Journal of Intelligent & Fuzzy Systems 32 (2017) 3273–3284
DOI:10.3233/JIFS-169270
IOS Press

Parallel phishing attack recognition using software agents

S. Sarika[a,*] and Varghese Paul[b]
[a]*Department of Computer Science, Bharathiar University, Coimbatore, India*
[b]*Department of Information Technology, Cochin University of Science and Technology, Cochin, India*

Abstract. Internet phishing has become a continual threat that keeps growing day by day. Phishing takes advantage of the user's trust and use social engineering techniques to deceive them. Despite having several anti-phishing strategies, the threat of phishing is not mitigated as modern types of attack keeps coming to the fore. Nowadays, phishers launch sophisticated phishing attacks which tricks users to submit their credentials by leveraging the facilities tabs offer to web browsers. Browser security is important in this perspective. This paper explains a heuristic method to defend phishing attacks by utilizing software agents for parallel attack recognition. The main focus is on a browser based attack called Tabnabbing which takes action in inactive browser tabs. The proposed method uses agents in three levels to continuously monitor the presence of attack in regular intervals at multiple tabs and warn the user at the earliest. This approach also protects the users against URL obfuscations and malicious links. Results show that the proposed method outperforms the state of the art phishing detection methods and achieves an accuracy of 97.3%.

Keywords: Tabnabbing, intelligent agents, internet security, antiphishing

1. Introduction

As Internet has claimed a major role in our daily life, criminals have become very active in stealing sensitive information using phishing websites. Phishing [19] is an art of deception wherein secure websites are so perfectly impersonated that even cautious users are tricked. There has been a greater focus on the subject of securing web services with increase in the use of the Internet for online transactions. Through the last decades, the web has become more and more client centric, where the browser has turned out to be a major platform for sophisticated web applications. Browsers display the web pages using an underlying web protocol called Hyper Text Transfer Protocol [25]. Eventhough HTTP allows for the quick and easy transmission of information, it is not secure as there is a possibility for someone to eavesdrop to the conversation between servers and clients.

In order to ensure secure transactions, websites use HTTPS (secure HTTP) rather than HTTP in its address. However, even if a site address displays HTTPS, it might still be a phishing web page as there are spoofing techniques [36] to make a bogus webpage appear to be using HTTPS protocol. Attackers can carry out several attacks within the web platform using scripting support [11], loopholes in software and design of web browsers.Some of the known web browser security risks include "social engineering", "clickjacking", "session hijacking" and cross domain vulnerabilities [22] like "cross site scripting" [37] and "cross site request forgery". These techniques run the gamut from simple eavesdropping, through theft of identity and personal information, to financial losses. Despite having a number of Internet security technologies available, none of them provide enough

*Corresponding author. S. Sarika, Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore 641046, India. Tel.: +91 9947948987; E-mail: sarika.anand08@gmail.com.

security to Internet users. In this regard, a new technology using intelligent agents is being introduced to increase the efficiency and security of the Internet, which may replace the existing technologies. Also, the same problems are more and more common on smart phones. As for the "fixed" world, also on such devices, there are several counter measures mainly based on Cryptography such as the ones presented in [8, 9, 13, 16, 26, 41].

The power of artificial intelligence can be utilized for a large number of applications and environments by creating mechanical maids behave like humans. The computational agents with intelligent behavior can be made available on the Internet for making Internet browsing secure.

This paper focuses on a browser-based attack called tabnabbing and a novel approach for its parallel detection in multiple tabs using distributed software agents. Tabnabbing [3] is a phishing attack within a browser and it targets a user who keeps many tabs open at a time. As the user navigates through a bunch of open tabs, phishers set up a rogue website which looks exactly like the genuine one and load the inactive tab with the counterfeited webpage. When the user switches back to the tab, it appears to be a site frequented by the user. As the user does not remember how each tab looked like before tab switch, he will give his credentials to the honest looking page and is trapped. Unlike other attacks, this deception technique is likely to betray even the most security-conscious web surfers as it exploits user's trust and inattention in browser tabs. The attack relies on human memory weakness and masquerades the favicon, title, and layout of a webpage familiar to the user. The proposed method uses structural features of a webpage to combat tabnabbing.

The rest of this paper is structured as follows. Next section describes related work. Section 3 gives an overview of multi agent systems. Section 4 discusses the system model for phishing attack recognition. Section 5 describes the methodology of the proposed framework. Section 6 discusses implementation followed by experimental results. Discussion part is included in Sections 7 and 8 concludes the paper.

2. Related work

Phishing is a social engineering attack to illegally acquire and use someone else's data on behalf of a legitimate website for financial or personal benefit. Phishing attacks target the human factor in browsing for evasion. Recently, online attacks are unbelievably widespread and use mechanisms that exploit weaknesses found in end-users. Adversaries employ a large number of spoofing tricks such as URL obfuscation, embedding malicious links and hiding dangerous codes to make a phishing web site look innocent to the victim. Truly speaking, there is no single silver-bullet solution to resist all the attacks effectively, thus multiple techniques are required to mitigate specific attacks. Also, as soon as we become competent to identify a particular type of attack, another more sophisticated version appears. So, phishing detection is a complicated and continuous process which should be handled technically using appropriate security measures implemented either at client or server side.

Among the existing strategies, the basic phishing detection method is a list-based approach (blacklist and whitelist). In blacklist approach, the system keeps a pre-compiled list of URLs which were found to be malicious at some point in time. In [29], Pawan Prakash et al. have proposed a heuristic and blacklist based approach called PhishNet, to detect phishing attacks. The method comprises a URL prediction component and an approximate URL matching component. Phishnet is effective at finding new URLs that were not part of the original blacklist and offers low false negative and false positive rates.

The whitelist based solution keeps a list of legitimate URLs that prevent access to phishing sites by URL similarity check. Some of the methods which use whitelist is explained in [1, 20, 39, 42]. The list based methods need the list to be periodically updated as it may become obsolete too soon. In order to solve this problem, different heuristic methods [1, 28] are proposed which uses characteristics of the webpages and URL to identify phishing sites. Heuristic methods often use machine learning methods for classification as explained in [2, 38, 45]. However, heuristic and machine learning techniques might fail when attackers host phishing attacks on servers and also they cannot detect the phishing sites designed fully with images. In [23], Juan Chen has explained LinkGuard, a character based antiphishing approach which utilizes the generic characteristics of the hyperlinks in phishing attacks. This technique is inefficient as it may create more false positives. Another method SpoofGuard [36], extracts phishing signatures via suspicious URLs, images, links, and passwords in a webpage. The approach is easy to evade as it cannot handle images with modifications.

Phishers often design fake pages with layout and content similar to a legitimate page to deceive users. The content-based approaches, analyzes the HTML code and text on a webpage, proved effective in detecting phishing pages. Zhang et al. proposed a content based solution, CANTINA [43] which uses TF-IDF information retrieval algorithm for phishing detection. The adversaries can evade this technique by using images and invisible text in webpages. GoldPhish [24] is another content based approach which captures the image of webpage and uses optical character recognition to convert the image to text. This method provides zero day phishing but is vulnerable to attacks on Google's PageRank algorithm and Google's search service. The content based methods are susceptible for attacks launched using fake pages with same visual layout as the genuine page. Nowadays, phishers have started compiling phishing pages with non-HTML components, such as images, Flash objects, and Java applets so as to beat existing techniques. In order to deal with it, some visual similarity based methods such as [4, 10, 14, 40] are proposed which considers the visual appearance of a website for similarity assessment.

The aforementioned approaches focus on the older variants of phishing. Tabnabbing is a modern browser security threat and its current management modalities are briefly explained below.

NoScript [27] and YesScript [44] are Firefox addons, preventing websites from running JavaScript [11], Java, Flash or other plugins. But they do not provide protection in other browsers. NoTabNab [34] is a Firefox add-on proposed by Unlu and Bicakci which protects users from tabnabbing attack by using the positioning of HTML elements of a webpage. The key problem in this technique is related to resizing the browser, as only some web pages are designed to re-layout themselves.

Suri et al. has presented a signature based detection mechanism [31] to deal with tabnabbing. The method defines a set of rules to scrutinize vulnerable JavaScript code. But this paper focuses only on iframe elements which is not always necessary for a tabnabbing attack.

Tab-Shots [12] is a browser extension which uses visual appearance of a webpage to detect tabnabbing. The method works by remembering what each tab looked like, whenever a tab is changed by recording the favicon and screenshots of the presently focused tab at regular time periods. The main limitation of this technique is the inefficiency in detecting small changes in a page.

TabsGuard [15] combines heuristic based metrics and data mining techniques to detect tabnabbing. The approach keeps track of the changes made to the structure of a page during the time when the page is idle.

Existent anti-tabnabbing methods detect the layout change and warn the user only when the tab is on focus after being nabbed and they focus on the change in page layout, title and favicon, not much attention is given to change in URL. Our approach is similar to the method proposed by Fahim et al. [15] where structural features of a webpage are analyzed but the selected features are different. The proposed method uses agents to concurrently monitor the change in webpage layout at regular intervals in all the tabs of a browser and alerts the user about the attack wherein he can act accordingly. The method also provides a mechanism to monitor fraudulent URLs and thus combat three types of phishing attacks simultaneously. The major contributions of this paper are:

- A multi agent based framework that combines blacklisting and heuristic-based approaches.
- An effective mechanism for simultaneous detection of phishing in multiple tabs.
- A security model to protect users from tabnabbing attack, URL obfuscations and malicious links by giving explicit warnings.

3. Multi Agent Systems (MAS)

People have always been fascinated with the idea of non-human agencies. It will be quite interesting if we can complete our task using a mechanical maid with artificial intelligence, capable of exhibiting human expertise. Here comes the significance of agent technology. The idea of an agent originated with John McCarthy in the mid-1950 s. As a definition, agents [7] are software entities that assist people and act on their behalf. The awesome power of agents has engendered a lot of excitement in recent years because of its efficacy as a new paradigm for solving critical problems. As the technology matures and addresses complex and sophisticated problems, the need for systems that consist of multiple agents become apparent. The MAS approach [35] seems to be the most feasible solution for such scenarios. Here multiple agents in a system co-ordinate with each other to solve some interdependent problems.

The significant properties of MAS are:

– Autonomy: Each agent is an autonomous module with different capabilities or functionalities

and can work in concert with other agents to achieve a variety of goals.
- Parallelism: Simultaneous working of agents is possible in the MAS approach. A complex problem could be solved in a reasonable time by using a number of agents in parallel.
- Adaptivity: The agents in MAS are able to learn and improve with experience. They react in a flexible manner according to the change in environment. The agents can assist each other to compensate any lack of capability or knowledge. They cooperate and share information with other agents to resolve a newly emerged problem, if one agent is unable to do so.
- Reactivity: The agents are able to selectively sense and act in Multi Agent System. Each agent thinks and acts locally to achieve its goals.

4. System model

The proposed method is a modular, multi-agent architecture, where the webpage activities are monitored and controlled by deliberative BDI (Belief, Desire, and Intention) agents [33]. The agents are hierarchically organized with the possibility to share and delegate activities and/or responsibilities. Multi-agent systems are incorporated to bring modularity and parallelism where all the computing tasks are delegated to the agents. The agent platform integrates a set of agents with specific functionalities. As the distributed Multi Agent platform is designed by means of modeling the functionalities of agents, all communications take place through agents thereby reducing the system control. The agents communicate in the platform using FIPA Agent Communication Language [6]. FIPA guarantees interoperability between the agents by coordinating communication and management of agents.

As shown in Fig. 1, the proposed framework consists of four operational agents when a webpage is opened in a browser tab. The agents in this system are:

- T-agent
- U-agent
- M-agent
- I-agent

Whenever a new tab is opened, fresh pair of level 1 agents (U-agent and T-agent) are created to observe the activities in that tab to detect a malicious nature. When the tab is closed, they are disposed off. Level one agent is managed by a level two agent (M-agent).

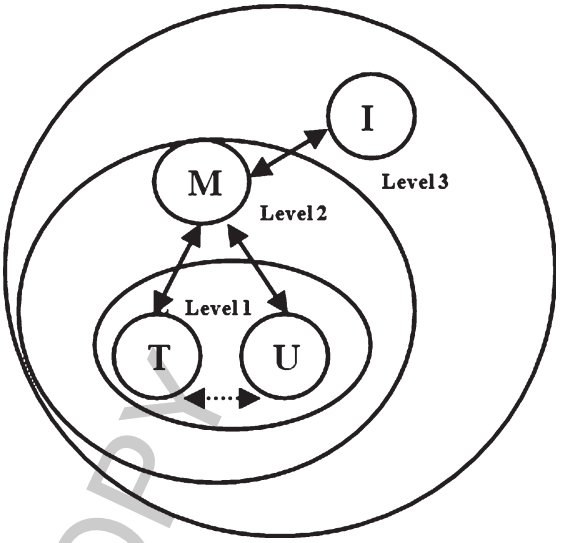

Fig. 1. Hierarchy of agents.

The interface between the user and the system is provided by level three agents (I-agent).

T-agent or Tabnab agent is a level 1 agent responsible for handling incoming requests from browsers for webpages. The T-agents in each tab performs its delegated task in two phases, Feature Extraction and Feature Comparison for detecting tabnabbing in a webpage.

When a webpage is loaded, T-agent extracts its five tuple information (text, image, URL, title, favicon) and record for the next phase. The procedure is continued every 60 seconds. The values from subsequent feature extractions are matched with the recorded values to obtain a resemblance score for each pair of elements. If the resemblance score is higher than a threshold t, the currently visited web page is considered as similar to the recorded one.

U-agents work when the URL of the webpage is changed after a tab switch event or inert tab. The technique uses a URL blacklist to find fraudulent URLs. Blacklisting works on the basis of a pre-compiled list of URLs which are found to be malicious at some point of time. The U-agent queries the URL blacklist to determine whether the currently visited URL is on this list. If the URL is included in the black list, the user is given an explicit message. Otherwise, the URL is given a structural analysis to check whether it is fake or genuine.

M-agent acts as a manager who is responsible for coordination, communication, decision-making and its evaluation. This agent evaluates the decisions taken and immediate actions follow.

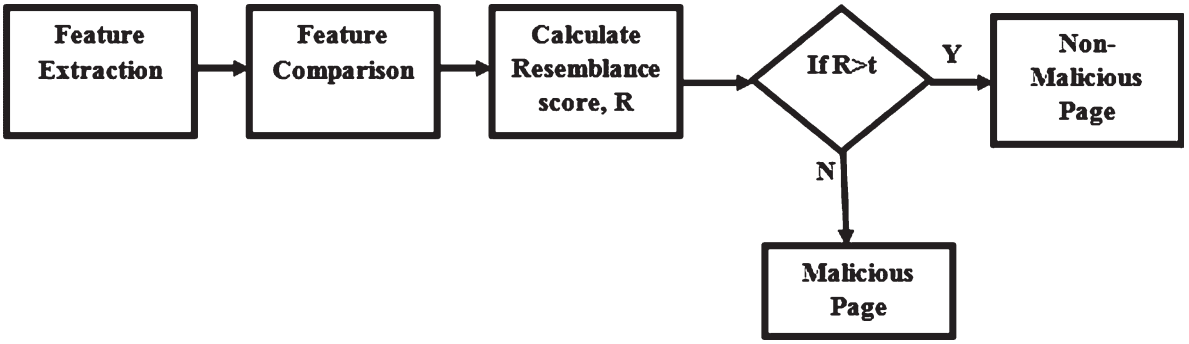

Fig. 2. Functionality of the system.

I-agent is an interface agent which deals with the interaction of the user with the system. The interface agent communicates to the user about the detection of attack and displays a proper message. They can act autonomously to perform operations without explicit directions from the user.

5. Methodology

The operation of the proposed system is illustrated in Fig. 2 and is summarized as follows:

5.1. Feature extraction

Feature extraction is a quantitative way of capturing a set of features that describe various aspects of a page. These features cover text, image, URL, title and favicon of the current page. During the first pass, T-agent stores these values for later use.

SAX parser [21] is used for text extraction as it is an effective mechanism to parse the webpage.SAX is an event based parser, which support more simple forms of interaction with the data and allows handling of larger documents. SAX processing is attractive as it does not load any XML documents into memory. Therefore it is lightweight and fast.

Concerning image extraction, the approach extracts the source address of the image src attribute which can be obtained from the SAX parser output, the area occupied by the image in pixel and its position in webpage, and RGB color histograms. The hyperlinks in the webpage are also extracted and stored to verify if they are mischievous. Feature extraction process is explained formally in Algorithm 1.

Algorithm 1. extractFeatures

Input: Webpage w
Output: Extracted features of w
1. Open the webpage w
2. Parsecount = 0
3. U <-webpage URL
4. T <-Parse w using SAX
5. H <-hyperlinksFilter(T)
6. Ti <-getTitle(T)
7. extractFavicon(w)
8. extractImages(w)
9. Parsecount++

5.2. Feature comparison

Feature comparison is performed by comparing matching elements separately. The syntactic similarity of two text documents $d1$ and $d2$ are calculated to get a resemblance score Rt, which is a number between 0 and 1. The resemblance of the corresponding documents can be computed in time linear in the size of the sketches [5]. In this method, each document is viewed as a sequence of words, and start by lexically analyzing it into a canonical sequence of tokens. A set of subsequences of tokens $s\ (d,\ n)$ are associated with every document d. A contiguous subsequence contained in d is called a shingle. For a given shingle size, the resemblance Rt of two documents $d1$ and $d2$ is defined as:

$$Rt(d1,\ d2) = \frac{s(d1) \cup s(d2)}{s(d1) \cap s(d2)}. \qquad (1)$$

Then, compare all image elements to obtain a resemblance score Ri. Comparison of each image is performed as follows:

- Comparisons of source address of the image src attribute

The resemblance between the two src attributes is computed using the Levenshtein distance.

- Comparison of RGB color histogram

The resemblance between the two matrices representing the color histograms H and H^1 using 1-norm distance as

$$1 - L1(H,\ H^1). \tag{2}$$

- Comparison of pixel positions occupied by the image.

The resemblance between the two positions in a webpage is computed as

$$1 - (\mathrm{d}/\mathrm{M_d}), \tag{3}$$

where d is the Euclidean distance between the two points and M_d is the maximum Euclidean distance between two points.

- Comparison of the area occupied by the image in pixel

The similarity between the two images areas A and A^1 are calculated as:

$$\left[1 - \frac{|A - A^1|}{\mathrm{Max}(A,\ A^1)}\right]. \tag{4}$$

Using these four scores, a single resemblance score $Ri \in [0, 1]$ is derived. The webpage addresses are monitored and recorded in regular intervals to obtain a resemblance score Ru. Favicons are compared by source to get a resemblance score Rf. These are indicated using Boolean values 0 and 1, where 0 means no resemblance and 1 means perfect match. Title of the webpage is matched with the old one to find a resemblance score Rti. Finally, the overall resemblance of the two pages are calculated as

$$R = Rt + Ri + Ru + Rf + Rti. \tag{5}$$

The resemblance score is greater than a threshold t, for similar pages. If there is a radical change, the user is informed about the presence of an attack by displaying an alert message. A desirable value of threshold t is to be chosen to identify the existence of an attack. A higher threshold is preferable in this case as tabnabbing may change the aforementioned parameters to launch an attack. Pseudocode for feature comparison is given in Algorithm 2. T-agent computes the overall resemblance score after getting the value of Ru from U-agent.

Algorithm 2. compareFeatures

```
Input: Webpage w
Output: Resemblance score
    1. Add new TickerBehaviour to T-agent for every 60 seconds
    2. extractFeatures(w)
    3. if (Parsecount >1) then
    4.   Rt <-compare Text()
    5.   Ri <-compareImages() //from stored directory
    6.   Rf <-compareFavicon()
    7.   Ru <-compareURL()
    8.   Rti <-compareTitle()
    9.   R = Rt + Ri + Rf + Ru + Rti
    10.  return R
    11. else
    12. return false
    13. end if
```

5.2.1. Structural analysis of URL

During the feature comparison phase, when the stored URL and currently visited URL are found to be different, the U-agent checks whether the recently visited URL is blacklisted. If the URL is included in the black list, the user is advised accordingly. For the blacklist to work properly, it should ideally contain every phishing website, which is impossible. As a result, it can lead to a number of false positives. So, the webpage addresses that are not blocked by the blacklist are given a structural analysis in which 25 salient features are selected from the doubtful URL and a total score is calculated. Occurrence of each feature in URL will add one to the total score of the URL check. If the score is above a certain threshold, the page is marked as phishing. The default threshold is three detections. Algorithm 3 shows the various steps in evaluating the URL of a webpage.

Algorithm 3. behaviourofURL

```
Input: Webpage w, Blacklist BL
Output: URL Check result 0: Legitimate
                         1: Phishing
    1. if URL is changed then
    2.   if changed URL in BL then
    3.   return 1
    4.   else
    5.   extractURLfeatures(U)
    6.   URLCheck()
    7.     if URLCheckscore >=3 then
    8.     return 1
    9.     else
    10.      return 0
    11.     end if
    12.  end if
    13. else
    14. return 0
    15. end if
```

Table 1
Feature types and its count

Feature Type	Count
Lexical	5
Token based	10
Target based	10

The result of URL check is forwarded to T-agent for further steps. If the result of URL check is less than 3, the referred URL is reliable and returns the value 0 to T-agent otherwise return 1 to convey that the URL is not reliable.

For structural analysis, the proposed method uses 25 features selected by observing the heuristics in the structure of phishing URLs and also by referring literature [17, 32]. As shown in Table 1, there are 5 lexical features, 10 token based features and 10 target based features.

- Lexical Features

The lexical (textual) features help us to identify that malicious URLs tend to "look different" from legal URLs. The approach has chosen 5 lexical features by analyzed the composition of phishing URLs in PhishTank.com. The lexical features include digit in host, IP address in URL, number of suspicious characters '@', number of dots in path and length of the URL.

- Token Based Features

The malicious URLs may contain some eye catching keywords or tokens to attract end users. The selected 10 keywords include login, signin, update, verify, secure, banking, webscr, dispatch, cgi and account.

- Target Name Features

From PhishTank data archive, an analysis was done for different monthly stats archive and collected top 10 brands used by fraudsters during the period from June to December 2015. The most popular target was Paypal. There were 4103 valid phishes against this site. The other targets include Apple, AOL, Facebook, eBay, Google, JPMorgan, WellsFargo, WalMart and Bradesco.

6. Implementation

The implementation of the proposed method uses JADE software framework [18] in java platform. Google chrome was selected as the browser as it is vulnerable to modern type of attacks. The method currently has a simple user interface, displaying an alert message to the user if a webpage is deemed as phishing.

6.1. Experimental evaluation

Two experiments were conducted to assess the performance of our agent based method. In the first experiment, we examined the effectiveness of our multi agent architecture for detecting tabnabbing. The second set of experiments is conducted to analyze the performance of the approach in identifying URL obfuscations and suspicious links in a webpage. Finally, we evaluated the overall effectiveness of our method by comparing with existing techniques in terms of accuracy and precision.

6.1.1. Tabnabbing attack detection

In this experiment, we evaluated how much effective our agent based method was in detecting tabnabbing. The data set consists of a set of common webpages with login forms such as banking sites, web mail clients, credit cards, and social networking sites as tabnabbing targets webpages which can provide confidential information of users. The approach used 1000 unique webpages with login forms from different sources for attack recognition as shown in Table 2. To make a list of blacklisted URLs, a collection of real phishing sites from PhishTank were taken.

Tabnabbing attack was simulated in these webpages by running a script. The simulation of attack used eight tabnabbing pages from different categories like social networking, email, banking and money transaction sites with the appearance similar to the real ones (Facebook, Twitter, eBay, Gmail, Hotmail, Paypal, Citibank and Bradesco). Tabnabbing attack was simulated in these webpages by running a script.

For the simulation, the agent platform is started and the webpages were loaded in the dataset in different tabs of the browser. The script was run for the currently focused window and a tab switch was performed to some other window. When the user returned to that inactive tab after 60 seconds, the webpage had changed to a tabnabbing page (already created). A time interval of 60 seconds is set with

Table 2
Sources of dataset

Sources	No. of webpages	Percentage
Alexa	270	27%
Banks	530	53%
DMOZ	200	20%

an assumption that a phisher may take at most 60 seconds, to reload the inactive page with a new look.

During this time, the relevant features from the webpages opened in various tabs are captured and recorded (feature extraction phase). The feature extractions conducted further in every 60 seconds use this recorded value for comparison phase. The output of feature comparison is a resemblance score of the original webpage with its currently opened version. In the case of text, images and title, percentage of similarity is considered. The similarity in webpage address and favicon are indicated using Boolean values. The resemblance score of webpages in the dataset are calculated for the eight tabnabbing pages. This process is continued with all the webpages in the dataset.

In order to separate legitimate and phishing pages, the resemblance score set is partitioned according to a threshold value t. In this framework, the value of t is set to 4 to get an accurate result. If resemblance score is greater than 4, the webpage is considered as genuine, otherwise as phishing and an alert message is displayed to the user.

The effectiveness of the method is assessed using false positive rate and true positive rate. FPR and TPR for various threshold values are computed using the below given formula and is shown in Fig. 3.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{(\mathrm{FP} + \mathrm{TN})}. \tag{6}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{(\mathrm{FN} + \mathrm{TP})}. \tag{7}$$

In the framework, phishers cannot influence the false positives as legitimacy of a site is directly proportional to the resemblance score, but there may be false negatives if a desirable value of threshold is not chosen. As it does not raise any false alarms, FP in the method is close to 0. Concerning security

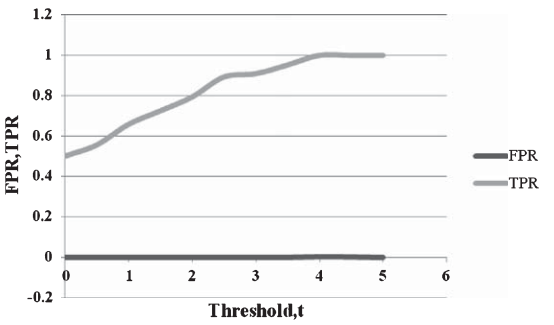

Fig. 3. False positive rate and True positive rate in various thresholds.

Table 3
Evaluation results

Threshold	Accuracy	Precision	Recall	F1 measure
0	90	98.7	91.1	94.7
0.5	90	98.8	91.0	94.7
1	90	98.9	90.9	94.7
1.5	91	99.6	91.3	95.2
2	92.8	99.6	93.0	96.2
2.5	93.5	99.6	93.7	96.5
3	95	99.6	95.1	97.3
3.5	96.8	99.8	96.8	98.3
4	97.3	99.9	97.2	98.5
4.5	96	99.8	95.9	97.8
5	96	99.7	96.1	97.8

as the major preference, the threshold value t should be tuned to minimize the false negatives. So a better choice of t is required for avoiding false predictions. It is noteworthy that there exists a particular threshold value for which the framework exhibits perfect behavior. Since the performance of the system is primarily determined by the choice of t, an effort was made to find the best t by varying it from 0 to 5 and found that the method performs best when $t = 4$.

In addition, we have also evaluated accuracy, precision, recall and F1 (harmonic mean of precision and sensitivity to measure the performance of the proposed method. Table 3 summarizes the evaluation results using the following measurements in various thresholds.

$$\mathrm{Accuracy} = \frac{(\mathrm{TP} + \mathrm{TN})}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{8}$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{9}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{10}$$

$$\mathrm{F1} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{11}$$

Figure 4 shows the percentage of false detections from various tabnabbing pages. From our analysis, it has been noted that impersonated versions of email services (Hotmail and Gmail) hasn't contributed to false detections. The percentage of false detections was mainly from fake versions of banking sites (Bradesco and Citibank).This shows that our method could detect all the cases of tabnabbing launched using email services.

6.1.2. URL and hyperlink analysis

In this experiment, we evaluate the robustness of our agent based method against URL obfuscations

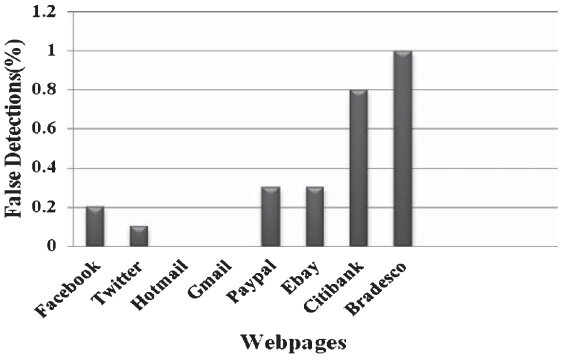

Fig. 4. False detections from tabnabbing pages.

and malicious links. The experiment is conducted in 1000 legitimate pages and 1000 phishing pages. The legitimate pages are some common webpages and phishing sites are taken from PhishTank. PhishTank is the largest collaborative clearing house for data and information about phishing scams on the Internet [30]. After submitting to PhishTank, a potential phishing URL is verified by a number of registered users to confirm it as phishing. We collected 500 confirmed phishing URLs from July 1 to September 30 of 2015 (Phishing dataset 1) and another 500 phishing URLs from October 1 to December 31 of 2015 (Phishing dataset 2). A program in java is written to determine the legitimacy of the URL. The occurrence pattern of each URL feature in phishing datasets are monitored and are plotted. Figure 5 shows the percentage of existence of lexical features in two phishing datasets. We have selected ten suspicious tokens frequently appearing in phishing URLs. Figure 6 shows the number of occurrences of suspicious tokens in the selected phishing URL set. It is noticed that some of these tokens have correlation

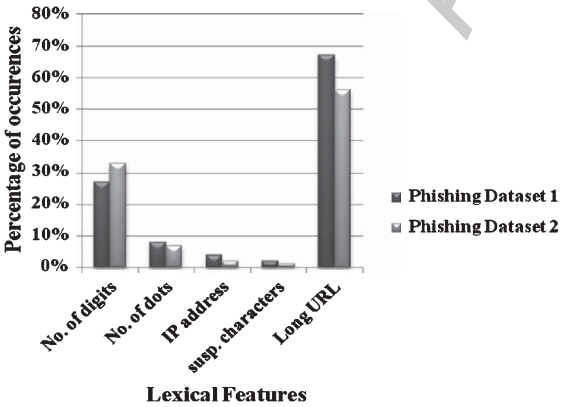

Fig. 5. Percentage of occurrence of lexical features.

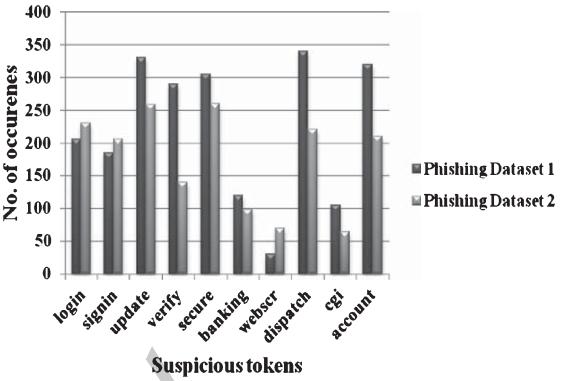

Fig. 6. Number of occurrences of suspicious tokens.

with each other and they appear together in a phishing URL.

These patterns are noted and are given in Table 4. There are ten popular targets identified from PhishTank which are frequently used by phishers to launch attacks. Figure 7 shows the percentage of phishing URLs selected from these targets for the two phishing dataset. Figure 8 shows the average miss rate with respect to the number of legitimate and phishing pages. The results show that the percentage of miss rate is reasonably low. It has been seen that the suitable selection of phishing features from malicious URLs have significant impact on the method's performance. The URL detection method has succeeded in appropriate detection of features from phishing and legitimate sites and thus contributed to lower miss rate. The approach achieved an even better result using fewer features.

After the URL structural analysis, the extracted hyperlinks from the webpages also undergo the same URL analysis procedure if they are not blacklisted. During the process of hyperlink analysis, we have gone through webpages with no hyperlinks, single hyperlinks and multiple hyperlinks.

Table 5 shows that the accuracy of the method increases when incorporating URL and hyperlink analysis.

6.2. *Comparative analysis*

In this section, a performance comparison of the proposed method against the existing methods is plotted. The proposed method is an agent based method which brings the power of agents in identifying phishing attacks. The first comparison is done with two existing anti-tabnabbing methods to prove the efficiency of our approach in detecting tabnabbing.

Table 4
Number of co-occurrences in suspicious keywords

Tokens	login	signin	update	verify	secure	banking	webscr	dispatch	cgi	account
login	–	√	√		√	√	√		√	√
signin	√	–	√		√	√				√
update	√	√	–		√	√	√	√	√	√
verify				–				√		
secure	√	√	√		–	√	√		√	√
banking	√	√	√		√	–				√
webscr	√		√		√		–		√	√
dispatch			√	√				–		
cgi	√		√		√		√		–	√
account	√	√	√		√	√	√		√	–

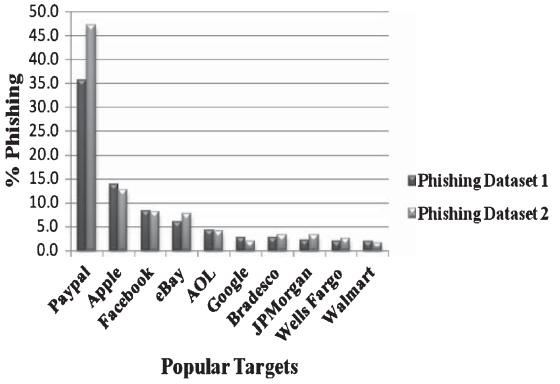

Fig. 7. Percentage of phishing URLs from popular targets.

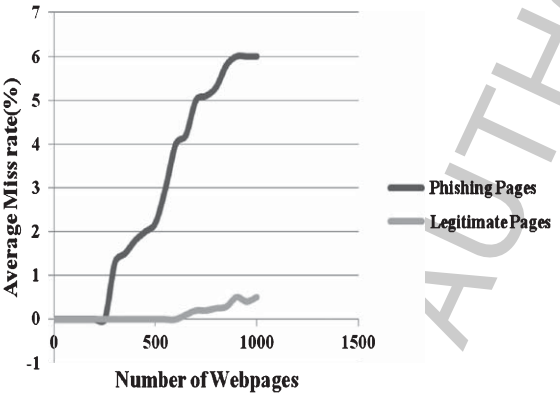

Fig. 8. Average miss rate with respect to the number of webpages.

The metric used is accuracy (of attack recognition). Figure 9 shows the comparison analysis of the proposed and existing two anti-tabnabbing methods TabShots [12] claiming an accuracy of 78% and TabsGuard [15] offers an accuracy of 96.5%. A quick glance at the results show that the proposed anti-phishing solution is able to detect phishing attacks with an accuracy of 97.3% and outperforms the existing methods by producing better results.

Table 5
Performance analysis of the method

Method	Accuracy
Agent Based Method + Blacklisting	91%
Agent Based Method + Blacklisting + URL Analysis	94.5%
Agent Based Method + Blacklisting + URL Analysis + Hyperlink Analysis	97.3%

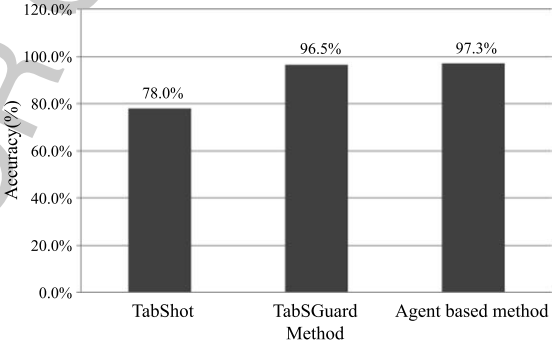

Fig. 9. Comparative analysis with anti-tabnabbing methods.

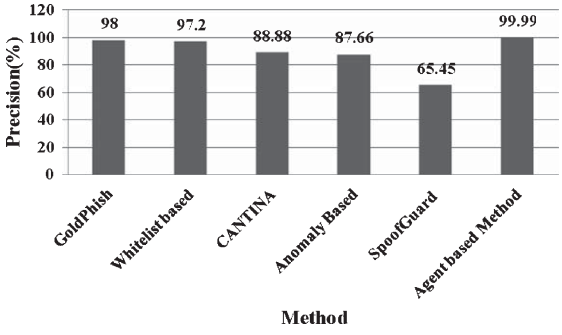

Fig. 10. Comparative analysis with anti-phishing methods.

The second comparison is done to show the overall efficiency of our method in phishing attack detection. Figure 10 shows the comparison analysis of

the proposed method and five existing antiphishing methods [1, 24, 28, 36, 43] in terms of precision.

7. Discussion

Eventhough agents are used in a variety of platforms, ours is the first attempt to utilize them in antiphishing process. By considering agent as a service [7], a lot of human effort in phishing monitoring and detection can be saved. The system consists of agents that cooperatively self-organize [26] to monitor and track fraudulent websites. The approach uses textual features of a webpage to recognize an attack and is able to capture visually similar or dissimilar phishing targets.

In contrast to the existing schemes [12, 23, 31, 34], our scheme is designed to neutralize three different types of phishing. Remarkably, our method has the virtue that the adversary has very little chance to evade detection, in comparison to other anti-tabnabbing schemes [12, 15, 31, 34]. In the framework, there is no chance of any false positives as we consider resemblance score as the basis for site's legitimacy. False negatives occur when a phisher tries to launch tabnabbing with a look-a-like webpage with very few changes in page layout. This could be alleviated by fine-tuning the threshold value. In this framework, user security was given utmost importance as attacks exploiting human have been on the rise.

This method alerts the user about the attack and gives explicit warning messages about the symptoms of attack which are simple to understand. The proposed framework is a simple and effective method which concentrates on data security and accuracy of attack recognition.

8. Conclusions and future work

This paper presents the design and evaluation of an agent based antiphishing method for identifying phishing websites. The approach is aimed to detect new types of phishing scams leading to identity theft and financial losses. This distributed agent based framework using JADE platform can monitor and detect phishing sites which masquerade as benevolent ones simultaneously in many tabs. In practice, the approach performs very well in perceiving tabnabbing attack, phishing URLs and malicious links in webpage. In future, the proposed method can be refined to work suitable for evading other phishing attacks and is thus robust over time.

References

[1] A. Belabed, E. Aimeur and A. Chikh, A Personalized Whitelist Approach for Phishing Webpage Detection, In *Proceedings of the 2012 Seventh International Conference on Availability, Reliability and Security, ARES '12, IEEE Computer Society*, 2012, pp. 249–254.

[2] A. Martin, et al., A framework for predicting phishing websites using neural networks, *International Journal of Computer Science Issues* **8**(2) (2011), 330–336.

[3] A. Raskin, Tabnabbing: A new type of phishing attack, 2010, http://www.azarask.in/blog/post/a-new-type-of-phishing-attack/.

[4] A.Y. Fu, L. Wenyin and X. Deng, Detecting phishing web pages with visual similarity assessment based on earth mover's distance, *IEEE TDSC* **3**(4) (2006), 301–311.

[5] A.Z. Broder, S.C. Glassman, M.S. Manasse and G. Zweig, Syntactic Clustering of the Web, 1997.

[6] R. Bellifemine and G. Poggi, JADE - A FIPA-compliant Agent Framework. *Proceedings of the 4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, London, 1999.

[7] Bradshaw. Software Agents. MIT Press. Cambridge: MA USA, 2002.

[8] A. Castiglione, et al., Secr3t: Secure end-to-end communication over 3g telecommunication networks, Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Fifth International Conference on IEEE 2011, pp.520-526.

[9] A. Castiglione, et al., Speech: Secure personal end-to-end communication with handheld. *ISSE 2006—Securing Electronic Business Processes Vieweg*, 2006, pp. 287–297.

[10] K.-T. Chen, et al., Fighting phishing with discriminative keypoint features, *IEEE Internet Computing* (2009), 56–63.

[11] M. Cova, C. Kruegel and G. Vigna, Detection and analysis of drive-by-download attacks and malicious JavaScript code, *Proceedings of the 19th International Conference on World Wide Web*, ACM, 2010, pp. 281–290.

[12] P. De Ryck, N. Nick, L. Desmet and W. Joosen, TabShots: Client-Side Detection of Tabnabbing Attacks. *Proceedings of the 8th ACM SIGSAC Symposium on Information*, 2013, pp. 447–456.

[13] A. De Santis, et al., An intelligent security architecture for distributed firewalling environments, *Journal of Ambient Intelligence and Humanized Computing* **4**(2) (2013), 223–234.

[14] E. Medvet, E. Kirda and C. Kruegel, Visual similarity-based phishing detection, Computational intelligence in cyber security, CICS 2009. IEEE Symposium, pp. 30–36.

[15] F. Hashemi, M. Zulkernine and K. Weldemariam, TabsGuard: A Hybrid Approach to Detect and Prevent Tabnabbing Attacks. Risks and Security of Internet and Systems. Springer International Publishing, 2014, pp. 196–212.

[16] He. Debia, et al., Efficient certificate less anonymous multi-receiver encryption scheme for mobile devices, *Soft Computing* (2016), 1–10.

[17] H. Huang, L. Qian and Y. Wang, A SVM-based technique to detect phishing URLs, *Information Technology Journal* **11**(7) (2012), 921–925.

[18] JADE (Java Agent DEvelopment Framework), http://jade.tilab.com/index.html
[19] T.N. Jagatic, Social phishing, *Communications of the ACM* (2007), 94–100.
[20] A.K. Jain and B.B. Gupta, A novel approach to protect against phishing attacks at client side using auto-updated white-list, *EURASIP Journal on Information Security* **1** (2016), 1–11.
[21] J. Whitehead, SAX Parsing, Spring 2006. https://classes.soe.ucsc.edu/cmps183/Spring06/lectures/sax-parsing.pdf
[22] J. Mitchell, Browser Security Model, Spring, 2010.
[23] J. Chen and C. Guo, Online Detection and Prevention of Phishing Attacks, *Proceedings of the First IEEE International Conference on Communications and Networking*, China, 2006, pp. 1–7.
[24] M. Dunlop, S. Groat and D. Shelly, GoldPhish: Using Images for Content-Based Phishing Analysis, *Fifth IEEE International Conference on Internet Monitoring and Protection*, 2010, pp. 123–128.
[25] J.C. Mogul, et al., Hypertext Transfer Protocol–HTTP/1.1, Network Working Group RFC 2068, 1997, pp. 1–161.
[26] F.A. Narudin, Evaluation of machine learning classifiers for mobile malware detection, *Soft Computing* **20**(1) (2016), 343–357.
[27] No Script - JavaScript/Java/Flash blocker for a safer Firefox experience. http://noscript.net/
[28] Y. Pan and X. Ding, Anomaly based web phishing page detection, *IEEE Computer Security Applications Conference*, 2006, Vol 6, pp. 381–392.
[29] P. Prakash, M. Kumar, R.R. Kompella and M. Gupta, PhishNet: Predictive Blacklisting to Detect Phishing Attacks. In *Proceedings of the 29th Conference on Information Communications, INFOCOM'10, IEEE Press*, Piscataway, NJ, USA, 2010, pp. 346–350.
[30] PhishTank. PhishTank - Out of the Net, into the Tank. http://www.phishtank.com/, April 2016.
[31] R.K. Suri, D.S. Tomar and D.R. Sahu, An approach to perceive tabnabbing attack, *International Journal of Scientific and Technology Research* **1**(6) (2012), 90–94.
[32] R.B. Basnet, A.H. Sung and Q. Liu, Learning to detect phishing URLs, *International Journal of Research in Engineering and Technology* **3**(6) (2014), 11–24.
[33] A.S. Rao and M.P. Georgeff, BDI agents: From theory to practice, *ICMAS* **95** (1995).
[34] S.A. Unlu and K. Bicakci, NoTabNab: Protection Against the Tabnabbing Attack, In eCrime Researchers Summit (eCrime), 2010, pp. 1–15.
[35] Sycara. Multiagent Systems, American Association for Artificial Intelligence, 1998.
[36] Y. Teraguchi, N. Chou, R. Ledesma, D. Boneh and J.C. Mitchell, Client-side defense against web-based identity theft, *11th Annual Network and Distributed System Security Symposium (NDSS'04)*, San Diego, 2004.
[37] The Web Application Security Consortium. Cross-site Scripting (XSS). http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting29
[38] M. Tsukada, T. Washio, et al., Automatic Web Page Classification using Machine Learning Methods, LNCS, *Proceedings of First Asia-Pacific Conference on Web Intelligence: Research and Development*, 2198 (2011), pp. 303–313.
[39] V.P. Reddy, V. Radha and M. Jindal, Client side protection from phishing attack, *International Journal of Advanced Engineering Sciences and Technologies* **3**(1) (2011), 39–45.
[40] L. Wenyin, et al. Detection of phishing webpages based on visual similarity. *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web ACM*, 2005, pp. 1060–1061.
[41] W. Wu, et al., Towards secure and cost-effective fuzzy access control in mobile cloud computing, *Soft Computing* (2015), 1–7.
[42] Y. Cao, W. Han and Y. Le, Anti-phishing Based on Automated Individual Whitelist, DIM'08, October 31, 2008, Fairfax, Virginia, USA.
[43] Y. Zhang, J.I. Hong and L.F. Cranor, CANTINA: A Content-based Approach to Detecting Phishing Web Sites, *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*, Banff, Alberta, CA, 2007.
[44] YesScript Firefox Add on, https://addons.mozilla.org/enUS/firefox/addon/4922
[45] Y.Z. Zhang, The Automatic Classification of Web Pages based on Neural Networks, *Neural Information Processing ICONIP 2001 Proceedings*, China, 2 (2001), pp. 570–575.