Browser Security: Attacks and Detection Techniques

Dr.Sarika S
Department of Computer Science
Naipunnya Institute of Management and Information Technology,Koratty,Pongam

Abstract

Nowadays, the growth of the technology reaches its height and everything goes online. At the same time Cyber attacks are increasing to breach the information system of another individual or organization. Phishing is a malicious and deliberate attempt of sending counterfeit messages that appear to come from a reputable source or mimicking a webpage which looks like an original website. The goal is to steal sensitive credentials like login information and credit card details or to install malware on the victim's machine. Browser-based cyber threats have become one of the biggest concerns in networked architectures. The most prolific form of browser attack is tabnabbing. This paper presents an overview of tabnabbing detection techniques and proves the efficiency of agent based tabnabbing detection by comparing with the state of the art techniques.

Keywords: Browser security , Social engineering , Tabnabbing attack , Phishing , software agents.

1.Introduction

The digital world is changing at a tremendous speed and social media communication tools have profoundly changed our interactions with the world around. The speedy transition to today's digital world has been boosted by network digitalization. The new communication technologies bring new opportunities, but also open up door for a number of risks. Technological advancement accelerates growth of social, cultural, managerial, political and organisational environments but also hasten the origin of new threats. People are subjected to direct and indirect security threats, for which they need some extra safety measures. Attackers will use a mix of social engineering and other sophisticated tools to trick users which require more skill and work hours.

The web has become a staple for information sharing and processing, and web browsers are the popular interface for deliverance of information. Web browsers are at the heart of the security problems that affect users as they are an appealing target for attackers.

Web browsers are the full-blown software suites, for locating and displaying webpages on the internet. As we use web browser is the initial link to the rest of the internet, they are the crucial point of vulnerability for attack or exploit to happen. Protecting a browser from today's cyber threats is an important and daunting task. The browser based attacks originate due to poor security coding of web applications. The browsers record the statistics of online activities using cookies, cached pages, and history. Now, users choose browsers with most security patches. But it is impossible to know how secure they are until hackers have poked at them. Mozilla Firefox ,Apple Safari, Windows Internet Explorer, Opera and Google Chrome are the major browsers that people use to surf internet today. Thus, it is important to increase the security of the browser while browsing or mailing. Latest technologies bring new opportunities but the browser should be updated to defend newer threats and to prevent divulging of sensitive information.

This paper gives an overview of various browser security threats with main focus on tabnabbing attack. Further, an analysis is done with the existing anti-tabnabbing techniques with the proposed agent based method to detect tabnabbing.

The paper is structured as follows. Section 2 details about various browser security threats. Section 4 explain about the state-of-the-art tabnabbing detection techniques. Section 5 gives a brief idea about agent based method for tabnabbing detection. Implementation is explained in section 6 and experimental evaluation is done in section 7.Further, a comparative analysis is done in section 8. Section 9 concludes the paper.

2. Browser Security Threats

2.1. Clickjacking

Clickjacking[1] is an act of tricking web users into revealing sensitive credentials when browsing on seemingly ordinary web pages. These attacks prompt users to click on links unawares. An interesting offer or game is displayed to the user forcing her to click multiple times at specific places. There will be an invisible frame loaded, along with the displayed content and user thinks she clicked somewhere in the game or offer while the actual click falls on an invisible layer programmed to perform some other action. The phisher is "hijacking" clicks meant for one web page and forwarding them to another.

2.2 Cross Site Scripting (XSS)

In Cross Site Scripting (XSS) [2], a malicious piece of code injected into an ordinary site is used to bypass access controls and to impersonate somebody else. XSS attacks occur when a web application is used to send a malicious code to another user in the form of a browser side script.

XSS can be used to send a malicious script to a user whose browser will execute the script accessing cookies, session tokens, or other private information in the browser. HTML page content may also be rewritten.

XSS attacks steal a user’s session cookie or take advantage of one of the web applications frequented by the user. A fraudulent website may be used to deliver a malicious script to the victim’s browser or a phishing site operating under the authentic domain may send credentials back to the fraud. JavaScript is mostly used for malicious scripting.

2.3 Cross Site Request Forgery

Cross Site Request Forgery [3] is abbreviated as CSRF or XSRF, is an offensive practice of attacking a website in which an attacker masquerades as a legitimate user and sends malicious scripts to a website where the user is recognized as authentic. The attack is executed in such a way that a hacker inserts malicious codes into a link on a website that seem to be from a legitimate source. When the user clicks on the link, the embedded code is submitted as a client web request and is executed on the user's computer. CSRF is also known as one-click attack or session riding as it happens from the valid session of user's browser. This attack exploits the trust that a website has in the user's browser and the trust that a user has for a specific site. The danger is not with the victim’s browser or the site hosting the CSRF but in the affected web application. Whenever a valid session request comes from a browser, the server is unable to confirm if the request is authorized or not. The web server just process the request what it is received but it might have been issued without the knowledge of user from a website with a hidden

script. Requests to transfer funds or change email address can be generated from a user's browser using social engineering techniques. This is also a CSRF attack. They specifically target web applications like email clients, social media, e-commerce applications, and online banking interfaces.

2.4 Session Hijacking

Session hijacking [4] is a kind of phishing. Here the attacker waits till the user logs into a personal account after which the session is hijacked to do fraudulent transactions unknown to the user. Session hijacking works by hijacking the user's authenticated session by transferring it to a different machine or browser enabling the attacker to continue working in the victim's session. This attack is also known by 'cookie hijacking' as it exploits the valid computer sessions.

2.5 Social Engineering

Social engineering [5] is the process of gaining access to secure systems by presenting themselves as a trustworthy site or application. It is a crime where usernames, passwords or credit card details are stolen. Human's social skills like trust are misused here.The attack may be simple and personalized or complicated and large scale. Just like an antivirus program is updated, social engineer too evolve to create ewer frauds utilizing technology and human nature. Of all social engineering techniques, phishing is the most common.

2.6 Phishing

Phishing [6], the most prolific form of social engineering is an act of harvesting sensitive information from users by making counterfeited websites which impersonate legitimate ones. Phishing employs both social engineering and technical subterfuge to steal personal and financial account credentials of consumers. According to the recent APWG Phishing Activity Trends Report, both deceptive phishing and spear-phishing that targets specific business employees have been on the rise. Through the past decades, the number of victims has increased exponentially as phishers improvise tactics by exploiting loopholes in software.

2.6.1 Tabnabbing

Tabnabbing [7] is a kind of phishing where open tabs of a browser are impersonated. The name was coined by Aza Raskin. When a user keeps many tabs open at a time, she may forget which were they. When she returns to an earlier tab, it will display a webpage frequently used (eg. login page like gmail) and she enters the username and password. If the tab was nabbed, the information goes to the fraudster.

Tabnabbing is a dominant browser attack which is likely to deceive even the most incredulous web surfers as it exploits user's trust and inattention in browser tabs. Unlike other attacks, this deception technique happens on the behind 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 using different techniques. One of the methods to launch tabnabbing is via scripting support. This is done by exploiting JavaScript embedded in the webpage. Using JavaScript, it is possible to observe mouse and keyboard events to detect the user activity on that page. The JavaScript can be modified with an impersonation of a popular website, when the page has lost its focus and hasn't been interacted with for a while. The inactivity of a tab is identified using JavaScript

onblur and *onfocus* events. Once the *blur* event is fired, the script replaces the favicon, title and the page layout with a well known login look-a-like. When the user returns to the tab after a while and see a legitimate looking webpage, he has no need to think that the displayed page is fake. The user is prompted to enter the login details and the data thus collected are redirected to the attacker. The attack can be made more effective by monitoring the websites that the user has loaded in the past or in other tabs, and loads a simulation of the same sites. CSS history probing technique [8] can be used to correctly generate the exact URL that the user has visited recently.

In order to execute the attack, it is not always required to change the tab. The attack can be launched when the user minimizes his window or he is not interacting with the system for doing some other activities. The attacker can detect the inert tab using JavaScript code and replaces the currently focused page with a deceptive site. In some cases, he can fool the user without even changing favicon and title. In this case, the background details of the webpage will be loaded slowly to make an impression to the user that it is taking time to render the webpage. In worst case of the attack, URL itself is unchanged which can even fool the most security conscious web surfers who always check the URL before proceeding their web activities.

Alternate ways of demonstrating this attack rely on the use of HTML refresh meta tag [9] in predetermined time intervals. This technique does not need scripting support instead the webpage refreshes itself in the background and turns into a fake page. In this variant, the phisher could use *window.onerror* events to recognize the user's login activity. This method does not rely on users to switch a tab, as the phisher assumes that the user might have been changed the tab after a predefined time interval.

Tabnabbing attack can also be performed with the use of *iframes*. An HTML <iframe> tag [10] specifies an inline frame which provides a provision for nesting of documents. In other words, it helps to insert another document within the current HTML document. They can be placed anywhere in the web document.

3.Anti-Tabnabbing Techniques

Account Manager [11] is a Firefox plugin released by Mozilla for online identity management. This plugin stores the details of logins which are already created, and generating random passwords to make the logins more secure. Firefox add-ons NoScript [12] and YesScript [13] prevent websites from running JavaScript, Flash or other plugins and provides effective protection against malicious scripts, XSS, CSR and clickjacking attacks. But they do not provide protection in other browsers.

NoTabNab [14] is a Firefox add-on which protects users from tabnabbing attack by using the positioning of HTML elements of a webpage. This add-on looks out open tabs and track changes in page layout, favicon and title for every single tab until a new URL is loaded. The approach considers only the topmost page element and records its attributes values and attaches critical CSS information to these records. When a tab switch occurs, the operation is repeated and the add-on compares the new attribute values with the previously recorded one. If an impersonation has happened, an alert is given to the user about changes in its page layout, favicon or title to impersonate another page by highlighting the address bar in yellow or red according to the severity of attack. The problems related to this technique are follows. As the method doesn't keep data about all elements on the page, it is possible to trick the add-on by putting very small invisible elements under grid points and thereby bypassing the controls. Also the method is not effective if the document contains many iframes within each other as it needs to check each of these elements recursively and compare them with the already recorded data. Another problem in this technique is about resizing the browser. Few web pages are designed to re-layout themselves.

A signature based detection mechanism [15] is there to deal with tabnabbing. Java script vulnerability is defined by rules in this method. Changes in page layout, favicon and title are observed indirectly by observing mouse and keyboard events of a user. They use a signature based detection mechanism to deal with tabnabbing attack with an assumption that dynamic iframes are never used with *onfocus* and *onblur* event of JavaScript. As a first step, the source file is converted into a text file and then into tokens by submitting to a tokenizer. The script tags are extracted by the tokenizer and are given to the rule based system to check for vulnerabilities. The JavaScript code is considered as vulnerable if both *onblur* and o*nfocus* events are used with dynamically generated iframes within the prescribed time and also a mouse click is not used to detect if an iframe is in focus or not. But this paper focuses only on iframe elements which is not always necessary for a tabnabbing attack.

Tab-Shots [16] is an anti tabnabbing technique in the form of a browser extension. It saves screenshots of a tab at regualar internals. Each screen shot is divided into a number of tiles and compared with the previously saved copy. If there is a change, the changed tiles are highlighted. It also records and compares the favicon of a page. This extension falls short at detecting minute changes in a page.

TabsGuard [17] combined 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 and uses five heuristic-based metrics to measure the degree of changes made to the tree representation of each webpage whenever a tab loses focus. In this method, three parameters such as title, favicon, and the HTML Document Object Model (DOM) tree of the page are fetched at two time frames T1 and T2. T1 is the time when the page is fully loaded and T2 is the time when the user returns back to that tab after a tab switch. The values of each parameter are compared at two time frames to monitor the changes made to the page between these time frames. For comparing the HTML DOM trees of the page, the heuristic-based metrics such as common paths, cosine similarity, tag frequency distribution analysis, input fields added to the page and iframes in the page before and after tab switch are used to perform the comparison with respect to syntactical similarity. The comparison results are then analyzed using data mining techniques to find the outlier score. When the outlier score of a webpage with respect to the degree of changes become higher than the average outlier score, the page is detected as tabnabbing. Otherwise, it is detected as legitimate. If a page is detected as tabnabbing, an alert message is displayed to the user and adds the detected page to a local blacklist in the user's browser.

TabSol [18] is based on comparing the hash value of a webpage when the webpage is in focus and when it regains focus after different tabs are switched. TabSol uses a URL whitelist of legitimate pages. Whenever a webpage is opened, the method checks whether the URL (address) is present in the domain whitelist or not. It is in the whitelist means the page is safe. Otherwise, it is unsafe. The webpage hash value is calculated using SHA-1 algorithm and is stored. After a tab switch and return, TabSol recomputes the hash digest of the page and compares with the stored digest. If it matches, the webpage is secure and is added to the domain white-list. Otherwise, the page is considered as doubtful. Now, the method check for login forms in the suspicious page. If a login form is detected, then the page is classified as phishing and otherwise as genuine.

The tabnabbing detection system (TDS) proposed by Al-Khamis and Khalafallah [19] is a combination of content based and non-content based techniques which considered visual and structural features of a webpage. The method is implemented as an extension to Google Chrome browser. In this method, a profile is created for each opened tab which consists of important elements. When the tab is on focus from a nap, a second profile of the same tab is created and is compared with the old one to find any suspicious activity. The detection mechanism proceeds in two layers. The first layer compares the

title, URL, and favicon and the second layer compares the HTML structure of the webpage to detect an attack. In order to overcome undetected cases, the method also incorporates user education which will enhance internet users protection.

TabSecure [20] is an anti-tabnabbing technique used ward off attacks through fake websites and fake emails. This has three parts, phishing website detection, tabnabbing detection and email phishing detection. The first part watches the browser and performs an IP address check with reference to a database of phishing sites. If a match is not found, a webpage content analysis is done to find suspicious links in it. The email phishing detection module extracts emails into a text file and send them to a bayesian classifier. This classifier separates legitimate and non legitimate emails. The tabnabbing detection module compares the webpage when the tab is in focus and out of focus for five seconds. If there is a change a phishing alert is displayed. TabSecure has an accuracy of 93% but consumes more time due to the complexity of work.

Existing anti-tabnabbing methods detect the change in layout and warn the user only when the tab is on focus after a tab switch and also they focus mainly on the change in page layout, title and favicon but not much attention is given to change in URL. Beyond that, they fail to detect parallelism in attack detection.

4. Agent Based Method

The work proposed in this paper use a heuristic method where structural features of a webpage are analyzed as explained in [21][22][26]. The proposed method is a multi agent system uses agents to concurrently monitor the change in webpage layout at regular intervals in all 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 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. As shown in Figure 5.2, there are four operational agents in the system when a webpage is opened in a browser tab. The agents in this system are named T-agent , U-agent, M-agent and I-agent.

The autonomous agents arranged in three levels continuously monitor the presence of attack at regular intervals in multiple tabs. The agents are organized in a hierarchical manner with an aim to share and delegate activities and/or responsibilities. The operation of the proposed system proceeds in two main phases.

- Feature Extraction
- Feature Comparison

The workflow of the proposed system is illustrated in Figure 4.1 and is summarized as follows. The major functionality of the agent based system is performed by level 1 agents (T-agent and U-agent). When webpages are loaded in the browser tabs, T-agents in each tab perform Feature Extraction and Feature Comparison every 60 seconds. Feature extraction is a quantitative way of capturing 5 tuple information from a website such as text, image, favicon, title and URL. The feature extraction value of tuples during the first pass are stored as the expected value. Further a resemblance score is calculated for each tuple by comparing the subsequent feature extractions along with the expected value. Using these scores, a single resemblance score is derived that captures the similarity between the stored version and currently opened version of a webpage. Based on the resemblance score, webpages are classified as legitimate or phishing according to a threshold *t*.

5. Implementation

The software implementation of the proposed method used JADE software framework (JADE, 2016) in java platform. The experiments are performed using Core i3 @2.20 GHz processor, 4GB of RAM memory, JDK 1.8 and JADE 4.2.0 in Windows 7 platform.

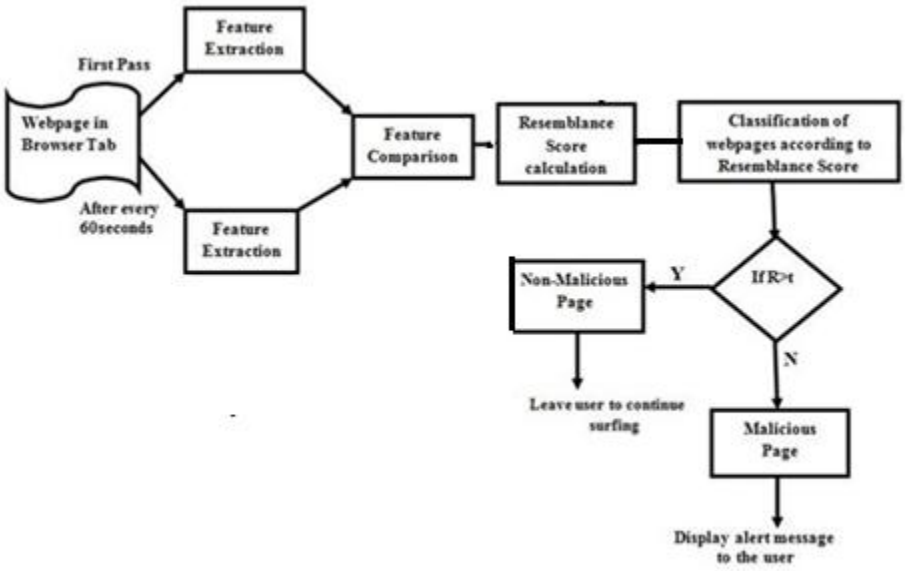

Fig. 4.1. Operational workflow of the proposed method

The JADE application consists of a set of agents which are given unique names and IDs. Agents can execute tasks and they communicate and cooperate with other agents by exchanging messages and beliefs. Agents run on a platform provide various services, such as message delivery, agent migration and resource management. A platform is composed of one or more containers, which can host multiple agents.

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. Experimental Evaluation

The positive data set consists of a set of common webpages with login forms such as money transaction sites, banking sites, web mail clients, credit cards, social networking sites etc. as tabnabbing targets webages which can provide confidential information of users. The approach used 2000 unique webpages with login forms from different sources.

To identify login forms, this method has used a heuristic based algorithm using HTML DOM. A login form is the typical sign of any webpage used by phishers which is characterized by FORM tags, INPUT tags and login keywords. INPUT fields are provided to enter user input and the login keywords give an appeal that the user is interacting with a login form. The method verifies that a webpage has any login form and then it is added to the dataset. This initial screening helps to avoid unnecessary method execution in webpages without having login forms. This approach adapted a login form finder implemented in CANTINA+ [23]. The proposed method has gathered 34 login and search keywords which can reveal its type.

For negative dataset,9 tabnabbing pages which are the fake versions of gmail, facebook, twitter, eBay, flipkart, hotmail, paypal, sbi and bradesco). To make a list of blacklisted URLs, a collection of verified phishing sites from phishtank [24]are taken. By simulating the attack, the relevant features from the webpages opened in various tabs are captured and recorded. 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. 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 (*FPR*) and False Negative Rate (*FNR*).FPR and FNR for various values of the threshold *t* are computed which is shown in Figure. They are calculated using the formulas given below:

$$FPR = \frac{FP}{(FP+TN)} \quad \ldots\ldots. (6.1)$$

$$FNR = \frac{FN}{(FN+TP)} \quad \ldots\ldots.(6.2)$$

where FP is false positives,FN is false negatives,TP is true positives and TN is true negatives.

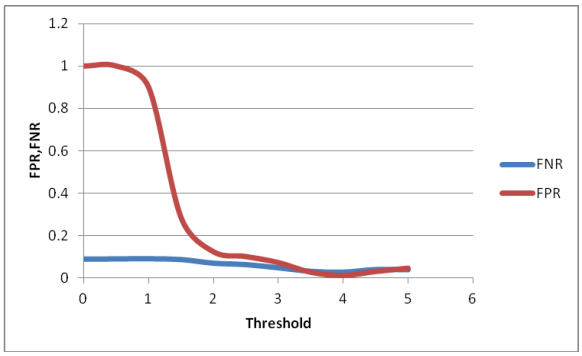

Fig 6.1 FPR and FNR in different thresholds

In figure 6.1, FPR and FNR is plotted with respect to varying threshold from 0 to 5. The results show that at threshold 4, the framework shows better result.

Figure 6.2 shows the percentage of false detections from negative dataset. It is seen that, the impersonated versions of email services (hotmail and gmail) give better result with no false detections. The percentage of false detections was mainly from paypal. This shows that the method could detect all the cases of tabnabbing launched using email services.

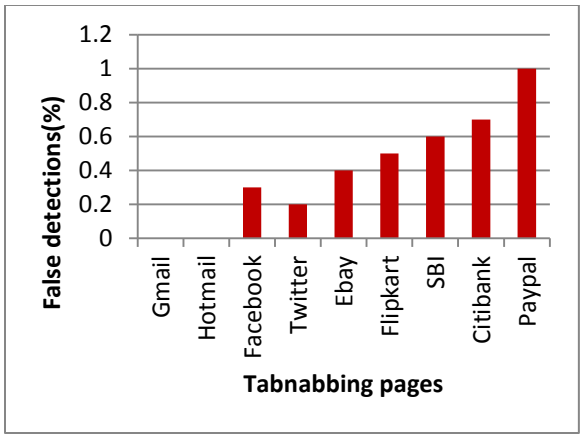

Fig 6.2 Number of False detections in negative dataset

6.1 Analysis of agent performance

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.

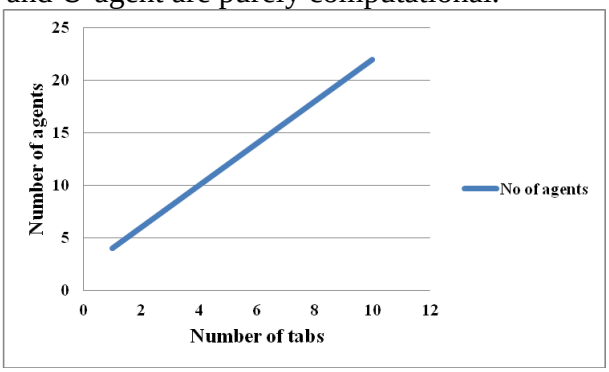

Fig 6.3 Number of agents vs Number of tabs

There are four agents in action while a browser tab is open. Figure 6.3 shows graph with the number of agents active plotted with the number of opened browser tabs.

The architecture of the proposed MAS[25] 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 1 agents.

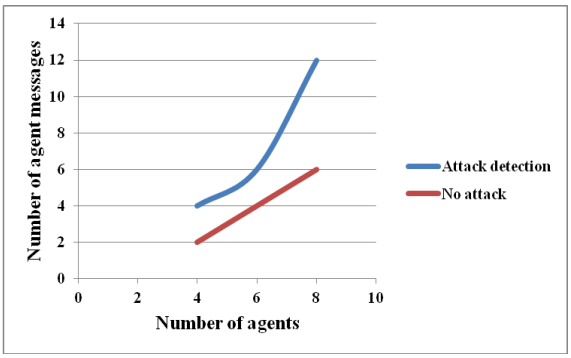

Fig 6.4 Number of messages sent vs Number of agents

Figure 6.4 shows the number of messages sent between agents in the case of attack detection and when no attack is detected. The number of sent messages will be more when an attack is found as there should be communication between each level of agents. When no attack is found, only the level 1 agents need to communicate for calculating the resemblance score of the currently opened webpage.

To evaluate the performance of the system in parallel attack recognition, multiple tabs are opened 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). Beyond that, there is a slight decrease in efficiency as it may cause delay in the system.

Table 6.1 shows the time taken by the method in milliseconds to calculate the resemblance score of three categories of webpages.

Table 6.1. Response time of the proposed method according to number of tabs

Number of tabs	Response time in milliseconds		
	Simple webpage	Medium webpage	Complex webpage
1	285	538	730
2	285	550	760
3	290	590	820
4	300	630	870
5	316	690	916
6	330	760	1020
7	382	811	1092
8	402	890	1180
9	440	960	1320
10	480	1070	1580

Category I consists of simple webpages (only textual contents with no images), Category II consists of medium webpages (with text and images) and Category III consists of complex webpages(with more images). The results show that the proposed method consumes more time in image comparison than textual comparison. Beyond that, the execution time is also increasing linearly with increase in the number of tabs. Still then, the method is able to finish its execution with a maximum time of 1580msec which is less than page load time of a normal website. The proposed method has used the following websites in different categories for calculating the response time.

Category I	www.gmail.com
Category II	www.facebook.com
Category III	www.bradesco.com

7. Comparative Analysis

Table 7.1 provides a comparative analysis of the existing anti-tabnabbing solutions as well as the proposed agent based approach with respect to the following criteria. The various anti-tabnabbing

techniques are compared in terms of the efficiency of techniques in multiple attack prevention, parallel attack recognition in multiple tabs of a browser etc. This table clearly shows that the proposed method stay unique from other methods and provides a novel solution to parallel attack recognition of multiple attacks in multiple tabs.

Table 7.1 Comparative analysis of existing anti-tabnabbing techniques with proposed method

Method	Technology in use	Browser	Use of Blacklist	Multiple attack prevention	Parallel attack recognition
NoTabnab(2010)	HTML layout	FireFox	X	X	X
Approach by Suri et al.(2011)	Signature based system	None	X	X	X
TabShots(2013)	Visual layout comparison by capturing screenshot	Chrome	X	X	X
TabSol(2014)	Hash value comparison	Chrome	X	X	X
Tabnabbing Detection System (TDS)(2015)	Profile creation and comparison using visual and structural features, User education	Chrome	X	X	X
TabsGuard(2015)	Textual layout comparison using HTML DOM	FireFox	√	√	X
TabSecure(2016)	Browser monitor	None	√	√	X
Proposed method	Structural features comparison, Multi Agent System	Chrome	√	√	√

8.Conclusion

Browser attacks have become very common and are likely to succeed against systems that are not designed to resist them specifically. This paper provides an overview of some of the techniques used by attackers to deceive users using browser as a platform. Further, a discussion is made on a recent browser attack called Tabnabbing and the sequence of steps to execute it. Tabnabbing is a modern and more sophisticated kind of phishing attack which targets a user who keeps many tabs open at a time. In addition, the paper has done a review on existing anti-tabnabbing solutions and a comparative analysis of the existing anti-tabnabbing techniques with the proposed method. It has been seen that the proposed method offers parallelism in attack detection which is not supported by any other techniques. The proposed method is able to resist multiple phishing attacks with few resources in hand and provides user intervention while implementing the technique where other methods fail.

References

[1] Huang, L. S., Moshchuk, A., Wang, H. J., Schecter, S., & Jackson, C. Clickjacking: Attacks and Defenses. *USENIX Security Symposium*, Aug 2012, pp. 413-428.

[2] Popa, R. A. *Server Side Security:Cross Site Scripting*. Retrieved February 2016, from http://www-inst.cs.berkeley.edu/~cs161/sp16/slides/2.8.XSS.pdf

[3] Yaakob, R., Joozdani, M., Abdullah, M. T., & Abdullah, A. Overview of cross site request forgery and client-side protection. *International Journal of Computer Technology and Applications*, *4*(4), Jul 2013,pp. 706-709.
[4]Johns, M. Session Hijacking Attacks. In *Encyclopedia of Cryptography and Security*, 2011, pp. 1189-1190.
[5] Krombholz, K., Hobel, H., Huber, M., & Weippl, E. Advanced social engineering attacks. *Journal of Information Security and applications, 22*, June 2015, pp. 113-122.
[6] Jagatic, T. N. Social Phishing. *Communications of the ACM*, Oct 2007, pp. 94-100.
[7] Raskin, A. *Tabnabbing: A new type of phishing attack*. Retrieved from http://www.azarask.in/blog/post/a-new-type-of-phishing-attack/.2010.
[8] Felten, E. W., & Schneider, M. A.. Timing Attacks on Web Privacy. *7th ACM conference on Computer and communications security*, Nov 2000, pp. 25-32.
[9] HTML meta tags. Retrieved from http://www.tutorialspoint. com/html/html-meta-tags.htm,Jan 2020.
[10] HTML iframe Tag. Retrieved from W3Schools: http://www.w3schools.com/tags/tag-iframe.asp, Jan 2020.
[11] Mozilla Phishing protection. Retrieved from http://www.mozilla.com/en-US/firefox/phishingprotection/, Jan 2020.
[12] No Script. Retrieved from No Script- JavaScript/Java/Flash blocker for a safer Firefox experience: http://noscript.net/, Jan 2020.
[13] YesScript. Retrieved from YesScript Firefox Add on: https://addons.mozilla.org/enUS/firefox/addon/4922, Jan 2020.
[14] Unlu, S. A., & Bicakci, K. NoTabNab: Protection Against The Tabnabbing Attack. *eCrime Researchers Summit (eCrime), IEEE*, Oct 2010, pp. 1-5.

[15] Suri, R. K., Tomar, D. S., & Sahu, D. R, An Approach to Perceive Tabnabbing Attack. *International Journal of Scientific and Technology Research*, Jul 2012, pp. 90-94.

[16] De Ryck, P., Nikiforakis, N., Desmet, L., & Joosen, W. TabShots: Client Side Detection of Tabnabbing Attacks. *8th ACM SIGSAC Symposium on Information*, May 2013, pp. 447-456.
[17] Hashemi, H. F., Zulkernine, M., & Weldemariam, K. TabsGuard: A Hybrid Approach to Detect and Prevent Tabnabbing Attacks. In *Risks and Security of Internet and Systems*. *Springer International Publishing*. Aug 2014, pp. 196-212.
[18] Singh, A., & Tripathy, S. TabSol: An efficient framework to defend Tabnabbing. *International Conference on IEEE, Information Tech- nology (ICIT)*, Dec 2014, pp. 173-178.
[19] Al-Khamis, A. K., & Khalafallah, A. A. Secure Internet on Google Chrome: Client side anti-tabnabbing extension. *Anti-Cybercrime (ICACC), First International Conference on IEEE*, Nov 2015, pp. 1-4

[20] Joshi, P., & Chatterjee, M. TabSecure: An Anti-Phishing Solution with Protection against Tabnabbing. *International Journal of Com- puter Networks and Applications (IJCNA), 3*(3), Jun 2016, pp. 63-69.

[21] Sarika, S., & Varghese, P. Parallel Phishing Attack Recognition using Software Agents. *Journal of Intelligent & Fuzzy Systems*, *32*(5), Apr 2017, pp. 3273-3284. [22] Sarika, S., & Varghese, P. Intelligent Agents in Securing Internet. *Journal of Internet Technology*, *19*(3), May 2018, pp 753-763.

[23] Xiang, G., Hong, J., Rose, C. P., & Cranor, L. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and Sys- tem Security (TISSEC), 14*(2), Sep 2011, pp. 21-43.

[24] PhishTank - Out of the Net, into the Tank. Retrieved from PhishTank: http://www.phishtank.com, Jan 2020.

[25] Sycara, K. P. Multiagent systems. *AI magazine, 19*(2), Jun 1998, pp. 79-92.

[26] Sarika, S., & Varghese, P. "An anti-phishing framework to defend Tabnabbing attack." *International Conference on Security and Authentication*. 2014. pp 132-135.