Hybrid Client Side Phishing Websites Detection Approach

Firdous Kausar, Bushra Al-Otaibi, Asma Al-Qadi, Nwayer Al-Dossari
Department of Computer Science
Imam University
Riyadh, Saudi Arabia

***Abstract*—Phishing tricks to steal personal or credential information by entering victims into a forged website similar to the original site, and urging them to enter their information believing that this site is legitimate. The number of internet users who are becoming victims of phishing attacks is increasing beside that phishing attacks have become more sophisticated. In this paper we propose a client-side solution to protect against phishing attacks which is a Firefox extension integrated as a toolbar that is responsible for checking whether recipient website is trusted or not by inspecting URLs of each requested webpage. If the site is suspicious the toolbar is going to block it. Every URL is evaluated corresponding to features extracted from it. Three heuristics (primary domain, sub domain, and path) and Naïve Bayes classification using four lexical features combined with page ranking received from two different services (Alexa, and Google page rank) used to classify URL. The proposed method requires no server changes and will prevent internet users from fraudulent sites especially from phishing attacks based on deceptive URLs. Experimental results show that our approach can achieve 48% accuracy ratio using a test set of 246 URL, and 87.5% accuracy ratio by excluding NB addition tested over 162 URL.**

Keywords—Phishing Attacks; Browser Plugin; Anti Phishing; Security; Firefox

I. INTRODUCTION

Phishing is an online identity theft in which attackers use social engineering to appear as a trusted identity to gain valuable information. Phishing exploits human vulnerabilities rather than software vulnerabilities. It targets many kinds of confidential information including usernames, passwords, social security numbers, credit card numbers, bank account, and other useful personal information.

In the past few years we have seen an increase in the number of phishing attacks with many variants of techniques targeting every sector of society. As reported by the Anti-Phishing Working Group (APWG) (Anti-Phishing Working Group. "Phishing Activity Trends Report: Third Quarter 2013 Report, 2014) "Payment Services continued to be the most-targeted industry sector throughout 2014". Many of phishing techniques are sophisticated, and it is very hard to internet users to defend against them. Damage caused by phishing ranges from minor to substantial financial loss. According to the statistics provided by APWG in their Phishing Activity Trends Report[1] "Overall phishing activity was up by 20 percent in 3rd Quarter of 2013 from the previous quarter ", and Cyveillance whitepaper 2008 reported phishing attacks against more than 2,000 brands across 30 countries which costs these organizations from thousands to millions of dollars per attack.

The phishing techniques usually involve impersonating legitimate web sites to submit personal information directly to the phisher, or using malicious software that sends victim's data without his knowledge. In a typical phishing attacks, the victim receives fraudulent email asking him to visit a web site and confirm his information in a given time. The email provides a legitimate-looking URL which direct to a spoofed web site where victims are going to enter their information.

Several of phishing solutions exist like blacklists which are databases of known phishing sites, whitelists, community ratings, analysis of the URLs and webpage content (images, and text), using machine learning techniques, and various heuristics to detect phishing attacks.

This paper makes the following contribution: We uses URL structure, four lexical features and page ranking to capture phishing attacks that depends on deceptive links. Every URL is evaluated corresponding to three heuristics (sub domain, primary domain, and path) and three lexical features extracted from the URL combined with page ranking received from ranking services. The proposed method requires no server changes and will prevent from phishing attacks based on fraudulent URLs. This solution uses resources like search engine suggestions, and third party services (Alexa, and Google Page Rank).

II. RELATED WORK

There are several methods that can be used to identify a web page as a phishing site, including Whitelists/Blacklists, URL and Heuristic-based, Similarity assessment techniques, and community ratings. In this section we will go through some of these solutions.

Whitelist/Blacklist-based is one of the common used approaches. It holds URL of verified phishing site. A whitelist contains URLs of legitimate sites while a blacklist contains phishing sites. It is effective to protect against phishing attacks and generates close-to-zero false positive rate but requires regular updating and is vulnerable to zero-day attacks. Many anti-phishing technologies rely on this approach. For example, Internet Explorer has built-in blacklist-based anti-phishing solution provided by Microsoft servers. Also Google's Safe Browsing extension which uses Google global blacklist and whitelist.

Content-based solutions which verify web pages by examining their contents (e.g. HTML, links, images, and text) against some previously defined characteristics. CANTINA [2] is an example on this approach which uses five words taken from the website to be classified as a signature using Term Frequency-Inverse Document Frequency (TF-IDF), then submits them to Google. If the site's URL is on top results it is legitimate, otherwise it is not. In CANTINA+ [3] which is an enhanced version from CANTINA, new features added and evaluated on a larger corpus to achieve better results. The new approach extended some of previous features combine them with ten more features. And this time the model built using state-of-the-art machine learning algorithms instead of a simple linear classifier. However, both of them have a drawback of time consumption caused by querying search engines.

A third approach developed to improve authentication between the user and the server. Authentication means that before user enters login information he needs to authenticate himself to that page. Also, it means that particular page authenticates to the user that it's the real page (called two way or mutual authentication). Some of anti-phishing techniques provide mutual authentication to prevent phishing attacks. This addresses the problem of user’s inability to authenticate the website he is communicating with. The typical method used in login helps to authenticate user to server side but not the opposite which leaves a chance to attackers to exploit this failure. The success of mutual authentication techniques depends on the way used to authenticate both the client and the server. Some of existing solutions are image-based like the one provided by Confident Technologies company [4] which based on providing a number of categories instead of a specific pictures and let the user choose from them in registration process. At login, server will generate a grid of pictures and asks the user to choose the pictures matches the categories and order chosen in registration level. As soon as the server failed to provide the right grid of images or the user failed to choose the correct images it considered as security warring. Unfortunately, this maximize user’s responsibilities by relying on userto memorize more than one category in specific order besides memorizing a password. Also, it requires changes on server side and login mechanisms. Other solutions uses Image-based user authentication to replace traditional methods (e.g. passwords, and security questions) this may provide stronger authentication but does not solve the server side authentication problem.

PwdHash [5] proposed a solution to strengthen web password authentication. It implements password hashing with domain name as a salt and keyed by the password itself. Server received password after hashing which makes it not useful if received by phishing website. As many of other solution this approach require user to remember using it every time he is about to enter a password.

Dhamija et al.[6] provide an authentication scheme where password is entered into a trusted window and user recognizes one image to perform visual matching to authenticate the received content. Images are generated by the server and they are unique for each transaction. The drawback of this solution is the large amount of changes required on server side.

BogusBiter[7] solves the problem from another point of view. It focuses on the stage after phishing attack occurred and user submitted his information to the wrong recipient. It automatically generates and sends a large number of fake credentials to phishing site to hide the real one. Unfortunately, BogusBiter can't work alone it needs help to be turned on from web browser or a third-party toolbar to detect phishing sites.

Aravind et al.[8] propose an anti-phishing framework which uses visual cryptography for authentication. Image is decomposed into two shares one stored with user and the other one is on website's database. An image captcha is created from those two shares in login time. The proposed method success to authenticate both user and website.

Web Wallet[9] is a sidebar login box which displayed when a user requested to login through a trusted path. It is responsible for preventing users form submitting their sensitive data directly to any website before checking that site. The developers of this sidebar used the negative visual feedback to solve the vulnerability of spoofing the sidebar and they provide cards to most of user's sensitive data not only user name and password.

TrueWallet[10] is another wallet-based approach which works as a proxy to manage user login and protect his password and credentials. It runs isolated from browser which adds an advantage to it compared to Web Wallet approach which means it is more secure and difficult to be attacked. TrueWallet uses the standard SSL-based authentication with some modification on server side. This approach has two disadvantages. First, it is vulnerable to DNS-spoofing attacks. Second, user need to be trained in order to rely only on this method to fill in any form.

One area of work relies on URL features to detect phishing webpage. Khonji et al.[11] propose a technique for detecting phishing websites by lexically analyzing suspect URLs depending on a novel heuristic phishing feature. This technique targets a subset of phishing attacks where the victim name is included in the URL. The approach achieved 63% and 83% true positive rate for loose and strict modes respectively. Whittaker et al.[12] present the design and evaluation of a large-scale machine learning based classifier .The proposed classifier evaluates the page according to its URL, content, and host information. The dataset used in training process consists of a noisy dataset of millions of samples. The evaluation concludes with more than 90% of phishing pages correctly identified.

An approach developed by Le et al.[13] to identify phishing target using only lexical features. Authors used an online method Adaptive Regularization of Weights in classifying URLs. Analysis showed that this methodology led to high classification accuracy comparable to full featured approaches. An approach that relies on 23 features derived from URL structure, lexical features, and from brand name of website is proposed in by Huang et al.[14]. These features model the SVM-based classifier used to inspect each requested URL .The evaluation done using three datasets containing more than 12,000 URLs and showed that the solution can obtain 99% accuracy.

Blum et al.[15] have proposed a method exploits URL's lexical features that are fed to the confidence-weighted algorithm, to indicate suspicious URL. This method uses a large lexical model trained using online approach which makes it capable of detecting zero hour threats. Zhang et al.[16] proposed different method based on repository to extract features and a statistical machine learning algorithm avoiding the complexity of computation caused by URL-based method. This method succeeds to identify phishing sites with more than 93% accuracy.

Nguyen et al.[17] presented a heuristic-based algorithm uses the characteristics of the URL combined with a third party services (e.g. PageRank) giving the URL a major role in phishing detection. Another classifier produced as a toolbar (PhishShark) proposed which is heuristic-based-only combining URL and HTML features led to promising results.

Finally, we will conclude with some of existing toolbars[18] built to prevent phishing attacks. Netcraft is a Mozilla browser plug-in that displays host location and risk rating of the accessed site. User can report sites to Netcraft to validate them then add them to its blacklist database if they are phished. TrustWatch is toolbar for Internet Explorer that checks the URL in the black listed database and displays its domain name. Searching blacklist is a time consuming process since they continuously growing and they are vulnerable to zero-day-attacks. Spoofguard is an anti-phishing Internet Explorer plug-in. It examines page characteristics such as images, links, and domain name against common features extracted from phishing site to decide whether this page is spoofed or not.

III. Proposed Approach

Our system is inspired by solutions proposed by Nguye et al. [18] and Xiaoqing et al. (GU Xiaoqing, 2013). It combines both approaches the heuristic-based approach and NB classifier. In Nguye et al. solution URL-related features and Page Ranks used to classify each website. Xiaoqing et al. approach depends onto two phases. The first one is an NB classifier which uses four lexical features to decide whether URL is phishing, suspicious, or legitimate. The second phase uses SVM classifier to parse the webpage against some features. The system will enter the second phase only if URL classified as suspicious in first phase. In our proposed system we combined the first approach with the first phase of second approach without entering into its second phase.

A. System Model

Our system model consists of seven main modules as illustrated in figure 1.

- Receiving URL Module

The system obtains the requested URL from the browser. The output of this module is page URL and it is a fundamental input in most of system modules.

- Features Extraction Module

This module extracts URL domain-related features. The URL separated into different components which are Primary Domain, Sub Domain, and Path. The pervious features will play an eminent role for investigating the URL and predicting phishing pages in next modules.

- Ranking module

Beside URL's feature extraction in previous modules, also this module collects URL metadata. Specifically, URL Page Rank to be used as input into next module. Google Page Rank, and Alexa Rank are used for this purpose.

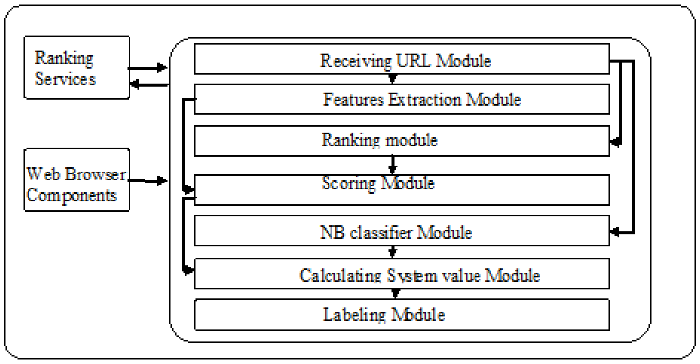

Fig. 1. System Modules

- Scoring Module

In this module heuristics derived from modules B and C are used as input and their values calculated as output. As a result, the site is considered as phishing if all calculated values are negative, and is considered as legitimate if they are all positives.

- NB classifier Module

This module is responsible for classifying a URL with a classification model developed in training process. The features used by the classification system are checking whether the URL contains an IP address because this method used by phishers to hide the owner of the site. Another feature is to examine the presence of a large number of dots separating hostname. Phishers tend to use more dots in their URLs to impersonate a legitimate look of URL because there is no restrictions on the number of dots can be used in sub domains. Checking URL against special symbols such as '@' or '-' is another feature because many of phishing URLs modified using these symbols which makes it possible to write URLs that appear legitimate but actually lead to different pages. URLs corresponding to legal websites usually do not have a large number of slashes [19]. As a result, URL that contains a large number of slashes is considered to be a phishing. The classifier entered into two phases training, and testing phases. Training phase used to build the classifier by calculating the probabilities that the given webpage belongs to a one of two classes (phishing, and legitimate). The testing phase is used to examine the ability of classifier to label real web pages with a correct class.

- Calculating System Value Module

In this step each heuristic is given a weight obtained by a classifier. After that, system values are calculated using this equation:

$VS = \sum_{i=1 \text{ to } 6} (\text{heuristic}_i \text{ value}) * (\text{heuristic}_i \text{ weight})$

- Labeling Module

This module deals with system value and compares it to threshold to give system output which is the URL final label. As a result, user may proceed safely, or warned about the website.

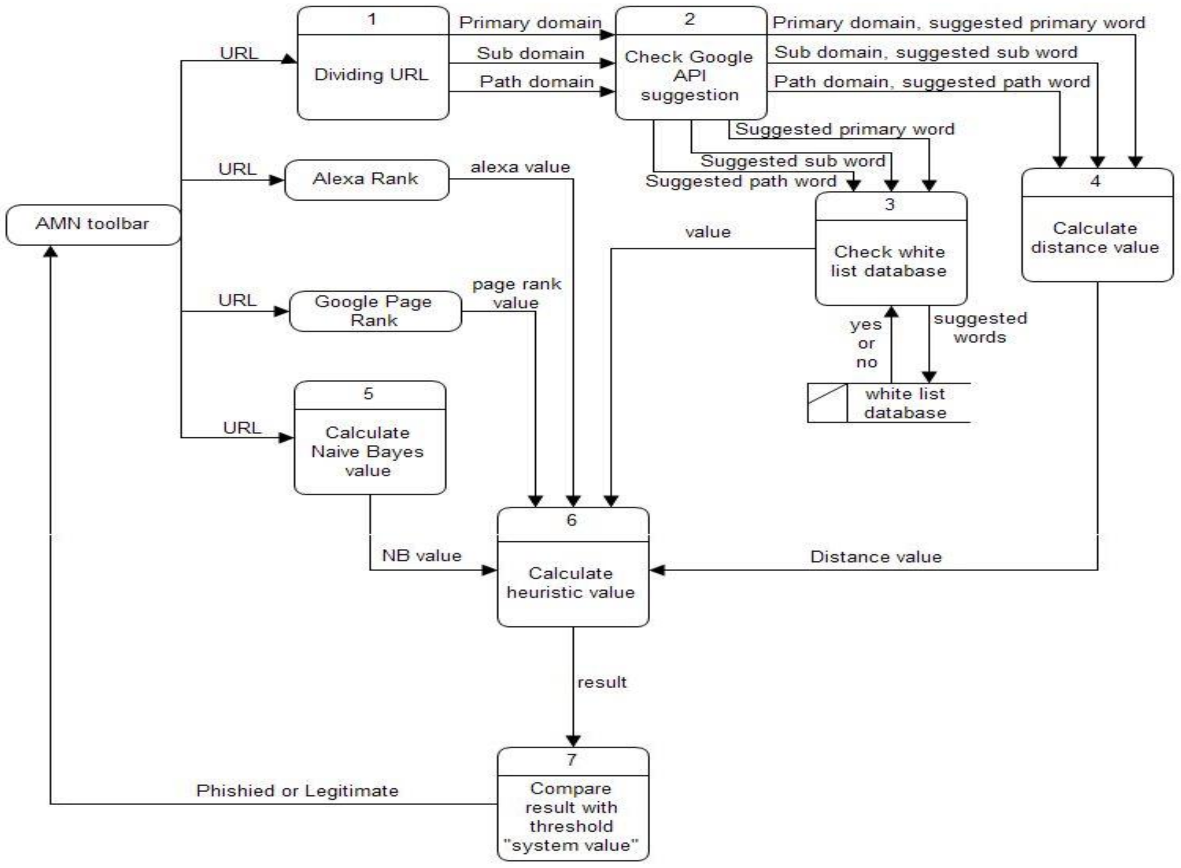

Fig. 2. Data Flow Diagram

B. Structured Design

It shows data exchanged between system components. As we can notice, from the Figure-2 URL is the main part of data. Most of major components and processes need URL value as input to produce their results. Also, URL in most cases need to be decomposed into three parts (sub domain, primary domain, and path) which is the responsibility of "Dividing URL" process. Process two receives URL parts and returns suggestions of each part separately. Suggestion values used by two processes three and four to check them in a list of popular phishing targets as in process three then return a value of yes or no. Process four produces edit distance value between each suggested word and its corresponding URL part. Process six is the major part of the system since all of the data produced in other system processes will be used here to calculate final system result. Process seven is the last process in system which communicates only with one process to receive result value and compare it with a predefined threshold to make system decision.

C. Proposed Algorithm

The pseudo code of our proposed algorithms to detect phishing websites are described below.

Algorithm Primary Domain Value

```
Input  Primary Domain
Output Primary Domain Value
if (Primary Domain=null)
    Value= -0.5
else
    S=Suggestion(Primary Domain)
if (S=null)
    Value= -0.25
else
    T=Whitelist(S)
    if (T)
        Ed=Levenshtein(S, Primary Domain)
        if (Ed=0)
            Value= 1
        else-if (0< Ed<3)
            Value= -0.5
        else-if ( Ed>=3)
            Value= 0.25
    else
        Ed=Levenshtein(S, Sub Domain)
        if (Ed=0)
            Value= 0.5
        else-if (0< Ed<3)
```

```
                Value= -0.25
            else-if ( Ed>=3)
                Value= 0.25
```

Algorithm Sub Domain Value

```
Input  Sub Domain
Output Sub Domain Value

if (Sub Domain=null)
    Value= 0
else
    S=Suggestion(Sub Domain)
if (S=null)
    Value= 1
else
    T=Whitelist(S)
    if (T)
                Value= -1
    else
        Ed=Levenshtein(S, Sub Domain)
        if (Ed=0)
                Value= -0.5
        else-if (0< Ed<3)
                Value= -0.25
        else-if ( Ed>=3)
                Value= 0.5
```

Algorithm Path Domain Value

```
Input  Path Domain
Output Path Domain Value

if (Path Domain=null)
    Value= 0
else
    S=Suggestion(Path Domain)
if (S=null)
    Value= 1
else
    T=Whitelist(S)
    if (T)
                Value= -1
    else
        Ed=Levenshtein(S, Path Domain)
        if (Ed=0)
                Value= -0.5
        else-if (0< Ed<3)
                Value= -0.25
        else-if ( Ed>=3)
                Value= 0.5
```

Algorithm Page Rank Value

```
Input  Page Rank
Output Page Rank Value

if (Page Rank<=0) then Value= -1
if (1<=Page Rank<=2) then Value= -0.5
if (3<=Page Rank<=4) then Value= -0.25
if (5<=Page Rank<=6) then Value= 0.25
if (7<=Page Rank<=8) then Value= 0.5
if (9<=Page Rank<=10) then Value= 1
```

Algorithm Alexa Rank Value

```
Input  Alexa Rank
Output Alexa Rank Value

if (Alexa Rank<300,000) then Value= 1
if (300,000<= Alexa Rank<=500,000) then Value= 0.5
if (500,000<= Alexa Rank<=1000,000) then Value= 0.25
if (1000,000<= Alexa Rank<=2000,000) then Value=- 0.25
if (2000,000<= Alexa Rank<=3000,000) then Value= -0.5
if (Alexa Rank>=3000,000) then Value= -1
```

Algorithm NB Classifier

```
Input  URL
Output   Class Value
host=Host(URL)
path=Path(URL)
features[x1, x2, x3, x4]=Extract(host,path)   //Each feature xi takes value 0, or 1
$P(C_p|$ features $)= P(C_p)*\prod_{1 \text{ to } 4} P(x_i|C_p)$ // $C_p$ is class phishing
$P(C_l|$ features $)= P(C_l)*\prod_{1 \text{ to } 4} P(x_i|C_l)$ // $C_l$ is class legitimate

if ($P(C_l|$ features $)/ P(C_p|$ features $) > \alpha$) Class=1  //legitimate
else-if ($P(C_p|$ features $)/ P(C_l|$ features $) > \alpha$) Class=-1 //phishing
else-if ($(1/\alpha) < P(C_l|$ features $)/ P(C_p|$ features $) < \alpha$)  Class=0 //suspicious
```

Algorithm Calculate System Value

```
Input  Heuristic Values and Weights
Output  Website Final Class

VS=∑ heuristici value * heuristici weight

if ( VS < Threshold ) Value= 0   // Phishing class
else  Value= 1  // Legitimate class
```

- ALEXA RANK

It is a service from Amazon Company since 1996, which gives a value for each page through 3 months in the Web. Increasing of this value is a good indicator. The value depends on 2 important things. First, the number of unique users entered to this site. Second, how many URL linked to this site, increasing of URL's lead to this site will increase its value (Alexa API).

This service serve project to detect phishing sites, because phishing sites has a few number of visitors and linking URLs compared to popular websites. Also, phishing sites usually have a short life cycle which helps to differentiate between legitimate and phishing sites.

- PAGE RANK

It is a service from Google Company. When Google needed to improve searching on web by giving best results to searchers, they thought about giving a value for each page (Karch). High values depend on how many URL linked to the site. Also the value depends on the domain age, the older domain get higher value (Strickland, 2006). So the proposed approach, used this value to be one of factors that affect the decision about whether the site is phishing or not.

- SUGGESTIONS

When user enters to "Google.com" and type a word, there is a drop down list to suggest many words related to user's typed word. The suggestions depend on word popularity in searching. And when you enter a word spelled wrong, there is a famous sentence says "Did you mean?" depends on common spellings (Autocomplete).

Since phishers try to make phishing URL similar to popular sites by adding some letters, removing others, or even substituting them with different letters to trick users that it is their targeted site. So, we used Google suggestions by taking the suspect URL and getting the relative spelling word, then compare those two words using levenshtein distance algorithm.

- LEVENSHTEIN ALGORITHM

It is an algorithm that compares two strings and returns the number of operations (insert, delete, and substitute) known as "distance" to let these words sound the same.

Since our paper use this algorithm to compare between suspect word and Google suggested word and return the number of operations to let those two words be equivalent. If the distance is 0, this means the two strings are the same. But if the distance is between 1 and 2 that means a probability of some phisher is trying to make those two words visually similar.

- WHITE LIST

Usually in phishing world, the white list is group of legitimate sites saved in database. But in our proposed algorithm white list means a list saving primary domains of sites targeted by phishers. This white list is extracted from a database of verified phishing URLs downloaded from PhishTank website. We need to check if the URL domains (primary domain, sub domain and path domain) not in the white list to ensure that there is no phisher exploits the name of a famous legitimate site to trick users.

D. *Functions Implementation*

Function Alexa Rank

Prototype: Function Alexa (url)

Input: URL.

Output: URL's value.

Description: This JavaScript function takes the URL as a parameter, and connects to the server using AJAX to send URL to PHP file which requests Alexa rank API for the URL. Then receive URL's global rank from the server. After that it assigns URL a value based on its rank. Whenever rank is higher, assigned value becomes bigger.

Function Sub Domain

Prototype: Function S_heuristics (SubDomain)

Input: Sub domain.

Output: Sub domain value.

Description: This JavaScript function takes URL's sub domain and passes it to three functions to compute sub domain value, those functions are:

1) Google's search suggestions: return the Google suggested word for the sub domain.

2) White list: check whether the suggested word is a primary domain of another targeted site. If it is in the white list, sub domain will assigned a low value.

3) Levenshtein algorithm: if the sub domain is not in the white list, this function will check the distance between sub domain and the suggested word to check if the sub domain attempts to be closed to another domain. Whenever the distance is lower, the value becomes higher.

Function White List

Prototype: Function whitelist(a)

Input: Google suggestion word.

Output: True or False.

Description: This JavaScript function connects to "PhishTank" database. Each phish site in database has phish id, phish URL, target and other columns. Important columns are:

* Phish URL: The URL of the phishing site.

* Phish id: Id for each phish URL.

* Target: The primary domain of legitimates site which are the phish site attempts to simulate. JavaScript function passes the Google suggested word to PHP file using Ajax to create connection to the database to check whether the suggested word matches any target. If it is exists that means this site attempt to disrupt the user to think it is the primary domain of another legitimate site. The returned value in matching case is true. If the result is true, white list will take low value.

Function NB Classifier

Prototype: Function NB_classifier(host, path)

Input: URL host and path.

Output: Class value.

Description: This function implemented based on Naïve Bayes classification approach. It is a learned classifier trained over a data set of 12,967 phishing URL downloaded from PhishTank and 150 legitimate URL collected manually with help of Alexa top 500 URLs. Features used for classification illustrated in table 1.

TABLE I. FEATURES USED BY NB

Heuristic	**Phishing URL**
Suspicious URL	URL contains @ or -
IP Address	URL contains IP address
Dots in URL	>=5 dots in URL
Slash in URL	>=5 slashes in URL

These features extracted from each URL. Finally, classifier will return one of three values (0, 1, or -1) suspicious, legitimate, or phishing, respectively.

Function Calculation

Prototype: Function calculation( )

Input: Nothing.

Output: System Value.

Description: This function is the main part of the program and the last step of calculations. It applies equation vs = ∑ (heuristici value) * (heuristici weight), where heuristic values are taken from global variables used to store results of previous functions. And heuristic weight calculated by experiments applied on phishing URLs. The result of these function vs returned to calling function to be compared with threshold before presenting the last decision of the program.

IV. Performance Evaluation

Our proposed architecture for anti phishing toolbar uses an extended approach from Nguye et al. [17] by combining their approach with NB classifier proposed in X. Gu, et al. [19]. Algorithms illustrated belloware based on experimental results of 9,661 phishing URL downloaded from PhishTank as Nguye et al. mentioned. Naïve Bayes classifier algorithm used for classification trained over 12,967 phishing URL from PhishTank and 253 legitimate URL collected manually.

Evaluation phase is done in three phases as shown in Table 9. The dataset used for testing is collected using two methods from PhishTank, and manually. URLs in data sets evaluated manually by installing the toolbar and testing each URL individually. Metrics used to calculate toolbar accuracy are True Positive (classifying legitimate URL correctly), False Positive (assigning phishing label to legitimate URL), True Negative (predicting phishing site correctly), and False Negative (assigning legitimate label to phishing URL).

First phase, we started by evaluating Naïve Bayes (NB) approach alone. NB classifier trained over 13117 URLs divided into 12967 phishing URLs, and 150 legitimate URLs. After that, NB tested using a test set of 13220 URLs (12967 phishing, 253 legitimate). We experimented NB using different values of α as illustrated in Table 3. The best value of α which maximizes TN and minimizes FP is 5.8.

In the second phase we evaluate system without Naïve Bayes addition. The test set consists of 162 URLs to be tested. Experiment are done using threshold of value 0. Toolbar detected 77 phishing URLs correctly out of 89, and 63 legitimate URL out of 71. The experiment results with 86.5% True Negative, 88.7%True positive, 11.2% false positive, 13.4% false negative.

Third phase, we combined both of previous approaches. The test set consists 156 phishing URLs (selected from 12,967 URLs) downloaded from PhishTank, and 90 legitimate URLs collected manually. We conclude with 246 URLs developed for testing. We experiment this approach using two different values of threshold 0, and 0.5. Threshold with 0.5 results with less False Negative, so we select it as threshold value. Although it returns a high False Positive it gives a good results of True Negative. False Positive can be reduced using "add to trusted list" feature. The toolbar detected 147 phishing URL correctly out of 156. The experiment results with 94% True Negative.

Accuracies of each approach calculated using this equation: Accuracy ratio= (TP+TN)/(TP+TN+FP+FN). Phase one has 34% accuracy ratio, phase two has 87.5% , and phase three has 48% accuracy ratio.

Finally, after these experiments we concluded by choosing threshold of value 0, and remove the Naive Bayes part as it does not add any improvement on system accuracy and increases false positive rate.

TABLE II. Evaluation Phases

Phase 1:	Naïve Bayes	Trained on	13117 (12967 P + 150 L) URL				
				TP	TN	FP	FN
		Tested on	13220 (12967 P + 253 L) URL	248	4257	5	8710
				98%	33%	2%	67%
Phase 2:	URL-based Approach	Tested on	162 (89 P + 71 L) URL Threshold=0	63	77	8	12
				88.7%	86.5%	11.2%	13.4%
Phase 3:	Combination of 1 and 2	Tested on	246 (156 P+90 L) URL Threshold=0.5	10	147	80	9
				11%	94%	88%	6%

TABLE III. α Values

a. **A**	**TN**	**FP**	**A**	**TN**	**FP**
1.1	6058	18	3.5	6058	18
1.3	6058	18	3.9	6058	18
1.5	6058	18	4.4	6058	18
1.7	6058	18	5.6	6058	18
2.0	6058	18	5.7	6058	18
2.4	6058	18	5.8	4257	5
2.6	6058	18	5.9	4257	5
2.9	6058	18	6	4257	5
3.2	6058	18	6.3	4257	5

V. CONCLUSION

This paper presents architecture for developing an anti-phishing toolbar integrated with Firefox browser to detect phished URLs. Our proposed anti-phishing toolbar will verify user's inputted URL, if the result is phished then it warns the user through changing indicator color and gives the user the choice to unblock the website by adding it to a trusted list. In case of phishing site verified user can know the reason by viewing a report. Our approach categorizes the URL based on its features including four lexical features and three other features (sub domain, primary domain, and path) with help of Naïve Bayes classifier. Our proposed approach can minimize the false positive by giving the user a feature of adding URLs after verified to a trusted list. Experimental results show that our approach can achieve 48% accuracy ratio using a test set of 246 URL, and 87.5% accuracy ratio by excluding NB addition tested over 162 URL.

REFERENCES

[1] Anti-Phishing Working Group. "Phishing Activity Trends Report: Third Quarter 2013 Report," April 2014. Available : http://docs.apwg.org/reports/apwg_trends_report_q3_2013.pdf.

[2] Y. Zhang, J.I. Hong, L. F. Cranor, " Cantina: A Content-Based Approach to Detecting Phishing Web Sites," In Proceedings of the 16th International Conference on World Wide Web (WWW '07). ACM, NY, USA, 639-648, 2007.

[3] G. Xiang, J. Hong, C. P. Rose, L. Cranor, "CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites," in ACM Transactions on Information and System Security (TISSEC), Volume 14 Issue 2, September 2011.

[4] Confident Technologies, "Dynamic, Mutual Authentication Technology for Anti-Phishing" Available: http://confidenttechnologies.com/products/anti-phishing.

[5] B. Ross, C. Jackson, N. Miyake, D. Boneh, J. C. Mitchell, " Stronger password authentication using browser extensions," In Proceedings of the 14th conference on USENIX Security Symposium - Vol. 14. USENIX Association, Berkeley, CA, USA, 2005.

[6] R. Dhamija and J. D. Tygar, " The battle against phishing: Dynamic Security Skins", In Proceedings of the 2005 symposium on Usable privacy and security (SOUPS '05). ACM, NY, USA, 77-88. 2005.

[7] C. Yue and H. Wang , "BogusBiter: A transparent protection against phishing attacks," in ACM Transactions on Internet Technology (TOIT), Volume 10, Issue 2, May 2010 .

[8] K.A.Aravind, R.M. Venkata Krishnan, "Anti-Phishing Framework for Banking Based on Visual Cryptography," International Journal of Computer Science and Mobile Applications, Vol. 2 Issue. 1, January, 2014.

[9] M. Wu, Robert C. Miller, G. Little., "Web wallet: preventing phishing attacks by revealing user intentions," In Proceedings of the second symposium on Usable privacy and security (SOUPS '06). ACM, New York, USA, 2006.

[10] S. Gajek, H. Löhr , A. R. Sadeghi, M. Winandy. , "TruWallet: trustworthy and migratable wallet-based web authentication," In Proceedings of the ACM workshop on Scalable trusted computing (STC '09). ACM, NY, USA, 19-28, 2009.

[11] M. Khonji, A. Jones,Y. Iraqi, "A Novel Phishing Classification Based on URL Features," in IEEE GCC Conference and Exhibition (GCC), Dubai, 2011.

[12] C. Whittaker, B. Ryner, M. Nazif , "Large-Scale Automatic Classification of Phishing Pages," in Network and Distributed System Security Symposium (NDSS), 2010.

[13] A. Le, A. Markopoulou, M. Faloutsos, "PhishDef: URL Names Say It Al," INFOCOM 2011,Shanghai, China, 191-195, 2011.

[14] H. Huang, L. Qian and Y. Wang , "A SVM-based Technique to Detect Phishing URLs," Information Technology Journal, Volume 11 Issue 7, page 921-925, 2012.

[15] Blum, B. Wardman, T. Solorio, G. Warner, "Lexical feature based phishing URL detection using online learning" In Proceedings of the 3rd ACM workshop on Artificial intelligence and security (AISec '10). ACM, NY, USA, 54-60 , 2010.

[16] J. Zhang, Y. Wang, "A real-time automatic detection of phishing URLs," Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on, vol., no., pp.1212,1216, 29-31 Dec. 2012.

[17] L. A. T. Nguyen, B.L. To, H. K. Nguyen, M. H. Nguyen , "Detecting Phishing Web sites: A Heuristic URL-Based Approach," in International

Conference on Advanced Technologies for Communications (ATC'13), Oct. 2013.

[18] L. F. Cranor, S. Egelman, J. Hong, Y. Zhang, "Phinding phish: Evaluating anti-phishing tools," in 14th Annual Network & Distributed System Security Symposium (NDSS 2007), 2007.

[19] X. Gu, H. Wang, T. Ni , "An Efficient Approach to Detecting Phishing Web," Journal of Computational Information Systems 9, pp. 5553-5560, 2013.

[20] "Alexa API," [Online]. Available: http://data.alexa.com/data?cli=10&dat=snbamz&url.

[21] M. Karch, "What Is PageRank and How Do I Use It?," [Online]. Available: http://google.about.com/od/searchengineoptimization/a/pagerankexplain.htm.

[22] J. Strickland, "How Google Works," 20 December 2006 . [Online]. Available: http://computer.howstuffworks.com/internet/basics/google1.htm.

[23] "Autocomplete," [Online]. Available: https://support.google.com/websearch/answer/106230?hl=en.