Intelligent Agents in Securing Internet

S. Sarika, Paul Varghese[1]

[1] Department of Computer Science, Bharathiar University, India
sarika.anand08@gmail.com, vp.itcusat@gmail.com

Abstract

Internet phishing has become one of the most prevalent problems of online security. Phishing takes advantage of the user's trust and use social engineering techniques to deceive them. As browsers have become a major platform for attacks to take place, a greater focus should be given for securing web services in this perspective. This paper discusses a framework to resist phishing attacks utilizing the power of intelligent agents. The main focus is on a browser based attack called tabnabbing which executes in inactive browser tabs. The proposed method uses agents in three levels to monitor browser tabs, at regular intervals and warn the user at the earliest. This approach excellently protects user against Tabnabbing, 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: Distributed software agents, Multi agent systems, Tabnabbing, Antiphishing

1 Introduction

Phishing [1] is an art of deception wherein secure websites are so perfectly impersonated that even cautious users are tricked. Through the past decades, the number of victims has increased exponentially as phishers improvise tactics by exploiting loopholes in software.

Recently, the web has become client-centric, and web browsers, tools of deliverance of information. Browsers display the web pages using an underlying web protocol called Hyper Text Transfer Protocol [2]. Eventhough http allows for the quick and easy transmission of information, it is not secure enough as the conversation between servers and clients can be eavesdropped in to. 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 [3] to fake https protocol.

Modern browsers are significantly more secure than before. Even then, the browsers are abused after monitoring the user's browsing habits. Some of the known web browser security risks include social engineering [4], clickjacking, session hijacking [5] and cross domain vulnerabilities [6-7] like cross site scripting, cross site request forgery etc. These techniques run the gamut from simple eavesdropping, through theft of identity and personal information, to financial losses. As for the fixed world, there are several counter measures based on various methods such as the ones presented in [8-10].

The power of artificial intelligence can be utilized for a large number of applications and environments. 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. The name 'tabnabbing attack' [11] was coined in early 2010 by Aza Raskin, creative lead of Mozilla Firefox. The attack is simple to implement and silently tracks the victims. A user has a bunch of open tabs and as he navigates through them, phishers set up a counterfeited web site which looks exactly like the legitimate one and load the inactive tabs with the fake page. When the user switches back to the tab, it appears to be a site frequently used by the user and prompts the user to enter his credentials convincing the user that the site is authentic. 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 happens on the behind and is likely to betray even the most incredulous persons and is able to change favicon, title, and layout of a webpage with some other site familiar to the user. Tabnabbing cannot be avoided by using https instead of http in web address. The attack can be launched via scripting support [12], and with the use of HTML refresh meta tag [13] in predetermined time intervals.

The rest of this paper is structured as follows. Next section describes related work. Section 3 explains our proposed method. Section 4 discusses about implementation followed by experimental results. Discussion part is included in Section 5 and Section 6 concludes the paper.

[*] Corresponding Author: S. Sarika; E-mail: sarika.anand08@gmail.com

DOI: 10.3966/160792642018051903012

2 Related Works

There has been a greater focus on the subject of securing web applications in the perspective of increasing interest in online security. Research works have already been conducted in this area describing various antiphishing approaches. The basic phishing detection method is list-based approach (blacklist and whitelist) [14-15]. In blacklist approach, the system keeps a pre-compiled list of URLs which were found to be malicious at some point in time. The whitelist based solution keeps a list of legitimate URLs that prevents access to phishing sites by URL similarity check. The list based methods suffer from timely updation of list as it may become obsolete. In order to solve the problem of list updation, different heuristic methods [15-16] 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 [17-19]. 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. Chen has explained LinkGuard [20], 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. Zhang et al. proposed a content based solution, CANTINA [21] 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 [22] 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 vulnerable to attacks on Google's PageRank algorithm and Google's search service. Another method SpoofGuard [3], 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. The aforementioned approaches focus on old types of phishing attacks. Tabnabbing is a modern browser security threat and its current management modalities are briefly explained below.

NoScript [23] and YesScript [24] are Firefox add-ons, preventing websites from running JavaScript, Java, Flash or other plugins. But they do not provide protection in other browsers. NoTabNab [25] 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 [26] 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 [27] 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 difficulty in detecting small changes in a page.

TabsGuard [28] combines heuristic based metrics and data mining techniques to detect tabnabbing. The approach uses five heuristic-based metrics to measure the degree of changes made to the tree representation of each webpage whenever a tab loses focus.

Existing anti-tabnabbing methods detect the layout change and warn the user only when the tab is on focus after being nabbed and also they focus mainly on the change in page layout, title and favicon but not much attention is given to change in URL. Our approach is similar to the method proposed by Fahim et al. [28] where structural features of a webpage are analyzed but the selected features are different. The proposed method uses software agents [29] to concurrently monitor the change in webpage layout at regular intervals in all the tabs of a browser and alerts the user during 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 multiagent 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 from tabnabbing attack, URL obfuscations and malicious links by giving explicit warnings.

3 Multi Agent Based Phishing Detection

Multi Agent based phishing detection is a scientific approach which needs planning, designing and deploying multiagents in a platform to protect webpages from a variety of phishing attacks. The proposed method is a modular, multi-agent architecture [30], where the webpage activities are monitored and controlled by deliberative BDI (Belief, Desire, Intention) agents [31]. They are able to cooperate and act on purely dynamic environments like internet and are able to face complex and real problems like phishing, even when they have few resources available. The architecture proposes a new and easier method of building distributed multi-agent systems. The proposed system uses the following classes of agents:

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

Figure 1 shows the architecture of the proposed system in two browser tabs T1 and T2. Whenever a new tab is opened, a fresh pair of level 1 agents (U-agent and T-agent) are created to observe the activities in that tab to discover a malicious action. When the tab is closed, they are disposed.

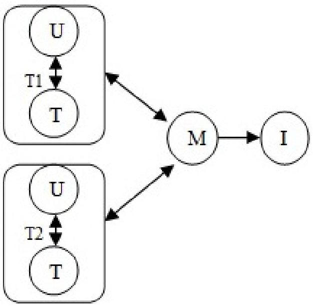


Figure 1. Architecture of multi agent system

The level one agents are managed by level two agent (M-agent). The interface between the user and the system is provided by level three agent (I-agent). The agents are arranged in a hierarchical fashion so that they are able to communicate and cooperate properly. They are having a set of characteristics, such as parallelism, autonomy, reactivity, adaptivity etc. which allow them to cover several needs for dynamic environments. Multi-agent systems are incorporated in this system to bring modularity and parallelism where all the computing tasks are delegated to the agents. There is an agent platform which integrates a set of agents with specific functionalities. As the distributed multi-agent platform is designed by means of modeling the functionalities of the agents, all communications take place via the agents and thus the system control is reduced. The communication among agents in the platform uses FIPA ACL [32].

3.1 T-agent

T-agent or Tabnab agent is a level 1 agent responsible for handling incoming requests from browsers for webpages. T-agents in each tab perform its delegated task in two phases, Feature Extraction and Feature Comparison to detect tabnabbing in a webpage. Figure 2 depicts the functionality of T-agent.

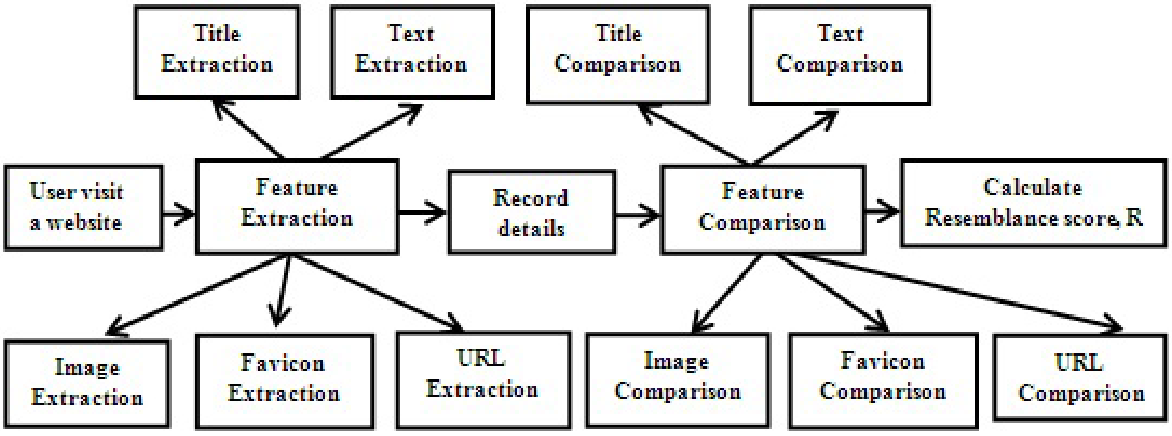


Figure 2. Functionality of T-agent

When a webpage is loaded, T-agent extracts its five tuple information which is recorded for the next phase. The procedure is continued every 60seconds and these features are matched with the recorded values to obtain a resemblance score for each pair of elements. A web page is considered as similar to the recorded one, if resemblance score is higher than a threshold *t*. The details are discussed in the next two subsections.

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 [33] 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 handles larger documents. SAX processing 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. Feature extraction process is explained formally in Algorithm 1.

```
Algorithm 1: extract Features
Input: Webage $w$
Output: Extracted features of $w$
1. Open the webpage $w$
2. Parse count=0
3. $U$ <-webpage URL
4. $T$ <-Parse w using SAX
5. $H$ <-hyperlinks Filter($T$)
6. $T_i$ <-getTitle($T$)
7. extract Favicon($w$)
8. extract Images($w$)
9. Parsecount++
```

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 [34]. 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:

$$Rt(d_1, d_2) = \frac{s(d_1) \cup s(d_2)}{s(d_1) \cap s(d_2)}. \quad \textbf{(1)}$$

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 are 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

$$1 - L_1(H, H^1). \quad \textbf{(2)}$$

- Comparison of pixel positions occupied by the image.
  The resemblance between the two positions in a webpage is computed as

$$1 - (d / M_d). \quad \textbf{(3)}$$

  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* andA^1 are calculated as:

$$\left[\frac{|A - A^1|}{Max(A, A^1)} \right] \quad \textbf{(4)}$$

Using these four scores, a single resemblance score *Ri* ∊ [0, 1] is derived. The webpage addresses are monitored and recorded at 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 using the above mentioned 5-tuple as

$$R=Rt+Ri+Ru+Rf+Rti. \quad \textbf{(5)}$$

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. The hyperlinks in the webpage are extracted and effectively examined to verify if they are mischievous.

```
Algorithm 2: compare Features
Input: Webage $w$
Output: Resemblance score
1. Add new Ticker Behaviour to T-agent for every 60
   seconds
2. extract Features($w$)
3. if (Parsecount>1) then
4.     $R_t$ <-compare Text()
5.     $R_i$ <-compare Images()//from stored directory
6.     $R_f$ <-compare Favicon()
7.     $R_u$ <-compare URL()
8.     $R_{ti}$ <-compare Title()
9. $R = Rt + Ri + Rf + Ru + Rti$
10.    returen $R$
11. else
12. returen false
13. end if
```

3.2 U-agent

U-agents(URL check agent) are in action if the URL of the webpage is changed after a tab switch event or inert tab. The approach maintains a URL blacklist which holds URLs that refer to sites that are considered malicious. 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 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. 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.

Algorithm 3: behaviour of URL

Input: Webage *w*, Blacklist *BL*

Output: URL Check result 0: Legitimate
1: Phishing

```
if URL is changed then
    if changed URL in BL then
    returen 1
    else
    extract URL features(U)
    URL Check ()
      if URL Check score>=3 then
         returen 1
         else
         return 0
      end if
   end if
 else
return 0
end if
```

URL features. Phishing URLs can be of various types. The obfuscating URL patterns include digits in the URL, long URLs, many dots in the URL, etc. The proposed method has used 25 features selected by observing the heuristics in the structure of phishing URLs and also by referring literatures [16, 35]. As shown in Table 1, there are 5 lexical features, 10 token based features and 10 target based features.

Table 1. Feature types and its count

Feature Type	Count
Lexical	5
Token based	10
Target based	10

(1) *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 noticing the composition of phishing URLs in phishtank.com. The lexical features include digit in host, length of hostname, number of suspicious characters '@', number of dots in path and length of URL.

(2) *Token Based Features:* The malicious URLs may contain some eye catching keywords or tokens to attract end users (eg. signin, confirm, login etc). The selected 10 keywords includes login, signin, confirm, verify, secure, banking, web, dispatch, pay and http.

(3) *Target Name Features:* From phishtank data archive, an analysis was done for different monthly stats archive and collected top10 brands used by fraudsters during the period 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, Yahoo, Itau, WalMart and Bradesco.

4 Implementation

The implementation of the proposed method uses JADE software framework [36] in java platform. In order to perform the experiments, we used Core i3 @2.20 GHz processor, 4GB of RAM memory, JDK 1.8 and JADE 4.2.0 in Windows 7 platform. In our method, we have used a layered architecture and have three main software layers. The lowest layer or reactive layer is the j2se/j2ee runtime environment; on a local computing machine. The middle layer is the JADE platform, which comprises a number of containers for providing services for multi-agent operations. The uppermost layer is the application layer where the agents communicate and co-ordinate with each other to perform their delegated tasks. 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.

4.1 Experimental Evaluation

We conducted two experiments 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 are conducted to analyse 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.

Experiment 1-Detection of tabnabbing attack. 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, social networking sites etc. 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 are taken.

Table 2. Sources of dataset

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

Tabnabbing attack is 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 is simulated in these webpages by running a script. For the simulation, start the agent platform and load the webpages in the dataset in different tabs of the browser. Run the script for the currently focused window and perform a tab switch to some other window. When the user returns to that inactive tab after 60 seconds, the webpage has changed to a tabnabbing page (already created). A time interval of 60 seconds is set with 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, we partitioned the resemblance score set 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. Figure 3 shows the snapshot of a webpage opened in one of the browser tabs and the webpage changed its layout to gmail login page. The alert generated by the agent based mechanism is shown in Figure 4.

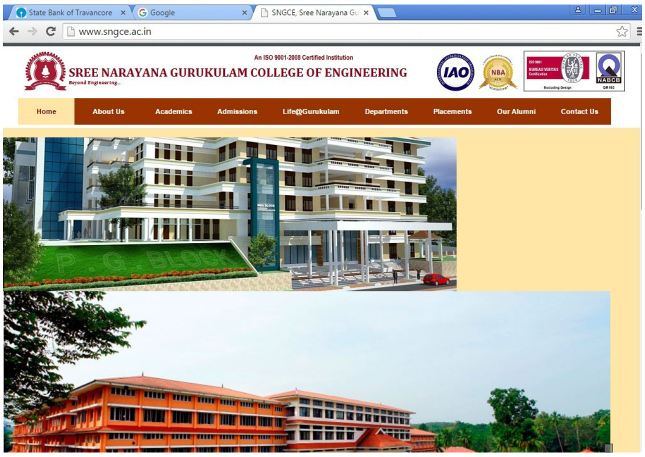
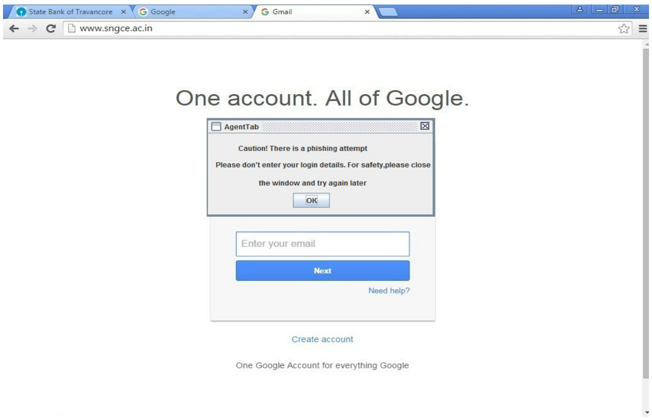


Figure 3. View of a webpage before tab switch



One account. All of Google.

Caution! There is a phishing attempt

Please don't enter your login details. For safety,please close the window and try again later

One Google Account for everything Google

Figure 4. Impersonated webpage and the alert generated by the framework

The effectiveness of the method is assessed using the following parameters.

True positive (*TP*) – Legitimate websites detected as legitimate.

True negative (*TN*) – Phishing websites detected as phishing sites.

False positive (*FP*) – Legitimate websites detected as phishing sites.

False negative (*FN*) – Phishing websites detected as legitimate ones

False Positive Rate (*FPR*) and False Negative Rate (*FNR*) for various values of the threshold *t* are computed which is shown in Figure 5. They are calculated using the formulas given below:

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

$$FNR = \frac{FN}{(FN + TN)}. \tag{7}$$

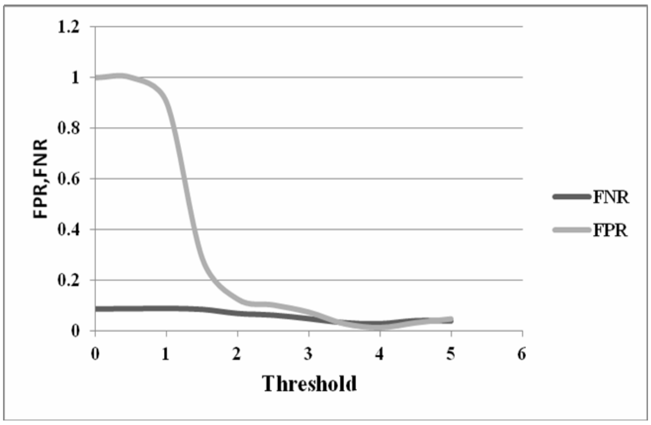


Figure 5. False positive rate and false negative rate in various thresholds

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 is done 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 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.

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

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

$$Precision = \frac{TP}{TP + FP}. \qquad \textbf{(9)}$$

$$Recall = \frac{TP}{TP + FP}. \qquad \textbf{(10)}$$

$$F1 = \frac{2TP}{2TP + FP + FN}. \qquad \textbf{(11)}$$

Figure 6 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.

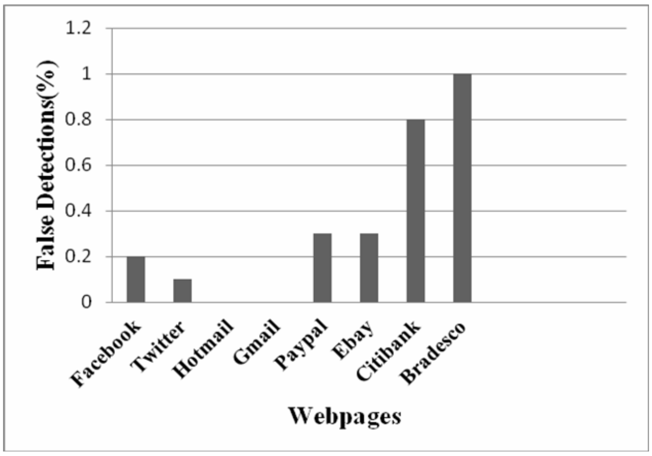


Figure 6. False detections from tabnabbing pages

Experiment 2-Evaluation of page URL. In this experiment, we evaluate the robustness of our agent based method against URL obfuscations and malicious links. The experiment is conducted in 500 legitimate pages and 400 phishing pages. The legitimate pages are some common webpages and phishing sites are taken from PhishTank [37]. PhishTank is the largest repository for data and information about phishing scams on the Internet. After submitting to PhishTank, a potential phishing URL is verified by a number of registered users to confirm it as phishing. A program in java is written to determine the legitimacy of the URL. Table 4 shows the percentage of existence of each feature in legitimate and phishing URLs.

Table 4. Percentage of occurrence of each feature in legitimate and phishing URLs.

Features	Legitimate URLs(%)	Phish URLs(%)
Lexical	0.3%	38.3%
Token based	1.60%	58%
Target based	0.80%	33.20%

Figure 7 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.

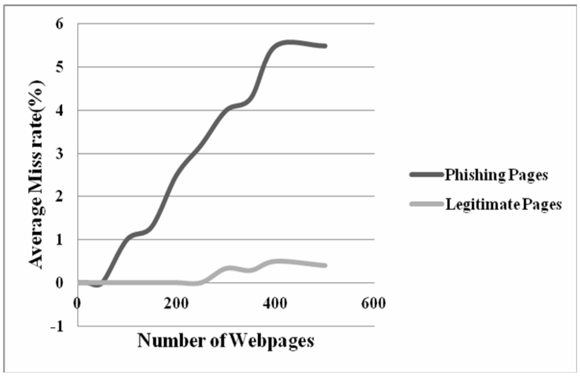


Figure 7. Average miss rate with respect to the number of webpages

4.2 Agent Performance

In this section, we evaluate the performance of individual agents and the multi agent system in detecting phishing attacks.

Performance evaluation of individual agents. In the proposed method, multiple agents are designed to operate in a complex environment to detect sophisticated phishing attacks. Moreover, the agents are equipped with computational behavior to perform tasks. Here, T-agent and U-agent are purely computational. There are four agents in action while a browser tab is open. Figure 8 shows graph with the number of agents active plotted with the number of opened browser tabs.

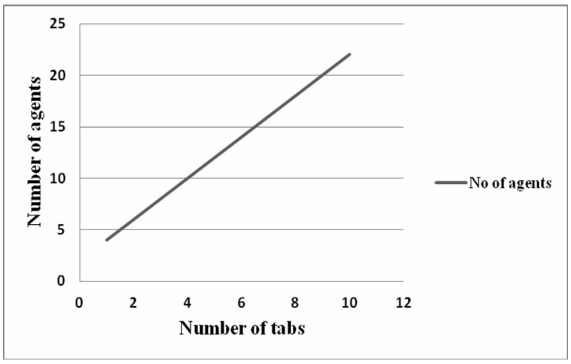


Figure 8. Number of agents vs Number of tabs

The complexity of each agent in performing a task is computed by measuring the number of methods implemented within the agent and the sum of cyclomatic complexities of these methods. A higher value indicates a complex agent. This is calculated as follows:

$$\text{Task Complexity, } (TC) = \sum_{i=0}^{n} C_{i.} \tag{12}$$

where, n is the number of methods implemented within the task; Ci is the i^{th} method complexity. The complexity for one module is Log n, thus the overall complexity for one task is n* Log n. The complexity level of each agent is shown in Figure 9.

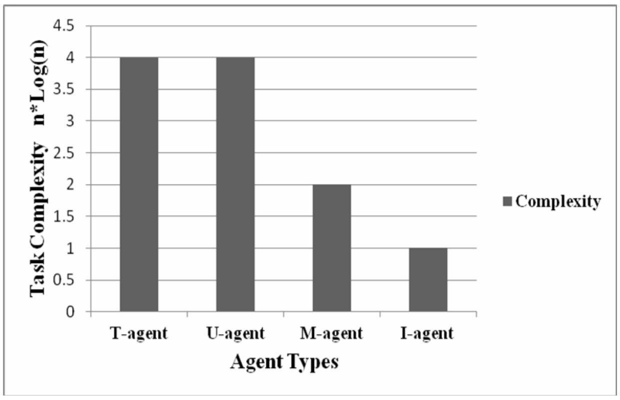


Figure 9. Task complexity of agents

The computation complexity involved in performing a task can be effectively reduced by using the following optimizations. The prime idea to improve performance is to execute the expensive comparison operations towards the end. In this method, more computational complexity is involved in image comparison. While evaluating the resemblance score of favicon and URL, if there is a mismatch with the stored value, we can skip text and image comparison as the overall resemblance score does not reach the threshold value *t.* This can result in reduced complexity and computation cost.

The agent technology described here shows promising results in building cost-effective, distributed systems that are powerful and flexible. The multiple agents in this system communicate and coordinate in a peer to peer fashion. They do need to send and receive messages using agent communication protocol. The agents communicate via asynchronous message passing and they use FIPA Agent Communication Language [32] in JADE platform.

FIPA guarantees the interoperability between agents by coordinating different aspects of systems such as agent communication, agent management, and agent message transportation.

The architecture of the proposed MAS is hierarchical and distributed, which significantly reduces communication cost and increases efficiency. The number of messages used for inter agent communication increases linearly with the number of agents but the number of messages sent between agents are minimized to reduce the communication overhead. Frequent communications are between level 1agents. We are planning to improve and fine-tune our current model to address the problem of communication delay between agents in the next step.

Performance evaluation of MAS. To evaluate the performance of our system in parallel attack recognition, we opened multiple tabs in parallel and ran the attack in each window. It has been noted that the method was able to detect attacks while running 10 browser tabs in parallel with good response time (in milliseconds). There is a slight decrease in efficiency beyond that as it may cause delay in the system.

Figure 10 shows the efficiency of the system with the number of browser tabs. Efficiency is expressed in percentage and is calculated by dividing the number of tabs in which attack detected by the total no of tabs opened.

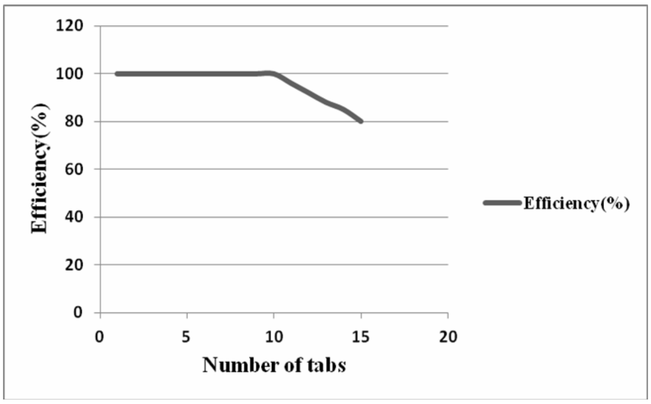


Figure 10. Efficiency of the method in parallel attack recognition

4.3 Comparative Analysis

In this section, a performance comparison of the proposed method against existing methods is plotted. The first comparison is done with two existing anti-tabnabbing methods to prove the efficiency of our approach in detecting tabnabbing. The metric used is accuracy (of attack recognition). Figure 11 shows the comparison analysis of the proposed and existing anti-tabnabbing methods TabShots [27] claim an accuracy of 78% and TabsGuard [28] offers an accuracy of 96.5%. A quick glance at the results show that the proposed anti-phishing solution is able to detect phishing with an accuracy of 97.3% and outperforms the existing methods with less false positives and false negatives. The second comparison is done to show the overall efficiency of our method in attack detection. Figure 12 shows the comparison analysis of the proposed method and five existing antiphishing methods GoldPhish [22], Whitelist based method [15], CANTINA [21], Anomaly based method [16] and SpoofGuard [3] in terms of precision. From the results it has been found that the proposed agent based method performs well, and achieved good results, than the other existing approaches.

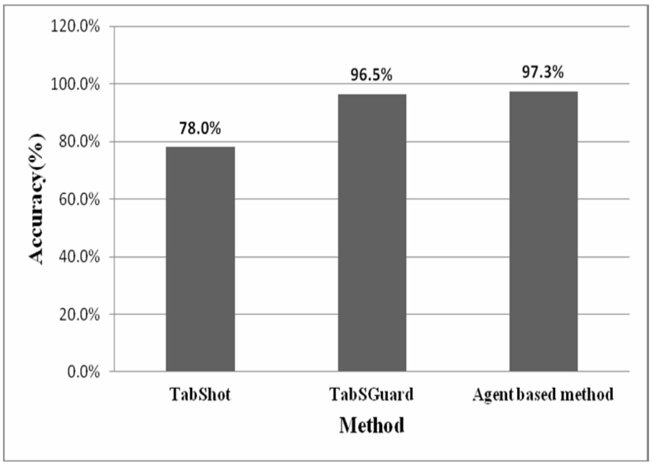


Figure 11. Comparative Analysis with Anti-Tabnabbing methods

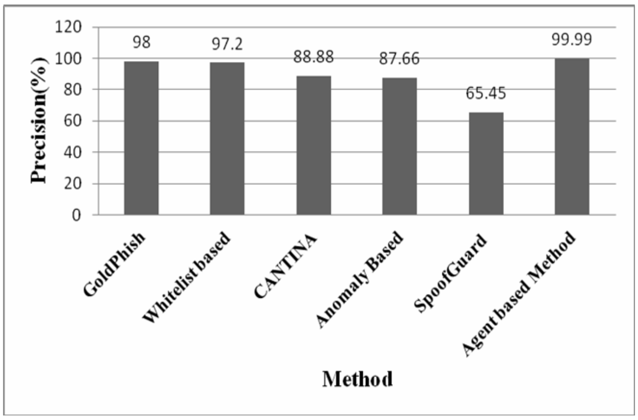


Figure 12. Comparative Analysis with Anti-phishing methods

5 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 [29], a lot of human effort in phishing monitoring and detection can be saved. The system consists of agents that cooperatively self-organize [30] to monitor and track fraudulent websites. The approach uses textual features of a webpage to recognize a phishing attack and is able to capture visually similar or dissimilar phishing targets as it is considering the resemblance score of the webpage features for classifying the current page as fake or authentic.

In contrast to existing schemes [3, 17, 21-22] our scheme is designed to neutralize three different types of phishing attacks. Remarkably, our method has the virtue that the adversary has very little possibility to evade detection, in comparison to other anti-tabnabbing schemes [25-28]. In the framework, there is less chance of 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 vulnerabilities have been on the rise and online security has become that much important [38].

This method has tried to do something different where it alerts the user about the attack and give 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.

6 Conclusion

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

References

[1] T. N. Jagatic, N. A. Johnson, M. Jakobsson, F. Menczer, Social Phishing, *Communications of the ACM*, Vol. 50, No. 10, pp. 94-100, October, 2007.

[2] J. Mogul, DEC, H. Frystyk, T. Bermers-Lee, Hypertext Transfer Protocol-HTTP/1.1, RFC 2068, January, 1997.

[3] N. Chou, R. Ledesma, Y. Teraguchi, J. C. Mitchell, Client-side Defense against Web-based Identity Theft, *11th Annual Network and Distributed System Security Symposium (NDSS'04)*, San Diego, California, 2004, pp. 1-16.

[4] C. Hadnagy, P. Wilson, *Social Engineering: The Art of Human Hacking*, John Wiley & Sons, 2010.

[5] M. Johns, Session Hijacking Attacks, in: H. C. A. van Tilborg, S. Jajodia (Eds.), *Encyclopedia of Cryptography and Security*, Springer, Boston, MA, pp. 1189-1190, 2011.

[6] J. Mitchell, *Browser Security Model*, CS155, Spring, 2010.

[7] The Web Application Security Consortium, *Cross-site Scripting (XSS)*, http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting29.

[8] M. A Ambusaidi, Z. Tan, X. He, P. Nanda, L. F. Lu, A. Jamdagni, Intrusion Detection Method based on Nonlinear Correlation Measure, *International Journal of Internet Protocol Technology*, Vol 8, No. 2-3, pp. 77-86, December, 2014.

[9] Z.-G. Chen, H.-S. Kang, S.-R. Kim, Design of a New Efficient Hybrid System for Intrusion Detection Based on HSM Fuzzy Decision Tree, *Journal of Internet Technology*, Vol 16, No. 5, pp. 885-891, September, 2015.

[10] Z. Baig, K. Salah, Distributed Hierarchical Pattern-Matching for Network Intrusion Detection, *Journal of Internet Technology*, Vol. 17, No. 2, pp. 167-178, March, 2016.

[11] A. Raskin, *Tabnabbing: A New type of Phishing Attack*, http://www.azarask.in/blog/post/a-new-type-of-phishing-attack/

[12] M. Cova, C. Kruegel, G. Vigna, Detection and Analysis of Drive-by-download Attacks and Malicious JavaScript Code, *Proceedings of the 19th international conference on World Wide Web*, Raleigh, North Carolina, 2010, pp. 281-290.

[13] Tutorialspoint, *HTML Meta Tags*, http://www.tutorialspoint.com/html/html_meta_tags.htm.

[14] P. Prakash, M. Kumar, R. R. Kompella, M. Gupta, PhishNet: Predictive Blacklisting to Detect Phishing Attacks, *Proceedings of the 29th Conference on Information Communications, INFOCOM'10*, San Diego, CA, 2010, pp. 1-5.

[15] A. Belabed, E. Aïmeur, A. Chikh, A Personalized Whitelist Approach for Phishing Webpage Detection, *Proceedings of the 2012 Seventh International Conference on Availability, Reliability and Security, ARES '12*, Prague, Czech Republic, 2012, pp. 249-254.

[16] Y. Pan, X. Ding, Anomaly based Web Phishing Page Detection, *22nd Annual Computer Security Applications Conference (ACSAC'06)*, Miami Beach, FL, 2006, pp. 381-392.

[17] A. Martin, N. B. Anutthamaa, M. Sathyavathy, M. M. S. Francois, P. Venkatesan, A Framework for Predicting Phishing Websites Using Neural Networks, *International Journal of Computer Science Issues*, Vol. 8, No. 2, pp. 330-336, March, 2011.

[18] M. Tsukada, T. Washio, H. Motoda, Automatic Web-Page Classification by Using Machine Learning Methods, *Proceedings of First Asia-Pacific Conference on Web Intelligence: Research and Development*, Maebashi, Japan, 2011, pp. 303-313.

[19] Y. Zhang, M. Zhao, Y. Wu, The Automatic Classification of Web Pages based on Neural Networks, *Proceedings of 8th International Conference on Neural Information Processing, ICONIP 2001*, Vol 2, Shanghai, China, 2001, pp. 570-575.

[20] J. Chen, C. Guo, Online Detection and Prevention of Phishing Attacks, *Proceedings of First IEEE International Conference on communications and Networking*, Beijing, China, 2006, pp. 1-7.

[21] 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, Canada, 2007, pp. 639-648.

[22] M. Dunlop, S. Groat, D. Shelly, GoldPhish: Using Images for Content-Based Phishing Analysis, *Fifth International Conference on Internet Monitoring and Protection*, Barcelona, Spain, 2010, pp. 123-128.

[23] InformAction, *NoScript - JavaScript/Java/Flash Blocker for a Safer Firefox Experience*, http://noscript.net/.

[24] J. Barnabe, A. Horvath, *YesScript, Firefox Add-ons*, https://addons.mozilla.org/enUS/firefox/addon/4922.

[25] S. A. Unlu, K. Bicakci, NoTabNab: Protection Against The "Tabnabbing Attack", *eCrime Researchers Summit*, Dallas, TX, 2010, pp. 1-5.

[26] R. K. Suri, D. S. Tomar, D. R. Sahu, An Approach to Perceive Tabnabbing Attack, *International Journal of Scientific and Technology Research*, Vol 1, No. 6, pp. 90-94, July, 2012.

[27] P. De Ryck, N. Nikiforakis, L. Desmet, W. Joosen, TabShots: Client-Side Detection of Tabnabbing Attacks, *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer*

and Communications Security, Hangzhou, China, 2013, pp. 447-456.

[28] H. F. Hashemi, M. Zulkernine, K. Weldemariam, TabsGuard: A Hybrid Approach to Detect and Prevent Tabnabbing Attacks, *9th International Conference on Risks and Security of Internet and Systems (CRiSIS 2014)*, Trento, Italy, 2014, pp. 196-212.

[29] J. M. Bradshaw, *Software Agents*, MIT Press, 2002.

[30] K. P. Sycara, Multiagent Systems, *AI Magazine*, Vol. 19, No. 2, pp. 79-92, June, 1998.

[31] A. S. Rao, M. P. Georgeff, BDI Agents: From Theory to practice, *Proceedings of the First International Conference on Multiagent Systems, ICMAS-95*, San Francisco, CA, 1995, pp. 312-319.

[32] F. Bellifemine, A. Poggi, G. Rimassa, JADE – A FIPA-compliant Agent Framework, *Proceedings of the 4th International Conference and Exhibition on the Practical Applications of Intelligent Agents and Multi-Agents*, London, UK, 1999, pp. 97-108.

[33] J. Whitehead, *SAX Parsing*, https://classes.soe.ucsc.edu/cmps183/Spring06/lectures/sax-parsing.pdf.

[34] A. Z. Broder, S. C. Glassman, M. S. Manasse, G. Zweig, Syntactic Clustering of the Web, *Computer Networks and ISDN Systems*, Vol. 29, No. 8-13, pp. 1157-1166, September, 1997.

[35] R. B. Basnet, A. H. Sung, Q. Liu, Learning To Detect Phishing Urls, *International Journal of Research in Engineering and Technology*, Vol. 3, No. 6, pp. 11-24, June, 2014.

[36] Jade, *Java Agent DEvelopment Framework*, http://jade.tilab.com/index.html.

[37] PhishTank, *Out of the Net, into the Tank*, http://www.phishtank.com/.

[38] M.-S. Kim, J.-K. Lee, J. H. Park, J.-H. Kang, Security Challenges in Recent Internet Threats and Enhanced Security Service Model for Future IT Environments, *Journal of Internet Technology*, Vol. 17, No. 5, pp. 947-955, September, 2016.

Biographies

S. Sarika is a B-Tech. degree holder in Information Technology, M.Tech. in Computer Science and Engineering and pursuing Ph.D in Computer Science from Bharathiar university.She is currently working as Assistant Professor in Computer Science and Engineering at Sree Narayana Gurukulam College of Engineering, Kolenchery, Kerala, India. Her research interest includes Internet security, Software agents and Network security.

Paul Varghese is a B-Tech. degree holder in Electrical Engineering, M.Tech in Electronics Engineering and Ph.D in Computer Science. Currently he is working as Professor in Information Technology at Cochin University of Science and Technology, Thrikkakara, Cochin, Kerala, India. His research interest includes Fault Tolerent Computing and Data security.