IEEE Transaction on Machine Learning,Volume:11,Issue Date:19.June.2023

PhishCatcher: Client-Side Defense Against Web Spoofing Attacks Using Machine Learning

MUZAMMIL AHMED[1], AHMED B. ALTAMIMI[2], WILAYAT KHAN[1], MOHAMMAD ALSAFFAR[3], AAKASH AHMAD[4], ZAWAR HUSSAIN KHAN[5], AND ABDULRAHMAN ALRESHIDI[3]

[1]Department of Electrical and Computer Engineering, COMSATS University Islamabad, Wah Campus, Rawalpindi 47040, Pakistan
[2]Department of Computer Engineering, University of Ha'il, Ha'il 55422, Saudi Arabia
[3]Department of Information and Computer Science, University of Ha'il, Ha'il 55422, Saudi Arabia
[4]School of Computing and Communications, Lancaster University Leipzig, 04109 Leipzig, Germany
[5]Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada

Corresponding author: Wilayat Khan (wilayat@ciitwah.edu.pk)

This work was supported by the Scientific Research Deanship at University of Ha'il, Saudi Arabia, under Project RG-22 020.

ABSTRACT Cyber security confronts a tremendous challenge of maintaining the confidentiality and integrity of user's private information such as password and PIN code. Billions of users are exposed daily to fake login pages requesting secret information. There are many ways to trick a user to visit a web page such as, phishing mails, tempting advertisements, click-jacking, malware, SQL injection, session hijacking, man-in-the-middle, denial of service and cross-site scripting attacks. Web spoofing or phishing is an electronic trick in which the attacker constructs a malicious copy of a legitimate web page and request users' private information such as password. To counter such exploits, researchers have proposed several security strategies but they face latency and accuracy issues. To overcome such issues, we propose and develop client-side defence mechanism based on machine learning techniques to detect spoofed web pages and protect users from phishing attacks. As a proof of concept, a Google Chrome extension dubbed as *PhishCatcher*, is developed that implements our machine learning algorithm that classifies a URL as suspicious or trustful. The algorithm takes four different types of web features as input and then random forest classifier decides whether a login web page is spoofed or not. To assess the accuracy and precision of the extension, multiple experiments were carried on real web applications. The experimental results show remarkable accuracy of 98.5% and precision as 98.5% from the trials performed on 400 classified phished and 400 legitimate URLs. Furthermore, to measure the latency of our tool, we performed experiments over forty phished URLs. The average recorded response time of *PhishCatcher* was just 62.5 milliseconds.

INDEX TERMS Web spoofing, security and privacy, machine learning, web security, browser extension.

I. INTRODUCTION

In Oct 2022,[1] the members/users of the National Institute for Research in Digital Science and Technology (Inria) in France received an email in French asking the users to confirm their webmail account with the direct link https://www.education-online.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html. When clicked on this link, it takes to a fake but appearing genuine central authentication login page of Inria. As this fake login page resembles the real login page of Inria from https://cas.inria.fr/cas/login?service=, users will mistakenly enter username and password of the Inria to a fake website which the attacker can later submit to the real Inria login page. This is a phishing attack on the Inria and users/members registered with Inria. The real and fake login pages of Inria are given in the Figures 1. Both of the web pages are exactly the same and it is easy for the users to fall victim of this phishing attack. We have tested our tool *PhishCatcher* against this and few other attacks as detailed in the Section V.

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

[1]An email, warning the users of Coq-club Inria https://www.inria.fr/en about this phishing attack, was received on Oct 10, 2022

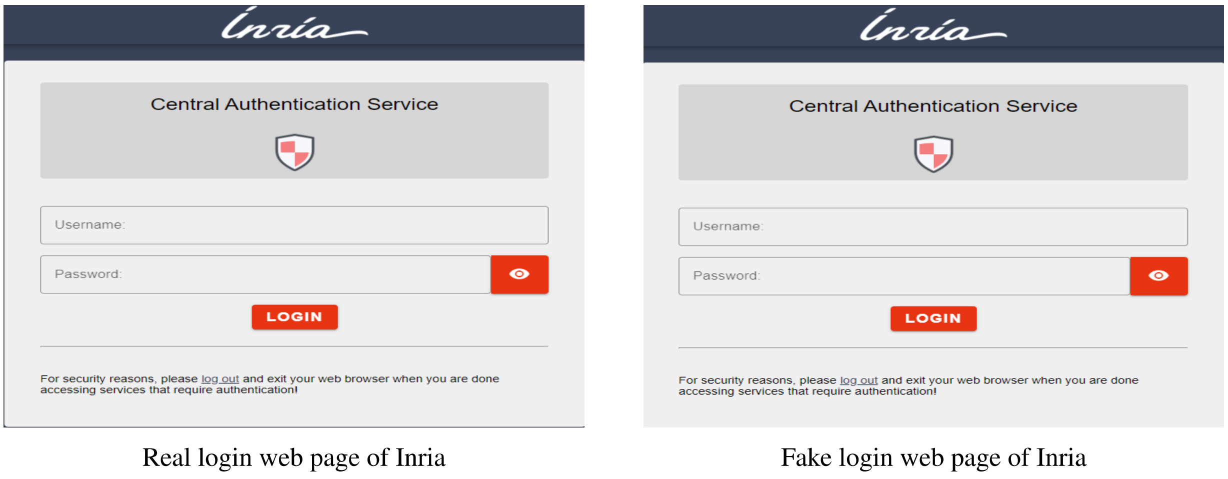

Inria
Central Authentication Service
Username:
Password:
LOGIN
For security reasons, please log out and exit your web browser when you are done accessing services that require authentication!
Real login web page of Inria
Inria
Central Authentication Service
Username:
Password:
LOGIN
For security reasons, please log out and exit your web browser when you are done accessing services that require authentication!
Fake login web page of Inria

FIGURE 1. **Phishing attack on Inria.**

With the tremendous advancement in modern technologies, there has been a great escalation in the online world, such as e-commerce, online banking, distant learning, e-health and e-governance. Since social networking applications, such as Facebook and Twitter, are performing leading role in the globalization of the modern era, billions of users have adopted this increasing trend. Numerous websites provide the web-users with an opportunity to create an account for a customized experience. To obtain online specialized services from the web-sites, users are required to create a personalized account. Conventionally, users are exposed to login web pages for this purpose where they have to set up an account by creating and registering an identification (e.g., username) and secret (e.g., password). Next time, when the user needs to access the remote resource or service, she/he sends a web requests and receives a login form for submitting the identification along with the secret. At this point, the users' privacy is at high risk in terms of identity theft and confidential information. A phishing attack scenario, as described in Figure 2, begins with receiving an email with a link to malicious website [1]. The email message might contain text convincing or luring the user to click and follow the pointer. When the unsuspecting user clicks and opens the web page, it appears genuine as the honest website where the user has an account. After the victim user enters his/her secret information, such as the username and password and presses the submit/login button, they are sent to the attacker. The attacker who sat up the phishing attack receives the secret credentials and logins to the legitimate website upon submitting the credentials to it.

Identity theft, online frauds and scams have immensely increased since the advent of web spoofing or phising attacks. Web spoofing or phishing is a type of cyber crime in which a malicious intruder tries to steal valuable data from the user. Attackers have adopted many phishing and web spoofing techniques to threaten online systems. Initially, web-spoofing was used for identity theft but now attackers are using it to steal sensitive information related to national

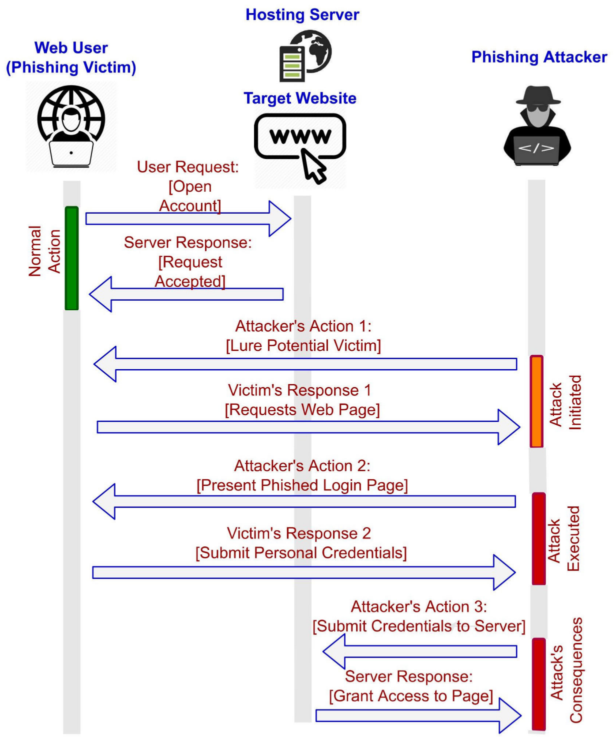

Hosting Server
Web User (Phishing Victim)
Phishing Attacker
Target Website
www
User Request: [Open Account]
Normal Action
Server Response: [Request Accepted]
Attacker's Action 1: [Lure Potential Victim]
Victim's Response 1 [Requests Web Page]
Attack Initiated
Attacker's Action 2: [Present Phished Login Page]
Victim's Response 2 [Submit Personal Credentials]
Attack Executed
Attacker's Action 3: [Submit Credentials to Server]
Server Response: [Grant Access to Page]
Attack's Consequences

FIGURE 2. **A typical phishing attack.**

security, intellectual property and organizational secrets. Current era's phishing attacks have already been entered into a new evolutionary dimension including, but not limited to, QR code phishing, spoofing application for mobile and spear phishing etc. Such attacks and scam approaches may circumvent the protections such as firewalls, digital certificates, encryption software and other mechanisms like the two-factor authentication. Numerous companies are using such two-factor authentication systems to avoid monetary scams and identity theft. Sadly, the advanced scam approaches have made all these systems vulnerable [2].

To deceive the victims, the attackers normally include logos, either by storing copies or adding links to logos, from the honest site onto their spoof sites to imitate their appearance. In addition to logos, the attacker may also include HTML from the honest site and make some necessary changes. The phishing attack vectors used by the attackers for tricking the users include email, trojan horse, key loggers and manin-the-middle proxies. The favourite attack targets of the attackers are online banking sites, third party payment systems (the most targeted industry sector) and e-commerce sites [3]. As the phisers target the non-cryptographic components, the cryptographic security protocols SSL/TLS do not provide a complete solution. To depend against spoofing attacks, these protocols must be complemented with additional protection mechanisms [4]. These mechanisms may be enforced at the server-side or client-side or both. The server-side solutions [5], [6] requires changes to the websites which is a tedious job and is often ignored by most of the developers [7]. The client-side solutions, on the other hand, provide protection to users without the server support. Server-side solutions may be effective in identifying spoofed site, however, the focus of this paper is on client-side solutions. Most of the anti-spoofing tools are based on either the third party certification [8], password [9] or URLs [3].

Anti-spoofing tools are sometimes categorized as stateful or stateless. They may also be classified based on the automatic phishing detection mechanism used: blacklists and heuristics. Tools that rely on black/white lists generate almost zero false positives (accuracy) and can recognize almost 90% of the phishing sites [10], however, they miss zero-day attacks [11]. Furthermore, black-listing methodologies come with several drawbacks as they cannot control the changing domain and new attacks and can easily be fooled by the spam URLs [12]. To capture phish sites not included in the black lists, the heuristic-based techniques have been very encouraging. The heuristic (content) based tool such as CANTINA [13] and SpoofCatch [1] can identify 90% phishing sites with 1% false positives. The latency of the tool SpoofCatch is in the order of seconds and it further increases with passage of time. While the stateful anti-phish techniques are good in accuracy, they quickly fill the local storage and the performance degrades with passage of time. In SpoofCatch, the visual similarity is initially compared with few login page images, but as the user browse further websites, the number of login page images increases in the local storage. In addition, this increases the time to compare the image of a received login page with every login image in the storage. Following this line of research, we design and develop a stateless anti-phish tool based on the Machine Learning (ML) technique.

From the last decade, many renowned researchers have proposed machine learning techniques for the detection of malicious URLs to avoid any kind of scam in future. Many sets of URLs are treated as training data in the ML approaches. On the basis of the statistical properties obtained by the training sets, it is proposed that whether the requested URL is a scam or scam free. Training data is the primary concern for the URL identification using ML. Once training data is obtained then it is further processed to obtain a mathematical model. The primary concern is to collect the features from the training data because simple strings may not help to predict the status of the URL under test. At final stage, an actual model is obtained through predicted model from the training data. Machine learning techniques, such as Naïve Bayes, Support Vector Machines (SVM) and Logistic Regression (LR), are a few algorithms being used for this purpose by many scholars but there are several issues which make them vulnerable [14].

In this paper, we propose and develop a stateless client-side tool, dubbed as *PhishCatcher*, to protect against web spoofing attacks. The *PhishCatcher*, a Google Chrome extension, is based on machine learning techniques and implements the random forest algorithm to classify whether or not a login web page is legitimate or spoofed. We have evaluated the efficiency and accuracy of the *PhishCatcher* on real web applications and the results were remarkable. The source code of the Google Chrome extension *PhishCatcher* is available online at the link https://github.com/wilstef/PhishCatcher. The major contributions of this research work are the following.

- A client-side anti-phishing mechanism based on the machine learning is proposed.
- Design and development of a Google Chrome extension, *PhishCatcher*, implementing the proposed mechanism.
- Careful selection of web features for the phish classifier algorithm used in the extension, and
- Experimental analysis of the *PhishCatcher*.

The rest of the paper is organized as follows. A summary of the related work in the literature is given in the next section. The detailed research methodology followed during this research is discussed in the Section III. The design and development of the Google Chrome extension is described in the Section IV. The testing results of the Chrome extension are included in the Section V and evaluated in the Section VI. The paper is concluded in section VII.

II. RELATED WORK

Currently, there are several open source techniques to prevent users from phishing attacks but most of them have some limitations such as latency, limited features set and generic database. This section provides an insight into the existing anti-phishing tools and frameworks used to discover and block phishing attacks. These anti-phishing tools and techniques are categorized into seven major schemes listed in the Table 1 and described in the following sub-sections.

A. VISUAL SIMILARITY AND PAGE CONTENT INVESTIGATION

An anti-phishing technique based on the visual similarity relies on the visual content of the web page received. Wilayat et al. [1] designed and developed a phish

TABLE 1. Summary of the anti-phishing schemes.

No.	Category
1	Visual similarity and page content investigation
2	Hybrid approach for phishing detection
3	Anti-phishing machine learning techniques
4	Online training procedures to prevent phishing
5	Automated classification of fake and genuine websites
6	URL analysis for detecting phishing
7	Significant anti-phishing tools

identification tool, called SpoofCatch, based on visual similarity. Using SpoofCatch, when the user first time visits a website, its login web page is identified and its screenshot is stored locally. When the user browse to the same website next time, the screenshot of its login page is compared with the locally stored ones. If a match is found with a local login page and the hosts of the login page received and previously visited are the same, the host is declared as genuine, otherwise, it is marked as phished. A promising strategy is offered in [15] for the visible distinction among a suspected phishing website and the legal one. This strategy utilizes three web features that play a key role to decide whether the two pages are suspiciously identical. These characteristics are the fragments of the text and their layout, pictures inserted inside the page, including the general visual presentation of the website presented by the browser. An experimental test, using a data collection consisting of 41 real-world phishing sites besides their respective genuine destinations, displayed remarkable returns regarding the error rate. Authors in the [16] suggest a novel way of phishing prevention based on the detailed spatial design features of the web pages. In this regard, two ways are suggested to extract the spatial arrangement attributes from a specified website as rectangle sections. A page similarity description is implied by considering the two web pages with their individual spatial layout attributes that take characteristics of their spatial architecture into account. An R-tree is created to list all the spatial layout characteristics of a valid page collection. Consequently, phishing identification based on the similarities of the spatial layout element is facilitated by appropriate spatial inquiries through the R-tree.

Zhang et al. [13] applied a content focused strategy to detect malicious phishing techniques. In the proposed methodology based on the Term Frequency-Inverse Document Frequency (TF-IDF) filter [17], 95% of the phishing URLs were detected accurately. A browser extension PWDHASH++ was proposed in the [18] for client-side protection against phishing. The authors suggested a method to identify visual similarities between the two web sites. The suggested solution, based on Gestalt philosophy [19], acknowledges a web page as a single indivisible entity. These indivisible super signals are explicitly evaluated using algorithmic complexity analysis.

B. HYBRID APPROACH FOR PHISHING DETECTION

A multidimensional spoofing and phishing detection feature has been modeled by the authors in [20]. This bi-step approach is primarily based on the deep learning algorithm. The authors proposed a Dynamic Category Decision Algorithm (DCDA) based on deep learning. More than a million malicious URLs were proceeded through this model. Results showed that their protection mechanism based on the proposed algorithm consumed less time to detect web-spoofing. A hybrid machine learning approach against phishing threats has been proposed by the authors in [21]. To build an effective model, five machine learning techniques have been used. The four-layered suggested model was then compared with the existing models after training on the necessary data set which included a significant number of URLs. Results demonstrated that the developed strategy was more efficient and effective. Kaur and Sharma [22] implemented the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [23] algorithm for malicious e-mail detection. An interesting feature of their implementation is that, after a phished URL is detected, it automatically generates a mail and sends it to the victim server. The email message includes the IP, location and contact info of the attacker server and blocks all the traffic coming from the server with malicious intentions. The authors in [24] have combined the machine learning and Resource Description Framework (RDF) to reduce false positives and enhance accuracy of their proposed model. Several machine learning approaches have been applied by the authors in [25] such as Linear Model (LM), Decision Tree (DT), Random Forest (RF) and Neural Networks (NNs) on the test data to detect phishing and malicious sites.

C. ANTI-PHISHING MACHINE LEARNING TECHNIQUES

Many researchers have designed effective, reliable and robust solutions for malicious URL detection based on machine learning techniques. Mao et al. [26] have described few attributes of web page that can be implemented to recognize phished URLs. They designed a logistic regression classifier and used it as a filter to distinguish phishing sites. It was observed that out of millions of URLs, approximately 777 phishing web sites were visited per day and almost 8.24% users were affected. In [14], the authors have evaluated nine techniques based on machine learning methodologies such as LR, RF, AdaBoost, SVM, NN, Naıve Bayes, Bagging and Bayesian additive regression. The trained data set was based on 1500 phishing URLs and it was classified by machine learning. The authors in [27] applied a new tactic for phishing detection by designing a scalable classifier based on the machine learning. They trained their proposed model on the noisy data-sets. Their results showed that about 90% of the malicious URLs were detected using this approach.

A PART-algorithm is used for spoof detection in [28] by the authors. They have implemented MAP-REDUCE [29] technique to boost-up the detection procedure. Jain et al. [30] carried out a comprehensive survey on existing techniques used for phishing detection across the globe. A Natural Language Processing (NLP) model based on machine learning has been described in [31] for identifying the illegitimate social media accounts. An SVM tool is used to speed up the over all process. Xiang et al. [32], proposed an anti-phishing approach based on CANTINA+ model. A filtering algorithm has been adopted to lower FP ratios. Moreover, the designed model was trained on linear and non-linear phishing test-beds. Lakshmi and Vijaya [33] applied supervised machine learning techniques including multi-layer perceptron, Naıve Bayes classifier and decision tree classifier to classify and predict malicious websites. Different features were extracted from a collection of 200 URLs and the HTML source codes of the bogus and legal websites. The two performance standards, predictive precision and quick learning combined with 10-fold cross validation determined the efficiency of the model. Their findings showed that the decision tree classifier outperformed the rest of the classifiers.

A detailed analysis and systematic interpretation of the adopted machine learning approaches for the malicious URL identification is proposed by the authors in [34]. The article further demonstrates the enhancement of literature studies that address different aspects of this issue (feature description, algorithm architecture, etc.). Random Forest Tree-based (RFT) algorithm is common in the computer vision and facial identification. The SVM is a form of machine learning used for the classification of facial recognition. Authors in the [35] evaluated the efficiency of facial recognition, output of the random forest and SVM by using the kernel parameters for optimization. Yu et al. [36] proposed a strategic advanced persistent threats (APT) [37] detection approach that utilizes deep learning in industrial internet of things (IIoT) [38]. In this approach, researchers used a well-known deep learning model called bidirectional encoder representations from transformers (BERT) [39], to detect APT attack patterns. The empirical findings confirm that the BERT system has high precision and a less error rate for spotting APT attack sequences than existing statistical models.

D. ONLINE TRAINING PROCEDURES PREVENTING PHISHING

While web spoofing and phishing attacks have sever effects on the users, several browser and server based techniques have been proposed to protect against such attacks [4], [40]. A comprehensive study on the login pages' security has been carried out in [41]. In this study, the authors have designed an efficient attacker model to check login security. To evaluate their model, a large number of login pages were tested and found that almost 63% of the pages were vulnerable to the attackers. In another study [42], the authors conducted a survey to identify fraudulent websites using online learning strategies that utilize lexical and host-based attributes of the corresponding URLs. They highlighted that this program is specifically relevant to online algorithms because the scale of the training data is larger. A real-time method was designed to capture URL attributes, together with a real-time source of labelled URLs, from a wide web mail provider. According to this research, newly established online algorithms are precise enough, such as batch strategies, delivering classification accuracy of 99% covering a diverse data collection. Authors in [43] demonstrated that phishing emails can be identified with great precision by applying a specific filter that utilizes parameters relevant to phishing attacks, rather than commonly used spam filters. In their study, the data set included 860 phishing and 6950 non-phishing emails. The results showed correct recognition rate of 96% with only 0.1% classification error. A phishing identification method was suggested in [44] that classifies website protection by testing the source code of the website. Certain phishing features, given by the World Wide Web Consortium (W3C) guidelines were extracted to determine website security. The source code of the website was tested for a phishing parameter and the initial secure weight was reduced if a phishing parameter was found. Ultimately, the security percentage was measured based on the final weight: the higher the percentage, the more stable a website would be.

Kumaraguru et al. [45] proposed the development and analysis of an embedded email training scheme to educate people regarding phishing. Laboratory operations contrast the efficacy of conventional phishing safety notifications with two integrated learning models proposed in the above cited paper. However, results showed that integrated training performs better than existing mailing safety notifications procedure, hence provided guidelines for the sound architecture of the embedded training systems.

E. AUTOMATED CLASSIFICATION OF FAKE AND GENUINE WEBSITES

Automated Individual White List (AWIL) is an innovative anti-phishing strategy which aims to preserve a white-list of all known Login User Interfaces (LUIs) of websites for visitors [46]. The AIWL warns the user of the potential threat if a user attempts to send sensitive details to a LUI which does not exist in the white list. Naive Bayesian classifier is implemented to maintain the white list automatically. The architecture and optimization techniques of a scalable machine learning classifier to detect suspicious websites is discussed in [47]. This classifier is used to manage the Google's blacklist dynamically. It investigates millions of pages a day by inspecting the URL and page content to decide if a website is fake or real.

F. URL ANALYSIS FOR DETECTING PHISHING

A lightweight URL based phishing detection approach was introduced by the authors in [48]. The data set consists of 1000 genuine and 1000 bogus URLs, whose evaluation is done by SVM. The suggested method only requires six URL characteristics to execute the identification. The most significant feature is the similarity index which is used first time ever. Another study [49] proposes an approach for the automated classification of fake and real URLs by implementing supervised learning over lexical and host based features. This scheme is complementary to the earlier techniques such as blacklisting. The status of the previously non-visited URLs cannot be predicted through blacklisting. Moreover, it is necessary to visit the potentially hazardous sites for the models which work on evaluating site content and behaviour.

Khonji et al. [50] initiated a research that seeks to test the functional efficacy of the website classification by lexical evaluation of URL tokens in enhancement to an innovative tokenization method to improve the prediction efficiency. This research implies an experimental HTTP proxy server to investigate over 70,000 valid and phishing URLs gathered during six months from PhishTank, Khalifa University HTTP logs and some volunteers. A predictive classification model is developed to determine the operative potency of the lexical URL study provided. As most of the phishing emails contain malicious URLs, magnifying website detection procedures can directly help the performance of anti-phishing email classifiers. Khonji et al. [51] expanded their study on enhancing the classification accuracy of the anti-phishing email filters with the suggested lexical URL analysis methodology.

G. SIGNIFICANT ANTI-PHISHING TOOLS

A browser extension *Spoofguard* was designed and developed by the authors in [52]. According to their proposed model, the browser extension was capable of displaying a window where photographic password displayed the credentials of the user. In this model, the user can select multiple images, against all the websites being visited by him/her, which are stored in the server. For efficient client-side-protection, a separate password is assigned by the extension to every URL under test. Furthermore, the browser extension, *Spoofguard*, informs the user in case of any scam. Yue et al. [11] developed an anti-phishing client side tool *BOGUSBITER* that operates on offensive defence strategy. It feeds bogus data to the malicious phishing site which makes it extremely hard for that bogus site to distinguish between actual and fake data-sets. In another attempt [53], the authors revealed the gravity of the threats based on the large scale web crawling. They found that hundreds of publisher pages were compromised by these attacks and breached major ad networks like *DoubleClick* [54]. Their perspectives obtained through the analysis led to create a new detection tool named as *MadTracer*. The assessment of *MadTracer* indicates that it successfully operates against malvertising and has captured 15 times more harmful domain tracks than Google 's Safe Browsing [55] and Microsoft Forefront combined.

Another tool, called *Prophiler* [56], aims to provide a filter capable of reducing the number of web pages that need to be automatically evaluated to recognize harmful websites. This framework acts as a front-end for *Wepawet*: a well known public complex analytics platform for network malware. The findings indicate that *Prophiler* is capable of significantly lowering the *Wepawet* [57] load with very low error level. Imran et al. developed DAISY [58], a simple lightweight identification and prevention system, to defend software defined networks (SDN) [59] against DoS assaults by restricting malicious activity from the hackers. In contrast to techniques that only restrict a host or a port, the suggested scheme is able to reactivate a port or a host when it is no longer receiving malicious traffic. The simulation findings demonstrate improved performance of SDN with DAISY in terms of CPU consumption, reaction speed, channel bandwidth and data rate.

III. RESEARCH METHODOLOGY

As part of our research methodology, we initially studied relevant literature to understand state of the art work on phishing attacks, web spoofing, machine learning and multiple mechanisms used for the detection of suspicious login pages with their pros and cons. In the next stage, several machine learning based frameworks for the detection of malicious login pages were investigated in the Section II. The comparison of these anti-phishing tools with our plug-ins is showcased in the Section VI. Furthermore, the Document Object Model (DOM) analysis, practice of JavaScript and Python were executed in order to develop a novel and sophisticated Google Chrome extension for the detection of spoofing attacks. The main idea was to develop a Google Chrome add-on to act as a classifier of fake and authentic login pages and show phishing warnings on the user screen.

Before choosing a suitable classifier model, selection of the desirable features is necessary. For this, we have focused primarily on the set of features widely implemented in the existing frameworks as elaborated in the related work section. Eventually we tried to carve out the most eminent, effective and easy to integrate features for our classifier. Our feature set includes the following features.

- Set of fake and their legitimate login page image pairs (visual similarity based)
- URL parameters (URL based)
- Web page content (content based), and
- Blacklist

Traditional classifiers used techniques like whitelisting, blacklisting, online learning strategies, lexical and host-based analysis of URLs as indicated in the Section II. Blacklisting, alone is not efficient as it does not anticipate the status of prior non-visited URLs. Moreover, classifiers based on online strategies were not accurate, while whitelisting and lexical based models had high latency. After web page feature extraction, a random forest classifier model is selected on the basis of the performance metrics such as latency, accuracy and efficiency. Subsequently, the classifier was trained using the supervised machine learning technique. The extracted features were then fed to the selected model in order to complete the learning process. After the completion of the learning process, the model is ready for testing and simulation. In other words, it can make prediction of whether the login web page response is spoofed or not. The main objective is to achieve efficiency in terms of latency, false positives and false negatives. The classifier tends to show better results after testing as illustrated in the Table 5.

A. MODEL SELECTION

Among the various methods proposed in the literature, data mining based methods are very handy in identifying phishing attacks. Subasi et al. [60] used various data mining tactics to categorize the web pages as valid or phished. Multiple classifiers were used to build an efficient phishing detection scheme. The random forest classifier seems to beat other techniques in detecting phishing attempts. These techniques, however, use machine learning libraries written in Python and hence they cannot be executed inside most of the browsers in real-time. The main objective of this research is to design a client-side tool to expose phishing attacks in real-time. One conventional strategy is that the prediction is made at server and then the plug-in is allowed to approach the server to check the status for each web page. This kind of server-based approach is good but web developers often do not follow standard practices and a web server compromise affects all the visiting users [61].

Unlike the classical approach, we propose to run the classification algorithm inside the browser rather than the server. This approach has numerous benefits like better privacy (the user 's browsing data is not required to leave the machine) and it is independent of the network latency. As in [60], we have implemented our technique using the scripting language JavaScript in a browser plug-in. Since JavaScript does not have sufficient ML libraries support and the client machines have limited processing abilities, the implementation needs to be made lightweight. The *PhishCatcher* enables the feature extraction process and classification inside the client 's browser and shows the warning on the user screen if there is a phishing threat.

B. PRE-PROCESSING

This step involves the choice of the relevant data-set for the purpose of extracting suitable features. Our data-set comprises of the data from the following four different resources.

- Mohammad et al. [62] highlighted very effective and adequate features which clearly demonstrated their efficiency in terms of detecting phishing attacks. This data-set is made available at the UCI Machine Learning Repository [63].
- Jalalian et al. published the most detailed collection of 90 hijacked journal websites [64]. We have used this collection for the testing and evaluation of our classifier.
- The set of 310 blacklisted URLs from the Phish-Tank [55].
- The set of 310 genuine URLs from moz.com/top500

C. FEATURES COLLECTION

This is the most tricky and difficult phase of this study. We confronted several challenges such as the absence of appropriate and well fitting data-sets. A number of authors [25], [28], [31] have proposed the anti-phishing mechanisms based on data mining and ML techniques. But most of those training data-sets are not sound, have no free access and are based on mere generalized set of rules. There is a disagreement in the literature regarding the ultimate attributes that distinguish phished websites. This makes it complicated to formulate a data-set that incorporates all the relevant features. Regardless of this fact, we tried to make a set of best suited features for our model by tactful analysis of existing strategies mentioned in the literature review. The most eminent among those techniques is the data-set suggested in [62].

We have categorized our features set into four groups.

1) Group-1: Address bar based
2) Group-2: Abnormal based
3) Group-3: HTML and JavaScript based
4) Group-4: Domain based

D. CLASSIFICATION AND CLASSIFIER SELECTION

For the classification process, which is known to be the foundation of the machine learning, we use the supervised learning approach in our model. Researchers have implemented various tools and machine learning techniques to validate their performance in identifying phishing attacks [14]. An interesting contrast of the most frequently used machine learning techniques for network intrusion detection is proposed in [65]. The standard machine learning classifiers are assessed using two openly available datasets, KDD99 and UNSW-NB15 [66]. The period required to develop a model for each classifier is also calculated in order to determine its efficiency. The research results reveal that the Decision Tree (DT), Random Forest (RF), Hoeffding Tree (HT) and K-Nearest Neighbors (KNN) classifiers outperform the other machine learning classifiers in the 10-fold cross validation test mode. Upon careful analysis of the existing strategies used for phishing attack identification, the random forest algorithm seems to surpass the rest of the techniques. The random forest creates and merges several decision trees to render a more reliable and sound forecast. It is a versatile, convenient-to-use and perhaps the most popular supervised machine learning algorithm. The random forest delivers a perfect result most of the time even without the hyper-parameter optimization. Along with its flexibility and versatility, it is applicable for both regression and classification problems (it covers 90% of the modern ML systems). The forest generated by the algorithm is an ensemble of decision trees generally practised with the *bagging* technique [67]. The basic principle for the *bagging* strategy is that the cumulative outcome is improved by a blend of learning models. By integrating several trees into one ensemble model, the random forest significantly reduces deviation from a stable design like a decision tree.

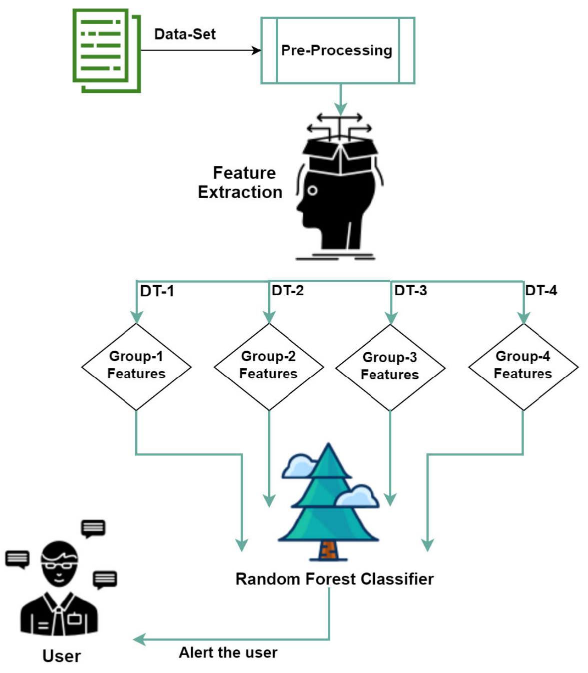


FIGURE 3. Random forest classifier for phishing detection.

It prevents data over-fitting and performs quicker training with the data-set. Furthermore, it can accommodate a high dimensional broad range of results which improves accuracy. Figure 3 depicts our proposed model for the lightweight phish identification method using random forest classifier.

Our proposed model for the lightweight phish identification method using random forest classifier is depicted in the Figure 3. Initially, a suitable data-set is selected as mentioned in the sub-section III-B. Subsequently, the desired features from the data-set are extracted based on their performance and compatibility. These features are grouped into four categories, as explained in the sub-section III-C, where each group acts as a decision tree. Finally, these groups of features are fed to the random forest classifier for the identification and classification of the phished URLs. In other words, the classifier informs the user about a potential phish attack. In the *PhishCatcher* browser extension, this is implemented through an alert notification to the user.

IV. PLUGIN DESIGN

Browser extensions or add-ons are miniature software packages that can modify and enhance the browsing experience according to the personal choice of the user. They are developed using web-based programming languages like HTML, CSS and JavaScript. This section provides a brief overview of the design and development of our tool *PhishCatcher*, a Google Chrome extension to identify and protect against phishing attacks. The main idea is to conduct the classification inside the client's browser and display the results simultaneously while improving the latency and privacy of the user's data. Figure 4 shows the basic layout of a Google Chrome extension.

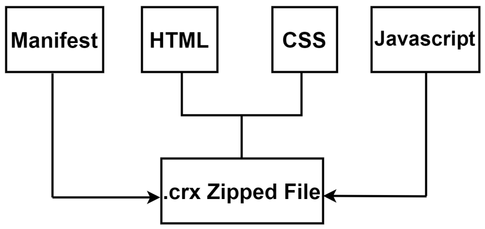


FIGURE 4. Layout of the Google Chrome extension.

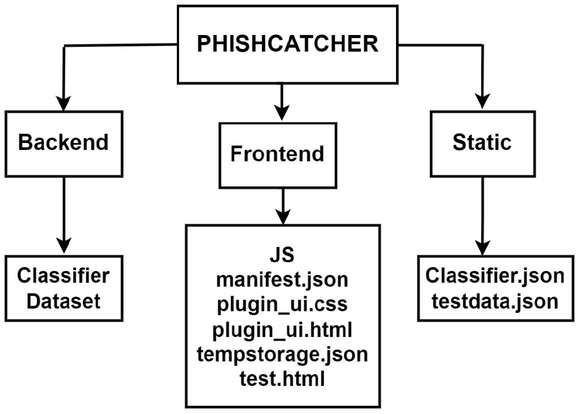
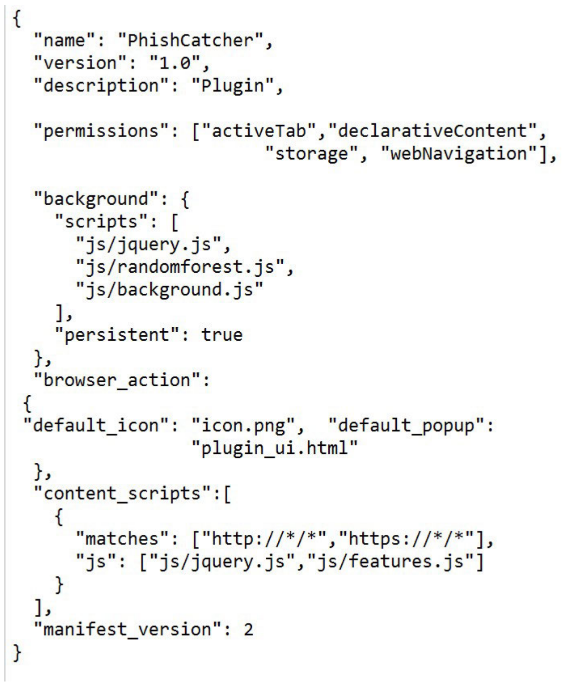


FIGURE 5. *PhishCatcher* architecture.

A browser extension is often designed as back-end, front-end and static modules. To implement the major desired functions through the *PhishCatcher* extension, these three modules are populated with different code and content type as described in the Figure 5. The classifier runs in the background (back-end) operating on the data set while the static part of the extension include the test data (*testdata.json*) and classifier parameters (*classifier.json*). The front-end consists of the executable code written in JavaScript, CSS and HTML.

A basic extension owns a manifest file having JSON format, called *manifest.json* that renders valuable information. The *manifest.json* file stores different information regarding the extension, including its name, version, description, manifest version, browser action and the permissions it requires. The most fundamental permissible extension is a directory having a *manifest.json* file. The manifest file of our extension is shown in the Figure 6.

The use case diagram (Figure 7) represents the interaction between the user and the system which illustrates the interaction of the user with the browser in different use cases. It is adapted to display the operational flow among the user and the system. The design of the *PhishCatcher* is ensured to be user friendly. It pops up a phishing alert as a browser response when a bogus URL is requested. The graphical user interface of the *PhishCatcher* has been designed so that it highlights the features responsible to classify a URL as a phish. Furthermore, it is capable of maintaining a white-list: if the user believe that the webpage marked as a phish by *PhishCatcher* is genuine and should not be blocked, it is



```
{
  "name": "PhishCatcher",
  "version": "1.0",
  "description": "Plugin",

  "permissions": ["activeTab","declarativeContent",
                      "storage", "webNavigation"],

  "background": {
    "scripts": [
      "js/jquery.js",
      "js/randomforest.js",
      "js/background.js"
    ],
    "persistent": true
  },
  "browser_action":
 {
 "default_icon": "icon.png",  "default_popup":
                 "plugin_ui.html"
  },
  "content_scripts":[
    {
      "matches": ["http://*/*","https://*/*"],
      "js": ["js/jquery.js","js/features.js"]
    }
  ],
  "manifest_version": 2
}
```

FIGURE 6. **Manifest file of the *PhishCatcher*.**

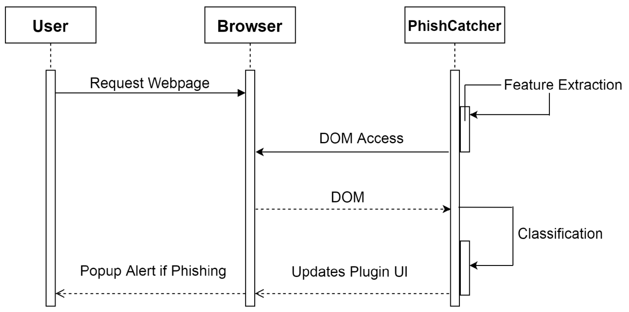


FIGURE 7. **Use case diagram.**

added to the white-list. In order to do so, the user just needs to click *OK* as shown in the test cases in the Section V.

V. TESTING

To assess the performance of the *PhishCatcher*, we tested and evaluated it against the real web application scenarios. This study mainly focuses on the aggregated analysis of all the features under consideration for the classification of legitimate and bogus URLs, rather than applying unit testing method for each feature. Nevertheless, screen shots of a few tested URLs taken by *PhishCatcher* are also presented here. The data-set considered in these tests contains:

- set of legitimate and corresponding fake URLs of 90 hijacked journals [64],
- set of 310 blacklisted URLs from PhishTank [55] and
- set of 310 legitimate URLs from the website https://moz.com/top500.

After several experiments, we extracted seventeen prominent features, categorically listed in the Table 2. Sub-sets of

TABLE 2. **Prominent features of the phishing URLs.**

No.	Feature Name	Category
1.	IP address	Address bar based features
2.	URL length	Address bar based features
3.	Tiny URL	Address bar based features
4.	“@” symbol	Address bar based features
5.	Redirecting using //	Address bar based features
6.	Prefix/suffix in the domain	Address bar based features
7.	No. of sub-domains	Address bar based features
8.	HTTPS	Address bar based features
9.	Favicon	Address bar based features
10.	Using non-standard port	Address bar based features
11.	HTTPS in URL’s domain part	Address bar based features
12.	Request URL	Abnormal based features
13.	URL of anchor	Abnormal based features
14.	Links in meta, script and link	Abnormal based features
15.	Server form handler	Abnormal based features
16.	Submitting to mail	Abnormal based features
17.	Using iFrame	HTML/JavaScript features

these features have been previously used in different tools and analysis [62], [68]. The significance of each feature varies according to the nature of the URL being tested by our plug-in.

A. TEST CASES

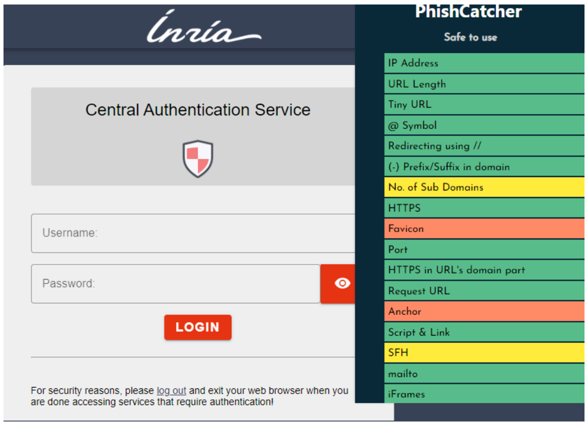
This section includes a concise view of the results generated by the *PhishCatcher* in terms of testing fraudulent and reliable URLs from our data-set. A few test cases are introduced here to provide a better understanding and performance analysis of our approach.

Test case 1
URL: https://www.education-online.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html
Result: Phishing

Our firs test case is testing *PhishCatcher* against the recent phishing attack on Inria as introduced in the Section I. This attack is so sophisticated that the attackers have designed the fake login web page as an exact copy of the login web page of Inria but still our tool captures it. Both, the real https://cas.inria.fr/cas/login?service= and fake https://www.education-online.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html URLs are tested and the *PhishCatcher* correctly identify the genuine and fake login pages as shown in the Figure 8. Out of the seventeen features of the phishing URLs (Table 2), six features namely *URL length*, *prefix/suffix* length in the domain, *favicon*, *request* URL, *anchor* and *script link* contributed towards correctly identifying this phishing attack.

Test case 2
URL: http://www.ijiq.com
Result: Phishing

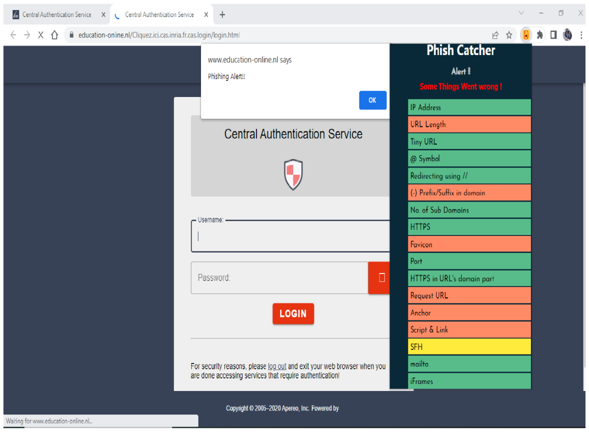
The second test case refers to the scenario of a spam URL of a hijacked journal in which the user receives a promotional email with the link to a bogus but seemingly legitimate site www.ijiq.com. When the user clicks the link,



IP Address	🟩
URL Length	🟩
Tiny URL	🟩
@ Symbol	🟩
Redirecting using //	🟩
(-) Prefix/Suffix in domain	🟩
No. of Sub Domains	🟨
HTTPS	🟩
Favicon	🟥
Port	🟩
HTTPS in URL's domain part	🟩
Request URL	🟩
Anchor	🟥
Script & Link	🟩
SFH	🟨
mailto	🟩
iFrames	🟩

For security reasons, please log out and exit your web browser when you are done accessing services that require authentication!

Genuine login web page


Central Authentication Service
Central Authentication Service
education-online.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html
Phish Catcher
Alert !
Some Things Went wrong !
OK
Central Authentication Service
Username:
Password:
LOGIN
Copyright © 2005–2020 Apereo, Inc. Powered by
Waiting for www.education-online.nl...

www.education-online.nl says

Phishing Alert!!

IP Address	🟩
URL Length	🟥
Tiny URL	🟩
@ Symbol	🟩
Redirecting using //	🟩
(-) Prefix/Suffix in domain	🟥
No. of Sub Domains	🟩
HTTPS	🟩
Favicon	🟥
Port	🟩
HTTPS in URL's domain part	🟩
Request URL	🟥
Anchor	🟥
Script & Link	🟥
SFH	🟨
mailto	🟩
iFrames	🟩

For security reasons, please log out and exit your web browser when you are done accessing services that require authentication!

Phished login web page

FIGURE 8. Genuine and phished login web pages of Inria.

the attacker's website loads in the user's web browser. The user inserts his/her credentials since the website seems reliable. This exploit was positively detected by our plugin. It is clear from the first sub-figure in Figure 9 that the *PhishCatcher* pops up a phishing alert as a browser response when a bogus URL is requested. Table 3 categorizes the features accountable for diagnosing the phished URL employed in the test case 2. Another case of the phished URL of a hijacked journal is depicted in the second sub-figure of Figure 9. The attributes involved in identifying the phished URL from the test case 3 are sorted in the Table 4.

Test case 3
URL: http://www.revistas-academicas.com
Result: Phishing

Test case 4
Authentic web page: http://www.ahistcon.org/revistaayer.html
Counterfit web page: http://www.ayeronline.com

This test case represents the performance of *PhishCatcher* by implementing the test over genuine and corresponding hijacked URL for the same journal. The first sub-figure in Figure 10 displays the result generated by the *PhishCatcher* in correctly identifying the genuine URL of the journal, while the second sub-figure in Figure 10 depicts the positive detection of the spam URL of the corresponding hijacked URL of the journal.

Test case 5
The testing results for a few top ranked commonly used legit websites are displayed in this case (Figure 11). The *PhishCatcher* correctly identifies these web pages as safe to use. The tool correctly identified the web pages of Facebook, Google, Microsoft and Apple accessed from the URLs https://facebook.com, https://support.google.com, https://www.microsoft.com/en-pk and https://www.apple.com, respectively.

VI. EVALUATION

The proposed model was tested over a succession of trials to assess the accuracy and latency of our tool. The results of latency experiments are given and discussed in the sub-section VI-B. The other findings related to the performance were recorded in the form of a confusion matrix for further calculation of precision, recall and accuracy of the model.

A. PERFORMANCE METRICS

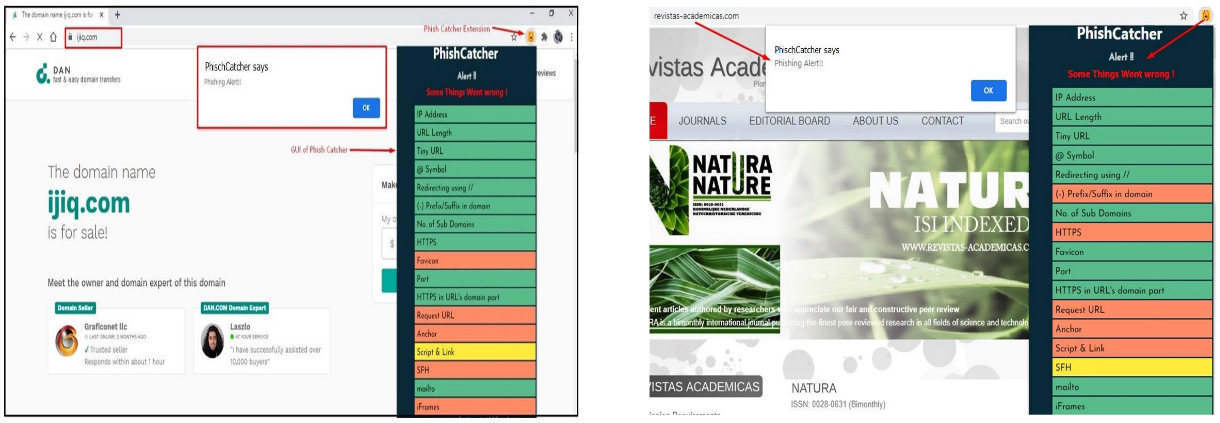
A confusion matrix is a tabular configuration utilised to characterize the performance of a classifier. It tends to anticipate the efficiency of a supervised learning algorithm over a collection of testing data for which the valid values are known. All matrix rows denote the occurrences in a projected class, while every column signifies the cases in an original class. The performance metrics such as *precision*, *recall* and *accuracy* of our plug-in have been calculated by the Equations 1, 2 and 3, respectively.

$$\frac{TP}{TP+FP} \tag{1}$$

$$\frac{TP}{TP+FN} \tag{2}$$

$$\frac{TP+TN}{TP+FP+TN+FN} \tag{3}$$

The values of variables have been assigned from a confusion matrix given in the Table 5, where `TP` stands for *True Positive*, `FP` stands for *False Positive*, `TN` denotes *True Negative* and `FN` represents *False Negative*. The letters `P` and `N` indicate *Positive* and *Negative*, respectively. True positive is a case when the phished URL is correctly identified, while in case of false positive, a legitimate URL is mistakenly identified as phished. Similarly, true negative is the scenario when a legitimate URL is correctly identified as genuine, while in the case of false negative, a phished URL is mistakenly declared as genuine. We performed the experiments over a dataset of 800, which included 400 phished and 400 benign, URLs for the classification of fake and authentic URLs. The



Phished http://www.ijiq.com/ Phished http://www.revistas-academicas.com/

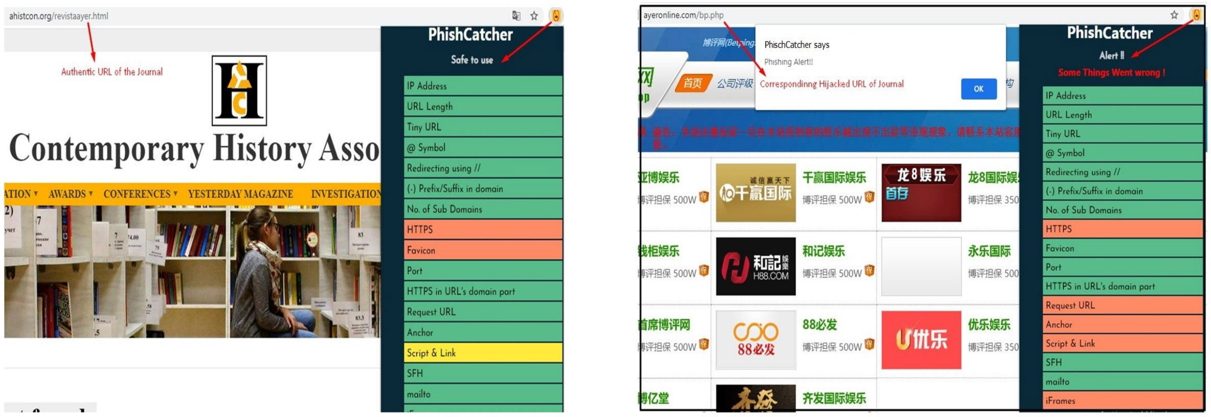
FIGURE 9. Test cases 2-3.

TABLE 3. Test case 2.

Feature	Rule	Category
Favicon	Phishing = Favicon loaded from the external domain	Address bar based
Request URL	Phishing = % of request URL > 61	Abnormal based
URL of anchor	Phishing = % of URL of anchor > 67	Abnormal based
SFH	Phishing = SFH is "about: blank" or empty	Abnormal based
iFrames	Phishing = Using iFrames for redirection	HTML and JavaScript based

TABLE 4. Test case 3.

Feature	Rule	Category
HTTPS	Phishing = https not used or used with non trusted user	Address bar based
Request URL	Phishing = % of request URL > 61	Abnormal based
URL of anchor	Phishing = % of URL of anchor > 67	Abnormal based
Meta, script, link	Phishing = % of links in (meta, script, link) > 81	Abnormal based



Genuine URL Hijacked URL

FIGURE 10. Genuine and hijacked URLs.

scores have been recorded after multiple iterations and careful analysis of the extension. Consequently, the *PhishCatcher* exhibited phenomenal *accuracy* of 98.5%, *precision* of 98.5% and *recall* turned out to be 98.5%.

Zhang et al. [69] developed an automated inspection plot for the evaluation of anti-phishing tools. In [69], the performance of ten common anti-phishing tools was measured using 200 tested phished URLs (from two sources)

TABLE 5. Confusion matrix.

Group	Classified Phishing	Classified Legitimate	Total
Actual phishiing	TP=394	FN=6	P=400
Actual legitimate	FP=6	TN=394	N=400
Total	P=400	N=400	800

and 516 valid URLs. Just one tool *SpoofGuard* was able to accurately classify more than 90% of phishing URLs;

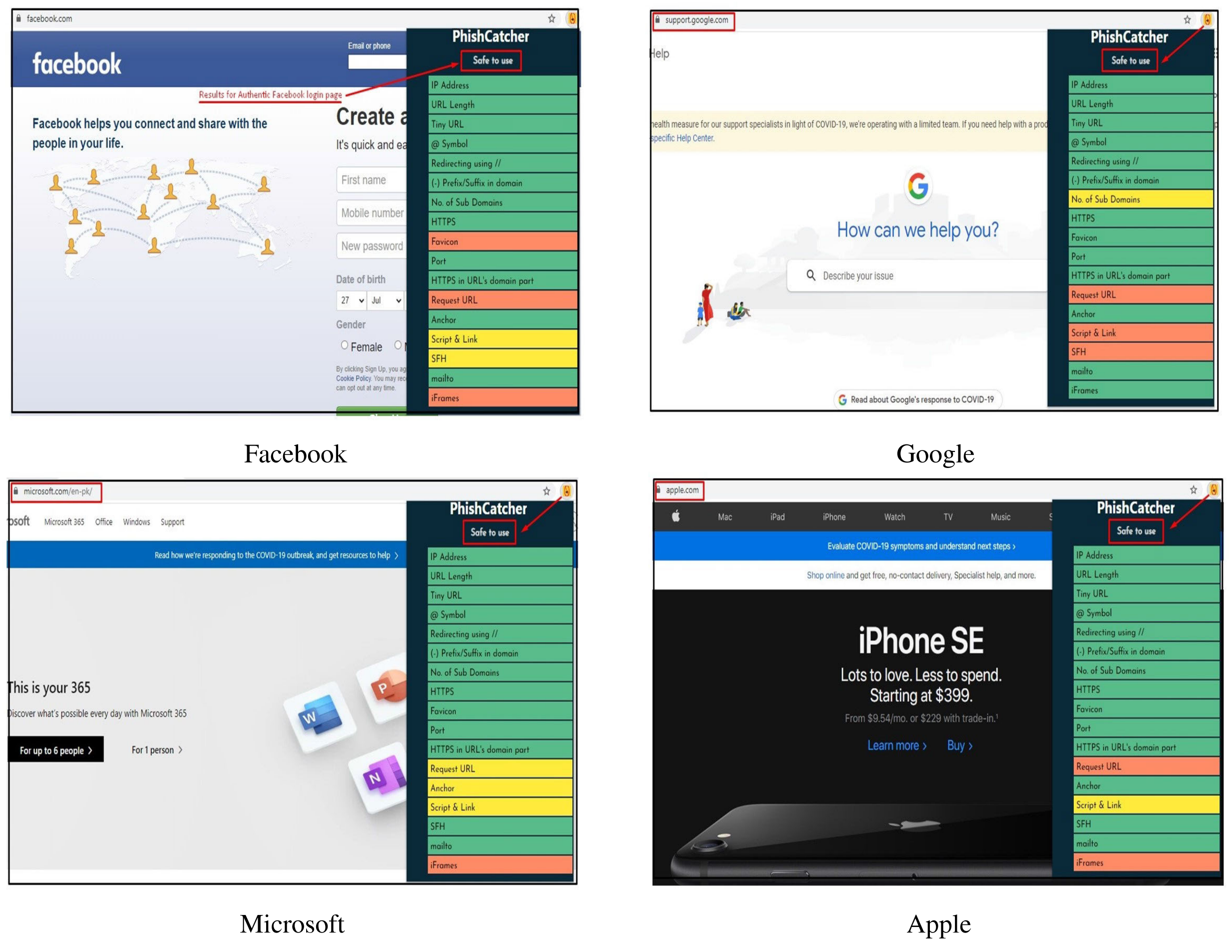


FIGURE 11. **Genuine login pages of Facebook, Google, Microsoft and Apple, respectively.**

nevertheless, 42% of genuine URLs were still mistakenly marked as a phish. The efficiency of other tools diversified considerably depending on the origin of the phishing URLs. Among these remaining tools, only one tool *IE7* correctly classified more than 60% among phishing URLs from both sources, however, it still failed to spot 25% of the Anti-Phishing Working Group(APWG) phishing URLs and 32% of phished URLs from *phishtank.com.* Table 6 represents a comparison of eminent anti-phishing tools from [69] with our plug-in *PhishCatcher* in terms of identifying a phishing URL. The results are evaluated using 100 bogus URLs from [55].

B. LATENCY

Latency can be defined as the speed or how fast an anti-phishing tool can detect a phish. It depends on a number of aspects such as the algorithm implemented, computing power, network speed and the nature of the tool (stateless or stateful). Assuming the computational resources are common, the decision by a stateful tool is made upon the current web page as well as the previous data stored locally or at a remote server. The latency of such tools depend on the algorithm implemented, the network speed as well as the size of the data. In a stateless tool such as the *PhishCatcher*, no previous data is required and hence the decision is dependent on the algorithm implemented.

To measure the latency of the *PhishCatcher*, we performed experiments by running it over forty phished URLs. Before loading and running the extension in the browser, we updated the code to record the time when it *start*s the computation and the *decision* time when it announces the result. The *start* is the time just before it starts the computation to extract features and then run the classifier to identify phishing attack. When the computation to identify the phishing attack ends, the decision time *decision* is captured. Finally, the difference between the *decision* and *start* time is the time it takes to decide whether or not a URL is phished. For a set of forty URLs, the average latency of our tool was 62.5 milliseconds. These experiments were performed on a Windows 10 powered 64 bit Intel ® Core i7 CPU @ 3.40GHz with 8GB RAM.

Clearly, the stateless tools are faster and hence the *PhishCatcher* leads the stateful tools in terms of latency. To experimentally compare the latency of *PhishCatcher* with a stateful tool, we run a tool *SpoofCatch* [1] over a small set of URLs on the the same machine. The experimental

TABLE 6. Comparison between *PhishCatcher* & other anti-phishing tools.

No.	Anti-phishing tool	Result	Machine Learning Used	Reference
1.	SpoofGuard	91%	No	Table 1, [69]
2.	EarthLink	83%	No	Table 1, [69]
3.	Netcraft	77%	No	Table 1, [69]
4.	Cloudmark	68%	No	Table 1, [69]
5.	TrustWatch	49%	No	Table 1, [69]
6.	PhishCatcher	99%	Yes	Equations 1-3
7.	Tool by Ankit et al.	98.4%	Yes	[70]

requirements of the tool *SpoofCatch* and *PhishCatcher* are different. The former requires that a legit URL is opened at least once in the browser before the phished URL is accessed. As with *PhishCatcher*, we added instructions in the source code of the *SpoofCatch* to capture the *start* and *decision* times. The average latency of the *SpoofCatch* was 512 milliseconds and it further raised when the number of experiments were increased. The reason for this latency degrade is that each time the *SpoofCatch* captures a login web page, it stores it in the local storage. As with passage of time, the number of web pages in the local storage increases, it increases the number of comparisons of a current web page with all the previously visited pages stored in the local storage.

VII. CONCLUSION AND FUTURE WORK

Users have become dependent on the online applications as they provide significant quality of service in many domains i.e., online banking, e-commerce, social connectivity, digital libraries, online health services, virtual education, digital marketing and multi-player gaming applications. Commonly, an authentication procedure is followed by the users for the creation of their online account to access the private web content. The security and privacy of users is at stack amid highly sophisticated web spoofing attacks. Several research and commercial tools have been developed to fight against web spoofing attacks but most of them appear with a few lapses. We have developed an optimized user-friendly browser plug-in dubbed as *PhishCatcher* for the smart disclosure of phishing attacks based on supervised machine learning. Contrary to the traditional approaches, our scheme offers to run the classification in the browser itself. It addresses the loopholes in the existing web applications by fixing the latency issues and improving the efficiency of the tool. The user interface of our plug-in is made simple for the better understanding of the user. When a user enters a phished URL, it displays a phishing alert on the screen and highlights the corresponding phishing features of that URL in a drop-down menu.

The feature-set contains thirty features which are categorized into four groups where each group is acknowledged as a decision tree. Random forest classifier employs the aggregated outcome of the decision trees to identify the bogus and genuine login web pages. The data-set for testing and evaluation comprises of 400 malicious and 400 legitimate URLs. The criteria for testing and evaluation is based on a confusion matrix which enlists the true positives, true negatives, false positives and false negatives. Our plug-in displayed remarkable classification results with the precision and recall, both to be 98.5% and accuracy of 98.5%. Furthermore, the average latency of the plug-in was just 62.5 milliseconds which was measured by running it over forty phished URLs.

The feature set contains thirty features, though, the addition of more automated features might be a great idea to improve the overall performance. Some other discriminative classifiers such as SVM can also be implemented for the prediction of fake or real URL by training larger data-sets. Evaluation metrics may also be evolved by using different tools for a better performance analysis.

REFERENCES

[1] W. Khan, A. Ahmad, A. Qamar, M. Kamran, and M. Altaf, "SpoofCatch: A client-side protection tool against phishing attacks," *IT Prof.*, vol. 23, no. 2, pp. 65–74, Mar. 2021.

[2] B. Schneier, "Two-factor authentication: Too little, too late," *Commun. ACM*, vol. 48, no. 4, p. 136, Apr. 2005.

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

[4] R. Oppliger and S. Gajek, "Effective protection against phishing and web spoofing," in *Proc. IFIP Int. Conf. Commun. Multimedia Secur.* Cham, Switzerland: Springer, 2005, pp. 32–41.

[5] T. Pietraszek and C. V. Berghe, "Defending against injection attacks through context-sensitive string evaluation," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Cham, Switzerland: Springer, 2005, pp. 124–145.

[6] M. Johns, B. Braun, M. Schrank, and J. Posegga, "Reliable protection against session fixation attacks," in *Proc. ACM Symp. Appl. Comput.*, 2011, pp. 1531–1537.

[7] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, "Automatic and robust client-side protection for cookie-based sessions," in *Proc. Int. Symp. Eng. Secure Softw. Syst.* Cham, Switzerland: Springer, 2014, pp. 161–178.

[8] A. Herzberg and A. Gbara, "Protecting (even naıve) web users from spoofing and phishing attacks," Cryptol. ePrint Arch., Dept. Comput. Sci. Eng., Univ. Connecticut, Storrs, CT, USA, Tech. Rep. 2004/155, 2004.

[9] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, "Client-side defense against web-based identity theft," in *Proc. NDSS*, 2004, 1–16.

[10] B. Hämmerli and R. Sommer, *Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings*, vol. 4579. Cham, Switzerland: Springer, 2007.

[11] C. Yue and H. Wang, "BogusBiter: A transparent protection against phishing attacks," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 1–31, May 2010.

[12] W. Chu, B. B. Zhu, F. Xue, X. Guan, and Z. Cai, "Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 1990–1994.

[13] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 639–648.

[14] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "An evaluation of machine learning-based methods for detection of phishing sites," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2008, pp. 539–546.

[15] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. 4th Int. Conf. Secur. privacy Commun. Netowrks*, Sep. 2008, pp. 1–6.

[16] W. Zhang, H. Lu, B. Xu, and H. Yang, "Web phishing detection based on page spatial layout similarity," *Informatica*, vol. 37, no. 3, pp. 1–14, 2013.

[17] J. Ni, Y. Cai, G. Tang, and Y. Xie, "Collaborative filtering recommendation algorithm based on TF-IDF and user characteristics," *Appl. Sci.*, vol. 11, no. 20, p. 9554, Oct. 2021.

[18] W. Liu, X. Deng, G. Huang, and A. Y. Fu, "An antiphishing strategy based on visual similarity assessment," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 58–65, Mar. 2006.

[19] A. Rusu and V. Govindaraju, "Visual CAPTCHA with handwritten image analysis," in *Proc. Int. Workshop Human Interact. Proofs*. Berlin, Germany: Springer, 2005, pp. 42–52.

[20] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019.

[21] P. Sornsuwit and S. Jaiyen, "A new hybrid machine learning for cybersecurity threat detection based on adaptive boosting," *Appl. Artif. Intell.*, vol. 33, no. 5, pp. 462–482, Apr. 2019.

[22] S. Kaur and S. Sharma, "Detection of phishing websites using the hybrid approach," *Int. J. Advance Res. Eng. Technol.*, vol. 3, no. 8, pp. 54–57, 2015.

[23] W. W. Cohen, "Fast effective rule induction," in *Machine Learning Proceedings*. Amsterdam, The Netherlands: Elsevier, 1995, pp. 115–123.

[24] V. Muppavarapu, A. Rajendran, and S. K. Vasudevan, "Phishing detection using RDF and random forests," *Int. Arab J. Inf. Technol.*, vol. 15, no. 5, pp. 817–824, 2018.

[25] V. K. Nadar, B. Patel, V. Devmane, and U. Bhave, "Detection of phishing websites using machine learning approach," in *Proc. 2nd Global Conf. Advancement Technol. (GCAT)*. Rajasthan, Jaipur, India: Amity University, Oct. 2021, pp. 1–8.

[26] J. Mao, W. Tian, P. Li, T. Wei, and Z. Liang, "Phishing-alarm: Robust and efficient phishing detection via page component similarity," *IEEE Access*, vol. 5, pp. 17020–17030, 2017.

[27] N. C. R. L. Y. Teraguchi and J. C. Mitchell, "Client-side defense against web-based identity theft," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2004. [Online]. Available: http://crypto.stanford.edu/SpoofGuard/webspoof.pdf

[28] W. Ali, "Phishing website detection based on supervised machine learning with wrapper features selection," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 9, pp. 72–78, 2017.

[29] A. Sharma and D. Upadhyay, "VDBSCAN clustering with map-reduce technique," in *Recent Findings in Intelligent Computing Techniques*. Singapore: Springer, 2018, pp. 305–314.

[30] A. K. Jain and B. B. Gupta, "Comparative analysis of features based machine learning approaches for phishing detection," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 2125–2130.

[31] P. Rao, J. Gyani, and G. Narsimha, "Fake profiles identification in online social networks using machine learning and NLP," *Int. J. Appl. Eng. Res.*, vol. 13, no. 6, pp. 973–4562, 2018.

[32] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 1–28, Sep. 2011.

[33] V. S. Lakshmi and M. S. Vijaya, "Efficient prediction of phishing websites using supervised learning algorithms," *Proc. Eng.*, vol. 30, pp. 798–805, 2012.

[34] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," 2017, *arXiv:1701.07179*.

[35] E. Kremic and A. Subasi, "Performance of random forest and SVM in face recognition," *Int. Arab J. Inf. Technol.*, vol. 13, no. 2, pp. 287–293, 2016.

[36] K. Yu, L. Tan, S. Mumtaz, S. Al-Rubaye, A. Al-Dulaimi, A. K. Bashir, and F. A. Khan, "Securing critical infrastructures: Deep-learning-based threat detection in IIoT," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 76–82, Oct. 2021.

[37] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *Communications and Multimedia Security*. Aveiro, Portugal: Springer, Sep. 2014, pp. 63–72.

[38] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[39] S. Alaparthi and M. Mishra, "Bidirectional encoder representations from transformers (BERT): A sentiment analysis Odyssey," 2020, *arXiv:2007.01127*.

[40] P. A. Barraclough, M. A. Hossain, M. A. Tahir, G. Sexton, and N. Aslam, "Intelligent phishing detection and protection scheme for online transactions," *Exp. Syst. Appl.*, vol. 40, no. 11, pp. 4697–4706, Sep. 2013.

[41] S. Van Acker, D. Hausknecht, and A. Sabelfeld, "Measuring login webpage security," in *Proc. Symp. Appl. Comput.*, Apr. 2017, pp. 1753–1760.

[42] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, Jun. 2009, pp. 681–688.

[43] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 649–656.

[44] M. G. Alkhozae and O. A. Batarfi, "Phishing websites detection based on phishing characteristics in the webpage source code," *Int. J. Inf. Commun. Technol. Res.*, vol. 1, no. 6, pp. 1–9, 2011.

[45] P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Protecting people from phishing: The design and evaluation of an embedded training email system," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Apr. 2007, pp. 905–914.

[46] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in *Proc. 4th ACM workshop Digit. identity Manage.*, Oct. 2008, pp. 51–60.

[47] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb./Mar. 2010.

[48] M. Zouina and B. Outtaj, "A novel lightweight URL phishing detection system using SVM and similarity index," *Human-Centric Comput. Inf. Sci.*, vol. 7, no. 1, p. 17, Dec. 2017.

[49] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun. 2009, pp. 1245–1254.

[50] M. Khonji, Y. Iraqi, and A. Jones, "Lexical URL analysis for discriminating phishing and legitimate websites," in *Proc. 8th Annu. Collaboration, Electron. Messaging, Anti-Abuse Spam Conf.*, Sep. 2011, pp. 109–115.

[51] M. Khonji and Y. Iraqi, "Enhancing phishing e-mail classifiers: A lexical URL analysis approach," *Int. J. Inf. Secur. Res.*, vol. 2, nos. 1–2, p. 40, 2012.

[52] V. P. Reddy, V. Radha, and M. Jindal, "Client side protection from phishing attack," *Int. J. Adv. Eng. Sci. Technol.*, vol. 3, no. 1, pp. 39–45, 2011.

[53] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, "Knowing your enemy: Understanding and detecting malicious web advertising," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 674–686.

[54] Y. Mansour, S. Muthukrishnan, and N. Nisan, "Doubleclick AD exchange auction," 2012, *arXiv:1204.0535*.

[55] S. Bell and P. Komisarczuk, "An analysis of phishing blacklists: Google safe browsing, OpenPhish, and PhishTank," in *Proc. Australas. Comput. Sci. Week Multiconference*, Feb. 2020, pp. 1–11.

[56] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A fast filter for the large-scale detection of malicious web pages," in *Proc. 20th Int. Conf. World wide web*, Mar. 2011, pp. 197–206.

[57] S. Ford. *Wepawet*. (2009). [Online]. Available: http://wepawet.cs.ucsb.edu/index.php

[58] M. Imran, M. H. Durad, F. A. Khan, and H. Abbas, "DAISY: A detection and mitigation system against denial-of-service attacks in software-defined networks," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1933–1944, Jun. 2020.

[59] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[60] A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," in *Proc. Int. Conf. Electr. Comput. Technol. Appl. (ICECTA)*, Nov. 2017, pp. 1–5.

[61] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, "CookiExt: Patching the browser against session hijacking attacks," *J. Comput. Secur.*, vol. 23, no. 4, pp. 509–537, Sep. 2015.

[62] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Phishing websites features," School Comput. Eng., Univ. Huddersfield, West Yorkshire, U.K., Tech. Rep., 2015. [Online]. Available: http://eprints.hud.ac.uk/id/eprint/24330/6/MohammadPhishing14July2015.pdf

[63] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[64] M. Jalalian and M. Dadkhah, "The full story of 90 hijacked journals from August 2011 to June 2015," *Geographica Pannonica*, vol. 19, no. 2, pp. 73–87, 2015.

[65] F. A. Khan and A. Gumaei, "A comparative study of machine learning classifiers for network intrusion detection," in *Artificial Intelligence and Security*. New York, NY, USA: Springer, Jun. 2019, pp. 75–86.

[66] N. Moustafa and J. Slay, "The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems," in *Proc. 4th Int. Workshop Building Anal. Datasets Gathering Exper. Returns Secur. (BADGERS)*, Nov. 2015, pp. 25–31.

[67] Y. T. Ho, C. Wu, M. Yang, T. Chen, and Y. Chang, "Replanting your forest: NVM-friendly bagging strategy for random forest," in *Proc. IEEE Non-Volatile Memory Syst. Appl. Symp. (NVMSA)*, Aug. 2019, pp. 1–6.

[68] G. Sonowal and K. S. Kuppusamy, "PhiDMA—A phishing detection model with multi-filter approach," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 1, pp. 99–112, Jan. 2020.

[69] Y. Zhang, S. Egelman, L. Cranor, and J. Hong, "Phinding phish: Evaluating anti-phishing tools," Carnegie Mellon Univ., 2018, doi: 10.1184/R1/6470321.v1.

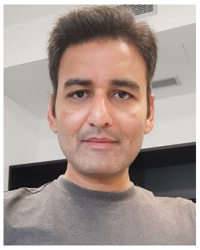
[70] A. K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 5, pp. 2015–2028, May 2019.

MUZAMMIL AHMED is currently pursuing the master's degree with the Department of Electrical and Computer Engineering, COMSATS University Islamabad, Wah Campus, Pakistan. His research interests include information security and machine learning.

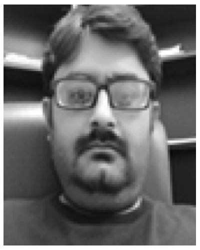
AHMED B. ALTAMIMI received the Ph.D. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada, in 2014. He is currently an Associate Professor with the University of Ha'il, Hail, Saudi Arabia. His research interests include routing and mobility modeling in wireless networks and security and privacy threats in the Internet of Things (IoT) environment.

WILAYAT KHAN received the M.S. degree in IT from Kungliga Tekniska Hogskolan (KTH), Sweden, in 2009, and the Ph.D. degree from the University of Venice, Italy, in 2015. He is an Assistant Professor with COMSATS University Islamabad, Wah Campus, Pakistan. He was an Exchange Researcher with KU Leuven University, Belgium, and a Research Fellow with Nanyang Technological University, Singapore. He has published a number of research articles in reputed journals and conference proceedings. He is the author of the hardware description language *VeriFormal*, chrome extensions *CookiExt* and *SpoofCatch*, and a formal tool *CoCEC*. His research interests include theorem proving, information security, and programming languages design.

MOHAMMAD ALSAFFAR received the B.Sc. degree in computer science from the University of Ha'il, Saudi Arabia, in 2008, the M.Sc. degree in computing from De Montfort University, U.K., in 2011, and the Ph.D. degree in computer science from Brighton University, U.K., in 2019. He is an Assistant Professor with the Information and Computer science Department, Computer Science and Engineering College, University of Ha'il. He was the Vice Dean of Community Service and Continuing Education Deanship, from 2020 to 2022. He returned from the U.K., in 2019, and started to build up his research profile. He seeks to expand his scientific awareness by collaborating with several researchers from several other disciplines to produce various scientific articles in the fields of data science, artificial intelligence, cyber security, and the Internet of Things. He has participated in three group research projects and each group has published several scientific articles in ISI journals. The three group research projects have three different themes or area of research: first one focuses on formal methods and tool support for secure web and cloud computing systems, the second one focuses on IoT based organic waste management systems in Saudi Arabia, and the third project focuses on global optimization by using an algorithm of novel hybrid kidney-inspired. His research interests include user experience, human–computer interaction, the Internet of Things, data science, and cyber security.



AAKASH AHMAD received the Ph.D. degree in software engineering from Dublin City University, Ireland. He is an Assistant Professor with the School of Computing and Communications, Lancaster University Leipzig, Germany. His Ph.D. study was funded by Lero—the Irish Software Engineering Research Center. His research focuses on engineering software-intensive systems and services for mobile and pervasive computing in the context of the Internet of Things.



ZAWAR HUSSAIN KHAN received the M.S. degree in electrical engineering from George Washington University, Washington, DC, USA, in 2009, and the Ph.D. degree in electrical engineering from the University of Victoria, Victoria, BC, Canada, in 2016. He is currently an Adjunct Professor with the Department of Electrical and Computer Engineering, University of Victoria. His research interests include the Internet of Things, traffic analysis, and intelligent transportation systems.

ABDULRAHMAN ALRESHIDI received the M.Sc. degree from the University of Birmingham, Birmingham, U.K., in 2010, and the Ph.D. degree from King's College London, London, U.K., in 2016. He is currently an Associate Professor with the College of Computer Science and Engineering, University of Ha'il, Hail, Saudi Arabia. His research interests include software and system engineering, with a focus on architecture, software patterns, security, and privacy aspects of mobile and the Internet of Things systems.