the evolving nature of phishing attacks. By continuously updating its knowledge base with newly encountered phishing URLs, the framework remains effective and up-to-date against emerging threats. The framework utilizes various techniques, such as extracting URL features, downloading webpages to extract HTML features, and deriving new features from existing information to construct a comprehensive phishing URL dataset. The article provides extensive experimentation and evaluation using the PhiUSIIL dataset, which includes both legitimate and phishing URLs. In conclusion, the PhiUSIIL phishing URL detection framework presents a promising approach to tackle the ongoing threat of phishing attacks.

Funding

The authors did not receive support from any organization for the submitted work.

CRediT authorship contribution statement

Arvind Prasad: Design & development of the framework, data collection, experiments and result analysis.

Shalini Chandra: Revised it critically for important intellectual content.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared the link to download the dataset and code at the 'Attach File' step and in the manuscript.

References

Ahammad, S.H., Kale, S.D., Upadhye, G.D., Pande, S.D., Babu, E.V., Dhumane, A.V., Bahadur, M.D.K.J., 2022. Phishing URL detection using machine learning methods. Adv. Eng. Softw. 173, 103288. https://doi.org/10.1016/j.advengsoft.2022.103288.

Alani, M.M., Tawfik, H., 2022. PhishNot: a cloud-based machine-learning approach to phishing URL detection. Comput. Netw. 218, 109407. https://doi.org/10.1016/j.comnet.2022.109407.

Anon, 2023a. Retrieved May, 12, 2023. https://isc.sans.edu/diary/The+zip+gTLD+Risks+and+Opportunities/29838.

Anon, 2023b. Retrieved May 18, 2023. https://www.malwareworld.com/textlists/suspiciousDomains.txts.

Anon, 2023c. Retrieved May 16, 2023. https://apwg.org.

Anon, 2023d. Retrieved from October 1, 2022 to May 21, 2023. https://www.phishtank.com.

Anon, 2023e. Retrieved from October 1, 2022 to May 21, 2023. https://openphish.com.
Anon, 2023f. Retrieved from October 1, 2022 to May 21, 2023. https://www.malwareworld.com.
Anon, 2023g. Retrieved on October 1, 2023. https://www.domcop.com/top-10-million-websites.
Bountakas, P., Xenakis, C., 2023. Helphed: hybrid ensemble learning phishing email detection. J. Netw. Comput. Appl. 210, 103545. https://doi.org/10.1016/j.jnca.2022.103545.
Ding, Y., Luktarhan, N., Li, K., Slamu, W., 2019. A keyword-based combination approach for detecting phishing webpages. Comput. Secur. 84, 256–275. https://doi.org/10.1016/j.cose.2019.03.018.
Gupta, B.B., Yadav, K., Razzak, I., Psannis, K., Castiglione, A., Chang, X., 2021. A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment. Comput. Commun. 175, 47–57. https://doi.org/10.1016/j.comcom.2021.04.023.
Huang, Y., Ma, M., 2023. Ill-ids: an incremental lifetime learning ids for vanets. Comput. Secur. 124, 102992. https://doi.org/10.1016/j.cose.2022.102992.
Jain, A.K., Debnath, N., Jain, A.K., 2022. APuML: an efficient approach to detect mobile phishing webpages using machine learning. Wirel. Pers. Commun. 125 (4), 3227–3248. https://doi.org/10.1007/s11277-022-09707-w.
Kävrestad, J., Hagberg, A., Nohlberg, M., Rambusch, J., Roos, R., Furnell, S., 2022. Evaluation of contextual and game-based training for phishing detection. Future Internet 14 (4), 104. https://doi.org/10.3390/fi14040104.
Maroofi, S., Korczyński, M., Hölzel, A., Duda, A., 2021. Adoption of email anti-spoofing schemes: a large scale analysis. IEEE Trans. Netw. Serv. Manag. 18 (3), 3184–3196. https://doi.org/10.1109/TNSM.2021.3065422.
Nagunwa, T., Kearney, P., Fouad, S., 2022. A machine learning approach for detecting fast flux phishing hostnames. J. Inf. Secur. Appl. 65, 103125. https://doi.org/10.1016/j.jisa.2022.103125.
Nurhas, I., Aditya, B.R., Jacob, D.W., Pawlowski, J.M., 2022. Understanding the challenges of rapid digital transformation: the case of COVID-19 pandemic in higher education. Behav. Inf. Technol. 41 (13), 2924–2940. https://doi.org/10.1080/0144929X.2021.1962977.
Pandey, P., Mishra, N., 2023. Phish-Sight: a new approach for phishing detection using dominant colors on web pages and machine learning. Int. J. Inf. Secur. 1 (11). https://doi.org/10.1007/s10207-023-00672-4.
Prapas, I., Derakhshan, B., Mahdiraji, A.R., Markl, V., 2021. Continuous training and deployment of deep learning models. Datenbank Spektrum 21 (3), 203–212. https://doi.org/10.1007/s13222-021-00386-8.
Prasad, A., Chandra, S., 2022a. VMFCVD: an optimized framework to combat volumetric DDoS attacks using machine learning. Arab. J. Sci. Eng. 47 (8), 9965–9983. https://doi.org/10.1007/s13369-021-06484-9.
Prasad, A., Chandra, S., 2022b. Machine learning to combat cyberattack: a survey of datasets and challenges. J. Defense Model. Simul., 15485129221094881. https://doi.org/10.1177/15485129221094881.
Prasad, A., Chandra, S., 2023. BotDefender: a collaborative defense framework against botnet attacks using network traffic analysis and machine learning. Arab. J. Sci. Eng. https://doi.org/10.1007/s13369-023-08016-z.
Purwanto, R.W., Pal, A., Blair, A., Jha, S., 2022. PhishSim: aiding phishing website detection with a feature-free tool. IEEE Trans. Inf. Forensics Secur. 17, 1497–1512. https://doi.org/10.1109/TIFS.2022.3164212.
Rao, R.S., Umarekar, A., Pais, A.R., 2022. Application of word embedding and machine learning in detecting phishing websites. Telecommun. Syst., 1–13. https://doi.org/10.1007/s11235-021-00850-6.
Rupa, C., Srivastava, G., Bhattacharya, S., Reddy, P., Gadekallu, T.R., 2021. A machine learning driven threat intelligence system for malicious URL detection. In: Proceedings of the 16th International Conference on Availability, Reliability and Security, pp. 1–7.
Sáez-de-Cámara, X., Flores, J.L., Arellano, C., Urbieta, A., Zurutuza, U., 2023. Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks. Comput. Secur. 131, 103299. https://doi.org/10.1016/j.cose.2023.103299.
Sahingoz, O.K., Buber, E., Demir, O., Diri, B., 2019. Machine learning based phishing detection from URLs. Expert Syst. Appl. 117, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029.
Sameen, M., Han, K., Hwang, S.O., 2020. PhishHaven – an efficient real-time ai phishing URLs detection system. IEEE Access 8, 83425–83443. https://doi.org/10.1109/ACCESS.2020.2991403.
Sharma, B., Singh, P., 2022. An improved anti-phishing model utilizing TF-IDF and AdaBoost. Concurr. Comput., Pract. Exp. 34 (26), e7287. https://doi.org/10.1002/cpe.7287.
Shih, W.C., Yang, C.T., Jiang, C.T., Kristiani, E., 2023. Implementation and visualization of a netflow log data lake system for cyberattack detection using distributed deep learning. J. Supercomput. 79 (5), 4983–5012. https://doi.org/10.1007/s11227-022-04802-y.
Srijayanthi, S., Sethukarasi, T., 2023. Design of privacy preserving model based on clustering involved anonymization along with feature selection. Comput. Secur. 126, 103027. https://doi.org/10.1016/j.cose.2022.103027.
Taha, A., 2021. Intelligent ensemble learning approach for phishing website detection based on weighted soft voting. Mathematics 9 (21), 2799. https://doi.org/10.3390/math9212799.
Unal, D., Hammoudeh, M., Khan, M.A., Abuarqoub, A., Epiphaniou, G., Hamila, R., 2021. Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of things. Comput. Secur. 109, 102393. https://doi.org/10.1016/j.cose.2021.102393.
Verma, R., Chandra, S., 2023. RepuTE: a soft voting ensemble learning framework for reputation-based attack detection in fog-IoT milieu. Eng. Appl. Artif. Intell. 118, 105670. https://doi.org/10.1016/j.engappai.2022.105670.
Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., Woźniak, M., 2020. Accurate and fast URL phishing detector: a convolutional neural network approach. Comput. Netw. 178, 107275. https://doi.org/10.1016/j.comnet.2020.107275.
Wen, T., Xiao, Y., Wang, A., Wang, H., 2023. A novel hybrid feature fusion model for detecting phishing scam on Ethereum using deep neural network. Expert Syst. Appl. 211, 118463. https://doi.org/10.1016/j.eswa.2022.118463.
Xiao, Y., Liu, L., Ma, Z., Wang, Z., Meng, W., 2021. Defending co-resident attack using reputation-based virtual machine deployment policy in cloud computing. Trans. Emerg. Telecommun. Technol. 32 (9), e4271. https://doi.org/10.1002/ett.4271.
Zhang, W., Jiang, Q., Chen, L., Li, C., 2017. Two-stage ELM for phishing web pages detection using hybrid features. World Wide Web 20, 797–813. https://doi.org/10.1007/s11280-016-0418-9.

Arvind Prasad is pursuing PhD in Computer Science from the Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, UP. His research interest includes cybersecurity, IoT, reverse engineering, network traffic analysis, and machine learning.

Shalini Chandra is an assistant professor in the Department of Computer Science at the Babasaheb Bhimrao Ambedkar University, Lucknow, UP. Her research area is software security, software quality, cloud computing, and IoT.