# Review on Phishing Attack Detection Techniques

Neel Dholakia[1], Pragati Agrawal[2]

Student[1], Assistant Professor[2]

Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal-462003.

***Abstract*—Phishing attacks capitalize on human errors and target the vulnerabilities formed due to it. Most of the attacks are aimed at stealing private information from users, which spread via different mechanisms. There is no single solution to this problem to effectively nullify all the attacks but multiple techniques have been developed to defend against these attacks. This paper reviews the work on the detection of phishing attacks. In this paper, we aim to study the techniques which mainly detect and help in preventing phishing attacks rather than mitigating them. A general run-through of the most successful techniques for phishing attack detection has been presented here.**

***Index Terms*—Machine Learning, Phishing Detection, Blacklist, Heuristics, Classification**

## I. INTRODUCTION

Phishing is a type of attack that targets users to steal their private data, including credit card numbers and login credentials. Aim of a phishing webpage is to lure unsuspecting web surfers' into revealing their private information. The attackers use varied methods, but the most common method is through forging an email acting as a third-party to dupe the victim into providing their data. Hence, a situation occurs where training users, against these attacks, could prove as a good defensive measure.

Some special attacks are mainly aimed towards an organizational unit to have them incur a loss or to gain access to their networks by making their employees as a victim. To accomplish the task, specially engineered phishing emails and websites to that organizational unit are used by attackers which makes the victim put their guard down thinking the email to be a legitimate email. The type of attack which targets a specific person or organizational unit is known as a variant of a phishing attack called spear phishing.

Phishing has a long history with the term *phishing* being coined from fishers or attacks which used phones to fish for information from their targets. As surveyed in [1], the data indicates, as the use of computer and technology increases, the number of reports of phishing webpages also increases, so detection or prevention of the attacks has become a necessity for some years.

The definition for phishing is not consistent as it changes with the change in the type of attacks. A definition in [2], that is coined by PhishTank states:

*"Phishing is a fraudulent attempt, usually made through email, to steal your personal information."*

The aforementioned definition holds for most scenarios, but in the recent few years, phishing attacks have transitioned to using malware as well as Domain Name System (DNS) based phishing as a means to invade user end-systems and redirect them as and when necessary for them.

For example, malware could be installed on the end-user system by using a phishing email. At any point in time, it can use that system for a Distributed Denial of Service (DDoS) attack on any organization or server in the world. So, it is not considered to be fully suitable to define phishing.

A definition by Colin Whittaker *et al.* in [3] states:

*"We define a phishing page as any web page that, without permission, alleges to act on behalf of a third party to confuse viewers into performing an action with which the viewer would only trust a true agent of the third party."*

Unlike the previous definition, here the definition makes it clear that the attackers are not just limited to stealing personal information. But it still restricts the concept of using malware in phishing attacks as attackers do not always behave as on behalf of a third party.

Phishing attacks aim at exploiting human errors and weaknesses, so it becomes difficult to detect them. These errors and weaknesses can be reduced, but they cannot be removed completely.

Many types of countermeasures have proposed to detect and prevent phishing attacks which can be divided into two major categories:

1) Human-based techniques
2) Software-based techniques

But over the years, studies performed as shown in [4], indicate that after training users with special phishing awareness programs, there was a 40% decrease in the attack being successful, but 29% users still failed to detect the attacks. Human-based techniques like communities are created to make people aware and document different details regarding the statistical values of phishing attacks, but due to the massive amount of attacks, the communities cannot

handle major attacks. As these human-based techniques are not successful in detecting phishing attacks, software-based techniques that were developed in recent years have been used to detect phishing attacks in bulk. But, even so, as software-based techniques are not able to detect some specially engineered attacks, it has led to many organizations having several security breaches over the years as seen from the statistics in [5]. Some of the thoroughly developed software-based techniques to be discussed here are:

1) Blacklist and whitelist generation
2) Rule-based Heuristics
3) Machine Learning

There are some types of attacks that have been hard to defend against that are termed as zero-hour attacks. The term signifies a phishing attack that has not been identified or registered by anyone yet and has a higher probability for anyone to get caught up in. Even some software-based techniques are not able to detect these attacks which makes them dangerous from a web surfer's point of view.

Here, we aim to focus on software-based techniques, their varied designed frameworks or approaches used, and how they can be improved upon to provide maximum efficiency in detecting phishing attacks. The rest of the paper is organized as follows, in Section II to Section IV we present the idea and methodology for the developed software-based techniques. Then, Section V presents the evaluation of the discussed software-based techniques. Section VI then, presents the conclusion and future work.

## II. Phishing Detection using Blacklist and Whitelist Generation

The most novel method to detect and block phishing websites is by generating a list that blocks the website before it loads. As explained in [6], we have a whitelist and a corresponding blacklist that contains websites that are allowed to be opened and which are not, respectively, effectively notifying the browser which websites are allowed to be opened and which are blocked. The check occurs before a request is sent to the webserver for the website. At that point, the Uniform Resource Locator (URL) is checked in the blacklist to check if it is a phishing website or not. If found, the browser blocks access to the website. If not then it allows the user to continue. Many of the popular web browsers these days use this method to protect even unaware users.

### *A. Google Safe Browsing API*

Google Safe Browsing browser plugin is mainly used in Google Chrome and Mozilla Firefox. It maintains a record of all URLs which have been reported to be malware or phishing websites which can be found at [7]. So, the Application Program Interface (API) crawls the web every day in search of infected websites and adds them to its record. Every day the API sends an update to the plugin on the end-user systems running it and updates it to the latest list of websites to block. This API released with two versions, both have their advantages and drawbacks. Seeing these drawbacks, another version, i.e., Version 3 was released which improved the security over time using the testing data acquired from Version 1 and Version 2.

Version 3 uses protocol buffers to encode chunk of data over the hashing algorithms as explained in [8]. The major difference between Version 3 and the previous versions is that Version 3 started using hashes of 32-bit to match with URL hashes. If the first 32-bit hash matches, the full-length hash is requested from the server and matched with the URL hash. This approach increases the efficiency of the API as the majority of the cases encountered do not match which saves processing time.

### *B. Microsoft SmartScreen*

Microsoft SmartScreen plugin was first introduced in Internet Explorer 7 by the name of Phishing Filter. In its early stage, every URL was not checked and only suspicious URLs were checked. As it developed, its name was changed to SmartScreen and a feature of comparing each URL was added. SmartScreen in [9] keeps a local list of popular legitimate websites. Hence, instead of keeping a blacklist, it keeps a local whitelist and if a website is not listed in the whitelist then it is sent to the Microsoft servers for further investigation.

By maintaining a list locally, the time required to verify a URL is decreased drastically than looking it up online but this means that the list needs to be kept updated frequently. If the website is found to be harmful, then a warning is issued letting the user know it has been blocked. Users can also report websites they deduce to be suspicious. Similarly, SmartScreen also issues a warning of caution to the user for a website it finds to be suspicious.

## III. Phishing Detection using Rule-based Heuristics

The second method that we discuss here is to generate rules by extensively studying a set of features and classifying the webpages based on those rules. The features are selected nominally by user discretion which potentially impacts the performance of the system as a whole for better or worse. The method is generally clustered as an add-on, set up on a client's system, and can be used to detect zero-hour phishing attacks as well as can be used in tandem with machine learning techniques which will be discussed later on to make the detection more efficient.

### *A. CANTINA*

CANTINA from [10] is a browser plugin that performs phishing detection by analyzing the content and calculating

its Term Frequency-Inverse Document Frequency (TF-IDF) values and using it with Robust Hyperlinks framework from [11]. Then, it provides the values to search engine and performs heuristics to generate a result in Boolean. The procedure is as follows:

1) Calculate the TF-IDF values of each term on the webpage
2) Sort the values in descending order and top 5 values are selected to represent the page named to be its Lexical Signature
3) Submit top 5 terms to a search engine as a query, e.g., Google
4) If the suspected domain is found within a range of $n$ results, the site is confirmed to be legitimate.

To decrease false positives, we use the following heuristics:

1) *Domain Age* - Many of the phishing sites have domains registered only a few days before phishing attacks are commenced
2) *Known Images* - Checks if there are any inconsistent logos or images on the page which do not match with the domain name
3) *Suspicious URL* - As (-) or (@) are rarely used in a domain name, if they are encountered, it is a phishing website
4) *Suspicious Links* - The above heuristic is applied to all links on the page. If any URL fails the check, it is a probable phishing website
5) *IP Address* - Checks if the domain name is an IP address
6) *Dots in URL* - Checks the number of dots in pages URL, if more than 5 dots then it is a phishing website
7) *Forms* - If any forms encountered with a credit card or password label then it is likely to be a phishing website

Each of the heuristic is calculated by the given function shown in [10]:

$$S = f(\Sigma\, w_i \cdot h_i) \quad (1)$$

Where $h_i$ is the result of each heuristic function, $w_i$ is the weight of each function and $f$ is a simple threshold function that returns $+1$ and $-1$ to represent legitimate and phishing websites respectively.

The drawback here is that, if a webpage instead of content only uses an image as an embedded object then this plugin cannot detect it and the website is allowed to pass as a legitimate website.

*B. CANTINA+*

To effectively detect zero-hour attacks, an improved version of CANTINA has been developed named CANTINA+ in [12]. CANTINA+ instead of calculating the TF-IDF values and using them, it combines the use of heuristics with Machine Learning. The procedure is divided into two phases: The training and testing phase.

1) Training Phase
   a) Webpages are sent to the feature extractor.
   b) Feature extractor uses ML algorithms like Bayesian Network or SVM to classify the webpages.
   c) It generates predictions based on the values generated.
2) Testing Phase
   a) Webpages are sent to a hash-based duplicate remover so any known phishing webpages are eliminated early on.
   b) After removing the duplicates, the webpage is checked for a Login Form.
   c) The features are then extracted and using a pre-trained model we generate a result whether the website is a phishing website or legitimate website.

To elaborate on both the phases, the training phase contains the feature extractor where the below mentioned 15 features are stored and their feature values are calculated for each webpage. Based on these feature values, the machine learning engine builds a classifier that is used in the testing phase. The goal of the testing phase is to label whether the page is a phishing page or not. The testing phase has two filters before the webpage is passed on to the feature extractor to generate feature values, namely hash-based duplicate remover and login form detector.

Generally, phishers use toolkits to generate phishing webpages which generates a massive volume of them but identical in terms of HyperText Markup Language (HTML). So, to remove the duplicates, we generate a 160-bit hash value of the HTML which is then compared to PhishTank's verified blacklist. The webpages which bypass this filter are checked by the Login Form Detector filter. The HTML DOM file is scanned to find the properties of a login form namely FORM tags, INPUT tags, and login keywords. It uses an algorithm to handle any anomalies encountered and prevents false positives. The final step is for the feature extractor to calculate the feature values which are used by the generated classifier to generate a result for the webpage.

The following heuristics are used here:

1) *Embedded Domain* - If dot-separated domain name present in path, then it is likely to be a phishing website
2) *IP address* - Checks if the domain name is a IP address
3) *Dots in URL* - Checks number of dots in pages URL, if more than 5 dots then it is a phishing website
4) *Suspicious URL* - Checks for (-) or (@) in the domain name. If found, then it is a phishing website
5) *Number of sensitive words in URL* - A set of specific sensitive words found in phishing page summarized by Garera *et al.* in [13]
6) *Out-of-position top-level domain* - Checks for uncommon TLD position. If found, then it is a website

7) *Bad forms* - Checks for potentially harmful forms
8) *Bad action fields* - Checks for empty action field or if it points to a domain outside the webpage domain
9) *Non-matching URLs* - Checks the links on the webpage for their validity and if they coincide with the webpage domain
10) *Out-of-position brand name* - Checks for uncommon brand name position. If found, then it is a phishing website
11) *Domain Age* - Generally the domains of phishing sites are registered just a few days before they are used to perform attacks
12) *Page in top search results* - Uses TF-IDF to check whether the page is seen within the top n results
13) *PageRank* - Checks for the PageRank of the webpage and if the value is low then it is likely to be a phishing website
14) *Page in top results with copyright company name & domain* - Uses TF-IDF in combination with company name and domain to find the website in a range of nresults
15) *Page in top results with copyright company name & hostname* - Uses TF-IDF in combination with company name and hostname to find the website in a range of n results

Using all these features in Bayesian Network and SVM Machine Learning algorithms, BN was found to be performing as one the best consistently because of its non-linear and probabilistic nature. There are still some scenarios where CANTINA+ cannot detect attacks like if the attackers hijack legitimate domains and host phishing websites on them. This nullifies most of the features except some which reduce the chances of detection.

## IV. Phishing Detection using Machine Learning Classifier

The third method, having the most potential is to use machine learning models or engines to build a classifier by training the model using training datasets. The machine learning models are models that work based on the inputs and learn to find patterns and classify different items into their item sets. It is particularly useful here as it can be combined with any of the previous methods to give a higher efficiency than the prior system and can detect zero-hour attacks. These techniques have been developed in the recent few years which makes them good as a defense mechanism because of little information to the attackers. The techniques can be designed or customized to capture or detect some specific types of

target pages as well.

### *A. R-Boost Classifier*

R-boost is a combination of multiple algorithms namely C5.0, *k*-Nearest Neighbour (*k*-NN), and Support Vector Machine (SVM). The classifier developed by Toolan F. *et al.* in [14],

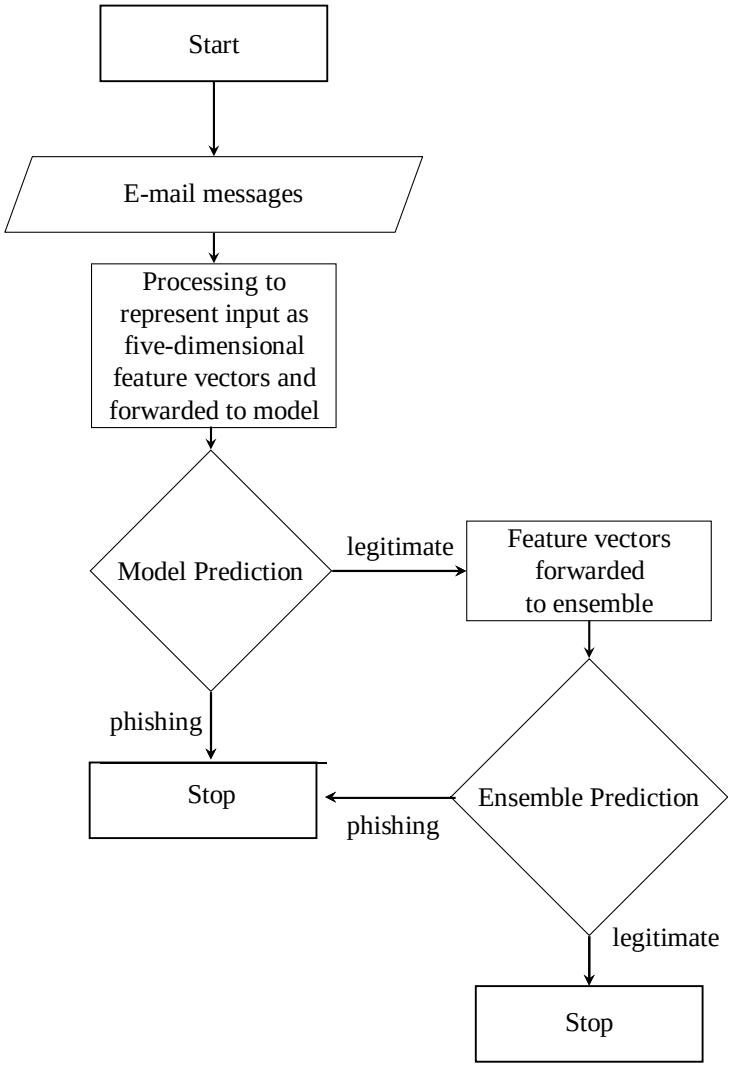


Fig. 1. R-Boost Classifier Procedure

uses a majority voting method to classify the output and hence odd number of learners are selected. The classifier is a combination of 4 models namely,

(i) C5.0
(ii) *k*-NN with $k = 3$
(iii) *k*-NN with $k = 4$
(iv) SVM

Excluding C5.0, models are then combined into an ensemble using majority voting. The procedure for classification is as depicted in the flowchart depicted above.
Here, to represent input emails as feature vectors, only 5 features have been used to speed up the classification task namely:

1) *IP address* - Checks for an IP address found in the email
2) *HTML* - Checks if content-type is "text-html"
3) *Script* - Checks for Javascript in the email body or link

4) *Number of URLs* - Total number of links found in the email
5) *Maximum number of periods in URL* - Highest number of period/dots in a suspected email

The reason this method of the ensemble is used is so that it increases the recall rate of the classifier and decrease the FN rate of the classifier.

### *B. Off-the-Hook*

Off-the-Hook model developed in [15] is a machine learning model which is turned into add-on developed to counter the problems encountered in prior developed models resolving same problems. These problems are,

1) Accuracy
2) Context-independent detection
3) Temporal resilience
4) Resilience to dynamic phishs
5) User privacy
6) Effective protection

Since it runs as an add-on on the client's system, the information is extracted solely from the data sources of a website. This preserves data privacy and provides real-time protection to a system.

Off-the-Hook involves two main components, a target identifier and a phish detector. The target identifier, uses key-terms in a webpage to identify the likely targets of a phish. The phish detector, on the other hand, is a classifier generated using 210 features which identify whether the website is a phishing page or not, but before the page is subjected to analysis it is compared to a local whitelist to preempt any page from the analysis.

When the browser visits a URL, its data sources are extracted and the landing URL is transferred to the phish detector background processing through a dispatcher. Before this, it is checked with the local whitelist and if it belongs to the list then it is considered to be legitimate and no further analysis is needed.

A Phish detector determines how a phisher can construct his phishing page or URL and how much limitations does a phisher has. It is mainly divided into two types.

1) Control
2) Constraints

According to control of a phisher, the features are divided into 5 sets namely:

(i) URL
(ii) Term usage consistency
(iii) Usage of starting and landing mld
(iv) Registered Domain Name (RDN) usage
(v) Webpage content

Generally, in a phishing webpage, the phisher tries to include as much content from URLs out of his control area, as he can. So, the data sources are divided into categories of the feature sets. Then, the terms are calculated which are generally used to fool a target into falling for the phish by a phisher. The 210 features are then evaluated using the Gradient Boosting machine learning model to generate their feature values, to create a classifier model to label the webpage. Gradient Boosting gives a value between 0 to 1 which can be somewhat favored by the user, hence we use a discrimination threshold to class an instance.

If the webpage is labeled to be a *phish*, it is passed on to the target identifier which starts extracting the key-terms and ranks them according to their overall frequency at visible parts of the website. After this, the top $N$ -terms are selected as key-terms and the fully-qualified domain name of the website is guessed as a list of likely targets through a search engine. If the result is found in the search engine the phish detector ruling is overruled and the website is considered to be legitimate but if not found, the top $N$ -terms we found are queried against the search engine. If suspected RDN is found, then it is a legitimate website. If nothing is found, then it returns the three most frequent potential targets as a warning message and confirms the decision of the phish detector to be the final decision.

### *C. Random Forest Machine*

Random Forest is a classifier model that is trained to generate and predict classes of anything by analyzing its features. The Random Forest classifier is considered to be one of the best classifiers for phishing websites as seen in [16]. But, even if it has a low error rate, it has a high false-positive rate. This can be reduced if we use a different combination of heuristic features which has been implemented in [17].

The following is the feature set used:

1) *IP address* - Checks if the URL is an IP address.
2) *Disparities between "href" and Link text* - Checks for differences between href tag and the link text.
3) *Presence of "Link", "Click", "Here" in Link Text* - Checks for the given words in the link text.
4) *Number of Dots in Domain Name* - Checks the number of dots in URL and sets a threshold of 3 dots.
5) *HTML Email* - Checks the content type of an email. If found to be "text/html" then it is likely to be a phishing website.
6) *Presence of Javascript* - Checks for javascript in the body or link of the email. If found, then it is likely to be a phishing website.
7) *Number of Links* - The total number of links embedded in email are recorded.
8) *Number of Linked To Domain* - Total domains the email is linked to is recorded.
9) *From-Body-MatchDomain Check* - All linked domain names are extracted and compared with the sender domain. If any disparity found then, it is likely to be a phishing email.
10) *Word List Features* - Groups of words that frequently appear in

phishing emails. Each of the groups is used as a feature.

- Update; Confirm;
- User; Customer; Client;
- Suspend; Restrict; Hold;
- Verify; Account; Notif;
- Login; Username; Password; Click; Log;
- SSN; Social Security; Secur; Inconvinien;

The algorithm used for RF construction is as follows:

**Algorithm 1:** Random Forest Construction Algorithm

**Result:** Class Classification as Output
**while** *Repeat until D number of trees constructed* **do**
  Select $m$ features from total of $M$ features.
  **while** *Repeat until n number of nodes reached* **do**
    For a node, calculate best split among $m$ features.
  **end**
**end**
Apply new sample to each tree and take output.

This method uses a collection of Decision Trees to generate the output. It combines the output of multiple decision trees each with their classification values and the majority value after the combination is selected as the class of the feature.

There are still ways to improve this approach by combining it with nature-inspired techniques like the Ant Colony Optimization technique. This could effectively allow the classifier to automatically and dynamically train itself to identify the best feature set to be used.

### *D. Deep Learning-Based Sensor*

Deep Learning is a field of machine learning which uses artificial neural networks to create simulations to adapt to attacks and detect them. Deep Learning-Based Sensor as seen in [18] uses multiple layers provide to accuracy to detect real-time attacks. The main problem with Deep Learning-based approaches is the number of resources it uses to run. This challenge has been slightly overcome here with the usage of multiple numbers of layers that divide up the task.

Following layers have been used here in order:

1) *Embedding Layer*
2) *Convolutional Layer*
3) *Concatenation Layer*
4) *Dropout Layer*
5) *Dense Layer*
6) *Sigmoid Layer*

The Deep Learning process is divided into these 6 layers and their working is as follows:

1) First Layer performs simple tokenization of words and performs one-hot encoding of each character. This generates a vector for a word which indicates the relation between the characters.
2) Second Layer is divided up into 5 sub-layers that perform the work of a selection of features. All sub-layers have different filters and kernel values set to identify the best features. Features are extracted by a window of characters equal to the size of consecutive characters. A rectified linear unit (ReLU) activation function is used for each sub-layer. The final result is then flattened and passed to the next layer.
3) Third Layer concatenates the features for further processing. The features received here are from the sub-layers as well as the result from the First layer flattened which preserves the original content from the first layer.
4) Fourth Layer is a regularisation technique used to prevent the overfitting of data in model generation during the training phase. Random neurons are selected and ignored so they are not forwarded for processing.
5) Fifth Layer is a combination of 3 sub-layers and extracts informative features and their patterns are analyzed. Each sub-layer uses a ReLU activation function for processing the patterns.
6) Sixth and final Layer finally determines the maliciousness of the URL with the range being from 0 to 1. It shows the probability of its prediction as its output.

The system used for evaluation of this sensor is a Raspberry Pi 3 B+ with Quad Core 1.4 GHz 64-bit CPU and 1 GB RAM. The system is first trained using multiple server racks or cloud servers. This has still been a challenge as training requires an abundant amount of resources. The system is installed on the Raspberry Pi after training it and it is then transferred onto the WiFi system where it acts as a malicious URL sensor. On a positive hit, it will alert the user and block access to that domain.

## V. Evaluation

The detection techniques can be evaluated by using their false-positive rate and false-negative rate from their literature. As the data sets used are different, the results are not directly comparable but, the source of the data sets is the Internet, so the difference is considered to be negligible.

Blacklists can achieve low FP rates but they cannot detect zero-hour attacks. Hence, there is no future scope for blacklists except being combined with other techniques.

Rule-based heuristics on the other hand can detect zero-hour attacks but there is manual labor required to make them adapt to future phishing trends as well as manage their FP rateswhich tend to be relatively high. CANTINA gives an FP rate of 3% and an FN rate of 11% which is considered to be high.

Machine learning techniques can detect zero-hour attacks as well as maintain low FP rates by combining with rule-based heuristics and using properly developed feature sets. Seeing the statistical values of the techniques, the R-boost classifier gives an FP rate of 1.3% and an FN rate of 0%. R-boost classifiers aimed to decrease the FN rate as it wanted to increase the recall rate which has been properly achieved. Similarly, the RF machine gives an FP rate of 0.06% and an FN rate of 2.5% and can still be improved upon. Machine learning

techniques can automatically adapt to future phishing trends which makes them easy to handle. In recent years, only Machine learning techniques have shown extensive growth in phishing detection which makes it a good field of study for future research. Deep learning being a field which deals with simulations could become a field with major success in this area, but deep learning requires ample resources to even train the model for use which has been a problem. The discussed sensor here uses cloud servers for training and is deployed on resource-constrained devices giving an accuracy of 86.630% which still has more room for improvement.

## VI. CONCLUSION

The reveiw here surveys the current technical progress in the sector of phishing detection attacks as well as the important aspects which could impact the improvement of the techniques. Hence, the aspects considered to be important for future improvement from the performed survey are *Zero-hour attacks detection* and *Low false positive rate.*

- Zero-hour attacks are the attacks which have not been identified, so the best measure for a phishing detection technique is whether it can detect zero-hour attacks and its accuracy regarding them.
- If a system has a high accuracy rating but a high false-positive rating as well then that system will not be able to perform properly as it will do more harm than identify and classify potential attacks.

The most successful software-based techniques reviewed here are:

1) Blacklist and whitelist generation
2) Rule-based Heuristics
3) Machine Learning

From there, the most potential for growth resides in machine learning techniques as seen in Section 3. Machine learning based detection approaches achieved low FP rates and high accuracy ratings and has more potential to grow when combined with rule-based heuristics.

Future work in the field can be done by conducting a study on,

Using deep learning and neural network models taking the Deep Learning-based Sensor as a reference to improve upon for phishing attack detection
Using nature inspired techniques to select the best feature set.

## REFERENCES

[1] E. Khonji, Y. Iraqi, and A. Jones, *Phishing detection:a literature survey*, pp. 2091–2121. IEEE Communications Surveys and Tutorials, Vol. 15, Issue 4, 2013.
[2] PhishTank, "What is phishing." https://www.phishtank.com/what is phishing.php. Accessed March 2020.
[3] C. Whittaker, B. Ryner, and M. Nazif, *Large-scale automatic classification of phishing pages*. NDSS '10, 2010.
[4] S. Sheng, M. Holbrook, P. Kumaraguru, L. Cranor, and J. Downs, *Who falls for phish? a demographic analysis of phishing susceptibility and effectiveness of interventions*, p. 373–382. Proceedings of the 28th international conference on Human factors in computing systems, ser. CHI '10.New York, NY, USA: ACM, 2010.
[5] K. Mitnick, "Phishing attacks are the number one data breach attack vector in the u.k.." https://blog.knowbe4.com/ phishing-attacks-are-the-number-one-data-breach-attack-vector-in-the-u.k. Accessed March 2020.
[6] V. Patil, P. Thakkar, C. Shah, T. Bhat, and S. Godse, *Detection and prevention of phishing websites using machine learning approach*. Fourth International Conference on Computing Communication Control and Automaton, Pune, India: IEEE, 2018.
[7] Google, "Safe browsing." https://developers.google.com/safe-browsing. Accessed March 2020.
[8] P. Sandhu, *Google safe browsing - web security*, p. 283–287. IJCSET: July, Vol 5, Issue 7, 2015.
[9] Microsoft, "Smartscreen faq." https://support.microsoft.com/en-in/help/17443/microsoft-edge-smartscreen-faq. Accessed March 2020.
[10] Y. Zhang, J. Hong, and L. Cranor, *Cantina: a content-based approach to detecting phishing web sites*, p. 639–648. Proceedings of the 16th international conference on World Wide Web, ser. WWW '07. New York, NY, USA: ACM, 2007.
[11] T. Phelps and R. Wilensky, *Robust Hyperlinks and Locations*. D-Lib Magazine, vol. 6, no. 7/8, July, 2000.
[12] 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. 14, 2, Article 21, 2011.
[13] S. Garera, N. Provos, M. Chew, and L. Rubin, *A framework for detection and measurement of phishing attacks*, p. 1–8. Proceedings of the 2007 ACM Workshop on Recurring Malcode (WORM'07), 2007.
[14] F. Toolan and J. Carthy, *Phishing detection using classifier ensembles*, p. 1–9. eCrime Researchers Summit, 2009. eCRIME '09, 2009.
[15] S. Marchal, G. Armano, T. Gröndahl, K. Saari, N. Singh, and N. Asokan, *Off-the-hook: an efficient and usable client-side phishing prevention application*, p. 1717–1733. IEEE Transactions on Computers, Vol. 66, Issue 10, Oct., 2017.
[16] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, *A comparison of machine learning techniques for phishing detection*. Conference: Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit 2007, Pittsburgh, Pennsylvania, USA, October, 2007.
[17] A. Akinyelu and A. Adewumi, *Classification of phishing email using random forest machine learning technique*. Journal of Applied Mathematics, 2014.
[18] B. Wei, R. Hamad, L. Yang, X. He, H. Wang, B. Gao, and W. Woo, *A Deep-Learning-Driven Light-Weight Phishing Detection Sensor*. Sensor Signal and Information Processing II, 2019.