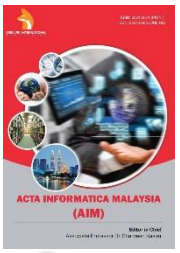
ZIBELINE INTERNATIONAL PUBLISHING

ISSN: 2521-0874 (Print)
ISSN: 2521-0505(Online)
CODEN: AIMCCO

Acta Informatica Malaysia (AIM)

DOI: http://doi.org/10.26480/aim.02.2020.37.44



CrossMark

REVIEW ARTICLE

LINKCALCULATOR – AN EFFICIENT LINK-BASED PHISHING DETECTION TOOL

Orunsolu Abiodun[a], Sodiya A.S[b], Kareem S.O[a]

[a] Moshood Abiola Polytechnic Abeokuta, Ogun NIGERIA.
[b] Federal University of Agriculture, Abeokuta, South west NIGERIA.
**Corresponding Author Email: orunsolu.abdul@mapoly.edu.ng*

ARTICLE DETAILS

Article History:

Received 10 August 2020
Accepted 14 September 2020
Available online 02 October 2020

ABSTRACT

The problem of phishing attacks continues to demand new solutions as existing solutions are limited by various challenges such as high computational requirements, zero-day attacks, needs for updates, complex ruled-based, etc. Besides, the emerging mobile market demands simple solutions to phishing due to several factors such as memory, fragmentation, etc. In response to the above challenges, a simple anti-phishing tool called LinkCalculator is presented. The proposed LinkCalculator anti-phishing scheme is based on an algorithm designed to extract link characteristics from loading URLs to determine their legitimacy. Unlike the other link-based extraction approaches, the proposed approach introduced the concept of weight to represent the different links found in a URL. This is because certain link information within parsed webpages or requests is sufficient to classify them as phishing without loss of generality. The approach is experimented using a dataset of 300 instances consisting of 150 legitimate URLs and 150 phishing URLs from openly-available research datasets. The experimental results indicate a significance performance of 100%. True Negative Rate and 0.00% False Positive Rate for legitimate instances and True Positive Rate of 96.67% with 0.03 % False Negative Rate for phishing instances which indicate that the approach offers a more efficient lightweight approach to phishing detection.

KEYWORDS

Anti-Phishing, Cyber-attacks, Identity theft, Middleware, Threats

1. INTRODUCTION

Phishing is a significant security threat to the cyber community, a plague that causes tremendous loss every year to both experienced and unwary internet users. The ubiquity of the Internet, the explosion in Web-based applications and the rise of consumer e-commerce have made phishing a profitable business for phishers in the ever-busy cyberspace (Hamid et al., 2014). In a phishing attack, the criminals (called phishers) set up fake websites, and then send phishing emails or embedded a malicious link in a social network communication or send phishing SMS to potential victims, who may be lured to access the phishing sites and expose their sensitive credentials to the phisher. The credentials harvested by the phishers usually include bank account numbers, passwords or PINs, credit card numbers, security questions, security codes etc.

With the harvested credentials, the phishers can log in into the genuine websites to steal the victim's money or launch other related attacks. For instance, RSA's online fraud report showed estimated losses of over $4.5 billion by global organizations in 2014. The report further forecast that the total cost to U.S organization from e-fraud may surpass $ 6.4 billion by 2018. In the same vein, a 2016 security report by Phish-Labs indicated more than one million confirmed malicious phishing sites residing on more than 130,000 unique domains. In another development, the Central Bank of Nigeria White paper estimated that about $250 million was lost to cybercrime in 2013 (Longe, 2014). Users are vulnerable to phishing threat because they are so comfortable navigating Web pages and the URL that name them.

As the number of mobile-connected devices using social networking sites such as Facebook continues to grow, the motivation for phishers to target the platform also increases as malicious links can be easily embedded into e-chat (Aggarwal et al., 2012; Kumar and Kumar, 2014; Orunsolu et al., 2018). Also, the growth of Android and Blackberry activations has increased the attack surface available for phishers while the effectiveness of traditional perimeter defences is low (Verison, 2013; PandaLabs, 2012; Jema et al., 2017). Moreover, Arun (2016) showed that the use of notification sounds and other real-time alerts on smartphones may increase the likelihood of online deception (Arun, 2016). Moreover, the open-source model of web pages makes it easy for attackers to create a replica of a legitimate site (Han et al., 2012; Qabajeh et al., 2018). Because such a replica can easily be created with little cost (e.g. use of toolkit) and looks very convincing to users, many such fraudulent web sites continuously appear (Orunsolu et al., 2017).

To combat the problem of phishing attacks, various countermeasures have been proposed from the complex machine learning system to a simple browser's plugin defence system. However, despite the existence of various countermeasures, the phisher continues evolving new patterns to defeat the current defence system. These challenges are aggravated by the

Quick Response Code	Access this article online	
	Website: www.actainformaticamalaysia.com	**DOI:** 10.26480/aim.02.2020.37.44

Cite the Article: Orunsolu Abiodun, Sodiya A.S, Kareem S.O (2020). Linkcalculator – An Efficient Link-Based Phishing Detection Tool. *Acta Informatica Malaysia, 4(2): 37-44.*

problems of a zero-day attack, high true positive rate, low false positive rate and high computational overhead of some existing solutions. Besides, the problem of intensive administration due to frequent updates limits the application of the anti-phishing system on mobile devices (Sonowal and Kuppusamy, 2018). This development necessitates the need for a simple and efficient anti-phishing system without undue upgrade on the applied devices.

Motivated by extant literature on weblink information analysis, we improved on the concept of these works to design a new anti-phishing system with significant performance improvement and efficient detection approach (Gowtham et al., 2017; Tan et al., 2017). Hence, in this paper, we proposed a simple anti-phishing tool called LinkCalculator is presented. The proposed LinkCalculator anti-phishing scheme is based on an algorithm designed to extract link characteristics from loading URLs to determine their legitimacy. Unlike the other link-based extraction approaches, the proposed approach introduced the concept of the weighting of the incoming request for its prediction without using the machine learning approach. The weighting concept allows the system to prevent superfluous computations on non-essential links within the parsed page. The advantage of this is to reduce the problems of false-positive and negatives occasioned by other methods where this idea is missing. This is because certain link information within parsed webpages or requests is sufficient to classify them as phishing without loss of generality. To this end, the proposed LinkCalculator makes the following contributions to anti-phishing study:

a. This scheme proposes a simple and efficient anti-phishing technique based on links extraction and its associated weight. The weighting technique for extracted links provides the basis for normalizing the contribution of link information against undue skewness of webpage "status". The weighting technique, w, modelled as $\sum_0^1 w_i$, provides a bistate numerical value to indicate the presence or absence of expected link information. The normalizing effect of this prevents abnormal computation of the numerator against the denominator as presented in the algorithm in Section 3.

b. The scheme provides a practical approach that can be easily implemented on a variety of devices ranging from mobile phones to personal computers (PCs). This is possible because the approach does not require any machine learning training and optimization on the user's device.

c. The scheme uses experiments to benchmark the efficiency of the approach. The experimental procedure uses the standard metrics such as True Positive, False Positive etc. to compare with existing approaches.

The rest of the paper is organized as follows: Section II presents a literature review. The overall architecture and design details of our proposed methodology are discussed in Section III. In Section IV, the implementation and evaluation of the proposed method are presented. Conclusions and future works are presented in Section V.

2. TAXONOMY OF PHISHING ATTACKS

Phishing attacks are broadly categorized into two namely social engineering and technical subterfuge-based phishing attacks (Figure 1). In the social engineering-based phishing technique, the phishers try to acquire the victims' credentials by using some fake websites which have the look and feel of the original website through redirection from another website or by sending fake emails or SMS which appear to be legitimate.

The social engineering approach to phishing can be achieved through the following techniques:

Website phishing: This is a kind of phishing attack in which a malicious website assumes the functionalities of a legitimate website to deceive internet users to obtain their sensitive personal information. It typically involves the use of weblink manipulation, browser vulnerabilities, Java Scripts, Pop-ups, fake address bars, malicious use of scripting languages, text and images of legitimate sites. The website phishing is usually the final destination of most phishing attacks where the phishers' payload the on their victims.

Email Phishing: Email phishing is a kind of phishing attack which involves the forgery of an email header so that the message appears to have originated from a legitimate source (Figure 4). The goal is to get the recipients to open and possibly even respond to, a solicitation through tricks such as "verify your account" or "confirm your password" etc. Email phishing is possible because the Simple Mail Transfer Protocol does not provide a mechanism for address authentication in email communication. Although some secure protocols such as Sender Policy Framework Protocol, SenderID, RFC 4954 etc.have been proposed, their adoption has been slow without any significant deployment on an industrial scale (TechTarget report, 2017).

Spear Phishing: This is an email spoofing attack that targets specific organizations or individuals through customized greeting where the name of the victim is usually mentioned, seeking unauthorized access to sensitive information. This kind of phishing attack becomes easier nowadays as more and more personal information is publicly available at online social networks (Li and Schmitz, 2009; Stats and Trends, 2017). Three criteria that determine the success of spear phishing attacks are:

- The source of spear-phishing appears to be a known, credible and trusted individual or organization
- There is information within the message that supports its validity and credibility.
- The request of spear-phishing seems to possess a logical basis or established tradition

Fake promo/adverts: Fraudsters sometimes create an advertisement on Google AdWords for keywords relating to the website they are mimicking. Sometimes, these fake advertisements appear as the first result at the top of the search page, which lead unsuspecting users to a phishing website. Also, phishers spoofed genuine news website and create fake news to discredit individual, sow discord, harvest users' credentials or to generate profit (Digital Shadows Report, 2017)

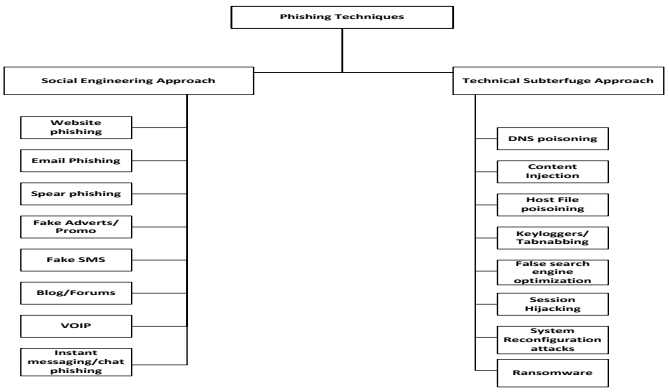


Figure 1. Taxonomy of phishing attacks (Gupta et al., 2016)

SMS phishing/Smishing: This a form of phishing attack conducted via SMS in an attempt to entice a victim into revealing personal information via a link that leads to a phishing website

Blog/Forum phishing: This is a form of a phishing attack in which malicious links are embedded into a blog or forum.

VOIP phishing: This involves making phone calls to a user and asks the user to dial a number or request for particular information. This is mostly done with a fake caller ID

Instant messaging or chat phishing: This is the use of the malicious link in an instant message to conduct a phishing attack.

On the other hand, the technical subterfuge methods use a variety of malicious programs that are installed on the victims' computer unknowingly. It uses methods such as DNS poisoning, content injection, host file poisoning, etc.

DNS poisoning: This a type of attack that exploits vulnerabilities in the domain name system to divert Internet traffic away from legitimate servers to phishing or fake ones.

Content Injection: This is the technique where a phisher modifies a part of the content on the page of a reliable website. This is often done to mislead the users to navigate a page outside the benign website where they (i.e. online users) are prompt to enter their sensitive personal information.

Host File poisoning: This refers to a method in which new entries are injected for a website into a machine's host file which causes the redirection to another website where user's credentials are captured by the phishers.

Keylogger/Tab nabbing: This is a technique in which the user's input is unknowingly recorded by a crimeware installed by phisher after the user may have navigated through a fake page.

False search engine optimization: Some phishing attacks involve search engines that direct an unsuspecting user to certain online shopping sites where the cost of product/services is comparatively lower.

Session Hijacking: This is an attack where a phisher exploits a web session control mechanism to steal information from a user. In a simple form, a phisher can use a sniffer to intercept relevant information so that he can access the Web server illegally.

System reconfiguration attack: This is a kind of phishing attack where a user receives a message asking to reconfigure the computer settings with a web address that appears to be a trusted source. It can either be achieved using a pharming or proxy attack. In pharming attack, the host file of the victim's system is altered so that the target address is redirected to some other malicious location. In a proxy attack, Internet traffic is passed from a victim's machine to a server where a phisher can extract personal confidential information.

Ransomware: This is a malicious software which denies access to a device or file after a user may have opened a phishing email containing such programs until a ransom is paid.

2.1 Related Works

Anti-phishing defence mechanisms in which software are enhanced through design upgrades to mitigate phishing attacks. They are often deployed on the client or server-side for defeating phishing attacks. This main motivation for this method is to bridge the gap that is left due to human error or ignorance (Khonji et al., 2013; Sahingoz et al., 2019). In this section, the review of the current state of the art of anti-phishing research is presented. Common techniques in technical approaches include:

(i) List-based Technique (ii) Heuristics (iii) Machine Learning Algorithms (iv) Visual similarity (v) Search Engine-based method (vi) Web-link information.

These techniques have been used to achieve significant results in the fight against phishing attacks. However, there is some window of vulnerabilities with these methods which still make phishing attack strives (Varshney et al., 2016). For example, PhishNet proposed an active blacklist approach in which new malicious URLs can be effectively predicted from the existing blacklist entries (Prakash et al., 2010). This is achieved by processing blacklisted URLs and producing multiple variations of the same URL using IP address equivalence, query string substitution, brand name equivalence, directory structure similarity and top-level domain replacement. This led to the creation of multiple URLs with the same base domain information. To filter non-existent children URLs, the system performed DNS query, TCP connects, HTTP header response and content similarity. The technique produced promising results when applied to the real-time scenario using blacklist feeds from research dataset. However, the problem of false positives still exists.

In one of the earliest works on the White-list system, a group researchers designed a scheme called Automated Individual White-List (AIWL) (Han et al., 2012). This system allowed users to record their well-known benign sites along with its Login User Information. The LUI maintained by AIWL for any suspicious website includes the URL, the Input Area, and the IPs. This method is very effective against pharming and dynamic phishing attacks. In a similar recent work, Jain and Gupta 2016 proposed the use of an auto-updated whitelist of legitimate sites accessed by an individual user. The approach consists of a transaction matching module (where URL and DNS information of sites are maintained) and phishing detection module. The system was evaluated with 1525 URL instances with an overall true positive rate of 86.02%. However, the False-Positive rate of over 1.0% associated with the approach makes it infeasible for use in a critical online transaction.

Using heuristics technique, some researchers investigated an approach called Remove-Replace Feature selection method (Hota et al., 2018). This method randomly selects a feature from a large feature corpus and remove such a feature if it does not reach a certain accuracy threshold. The approach achieved an accuracy of 99.27%. In a related approach, examined an anti-phishing approach called PhiDMA which can be deployed in a visual impairment scenario (Sonowal et al., 2017). The method consists of auto-whitelist layer, URL features layer, Lexical signature layer, string matching layer and accessibility score comparison. In other related works, proposed a predictive model for phishing detection using frequency analysis of existing feature corpus to create a more discriminative feature class (Orunsolu et al., 2019). The system used an aggregate of 15-dimensional feature set trained using Naïve Bayes and Support Vector Machine. The system achieved a remarkable performance with 99.96% accuracy with low false positive.

A group researchers investigated an anti-phishing system based on visual analysis by examining webpage layouts (Mao et al., 2019). In their method, the authors considered a machine learning scheme that used property vector extraction, property vector generation and comparison vector generation. The scheme achieved an accuracy of more than 93.0%. In a related approach, used machine learning approach to extract logo and utilized the Google image search to determine the identity consistency between the real and the portrayed identity of a URL (Chiew et al., 2015). The approach was evaluated with different categories of websites such as banking, news blogs etc. The accuracy of the system was 93.4%.

A group researcher designed one of the earliest search-engine anti-phishing techniques called Goldphish (Dunlop et al., 2010). In this method, the logo of a site is extracted and converted using the Optical Character Recognition system into text which is subsequently queried in a google search operation. The approach produced a remarkable accuracy of 98%. However, the delay in rendering the webpage image into text is a limiting factor for real-time implementation of this approach. In more recent work, Varshney et al. 2016 investigated a simple search engine anti-phishing system which involved the use of page title and URL alone to uniquely identify the status of a loading page. The approach was implemented as a prototype in a Google Chrome browser with an accuracy of 99.5%.

Based on webpage link information, present a dynamic defence approach in which direct and indirect links associated with a malicious page is generated (Gowtham et al., 2014). In this way, the target domain set is constructed as input into Target Identification algorithm to recognize a phishing page. Using DNS lookup and IP address resolution, the suspicious page can be predicted without the use of machine learning algorithms or existing restriction lists. However, the performance of this scheme is mostly dependent on the search engine optimization, DNS information and its underlying TF-IDF algorithms.

Besides, this approach is inefficient when used for phishing target detection. In related work, the researchers improved on their earlier approach by avoiding much reliance on search engines (Gowtham et al., 2017). In this new approach, the target domains are identified on loading sites and a target domain algorithm is constructed to determine the status of the loading page. The system achieved a remarkable accuracy of 99.53%. This class of anti-phishing technique usually have less overhead in term of memory and training time (Varshney et al., 2017). Besides, the system avoids the large-scale image processing of visual similarity (Jain and Gupta, 2017).

However, existing weblink information systems have the following limitations:

The construction of Target Domain Algorithm can be infeasible on a large-scale deployment (Gowtham et al., 2017). Besides, phishers can evade this method by using shortening service. The reliance of some weblink

information technique on the search engine is susceptible to search engine poisoning (Jain and Gupta 2017; Gowtham et al., 2014).

3. Overview of The Proposed System

The proposed system involves a phase where a user's request is retrieved from a client's browser that is attempting to connect to any resources on a particular webserver. A request, in this case, may be defined as any transaction requiring connection to a particular web server on the Internet. Usually, such request may be represented as a URL of a webpage or a link within any webpage. When a user opens any request in a browser, a module called Service Handler is created using JSoup Java implementation to access a Document Object Model (DOM) tree of the downloaded request from a web browser. The Document Object Model is a World Wide Web consortium standard, that offers a programming interface for web documents. That is, the DOM represents the web documents as nodes and object (i.e. object-oriented platform for web pages) which allows programs and scripts to dynamically connect and update the content, structure and style of documents. The advantage of using the DOM is that it was designed to be independent of any particular programming language and thereby making the structural representation of the web document available from a single consistent API. Upon the creation of DOM, the request can further be tokenized/parsed by JSoup.parse (request) method and getElementById (request) method to extract any useful information e.g. hyperlinks, URL symbols, URL characters etc. in the body of the transaction or webpage. Tokenization is the process through which web document is transformed into a sequence of string characters while parsing is the method of identifying the structure of a document and extraction of data. The parsing and tokenization process helps to further identify the kind of structure within the request e.g. presence of login tags, null links, anchor tag etc. For instance, abnormal status detection can be found on a loading URL by specifying the error code in Jsoup implementation. Algorithm 1 presents the basic function of the Service Handler (SH) in the client/user interface layer.

Algorithm 1: Service Handler Module (SHM)

Input: transaction, t (e.g. URL, links within a webpage etc.), client's browser, b

Output: DOM tree of t

Initialization: httplistener_browser ← invoke_SHM

Read_links_tags ← invoke_SHM (t)

Start: if transaction t ∈ *b* then

HTTP listener of b invoke the SH engine

Do {

String Url = "t"

Document doc = Jsoup.connect (Url). Get ( ) // scrape and parse the HTML of t

If ({ try

{ document doc404 = Jsoup.connect(t-not-found).get () // abnormal status code detection

}

Catch (httpstatus exception ex)

}) return True

Then t is phishing

Else

Element content = doc.getElementbyId ("content")//Jsoup transversing content of t

Element links = content.getElementbyTag("a")// Jsoup transversing tags of t

If (Element links = do.select ("<input tag>") is true)

Then Goto Phishing Detection Layer

If (String linkHref = link.attr("href") is true) **Then Goto** Phishing Detection Layer

SH creates the DOM(t)

If ∄ (links and tags) ∈ DOM(t) then exit

return Secure Connection

Else

}

End do

Exit and Continue

End

The next phase of the scheme consists of a non-computational module which detects phishing without machine learning algorithm using the link features within the parsed page. This is necessary to manage the system resources, memory and time from undue consumption by phishing detection process without justification. This module is especially suited for a range of websites from preapproved sites to websites with unknown/known popularity. The preapproved sites are sites in which the user has signed up for one service or the other. It is observed from extant literature and experience that most users have a limited number of sites where they signed up for such service. On average, an individual maintains more than three signed up sites but most of the times less than ten. For instance, most online users maintain a mailing website e.g. Yahoo or Gmail (or Both), one or two social media accounts e.g. Facebook or Tweeter or Instagram and a service-based website e.g. journal submission site or e-commerce site, etc. On the other hand, websites with known popularity are sites that are used across disciplines by many online users. Such websites are continuously being used by the phisher to deceive unsuspecting online users. Besides, unknown sites are created during different times to deceive online users. For instance, during the death of popular personality or worldwide epidemic e.g. malicious COVID-19 sites.

The following types of links are considered as phishing "fingerprints" in this approach if they are present in a parsed page in a particular specified threshold.

I. A parsed page that does not contain links
II. Parsed page with an extraordinary number of null links defined on prominent tags
III. Parsed page with a high ratio of links directing to the foreign domain than own domain

3.1 Parsed page does not contain links

Hyperlinks are usually defined in a webpage using anchor <a>tag and a trusted or legitimate website has a presence of at least one hyperlink in the body of the HTML of the webpage. Therefore, if the parsed page indicated the presence of login form from the previous module, then the webpage must contain hyperlinks such as sign-on, sign up, forgot password, etc. If such webpage now behaves otherwise, by returning zero counts on the number of links in the webpage then it is phishing. Also, the phisher can insert scripts that will block a webpage with login field from the extraction of its links and making its link count be zero. In these two cases, the proposed system returns phishing status for the parsed page based on a Link Calculator (LC) define in the foregoing section.

3.2 Parsed page with an extraordinary number of null links defined on prominent tags

A null pointer is hyperlinks that are not pointing to any other document or webpage usually defined denoted in href tag as <a href = '≠'>. An anchor tag containing a null value is known as a null anchor. For example, the

execution of the PHP code (Figure 2) can detect the presence of a null tag within a parsed page:

```
<?PHP
    if(isset(html_entity_decode() == none))
      {
echo 'suspect null tag';
      }
?>
```

Figure 2: Pseudocode in PHP for null links

The system uses these characteristics by checking if the parsed page contains lots of null pointers on prominent tags such as confirm password, forgot password, profile, etc. The attackers usually used null anchor because they do not have information about these services and the one way to deceive users is to make them available on the webpage on a null pointer. In the example (as shown in Figure 3 using PHP code), the link on prominent tags can reload without validating the user's login details.

This is usually a malicious practice not known to legitimate webpages. Besides, attackers used JavaScript with null pointers to exploit the vulnerability in a web browser. In this case, if a user scrolls over the link, the link displays something else instead of the actual link to deceive user from noticing the trick. Although this is not a problem in our proposed approach, it is necessary to make the system more robust. Therefore, the proposed approach detects the status of a parsed page by checking if the null pointers on prominent tags exceed certain threshold defined in an LC.

```
<!doctype html>
  <html>
      ……..
      ……..
 <body>
      <form action="" method="post">
<input placeholder="search....." name="" id="" type="text" />
 <button type="submit" name="search" >search</button>
        <a href="redirect.html">more</a>
    </form>
      <?php
      if(isset($_POST['search'])){
          echo '<script type="text/javascript">
window.location.replace("redirect.html");</script>';
    }
```

Figure 3: Malicious pseudocode for redirecting webpages

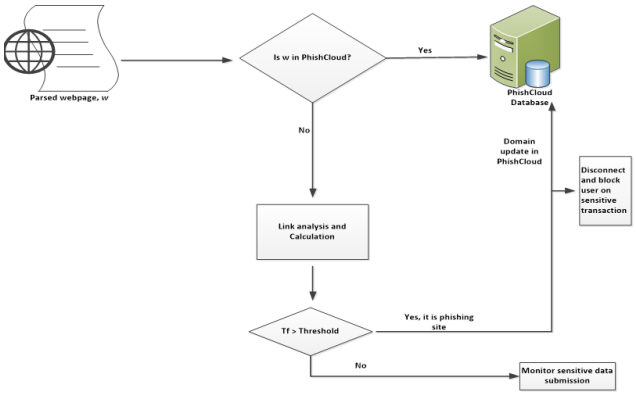


Figure 4: Architecture of Link Calculator system

3.3 Parsed page with a high ratio of links directing to the foreign domain than own domain

Foreign domains are created links to a document file from anywhere on the internet and are usually represented in a foreign anchor using the header function in PHP code such that the time indicates the duration of time for linking up with the foreign anchor whose address is xyz.com

Header ('2s', location: http://xyz.com)

Some webpages may contain too many foreign anchors than hyperlinks pointing to the same domain. This trick is usually found on phishing pages. Therefore, a ratio, R, is defined to determine if the foreign domain on a webpage exceeded its domain-based anchor or tags through the equation that follows and a threshold value is set within the LC to determine the value of R that corresponds to phishing or legitimate activities.

$$R = \frac{\sum F - D}{\sum F} \quad (1)$$

Where D is the aggregate number of links indicating the own domain within the parsed page and $\sum F$ is the aggregate number of links extracted from the parsed page of the investigating webpage.

3.4 Link Calculator

The Link Calculator, LC, is defined on the three types of links considered for determining the status of a webpage in the non-computational module. Algorithm 2 presents a description of how LC works on each of the links. In LC, each link is assigned a configurable value, V_i. The LC analyses the parsed page to decide whether or not a link is applicable for consideration within the subsystem of the non-computational module or not. This is because other complex links within the parsed page can increase overhead if considered within this subsystem. For each applicable link, $C_i = 1$ is used to represent the availability of such link e.g. presence of null links. Otherwise, the value is set to zero. Each applicable link produces a point estimation, P_i, ranging from 0 to 1 to indicate the frequency of the links within the parsed page. The threshold score, T, is an aggregate on the weighted sum of the points assigned by the links divided by a weighted sum of the number of links as defined as:

$$T = 100 \times \left(\frac{\sum V_i C_i P_i}{\sum V_i C_i}\right) \quad (2)$$

For each of the three-link characteristics defined for LC, the immediate result for each link is given as:

$$R_i = T \; \forall \; i = 1, 2, 3. \quad (3)$$

The final threshold, $T_f = \frac{Sum\ of\ R_i}{sum\ of\ C_i}$ determines the status of the investigating page and its value between 0 to 100 and the configurable value helps the system from approaching infinity in the estimation of immediate T value. If the value of T_f is greater than 80%, then the page is considered phishing and the URL of the parsed page is added to the middleware blacklist database called PhishCloud. The threshold is arrived at after initial tuning experiments with various thresholds (e.g. 50%, 60%, 70%) with their resultant false negatives until 80% when the resultant false negatives become negligible. However, if the value of T_f is less than that, then the page is sent to the machine learning module to complete the classification process. Figure 4 presents the architectural outline of the LC.

The PhishCloud provides an immediate database of crosschecking the status of the suspicious page to increase the response time of the proposed system of classified pages as against dependent on phishing archives such as PhishTank where classification takes a few days. This is because a link can be classified only if the input from the user reaches a certain threshold as their approach is based on community rating of links. The PhishCloud does not incur any significant overhead on the browser as the database is only available as a middleware service through pragmatic communication with the browser using a standard protocol. Besides, PhishCloud does not depend on the third-party application like a search engine to be activated as the case of PhishTank, SpamAssassin and other phishing archives.

Algorithm 2: Link Calculator

Input: DOM (t), PhishCloud, Threshold_limit, ∅

Output: unclassified URL, phishing Alert

Start: Check_domain of DOM(t) = Phish_domain ∈ PhishCloud

scan = Compare (DOM (t), Phish_domain)

If (scan) return match, then

URL_DOM (t) = phishing page and exit

Else

{

Calculate the number of links, n(l), ∈ DOM (t)

If n(l) = 0 // no link extracted or found on the parsed page

Compute $T = 100 \times (\frac{\sum V_i C_i P_i}{\sum V_i C_i})$

Continue

Return $R_1 = T$

If n(l) = '≠'// null anchor or pointers on prominent tags

Compute $T = 100 \times (\frac{\sum V_i C_i P_i}{\sum V_i C_i})$

Return $R_2 = T$

If n(l) = $\frac{foreign\ anchor - own\ anchor}{foreign\ anchor} > \emptyset$// foreign anchors exceeded the tolerance level

Compute $T = 100 \times (\frac{\sum V_i C_i P_i}{\sum V_i C_i})$

Return $R_3 = T$

}

P: Final threshold = Phish_Calculator = $\frac{Sum\ of\ R_i}{sum\ of\ C_i}$

If (P) > 80%, Return "webpage is phishing and block connection for the sensitive transaction"

Cached URL_DOM (t) into PhishCloud and update

Else

If (P) < 80%

Return temporary insecure connection block sensitive transaction in the background while this module is activated

End

4. Implementation and Evaluation

The system is implemented using JAVA programming language and its associate libraries such as JSoup HTML Parser (JHP), Secure Socket Extension (SSE), etc. These libraries facilitate efficient link extraction from the querying page. These links are then examined by the application for determining the status of the webpage. Other links such as dead links, static links and external links are also defined within the application module to improve page classification. The dead links are links with no action attached to them which are often used by developers to execute scripts and perform actions. On the other hand, the static links are links that lead nowhere. These links, in most cases, are used by malicious websites who have no content to place under these links, so they use static links in making the user believe they have enough data to run their website. External links are links that lead to other websites that are not part of the website that has been loaded by the user.

Scripts such as file_get _content method are employed to retrieve the details of a website. In this process, the scripts download the source code that is readable by the browser. In the process, the Link calculator picks up the webpage source's code from the browser and examines its legitimacy by checking the page conformity with the malicious script definition within the proposed system. To accelerate the process of loading the URLs into the calculator, a Link Loader app was developed to automate the procedure.

To determine the performance of the proposed system, the scheme employed metrics such as True Positive Rate, False Positive Rate, True Negative Rate and False Negative Rate. These metrics evaluate the ratio of correctly classified phishing page and legitimate page from standard available dataset such as PhishTank, Alexa, Millersmiles etc. The TP is the rate of phishing instances that are correctly predicted as phishing out of the total phishing instances while the FP is the rate of phishing instances that are misclassified as legitimate out of the total phishing instances. On the other hand, the TN is the proportion of experimental instances with a known phishing status for which the experimental result is negative (i.e. phishing) while the FN is the proportion of experimental instances with known legitimate status for which the experimental result is negative (i.e. phishing)

4.1 Experiment for Performance assessment

The proposed LinkCalculator is evaluated by using experimental dataset instances from PhishTank and Alexa. The dataset consists of 150 phishing URLs from PhishTank (www. phishtank.com) and 150 benign URLs from Alexa (www.alexa.com). The small dataset is used to ensure that each entry into the proposed application is unique to avoid repetitive processing of the same URL more than once. Figure 5 shows the interface where URLs are manually fed into the application to determine the status of the webpage. As the URLs are fed into the application, the system invokes the appropriate module to crawl all the links within the URLs to determine its status whether phishing or legitimate. An example of such analysis is shown in Figure 6. The experimental results for the entire testing dataset indicate that the system returned 100% True Negative Rate and 0.00% False Positive Rate for legitimate instances and True Positive Rate of 96.67% with 0.03 % False Negative Rate for phishing instances. Figure 7 provides the graphical illustration of the experimental results. These results are compared with other existing works as presented in Table 1.

Table 1: The experimental results for the entire testing dataset

Works	TP	FP	TN	FN
Jain and Gupta, 2017	96.10	0.125	99.95	0.90
Gowtham et al. 2017	99.53	0.45	99.54	0.46
Tan et al. 2017	99.20	7.80	92.20	0.80
Kaur et al. 2016	99.44	4.26	95.74	0.56
Our work	96.67	0.00	100	0.03

These results indicate the superior performance of the Link calculator over other existing link analysis methods. For instance, Jain and Gupta used the Hyperlinks IP matching and No hyperlink link attributes of querying page to detect phishing (Jain and Gupta, 2016). This method is very similar to our approach except for the concept of threshold which reduces the false positives associated with our approach. The false positive in Kaur et al. 2016 is somehow which can limit the application of the approach in critical web transaction.

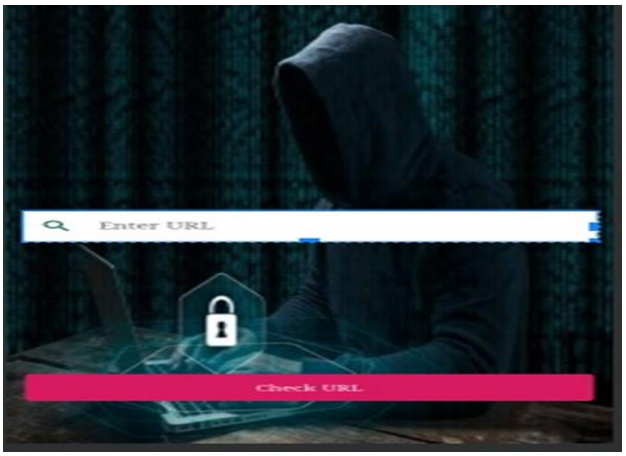


Figure 5. URL check interface

4.2 Experiment for Efficiency assessment

The efficiency of the proposed system is assessed by using Precision, Recall and F1-score. The precision, p, is used to measure the rate of phishing instances which are identified correctly as the instances detected as phishing. That is, p is the number of correct positive results divided by the

number of all positive results returned by a system. The Recall, r, is a measure of phishing instances identified correctly as existing phishing instances. This implies that r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1-Score is the harmonic mean of Precision and Recall and F1-score reaches its best value at 1 and worst at 0.

Figure 6: Analysis interface of a typical URL

The efficiency assessment of the proposed system indicates a precision rate of 100%, a recall of 99.96% and F1-score of 99.97%. These values are compared with other related works to justify the significance of the proposed system (Table 2) (Orunsolu et al., 2019; PhiDMA, 2017; Mao et al., 2019).

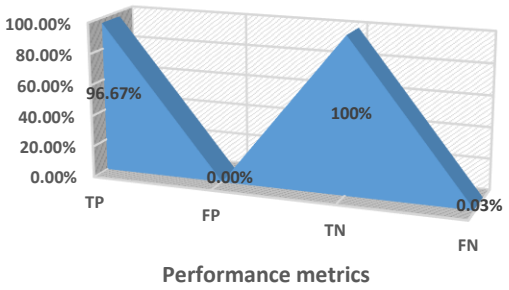


Figure 7. Performance analysis of Link calculator

Figure 8 presents the graphical illustration of the efficiency assessment of the proposed system in comparison with other works. Most of these works use Machine learning approach using classifiers. For instance, some researchers used ensemble classifier on web layout technique to detect phishing while PhiDMA used a 5-layered method based on string pattern matching and accessibility score to detect phishing (Mao et al., 2019). Similarly, some researchers used some features obtained through frequency assessment technique on Support Vector Machine (SVM) and Naïve Bayes (NB) to detect phishing (Orunsolu et al., 2017). However, despite the machine learning application by these approaches, the LinkCalculator method offers superior machine without the attendant machine learning computations of the other methods.

Table 2: Efficiency assessment of the proposed system in comparison with other works

Work	TPI	PTD	ZDP	EPS
Gowtham et al. 2017	No	Yes	Yes	Yes
Orunsolu et al. 2019	No	Yes	Yes	Yes
Jain and Gupta 2017	No	Yes	Yes	Yes
Kaur et al. 2016	No	Yes	Yes	Yes
Our approach	Yes	Yes	Yes	Yes

4.3 Attribute Comparison with existing methods

Table 3 presents the attribute comparison of the proposed technique with the existing anti-phishing methods. The comparison uses the following attributes:

i. Third-party Independence (TPI): This is the ability of a system that depends on how many external resources (such as search engine results) that are incurred to obtain a correct prediction.

ii. Phishing Target detection (PTD): This is the ability of the anti-phishing scheme to determine the target domain that a phishing page is trying to mimic.

iii. Zero-day protection (ZDP): This is the ability of an anti-phishing scheme to detect a newly launched phishing activity on the webspace.

iv. Efficient Protection Scheme (EPS): This is the ability of the anti-phishing method to return remarkable evaluation results for various experiments carried out to benchmark the approach against existing methods.

5. Conclusions

In this paper, we present the architecture of a simple anti-phishing tool called LinkCalculator. The proposed LinkCalculator uses a simple algorithm designed to extract link characteristics from loading URLs to determine their legitimacy. Unlike the other link-based extraction approaches, the proposed method introduced the concept of weight into incoming web link information for efficient representation. This is because certain link information within parsed webpages or requests is sufficient to classify them as phishing without loss of generality. The approach was implemented using several libraries in JAVA programming language to retrieve and parse the loading webpage to determine the volume of link manipulation within the querying page to verify its legitimacy. The approach was evaluated using experiments that determine its performance capability based on standard metrics such as False Positives rate, True Positive rate etc. The experiment results were compared with other existing works with a clear indication of the superior performance of the proposed system.

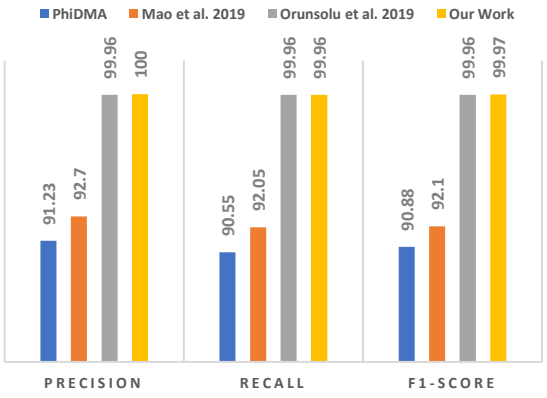


Figure 8: Comparison of Efficiency Assessment

In the future, we hope to investigate the approach on a large dataset using an incremental link analysis method. In this method, we hope to investigate the effectiveness of individual link-based feature in the overall performance of the system. Also, the problem of link evasion by phisher needs further investigation to prevent null-return by the proposed scheme.

References

Adebowale, M., Lwin, K., Sanchez, E., Hossain, M., 2018. Intelligent Web-Phishing Detection and Protection Scheme using integrated Features of Images, Frames and Text. Expert System with Applications.

Chiew, L., Chang, H., Sze, N., and Tiong, K., 2015. Utilization of website logo for phishing detection. Computer and Security Journal.

CSO Online report on phishing activities. Accessed 2016 (www.csoonline.com/articles)

Dunlop, M., Groat, S., and Shelly, D., 2010. GoldPhish: using images for content-based phishing analysis. In: International conference on

internet monitoring and protection. Barcelona, Spain, Pp. 123–128.

Gowtham, R., Gupta, J., and Gamya, P.G., 2017. Identification of phishing web pages and their target domains by analyzing the feign relationship. Journal of Information Security and Applications, 35, Pp. 75-84.

Gowtham, R., Krishnamurthi, I., 2014. PhishTackle-a web services architecture for anti-phishing. Cluster Comput.

Hamid, A., Abawajy, J., 2014. An approach to profiling phishing activities. Journal of computer and security. Elsevier Press.

Han, W., Cao, Y., Bertino, E., Yong, J., 2012. Using automated individual white-list to protect web digital identities. Expert Systems with Applications.

Hota, H.S., Shrivas, A.K., Hota, R., 2018. An Ensemble Model for Detecting Phishing Attack with Proposed Remove-Replace Feature Selection Technique. International Conference on Computational Intelligence and Data Science. Procedia Computer Science, 123, Pp. 900-907.

Jain, A., Gupta, B., 2017. Two-level authentication approach to protect from phishing attacks in real-time. J. Ambient Intell Human Comp., DOI 10.1007/s12652-017-0616-z

Jain, A.K., Gupta, B.B., 2016. A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP J Inf Secur., Pp. 1–11.

Mao, J., Bian, J., Tian, W., Zhu, S., Wei, T., Li, A., Liang, Z., 2019. Phishing Page detection via classifier from page layout feature. EURASIP Journal of Wireless Communication and Networking. 43,

Orunsolu, A., Sodiya, S., Akinwale, A., 2019. A Predictive Model for Phishing Detection. Journal of King Saud University-Computer and Information Sciences.

Phishtank dataset, 2018. http://www.phishtank.com.

Qabajeh, I., Thabtah, F., Chiclana, F., 2018. A recent review of conventional vs. automated cybersecurity anti-phishing techniques. Computer Science Review.

Sonowal, G., Kuppusamy, K.S., 2017. PhisDMA- A phishing detection model with a multi-filter approach. Journal of King Saud University.

Tan, C., Chiew, L., Sze, N., 2017. Phishing Webpage Detection Using Weighted URL Tokens for Identity Keywords Retrieval. Lecture Notes in Electrical Engineering, 398.

Varshney, G., Misra, M., Atrey, K., 2016. A phish detector using lightweight search features. Comput Secur., 62, Pp. 213–28.

Varshney, G., Misra, M., Atrey, P., 2016. A survey and classification of web phishing detection Schemes. Security Comm. Networks.

Wikipedia, www.wikipedia.com, accessed 2018.