Spoof Guard to Avoid Phishing

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Ramakrishna Reddy. Vodeti
Fall 2007

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. Dulal Kar
Committee Member

Dr. Long-zhuang Li
Committee Member
ABSTRACT

Identity Theft has become one of the major crimes in the world since the beginning of the decade. Phishing is a form of identity theft. Phishing (or identity fraud, hijackings) occurs when a phisher (hacker) wrongfully acquires or uses another person's personal data, typically for their own financial gain. Spoof Guard is an Anti-phishing tool implemented to detect Phishing websites which are constructed to steal an Internet user’s private information, masking the user’s knowledge with false impressions.

A new version of Spoof Guard is designed which overcomes one of the major disadvantage of the earlier version and also implements new features that are user friendly. The mostly widely used phishing attacks are implemented in this project.

This research project can be extended to help numerous organizations by exploring the different steps and precautions that can be taken to save their customers from identity theft.
# TABLE OF CONTENTS

**ABSTRACT** .................................................................................................................. ii

**TABLE OF CONTENTS** .................................................................................................. iii

**LIST OF FIGURES** ...................................................................................................... vi

1. **INTRODUCTION AND BACKGROUND** .............................................................. 1
   1.1 Identity theft. ......................................................................................................... 1
   1.2 Phishing ................................................................................................................ 2
       1.2.1 The Phishing Process .................................................................................... 4
       1.2.2 Cases of Phishing Attacks .......................................................................... 7
   1.3 Types of Phishing .................................................................................................. 9
       1.3.1 Malware-Based Phishing ............................................................................ 9
       1.3.2 DNS-Based Phishing ................................................................................ 10
       1.3.3 Deceptive Attacks ..................................................................................... 11
   1.4 Why Phishing Works? ......................................................................................... 13
       1.4.1 User-Side Limitations ............................................................................... 13
       1.4.2 System-Side Limitations .......................................................................... 14
   1.4 Previous Work? .................................................................................................. 16
       1.5.1 Spoof Guard ............................................................................................... 16
       1.5.2 PwdHash .................................................................................................. 16
       1.5.3 Spoof Stick ............................................................................................... 16
4.1.1. Test.................................................................46
4.1.2. Results..........................................................48

4.2 Frame Testing.......................................................48
4.2.1. Test.................................................................48
4.2.2. Results..........................................................49

4.3 Email Configuration Testing.....................................49
4.3.1. Test.................................................................49
4.3.2. Results..........................................................50

4.4 Tool Survey..........................................................51

5. CONCLUSION..................................................................53

ACKNOWLEDGEMENTS..................................................55

BIBLIOGRAPHY & REFERENCES..........................................56

APPENDIX A. TERMINOLOGY...........................................58

APPENDIX B. HEADER CODE............................................59
LIST OF FIGURES

Figure 1.1 Phishing Reports in May’06 – May ‘07 [Anti-Phishing]…………………………3
Figure 1.2. Top 10 Phishing Hosting Countries.................................................................4
Figure 1.3 Phishing Process.........................................................................................6
Figure 1.4 Advanced Key Logger ..............................................................10
Figure 1.5 A Typical Deceptive Attack .................................................................12
Figure 1.6 Basic Relation between User’s Mental Model and System Model [Wu 2005]...........................................................................................................15
Figure 2.1 Prompt Shown by Spoof Guard that the Yahoo Site Opened by the Browser is a Spoof Website .................................................................19
Figure 2.2. Spoof Guard Architecture..............................................................26
Figure 2.3. Spoof Guard User Interface .................................................................29
Figure 2.5. Options Tab .....................................................................................31
Figure 3.1. Password Check ..............................................................................39
Figure 3.2. Webpage with a frame..............................................................41
Figure 3.3. Confirmation Warning asking the user if to send the spoof information......43
Figure 4.1. Spoof Page containing three password fields for one particular user-id........47
Figure 4.2. Spoof Web Page having a frame.........................................................49
Figure 4.3. Outlook prompt after a spoof webpage is found ..............................51
Figure 4.4. Statistical Analysis of the Survey..........................................................52
1. INTRODUCTION AND BACKGROUND

1.1 Identity Theft

Identity Theft is an attack through which a hacker steals compromised information from an Internet user for a financial or personal gain. Identity theft had turned out to be a most productive activity for Internet criminals. The consequence of these kinds of traps on the Internet is a loss of millions of dollars all over the world. Identity theft is the fastest growing fraud, affected hundreds of normal Internet users from the past decade, resulting in over billions of losses at an average cost of $30,000 per victim.

“From January through September of 2003, nearly ten million Americans were identified as victims of identity fraud, resulting in over $5 billion in victim out-of-pocket losses and $48 billion in losses to businesses and financial institutions according to a Federal Trade Commission report” [Arnold 2001].

For a well established thief on the Internet, identity theft is a most attractive and common form of activity with the goal to obtain a financial gain. It has been noted from the Anti-phishing website that lot of time, energy and money was spent by numerous organizations like credit reporting companies, law enforcement organizations and financial institutions to avoid phishing activities.

The act of identity theft starts when the attacker tries to disclose personal information from an Internet user without user’s knowledge. Personal information includes information such as name, date of birth, address, phone numbers, credit card details, bank account information, and social security numbers. Victims of these kinds
of crimes face consequences such as loss of money, jobs, and sometimes get arrested in spite of their innocence.

Software Professionals are facing a tough game from the phishers to save the Internet users from identity thefts. During the establishment of a secure connection networking managers are trying to minimize the opportunities for a phisher to do the business. “When attempting to minimize the opportunities for identity theft, it is imperative to guard our personal information. This information includes but is not limited to our social security number, maiden name (yours and your mother's), date of birth, past addresses, and driver's license number” [Groves 2002].

Having the disasters of Identity Theft in consideration, a number of laws have been enacted to dissuade Identity Theft and save privacy information. Unfortunately, implementation of these laws did not show up with much difference on the growing crime rate of Identity Theft.

1.2 Phishing

Phishing is a form of Identity Theft where a phisher tries to attack a user on the Internet to acquire compromised information for a financial gain. Phishers utilize social-engineering and spoofing techniques to obtain sensitive information from a user. “The word “phishing” originally came from the analogy that early Internet criminals used email lures to “phish” for passwords and financial data from sea of Internet users.” [NGS 2004].

These techniques include processes like Web-spoofing where the user is prompted to enter personal information. The phisher creates an impression on the user that the information is given to a legitimate organization and makes the user to
compromise in giving the personal information. Phishing is distinguished from offline Identity Theft like card skimming and dumpster driving, but both the online and offline activities have the same target to grab confidential information. There is an increase in the frequency of phishing attacks and their sophistication [Anti-Phishing].

Figure 1.1. Phishing Reports in May’06 – May ‘07 [Anti-Phishing]

The figure 1.1 shows the statistical form of the number of phishing reports received between May 2006 and May 2007. [Anti-Phishing]

Phishing is complex phenomenon that includes social factors as well as technology. There is no way to prevent phishing from the Internet. However, proper use of applied technology can come up with reduced risk of Identity Theft [Emigh 2005]. Phishing includes numerous social and technical factors and opportunities like:

- Monitoring potentially malicious activity;
- Authenticating email messages;
- Detecting the unauthorized use of trademarks;
- Improving the security patching infrastructure;
- Using personalized information to authenticate an email directly to a user;
- Detecting a fraudulent Website;
- Using a mutual authentication protocol;
- Forcing passwords to be site-specific etc.

![Top 10 Phishing Sites Hosting Countries](image)

**Figure 1.2. Top 10 Phishing Hosting Countries** [Anti-Phishing]

The figure 1.2 shows the study of researchers in China on the Top 10 phishing hosting countries in the world. The United States stands first with 23.74% and China stands second with 22.94%. According to the study, 23917 unique phishing reports and 30999 unique phishing websites were noticed in July 2007.

**1.2.1 The Phishing Process**

Initially the phisher prepares for the attack through some servers or using malicious software. Then a malicious code reaches the user through some e-mail or
fraudulent site and other different ways the phisher wants to use. Depending on the method the phisher uses to attack, the user takes an action and is compromised to leak out his/her information. The user then gives the confidential information through this forced method. The user compromises confidential information such as a credential, by providing it to a malicious server or software. This information is transmitted to the phisher through certain means of service. This confidential information is used to impersonate the user. The illegal party obtains monetary gain or participates in a fraudulent activity. This entire process is complex for an apprentice in this field but is not a complicated matter for a well experienced phisher. The growth in the phishing technology has also implemented with various ways of attacking. As an example, besides email, phishers have started using instant messengers to persuade users to spoof websites.
Figure 1.3. Phishing Process

The figure 1.3 shows the phishing process. In step 1 the phisher sends a fraud email to the user, and in response to that the user is compromised to provide the confidential information to the phisher in step 2. In step 3 phisher submits the confidential information of the user to the respective organization. The phisher obtains the financial gain from the organization in step 4.

The newest type of phishing scam is one that focuses on a single user or a department within an organization. The phish appears as if it has been legitimately addressed from an email within that company, and request information such as login IDs and passwords. Typical phishing scams often appear to be from a company’s own human resources or technical support divisions and may ask employees to update their username and password. Once hackers get this data they enter into secured networks. Another type of phishing attack prompts to the users to click on a link,
which deploys malicious programs that can steal data. Key logger is an example for such kind of malicious program. Key logger stores all the password strokes entered in the authentication fields after its installation. Another program communicates these stored key strokes to the phisher.

“In the case of the user, data suggest that some phishing attacks have convinced up to 5% of their recipients to provide sensitive information to spoofed Websites” [Loftesness 2004].

“About two million users gave information to spoofed Websites resulting in direct losses of $1.2 billion for U.S. banks and card issuers in 2003” [Litan 2004].

1.2.2 Cases of Phishing Attacks

Phishing was first reported when America Online users were lured in the mid 1990’s by phishers with their user names and passwords [Gerber 2006]. This was the first time when people came to know about phishing, and research has been initiated to avoid phishing.

A recent attack against the eBay customers was reported to the Anti-phishing Working group on March 9, 2004 which originated in South Korea. A phisher sent an email to an eBay customer as if sent from eBay asking her to provide her personal information because the eBay database lost her information. The email asked personal information like name, date of birth, social security number, credit card number and details. It was later detected from the IP address that the email was sent from South Korea [Mitchell 2004].

Another example is a phishing of a smaller regional (Chicago) bank. The goal of the phisher was similar to the first one to get credit card information. The subject
of the email was 'IMPORTANT: Account Verification '. This was spoofing email through which the phisher have taken the help of a fraud URL link [Ollmon 2004].

According to a study by Gartner, 57 million US Internet users have identified the receipt of e-mail linked to phishing scams and about 2 million of them are estimated to have been tricked into giving away sensitive information. The indirect losses are much higher than the direct losses to the US banks and credit card issuers because of phishing in 2003 are almost $1.2 billion [Engin 2005].

The US Military Academy at West Point found that more than 80% of its cadets were victims of a phishing attack by a fictional colonel. The State of New York mounted two attacks on its 10,000 employees; 15% were spoofed by the first attack, but only 8% by the second, which came three months later. There are other anti-phishing solutions that help users to differentiate the legitimate websites from the phishing ones [Bank, 2005].

In late 2005, another email claimed that recipients were eligible to receive a tax refund for $571.94. The email appeared to be sent from tax-returns@irs.gov with the subject line of "IRS Tax Refund." A link was provided in the email to access a form that had to be completed in order to receive the refund. The link appears to connect to the IRS website, but actually redirects the recipient to an entirely different website where personal data, including credit card information, is captured [Symantec Solutions].
1.3 Types of Phishing

There are several different techniques of Phishing which can be categorized into three different attacks. These three different ways of attacking vary based on attack types. The distinctions between attack types are porous, as many physical attacks use different technologies.

1.3.1 Malware-Based Phishing

Malware based phishing is a scheme which works by running malicious software on the user’s machine. Social-engineering activities or exploiting security vulnerability of the user may cause this to take place. As mentioned earlier, Key logger is an example for this kind of phishing. In this case, the programs install themselves either into a web browser or as device driver, which monitor data being input and send relevant data to a phishing server.

Key loggers work based on technologies such as the browser helper object, a device driver and a screen logger that monitors both the user’s input and the on screen input security measures. These Key Loggers handle many sites and track credential data from credit reporting agencies which is an example for Malware Based Phishing [Emigh 2005].

The latest version of these key loggers is the Advanced Keylogger, a powerful invisible surveillance tool. This tool records every keystroke to encrypted user-friendly easy-to-understand logs. Record all keystrokes typed, example, chat conversations, emails, desktop and Internet activity, clipboard information and more with the most powerful Key logger on the net.
Figure 1.4. Advanced Key Logger

The figure 1.4 shows the advanced key logger software. In case of a phishing activity the user has no idea of the installation on this kind of tools unless verified on a periodical routine.

1.3.2 DNS-Based Phishing

Any type of phishing that deals with DNS based activities, host file poisoning or tracking IP addresses, are considered a DNS-based phishing. It is also possible to pollute the user’s DNS cache with misleading information which launches the user’s browser into a phishing site. This sort of activity is also considered as DNS-based phishing. Modifying a user’s DNS server’s information and installing a Web proxy of the user’s Internet traffic are examples of DNS-Based phishing techniques. These are considered to be system reconfiguration techniques.

This kind of phishing is categorized into two different techniques. Directly inserting malicious code on to a server is called content-injection phishing which is a
form of DNS-based phishing. Replacing legitimate information on the server with malicious data through security vulnerability, cross-site scripting vulnerability, and through SQL injection vulnerability is considered as in-content-injection phishing [Emigh 2005].

1.3.3 Deceptive Attacks

The most common method of deceptive phishing today is email. This action uses a trick by sending an email to the user saying things such as there is a problem with the recipient’s account at a financial institution or other business. The email asks the recipient to visit a Web site to correct the problem, using a deceptive link in the email, the recipient’s account is at risk, a false invoice for merchandise, an fraudulent notice of an undesirable change made to the user’s account etc. In every case the user’s confidential information is tracked. In most cases the phisher uses the information in a secondary market.

This kind of phishing is to be the easiest way to perform a fraudulent activity for the phishers. A user with enough knowledge on Internet could sustain such threats but a normal user with less knowledge of Internet activity has more chances of risk for this kind of phishing. Although it may not solve the problems, to avoid this kind illegal activities financial and business oriented organizations have started introducing a security validation in the messages sent to the users.

The figure 1.4 shows is an email sent to a Bank of America customer by a phisher, which is an example of a typical deceptive attack.
Figure 1.5. A Typical Deceptive Attack

One among many variations of deception-based phishing schemes is with HTML readers, the possibility of providing a login page to redirect them to a different web browser by prompting the user to do so. In a few cases, IP address is used as a web address. In this case a cousin domain attack avoids the need for the complexity by using a domain name controlled by a phisher that is deceptively similar to a legitimate domain name, such as www.banofamerica.com instead of www.bankofamerica.com. This example has a fake website with missing “K” in the word “Bank”.

The users have less prospective to notice such kinds of URLs on the web pages and provide compromised information to the phishers. Though the security prompt “https” in the URL can help the users if noticed, most of the users lack knowledge that a secure webpage has “https” and a non-secure page has “http”.

12
1.4 Why Phishing Works

1.4.1 User Side Limitations

“Phishing is a semantic attack. “The third wave of network attacks is semantic attacks: attacks that target the way we, as humans, assign meaning to content.” Successful phishing depends on a discrepancy between the way a user perceives a communication, such as an email message or a Web page, and the actual effect of the communication” [Wu 2005].

There is a major impact on the phishing process based on the user’s mental model. This mental model is how the user behaves with the attack he/she is facing unknowingly.

These are the limitations from the user side in a phishing attack.

(a) Lack of Knowledge: Most of the users lack of the underlying knowledge of how the computer applications, email and web work. Phishers take advantage of this situation in several ways. For example, most of the users have no idea how the syntax of domain names and IP addresses work and hence the phishers use illegitimate domain names to hack the users. Lack of knowledge in security and security indicators such as SSL certificates and padlock icons is also a problem.

(b) Visual Deception: Visual deceptive tricks are used by the phishers to mimic legitimate text, images and windows. Sometimes phishers place unnoticed text in the domain names known as “typejacking” which is method of using visually deceptive text. Another common technique used by the phishers is to use an image of a legitimate hyperlink and images in the Website.

(c) Bounded Attention: Security is the second goal. Lack of attention towards security indicators forces the user to be tricked by the phishers. Users do not notice
even if there exist a security toolbar or not and if exists, do not care to know if it is legitimate.

1.4.2 System Side Limitations

To design a secure system to avoid phishing is a hard task. These are the following security limitations that come into play while designing a phishing based secure system.

(a) Limited Human skills: The limited human skills property notes about the limited inherent skills of a human compared to a computer. The designer who implements the tool to avoid phishing will not have a better estimate on the knowledge of the user about security.

(b) General purpose graphics: The general purpose graphics option in the operating systems and a windows platform covers the limitations provided by an operating system. While building a system that is designed to resist spoofing the designer must assume that uniform graphic designs can be easily copied.

(c) Golden arches: As the name suggests the golden arches property informs about the logos and security indicators on the spoofing websites which cheat a user. Logos and security indicators such as padlock icons and SSL certificates are not enough to secure user’s information.

(d) Barn Door: The barn door property is the last property informs that even a small leakage of information could cause a massive disaster with respect to the user. The systems are not enough secure to protect the passwords of the users.
Figure 1.6. Basic Relation between User’s Mental Model and System Model [Wu 2005]

The figure 1.5 shows the user’s mental model, against the system model. The figure points the factors to be considered by the user when a web page is opened on a browser through an email, against to the server side factors of the webpage.

Though the causes for turning out to be a victim by a phishing attack may not be limited to above list, the most common factors would be the above.
1.5. Previous Work

Research on phishing had seen the development of numerous toolbars to avoid phishing. The following are some of them.

1.5.1. Spoof Guard

Spoof Guard, a toolbar to detect phishing webpages, was initially developed by a group of Students and professors from Stanford University. The plug-in works based on the history file of the explorer and have three more files to store certain features of the webpages. It detects phishing webpages based on certain checks like, Domain name, URL, Link, Images on the webpage, Password fields, password used by the user on that webpage and couple more features. Based on these checks the plug-in decides if the opened webpage is a spoof or not.

1.5.2. PwdHash

PwdHash is a plug-in developed by the developers of Spoof Guard. This is used for the purpose of securing passwords and avoiding password phishing and other attacks. In order improve the website security; this browser extension generates different passwords for different site, though the used uses same password for two different websites. This hashed password is transmitted to the server where it is decrypted. This is makes the password to be site specific and reduces the number of chances for password theft through phishing.

1.5.3. Spoof Stick

Spoof Stick was released to the world of Internet by Core Street. This plug-in uses the domain name of the webpage to detect if that particular webpage is a spoof or not. The actual domain of any particular webpage opened by the user is shown to the user.
The user can verify the domain name shown by the plug-in with the user-required domain and detect if the opened webpage is a spoof or not.

1.5.4. Netcraft Toolbar

Netcraft LTD released Netcraft Toolbar to the Internet. This plug-in works on the basis of domain name and it’s hosting country. As the user opens the webpage, the plug-in shows the domain registration date and the country of its registration. This provides the user with the information about domain.

1.5.5. Trustbar

Trustbar is a toolbar developed by a professor in Bar-Ilan University, Israel. Trustbar highlights secure web connections (SSL) by displaying the logos of the website and its certificate authority. This is useful because most phishing sites do not use SSL to encrypt the user’s sensitive data transmission.

1.5.6. EBay Account Guard

This is plug-in was developed by eBay specifically for purpose of detecting webpages that look similar to eBay’s webpage. The tool works to detect pages that look similar to eBay and Paypal, based on the images on the webpages.

1.5.7. Web Wallet

This plug-in was developed by a group of professors in Massachusetts Institute of Technology. This is a browser sidebar for entering sensitive information. The plug-in checks if the current site is good enough to receive sensitive data or not. If the current site is not qualified, the Web Wallet requires the user to explicitly indicate where the data needs to go. If the user’s intended site is not the current, the plug-in shows a warning to the user, and provides a safe path to the requested site.
2. NARRATIVE

2.1 Spoof Guard

Web spoofing is a method of revealing private information through fraudulent emails and websites. Most of these web spoofing are not detected. A new approach for detection of spoof is made by using a browser plug-in called Spoof Guard. The complete users Internet activity is monitored by this browser plug-in, based on the certain threshold level, provided by user, it warns the user if a spoof page is detected. This threshold level is named as Spoof Index, a measure that a particular page is susceptible to a spoof attack.

In order to detect if a particular page is a part of a spoof attack or not, Spoof Guard uses the factors such as the domain name, URL, images, and links on the page. This plug-in works on the history in its cache, and checks its history each time a new URL is opened, if the domain was previously opened.

The user posted data and the previous data, from different domains, in the cache is compared by Spoof Guard, and prompts the user if a password misuse takes place. The password is sent in an encrypted form and thus providing security to the data.

2.1.1. Problems

The main factor for the web spoofing attacks is that many users use the same Username and Password at several sites. The attackers use the site which doesn’t provide adequate security for the user’s private information and gain the username and password. These passwords stolen from one site can be used at other sites, as the possibility for a user to use same passwords in different web pages is high. This is achieved by attracting the victim to the site which is identical to the original site with matching logos and using evading URL. The attacker duplicates the original site, copying the actual HTML from
the original site and making unnoticeable changes to it. This makes the user to be trapped and provide confidential data.

![Image of Spoof Guard prompt]

**Figure 2.1 Prompt Shown by Spoof Guard that the Yahoo Site Opened by the Browser is a Spoof Website**

The figure 2.1 shows a Spoof Guard prompt when a spoof page is detected.

### 2.1.2 Solutions

The solution to this kind of web spoofing is achieved by conducting several tests to distinguish the original page from spoofing pages, using Spoof Guard. An incrementor is used as spoof index. This is conveyed to the user by using passive toolbar indicator
instead of the pop-ups. The user is given the flexibility to uncheck this feature to avoid all the pop-ups.

The other important factor is misleading URLs. In this process, the URL provided to the user in the email is a mask of an actual spoof URL. URL can be used to mislead the user (victim). The spoof site created by the phisher looks similar to that of the actual website. The phisher creates the website using the same images that are present in the actual website. For example the “@” in the URL causes the string on left to be neglected and the string to the right treated as the actual URL. The URL check makes if the page is spoof susceptible to an attack. The storing or modifying the actual image present in the spoof site is prevented by storing the image of the honest site, as an image hash, in its cache. The optical character recognition (OCR) algorithm is used as image hashing algorithm (i.e., detecting duplicate images on the spoof website) for this purpose, thus preventing the phisher to cheat the user by using similar images of the original site. The attacker may use similar links on the spoof site. Link checks in this algorithm fail if at least one-fourth of the URL fails to be same.

The pages requesting for passwords have more scrutiny than non password pages. If a page requests for password then its HTML certificate is checked. The browser history of the page and the history stored by Spoof Guard are checked to reduce the number of false alarms. Spoof Guards do not issue any warning for visiting the sites which have an entry in the user’s history. This is done considering that the user has sufficient reason for visiting the site.

Outgoing password check is performed by the Spoof Guard maintaining the database of (domain, username, and password). If the same password is being used by the
user in other domain, then password check is performed. The possibility of leaking the private information is reduced using Secure Hash Algorithm-1 and the comparisons are performed on hashed values. The outgoing data in an HTML is hashed and checked with that of database termed sensitive. Thus Spoof Guard prevents the password leaking.

2.1.3. Limitations

There exist a couple of limitations due to which the attacker may attack the websites. One of them would be the attacker could fool the above test by breaking the password input field into multiple adjacent fields. The other one would be by slicing an image into adjacent vertical slices and presenting them in order. This would not allow Spoof Guard to detect spoof websites based on frame sets. To improve the working of the Spoof Guard, the framework used in it can be extended with better password management system to overcome these limitations.

2.1.4. Advantages

In the earlier version of Spoof Guard the phisher can attack by breaking the password field into two or more fields. This limitation is avoided in this version of Spoof Guard. A new function is created which looks into the number of password fields in sequence and concatenate (place in a sequence) all the password characters in those fields into one single string and then compares the hashed string with the hashed value in the cache.

Frames in webpages have always become a problem in server side and user side. The earlier version of Spoof Guard considers frames as individual pages instead of splitting the frame set to determine its frames. A webpage having frames on different domains is considered as possibly malicious. In the new version, the plug-in counts the
number of frames on the page compares with the existing page in the cache. The initial checks on URL, Domain and image are carried out and then it compares with frame count. Though the absence of this feature in the earlier version does not cause much difference in detecting the spoof pages, it may help in resolving the frames in a spoof page.

Many users may find spoof pages on a single day, and may enter personal information and then recognize their mistake. There must be an easy way for them to report this to authorities. A feature added to the new version is, whenever a spoof page is detected, the plug-in prompts the user if they want to report it to the respected authorities. When the user accepts and says “Yes”, a Pop3 tool (Outlook) prompts, with the default To-Address and an inserted message about the spoof page in the main section.

More detail information about these features is explained in Section 3.

2.2 Architecture

The architecture of the new version of Spoof Guard is similar to the earlier version but has effective changes to overcome the major limitation and adds couple more extra features which are more user-friendly.

Spoof Guard is a Browser Helper Object, a COM component, and loads when Internet Explorer starts up. It works on the same memory context as the explorer. The history file and three additional files in the explorer are used by the Spoof Guard. The read-only file containing the domains of the email sites such as Hotmail, the hashed password history file and the image history file are the three files used by the plug-in.

IDeskband is the extension of the COM component that exists in Spoof Guard. It is an interface that causes explorer to load Spoof Guard as a standard toolbar.
CWindowImpl is the interface implemented by two of the window classes, which define
the look and user interaction of the toolbar.

2.2.1. Spoof Guard Classes:

The following are the main modules of the toolbar:

- **WarnBar**: This is the key COM component that addresses Spoof Guard. This
extends the IDeskband interface to other required components and also
all the site evaluations and post data checks are conceded out here. The
initialization of the Warnbar creates and registers the new instance of
ReflectionWnd, and consequently a new instance of UWToolbar is created.

- **ReflectionWnd**: A transparent window is implemented by
ReflectionWnd, which sets itself on the peak of the toolbar and transmits
user messages such as inputs to UWToolbar. A new UWToolbar is created
by this component once it is created by Warnbar, and acts as an interface
between Warnbar and UWToolbar for executing user inputs. WarnBar
requests pop-up messages to UWToolbar through ReflectionWnd. It
registers its instance of UWToolbar in a message map constructed with
macros of Microsoft ATL, and hence passes its messages through this as
below.

```
BEGIN_MSG_MAP(CReflectionWnd)
MESSAGE_HANDLER(WM_CREATE, OnCreate);
CHAIN_MSG_MAP_MEMBER(m_ToolBarWnd);
END_MSG_MAP()
```

- **UWToolbar**: This is a CWindowImpl class that defines the buttons and
appearance of the toolbar. The messages from the ReflectionWnd are
received through the above command handlers. The first handler is initialized as a standard method handler. The commands that are to be passed through ReflectionWnd are handled by the second handler. Nothing else handled here.

The user settings entered by the user at the runtime are also stored in this class. These settings include options such as threshold level, sensitivity etc, each represented by a variable. WarnBar needs to access this data in UWToolbar each time as the options are dynamic to runtime based on the user settings, as UWToolbar does not interact with Warnbar to pop-up the settings window to the user. UWToolbar also transmits methods to create the current status of the page, as other classes cannot change the icon and domain name on the Status Button and also the message that appears on the Status Dialog.

The initialization of the UWToolbar creates and registers two buttons that trigger the Settings and Domain Dialogs, by firing the IDM_OPTIONS and IDM_STATUSBUTTON events respectively. As a user clicks on any of these buttons, the ReflectionWnd retrieves the message and passes to UWToolBar, which fires OnCommand. OnCommand interprets the command and, if the event is an IDM_OPTIONS or an IDM_STATUSBUTTON, the appropriate dialog is initialized and fired.

If the Settings Dialog is fired, a new ConfigDlg is initialized with the current state of the user settings, and the UWToolbar updates the state based on the result that ConfigDlg returns when the dialog terminates.
• **ConfigDlg:** This class turns-on once the user clicks on the “Options” button. The Settings Dialog is handled by this class by the TestDlg. The TestDlg retains the instance variable values such as the threshold level, once the user enters them. It also intercepts all its fired events, until all the changes are made. The state of the Status Dialog is updated after the changes are made. After the changes are made and saved the UWToolbar extracts the instance variable values form the ConfigDlg and saves them.

• **DomainDlg:** This class works for showing up the status of the current domain. It is associated with the Status Dialog. DomainDlg acts similar to ConfigDlg but is simpler than it. It contains the messages allied with the current page, and works on initialization.

Based on the output of the other classes Warnbar sets the status of the current page after retrieving the current state of the user settings. On the initialization or termination of the toolbar, the user settings are exported into LoadRegVals() and saved in SaveRegVals().
The Figure 2.2 shows the architecture of the Spoof Guard. It can be noticed that the UWToolbar acts as an interface between most of the classes.

2.2.2. Browser Events:

There exist other browser events which interface and control the tool bar operation integrated with the explorer. The following are the browser events:

- **IObjectWithSite**: The Warnbar is initialized by the SetSite method in the IObjectWithSite after the Internet explorer is initiated. This method is called by the containing browser at the beginning of its execution and passes it a pointer to the containing object, the web browser. SetSite is also called immediately before the web browser terminates WarnBar, with a null value passed in. In this case,
this is used as a destructor, which deletes the history list and un-registers the UWToolBar.

- **DWebBrowserEvents2**: The BeforeNavigate2 and DocumentComplete are the two classes implemented in the Warnbar. DWebBrowserEvents2 class in the explorer event handler calls these two classes when Warnbar is initialized.

- **BeforeNavigate2**: Before navigating the actual page, the BeforeNavigate2 event occurs in the given window or frameset element. This also gives WarnBar the URL that the browser is attempting to navigate to and allows WarnBar to cancel the browser’s navigation.

  WarnBar also checks redirects, which can be a sort of trick, that are not in the browser’s history. The first round of checks by Warnbar is performed here. Domain Check, URL Check and Email Check are the three checks performed at this class. The Domain Check iterates through the history list initialized in SetSize and compares the current URL to each of them, using the Edit Distance as a metric. A series of static checks on the structure that represents the parsed URL is performed at the URL Check.

- **DocumentComplete**: The DocumentComplete event fires after the complete page is loaded. This is very similar to BeforeNavigate2, but navigation cannot be cancelled. If the page already exists in the browser history, the hash values of the URLs, images, and passwords are not expected. In the other case, the second round checks by WarnBar are performed here. These include the Image Check, Link Check and Password Check.
2.3. User Interface

Spoof Guard is a toolbar which is easy to install. Once installed, the browser must be restarted and the “Spoof Guard” option on the toolbar list must be enabled, which initiates the toolbar. Spoof Guard can be turned-off by disabling it in the toolbar list.

2.3.1 Status Button

The tool-bar has three buttons on the explorer. The “Status Button” prompts the status of the current page, if the current page is spoof or not. This button contains a traffic light with either Green, Red or Yellow colors. Green does indicate that Spoof Guard has detected the current page as legitimate, if Red as spoof page, and if Yellow as a chance to be a spoof page. The second is the “Settings Button”, which is the front end interface for the user to provide the desired level of threshold and other provide options. The third is the “Reset Button” which deletes all the previous hash values and resets the tool bar, when pressed.

The traffic light provides an indication to the user of the existing page where the options button is used to configure the tool. Depending on the spoof rating if the threshold is above the user specified limit, additional warning windows will pop-up by the Spoof Guard. The analysis of the Spoof Guard is conveyed to the user in two ways. The traffic light symbol on the browser bar is used to display the degree of spoof. The actual colors are used to display the threshold setting set by the user. By clicking on the traffic light the user can get more information. The traffic light and pop-up provide an effective alerting means.
Figure 2.3. Spoof Guard User Interface
Figure 2.3 shows the user interface of the spoof guard tool. The tool bar buttons, Status, Options and Reset can be noticed in the above figure. The Green signal indicates that the page www.tamucc.edu is legitimate.

In order to detect if the user has accessed the email, the phisher sends a couple of emails which contain spoof sites. This Spoof Guard makes an action in order to know if the user has access to the website. The host names of the email providers will be helpful in this case. The other way is to see if the referrer field is empty. As different sites use different field names for the user passwords, Spoof Guards are predefined with it. But there comes a problem in the case of other sites which use other input field names.

Spoof Guard scans for sensitive information after comparing outgoing passwords with its database and stores a resulting hash. This is called post data checking which is more advantageous.

When the browser navigates to a new page, the plug-in performs eight checks in two rounds. This can be adjusted in the Options tab. The weight on each check and the threshold can be set to the desired level. Each check returns a boolean result. If the sum of the activated checks is greater than or equal to the threshold level, a flag is triggered. This flag would make the red light to appear on the ‘Status’ button. At the point where the browser navigates to a new web page, the plug-in has only the URL of the webpage available. The first round of checks is performed here. The Domain Name check and URL check are the two checks performed in the first round. If the result is negative from these two checks, a flag is triggered saying the page is a possible spoof page and a prompt is shown to the user before navigation.
2.3.2 Settings Button

![Figure 2.4. Options Tab](image)

**Options - SpoofGuard**

- **Total Alert Level**: Sets the sum of individual alerts of the below activated checks before flagging a Web site.
- **Domain Name Check Level**: Compares the current domain name with the previously visited domain names.
- **Domain Name Check Sensitivity**: Edit Distance tolerance—the number of changes required to change one domain name into another.
- **Maximum Delete Name History Entries**: Stores host names in a list in order to perform the Domain Name Check.
- **URL Check Level**: Checks the URL for any possible spoofing content.
- **Links Check Level**: Examines the links on the current web page using the URL check.
- **Links Check Sensitivity**: Minimum level for the links on the current web page to be considered a suspicious link.
- **Image Check Level**: Compares images on the current page with stored images of previously visited pages.
- **Password Fields Check Level**: Checks the current web page for password input fields.
- **Unencrypted Password Check Level**: Checks the current web page for any unencrypted password input fields.
- **Email Referral Check Weight**: Checks for pages that you may have been referred to by an email link.
- **Frame Referral Check**: Compares the Frames on Current Page with the Frames on the stored web pages.

- **Enable Confirmation Warning**: Requests confirmation before navigating to a page that triggers the "Total Alert Level" threshold.
- **Save Image Hashes**: Allows maintaining a list of image hashes with which SpoofGuard performs the Image Check.
- **Enable Password Protection**: Keeps track of your passwords and displays a warning when you try to input the same username and password on different sites.
- **Send Mail**: Send E-Mail to respected websites if spoof site is found.

---

**Figure 2.4. Options Tab**
The Figure 2.4 shows the options tab where the user can enter the desired levels of threshold and other checks performed. The tool also has the check boxes to provide user convenience in popping-up the messages. These values are stored in the instant variables, which are then exported to the Warnbar from the ConfigDlg, through ReflectionWnd.

**Total Alert Level:**

This is the threshold level on which it is decided if the web page is spoof or not. The plug-in verifies if the particular page is a spoof or not based alert levels of different checks made. The sum of all the different alert levels is calculated as the Total Threshold Level. If the value of the desired total threshold level is greater than or equal to the sum of the activated checks, the site is considered to be a spoofed page and a “Red Light” is prompted on the status bar.

**Domain Name Check:**

The domain name of the current URL is validated against the domains in the most recent browser history entries. If the current domain is the same as the one in the cache, it receives a green signal to move forward. If the domain is somewhat similar to the any of them in the history file, the Domain Name check is activated. This prompts an attention if misleading links such as [www.banofamerica.com](http://www.banofamerica.com) comes up which is lot similar to [www.bankofamerica.com](http://www.bankofamerica.com) (notice the missing ‘k’). The factor calculating the difference between the two is the Edit Distance between the two domains. The Edit Distance is the number of characters that need to be inserted or deleted in order to transform one domain into another. This factor can be adjusted in the user settings; describing the required distance between two similar pages. The maximum number of entries of the domain names can also be selected to a desired number.
Maximum Domain Names:

This is the maximum number of domain names that are stored in the cache.

URL Check:

Leaving the Domain Name Check to check domains, the rest of the URL will be checked by the URL check. It performs three different tests on URL, and activates itself if any of the test results to failure. The first check is on finding if any suspicious usernames are inserted in the URL, such as ‘www’, ‘.com’, ‘.org’ or any similar terms in the URL itself, on the left to ‘@’. The second test would be checking if any masked or odd content such as [www.bankofamerica.com@1000.1000.100](http://www.bankofamerica.com@1000.1000.100) is present in the URL. The third and final check will be on the port number. It shows up if the port is not standard. Standard ports are such as ‘http’, ‘https’ etc.

Link Check:

Many webpages have numerous links on them. The Link Check searches the current web page body of the document for suspicious links. A suspicious link is a link that references a suspicious URL. A suspicious URL is one such that the sum of the weights of the activated first round checks, both of which work on only a URL, is greater than the Link Sensitivity. The tool-bar prompts up a message when an attempt to open such kind of webpages.

Image Check:

This is an optional feature, which works when activated. The Image Check compares images on the current web page against the images in the webpages in the cache. It retrieves every image from the current webpage, hashes them and compares
against the hashed image values in the history. This task is performed only if the user wants it to be done. That is un-checking the check box ‘save image hashes’ will disable this feature.

Once it ends up the checks, it computes the sum of the weights of all the activated checks from both the first and second round, and if greater than the Total threshold Level, a prompt will be shown up on the screen. This prompt does not show up if the option to show pop-up warnings is un-checked. Also on the other way the user can navigate to the page by saying “Yes’ on the prompt.

**Password Check:**

This check investigates the parsed body of the webpage, if any input fields exist on the web page, which prompts for passwords. If it finds any, it gets activated. It does not conclude the page to be spoof, if it gets activated, but reduces the threshold level for that particular page. This reduction would increase the chance for the user to notice more on the kind of webpages accepting passwords, as this is way to leak compromised information. It is the same case for password fields on unencrypted frames. Unencrypted Password Check triggers a flag if there is a password field on an unencrypted frame.

**Email Check:**

Email check will perform a check if the browser has been directed to the attempted URL from an email. Initially it checks for an empty referral field or one which indicates a referral from a web-based email site such as mail.yahoo.com or hotmail.com

If the sum of the weights of the activated options is greater than or equal to the Total Alert Level, a prompt if to navigate to that particular site is shown on the screen. This is based on the first three flags.
If the user asks to navigate to that particular page, a second round of checks is performed on that complete document.

**Frame Referral Check:**

This is a new feature implemented to compare the frames. The Frame Referral check compares the frames of the current page with the frames of the previous visited pages. When a web document is opened on the explorer, the plug-in searches for frames and stores the length, width and border color of the frame. This measurement is compared with that of the measurements of the previously visited frames. If the values are same as any visited page, the Frame referral check gets activated.

**Prompt For Confirmation:**

Populating confirmation prompts frequently may irritate the user. For this reason, the confirmation warnings pop-up only when the user enables ‘Prompt For Confirmation’.

**Store Image Hashes:**

This feature when enabled hashes the images on the webpage and saves into the cache.

**Password Tracking:**

The Password Tracking feature will interrupt the user before typing the same username and password into more than one site. This is done by storing the hash of a user's username and password in the cache and retrieves whenever a password check is to be made. If the phisher tries to create multiple password fields on the page and mask the
up to show as one, the newly implemented password tracking feature will sum up the password to a single string and compares it with the passwords in the cache.

**Send Email:**

This feature when turned on prompts every single, the Password Tracking gets activated. If a same password is used for different sites, or a spoof page with password fields is detected, then a prompt is populated asking if the user wants to send an email to the actual web page, that the current web page is spoof. The prompt does not show up when disabled.
3. SYSTEM DESIGN

3.1. System Requirements

The toolbar was developed on a Windows System. The tool used to develop was Microsoft Visual Studio 6.0, and was coded in Visual C++ platform. The flexibility of this platform had been used to acquire the different requirements of Spoof Guard.

Visual Studio 6.0 provides an easy way to develop applications, and it is a productive language tool for rapidly building applications for Microsoft Windows. Ideal environment for existing Visual C++ developers as well as new developers in the Microsoft .NET development environment, Visual Studio delivers enhanced visual designers, increased application performance, and a powerful integrated development environment (IDE) to get one on the fast track to application development.

3.2. Passwords Check

The web pages which have password fields have more scrutiny than the pages which do not have password fields. The initial version of the plug-in stores the domain names, user names, and passwords of webpages in the format (domain, user, pwd). In this case when a user enters data in the password fields, the plug-in hashes the data and compares it with password hashes of the previously visited web pages, stored in the cache. This works well for webpages having one password field.

The phishers, having the working knowledge of Spoof Guard, can overcome this password check by breaking the password field into multiple password fields on the web page, placing them in sequence and masking them into one password field. On the user side the webpage looks such as having one password field, but in the backend the plug-in considers the data in each password field as a different password for the same user name.
and fails to recognize the complete password. This makes the password check to fail as it compares the parts of password and does not show up a confirmation warning, even though the web page is a spoof. This is one of the major disadvantages of Spoof Guard.

A new function has been developed to overcome this limitation. The function initially counts the number of password fields on the page for one particular user name, then collects the posted test in each of the password fields in the same order the passwords are arranged on the webpage, and concatenates the data into one string which is considered as the password. The plug-in then hashes this string, considered as password, and compares the hash values of the passwords in the history. If any matches are found, it comes up with two cases. If any matches are found it activates the password Check flag. Based on this a confirmation warning is popped-up informing that either the web page is spoof page or same password is used for a different webpage.

For checking multiple passwords:

```cpp
list<CString> getPasswordsList(const CString& token, const CString ids[], int len);
```

( if multiple passwords exist then get all passwords in list )
Figure 3.1. Password Check

Figure 3.1 shows the architecture of how the password fields check works. The Warnbar class checks the number of password fields and uses the user settings from ReflectionWnd, if to prompt the confirmation warning and navigates to UWToolBar.

The above are the headers of the functions which verify the existence of multiple passwords and grab the data posted in the password fields. The function initially verifies for the number of password fields in the webpage for one particular user name. If multiple fields exist, the passwords posted in the fields are concatenated into the password string. And if single password field exists, the plug-in directly collects the
posted string, hashes it and compares with the hashed passwords in the cache. This string is then hashed to compare with the stored hashed passwords.

This overcomes one of the major limitations of initial version of spoof guard.

3.3. Frame Check

Frames in webpages have always become a problem in server side and user side. Frames in web pages can be divisions such as flash images, dynamic changes on the web pages etc. The earlier version of Spoof Guard considers frames as individual pages instead of splitting the frame set to determine its frames. A webpage having frames on different domains is considered as possibly malicious.

In the new version, the Spoof Guard counts the number of frames on the page, collects the length, width and border coloring and hashes the values into the format (len, wid, col) for each URL. When a new web document is opened by the explorer, this Frame check verifies the existence of Frames. If any frames exist, it initially checks the number of frames, hashes the values of length, width and border color, and then compares with the hashed values in the cache. It gets activated only if the alert level is greater than the user desired level in the options.
Figure 3.2. Webpage with a frame in it.

Figure 3.2 shows a webpage having a frame in it. The frame contains a flash image (a moving bicycle).

The following is the header of the function used to collect the number of frames and their dimensions:

```c++
int CWarnBar::FrameCheck(IHTMLDocument2 *pHTMLDoc, VARIANT* url)
{
    //get the collection of Frames
    HRESULT hr = pHTMLDoc->get_frames(&pHtmlElements);

    //get length of collection (answer)
    hr = pHtmlElements->get_length(&nLength);
}
```

If the frame check gets activated, a message is sent to the UWToolBar using ReflectionWnd as an interface asking to pop-up a confirmation warning.
IHTMLFramesCollection2 *pHtmlElements gets the collection of frames, and pHnlElements->get_length(&nLength); calculates the dimensions.

The frame check would be the last check performed on the web page. Though the absence of this feature in the earlier version does not cause much difference in detecting the spoof pages, it may help in resolving the frames in a spoof page.

3.4. Email Configuration

According a study by Garner, every year on an average of 1000 people get phishing emails and more than 30% of them get tricked and turn out to be victims of phishing. [Garner]

The victims may realize sooner or later about this and may want to let the right authorities know about this. For this, Spoof Guard is provided with a feature where a user can directly send an email to the authorities concerned with the legitimate website. Each time the user enters the username and password to an unknown page, Spoof Guard prompts the user if the user wants to send an email to the authorities of the legitimate website. If the user likes to do so and clicks “Yes”, a Pop-3 window, an Outlook Window, is popped-up. The To-address is generated as a default from the tool and the message box contains the message providing the information of the URL of the spoof page. If enough number of users identifies the page as a spoof the server could alert all plug-ins to block the page. This might consequently reduce the number of users who get tricked by the spoof site.

The user doesn’t need to be a victim of phishing, but any user who recognizes a spoof page can use this feature. If the user is not interested to send the email, user can click “No”, when the plug-in prompts.
Figure 3.3. Spoof Confirmation Warning asking the user if to send the spoof information

Figure 3.3 shows the prompt asking the user if to send an email to the website a fraud website exists with similar features.

To avoid interruption, this email configuration works only if the user checks the “Send Mail” check box in the options.

\[
\text{enableSendMail} = (\text{IsDlgButtonChecked(IDC_SENDMAIL)} == \text{BST_CHECKED});
\]

\[
\text{SendMessage(m_hWnd, TB_SETEXTENDEDSTYLE, 0, TBSTYLE_EX_MIXEDBUTTONS)};
\]

The enableSendMail function checks if the “Send Mail” option is checked. If enabled, the sub-routine to open the outlook window is called. The To-email id is then uploaded from the predefined declaration for that particular legitimate webpage. The Text is then loaded into the message box. The user also has option to delete the original text in the message box or add additional information to the existing text.

This feature may be speculative in couple of issues. It is quite complex to define the “To” email address for all the honest webpages which are most possible to get spoofed. As of now the plug-in is defined with sample defined email ids. The users can also figure out the email address where to the report the phishing site.
Secondly, Outlook must already have been installed and configured on the user machine. This feature does not work in the case where outlook is not configured.

Another issue is that this feature could potentially attack an honest website on the basis of denial of service attack against it. A user with enough knowledge on Spoof Guard can empty the cache and use wide variety of ways to potentially attack an honest website too. After being indicated the honest page must be in position to verify and take an action immediately, if the page detected is proved to be spoof. This could help in protecting other users from being tricked and can also give a chance to get hold of the phishers if possible.
4. EVALUATION & RESULTS

Security is a primary issue in a case of testing tools involved in Internet based fraud activities. To test Spoof Guard specific security issues were considered to avoid users from having confusion on the actual and fraud websites.

The .msi (installation) is run and the plug-in is installed. The Internet Explorer is restarted and under the View → Toolbars, the option Spoof Guard is enabled. This showed up the three buttons on the tool bar of the explorer.

As the Spoof guard works on cache, a website of two copies is required for testing each newly implemented feature. A set of websites are created to test Multiple Password Check, Frame Check and Email Configuration. The websites and the combination of images used are unique and have no similarity with any other websites online. Considering the security factor, each website has the message as, “This is Sample Spoof Testing Page. Please Do Not enter any information”. The websites are connected to Mysql database in the backend.

Two different servers are necessary for testing these features in order to locate each copy of website. The sub-domain www.spoofproject.freeservers.com in the domain www.freeservers.com is used considering at as a spoof server, and the www.sci.tamucc.edu sub-domain in the www.tamucc.edu domain is used as a second domain, considered as a legitimate site. Each set of different websites are placed on the two servers for testing.
4.1 Multiple Password Testing

4.1.1. Test

One of the websites created for testing Multiple Password Check has one ‘User Name’ field and one ‘Password’ field for posting user id and password. This website is treated to be legitimate according to the testing schema. This is placed on the www.sci.tamucc.edu domain and connected to the Mysql database. The second website used for the Multiple Password Testing appears same as the first website but has one ‘User Name’ field and three ‘Password’ fields. Though there exist three password fields, the three fields are placed near by and masked such that only one password field appears on the user interface. This website is placed on the www.spoofproject.freeservers.com domain and connected to the Mysql database. The password fields are designed in such a way that after every three posted characters in field, the cursor directly moves to the next field. This is to explain how the phishers mislead the users.
Figure 4.1. Spoof Page containing three password fields for one particular user-id.

The figure 4.1 shows the page placed in the local apache server, and assumed to be as a spoof page in the testing. The page contains three password fields which will be masked to show up as one field.

To test the feature, the webpage on the www.sci.tamucc.edu domain is opened on the browser initially and the user-name and password is entered which stores the posted text in its database. This makes the explorer store the domain name, images and posted text of the webpage in its cache. Then the webpage located on the www.spoofproject.freeservers.com domain is opened on the explorer. Assuming the user
had been mis-leded by the spoof page, the same user name and password entered in the similar website is posted here.

The goal of this testing is for the Spoof Guard to recognize the web page in the www.spoofproject.freeservers.com domain as a spoof page and prompt a confirmation warning asking the user if to proceed in opening the web page.

4.1.2. Results

As the web page on the www.spoofproject.freeservers.com domain is opened by the explorer a message was prompted by Spoof Guard saying that the page is a spoof and a similar webpage exists on the www.sci.tamucc.edu domain.

4.2 Frame Testing

4.2.1. Test

A set of two websites are created to test the frame check feature. The website contains a frame with a flash image inserted in it. The flash image used here is a moving image. A copy of one website is placed on the www.sci.tamucc.edu domain and another one on the www.spoofproject.freeservers.com domain. Even in this case of testing, the web page on the www.sci.tamucc.edu domain is assumed as a legitimate one and the one on the www.spoofproject.freeservers.com domain as a spoof page. To test the feature, initially the web document located on the first domain was opened. This makes the explorer to load all the features on the web page and frame in the web page. Next the web page on the www.spoofproject.freeservers.com domain is opened in the explorer.

The goal of the Frame testing is that the plug-in must detect the web page on the www.spoofproject.freeservers.com domain as a spoof web page as it has the hashed values of a similar web page in the cache.
4.2.2. Results

The plug-in worked fine with the Frames and detected the web document on the www.spoofproject.freeservers.com domain as a spoof. A message was prompted on the screen that a similar web page exists on the www.sci.tamu.cc.edu domain.

![Image](Figure 4.2. Spoof Web Page having a frame)

The figure 4.2 shows a web page with a frame, having a flash image. The image also shows the pop-up message generated by the plug-in asking the users if to navigate to the page or not.

4.3 Email Configuration Testing

4.3.1. Test

In order to test the Email configuration feature, any web page having mis-leading links or any spoof web page can be used. In this case, the same two websites used for
testing multiple passwords check feature can be used. Using the later case, the web page on the [www.sci.tamucc.edu](http://www.sci.tamucc.edu) domain is assumed as a legitimate one and one on the [www.spoofproject.freeservers.com](http://www.spoofproject.freeservers.com) domain as a spoof page. The testing is performed under two considerations; the first one is with the ‘Send Email’ option enabled in the options and having disabled is the second one. The first web page is opened having the option enabled. As the web page is already opened, it is considered as legitimate. The second web page on the [www.spoofproject.freeservers.com](http://www.spoofproject.freeservers.com) domain is then opened.

As a result of this testing, the spoof guard must show up a pop-up warning, on detecting a spoof page, asking if the user likes to send an email to the legitimate web page. As ‘Yes’ was clicked on the prompt, the pop-3 outlook was opened.

### 4.3.2. Results

As a result of finding a spoof page a message was prompted on the screen that the web page is a spoof and if the user wants email this to the actual website. An Outlook message was prompted after clicking ‘Yes’ with a default ‘To’ address and a default message in the message box. A second attempt was made on the test for clicking ‘No’ and the plug-in did nothing but just directed the same spoof page.
Figure 4.3. Outlook prompt after a spoof webpage is found.

Figure 4.3 shows an outlook box after a spoof page is found. The ‘To’ address is generated as default and can be changed. The text in the message box is also generated and can be edited as desired.

4.4 Tool Survey

Using the new version of Spoof Guard a brief survey was performed to know the user acceptance level of the tool. The survey had ten students from different majors. The students were asked to install Spoof Guard and use it for two days at their usual times of browsing. After two days the students came up with the answers to the survey questions.
The figure 4.4 shows the statistical representation of the survey performed. Eight out of ten students said they had never heard of Spoof Guard before. The students were asked to rate the features of the new version on the basis of 1 to 5 with 1 as the worst and 5 as the best. On an average the machine performance after installing Spoof Guard is detected to be more than 4 and the overall satisfaction of the students of the users was between 3.3.
5. CONCLUSION

Spoof Detection is still a growing platform for research. However, it is the commencement to suppose enormous significance in ongoing computing environment. The issues such as the growth of the Internet, the opening up of electronic trade for financial activities and the need for actual secure systems turns it out into an interesting area of research.

As email has become a favorite vehicle to transport messages all over the world, the increasing rate of phishing has taken an advantage of it. The reason for this is the lack of efficient security to the Internet based activities. Though a number of intrusion detection tools are invented by the forensic department, there is also dramatic increase on the negative aspects too. The most effective solution to avoid phishing is training users not to blindly follow links to web sites where they have to enter sensitive information such as passwords. It is highly impossible to assume that all users will understand the phishing threat and step ahead on the Internet cautiously. The number of users tricked by the phishing websites is never nil. Hence, it rests as a responsibility for researchers and industry to provide solutions for the phishing threats.

As the expected results are obtained, Spoof Guard can be considered to be a tool which can detect such kind of spoof attacks. This plug-in is not efficient for an Internet user with better knowledge of explorer and its activity, but users with not much knowledge about browsing and explorer can be assisted with it. The Internet users also need to be educated about the fraudulent activities and precautions to be taken while dealing with Internet and emails. In order to effectively reduce the impact of Internet fraud based on web spoofing, Spoof Guard must be distributed and deployed, or the
mechanisms tested in the development of the tool must be adopted by browser companies and integrated into standard browser security mechanisms.

The future scope for working on spoof guard can be using more specific methods in detecting spoof pages. A phisher, with good knowledge about Spoof Guard can figure out technical ways to deceive the plug-in and collect compromised data from the Internet users. As the rivalry between the phishers and phishing detectors is never ending, these technical methods used by the phishers can be studied and new features can be implemented to overcome the detected limitations.
ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Mario Al. Garcia, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his expertise in the area of Expert Systems, excellent guidance and encouragement during the period of this project work.

My sincere thanks to Dr. Dulal Kar, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his keen interest, unending support and warm wishes that provided me the much-needed motivation in completing my project.

My sincere thanks to Dr. Long-zhuang Li, Assistant Professor of Computer Science, Texas A&M University – Corpus Christi, for his keen interest in the field of Network Security, his patience and timely help has been very useful.

Finally my heartfelt gratitude to all the faculty and staff of the Department of Computing Sciences, my colleagues and friends for their moral support.
BIBLIOGRAPHY & REFERENCES

[Anti-Phishing] www.antiphishing.org


APPENDIX A. TERMINOLOGY

- **Phishing**: An Internet crime to gain compromised information from an Internet user.

- **Phisher**: The person who performs the phishing attack.

- **Spoof Index**: A threshold level at which the possibility of a page being spoof is calculated.

- **Cache**: A small size memory to store temporary information, used frequently, like the browser activity.

- **Domain**: The base name of a particular URL. Example: yahoo.com is domain name of www.mail.yahoo.com.

- **Hashing**: A process of converting simple text into a machine readable code.

- **Browser Helper Object**: A module used in Internet explorer to provide flexible functionality.

- **COM component**: A platform to provide inter-process communication and create dynamic objects in any suitable programming language.

- **Window Class**: Module built in Spoof Guard to perform a specific functionality.

- **Microsoft ATL**: Microsoft’s Active Template Library (ATL) is used to create flexible functionality in C++.

- **Command Handlers**: A set of statements executed when a particular application command is called.

- **POP-3**: Post Office Protocol. A version of protocol used to send and receive emails.

- **Denial of Service**: A method of making a system or a network unavailable to intended users.

- **Edit Distance**: The Edit Distance is the number of characters that need to be inserted or deleted in order to transform one domain into another.
APPENDIX B. HEADER CODE

************************************************************************
* The following code is for the header Warnbar.h. This is the major header file which *
* calls all other functions.                                                              *
************************************************************************

//WarnBar.h : Declaration of the CWarnBar

#ifndef __WARNBAR_H_
#define __WARNBAR_H_

#include "resource.h"      // main symbols
#include "ReflectionWnd.h"  // To channel messages through to UWToolBar

// These are needed for IDeskBand
#include <shlguid.h>
#include <shlobj.h>
#include <mshtml.h>
#include <wchar.h>

// For IDispEvent Sinkhole
#include <exdispid.h>

#include <sys/timeb.h>
#include <time.h>
#include <list>

using namespace std;

#include <ExDisp.h>
#include <mshtml.h>
#pragma warning(disable : 4192) // multiple defines on typical ifaces, okay to
#pragma warning(disable : 4049) // MSHTML is a *big* TLB
#pragma warning(disable : 4146) // unary minus operator in TLB, doesn't always ignore though
#include "c:\winnt\system32\mshtml.tlb" raw_interfaces_only, raw_native_types, named_guids
#include "c:\winnt\system32\shdocvw.dll" exclude("tagREADYSTATE")
raw_interfaces_only, raw_native_types, named_guids

//URL Check Subs
#define SUBS 12
#define SUB1 _T(".com")
#define SUB2 _T(".net")
#define SUB3 _T(".org")

59
#define SUB4 _T(".us")
#define SUB5 _T(".biz")
#define SUB6 _T(".info")
#define SUB7 _T(".tv")
#define SUB8 _T(".ws")
#define SUB9 _T(".cc")
#define SUB10 _T(".edu")
#define SUB11 _T("http")
#define SUB12 _T("www.")

//Username Check Defs
#define USER 20
#define USER1 _T("login")
#define USER2 _T("user")
#define USER3 _T("username")
#define USER4 _T("screenname")
#define USER5 _T("userid")
#define USER6 _T("usr_name")
#define USER7 _T("user_name")
#define USER8 _T("screen_name")
#define USER9 _T("customer")
#define USER10 _T("cust")
#define USER11 _T("loginname")
#define USER12 _T("custid")
#define USER13 _T("accessid")
#define USER14 _T("access_id")
#define USER15 _T("loginid")
#define USER16 _T("login_id")
#define USER17 _T("user_id")
#define USER18 _T("login_name")
//the last two are not "usernames"
//but some sites use email instead of login names
//if none of USER1-USER18 is found, check for email
#define USER19 _T("login_email")
#define USER20 _T("email")

//Password Check Defs
#define PASS 10
#define PASS1 _T("password")
#define PASS2 _T("passcode")
#define PASS3 _T("passwd")
#define PASS4 _T("pass")
#define PASS5 _T("user_password")
#define PASS6 _T("usr_password")
#define PASS7 _T("login_password")
#define PASS8 _T("login_passcode")
// For Scanner
#define WHITE_SPACE_DELIMITERS " \t\n "
#define TOKEN_LEN 128
#define DEFAULT_DOMAINNAME "DOMAIN"
#define IMAGE_FILENAME _T("..\\SpoofImg.bin")
#define REFERRER_LIST_FILENAME _T("..\\SpoofWatchReferrerList.txt")
#define MAX_IMAGE_HASHES 1023
#define IMG_MINWIDTH 25
#define IMG_MAXWIDTH 700
#define IMG_MINHEIGHT 25
#define IMG_MAXHEIGHT 700
#define IMG_MINSIZE 64
#define IMG_MAXSIZE 75 * 1024
#define SAFE_RELEASE(x) {x->Release(); x = NULL;}

#define TEST_DOMAIN 0
#define TEST_URL 1
#define TEST_EMAIL 2
#define TEST_LINK 3
#define TEST_IMAGE 4
#define TEST_INPUT 5
#define TEST_HTTPS 6

#define AVAILABLE_HISTORY 7
#define AVAILABLE_URL 8
#define IN_HISTORY 9
#define NUMBER_OF_TESTS 10
#define NUMBER_OF_WARNINGS 10
#define TESTING_PREFIX _T("http://crypto.stanford.edu/SpoofGuard/BEWARE_FAKE_SITE/")
#define TESTING_INDEX 55

////////////////////////////////////////////////////////////////////////////
// CWarnBar
class ATL_NO_VTABLE CWarnBar :
public CComObjectRootEx<CComSingleThreadModel>,
public CComCoClass<CWarnBar, &CLSID_WarnBar>,
public IDeskBand,
public IObjectWithSite,
public IPersistStream,
public IDispatchImpl<IWarnBar, &IID_IWarnBar, &LIBID_URLWARNINGLib>,
public IDispEventImpl<0, CWarnBar,
&__uuidof/*SHDocVw::*'/DWebBrowserEvents2), &LIBID_SHDocVw, 1, 0>
{
  typedef IDispEventImpl<0, CWarnBar,
&__uuidof/*SHDocVw::*'/DWebBrowserEvents2), &LIBID_SHDocVw, 1, 0>
theDispEvent;
public:
CWarnBar();
DECLARE_REGISTRY_RESOURCEID(IDR_WARNBAR)
DECLARE_PROTECT_FINAL_CONSTRUCT()
BEGIN_CATEGORY_MAP(CWarnBar)
  // IMPLEMENTED_CATEGORY(CATID_InfoBand)
  // IMPLEMENTED_CATEGORY(CATID_CommBand)
  // IMPLEMENTED_CATEGORY(CATID_DeskBand)
END_CATEGORY_MAP()

//COM interfaces supported
//we mainly care about IObjectWithSite
BEGIN_COM_MAP(CWarnBar)
COM_INTERFACE_ENTRY(IWarnBar)
COM_INTERFACE_ENTRY(IOleWindow)
COM_INTERFACE_ENTRY_IID(IID_IDockingWindow, IDockingWindow)
COM_INTERFACE_ENTRY(IOleWindow)
COM_INTERFACE_ENTRY_IID(IID_IDeskBand, IDeskBand)
COM_INTERFACE_ENTRY(IPersist)
COM_INTERFACE_ENTRY(IPersistStream)
COM_INTERFACE_ENTRY(IDispatch)
END_COM_MAP()

//this registers our event sinks (beforenavigate, documentcomplete)
//that allow us to get useful information from the browser
BEGIN_SINK_MAP(CWarnBar)
//SINK_ENTRY(0, (__uuidof(DWebBrowserEvents2)), BeforeNavigate2);
SINK_ENTRY_EX(0, (__uuidof(DWebBrowserEvents2)), DISPID_BEFORENAVIGATE2, BeforeNavigate2)
SINK_ENTRY_EX(0, (__uuidof(DWebBrowserEvents2)), DISPID_DOCUMENTCOMPLETE, DocumentComplete)
// SINK_ENTRY_EX(0, (__uuidof(DWebBrowserEvents2)),
DISPID_NEWWINDOW2, NewWindow2)

//DISPID_NEWWINDOW2
//SINK_ENTRY_EX(0, (__uuidof(DWebBrowserEvents2)),
DISPID_NAVIGATECOMPLETE2, NavigateComplete2)

//SINK_ENTRY_EX(0, (__uuidof(DWebBrowserEvents2)), 0x000000fc,
NavigateComplete2)
END_SINK_MAP()

//these are the methods that our interfaces must implement

// DWebBrowserEvents2
public:
HRESULT __stdcall DocumentComplete(IDispatch* pDisp, VARIANT* URL);
HRESULT __stdcall NavigateComplete2(IDispatch* pDisp, VARIANT* URL);
HRESULT __stdcall BeforeNavigate2(IDispatch *pDisp, VARIANT *URL,
VARIANT *Flags,
VARIANT *TargetFrameName, VARIANT *PostData,
VARIANT *Headers, VARIANT_BOOL *Cancel);

HRESULT __stdcall NewWindow2(IDispatch** &ppDisp,VARIANT_BOOL* &Cancel);

// IDeskBand
public:
STDMETHOD(GetBandInfo)(DWORD dwBandID, DWORD dwViewMode,
DESKBANDINFO* pdbi);

// IObjectWithSite
public:
STDMETHOD(SetSite)(IUnknown* pUnkSite);
STDMETHOD(GetSite)(REFIID riid, void **ppvSite);

// IOleWindow
public:
STDMETHOD(GetWindow)(HWND* phwnd);
STDMETHOD(ContextSensitiveHelp)(BOOL fEnterMode);

// IDockingWindow
public:
STDMETHOD(CloseDW)(DWORD dwReserved);
STDMETHOD(ResizeBorderDW)(LPCRECT prcBorder, IUnknown *punkToolBarSite, BOOL fReserved);
STDMETHOD(ShowDW)(BOOL fShow);

// IPersist
public:
STDMETHOD(GetClassID)(CLSID *pClassID);

// IPersistStream
public:
STDMETHOD(IsDirty)(void);
STDMETHOD(Load)(IStream *pStm);
STDMETHOD(Save)(IStream *pStm, BOOL fClearDirty);
STDMETHOD(GetSizeMax)(ULARGE_INTEGER *pcbSize);

//ILED242
STDMETHOD(SendEmail)(int isBefore);

// IWarnBar

//helper
BOOL RegisterAndCreateWindow();

DWORD m_dwBandID;
DWORD m_dwViewMode;
BOOL m_bShow;
BOOL m_bEnterHelpMode;
HWND m_hWndParent;

//HWND m_hWnd;
//our reflection window
CReflectionWnd m_ReflectionWnd;
IInputObjectSite* m_pSite;

//our history
IUrlHistoryStg* m_pHistory;

//our list of history entries
CStringList* m_pHistoryList;

IWebBrowser2* m_webBrowser;

//temporary variables to store values from the registry
BOOL m_LoadedFromReg;
int m_RegHist;
int m_RegDist;
BOOL m_RegWarnSendMail;
BOOL m_RegWarn;
BOOL m_RegWarnI;
BOOL m_RegWarnL;

int m_RegDWeight;
int m_RegUWeight;
int m_RegHWeight;
int m_RegLWeight;
int m_RegIWeight;
int m_RegTotalAlert;

//LED
int m_RegLSensitivity;
int m_RegPWeight;
int m_RegXWeight;
int m_RegXFrameWeight;

//control variables to keep track of the state of our warnings
//int preSum;
//int postSum;
int flags[NUMBER_OF_TESTS];
CString warnings[NUMBER_OF_WARNINGS];

int alreadyWarned;
int outerFramesInHistory;
int repeatedImageFound;
int badLinkFound;
int inputFieldsFound;

// int beforeNaved;
int registered;
CString outerDomain;

// int redirect;
// int framesInHistory;

int cancelledNavigation;
int encryptedSite;
int isForm;
int isSubmit;
//for performance checks
struct _timeb timebuffer1;
struct _timeb timebuffer2;

//int isTopFrame;
//int depth;

/*
 int m_D;
 CString m_DStr;
 int m_U;
 CString m_UStr;
 int m_H;
 CString m_HStr;
*/
//CString totalErrorMessage;

/*
 CString m_CompositeErrorMessage;
 CString m_CurrentCompositeErrorMessage;
*/
//CString domainSubstrings[SUBS];

// IDispatch Object
//IDispatch* m_pDisp;
public:
 // IUnknown* globalpUnkSite;

// Helper functions
int LevenshteinDistance(const char* cStr1, const char* cStr2);
void EnumerateHistory();
CString IsCloseMatch(const char *hostName);
void LoadRegVals();
void SaveRegVals();

int FirstAlert(URL_COMPONENTS urlComps);
int FirstAlertClean(URL_COMPONENTS urlComps);
int SecondAlert(IDispatch* pDisp, VARIANT* URL);
int DomainCheck(URL_COMPONENTS urlComps, int str);
int EmailCheck(URL_COMPONENTS urlComps, int str);
int HTMLCheckPasswords(IDispatch &myDispatch2);
void HTMLCheckButtons(IDispatch &myDispatch2);

int URLCheck(URL_COMPONENTS urlComps, int str);
int HTMLCheckInput(IHTMLDocument2 *doc, VARIANT* URL);
int HTMLCheckLinks(IHTMLDocument2 *doc, VARIANT* URL);
void ChangeLight(int sum);

int ImageCheck(IHTMLDocument2 *doc, VARIANT* URL);
unsigned int BstrToInt(BSTR bstr);

int ReadImageIntoBuffer(const CString& imageURL, BYTE *imageBuffer, unsigned int imageSize);

PBYTE HashBuffer(BYTE *imageBuffer, unsigned int imageSize, DWORD *hashLength);

int IterateHashFile(BYTE *hashArray[], unsigned int numHashes, unsigned int hashLength, unsigned int domainLength);

int FileExists();

int GetDocument(IDispatch* pDisp, IHTMLDocument2 **returnDoc);
// int GetDocument3(IDispatch* pDisp, IHTMLDocument3 **returnDoc);
void PostDataCheck(VARIANT *PostData, VARIANT_BOOL *Cancel);
// void ShowHistoryMessage(URL_COMPONENTS &urlComps);
// void ShowNoHistoryMessage(URL_COMPONENTS &urlComps);
// void ShowOKFirstMessage(URL_COMPONENTS &urlComps);
// void ShowNoFirstMessage(URL_COMPONENTS &urlComps);
// void ShowSpoofAlert(URL_COMPONENTS &urlComps, VARIANT_BOOL *Cancel);

void UpdateStatus();

void ScanPostData(char *postData, VARIANT_BOOL *Cancel);

CString GetToken(const CString& token, const CString ids[], int len);

bool isHavingMultiplePasswords(const CString& token, const CString ids[], int len);
list<CString> getPasswordsList(const CString& token, const CString ids[], int len);

FILE* OpenPasswordFile();
void CheckPassword(const CString& user, const CString& password, VARIANT_BOOL *Cancel);

FILE* OpenPasswordFileAdd();

CString HashPassword(const CString& pass);

int FrameCheck(IHTMLDocument2 *pHTMLDoc, VARIANT* url);
{
    int Thresh();
    int Total();
}
#endif //__WARNBAR_H_