Routledge
Taylor & Francis Group

Journal of Applied Security Research

ISSN: 1936-1610 (Print) 1936-1629 (Online) Journal homepage: www.tandfonline.com/journals/wasr20

Exquisite Analysis of Popular Machine Learning–Based Phishing Detection Techniques for Cyber Systems

Meenakshi Das, Sowmya Saraswathi, Rashmi Panda, Alekha Kumar Mishra & Asis Kumar Tripathy

To cite this article: Meenakshi Das, Sowmya Saraswathi, Rashmi Panda, Alekha Kumar Mishra & Asis Kumar Tripathy (2021) Exquisite Analysis of Popular Machine Learning–Based Phishing Detection Techniques for Cyber Systems, Journal of Applied Security Research, 16:4, 538-562, DOI: 10.1080/19361610.2020.1816440

To link to this article: https://doi.org/10.1080/19361610.2020.1816440

JOURNAL OF APPLIED SECURITY RESEARCH
2021, VOL. 16, NO. 4, 538–562
https://doi.org/10.1080/19361610.2020.1816440

Routledge
Taylor & Francis Group

Exquisite Analysis of Popular Machine Learning–Based Phishing Detection Techniques for Cyber Systems

Meenakshi Das[a], Sowmya Saraswathi[a], Rashmi Panda[b], Alekha Kumar Mishra[c], and Asis Kumar Tripathy[d]

[a]School of Computer Science and Engineering, VIT Vellore, Vellore, India; [b]Department of ECE, IIIT Ranchi, Ranchi, India; [c]Department of Computer Applications, NIT Jamshedpur, Jamshedpur, India; [d]School of IT and Engineering, VIT Vellore, Vellore, India

ABSTRACT

Recent advances in data science have made available many URL-analysis–based detection and machine learning algorithms, which are being leveraged for phishing detection. It is necessary to measure the efficacy of each technique so as to select an efficient phishing detection technique for integrating into a system. There have been a number of studies stated so far in this related field to assist with this task. While most of the studies have covered all the approaches, only a few have concentrated their study on machine learning–based techniques. Additionally, the parameters such as true-positive, false-positive, true-negative, and false-negative rates have been analyzed in most of the studies in these related articles. In order to achieve new objectives in the field of security, the study on analysis of phishing detection techniques needs to be expanded. In contrast to state-of-the-art methods, our research scrutinizes the different mechanisms and taxonomy used in each machine learning–based detection technique succeeded by an upper bound computing time.

KEYWORDS

Phishing; phishing detection; machine learning and data mining based detection; computational time

INTRODUCTION

The advent of new technology has provided users with a multitude of data that are accessible anywhere and anytime (Afonso et al., 2017; de Jesus Prado et al., 2020). The data can be sensitive information which should be protected from unauthorized access and malicious intent. Ease of access has paved the way for several threats wherein a malicious person can pose as a legitimate owner of the information and gain access to confidential data of the victim to bring financial loss and defamation (Bhatt & Thakker, 2020). One such attack is phishing (Hong, 2012; Kumaraguru et al., 2010). There are three components in a phishing attack (Shi & Saleem, 2012). First, the mailer sends out a huge number of phishing mails with a catchy

CONTACT Asis Kumar Tripathy asistripathy@gmail.com School of IT and Engineering, VIT Vellore, Vellore, India.

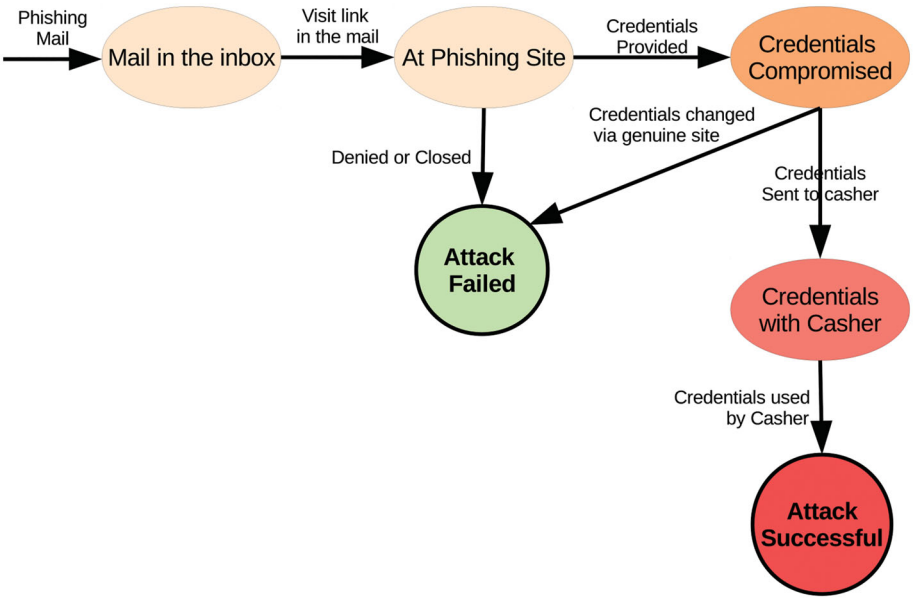

Figure 1. Automata for phishing attack.

subject line and malicious attachments, images, and links using botnets. When a user clicks on these links or attachments in these mails without suspecting its legitimacy, he/she is directed to a fraudulent website called a collector where a user is asked to enter personal information like login credentials and bank and card details. Finally, the casher uses the collected information to achieve a payout. An automaton of the above phishing procedure is depicted in Figure 1. In the last few years, phishers are using malware to freeze the victim's system and, in return, demand a huge ransom to unfreeze the system (Komatwar & Kokare, 2020). A phishing attack has some typical characteristics (Edith & Chandrasekar, 2014). These are as follows:

- Phishing mails create a sense of urgency in the recipient, prompting the user to open the mail.
- The most popular operating system is the Windows OS. Hence, it is the number one attack target.
- Phishing attacks have evolved from sending attachments or links via mail to privilege escalation, where an attacker gains elevated access to resources that are normally secured.

A detection technique can use blacklisting, machine mearning (ML), or heuristic-based approaches. A generic phishing detection of the recent trend is shown in Figure 2. The hardest part is how to detect a new phishing mail or its site. This is due to flexibility available with phishing

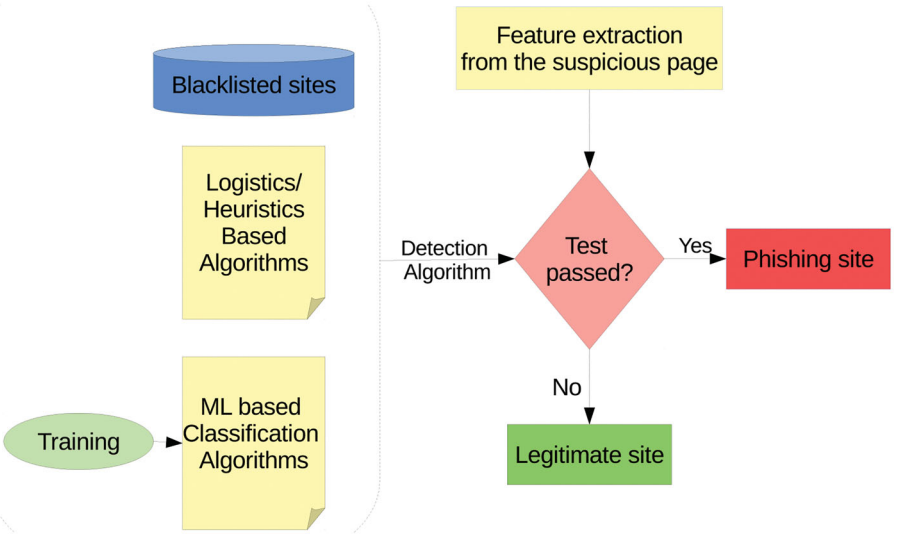

Figure 2. A generic phishing detection process.

compared to other cyberthreats, and phishers are launching new and unique phishing domains each day. Numerous attempts at phishing detection have been made, and these techniques are classified based on the procedure and information used for phishing detection.

In this article, we present a survey and analysis of ML-based phishing detection mechanisms. We have selected popular ML-based phishing detection papers for our study. The popularity is considered in terms of mostly referred mechanisms in the articles of the cybersecurity field. The novel contributions of this survey compared to existing works are listed as follows:

- The overview of a mechanism supported with a pseudocode depicting the overall detection procedure to improve clarity
- The computation and comparative analysis of computational time for training and detection procedures
- The analysis of most and least used ML techniques and number of ML techniques considered by each mechanism for selecting the best one
- Analysis of features used in each mechanism and their relevancy for efficiently classifying a phishing page from a legitimate one
- The result strength analysis by comparing the true-positive rate (TPR) and the data set sample size used for training and testing purposes

The rest of the article is organized as follows. The next section provides a survey of important contributions toward phishing detection techniques. Then is a discussion and analysis on recent ML-based detection techniques, followed by a conclusion.

LITERATURE SURVEY

Numerous studies have been involved in interpreting and assessing the methods of identification of the phishing attack. The state of the art comprises Khonji et al. (2013) reviewed in the year 2013, Dou et al. (2017) in the year 2017, and Varshney et al. (2016) in the year 2016. A study described by Chiew et al. (2018) on various alternatives and classifications of phishing attacks also specifies the features of these attacks. This article focuses on ML-based techniques, which are abundantly used in the recent literature. The first and foremost unique contribution of this article is the pseudocode representation of all the detection mechanisms, which is not reported in any of the works so far. The intention of this pseudocode is to ease the task of understanding the procedure and implementing the same. Second, the upper bound of computational time of each detection procedure supports the selection of a suitable mechanism. Third, analysis of the strength of the detection procedure compares the number of ML techniques considered to select the optimum one and the sample size for the training and testing data set.

ML-BASED PHISHING DETECTION TECHNIQUES

In this section, phishing detection techniques based on ML are explored and analyzed.

Content-based phishing email detection

Che et al. (2017) proposed a novel algorithm to detect phishing emails based on event pairs. A semantic web database is built that produces the association between an event (features in the email content is referred to) and words. A category database is built to categorize the phishing emails. The detection is done using a semantic web database and category web database. The use of events for the detection of phishing emails, specification on precision of categorizing emails, and the association between events and words have been described. The pseudocode of the detection process is provided in Algorithm 1. The first function SemanticWeb creates the semantic web database by adding keyword-based events to it. The GenEventPairs function generates event and keyword pairs for each mail. Finally, the Detection function detects whether its a phishing mail or not. It applies fuzzy control to find the highest matching ratio of the input email with category database. It is reported that the fuzzy control mechanism has produced accurate results for phishing mails.

Algorithm 1: Content-Based Phishing Email Detection

```
Input: Phishing-emails
Output: phishing or legitimate email
Function SemanticWeb(phishing_email):
    input = Collect_Sample(phishing − email)
      Keywords_List = Extract_Keywords(input) if
      Keywords_List.length < 1 × 10^5 then
        collect next sample as input
    else
        Keyword_Types=Classify_Keywords
        (Keywords_List)
        EventName = names in Keyword_Types
        Add EventName to SemanticWeb
    return SemanticWeb

Function
  GenEventPairs(phish_email,SemanticWeb):
    input = phishing-emails
    Keywords_List = Extract_Keywords(input)
    (keyword,event)=Match(Keywords_List,
      SemanticWeb)
    add (keyword,event) to EventPairs
    return EventPairs

Function
  Detection(phish_email, EventPairs):
    for each event-pair in EventPairs do
        if event-pair < 100 then
            Add phish_email to the No-phishing
              catetory
            detresult="legitimate"
        else if phish_email.length > 10^5 then
            fuzzy-control = Fuzzy_Control_Match
              (phish_email,Category_DB)
            detresult = fuzzy-control
    return Detection
```

Phishing detection using DBSCAN clustering

Liu et al. (2010) have proposed a detection mechanism that matches a popular brand's webpage with a suspicious one using the DBSCAN clustering technique. Webpages with higher similarity are blacklisted. The mechanism uses the help of a search engine to identify similar webpages along with the rank. This network communication overhead may be considered

as a weakness while assuming networks with constrained resources. The pseudocode of the detection process is provided in Algorithm 2. The function processURL generates a score vector of link strength, page rank, page similarity, and layout similarity. The detection function uses this vector to cluster the URL pages using DBSCAN. If a cluster is found, then the URL is detected as a phishing page and if the URL does not belong to any cluster, then its considered as legitimate.

Algorithm 2: Phishing detection using DBSCAN clustering

```
Input: url
Output: phishing or legitimate url
Function ProcessURL(url):
    if url found in blacklist then
        detresult = "phishing"
    else
        url-pages=Gather_RelPages(url)
        for each page in url-pages do
            s=Link_Strength(page)
            r=Page_Rank(page)
            t=Text_Similarity(page)
            l=Layout_Similarity(page)
            ScoreVector = (s,r,t,l)

Function Detection(ScoreVector,url):
    Clusters=DBSCAN(ScoreVector)
    if a cluster found for url in Clusters then
        detresult="phishing"
        add url to the blacklist
    else
        detresult="legitimate"
    return detresult
```

DNS-poisoning–based phishing detection

Kim and Hun (Kim & Huh, 2011) have implemented a detection algorithm using features of network efficacy of the websites. Routing information is obtained and mean and standard deviation of round-trip time (RTT) and total length of route in each hop are calculated. Algorithm 3 shows the pseudocode of this approach. The function DataSet prepares data such as mean, standard deviation of RTT, and length of routes. Four ML-based classification algorithms are tested for the training data set. There are Linear Discriminant Analysis (LDA), Naive Bayes, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). The metrics used for evaluations are accuracy, specificity, and sensitivity. The

results shown that KNN performed the best on basis of accuracy, specificity, and sensitivity. TrainClassifier trains the data set using KNN classifier and, finally, the Detection function detects the legitimacy of the URL.

Algorithm 3: DNS-poisoning based phishing detection

```
Input: url of the website
Output: url classification results
Function DataSet(url):
    routing-info = GetRoutingInfo(url)
    mean = Mean(routing-info.RTT)
    sd = StdDev(routing-info.RTT)
    length = routing-info.route_len
    add (mean, sd, length) to data
    training-data= 2/3 of data
    testing-data = 1/3 of data
    DataSet = (training-data,testing-data)
    return DataSet

Function TrainClassifier(DataSet):
    classifier= KNN-classifier(DataSet)
    return classifier

Function Detection(classifier,url):
    detresult=Detect(url,classifier)
    return detresult
```

Two-level filtering–based phishing detection

Srinivasa et al. (Rao & Pais, 2019) have proposed a two-level phishing detection algorithm in which a blacklist approach is used as a first-level filter for the detection of near-duplicate phishing sites and a heuristic mechanism as a second-level filter. It has been observed that using the first-level filter 55.58% of the phishing sites were detected. By using the second-level filter, the accuracy achieved was 98.72%. The pseudocode of the detection process is shown in Algorithm 4. The BlacklistFilter function checks for already blacklisted URLs. The HeuristicCheckFilter first extracts DOM features and generates a feature vector. Then, it applies the Ensemble function with a combination of XGBoost, Extra-Tree (ET), and Random Forest (RF) techniques to generate the heuristic filter result of the URL.

Algorithm 4: Two-level filtering based phishing detection

```
Input: url of the website
Output: phishing or legitimate url
Function BlacklistFilter(url):
    Feature-Set= Extract_Features(url)
    hash=Simhash(Feature-Set)
    if hash found in blacklist then
        detresult = "phishing"
    else
        detresult =
          HeuristicCheckFilter(url,Feature-Set)
    return detresult

Function
  HeuristicCheckFilter(url,Feature − Set):
    DOMFeatures = Extract_DOM(url)
    FeatureVector=
      GenFeatureVector(Feature-set,DOMFeatures)
    checkresult =
      Ensemble(XGBoost(FeatureVector),Extra-
      Tree(FeatureVector),RandomForest(FeatureVector))
      return checkresult
```

Phishing detection using feature vector of webpage

Feroz and Mengel (Feroz & Mengel, 2014) have classified the address of the websites using lexical and host-based features. These features are analyzed to deduce the legitimacy of a URL. The features of the URL are extracted using a special Java application. Algorithm 5 shows the pseudocode of this approach. The feature vector changes the value of raw features to significant feature vectors. In this contribution, seven ML schemes are used to check the presence/absence of information gain. The list of ML schemes used are j48-decision tree (j48-DT), RF, Naive Bayes, Bayesian Network (BN), BFTree, and Logistic Regression (LR). Among all the techniques, LR performed better with respect to accuracy and false-positive rate. The HostBased function generates the DNS hierarchy and the feature vector. After that, the feature selection function is used to determine the selected features based on the chi-res and information gain. Then, the TrainClassifier function classifies the values based on the selected features. Finally, the Detection function detects the phishing attack depending upon the classification values.

Algorithm 5: Phishing detection using feature vector.

```
Input: url-page
Output: the legitimacy of the page
Function LexicalFeatures(url − page):
    tokens = Extract_URLTokens(url-page)
    if occurrences(tokens,"www") > 1 then
        binary = 1 /* phishing site */
    else
        binary = 0 /* legitimate site */
    continuous-valued= (url-page.length,
     url-page.noOfDots)
    lexfeatures = (binary, continuous-valued)
    return lexfeatures

Function HostBased(url − page):
    java-app = HostBasedFeaturesJava(url-page)
    DNS-hier = resolve URL and acquire DNS
     hierarchy records
    FeatureVector = GenFeatureVector(DNS-hier)
    Add FeatureVector to HostBasedFeatures
    return HostBasedFeatures

Function
 FeatureSelection(HostBasedFeatures):
    chi-res = chi-squares-test(HostBasedFeatures)
    gain= Information_Gain(HostBasedFeatures)
    SelectedFeatures = (chi-res,gain)
    return SelectedFeatures

Function
 TrainClassifier(SelectedFeatures):
    LRclassifier = LogisticRegression(
     SelectedFeatures)
    return LRclassifier

Function Detection(LRclassifier,url):
    detresult=Detect(url,LRclassifier)
    return detresult
```

PhishStorm

Marchal et al. (2014) proposed a URL-based real-time phishing detection system that can be interfaced with an email server or HTTP proxy. It has been observed that the intra-URL relatedness builds the relation between registered domain name. The rest of the URL is absent in phishing sites. PhishStorm used this concept to detect phishing pages. The pseudocode of the PhishStorm is provided in Algorithm 6. A list of seven ML techniques are used for training the feature set. They are Random Tree, RF, C4.5,

LMT, PART, JRip, and SVM. The performance is evaluated using accuracy, true-positive rate, and true-negative rate. It is observed that tree-based classifiers are the best performers for this problem statement. Therefore, RF was selected for the purpose of classification. The IntraURLRelatedness function extracts the related words and associated words to find out the relatedness among them. Then, the TrainClassifier function classifies the relatedness features by using the RF mechanism. Finally, the Classify function produces the detection result.

Algorithm 6: PhishStorm.

Input: url-page
Output: the legitimacy of url-page

```
1  Function
     IntraURLRelatedness(url − page):
2      Breakdown the URL into RDurl and REMurl
3      ASRDurl,ASREMurl =
         Extract_AssociatedWords( RDurl,REMurl)
4      RELRDurl,RELREMurl =
         Extract_RelatedWords( RDurl,REMurl)
5      Relatedness= ((ASRDurl ∪ ASREMurl ∪
         RELRDurl ∪ RELREMurl) , (ASRDurl ∩
         ASREMurl ∩ RELRDurl ∩ RELREMurl))
6      return Relatedness
7
8  Function
     TrainClassifier(RelatednessFeatures):
9      RFclassifier =
         Random-Forest(RelatednessFeatures)
10     return RFclassifier
11
12 Function Classify(RFclassifier,url):
13     detresult=Classify(url,RFclassifier)
14     return detresult
15
```

Binary classification algorithm for phishing detection

Mohammad et al. (2014) have proposed an algorithm that uses artificial neural networks (ANNs) for classification of legitimate or phishing sites. Here, the network is automatically self-structured. A list of features such as IP, SSL, and domain age are extracted for training purposes. These features represented the input neurons of the ANN. The pseudocode of the algorithm is presented in Algorithm 7. Initially, the data are collected from different sources such as PhishTank (available at PhishTank, 2020a) and Phishbank (available at Phishbank, 2020b), and the features are extracted. Next, the Network function calculates the error rate based on the trained

data set. Finally, the Detection function is able to detect the phishing attacks.

Algorithm 7: Binary classification for phishing detection.

Input: url-database
Output: the classification results

```
Function DataCollection(PhishTank, Millersmiles):
    DataCollection = collect websites and extract 17 features
    return DataCollection

Function Network(PhishTank, Millersmiles):
    collect websites from PhishTank and Millersmiles
    data = Extract_Listed_Features(websites)
    network =3-ANN()
    trained-net = train(network)
    CER = calc_error(trained-data)
    if CER < DER then
        DER=CER
        trained-net = train(trained-net) /* train again*/
    else
        if DER, no of epochs achieved max values then
            trained-net = train(trained-net) /* train again*/
        else
            learning_rate_new = learning_rate_old*0.9
            add neurons to input
            repeat
                trained-net = train(trained-net)
                new-DER = DER
            until DER, no of epochs not reached max values
        /* Print network cannot predict */
    return trained-net

Function Detection(trained − net,url):
    detresult=Detection(url,trained-net)
    return detresult
```

Phishing detection using URL feature extraction and classification

Aydin and Baykal (2015) designed an unsophisticated method of phishing detection based on the URL of the website. Selected features from the URL are extracted and analyzed to conclude the legitimacy of the site.

Correlation-based Feature Selection (CFS) is used to reduce the dimension of the data. The classification accuracy of CFS features was tested using the Naive-Bayes and Sequential-Minimal-Optimization (SMO) techniques for SVM. The results indicated that SMO for SVM (SMO-SVM) has better accuracy compared to Naive-Bayes. Algorithm 8 shows the pseudocode of the proposed method. The FeatureSelection function uses the top-band URLs to determine the correlation-based features. Next, the TrainClassifier function uses the SMO-SVM classifier to classify the optimized data set. Finally, the Detection function publishes the detection result.

Algorithm 8: Phishing detection using URL feature extraction and classification.

```
Input: url-database
Output: classification accuracy of top-brand urls
Function FeatureSelection(PhishTank):
    top-brands = the top targeted brand names
    top-brand-urls=getPhishingURLs(top-
      brands,PhishTank)
    Features=Extract_Features(top-brand-urls)
    FeatureMat =
      Compose_FeatureMatrix(Features)
    Feature-Set-CFS =
      Correlation_Based_Features(FeatureMat)
    Feature-Set-CS =
      Consistency_Features(FeatureMat)
    Feature-Final = (Feature-Set-CFS,
      Feature-Set-CS)
    return Feature-Final

Function
  TrainClassifier(Feature − Final):
    SMOclassifier = Sequential-Minimal-
      Optimization(Feature-Set-CFS,Feature-Set-CS)
    return (NBclassifier,SMOclassifier)

Function Detection(SMOclassifier,url):
    detresult=Detection(url,SMOclassifier)
    return detresult
```

New rule-based phishing detection

Moghimi and Varjani (2016) have designed a new rule-based method that uses two unique feature sets and analyzes the features to conclude the legitimacy of the site. They have implemented a support vector

machine approach and extracted the rules from it. The rules were placed in the browser extensions for further classification of the webpage into phishing or legitimate sites. The pseudocode of the method is shown in Algorithm 9. The relevant features are extracted in the function ExtractFeatures. Then, the Classify function classifies the featured set by using the SVM. Finally, the legitimate URLs can be detected by the Detection function. Rules obtained from the SVM model have achieved the highest accuracy and lower error rates compared to other phishing techniques.

Algorithm 9: New rule based phishing detection

```
Input: url-webpage
Output: legitimacy of url-webpage
Function ExtractFeatures(webpage − url):
    RelevantFeatures =
     (webpage-url.ipaddr,webpage-url.url_length,
     webpage-url.noOfDots,
     webpage-url.blacklist_keywords)
    Feature-1 = normalize(levenshtein
     distance(webpage-url))
    FirstFeaset = any four features from Feature-1
    Feature-2 =
     (webpage-url.links,webpage-url.jsfiles,
     webpage-url.stylesheets, webpage-url.images)
    SecondFeaset = four accessible features from
     Feature-2
    Add
     (RelevantFeatures,FirstFeaset,SecondFeaset) to
     FeatureSet
    return FeatureSet

Function Classify(FeatureSet):
    classifier =
     SVM(RelevantFeatures,FirstFeaset,SecondFeaset)

    return classifier

Function Detection(classifier,url):
    detresult = Detect(url,classifier)
    return detresult
```

Know your phish

The technique proposed by Marchal et al. (2016) is based on two inferences: (1) There is a limitation to mimic a legitimate website. (2) Phishing webpages do not have consistency in the usage of keywords. The proposed system uses fewer features by filtering them based on generic, adaptable,

and usable properties to achieve higher accuracy in phishing detection by using these two inferences. Algorithm 10 displays the pseudocode of the proposed mechanism. The Scraper function extracts the required features from the URL data set, and the Classifier function classifies the feature vectors by using the learned threshold generated by the Gradient Boosting (GB) mechanism.

Algorithm 10: Know your Phish

```
Input: url of the website
Output: phishing or legitimate url
Function Scraper(url):
    data = Extract_Data(url)
    scraper = (data.json-data,screenshots)
    return scraper

Function Classifier(scraper):
    extractor = GenFeatureVector(scraper)
    limit = 0.7 // the threshold of the classifier
    threshold = GradientBoosting(extractor)
    if 0 ≤ threshold < limit then
        detresult = "legitimate"
    else if limit ≤ threshold < 1 then
        detresult = "phishing"
    return detresult
```

Email phishing detection using data mining

Senturk et al. (2017) have proposed a mechanism that prevents phishing attacks in email using data-mining techniques. The pseudocode of the proposed method is given in Algorithm 11. In the first step, features such as URL text, the form, and presence of $ sign and other keywords are extracted and transformed to discrete values. Then, the association rules are generated. All the above tasks are done in function FsConvert. The WEKA classification success rate is 89%. The ML techniques such as ANN, K-Means, and Naive-Bayes are tested to choose the best classifier. Among all tested techniques, the decision tree–based WEKA algorithm performs better than others and hence is used as the classifier in this method.

Algorithm 11: E-mail Phishing detection using data mining

```
Input: url of the website
Output: legitimate or phishing website
Function FsConvert(url):
    obtain the files with .eml extensions as
      email-data
    features = Extract_Features(email-data)
    convrules = ConvertToCAR(features)
    return convrules

Function Classifier(convrules):
    wekaclassifier = WEKA(convrules)
    return wekaclassifier

Function Detection(url):
    detresult=Detect(url,wekaclassifier)
    return detresult
```

URL-based phishing detection using NLP

Sahingoz et al. (2019) have proposed a phishing classification algorithm that uses NLP-based features. The features extracted are analyzed using ML approaches to detect legitimate or phishing sites. Algorithm 12 is used to showcase the pseudocode of the proposed mechanism. The classifiers used to test the accuracy are Decision Tree (DT), Adaboost, K-star, KNN, RF, SMO-SVM, and Naive-Bayes. Among all the approaches, RF achieves the highest accuracy. First of all, the collected data are pre-processed by the DataPreProcess function. The required features are extracted in this function. Then the word-vector is filtered further by using the DimReduction function. Next, the features are classified by using the RFclassifier. Finally, the legitimate URLs are detected by using the Detection function.

Algorithm 12: URL based Phishing detection using NLP.

```
Input: url,phishtank
Output: legitimate or phishing url
Function DataPreProcess(url):
    tokens = ParseURL(url)
    if tokens has brand-name then
        increment the brand-name-count
        increment the keywords-count
        random-word = detect the word
         count.Check for length is > 7 then add to
         word list to be analyzed
    else
        if tokens has random words then
            increment random-word count
        else
            if token.length > 7 then
                Decompose(token)
            add words to the to be analysed list
            anlysis-data=MaliciousAnalysis(tokens)
    features = ExtractFeatures(brand-name-
     count,keywords-count,random-word
     count,anlysis-data)
    return features

Function WordVector(url):
    word-vector =
     ConvertToWordVector_WEKA(url)
    word-vector1 = DimReduction(word-vector)
    return word-vector1

Function
 Classifier(features, word − vector1):
    hybrid-features = features + word-vector1
    RFclassifier = Random Forest(features,
     word-vector1, hybrid-features)
    return RFClassifier
Function Detection(RFClassifier,url):
    detresult=Detect(url,RFClassifier)
    return detresult
```

Keyword-based phishing detection

Ding et al. (2019) have proposed a method which is the aggregation of search heuristic and logistic regression. This method filters webpages and further classifies them using logistic regression to determine the legitimacy of the sites. The pseudocode of the proposed mechanism is shown in Algorithm 13. Extract the features from the URL such as DNS, HTML tags,

and phishing similarity by using the SearchEngine function. Then the extracted features are classified by using the HeuristicRule function and train the classifier. To train the classifier, Naive-Bayes, LR, and SVM are used. It is reported that LR has outperformed others using heuristic and search engine and achieves the highest accuracy of 98.9%.

Algorithm 13: Keyword-based Phishing Detection

```
Input: url
Output: legitimacy of the url
Function SearchEngine(url):
    keyword = Extract_TitleTag(url.page)
    use search engine to find match for keyword
    if keyword does not match with the original site
     then
        Apply Heuristic-rule-based detection
        HeuristicRule(url)
    else
        detresult = "legitimate"
    return

Function HeuristicRule(url):
    rules = Define_Heuristics()
    features = Extract_Features(url)
    if features satisfies the rules then
        detresult = "phishing"
    else
        Classify(features)
    return

Function TrainClassifier(features):
    classifier = logistic-regression(features,rules)
    return classifier

Function Classify(features):
    detresult=Detect(features,classifier)
    return detresult
```

ANALYSIS

Analysis based on the use of ML techniques

In this section, analysis of the detection mechanisms is performed considering the number of types of ML technique used. Figure 3 shows the comparison of the number of ML techniques used to select a classifier by the mechanisms. It is observed that the detection mechanism by Sahingoz et al. (2019) and PhishStorm (Marchal et al., 2014) have used a maximum of seven ML techniques to select the best classifier. The mechanism by Feroz

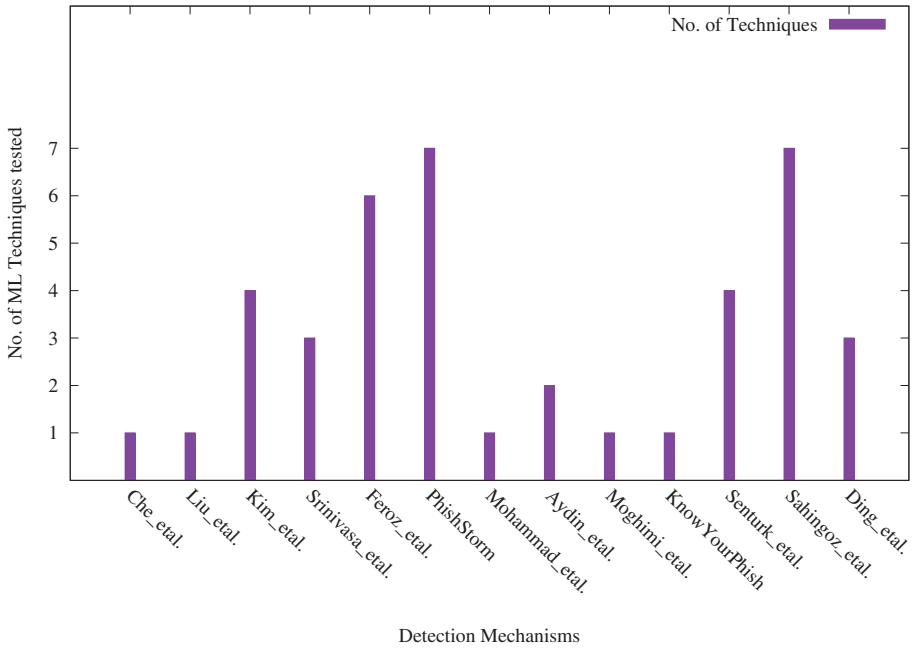

Figure 3. The comparison of number of ML-based techniques used for selecting classifiers.

Table 1. Summary of ML techniques for phishing detection.

Article	Tested Techniques	The Outperforming Technique
[14]	Fuzzy Control	Fuzzy Control
[15]	DBSCAN Clustering	DBSCAN Clustering
[16]	LDA, Naive-Bayes, KNN, SVM	KNN
[17]	XGBoost, ET, RF	RF
[18]	j48-DT, RF, Naive-Bayes, BN, BFTree, LR	LR
[19]	Random Tree, RF, C4.5, LMT, PART, JRip, SVM	RF
[20]	ANN	ANN
[23]	Naive-Bayes, SMO-SVM	SMO-SVM
[24]	SVM	SVM
[25]	GB	GB
[26]	ANN, Naive-Bayes, K-Means, WEKA	WEKA
[27]	DT, Adaboost, K-star, KNN, RF, SMO-SVM, Naive-Bayes	RF
[28]	Naive-Bayes, LR, SVM	LR

and Mengel (2014) uses six ML techniques. It can be inferred that these techniques claimed that accuracy is promising compared to others as they have explored sufficient techniques to choose an optimized one. Table 1 shows the summary of ML techniques used by detection mechanisms for testing their classifiers' accuracy and the outperforming technique among them. It can be observed from the second column of the table that RF and LR techniques have proven to be better than others in cases in which multiple detection techniques are used. This is due to the ability of these mechanisms to classify optimally using fewer features compared to other techniques. However, when it comes to the most frequently used ML technique in these mechanisms for training a classifier, as shown in Figure 4,

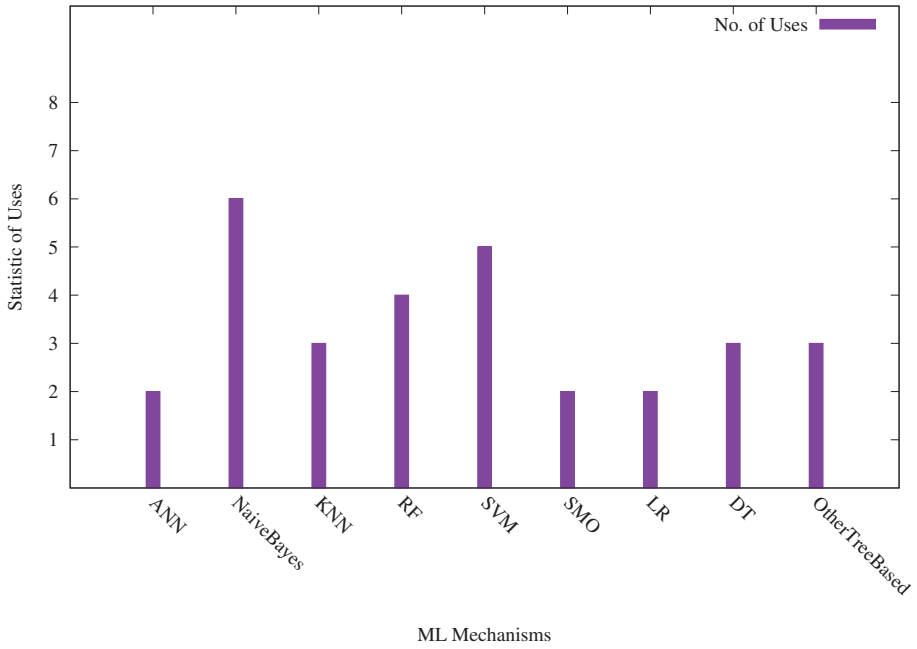

Figure 4. Statistics of ML-based techniques used more than once for testing classifiers in the survey articles.

Naive-Bayes mechanism is the first choice followed by SVM. This is due to the simplicity of Naive-Bayes to model any classification problem. However, the SVM is also a preferred technique due to its ability to optimally classify the nonlinear feature vectors with an affordable computation and storage requirement.

Analysis of feature selection

Table 2 summarizes the list of features used by each mechanism for training their classifier. It is observed from the table that most of the URL analysis–based mechanisms such as in Marchal et al. (2014) and Feroz and Mengel (2014) use URL features such as its length, presences of keywords, number of dots, presence of special characters, URL domain, and digits. The mechanisms that are based on URL page analysis such as Kim and Huh (2011) and Marchal et al. (2016) use DOM tree structure, tags present in the page, HREF links, and many more. The detection mechanism that relies on network behavior of a URL request such as Rao and Pais (2019) and PhishTank (2020a) uses network features such as IP address, DNS record, and web traffic. It is also observed from the table that most of the mechanism combines the features of all the above categories to achieve accurate classification.

Table 2. Summary of features used for training phishing classifiers.

Article	Features
[14]	Keywords in URL
[15]	Page rank, ratio of the weighted number of matched blocks to the total number of blocks in a page and associated pages.
[16]	Mean and standard deviation of RTT, total length of route of each hop, IP datagram packet fields
[17]	URL length, presence of special characters, digit count, presence of domain name in title, common ratio, foreign links ratio
[18]	URL tokens, length of URL, number of dots in URL, Tags:(A,CNAME,MX,NS), IP address, AS (autonomous system) numbers, and BGP routing prefix
[19]	Registered domain, similarity index of URL
[20]	IP address, URL length, URL domain, @ symbol, prefix and suffix to URL, subdomain, HTTPS, server form handler, redirect URL, pop-up window, suspicious link, DNS record, web traffic, age of domain, and right click disabled
[23]	Textual properties of the URL
[24]	IP address, SSL certificate, number of dots, URL length, blacklist keywords, DOM, page content, HTTPS, and page resource identity
[25]	Protocol, URL word count, level domains count, HREF links, number of input fields, and images
[26]	Keywords in URL, and presence of $ symbol
[27]	Words with random characters, brand name and keywords, special characters, and number of dots
[28]	Keywords in URL, logo image, IP as domain name, phishing target words hidden, DNS doubt, HTML, similarities in vocabulary, Whois

Table 3. Symbols.

Symbol	Meaning
N	Data set size
f	Number of features (keywords)
tree	Number of trees (for tree based ML techniques)
sv	Number of support vectors (for SVM)
neurons	Number of neurons (for ANN)
urlAttrib	Number of URL attributes
noDOM	Size of DOM objects in a webpage

Computational time analysis

The symbols used in this section are listed in Table 3.

The computation time of Liu et al. (2010) depends on the DBSCAN clustering. Assuming that DBSCAN clustering is implemented with indexing, it requires $O(N \log N)$ computational time.

The mechanism proposed by Kim and Huh (2011) uses KNN for detection of a phishing URL. The KNN needs $O(N.nf)$ and $O(nf)$ time for training and classifying, respectively (Computational Complexity of Machine Learning Algorithm, 2018).

Feroz and Mengel (2014) have shown that LR achieves high accuracy with a low false-positive rate. The time required by LR for the training classifier and detection are $O(nf)$ and $O(nf^2.N + N^3)$, respectively.

Mohammad et al. used ANN for phishing detection. The time required by ANN for the training and detection are $O(f.N.(neurons_1.neurons_2 + \ldots +$

$neurons_{m-1}.neurons_m))$ and $O(f.(neurons_1.neurons_2 + \ldots + neurons_{m-1}.neurons_m))$, respectively, where $neurons_i$ represents number of neurons added at layer i (Computational Complexity of Machine Learning Algorithm, 2018; Computational Complexity of Training Neural Networks Using Back Propagation, 2018).

The phishing detection mechanism in Rao and Pais (2019), Sahingoz et al. (2019), and Marchal et al. (2014) uses RF Classifier. Therefore, the training and detection computational time are $O(N^2.f.trees)$ and $O(urlAttrib) + O(f.tree)$, respectively.

SMO for SVM is used in Aydin and Baykal (2015). Time needed for training the classifier is $O(N^2.f + N^2)$, whereas the detection process would need $O(urlAttrib)$+.., respectively.

The phishing detection mechanism by Moghimi et al. involves SVM for classifying phishing pages. The computational time need by Moghimi and Varjani (2016) for training and detection processes are $O(N^2.f + N^3)$ and $O(noDOM)+O(sv.f)$, respectively.

The mechanism in Marchal et al. (2016) uses Gradient Boosting for phishing detection. It needs $O(N.f.trees)$ computational time for training and $O(urlAttrib)+O(f.trees)$ time for detection, respectively.

Finally, the WEKA classification algorithm used by Sentürk et al. (2017) requires $O(N^2.f)$ for training the classifier and $O(urlAttrib)+O(f)$ time for detection, respectively.

Performance analysis

In this section, we use three metrics for evaluating performance of the detection mechanism as reported in the literature. These are sample size ratio for experiment, TPR, and false-positive rate (FPR).

A detailed sample size analysis of the considered mechanisms is shown in Figure 5. The analysis shows that the PhishStorm (Marchal et al., 2014) and the mechanism by Sahingoz et al. (2019) have used large sample sizes compared to any other techniques. However, Mohammad et al. (2014) have used the minimum amount data samples for their experiments. For comparison of TPR and FPR, both TPR and FPR values are normalized to the range [0,1] for convenience. However, sample size may vary from technique to technique. Figure 6 depicts the comparison of TPR for phishing detection mechanisms. It can be observed from the figure that almost all mechanisms are able to achieve detection accuracy close to 100%. The phishing detection approach by Ding et al. claimed to have the highest TPR with 0.99, whereas Know Your Phish and the mechanism by Senturk et al. have the lowest, with 0.90 each. It can be observed from Figures 5 and 6 that despite the TPR of PhishStorm (Marchal et al., 2014), and

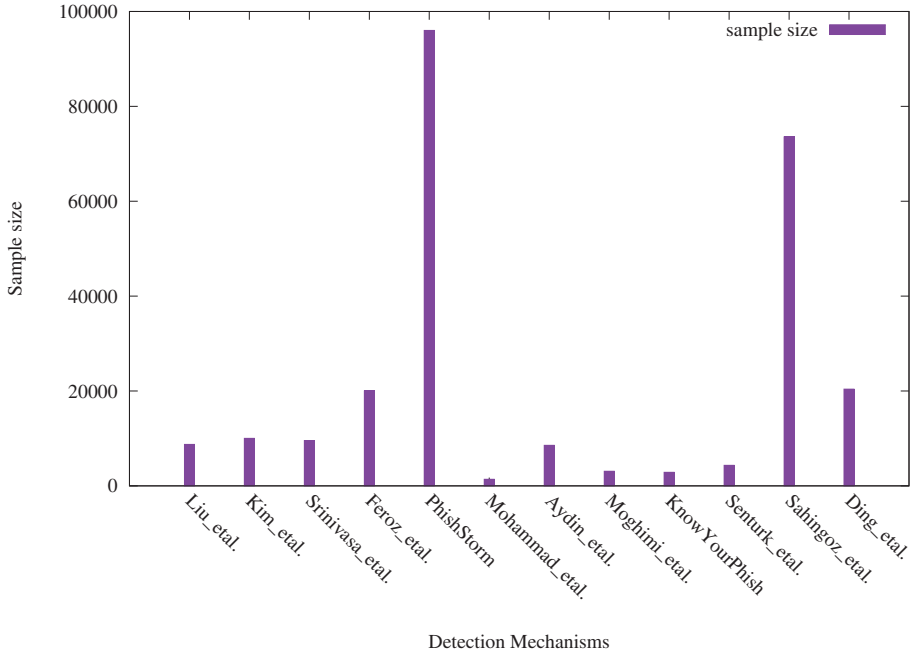

Figure 5. Comparison of sample size used for training and validation by various detection mechanisms.

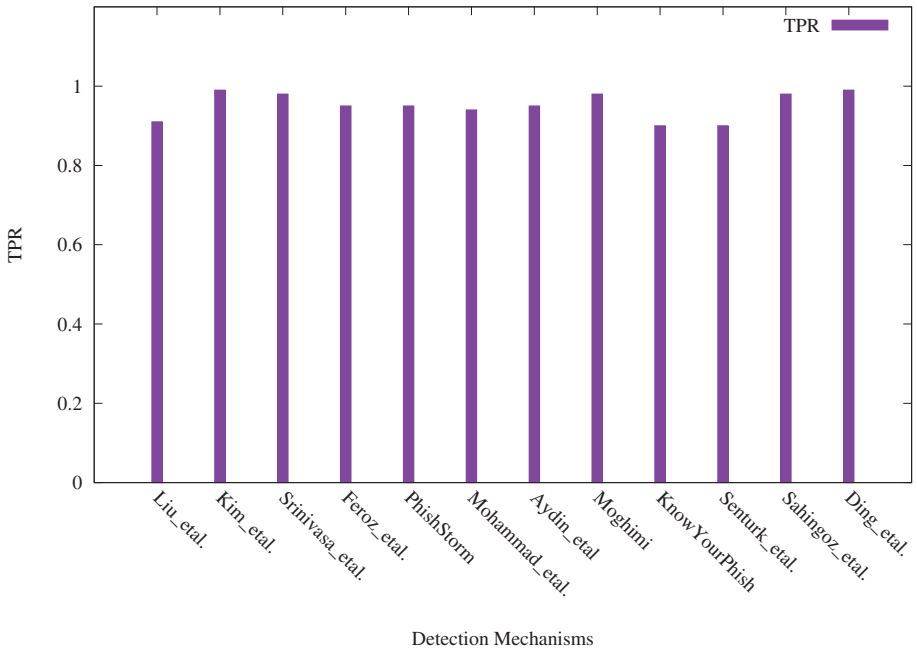

Figure 6. True-positive rate.

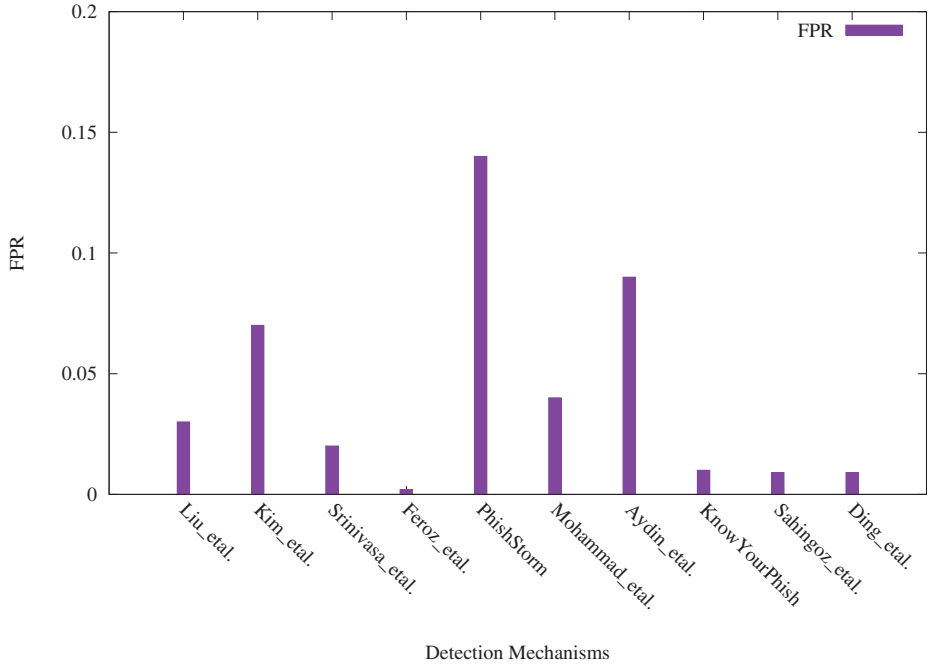

Figure 7. False-positive rate.

mechanism by Sahingoz et al. (2019) are marginally equal to Moghimi and Varjani (2016), Mohammad et al. (2014), Rao and Pais (2019), and Aydin and Baykal (2015). However, PhishStorm's (Marchal et al., 2014) and Sahingoz et al.'s (2019) mechanism accuracy is more promising and has more strength compared to others due to the sample range used for training and validating purposes compared to later ones. The comparison of FPR between the detection mechanism is shown in Figure 7. Both Ding et al. and Sahingoz et al. were found to have the lowest FPR with values of 0.009 each. However, PhishStorm incurs the highest FPR among all the compared mechanisms.

CONCLUSION AND FUTURE DIRECTIONS

In this article, we have discussed and analyzed various phishing detection techniques that use ML-based training and classification approaches. Phishing detection based on the ML approach is considered the recent trend in the state of the art. We have appended each detection mechanism with its pseudocode to improve readability. An exhaustive analysis is provided on the ML techniques used for training phishing attack classifiers, the feature and performance analysis, and computational time analysis of each detection mechanism. Even though the ML-based techniques have an additional computational complexity for training the classification model,

most of them have a detection accuracy nearly equal to 1. We have also analyzed the number of samples taken by each of the algorithms and calculated their TPR and FPR, respectively. This survey will help Internet security framework developers to understand and select the detection mechanism according to implementation requirements. In the future, we plan to use a single large phishing site data set and evaluate the experimental performance of a limited referred detection techniques.

References

Kumaraguru, P., Rhee, Y., Sheng, S., Hasan, S., Acquisti, A., Cranor, L. F., & Hong, J. (2010). Teaching Johnny not to fall for Phish. *ACM Transactions on Internet Technology*, *10*(2), 1–31. https://doi.org/10.1145/1754393.1754396

Afonso, J. A., Sousa, R. A., Ferreira, J. C., Monteiro, V., Pedrosa, D., Afonso, J. L. (2017). *IoT system for anytime/anywhere monitoring and control of vehicles' parameters* [Paper presentation]. 2017 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), September 2017, pp. 193–198.

Aydin, M., Baykal, N. (2015). *Feature extraction and classification phishing websites based on url* [Paper presentation]. Proceeding of IEEE Conference on Communications and Network Security (CNS), September pp. 769–770.

Bhatt, P., & Thakker, B. (2020). A novel forecastive anomaly based botnet revelation framework for competing concerns in internet of things. *Journal of Applied Security Research*, *0*(0), 1–21.

Che, H., Liu, Q., Zou, L., Yang, H., Zhou, D., and F. Yu. (2017). *A content-based phishing email detection method* [Paper presentation]. 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, 2017, pp. 415–422. https://doi.org/10.1109/QRS-C.2017.75

Chiew, K. L., Yong, K. S. C., & Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications*, *106*, 1–20. https://doi.org/10.1016/j.eswa.2018.03.050

Computational Complexity of Machine Learning Algorithm. (2018). https://www.thekernel-trip.com/machine/learning/computational-complexity-learning-algorithms/.

Computational Complexity of Training Neural Networks Using Back Propagation. (2018). https://ai.stackexchange.com/questions/5728/what-is-the-time-complexity-for-training-a-neural-network-using-back-propagation.

de Jesus Prado, K. H., Souza, L. S., D. de Jesus Junior, I., & Júnior, M. C. (2020). Applied intelligent data analysis to government data related to criminal incident: A systematic review. *Journal of Applied Security Research*, *15*(3), 297–331.

Ding, Y., Luktarhan, N., Li, K., & Slamu, W. (2019). A keyword-based combination approach for detecting phishing webpages. *Computers & Security*, *84*, 256–275. https://doi.org/10.1016/j.cose.2019.03.018

Dou, Z., Khalil, I., & Khreishah, A. (2017). CLAS: A novel communications latency based authentication scheme. *Security and Communication Networks*, *2017*, 1–20. https://doi.org/10.1155/2017/4286903

Edith, J. J., & Chandrasekar, A. (2014). Layered architecture to detect attacks using asymmetric support vector machine. *Journal of Applied Security Research*, *9*(2), 133–149. https://doi.org/10.1080/19361610.2014.883272

Feroz, M. N., & Mengel, S. (2014). *Examination of data, rule generation and detection of phishing urls using online logistic regression* [Paper presentation]. 2014 IEEE International Conference on Big Data (Big Data), October 2014, pp. 241–250. https://doi.org/10.1109/BigData.2014.7004239

Hong, J. (2012). The state of phishing attacks. *Communications of the ACM*, *55*(1), 74–81. https://doi.org/10.1145/2063176.2063197

Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing detection: A literature survey. *IEEE Communications Surveys & Tutorials*, *15*(4), 2091–2121. https://doi.org/10.1109/SURV.2013.032213.00009

Kim, H., & Huh, J. H. (2011). Detecting DNS-poisoning-based phishing attacks from their network performance characteristics. *Electronics Letters*, *47*(11), 656–658. https://doi.org/10.1049/el.2011.0399

Komatwar, R., & Kokare, M. (2020). A survey on malware detection and classification. *Journal of Applied Security Research*, *0*(0), 1–31.

Liu, G., Qiu, B., Wenyin, L. (2010). *Automatic detection of phishing target from phishing webpage* [Paper presentation]. 20th International Conference on Pattern Recognition, Aug 2010, pp. 4153–4156.

Marchal, S., François, J., State, R., & Engel, T. (2014). Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, *11*(4), 458–471. https://doi.org/10.1109/TNSM.2014.2377295

Marchal, S., Saari, K., Singh, N., & Asokan, N. (2016). *Know your phish: Novel techniques for detecting phishing sites and their targets* [Paper presentation]. 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), June 2016, pp. 323–333. https://doi.org/10.1109/ICDCS.2016.10

Moghimi, M., & Varjani, A. Y. (2016). New rule-based phishing detection method. *Expert Systems with Applications*, *53*, 231–242. https://doi.org/10.1016/j.eswa.2016.01.028

Mohammad, R. M., Thabtah, F., & Mccluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, *25*(2), 443–458. https://doi.org/10.1007/s00521-013-1490-z

PhishTank. (2020a). https://www.phishtank.com/.

Phishbank. (2020b). https://phishbank.org/.

Rao, R. S., & Pais, A. R. (2019). Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach. *Journal of Ambient Intelligence and Humanized Computing*, *11*, 1–20.

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from urls. *Expert Systems with Applications*, *117*, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029

Sentürk, S., Yerli, E., & Soukpnar, I. (2017). *Email Phishing Detection and Prevention by Using Data Mining Techniques* [Paper presentation]. Proceedings of International Conference on Computer Science and Engineering (UBMK), pp. 707–712.

Shi, J., & Saleem, S. (2012). *Phishing*. University of Arizona, Department of Computer Science, Tech. Rep.

Varshney, G., Misra, M., & Atrey, P. K. (2016). A survey and classification of web phishing detection schemes. *Security and Communication Networks*, *9*(18), 6266–6284. https://doi.org/10.1002/sec.1674