A FRAMEWORK FOR SMART TRUSTED INDICATORS FOR BROWSERS (STIB)

By

JUDE DESTI

B.S. Computer Information System, Florida A&M University, 2008

M.S. Software Engineering, Florida A&M University, 2010

Submitted to the Graduate Faculty of

Florida A&M University in partial fulfillment

Of the requirements for the degree of

Masters in Science

Florida A&M University

2010
This thesis was presented

by

Jude Desti

It was defended on

December, 2010

and approved by

Dr. Clement Allen, Associate Professor, Computer Information Sciences

Dr. Bobby Granville, Associate Professor, Computer Information Sciences

Dr. G. Thomas Bellarmine, Program Area Coordinator, FAMU-Electrical Engineering Tech.

Thesis Director Advisor: Dr. Hongmei Chi, Assistant Professor, Computer Info. Sciences
Copyright © by Jude Desti

2010
Web browsers currently have security indicators which provide security features that notify users of malicious or un-trusted websites. Most of these security indicators are normally synced with a blacklist database that contains a list with websites that are known to be malicious. When users surf websites that have already been comprised in a blacklist database, the web browsers’ security indicator then notifies the user with a warning message indicating that the desired website to be viewed has been identified as a malicious or un-trusted site, and then offers the user the option to continue or to exit the current site. Because these blacklist databases may not possibly contain every malicious website, users will come across a website they feel may be an un-trusted site, but have not received a warning message from the security indicator indicating otherwise. In this thesis, we propose an extension security indicator called Smart Trusted Indicators for Browsers (STIB) which will perform an extensive web activity history check on desired websites determining how often that website has been viewed or transacted to provide the user with more information about the site and to gain the confidentiality of the legitimacy of the website.

Keywords: Browser security, trusted Indicator, mal-ware, phishing, vulnerability, browser history.
# Table of Contents

TABLE OF FIGURES.................................................................................................................... VII

ACKNOWLEDGEMENTS.................................................................................................................. IX

1. INTRODUCTION/BACKGROUND.............................................................................................. 1
   1.1 ISSUES AT HAND.................................................................................................................... 3
   1.2 BLACKLISTING SERVICE...................................................................................................... 5
   1.3 WEB BROWSER HISTORY DATA............................................................................................. 6
   1.4 MY CONTRIBUTIONS.............................................................................................................. 7
   1.5 THESIS OUTLINE .................................................................................................................... 8

2. VULNERABILITY OF BROWSERS ............................................................................................... 9
   2.1 INTERNET EXPLORER (IE).................................................................................................... 10
   2.2 MOZILLA FIREFOX.................................................................................................................. 12
   2.3 GOOGLE CHROME.................................................................................................................. 14
   2.4 SAFARI ................................................................................................................................... 15
   2.5 MOBILE WEB BROWSERS ..................................................................................................... 16

3. RELATED WORK.......................................................................................................................... 18
   3.1 VISUAL SECURITY INDICATORS............................................................................................ 18
   3.2 GRAPHICAL SECURITY INDICATORS..................................................................................... 20
   3.3 DYNAMIC SECURITY SKINS.................................................................................................. 21
   3.4 OTHER WORKS ....................................................................................................................... 22

4. SMART TRUSTED INDICATORS FOR WEB BROWSERS............................................................ 24
   4.1 RATIONAL FOR APPROACH................................................................................................. 25
## TABLE OF FIGURES

Figure 1: Process of a typical Phishing Spoofing Attack................................................................. 2

Figure 2: Snippet of Firefox indicatory warning of a malicious site identified from the blacklist
database. ........................................................................................................................................... 13

Figure 3: Google Chrome's warning indicator for malicious websites. .......................................... 15

Figure 4: (left) Identity indicator used in Firefox 3 Beta 1 and modified Firefox 3 Beta 1. (right) The
identity confidence button in its three different states.................................................................. 19

Figure 5: Screen-Shots of secure sites with logo in TrustBar......................................................... 21

Figure 6: The browser displays the visual hash as a border around the authenticated website. ......... 22

Figure 7: Basic Design of the STIB extension on the browser. ..................................................... 28

Figure 8: ERD Diagram for the places.sqlite data base................................................................. 37

Figure 9: Mathematical PageRanks (out of 100) for a simple network. Page C has a higher PageRank
than Page E, even though it has fewer links to it; the link it has is of a much higher value............. 41
Table 1: Phishing attacks within the past 12-month peak. ........................................................................3

Table 2: The Q3 2009 Socially Engineered Malware Test Report Summary generated by NSS Labs for the past 12 months. ................................................................................................................................. 10

Table 3: Q3 2009 Phishing Test Report Summary generated by NSS Labs for the past 12 months... 11

Table 4: Content with ‘browser.XUL’ file. .................................................................................................................. 31

Table 5: Contents within the ‘chrome.manifest’ file.................................................................................................. 32

Table 6: List of fields that will be provided to user when STIB extension is triggered. ................................. 48

Table 7: Result presented after STIB extension button is triggered................................................................. 50
ACKNOWLEDGEMENTS

This work was supported by U.S. Department of Education award P120A070081. My sincere gratitude goes to Dr. Hongmei Chi for guiding, advising and introducing me to the topic of security indicators for web browsers.
1. INTRODUCTION/BACKGROUND

Security is becoming increasingly apparent and more of a major priority in aspects of computer security. In the context of security systems, web browsers are often thought of as an "open door" for information flow through the World Wide Web. These pieces of information include users' personal sensitive credentials and corporate information which allows large amount of everyday business and other sensitive transactions. These mass amounts of activities demand users to provide online credentials needed to carry out these transitions. Everyday users are online making personal purchases, socializing on social networks and performing online banking transactions on various websites. Knowing that mass amounts of sensitive information are being inserted into these websites and then being sent across the World Wide Web becomes an attraction to malicious users of the web. These malicious users develop different types of web attacks aimed to trick users by posing as a trusted site, or a "look alike" site in effort to gain the users' sensitive data.

Web Spoofing and Phishing attacks are two of the most highly favored attacks used against web browsers. Spoofing Attack is a situation in which one person or program successfully masquerades as another by falsifying data and
thereby gaining an illegitimate advantage (Li & Wu, 2003). A Webpage Spoofing, also known as Phishing Attack, is another type of spoofing attack. In this attack, a legitimate web page such as a bank’s site is reproduced in "look and feel” on another server under control of the attacker. The main intent is to fool the users into thinking that they are connected to a trusted site, for instance to harvest user names and passwords. This attack is sometimes usually used in spam e-mail messages with a link to the attackers’ fake website (Herzberg & Jbara, 2008). See Figure 1.

Figure 1: Process of a typical Phishing Spoofing Attack

Unfortunately, users of the web lack knowledge of these malicious activities. Most naïve users are unaware of security indicators that are presented by web browsers. The intent of the indicators is to effectively allow users to identify fake or non-trusted websites. Most users are unaware of these indicators and malicious activities; they carelessly browse the web and provide their sensitive information to random sites believing that every web site is a trusted website. As long as careless web surfing takes place, phishing and
spoofing attacks will continue to expand, further compromising users’ credentials (Help Net Security, 2009). This growth is shown in Table 1.

<table>
<thead>
<tr>
<th>PAST 12 MONTHS</th>
<th># OF PHISHING ATTACKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAY 09</td>
<td>11,887</td>
</tr>
<tr>
<td>JUN. 09</td>
<td>13,021</td>
</tr>
<tr>
<td>JUL. 09</td>
<td>13,212</td>
</tr>
<tr>
<td>AUG. 09</td>
<td>16,164</td>
</tr>
<tr>
<td>SEPT.09</td>
<td>17,365</td>
</tr>
<tr>
<td>OCT. 09</td>
<td>17,900</td>
</tr>
<tr>
<td>NOV. 09</td>
<td>15,127</td>
</tr>
<tr>
<td>DEC. 09</td>
<td>15,596</td>
</tr>
<tr>
<td>JAN. 10</td>
<td>18,820</td>
</tr>
<tr>
<td>FEB. 10</td>
<td>18,503</td>
</tr>
<tr>
<td>MAR. 10</td>
<td>17,579</td>
</tr>
<tr>
<td>APR. 10</td>
<td>18,080</td>
</tr>
<tr>
<td>MAY 10</td>
<td>16,541</td>
</tr>
</tbody>
</table>

This all implies that there is indeed an extreme need of effort towards developing better indicator tools or extending the current existing functionalities for the security indicators on web browsers that will more effectively allow user to identify fake websites.

1.1 ISSUES AT HAND

The value of personal data to hackers and malicious users of the web is increasing and access to these data allows them to conduct fraudulent activities even with the slightest of information. Most of our modern web browsers like Internet Explorer (IE), Firefox, Google Chrome, etc. are currently embedded
with security indicators that notify the user of malicious or non-trusted websites. These security indicators are sourced from a blacklist database which contains websites that are known to be malicious and not trusted. These malicious websites are appended to these blacklist databases via users that initially visit these sites than reports the site being malicious to some third party resource which will then update these blacklist databases. Apparently not all malicious websites will be reported to these third party resources and updated to the blacklist database; these databases cannot logically comprise every malicious website. There are users that lack knowledge of such resource that will allow them to report these websites. There are also cases where users will visit these websites, not knowing that it is a Phishing site, and continue thinking that the activities or transactions they performed were successfully processed. Such actions are sometimes either never identified or identified months or years later after finding out that some fraudulent act was performed with the user’s credentials. These fraudulent acts still continue to grow as shown before in Figure 1.

Websites that are not categorized as malicious sites are either acknowledged as trusted or unknown websites. The trusted websites are identified with Secure Sockets Layer/Transport Layer Security (SSL/TLS) Certifications which provides the user with non-forgeable proof of the identity of the website they are connecting to. It also ensures the data exchanged between the web browser and the site the user is attempting to connect to
cannot be read, deciphered or decrypted by a third party (hacker) that may be tapping in to the data exchange between the Web Browser and the Website.

Unknown websites are neither identified as trusted (SSL/TLS Certified) nor malicious websites. These websites are normally recognized as an HTTP (Hypertext Transfer Protocol) on the URL which indicates that the data inserted through the web browser is being sent to the connected website via plain text, not encrypted text. These websites are also not recognized by the web browser’s security indicator as a malicious site. Therefore, these sites are unknown but don’t necessarily mean that the website is not a malicious site.

As mentioned before, malicious websites are reported by users then updated to a blacklist database which is linked to the web browser’s security indicator. Malicious websites that are not reported and updated to these blacklist databases therefore remain as an unknown website which leads to the issue of how a user will be able to determine whether or not if they are connected to a legitimate website.

1.2 BLACKLISTING SERVICE

A blacklist database is an approach that most of our modern web browsers currently use for fighting phishing and spoofing attacks. This approach is to maintain a list of known phishing sites and to check websites
against the list. Currently, Internet Explorer (IE), Firefox, Google Chrome, Safari, and Opera all contain this type of anti-phishing security protocol. There are four different types of ways that these blacklist databases collect malicious phishing sites: (1) URLs are extracted from spam/phishing filters at mail exchange gateways, (2) URLs are extracted from user reports of phishing emails, (3) phishing websites are identified by heuristics, and (4) user reports of phishing websites (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). In our approach with providing the user with history statistics of about a particular requested website, any website that is not identified within a blacklist database and is not SSL/TLS Certified will be classified as an "UNKNOWN" website. We will then perform further investigation on the "unknown" website by providing the user with history information which will allow the user to know more about the site and gain the confidentiality of the legitimacy of the site.

1.3 WEB BROWSER HISTORY DATA

A browser's history is a log of sites that a user has visited. Each time a user access a file through your web browser, the browser caches (i.e., stores) it and creates a cookie. A cookie is a file created by a web browser, at the request of a website, which is stored on the computer. These files typically store user-specific information such as selections in a form, shopping cart contents, or
authentication data. The history log file will usually have an expiration date associated with it. Once that date has passed, the cookies stored by the client will automatically be deleted. In our approach to provide the user with history statistics of an unknown website, we will utilize this log file by scanning the requested site against this file to be able to evaluate the number of times the website was visited. This log file will also be utilized to evaluate other similar trusted websites that may have been intended to be viewed by the user.

1.4 MY CONTRIBUTIONS

After evaluating and researching how modern web browsers deals with identifying the difference between a trusted website and a malicious website, I realized that there still remains websites that are not categorized as trusted nor malicious sites. These "Unknown" websites are neither SSL/TLS Certified nor identified by the web browser’s security indicator as a malicious site. But, as mention before, an "Unknown" website does not necessarily imply that it is not a malicious website.

My focus and contribution to this topic is geared towards the "Unknown" websites that users may pass over as a trusted websites which may turn out to be a malicious site. My approach will aim to perform extensive web activity history checks on Unknown websites determining how often that website is viewed or transacted to provide the user with more information.
about the site and the confidentiality of the legitimacy of the site. This approach will also consist of similarity checks on URLs against sites that are normally viewed on the current machine on a daily basis to avoid possible user typing mistakes which could lead to connecting to a website "look alike" aimed to compromise the users' credentials.

1.5 THESIS OUTLINE

In Chapter 2, we will discuss and explore the security indicators that our most popular web browsers use. In Chapter 3, we will then showcase some of the related efforts and researches that are aimed toward providing better web browser indicators that will allow users to better recognize and distinguish the difference between a trusted website from an untrusted website. Chapters 4 and 5 will then elaborate on our approach and method to recognizing trusted websites and provide our implementation scheme to those approaches. We will conclude with Chapter 6 why our approach will be a more superior method in being able to notify a user whether or not they are surfing through trusted websites.
2. VULNERABILITY OF BROWSERS

In this section, we explore and identify several different web browsers and how their security indicators handle web browser attacks such as Web Phishing and Spoofing attacks. We will diagnose and explore the different methods taken by the security indicators to identify trusted or malicious sites for each browser. We will see that modern web browsers offer additional layers of protection against web-based threats.

Web browser protection contains two main functional components: In-the-cloud reputation-based system and internal browser request reputation information. However, not all of the browsers take the same approach to browser protection. In-the-cloud reputation-based system searches the internet for malicious websites and categorizes them according to content. This is sometimes done automatically, manually, or sometimes a combination of both. Internal browser request reputation information resides on the web browser and the information that it request from the in-the-cloud system about specified URLs. Once results from the request regarding a URL listing reports as a malicious site, redirection to warning messages is executed.
2.1 INTERNET EXPLORER (IE)

Internet Explorer is a free web browser from Microsoft and is also one of the most popular internet browsers today. Microsoft claims that IE is the fastest and safest browser for web users. The Q3 2009 Socially Engineered Malware and Phishing Test Report Summary generated by NSS Labs reported that IE is, and continues to be the best protected browser against Socially-Engineered Malware, downloading delivering malicious payloads, and Phishing attacks (NSS Labs, Security Certified, 2009). Results from the report showed that IE had a block rate of 81% for Socially-Engineered Malware, while other browsers declined from previous test results. See Table 2. Results for Phishing attacks showed that IE had a block rate of 83%. See Table 3.

Table 2: The Q3 2009 Socially Engineered Malware Test Report Summary generated by NSS Labs for the past 12 months.

<table>
<thead>
<tr>
<th>INTERNET BROWSERS</th>
<th>Q1 2009 TEST (% of Social Engineered Malware Blocked)</th>
<th>Q3 2009 TEST (% of Social Engineered Malware Blocked)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer 8 (IE)</td>
<td>69%</td>
<td>81%</td>
</tr>
<tr>
<td>Mozilla Firefox 3</td>
<td>30%</td>
<td>27%</td>
</tr>
<tr>
<td>Apple Safari 4</td>
<td>24%</td>
<td>21%</td>
</tr>
<tr>
<td>Google Chrome 2</td>
<td>16%</td>
<td>7%</td>
</tr>
<tr>
<td>Opera 10 (Beta)</td>
<td>5%</td>
<td>1%</td>
</tr>
</tbody>
</table>

The Socially-Engineered Malware result were based upon experientially validated evidence gathered during 12 of 24x7 testing, performed every 4 hours, over 70 test runs, each one adding fresh new malware URLs. Internet Explorer improved 12% from previous test taken,
evidence of ongoing concerted efforts made in the SmartScreen technology (NSS Labs, Security Certified, 2009).

Table 3: Q3 2009 Phishing Test Report Summary generated by NSS Labs for the past 12 months

<table>
<thead>
<tr>
<th>INTERNET BROWSER</th>
<th>PHISHING ATTACKS % Blocked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer 8 (IE)</td>
<td>83%</td>
</tr>
<tr>
<td>Mozilla Firefox 3</td>
<td>80%</td>
</tr>
<tr>
<td>Opera 10 (Beta)</td>
<td>54%</td>
</tr>
<tr>
<td>Google Chrome 2</td>
<td>26%</td>
</tr>
<tr>
<td>Apple Safari 4</td>
<td>2%</td>
</tr>
</tbody>
</table>

*SmartScreen® Filter* is the technology that Internet Explorer uses that focuses on preventing phishing and malware attacks. As a reputation-based feature, SmartScreen blocks new threats from existing malicious sites that traditional anti-virus or anti-malware signatures does not block. This mechanism is able to block in the navigation experience and in the file download experience depending on the situation. With this type of control, SmartScreen is able to block entirely malicious sites, portions of sites or just a single malicious download on sites like social networking or file-sharing sites. The known malicious sites and malware downloads are sourced by Microsoft Internal and 3rd party database. The database also includes sites with Extended Validation Certificates which attest to the identity of legitimate business. When such sites are requested, the background of the URLs is highlighted in green indicating a secure and/or trusted website and increases the appearance of legitimacy to the user.
2.2 MOZILLA FIREFOX

Firefox is another popular web browser from Mozilla. Firefox is an open-source browser that is free, small and fast. It is based on Mozilla code and is one of the most standard-compliant browsers available on Microsoft Windows (Window 98 – Vista), Mac OS X, and Linux (Google, Inc, 2007). The Q3 2009 Socially Engineered Malware and Phishing Test Report Summary generated that Firefox caught 27% of the threats issued to the browsers. Far fewer then the Internet Explorer, Firefox came in second place on blocking malicious sites and downloading malicious payloads. Unlike IE, Firefox enables built-in Phishing and Malware Protection by default. Once users enter an address in the URL, that address is checked against a blacklist that Firefox downloads periodically (Felegyhazi, Kreibich, & V., 2010). If sited on the blacklist, Firefox displays a popup which warns the user that the visited site is suspected to be a fraudulent site and provides the user the option to leave the site or to ignore the warning. See Figure 2.
Like several other web browsers, Firefox also utilize the Google SafeBrowsing API. Google SafeBrowsing for Firefox is an extension that alerts the user if a web page that that is being visited appears to be asking for the user’s personal or financial information under false pretences. This tool combines advance algorithms with reports about misleading pages from a number of sources and warns the user if they have encountered a page that is trying to trick them into disclosing personal credentials. For Firefox, this is an optional feature that sends the URL information to Google to determine the likelihood of being scammed.
2.3 GOOGLE CHROME

Google Chrome is a fairly new web browser developed by Google. Chrome was released in 2008 and has been one of the top three most widely used browsers with 7.16% of the worldwide usage share of web browsers, according to Net Applications (Net Applications, 2010). In aspects of security, Chrome security architecture design is focused on mainly protecting the user throughout their browsing experience. Unlike the other web browsers that uses SSL to guarantee their website identity and prevent malicious web users from tampering with what users see and the information they provide, Chrome alerts the user when an error prevents websites from establishing a fully secure connection by analyzing the identity, connection and visit history (Google Chrome, 2010). Chrome uses technologies such as Safe Browsing API, sandboxing, and auto-updates to help with the protection against Phishing and malware attacks.

The Safe Browsing service for Chrome is an approach that downloads updates of two blacklist databases (one for Phishing and one for malware) and warns the user when there is an attempt to visit a malicious website identified from the black list. Figure 3 shows a snippet of the warning page that a user will see once attempting to visit malicious website.
The Q3 2009 Socially Engineered Malware and Phishing Test Report Summary generated that Google Chrome blocked 26% of the live Phishing attack attempts and blocked just 7% of live Malware threats (NSS Labs, Security Certified, 2009). Overall, Google Chrome remained in fourth place in blocking both Malware and Phishing attacks.

2.4 SAFARI

Safari is a web browser developed by Apple in 2003 for the Mac OS X v10.2. Safari is the first official "out-of-beta" version created as the default web browser for the Mac System. As many of Apple's products, Safari is renowned for its sleek design and ease of use. Apple's goal for Safari is usability, speed, standard compliance, and integration with OS X. In aspects of Security, Safari does not hold up a strong edge in security matters. The Q3
2009 Socially Engineered Malware and Phishing Test Report Summary generated that Safari caught 21% of the live threats (NSS Labs, Security Certified, 2009). Unfortunately, Safari fell third for the Social-Engineer Malware test and fell dead last to all the web browsers (IE, Firefox, Chrome, & Opera) tested for the Phishing test. When a suspicious phishing site or sites with harbor malware is identified by safari, it sometimes gives one warning pop-up message about the suspected nature of the site and sometimes prevents the site from loading. In aspects of SSL Certified sites, Safari is also weaker than its competitors in identifying digital certificates traffic. Safari does warn of invalid digital certificates, but it isn’t nearly as superior as the other top browsers. Unlike other browsers who alter the entire web page with red or multicolored warnings, Safari, once again, only warns the user once with a small pop-message indicative of the suspicion of the site. Safari fails to point out Extended Validation Certificates and sometimes never highlight the domain name making it more difficult to tell a malicious site from a trusted/secure site.

2.5 MOBILE WEB BROWSERS

Mobile web browsers are becoming increasingly popular in this era and are utilized in many different ways. Mobile web browsers are also another source of target for phishing and spoofing schemes. Some of the more popular
mobile web browsers utilized today are BOLT, Opera Mobile, Opera Martini, Skyfire, Safari, Mozilla’s Minimo, and many others. Some mobile web browsers are capable of displaying a full web page on the small mobile screen instead of a scaled down version. Other mobile web browsers are typically stripped-down browsers that must be small and efficient to accommodate the low memory capacity and low-bandwidth of the mobile devices. The displaying of the websites on web browsers are achieved by first parsing it on a server that partly or completely renders the page and sends the results to the mobile device.

For most of these mobile web browsers, the security features performs very similar to laptop and desktop web browsers security features. BOLT is a popular mobile web browser used on most blackberry devices and other smart phones as well (Bolt Browser, 2009). Some of the browser security features on BOLT are similar to features on other mobile web browsers. BOLT uses a 128-bit secure SSL protocol connection to access secure web pages. For filtering, it utilizes a server filter that protects users from spyware, malicious or faulty addons, online fraud and spoof websites. BOLT along with several other mobile web browsers utilizes the certificate error notification mechanism. The certificate is an electronic document that can help identify a website’s owner, and can help users make decisions about trusting the site with personal or financial information. The indicator warns users about certificate error when there is a problem with a certificate or the server’s use of the certificate.
Statistics have shown and proven that there are major efforts needed towards web browser security. Malicious web users are gaining more intelligence in aspects of web security and are developing more complex Phishing and Spoofing tools for the compromising of user data over the web. In this section we will showcase some of the major efforts that are geared towards preventing these malicious web users from being able to compromise user data over the web.

3.1 VISUAL SECURITY INDICATORS

Jennifer Sobey with the Ottawa-Carleton Institute for Computer Science, School of Computer Science, Carleton University, has been involved with research in aspect of SSL/TLS Certification of web sites (Sobey, 2008). They introduced the Extended Validation (EV) SSL certificates for Internet Explorer 7.0, web browsers which included new indicators to convey information about different types of certificates. The Extended Validation SSL certificate indicator is a more noticeable indicator that draws users' attention.
A snippet of the indicator is shown in Figure 4. The indicator they evaluated and extended upon was on the Firefox 3.0 Beta 1 version, which already had a small buttonized portion of the browser chrome (chrome is where all the menus and toolbars on the browser window are located) to the left of the URL bar that contains a web site's icon. Sobey designed the EV indicator which would be displayed in the same location but was larger and displayed information about the web site's identity on the button. The indicator would trigger a pop-up information box that provided identity information for the web site based on the site's certificate.

Figure 4: (left) Identity indicator used in Firefox 3 Beta 1 and modified Firefox 3 Beta 1. (right) The identity confidence button in its three different states
3.2 GRAPHICAL SECURITY INDICATORS

Amir Herzberg and Ahmad Jbara from Bar Ilan University has been involved in research dealing with web security measures in aspects of users entering sensitive credentials into fake websites (Herzberg & Jbara, 2008). They have introduced and contributed a security and identification indicator called, the TrustBar (a browser extension). This extension allows users to assign a name or logo to identify SSL/TLS-protected sites. The goal of the Trustbar is to present a highly visible, graphical interface, establishing securely the identity of a web site. The Trustbar was implemented as an extension for the open-source Mozilla TM and Firefox TM browsers. Trustbar allow the site and the certificate authority to be identified by the logo of the website. A snippet of the indicator is shown in Figure 5. It controls a significant area, located at the top of every browser window that is large enough to contain highly visible logos and other graphical icons for credentials. This approach prevents many security indicators spoofing attacks as described by Li and Yongdong (Li & Wu, 2003), where a spoof site opens windows to hide browser indicators like padlocks and location area and to overwrite them with misleading indicators.
Rachna Dhamija and J.D. Tygar from the University of California, Berkeley have proposed Dynamic Security Skins, a scheme that allows a remote web server to prove the website’s identity in a simplified way for users to verify (Dhamija, Tygar, & Hearst, 2006). First, the browser extension provides a trusted window in the dedicated to username and password entry. A photographic image is used to create a trusted connection between the user and the window to prevent spoofed windows and texted entry fields. The second part of the scheme allows the remote server to generate unique abstract image for each user and each transaction. This image creates a “skin” that automatically customizes the browser window for identified authenticated web
pages. A snippet of the indicator is shown in Figure 6. This proposal simply allows the user to visually identify a trusted site with a "skin" image that appears on the border of the browser window.

![Figure 6](image.png)

*Figure 6: The browser displays the visual hash as a border around the authenticated website.*

### 3.4 OTHER WORKS

Bryan Parno, Cynthia Kuo, and Adrian Perrig from Carnegie Mellon University are involved in proposing using a trusted device to perform mutual authentication that eliminates reliance on perfect user behavior, thwarts Man-in-the-Middle attacks after setup, and protects a user’s account even in the presence of key loggers and most forms of spyware (Parno, Kuo, & Perrig, 2006). The goals of this tool are to prevent attacker from modifying, and viewing the users account. These goals are with the assumption that users can
be trusted to correctly identify sites at which they wish to establish accounts with. This assumption is made due to the fact that phishing attacks generally target users with existing accounts. The trusted device takes form of a cell phone, PDA, or even smart phones. Users cannot readily disclose the authenticator on the cell phone to a third party, and servers will refuse to act on instructions received from someone claiming to be a particular user without presenting the proper authenticator. Assuming that there is a secure connection between the cell phone and the browser, the trusted device tool provides an additional authenticator, such that the Man-in-the-Middle attacker must compromise the device and obtain the users password to access the users' account. This approach simply reduces the reliance of users protecting their selves against phishing attacks and will provide a mechanism based on cryptographic operations on a trusted mobile device.
4. SMART TRUSTED INDICATORS FOR WEB BROWSERS

We propose an extension to the current web browser security indicators that will provide users with more information about a website. Users sometimes come across sites that they may feel distrustful about, because they are unfamiliar with the site, the site not having an SSL/TLS Certification or maybe just the appearance of the site. Because security indicators not being able include every malicious website in their blacklist database, sites that may be distrustful to the user may not be distrustful to the security indicators which will not strike a warning message to the user. In this case, we propose the ŠTIB indicator button which will provide the user with history information on how often the site is viewed or transacted upon providing the user with more in dept information about the site which will lead to the user gaining the confidentiality of the legitimacy of the site.

The ŠTIB is an extended security indicator that further helps determine the legitimacy of a website through a history check of the site. The history check would be scanned upon how many times a site has been view by users over a specified period of time, and upon the amount of times the current active machine has viewed the site. Note that the STIB does not notify the
users of a malicious site such as spoofing and phishing sites. It simply extends history information on the viewing activities for the website. Determining the statistics on how often a certain website is viewed by users will allow the STIB to provide the user with more in-depth information to guide the user into making a "smarter" choice throughout their browsing experience. The STIB will also perform similarity checks on URLs against sites that are normally viewed by the machine on a daily basis. This feature will allow the user to notice that there might have been a typo of the web address and allow the user to view similar website that has been viewed before. For example, a user may check "www.cnn.com" on a daily basis. If the user mistype the web address with "www.cmnn.com" a website that has never been viewed before by the machine and is not recognized in the blacklist database, then the STIB will provide warning message with the similarity and history information of the website allowing the user to make a "smarter" choice on whether or not they should continue to view the website.

4.1 RATIONALE FOR APPROACH

The reasoning for this approach is to utilize the data stored in the browser’s temporary history directory to help users make reasonable choices when browsing the web and decrease the possibility of the user becoming a victim of a Phishing or Spoofing attack. Users of the web become victims of
these attacks by browsing through these unknown websites that may very well be malicious websites but may not have yet been reported as one. The security indicators on these web browsers are blind to these possible malicious websites because they are only linked to a blacklist database system that includes websites that have been reported to be malicious. By utilizing these temporary history data, the STIB extension will help the users to make smarter choices when viewing unknown websites by providing them with history statistics of how often the unknown website has been viewed in the past.

4.2 TARGETED ENVIRONMENTS

This approach with the STIB conducting history checks against websites will have more of an impact in public environments or networks. Utilizing machines that are randomly used by a wide variety of users will increase our data stored in our web history file which will then allow our STIB extension to provide more efficient and effective history information to the user. This approach will also cause a more effective and accurate ranking when our STIB extension provide ranking levels for each website.
4.3 RESEARCH APPROACH

To show that our Smart Trusted Indicator for Browsers will provide history information of URLs, we will take the following approach:

- Research and determine the different ways to discover the number of views a URL receives.

- Create the policies that we will use to categorize the viewing result numbers for the URLs to either being "Below Normal Viewing", "Average Normal Viewing" or "Above Normal Viewing". These results will result from both the current machine being used and the network that machine is on. (This approach will allow the user to have a better understanding of results returned by the STIB extension)

- Create the policies to identify common or similar URLs used by the machine by analyzing the temporary history data of URLs visited.

- Implement these policies to a browser indicator button located in a visible and recognizable location on the browser's window.

Below is a basic architectural frame work of how our STIB will perform on web browsers:
Figure 7: Basic Design of the STIB extension on the browser.
5. IMPLEMENTING STIB

In this section we elaborate more on the implementation phase for the STIB extension and all the different methods and strategy that was taken to develop and compose the information that will be presented to the user. We will also present some results and snapshots of our STIB extension and also talk about the installation process. This implementation process is broken up in four phases:

<table>
<thead>
<tr>
<th>PHASES</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1: Install STIB Skeleton</td>
<td>This phase is where we describe how install the STIB extension button and the configuration needed for the development environment.</td>
</tr>
<tr>
<td>Phase 2: Determine Access Count</td>
<td>This phase is where describe who were are able to acquire number of times a website is viewed</td>
</tr>
<tr>
<td>Phase 3: Ranking Strategy</td>
<td>This phase is where we describe our ranking strategy for each website.</td>
</tr>
<tr>
<td>Phase 4: Similarity Check</td>
<td>This phase is where we describe how we are able to present the similar website of the current website.</td>
</tr>
</tbody>
</table>
5.1 INSTALL STIB SKELETON

As mentioned previously, we're proposing an extension security indicator that will provide users with viewing history information from desired sites. To implement this proposal we will be utilize Mozilla Firefox, an open source web browser which will enable us to develop our STIB extension. We'll use Mozilla Developer Center (MDC) and Mozilla Developer Network (MDN) forums to guide us with this implementation phase (MDN, 2005).

To start our development phase we first start off by installing the skeleton of our extension which is basically setting-up and creating our development environment. An extension, also known as Add-On, is a package of files that modifies Mozilla Firefox’s appearance and/or behavior. Extensions are capable of inspecting and configuring the underlying framework of websites utilizing 3 languages: JavaScript, XUL, and CSS. For our case, Firefox extension will allow us to modify the web browsers window by adding our STIB extension button and allow us to utilize the history APIs in Firefox to manipulate and execute queries against the browsers places.sqlite database system file (history file). To create our skeleton for our extension, we simple went to Add-on Builder (Add-on Developer Hub) which would perform the extension packaging. This process generates the install manifest, which we'll elaborate more on later, and allows you to select your interface element like toolbars, preference dialog, toolbar button, main menu command, context
menu command, or sidebar support. Once this process is complete, your basic extension skeleton and your development environment is prepared.

We mentioned before that to develop our STIB extension for Firefox, we will utilize 3 techniques, XUL, JavaScript, and CSS to accomplish our task. XUL is Mozilla’s XML-based language which allows us to build feature-rich cross platform applications. For our STIB extension, this is the file that creates our STIB extension button on the browsers’ window. We will use a CSS file to alter the display of the XUL file. Our JavaScript file is where we implement what we want our STIB extension button to perform and execute. The JavaScript file is where the heart of our work is accomplished.

![XUL file content](image)

Table 4: Content with ‘browser.XUL’ file.

| 1) | <xml version="1.0"/> |
| 2) | <xml-stylesheet href="STIB.css" type="text/css" /> |
| 3) | <overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"> |
| 4) | <script type="application/x-javascript" src="chrome://STIB/content/browser.js"/> |
| 5) | <toolbarpalette id="BrowserToolbarPalette"/> |
| 6) | <toolbarbutton id="STIB-toolbarbutton2" class="STIB-toolbarbutton-default" label="STIB" type="button" style="list-style-image:url('chrome://STIB/content/STIBicon.jpg')" onclick="STIB.onCommand()"/> |
| 7) | </toolbarpalette> |
| 8) | </overlay> |

Table 4 presents the content within the XUL file. As you may see, this is the file that we will use to reference our JavaScript file and CSS file (lines 2
& 4). The toolbarpalette on line 5 indicates where our STIB extension button will be located on the web browsers window. This button is represented by a jpg picture file that we call STIBicon.jpg located in our content directory, which we will talk about later in this section. The onclick element on line 6 signifies that once this button is triggered, it will execute the STIB.onCommand() method within the JavaScript that we referenced.

In the process of preparing our development environment, a Chrome package was created that packages all the GUI structural elements that go into the XUL application. All the elements including our extension button and our alert message that appears when the extension button is triggered will be embedded in this package. A chrome manifest file is also created during the preparation of the development environment. This file registers the chrome package with Firefox to use the extension. Below is the content that we have inserted in our chrome manifest file to register our STIB extension.

Table 5: Contents within the ‘chrome.manifest’ file.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>content</td>
<td>STIB</td>
</tr>
<tr>
<td>2)</td>
<td>locale</td>
<td>STIB</td>
</tr>
<tr>
<td>3)</td>
<td>overlay</td>
<td>chrome://browser/content/browser.xul</td>
</tr>
</tbody>
</table>

Line one of the chrome manifest file indicates that Firefox will find all my chrome/content directory packages which refers to my XUL, JavaScript and CSS applications. Line 2 indicates that my en-US language interpretation
information will be located in my chrome/local/en-US directory package. Line three of my chrome manifest file indicates that my STIB/content/browser.xul file should overlay the browser/content/browser.xul file. This basically means to join my XUL file with the browser's default XUL file to add my STIB extension.

Here is the structure layout for the packages on a windows environment:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Users[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\default\extensions[extensionName]\</td>
<td></td>
</tr>
<tr>
<td>install.rdf</td>
<td>file that gives information about extension</td>
</tr>
<tr>
<td>chrome.manifest</td>
<td>file that registers packages to Firefox</td>
</tr>
<tr>
<td>chrome\</td>
<td>chrome package</td>
</tr>
<tr>
<td>content\</td>
<td>package used to contain the main XUL and JavaScript source files</td>
</tr>
<tr>
<td>locale\</td>
<td>package used to contain language data that can be translated</td>
</tr>
<tr>
<td>skin\</td>
<td>package used to include source files used as visual elements in the GUI, including style sheet and images</td>
</tr>
</tbody>
</table>

Another file that is created when the extension skeleton is being prepared is the install.rdf file. This file is called the install manifest which gives basic information about your extension, and is required in order for the extension to be installed in Firefox. This is the file which we will use to describe what our STIB extension will execute and its purpose. This file is also stored under the chrome package. (See APPENDIX A for full content of the install.rdf file)
Once this environment has been correctly prepared, we are now able to start implementing exactly what we intend for our STIB extension to execute once the extension button is triggered.

5.2 DETERMINE ACCESS COUNT

To start off our implementation phase we first had to research on how, and what our modern web browsers were capable of storing to its history file. We first found out that the address of every web page you visit is recorded in your browser history file (Stewart, 2000). These history files are normally utilized for:

- Revisit Sites if a user cannot remember a website that he/she has visited in the past before, user can search browser history by date visited, or by text string search.

- Search user can search through browser history for a keyword to find common website that includes the keyword that has been visited before.

- Delete user can choose to erase entries that are stored in the browser history file.

To locate these history files stored by the web browsers differs depending on the browser. With Internet Explorer, you can access the history
data by simply navigating through: Favorites > History Tab or simply clicking Ctrl-H or Ctrl-Shift-H. To locate the actual history file, the user will have to navigate through:

C:\Users\[username]\AppData\Local\Microsoft\Windows\TemporaryInternetFiles\.

With Mozilla Firefox, you can access the history data simply by navigating through: History > Show All History or Ctrl-Shift-H. To access the actual history file, the user will have to navigate through:

C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[DefaultProfile]\places.sqlite

With Mozilla Firefox, the history file is represented by the places.sqlite file which stores annotations, bookmarks, favorite icons, input history, keywords, and browsing history. This is a database system file called places that not only store web history pages but bookmarks as well. Sqlite is an ACID-compliant embedded relational database management system contained in a relatively small C programming library (Newman, 2004). This is the database that we will utilize with Firefox to query the amount of times that a particular website was viewed.

To determine the amount of times a particular website was viewed, we have to first understand the data base that we would be querying from and also understand its attributes. The places.sqlite data base system file for Firefox includes a number of tables to query from. Below is a list of tables for the places.sqlite data base system file and ERD Diagram for the tables:
- moz_anno_attributes - Annotation Attributes
- moz_annos - Annotations
- moz_bookmarks - Bookmarks
- moz_bookmarks_roots - Bookmark roots i.e. places, menu, toolbar, tags, unfiled
- moz_favicons - Favourite icons - including URL of icon
- moz_historyvisits - A history of the number of times a site has been visited
- moz_inputhistory - A history of URLS typed by the user
- moz_items_annos - Item annotations
- moz_keywords - Keywords
- moz_places - Places/Sites visited - referenced by moz_historyvisits
Figure 8: ERD Diagram for the places.sqlite data base.

To access this data bases we used the JavaScript language to access the toolkit history service which provides Firefox different methods for adding, editing, deleting, and browser history (Places Developer Guide, 2005).
This is the jump-off point for querying and searching the Firefox web history file. This connection service is set to execute once our user clicks the STIB extension button on the web browser window.

Once the connection is made to our places.sqlite data base system file, we want to first declare which website the user wishes to generate history statistics on. We performed this task by acquiring the current web address on the browser windows'URL field and declare that string value to a variable.

```
1) var query = history.getNewQuery();
2) query.domain = window.content.document.location.host;
3) var result = history.executeQuery(query, history.getNewQueryOptions());
```

We than were able to execute the query, as shown on line 3 above, to search the moz_places table to find an exact match of the URL address. This execution would return an object that will allow us to access its properties. The properties included in this object that we wanted to provide to the user was the websites' title, URL, the number of times visited, and the last time the website was visited.
As you can see from the code above, we first want to be sure that the URL returned from the query was an exact match as the current website. If this case were true we were than able to assign the returned objects' properties to local variables which would enable us to utilize those variables in our message dialog box that is provided to the user.

5.3 RANKING STRATEGY

In our approach to providing users with viewing history for certain website, we also wanted to be able to rank these websites based upon the average time each website was accessed on the local machine. Since our first phase of our STIB implementation was to provide the user the amount of times a website was accessed, this approach would allow the user to have a better understanding of the results returned from the first phase. For example, once the user clicks on the STIB extension, the user would not only be provided the amount of times a particular website was viewed, but would also be ranked
based off the average amount of times websites are normally viewed on the machine.

There are many different strategies used for ranking websites. One very popular page ranking algorithm is called PageRank. PageRank is a link analysis algorithm used by Google internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents such as the World Wide Web, with the purpose of measuring its relative importance within the set (PageRank, 2001). It is considered by more than 500 million variables and 2 billion terms embedded within each webpage. Based off of the level of importance of a website, websites are also able to influence the importance of other websites that it is linked to. A web page that is considered to be of high importance is more than likely to appear at the top of the search results when users are utilizing Google’s search engine. Below, Figure 8 is a diagram which represents the elements (websites) with its percentage of importance compared to other elements.
Figure 9: Mathematical PageRanks (out of 100) for a simple network. Page C has a higher PageRank than Page E, even though it has fewer links to it; the link it has is of a much higher value.

The page ranking approach that we took for our STIB extension will only be utilizing the web browsers' temporary history file. Our ranking categories for each website would be characterized either:

- Below Average: this category will represent websites that fall under the minimum average viewing range for the current web browser.
- Average: this category will be represented within a number range.
Above Average: This category will represent websites that are above and beyond the maximum average viewing range for the current web browser.

To evaluate the average viewing count for each website, we composed an operation that would get the access count for each unique website in the history file, and then added it all up to divide that number by the total number of unique websites.

\[
\text{Average Access Count} = \frac{\text{Total sum of each websites' access count}}{\text{Total number of websites}}
\]

We then wanted to add constraints to the Average Access count to be within a certain range. For example, if the Average Access Count returned 50, we then wanted the average range to be constrained within 30 as the minimum average range, and 70 as the maximum average range or anything proportionate to this range. This approach would not limit the restraint of the average viewing count to exactly one number. Below is the JavaScript coding that we developed to implement this algorithm.
1) var AvgRangeMul = .4;
2) var AvgRange;
3) var OverallViewCount = 0;
4) var TotalSitesViewed = 0;
5) for (var i=0; i < resultNode.childCount; i++) {
   var CNode = resultNode.getChild(i);
   // gets overall total of each websites' access count
6) var OverallViewCount = OverallViewCount + CNode.accessCount;
   //the overall Average range
7) var AvgOf_OverallViewCount = OverallViewCount / resultNode.childCount;
8) AvgRange = AvgRangeMul * AvgOf_OverallViewCount;
9) var MinAvgRange = AvgOf_OverallViewCount - AvgRange;
10) var MaxAvgRange = AvgOf_OverallViewCount + AvgRange;

As you see, we set the variable $AvgRangeMul = .4$ on line 1 because as we mentioned before, we want the average range to be proportionate to the example above and do what we multiply the overall access count to .4 (line 8), which would compute the range number. This range number will then be added to the overall average number to compute the maximum average range, and also subtracted from the overall average number to compute the minimum average range (line 9 & 10). If the Access Count for a website was to compute a number less than the minimum average range, it would then be categorized as "Below Average" viewing. However, if the Access Count for a website was to compute a number within the minimum average range and the maximum average range, it would then be categorized as an "Average" viewing. And finally, if the Access Count for the website were to compute a number greater than the maximum average range, it would then be categorized as an "Above Average" viewing.
5.4 SIMILARITY CHECK

Our STIB extension was designed to be able to simply provide users with web history statistics about certain websites which would be of aide to the user when he/she is uncertain or unfamiliar with a website. Being able to view the amount of times an unfamiliar website was viewed will allow the user to be more cautious about the website depending on its popularity which is represented by the number of times it has been viewed.

In addition to the web history statistics, we also realized that users sometimes stumble to these unfamiliar websites by typing the wrong website address in the URL field. In many cases like this, users may then utilize our STIB extension for more information about the website. The information provided by the STIB extension may return a Řòð count for the amount of times that the website was viewed. In this case, our user would only be set back to square one where the user still has no knowledge or any type of information regarding the unknown website. Therefore, we extended the STIB extension to provide similar websites that may have been visited before by the machine. This method will allow the user to be aware of any possible mistakes or typos that may have been inserted into the URL address field.

Most similarity check techniques use key words or phonetic algorithms to search against large databases. Phonetic algorithms (Phonetic algorithm) are algorithm for indexing or words by their pronunciation. One popular phonetic
algorithm used for key word similarity is Soundex. Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English. The purpose of this algorithm is for homophone words to encode to the same representation so that they can be matched despite the minor differences in the spelling of the words. This is very well known algorithm that is the standard feature of MS SQL and Oracle.

Levenshtein distance is another popular algorithm for word matching. Levenshtein distance is a metric for measuring the amount of difference between two words (Levenshtein distance). This algorithm basically determines the minimum number of edits needed to modify one string to match another. For example, the levenshtein distance between řlettingô and řputtingô is 2 since the edits would include, substituting řlô with řpô and řeô with řuô to make the first string (letting) an exact match with the second string (putting).

With our STIB extension, we wanted to conduct a key word check against all website stored in the web history file. The search would then return the first 3 web address found in the web history file, excluding any exact match (only want the search to return similar web addresses with the key word). To accomplish this method we first had to select the key word that we want to search for. In our code we declared the host name of the web address (without the extension .com or .org or etc.) as our key word. For example, if the web address is řwww.Mozilla.orgô we would pull the řMozillaô as our key word to be searched. Below is the JavaScript implementation of this process.
//spliting the url address to find a specific word

1) var string = window.content.document.location.host;

2) var stringArray = string.split(".");

    //specify the key word to serch for when serching for similarity websites

3) SimQuery.searchTerms = stringArray[1];

Notice that in order to pull the host name alone without its extension, we had to split the string and store the separated parts of the string (the web address) within an array. The split function would conduct its split every time a character in the string matched the dot character (\`\.\`) shown on line 3. We then needed our key word searched not only against all the web addresses in the history file but also against each websites\`\`title as well. This strategy allowed us to broaden our searching scope to search through two properties of the objects (websites) returned. This type of search is represented at line 3. The \`\`SimQuery.searchTerms\`\` property will search for the key word that it has been assigned to at both the objects\`\`URL address and title properties.

The web history file for Firefox browser does not only store web page\`\` but also stores every local file that has been open by the browser. Therefore, when we execute our query to find the websites with the key word located either in the web address or the title, we also have to restrict the search to only search through websites that starts with either \`\`http://www.\`\` or \`\`https://www.\`\` Below is an example of how we implemented this process
| Line 1 | var SimOptions = SimHistory.getNewQueryOptions(); |
| Line 2 | SimOptions.maxResults = 4; |
| Line 3 | var SimResultContainerNode = SimResult.root; |
| Line 4 | SimResultContainerNode.containerOpen = true; |
| Line 5 | var SimURL = new Array(); |
| Line 6 | for(var i=0; i < SimResultContainerNode.childCount; i++) { |
| Line 7 | var SimChildNode = SimResultContainerNode.getChild(i); |
| Line 8 | if(SimChildNode.uri != window.content.document.location) { |
| Line 9 | if((CNode.uri.indexOf("http://www.")!=-1) || (CNode.uri.indexOf("https://www.")!=-1)) { |
| Line 10 | SimURL[i] = SimChildNode.uri; } } |

As you may notice on line 2, we set the max results returned equal to 4 because in the case of the same current URL address being returned, we wouldn’t want to count that address as one of our top three similar websites. On line 8, we then make the decision on which 3 websites that we will use as the similarity sites out of 4 sites that were returned; excluding the site that matches the current site that we are attempting to find similarities to. On line 9 is where we confirm that the similarity websites are either an HTTP://www or HTTPS://www website.

As mentioned before, this similarity extension to our STIB implementation will allow our users to be aware of any possible mistakes or typos made in the URL field. Providing the user with similar websites to the web address that was inserted in the URL field will allow the user to notice websites that they may have intended to view which have also been viewed in the past.
Here are the fields that our STIB extension will provide to the user once the STIB extension button is triggered. (See APPENDIX B for screen shots of STIB extension)

Table 6: List of fields that will be provided to user when STIB extension is triggered.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Title</td>
<td>Websites' title on the browser window</td>
</tr>
<tr>
<td>Websites' Domain Host</td>
<td>Websites Domain Name</td>
</tr>
<tr>
<td>Current Website</td>
<td>Websites full URL address</td>
</tr>
<tr>
<td>Number of times Visited</td>
<td>Amount of time site has been accessed</td>
</tr>
<tr>
<td>Last Time Visited</td>
<td>Date and Time the last viewed</td>
</tr>
<tr>
<td>The Websites’ viewing Rank</td>
<td>Rank site: Below Average, Average, or About average</td>
</tr>
<tr>
<td>Similar Websites Viewed</td>
<td>List top three web sites that are similar to current site IF websites access count returns 0.</td>
</tr>
</tbody>
</table>

5.5  EXAMPLE CASES

In this section, we will show how our STIB extension will be utilized by providing you with an example cases study.

CASE STUDY:

Bob, a student at Florida A&M University, goes to the library to assess the internet. Bob decides to browse his email account to view any new emails that he may have. He notices that he has a new email in his inbox from a known source, Bill. Bill tells Bob in the email that he should click on a link
(www.school.com) in the email if he wants to save over 80% of his money on school supplies. Bill tells Bob in the email that everybody knows about the link and that everybody is saving a lot of money by registering with the website. Bob than clicks on the link and is redirected to the website. The website looks somewhat legit to Bob but he is still little curious. Bob is also a Computer Science major and knows that just because a website me appear legit does not necessarily means that it is legit. He notices that the browser did not warn him that the website was considered to be malicious. He also noticed that the website was also not SSL/TLS certified with the HTTPS protocol indicating that the connection is not a secured connection. Bob has no knowledge of this website. Familiarized with our STIB extension button, Bob decides that he needs some history information about this website and click on the STIB extension button located on the browsers menu bar.
Table 7: Result presented after STIB extension button is triggered.

The history information that is provided to Bob by the STIB extension indicated that the website has never been accessed and was ranked "Below Average" viewing on the machine. Bob than realizes that, for a website that Bill claims to be popular and that everybody is accessing to save money does not seem to be true. Now, at this point, Bob has a little more knowledge about this website and decides to no longer continue to view the website. He later found out that Bill did not have any knowledge of the email that was sent to Bob and that the email was a spam email trying to capture some of Bobs personal credentials. The STIB extension successfully helped Bob make a smarter decision during his browsing experience.
In conclusion, our STIB extension is still in its infant stage. In the future, if we could combine more history data from more sources with blacklist databases, local users will be able to benefit even more from our STIB extension for these situations and from browsing experience through the increasingly popular cloud computing.
6. FUTURE WORKS

In future, STIB can be extended in the following two ways:

Throughout our implementation phase of our STIB extension, we realized that limiting our data source to only the machines’ web history file may also be limiting the efficiency and accuracy of our page ranking and access counts for the websites. This approach restricts us to accessing data from one source which is the machine that the user is using. For future work and research, instead of only utilizing the web history file on one machine, we plan on establishing a way to utilizing the HTTP packages that are traveling throughout the network that the machine is on. This approach would broaden our data resource capability and allow our STIB extension to provide a more efficient and more accurate page ranking mechanism. This also will allow our access count for each website to be sourced from several machines instead of the users’ local machine.

Future work will also enable us to implement our STIB extension to other open source web browsers such Google Chrome, Kazehakase, Netsurf, Arora, and many more. Implementing our extension on these different browser will also allow us expand STIB extension to different environments such as
UNIX, Linux, Mac OS X, and even mobile web browsers. As wireless browsing via mobile devices continues to increase, it will also become bigger targets for spam and spoofing attacks. Implementing our STIB extensions will also become beneficial to the mobile web browsers.

Especially for cloud computing, browser security is becoming more important ever. Dell, through its Kace unit, is making available free Web browser security software that works by creating a protective "sandbox" on the desktop to isolate the user's desktop from malware or other harmful actions that might be encountered browsing the Web. Our implementation could be included in this product as well.
7. CONCLUSIONS

In conclusion, we believe that our approach with the Smart Trusted Indicator for Browsers will be beneficial to users that will need more information about a particular website for the confidentiality of the website being legitimate. As mentioned before, most of the current browsers’ security indicators are able to identify malicious sites by comparing the URLs to a blacklist database system. We use the local data on the machines to remind and strike caution to users about the possibility of browsing through malicious websites. Since these blacklist database systems will not be able to include every known malicious website, our STIB extension will extend some additional history information about a website allowing the user to gain knowledge about the site compared to having zero knowledge about the website. STIB is an alternative way to give local users more information for the websites that they will visit. In the future, if the STIB extension is combined with other security indicators for browsers, it will include local, network and blacklist information. We hope that this approach will prove to be an extra aide for users to guide them into making smarter viewing choices throughout their browsing experience.
This is the install.rdf file which enables us to register our STIB extension to Mozilla Firefox Web Browser. This file is also called the install manifest. As you will see this file gives basic information about the extension, and is required in order for the extension to be installed in Firefox.

1)  <?xml version="1.0" encoding="UTF-8"?>
3)  <Description about="urn:mozilla:install-manifest">
4)  <em:id>STIB@Jude.Desti</em:id>
5)  <em:name>STIB History Extension</em:name>
6)  <em:version>1.0</em:version>
7)  <em:creator>Jude Desti</em:creator>
8)  <em:contributor>Hongmei Chi</em:contributor>
9)  <em:description>"Smart Trusted Indicators for Browsers". An extension that gives you viewing history and statistics information on a web site. This will allow viewers to make better decisions during their browsing experience to avoid being a victim of malicious websites.</em:description>
10) </em:description>
11) <em:targetApplication> <!-- Firefox -->
12) </Description>
13) <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
14) <em:minVersion>3.5</em:minVersion>
15) <em:maxVersion>4.0b8pre</em:maxVersion>
16) </Description>
17) </em:targetApplication>
18) </Description>
19) </RDF>
APPENDIX B

This is a screen shot of our STIB extension (circled in red) being utilized by a user for an unfamiliar website www.school.com. Since the number of times accessed for the website returned 0, the STIB extension provided the user with some similar websites that had the key word School in the websites’ Title name or the URL address.
This is the close-up screen shot of STIB extension shown above.

![Screen Shot of STIB Extension](image)
This is a screen shot of our STIB extension being utilized by a user for an unfamiliar website www.Justin.tv. Since the number of times accessed for the website returned 5, our STIB extension evaluated that number to be in the average ranking range and provided that information to the user.

This is the close-up screen shot of STIB extension shown above.
This is a screen shot of our STIB extension being utilized by a user for an unfamiliar website www.twitter.com. Since the number of times accessed for the website returned 211, our STIB extension evaluated that number to be above average ranking range and provided that information to the user.

This is the close-up screen shot of STIB extension shown above.
This is a screen shot of our STIB extension being utilized by a user for an unfamiliar website "www.kentucky.com". Since the number of times accessed for the website returned 0 and was unable to find any similar websites, our STIB extension evaluated that number to be below average ranking range and notified the user that there were no similar websites found.

This is the close-up screen shot of STIB extension shown above.
APPENDIX C

HISTORY OF WEB BROWSERS

The Web Browser, also known as the Internet Browser, is a software application that retrieves present, transfer information resources on the World Wide Web. Although web browsers are primarily used for browsing the web, it is also utilized for accessing information provided by web servers and private networks or files on a local machine.

World Wide Web was the first ever web browser developed by Tim Berners-Lee in 1991. It was the World Wide Web that then brought together a variety of new software and hardware technologies. In 1993, NCSA Mosaic was the first graphical web browser that led to an exceedingly increasing usage by users. Soon after that, Marc Andreessen, the leader of the Mosaic team at NCSA, soon started his own company, called Netscape, and released the Mosaic-influenced Netscape Navigator in 1994, which quickly became the world's most popular browser, accounting for 90% of all Web use at its peak.

Microsoft soon responded with its browser Internet Explorer in 1995, initiating the industry's first browser war. This new move allowed Microsoft a huge percent of dominance in the operating system market that then led to dominance in the browsing market.
In 1996, although it never achieved widespread usage, Opera was invented and had a large share of the fast-growing mobile phone web browser market. It also was later available on several other embedded systems including Nintendo’s Wii video game console.

In 2002, a free open source software version of Netscape developed called Mozilla, which was the internal name for the old Netscape browser. Mozilla has since gained in market share, particularly on non-Windows platforms, largely due to its open source foundation, and in 2004 was released in the quickly popular Firefox version.

The most recent and very popular web browser on the market was created by Google called Chrome which was released in 2008. Its market share has tremendously increased since then and is now one of the most popular web browsers in the web browser market.
REFERENCES


