Client Side Protection from Phishing Attacks

A Dissertation submitted to the University of Hyderabad in partial fulfillment of the degree of

MASTER OF TECHNOLOGY

in

Information Technology

by

VENKATA PRASAD REDDY K

Department of Computer and Information Sciences

School of Mathematics, Computer and Information Sciences

University of Hyderabad
(P.O.) Central University, Gachibowli
Hyderabad – 500 046
Andhra Pradesh
India

CERTIFICATE

This is to certify that the dissertation entitled **"Client side protection from Phishing attacks"** submitted by **Venkata Prasad Reddy K.** bearing Reg. No **09MCMB46** in partial fulfillment of the requirements for the award of Master of Technology in Information Technology is a bonafide work carried out by him under my supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. V. Radha
Assistant Professor, IDRBT.
Signature of the Supervisor

Head of the Department

Dean of the School

DECLARATION

I, **Venkata Prasad Reddy K,** hereby declare that this Dissertation entitled **"Client side protection from Phishing attacks"**, submitted by me under the guidance and supervision of **Dr. V. Radha, Assistant Professor, IDRBT**, is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: Name: **Venkata Prasad Reddy K.**

Signature of the Student:

Regd. No. **09MCMB46**

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. V. Radha, Assistant Professor, IDRBT, Hyderabad, who generously supported and guided me throughout my project, IDRBT for providing me with the infrastructure and technical support that I need for this project. The project would not have been possible without her assistance.

I also thank Mr. B.Sambamurthy, Director, IDRBT, and Prof. Dr. Arun Agarwal, Dean, School of MCIS, Prof. Dr. C.R.Rao, Head of the Department (DCIS), University of Hyderabad for extending their cooperation.

The guidance of all the faculty members of IDRBT and the Department of Computer and Information Sciences, University of Hyderabad has been precious and timely. I wish to thank them for being very patient, understanding and helpful.

Venkata Prasad Reddy K,

E-mail ID*: kvp.reddy@ymail.com*

University of Hyderabad & IDRBT,

Hyderabad

.

LIST OF PAPERS COMMUNICATED

- Venkata Prasad Reddy K , V.Radha and Manik Jindal, “Client Side protection from Phishing attack” International Journal of Advanced Engineering Sciences and Technologies (IJAEST) Vol No. 3, Issue No. 1, 039 – 045.

ABSTRACT

In phishing, users could be easily tricked into submitting their username and password into fraudulent web sites whose appearance look similar to the genuine one. In this paper, we present two approaches, where in, one prevents the user from entering into a spoofing site and the other, safeguards the users' password even if he interacts with the spoofing site.

The first approach is a spoof alert which maintains a white-list of all Indian bank websites. It will alert the user when the domain name of the site he visited has similarities with any Indian bank site and then has no match with the ones in the white-list. If the user enters the phishing site by chance the second approach will come into play. In this approach, we describe a browser extension which provides a trusted window dedicated for password entry displaying a photographic image. Users select specific images to be displayed against some of the sites they visit. No image would be displayed for a spoofing site and hence the user can easily detect the trick. Also, the images never leave the client and hence the spoofing site can't spoof the images. This browser extension transparently produces a different password for each site by using Pwdhash++, an enhanced Pwdhash. Pwdhash++ could resist scripting attacks that are made public.

TABLE OF COTENTS

1 INTRODUCTION

Phishing is a way of attempting to acquire sensitive information such as usernames, passwords and credit card details by masquerading as a trustworthy entity in an electronic communication [1, 2]. Communications purporting to be from popular social web sites, auction sites, online payment processors or IT administrators are commonly used to lure the unsuspecting public. Phishing is typically carried out by e-mail or instant messaging[3], and it often directs users to enter details at a fake website whose look and feel are almost identical to the legitimate one. Phishing is an example of social engineering techniques used to fool users [4], and exploits the poor usability of current web security technologies [5]. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and technical security measures.

A phishing technique was described in detail in 1987, and the first recorded use of the term "phishing" was made in 1996. The term is a variant of *fishing*, probably influenced by *phreaking*[6. 7], and alludes to baits used to "catch" financial information and passwords.

1.1 The nature of phishing attacks: Deception

Phishers use a wide variety of technologies, with one common thread. All technologies employed by phishers have the goal of deception [8]. For example:

- Deceiving a user into believing a message comes from a trusted source;
- Deceiving a user into believing that a web site is a trusted institution;
- Deceiving a spam filter to classify a phishing email is legitimate.

Phishers are technically innovative, and can afford to invest in technology. It is a common misconception that phishers are amateurs. This is not the case for the most dangerous phishing attacks. As financial institutions have increased their online presence, the economic value of compromising account information has increased

dramatically. Criminals such as phishers can afford to invest in technology commensurately with the illegal benefits gained by their crimes.

Given both the current sophistication and rapid evolution of phishing attacks, a comprehensive catalogue of technologies employed by phishers is not feasible. Given that, a brief review of typical practices will help illuminate the problem, and motivate the countermeasures.

Deceptive return address information

Phishing emails typically claim to come from a trusted source. There are two primary ways in which this is accomplished:

- Forging a return address;
- Registering an official-looking domain (e.g. “commerceflow-security.com” to spoof a company whose real domain is “commerceflow.com”) and sending email from that domain name.

Fraudulent request for action

Phishing emails typically claim to require user action to prevent a problem with an account. They may claim to be conducting a security audit, or to have detected fraud on the user’s account, or to require updated contact information. The request for action must seem authentic to prompt the user to reveal confidential information.

Deceptive appearance

Effective phishing emails and web sites must present a visual appearance consistent with the institutions that they are mimicking. There are many elements to this.

Deceptive visual appearance

Color schemes and imagery mimic the targeted institution.

Deceptive links

A call to action from a phisher typically requires a user to click on a link in a message to go to a web site. The phisher’s web site does not have a legitimate name, so the actual destination is often disguised. (An exception to this rule is web sites that are simply misleadingly named.) Phishers employ many technologies to obscure the true destination of a link. Examples include:

- *Misleadingly named links* – A link may display as http://security.commerceflow.com but actually lead to http://phisher.com.
- *Cloaked links* – URLs can incorporate a user name and password. This can be used to "cloak" the actual destination of a link. For example, the URL http://security.commerceflow.com@phisher.com actually leads to http://phisher.com.
- *Redirected links* – "Redirects" that translate a reference to one URL into another URL are commonly used in web programming. If a careless programmer at a targeted institution leaves an "open redirect" accessible that can be used to redirect to an arbitrary location, this can be used by phishers to provide a legitimate-looking URL that will redirect to their site.
- *Obfuscated links* – URLs can contain encoded characters that hide the meaning of the URL. This is commonly used in combination with other types of links, for example to obscure the target of a cloaked or redirected link.
- *Programmatically obscured links* – If scripts are allowed to run, Javascript can change the status text when the user mouses over a link to determine its destination.
- *Map links* – A link can be contained within an HTML "image map" that refers to a legitimate-looking URL. However, the actual location to which a click within the image map directs the browser will not be displayed to the user.

Deceptive location

Once a phisher has convinced a user to click on a link, the phisher must maintain the deception that the user is at a legitimate location. This again involves many rapidly changing technologies. One aspect of deceiving the user as to the location of the browser is to use deceptive links. Another is to ensure that deceptive information appears in the URL bar. For example, phishers have created Javascript programs that pop up a borderless window to obscure the real contents of the URL bar, and move the window when the user moves his window. Some of these Javascript programs simulate the window history if the user clicks on the history box. It is not possible to determine whether a connection to a site is secure (i.e. uses SSL) by looking at a lock icon in a browser. There are several reasons why a lock icon cannot be trusted:

- A lock icon by itself means only that the site has a certificate; it does not confirm that the certificate matches the URL being (deceptively) displayed. A user must click on a lock icon to determine what it means, and few ever do.
- It is possible to get a browser to display a lock icon using a self-signed certificate (i.e. a certificate that has not been issued by a valid certificate authority), with certain encryption settings.
- A lock icon may be overlaid on top of the browser using the same technologies used to fake the URL bar. This technology may even be used to present authentic-looking certificate data if the user clicks on the lock icon to confirm legitimacy.

While browser technologies are constantly being updated to address recent phishing tactics, browsers are large, complex programs that must provide considerable functionality and flexibility to satisfy the needs of legitimate web site designers. It is highly improbable that deceptive phishing appearances can be completely stopped solely by addressing phishing technologies piecemeal.

Deceptive information flow

To maximize the value of a compromise, the user should not know that he or she has provided confidential information to a phisher. After obtaining confidential information, phishing sites often inform the user that he or she must log back into her account now that the information is "confirmed," and redirect to the legitimate site.

Phishers sometimes construct elaborate information flows to cover their tracks and conceal the ultimate destination of compromised information. In some cases, these information flows can contain multiple media, such as compromised "zombie" machines, instant messaging, and anonymous peer-to-peer data transfer mechanisms.

1.2 The flow of information in a phishing attack and countermeasures

0. The phisher prepares for the attack. Step 0 countermeasures include monitoring malicious activity to detect a phishing attack before it begins.
1. A malicious payload arrives through some propagation vector. Step 1 countermeasures involve preventing a phishing message or security exploit from arriving.

2. The user takes an action that makes him or her vulnerable to an information compromise. Step 2 countermeasures involve detecting phishing tactics and rendering phishing messages less deceptive.
3. The user is prompted for confidential information, either by a remote web site or locally by a Web Trojan. Step 3 countermeasures are focused on preventing phishing content from reaching the user.
4. The user compromises confidential information. Step 4 countermeasures concentrate on preventing information from being compromised.
5. The confidential information is transmitted from a phishing server to the phisher. Step 5 countermeasures involve tracking information transmittal.
6. The confidential information is used to impersonate the user. Step 6 countermeasures centre on rendering the information useless to a phisher.
7. The phisher engages in fraud using the compromised information. Step 7 countermeasures focus on preventing the phisher from receiving money.

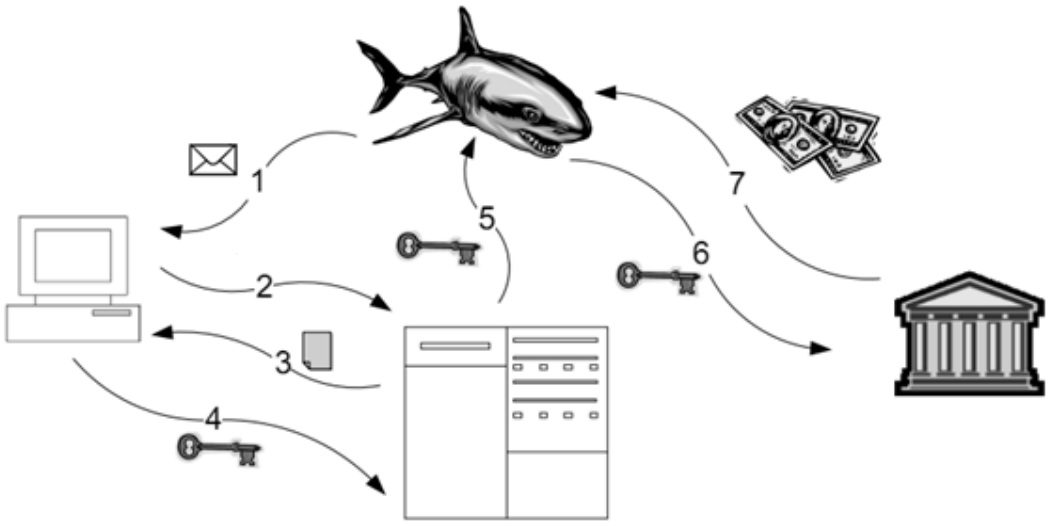

Figure 1: The information flow in a phishing attack.

Phishing is complex phenomenon that includes social factors as well as technology. There is no single "silver bullet" that can prevent all phishing. However, properly applied technology can significantly reduce the risk of identity theft. There are many opportunities to apply such technology, [9] including:

- Monitoring potentially malicious activity such as web site usage and domain registrations, detecting a phishing attack before it starts, and interrupting the phisher's preparations (step 0).

- Authenticating email messages so unauthenticated messages can be discarded (step 1).
- Detecting the unauthorized use of trademarks, logos and other proprietary imagery (step 1).
- Improving the security patching infrastructure to increase resistance to malware (step 1).
- Using personalized information to authenticate an email directly to a user (step 2).
- Detecting a fraudulent web site and alerting the user (step 4).
- Using a mutual authentication protocol (step 4).
- Establishing a trusted path between the user and a web site to ensure that information can be used only by its intended recipient (steps 4 and 6).
- Using two-factor authentication (step 6).
- Forcing passwords to be site-specific (step 6).
- Encoding credentials with restrictions on their validity, using public key cryptography (step 6).

1.3 Common Password Problem

Most users want a password they can remember easily; because of this they use their names, pet's names, or birthdays. Users tend to use a single password at many different web sites. This would be bad. Anyone who knows the user can easily guess this information. Although it may take a few tries, it is not a good password. Not only could the attacker break the original account, they could also break other accounts that the user has [10].

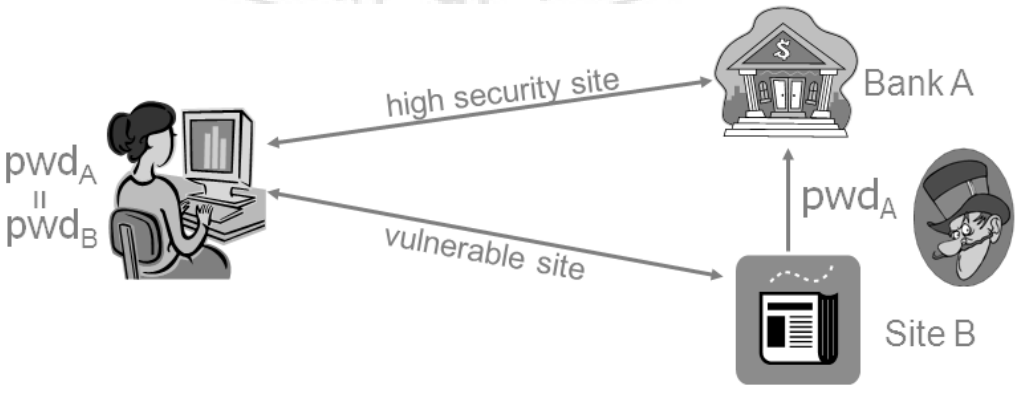

Figure 2: Flow of common password attack.

In common password attacks, hackers exploit the fact that web users often use the same password at many different sites. This allows hackers to break into a low security site that simply stores username/passwords in the clear and use the retrieved passwords at a high security site, such as a bank. This attack, which requires little work, can lead to the theft of thousands of banking passwords. While password authentication could be abandoned in favor of hardware tokens or client certificates, both options are difficult to adopt because of the cost and inconvenience of hardware tokens and the overhead of managing client certificates.

1.4 Problem Statement

Spoofguard does not use white lists or blacklists. Instead, the toolbar employs a series of heuristics to identify phishing pages. As per our survey, which is discussed thoroughly in the later sections, the false positive rate i.e., alerting a genuine site as a fraudulent site, is high for Spoofguard. This thesis addresses this drawback of Spoofguard.

PwdHash is a browser extension that transparently converts a user's password into a domain-specific password. This particular tool is vulnerable to javascript attack and also have certain drawbacks. This thesis also addresses these drawback of Pwdhash.

1.5 Organization of Thesis

The rest of the thesis is organized as follows. In the first chapter, the problem, phishing, is explained. Next, the second chapter is devoted to the survey made on some of the present anti-phishing approaches, being spam filters, anti-phishing toolbars and password protection mechanisms. In the third chapter an in detail description of Spoofguard toolbar is given, followed by Pwdhash in the fourth chapter. The fifth chapter presents two anti-phishing approaches that have been designed as an enhancement to Spoofguard and Pwdhash respectively. Finally, we conclude our solution.

2 SURVEY

To counter the phishing threat, a number of anti-phishing solutions have been proposed, both by industry and academic world[16]. The anti phishing techniques can in general be divided into three categories. 1. Spam Filters 2. Anti-phishing toolbars and 3. Password protection mechanism.

2.1 Spam Filters

A class of anti-phishing approaches aims to solve the phishing problem at the email level. The key idea is that when a phishing email does not reach its victims, they cannot fall for the scam. Hence, filters and content analysis techniques are often used to attempt to identify phishing emails before these emails are delivered to users. Clearly, this line of research is closely related to anti-spam research [17]. By continuously training filters (e.g., Bayesian filters), a large number of phishing emails can be blocked. This is because such emails often contain words that may be identified as suspicious tokens that do not frequently occur in legitimate emails (e.g., “update”, “login”, etc.). The main disadvantage of anti-spam techniques is that their success depends on the availability of these filters and their proper training. That is, when the user does not actively help in training the filter, the filter typically does not perform as expected. Furthermore, even when filters are trained well and a user rarely receives any spam or phishing emails, once a phishing email bypasses the filter, the user’s belief of the legitimacy of this mail is strengthened.

2.2 Anti-phishing toolbars

To identify a page as a phishing site, there are a variety of methods that can be used, such as white lists (lists of known safe sites), blacklists (lists of known fraudulent sites), various heuristics to see if a URL is similar to a well-known URL, and community ratings. The toolbars examined here employ different combinations of these methods. By using publicly available information provided on the toolbar download web sites as well as observations from using each toolbar we get a basic

understanding of how each toolbar functions. Some of the toolbars that are used for anti-phishing are [23],

1) eBay Toolbar

The eBay Toolbar [28] uses a combination of heuristics and blacklists. The Account Guard indicator has three modes: green, red, and gray. The icon is displayed with a green background when the user visits a site known to be operated by eBay (or PayPal). The icon is displayed with a red background when the site is a known phishing site. The icon is displayed with a gray background when the site is not operated by eBay and not known to be a phishing site. The toolbar also gives users the ability to report phishing sites, which will then be verified before being blacklisted.

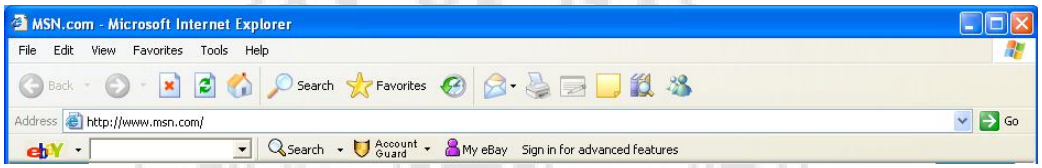

Figure 3: The eBay Toolbar at a site not owned by eBay that is not known to be a phishing site.

2) GeoTrust TrustWatch Toolbar

GeoTrust's TrustWatch Toolbar [27] labels sites as green, yellow or red. GeoTrust's web site provides no information about how TrustWatch determines if a site is fraudulent; however, we suspect that the company compiles a blacklist that includes sites reported by users through a button provided on the toolbar. The toolbar also allows the user to store a custom image or bit of text that is constantly displayed so that he or she knows that the toolbar is not being spoofed.

Figure 4: The GeoTrust TrustWatch Toolbar at a verified site.

3) Google Safe Browsing

The Google Toolbar [25] includes a tool called "Google Safe Browsing" designed to identify fraudulent web sites. Google provides the source code for the Safe Browsing feature and says that it checks URLs against a blacklist. However, it is not known how URLs are added to the blacklist. According to the download site, the tool

combines “advanced algorithms with reports about misleading pages from a number of sources” .We suspect that this means that the tool uses blacklists as well as heuristics. The toolbar also includes a PageRank feature that can be useful in identifying phishing sites, as most phishing sites have a very low PageRank. The toolbar displays a popup if it suspects the visited site to be fraudulent and provides users with a choice of leaving the site or ignoring the warning. The toolbar includes an “Enhanced Protection” option, which the software claims “will provide more up-to-date protection by sending the URLs of sites that you visit and limited information about the site content to Google for evaluation.”

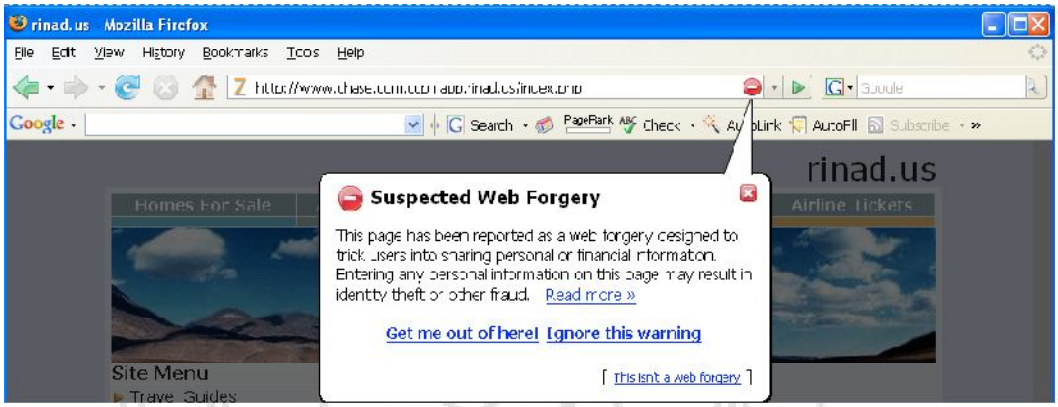

Figure 5: The Google Toolbar at a fraudulent site.

4) McAfee SiteAdvisor

SiteAdvisor [29] claims to detect not just phishing websites, but any sites that send spam, offer downloads containing spyware, or engage in other similar bad practices. The toolbar follows the traffic light model, similar to some of the other toolbars examined. However, when SiteAdvisor encounters a site for which it doesn’t have any information, it remains gray. The determination is made by a combination of automated heuristics and manual verification.

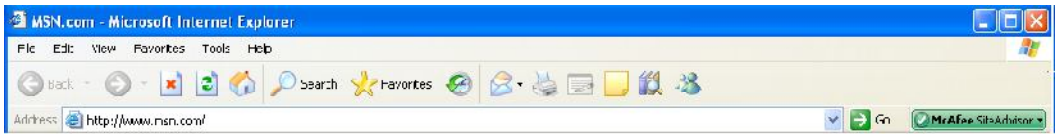

Figure 6: McAfee SiteAdvisor at a verified good site.

5) Microsoft Phishing Filter in Windows Internet Explorer

The toolbar largely relies on a blacklist hosted by Microsoft [30]. However, it also uses some heuristics when it encounters a site that isn’t in the blacklist. When a suspected phishing website is encountered, the user is redirected to a built in warning

message and asked if they would like to continue visiting the site or close the window. Users also have the option of using this feature to report suspected phishing sites or to report that a site has incorrectly been added to the blacklist.

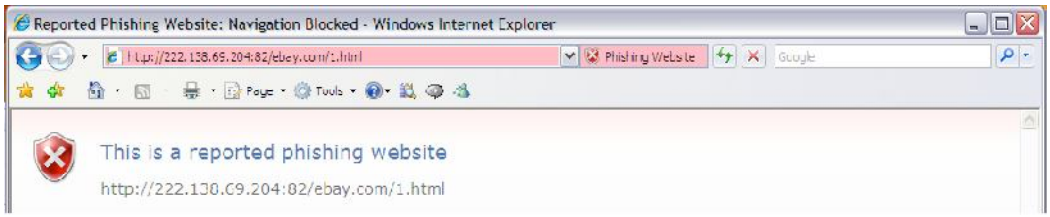

Figure 7: The Microsoft Phishing Filter in Windows Internet Explorer 7 at a fraudulent web site.

6) Netcraft Anti-Phishing Toolbar

The Netcraft [24] web site explains that the toolbar "traps suspicious URLs containing characters which have no common purpose other than to deceive," "enforces display of browser navigation controls in all windows, to defend against popup windows which attempt to hide the navigational controls," and "clearly displays sites' hosting location, including country helping you to evaluate fraudulent URLs. The Netcraft toolbar also uses a blacklist, which consists of fraudulent sites identified by Netcraft as well as sites submitted by users and verified by the company. The toolbar also displays a risk rating between one and ten as well as the hosting location of the site.

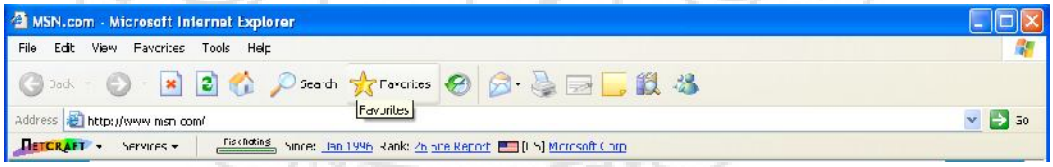

Figure 8: The Netcraft Anti-Phishing Toolbar at a legitimate web site.

7) Netscape Browser 8.1

It appears that the functionality of Netscape Browser [31] relies solely on a blacklist, which is maintained by AOL and updated frequently. When a suspected phishing site is encountered, the user is redirected to a built-in warning page. Users are shown the original URL and are asked whether or not they would like to proceed.

Figure 9: Netscape 8.1 web browser at a fraudulent web site.

8) Spoofguard

SpoofGuard [13] does not use white lists or blacklists. Instead, the toolbar employs a series of heuristics to identify phishing pages. The toolbar first checks the current domain name and compares it with sites that have been recently visited by the user to catch fraudulent web sites that have a similar-looking domain name. Next, the full URL is analyzed to detect obfuscation as well as non-standard port numbers. Afterwards, the contents of the page are analyzed, making note of any password fields, embedded links, and images. Following this, SpoofGuard analyzes links in the web page itself using the heuristics described above. Finally, it examines images on the web page by hashing them to see if it has found similar images on other sites the user has visited. If two identical images are spotted on different web sites, there is a chance that a fraudulent site has copied the images from the legitimate site.

Figure 10: SpoofGuard at a legitimate web site.

9) AntiPhish

AntiPhish[15] is an application that is integrated into the web browser. It keeps track of a user's sensitive information (e.g., a password) and prevents this information from being passed to a web site that is not considered ``trusted'' (i.e., "safe"). The development of AntiPhish was inspired by automated form-filler applications. Most browsers such as Mozilla or the Internet Explorer have integrated functionality that

allows form contents to be stored and automatically inserted if the user desires. This content is protected by a master password. Once this password is entered by the user, a login form that has previously been saved, for example, will automatically be filled by the browser whenever it is accessed. Antiphish takes this common functionality one step further and tracks where this information is sent.

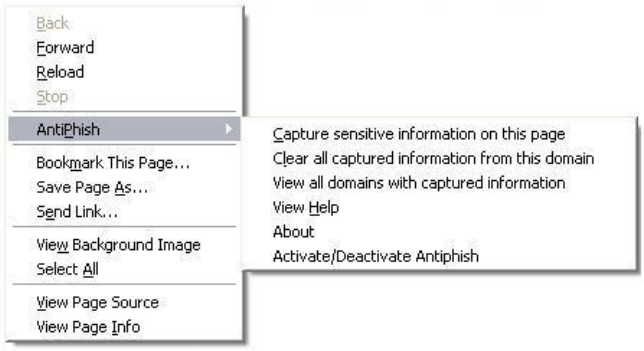

Figure 11: The AntiPhish menu item in the main menu

10) Dynamic security skins

Dynamic security skins[14] is also an academic solution which allow a remote server to prove its identity in a way that is easy for humans to verify. There are two novel interaction techniques to prevent spoofing.

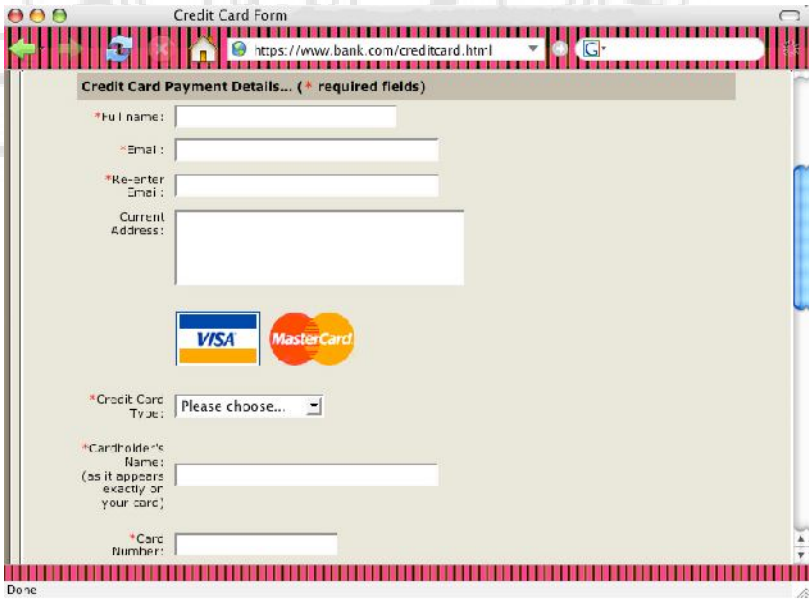

Figure 12: The browser displays the visual hash as a border around the authenticated website.

First, this browser extension provides a trusted window in the browser dedicated to username and password entry. We use a photographic image to create a trusted path between the user and this window to prevent spoofing of the window and of the text entry fields. Second, this scheme allows the remote server to generate a unique abstract image for each user and each transaction. This image creates a "skin" that automatically customizes the browser window or the user interface elements in the content of a remote web page. This extension allows the user's browser to independently compute the image that it expects to receive from the server. To authenticate content from the server, the user can visually verify that the images match.

Most of the tools that were tested used blacklists, but only half of them were able to identify the majority of phishing web sites. We don't know the size of the blacklists used by each toolbar, nor do we know what heuristics are used by any of the toolbars other than Spoofguard. We suspect that the toolbars that performed best use larger and more frequently updated black lists. They may also use heuristics that allow them to detect phishing sites that haven't yet been put on the blacklist.

2.3 Password Protection Mechanism

A password is a secret word or string of characters that is used for authentication, to prove identity or gain access to a resource. The password should be kept secret from those who are not allowed for access. So, the major concern for any user is to safeguard his/her password. The password can be cracked with the attacks such as Guessing attack, Brute-force attack, Dictionary attack, Phishing attack etc.,.

Another problem regarding password is single password problem where the user uses a single password for both vulnerable sites and financial sites. The hackers can break into the vulnerable sites that simply stores username and password and apply those retrieved combination of username and password on high security sites such as banking sites.

All these problems at a single stroke can be solved by hashing the master password using domain name as key on client side. Some of the applications/tools that use this powerful technique are

1) Password Composer

This extension [32] puts a tiny red icon to the left of a password entry field. If one clicks on this icon, the password field is overlaid with a replacement input, where one can supply a single, secure password (Master Password).

2) Magic Password Generator

This extension [33] combines master password and the domain name of the site to make another unique password for that site. For advanced users, with a catchall address at a domain, just put "@example.com" (whatever one's domain is) for the address, and MPWGen will make a different email for every site too. Alternately, use "foo+@…" and the value will be inserted after the + sign, for email accounts that support this feature, like gmail.

3) Password generator

Password Generator [34] gets the hostname from the page's URL and mixes it together with one's personal master password using a little cryptographic magic MD5. It always gets the same result if given that hostname and master password, but will never get that result if either changes.

4) Hassapass

Hasspass [35] automatically generates strong passwords from a master password and a parameter like domain name. The password generation is performed inside this very browser window in JavaScript

5) Genpass

GenPass [36] is a JavaScript/MD5 bookmarklet-based password generator. **GenPass is no longer being updated.** Presently consider using SuperGenPass; however, note that SuperGenPass is not compatible with GenPass—given the same input, they generate different passwords.

6) Password Hasher

When the master key is given to Password Hasher **[**37] and it enters the hash word into the site's password field. A hash word is the result of scrambling the master key with a site tag. Click on a # marker next to a password field or press the **Control-F6** key combination when in a password field or choose **Password Hasher** from either the Tools menu or the right-click popup menu on a password field to enter the master key.

7) Pwdhash

Pwdhash [10, 11] is a browser extension that transparently converts a user's password into a domain-specific password. The user can activate this hashing by choosing passwords that start with a special prefix (@@) or by pressing a special password key (F2). Pwdhash automatically replaces the contents of these password fields with a one-way hash of the pair (password, domain-name).

Based on the features like application type, hashing algorithm, security, password strength, spoof proof, visibility to webpage, visibility to user etc., Pwdhash is the best among the above mentioned applications.

3

SPOOFGUARD

3.1 Introduction

SpoofGuard[12, 13] does not use white lists or blacklists. Instead, the toolbar employs a series of heuristics to identify phishing pages. The toolbar first checks the current domain name and compares it with sites that have been recently visited by the user to catch fraudulent web sites that have a similar-looking domain name. Next, the full URL is analyzed to detect obfuscation as well as non-standard port numbers. Afterwards, the contents of the page are analyzed, making note of any password fields, embedded links, and images. Following this, SpoofGuard analyzes links in the web page itself using the heuristics described above. Finally, it examines images on the web page by hashing them to see if it has found similar images on other sites the user has visited. If two identical images are spotted on different web sites, there is a chance that a fraudulent site has copied the images from the legitimate site.

SpoofGuard computes a score for each web page in the form of a weighted sum of the results of each set of heuristics. Users can change the weights for each set of heuristics in an options menu. If the score surpasses a certain threshold, the toolbar displays a red icon, warning users that the site is a positively identified phishing site. If some of the heuristics are triggered but not enough to exceed the threshold, the icon turns yellow to indicate that it cannot make a determination about the site. If none of the heuristic are triggered, the icon turns green to indicate a safe site.

Properties of recent phishing attacks

We describe common properties of ten spoof web sites recently found in the wild.

- **Logos:** The spoof site uses logos found on the honest site to imitate its appearance.
- **Suspicious URLs:** Spoof sites are located on servers that have no relationship with the honest site. The spoof site's URL may contain the honest site's URL as a substring (http://www.ebaymode.com), or may be similar to the honest URL (http://www.paypaI.com). IP addresses are sometimes used to disguise

the host name (http://25255255255/top.htm). Others use @ marks to obscure their host names (http://ebay.com:top@255255255255/top.html), or contain suspicious usernames in their URLs (http://middleman/http://www.ebay.com)

- **User input:** All spoof sites contain messages to fool the user into entering sensitive information, such as password, social security number, etc. Some successful spoofs have even been so bold as to ask for name, address, mother's maiden name, driver's license, and so on.
- **Short lived:** Most spoof sites are available for only a few hours or days, just enough time for the attacker to spoof a high enough number of users. The implication is that defensive methods that alert the user to a spoof site are more effective than reactive methods that attempt to shutdown the site.
- **Copies:** Attackers copy html from the honest site and make minimal changes. Two consequences are some spoof pages actually contain links to images (e.g. logos and buttons) on the honest site, rather than storing copies and the names of fields and html code remain as on the honest site. We note that when a spoof site refers to the honest site for embedded images it gives the honest site an opportunity to detect the spoof: the honest site detects an http request for an embedded image where the referral header is not the honest site. Such requests should not occur unless the honest site is being plagiarized.
- **Sloppiness or lack of familiarity with English** Many spoof pages have silly misspellings, grammatical errors, and inconsistencies.
- **HTTPS** is uncommon. Most spoof web sites do not use https even if the honest site does. This simplifies setting up the spoof site.

3.2 Approach

A number of tests can be used to distinguish spoof pages from honest pages. We present the tests we implemented and evaluated in three groups: stateless methods that determine whether a downloaded page is suspicious, stateful methods that evaluate a downloaded page in light of previous user activity, and methods that evaluate outgoing html post data. This browser plug-in applies these tests to all downloaded pages and combines the results using a scoring mechanism. The total spoof index of a page determines whether the plug-in alerts the user and determines the severity and type of alert.

Stateless page evaluation

A collection of tests that work by examining the current page only are described below.

Url check: There are various methods that attackers can use to produce misleading urls. For example, an @ in a url causes the string to the left to be disregarded, with the string on the right treated as the actual url for retrieving the page. Combined with the limited size of the browser address bar, this makes it possible to write urls that appear legitimate within the address bar, but actually cause the browser to retrieve a page from an arbitrary site.

Image check: Spoof sites usually contain images taken from the honest site. For example, the eBay logo appears on spoofed eBay pages to give the user the impression that they are communicating with eBay. If the eBay logo appears on a login page unrelated to eBay, that page is suspicious. The same applies to other identifiably eBay-specific images such as banners and buttons. The corporate logos often legitimately appear on many e-commerce sites (e.g., the Amazon logo appears on sites that sell products through Amazon) and therefore this test is counted for pages that ask for private user input. In order to apply this check in a stateless way, the SpoofGuard plug-in is supplied with a fixed database of images and their associated domains. Since attackers generally do not have email lists for customers of specific sites, they must try to spoof sites that are used by a significant fraction of web users. Thus SpoofGuard can be useful even if we only account for relatively small number of frequently spoofed domains such as eBay, PayPal, AOL, and so on. When the browser downloads a login page all images on the page are compared to images in the SpoofGuard database. The spoof-score for the page is increased if a match is found but the page's domain is not a valid domain for the image. What if the spoof page contains a slight modification of the real image? The image comparison test might fail to detect the spoof. Fortunately, as noted earlier, attackers often directly copy or link to images on the honest site. Nevertheless, spoofguard defend against small image modification by storing an image hash rather than the actual image. Image hashing refers to a hashing algorithm that produces the same hash for similar images. While present technology does not provide ideal image hashes, there has been some progress in this area. In this case, image hashing can be strengthened by asking e-commerce sites to use images that are especially well suited for image hashing. For example, in

many cases we could use optical character recognition (OCR) as the image hashing algorithm. An added benefit of image hashing is that storing an image hash rather than the full image reduces plug-in storage requirements.

Link check: The links contained within a page are examined. The link check fails for a page if at least one fourth of the links fail the url check described above.

Password check: Pages that request a password merit closer scrutiny than pages that do not. If a page requests a password (or other sensitive information), spoofguard also checks whether https is used and, if so, whether the certificate check succeeded or failed.

Stateful page evaluation

In stateful page evaluation, the browser history file and additional history stored by SpoofGuard are used to evaluate the referring page. Since it is important to minimize the number of false alarms, SpoofGuard does not issue any warnings for visiting a site that is in the user's history file. The rationale for this is that if the user is warned the first time, and decides to proceed, the user is assumed to have sufficient reason to trust the site.

Domain check: If the domain of a page closely resembles a standard or previously visited domain, the page may be part of a spoof. Although crude, spoofguard currently compares domains by Hamming (edit) distance. For example efrade.com will raise the domain check if etrade.com is in the file of commonly spoofed sites or in the user history. Clearly, it is possible to improve the comparison algorithm by studying the way people are fooled; this is a significant direction for future work. A related issue is that some businesses outsource some of their web operations to contractors with different domain names. This poses an interesting challenge that can be addressed. However, outsourced web activity leads to false alarms in the current version of SpoofGuard.

Referring page: When a user follows a link, the browser maintains a record of the referring page. Since the typical web spoofing attack begins with an email message, a referring page from a web site where the user may have been reading email (such as Hotmail) raises the level of suspicion. One complication associated with Hotmail, for example, is that Hotmail uses numeric IP addresses instead of symbolic host names.

Therefore, when a user clicks on a link in a Hotmail message, the browser provides a numeric IP address to SpoofGuard as the referring page. In this situation, SpoofGuard uses reverse DNS to find the domain name associated with a numeric address, allowing us to identify Hotmail as the referring site.

Image-domain associations: The image check described above relies on a database associating images such as corporate logos with domains. The initial static database can be assembled using a web crawler or other tool, or it can be augmented using an individual's browsing history. An early version of SpoofGuard used a fixed database; the current SpoofGuard implementation uses a hashed image history file.

Evaluating post data

Evaluating post data is a critical part of any client-side defense against web-spoofing attacks, since the point of any defense is to prevent malicious sites from gaining confidential information from an honest web user. When a user fills in form data, SpoofGuard intercepts and checks the html post data, allowing the actual post to proceed only if the spoof index is below the userspecific threshold for posts. If a user confines pop-up warnings to posts only, then the user will never be interrupted when reading web pages, only (possibly) when filling in forms. Note, however, that even if no warnings are generated on pages leading up to the spoof form, the page checks described above are used, in combination with analysis of the post data, to determine the spoof index associated with an html post.

Outgoing password check: SpoofGuard maintains a database of <domain, user name, password> triples. If the user reuses a password on a new domain, this trips the password check. To avoid the possibility of leaking sensitive information, the stored passwords are hashed using SHA-1 and the comparison is performed on hashed values.

Interaction with image check: In the spoof index calculation, the image check interacts with the outgoing password check non-linearly. For example, if a user enters her E*Trade user name and password to a site that is not at etrade.com, this raise the spoof index a certain amount. The spoof index is raised multiplicatively higher if the site also contains the E*Trade logo.

Check of all post data: There are several ways that a web page might request a password. For example, a clever spoof site might use an image of the word

"password" instead of html text to request the user's password. To protect against this form of spoof attack, all outgoing data in an html post can be hashed and checked against a database of passwords and other information deemed sensitive. In this way, spoofguard can still detect password leakage, even if the spoof page does not contain the text "password."

Exception for search engines: Since a user may enter any data into a search engine, SpoofGuard is not suspicious of known search engines at known domains. It is also possible to ignore data posted into a "search" or "find" field in an arbitrary page. This allows for shopping sites, for example, that allow a customer to search the catalog by keyword or product name. Of course, good password practice would prevent an intelligent user from using a product name or other English word as an important password, rendering this exception unnecessary.

3.3 **Disadvantage:** False alarm rate

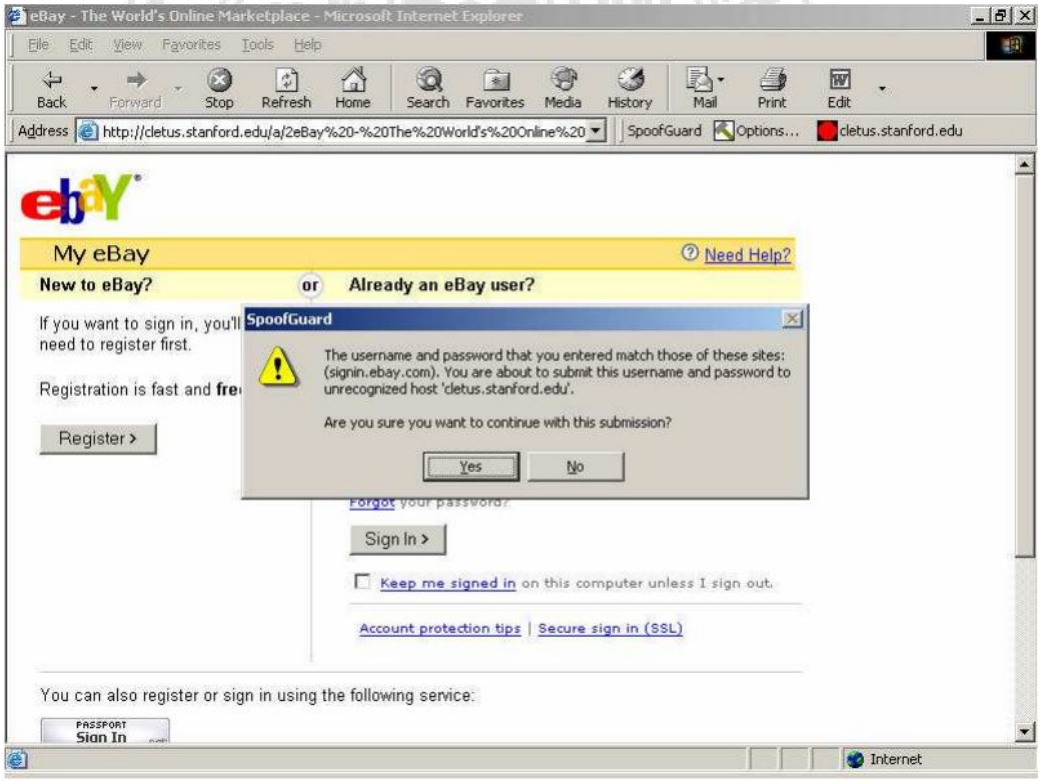

Figure 13: SpoofGuard detects honest username and password on spoof site

The false alarm rate depends in part on how frequently the user establishes new accounts and how frequently the user clears the browser history cache. With default settings, there are occasional spurious yellow lights while browsing, and sometimes

the first use of a legitimate site with user name and password input will trigger a false post warning. Many of the unnecessary warnings are the result of frame or redirection problems. If the user opens a new account, and intentionally uses the same password as another account, this will also produce an unwanted warning. However, second and subsequent visits (without clearing the history cache) do not lead to additional false alarms for this situation.

3.4 Conclusion

The only toolbar known to make no use of blacklists was Spoofguard. While it was able to identify the majority of phishing sites using only heuristics, it still missed some phishing sites and it had a very high false alarm rate. Spoofguard could potentially be improved through the use of a whitelist, which would prevent the problems that occurred when phishing sites were visited before their corresponding legitimate sites. The whitelist would not necessarily need to be extremely large or updated frequently to be effective.

4

PWDHASH

4.1 Introduction

Although techniques such as SSL/TLS with client-side certificates are well known in the security research community, most commercial web sites rely on a relatively weak form of password authentication: the browser simply sends a user's plaintext password to a remote web server, often using SSL. Even when used over an encrypted connection, this form of password authentication is vulnerable to attack. In *phishing scams*, an attacker sets up a web site that masquerades as a legitimate site. By tricking a user, the phishing site obtains the user's cleartext password for the legitimate site. Phishing has proven surprisingly effective at stealing user passwords, as documented in reports from the anti-phishing working group [APW]. In *common password attacks*, hackers exploit the fact that web users often use the same password at many different sites. This allows hackers to break into a low security site that simply stores username/passwords in the clear and use the retrieved passwords at a high security site, such as a bank. This attack, which requires little work, can lead to the theft of thousands of banking passwords. While password authentication could be abandoned in favor of hardware tokens or client certificates, both options are difficult to adopt because of the cost and inconvenience of hardware tokens and the overhead of managing client certificates.

Password cracking is the process of recovering passwords from data that has been stored in or transmitted by a computer system. A common approach is to repeatedly try guesses for the password. The purpose of password cracking is to gain unauthorized access to a remote system. Different forms of password cracking are as follows.

Brute force attack is a strategy that can in theory be used against any encrypted data by an attacker who is unable to take advantage of any weakness in an encryption system that would otherwise make his/her task easier. It involves systematically checking all possible keys until the correct key is found. There is no solution to

prevent from brute force attack. Only thing we can do is to extend time to get original password. It can be done by generating difficult passwords having lower and upper case letters, numbers and symbols.

Guess attack is also a trial and error strategy where the attacker tries to figure out the password using users' personal information as it is obvious that most of the user tend to choose such passwords that they can remember.

Dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by searching likely possibilities. A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary (from a pre-arranged list of values). In contrast with a brute force attack, where a large proportion key space is searched systematically, a dictionary attack tries only those possibilities which are most likely to succeed, typically derived from a list of words for example a dictionary (hence the phrase dictionary attack) or a bible etc.

By providing customized passwords, preferably over SSL, the threat of password attacks with *no server changes* and *little or no change to the user experience* can be reduced. Since the users who fall victim to many common attacks are technically unsophisticated, PwdHash[10, 11] is designed to transparently provide novice users with the benefits of password practices that are otherwise only feasible for security experts. We have experimented with Internet Explorer and Mozilla Firefox implementations and report the result of initial user studies.

4.2 Approach

In essence, this password hashing method is extremely simple: rather than send the user's cleartext password to a remote site, it sends a hash value derived from the user's password, *pwd*, and the site domain name. Specifically, PwdHash captures all user input to a password field and sends *hash(pwd,dom)* to the remote site, where *dom* is derived from the domain name of the remote site. *dom* is referred to the salt. The hash is implemented using a Pseudo Random Function keyed by the password. Since the hash output is tailored to meet server password requirements, the resulting hashed password is handled normally at the server; no server modifications are required. This technique deters password phishing since the password received at a phishing site is not useful at any other domain. The cryptographic hash makes it difficult to compute *hash(pwd,dom2)* from *hash(pwd,dom1)* for any domain *dom2* distinct from *dom1*. For

the same reason, passwords gathered by breaking into a low security site are not useful at any other site, thus protecting financial institutions from sites with lax security (e.g. those coordinating high school reunions).

The main idea of password hashing, which is attractively simple, has been explored in previous projects. The focus is on the implementation of password hashing as a secure and transparent extension (i.e. plug-in) to modern browsers. Password hashing is a seductively simple concept in theory that is surprisingly challenging to implement in practice, both technically and in terms of the user experience. First, password hashing alone is not a sufficient deterrent against phishing due to the considerable power afforded to web developers in modern browsers. For example, JavaScript on phishing pages could potentially intercept the user's cleartext password before it is hashed, whether it is typed in by the user or pasted from the clipboard. Since these types of interactions will also raise problems for a range of other possible browser extension projects. And second, simple ideas do not necessarily translate into simple user experiences. For example, the extension must recognize which user input to hash. If a user wishes to start using this extension, for example, she will have to visit the change-password page for her existing accounts and indicate to the extension to hash the new password she types in, but not the old. This is a new and potentially jarring step for novice users, but the extension cannot simply hash both password entries.

PwdHash follows a new mechanism for its activation when the user is about to enter the password. PwdHash can then take steps to protect the password as it is being entered. There are two closely related ways to do this. We call the first method *password-prefix* and the second *password-key*.

Password-prefix is an elegantly unobtrusive mechanism to defend against the JavaScript attacks. Users are asked to prefix their passwords with a short, publicly known sequence of printable characters. PwdHash monitors the entire key stream and takes protective action when it detects the password-prefix sequence. The extension has two modes: normal mode and password mode. The extension monitors all keyboard events. In normal mode, it passes all keyboard events to the page as it is. When the password-prefix (@@) is detected in the key stream, the extension switches to password mode and does the following: (1) it internally records all subsequent key presses, and (2) it replaces the user's keystrokes with a fixed sequence and passes the resulting events to the browser. This translation continues until focus leaves the password field, at which point the extension reverts back to normal mode. In other

words, all keystrokes entered following the password-prefix are hidden from the browser and from scripts running inside the browser until focus leaves the field.

Hashing can take place at one of two times. The first option is to replace the contents of the field with the hashed password when focus leaves the field. The second option is to trap the form submission event (called 'BeforeNavigate2' in Internet Explorer) and then replace the contents of all password fields with the appropriate hashed passwords. The first option is more jarring to the user, because his password could potentially change length immediately after entering it (once it gets hashed). However, it allows the extension to work automatically at sites like *yahoo.com* that implement their own password hashing algorithm using JavaScript on their login pages. PwdHash has implementations of both options.

Finally, if the password-prefix is ever detected while focus is not on a password field, our browser extension reminds the user not to enter a password.

Password-key is an alternative to the password prefix mechanism. Instead of using a printable sequence (@@) the idea is to use a dedicated keyboard key called a "password-key." Users are asked to press the password-key just before entering a password. The future keyboards might have a dedicated key marked "password," but for now the 'F2' key is used, which is not currently used by Internet Explorer, Firefox, or Opera.

The semantics of the password-key inside our extension are very similar to the password-prefix. When the user presses the password-key the extension enters password mode as described previously. All subsequent keystrokes are hidden from the browser and scripts running within the browser. The extension returns to normal mode when focus leaves the field. If the password-key is pressed while focus is not in a password field, the user is warned not to enter a password. The password-key, however, is less prone to mistake: whereas the password-prefix could appear naturally in the keystream and trigger undesired protection, password-key protection can only be initiated in response to decisive action by the user.

With respect to user experience, however, a password-key seems inferior to a password-prefix. First, novice users need to know to press the password-key when entering their password, but not to press the key when entering a PIN. While the prefix mechanism also demands a special attention to passwords, it may be easier to teach users that "all secure passwords begin with (@@)" than asking them to remember to press a certain key before entering a password. Second, upon resetting

their password at a password reset page just after installing PwdHash users need to know to press the password-key for their new password, but not to press the key for their old password.

4.3 Disadvantages

a) Invisible to user - Password hashing done by Pwdhash is invisible to user. If this extension stops working, user will not know about this, i.e., passwords will not be hashed.

b) Visibility of activation to webpage **-** Webpage gets the intimation about the activation of Pwdhash. This made Pwdhash vulnerable for JavaScript attacks. So webpage can put some efforts to know the original master password.

c) Password availability as plain text – The master password is directly filled in password field given by webpage. i.e., password is available in plain text.

d) Easily spoof-able – As activation is visible to webpage and by using Alex's corner method [22] it is very easy to know the master password of user by fake webpage.

e) Affect on others / Affecting webpage - Pwdhash have some side-effects on websites. Any JavaScript attached with password fields will not work properly. For ex. keyPress event will not work properly.

4.4 Conclusion

All the disadvantages that are mentioned above showcase the vulnerabilities of PwdHash. To overcome these disadvantages and to increase the potential of Pwdhash, we have made necessary enhancements and named it Pwdhash++.

5 PROPOSED SOLUTION

5.1 The First Approach

Most of the techniques for phishing detection are based on blacklist and/or combination of heuristics. In the blacklist approaches, when the user visits a website that is in the blacklist, he/she will be warned. But maintaining a blacklist requires a great deal of resources for exposure and confirmation of the deceptive websites.

Spoofguard computes a score for each web page based on the weight sum of results of each set of heuristics. If the score crosses certain threshold, the toolbar displays a red icon, warning the user. If some of the heuristics provoke but not enough to exceed the threshold, the icon turns yellow. If none of the heuristics provoke, the icon turns green to indicate a safe site.

The major concern about the Spoofguard is the false positive rate. The problem comes here when a fraudulent site is being visited before a legitimate site. In this case, the visit in the legitimate site causes Spoofguard to display a red signal. Considering a particular case in which only finite list of websites must be allowed, and then a white-listing approach does take precedence. Our main aim is to catch fraudulent banking sites and alert the user. A white-listing approach [18, 19] for anti phishing embedded in Spoofguard might solve our purpose. We can easily maintain a list of legitimate banking sites which is very limited in number.

A white-list or approved list is a list or register of entities that, for one reason or other, are being provided a particular privilege, service, mobility, access or recognition. A white-list is the best way to validate input. One will know exactly what desire is and that there is not any wrong types accepted. When compared to blacklisting method, white-listing data is short and more precise. White-list can contain data as per the comforts of the user, if the user is specific about the kind of sites he wishes to visit. On the other hand, blacklist needs dynamic update of list in order to warn the user about the site visited. We implemented white-listing concept. We used *Levenshtein*

edit distance algorithm [20, 21] to compare the user selected URL with white-listed URLs.

5.1.1 Levenshtein Distance

The *Levenshtein distance* is a metric for measuring the amount of difference between two sequences (i.e. an edit distance). The term *edit distance* is often used to refer specifically to Levenshtein distance.

The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character.

Computing Levenshtein Distance:

Algorithm

Step	*Description*
1.	Set n to be the length of s. Set m to be the length of t. If n = 0, return m and exit. If m = 0, return n and exit. Construct a matrix containing 0..m rows and 0..n columns.
2.	Initialize the first row to 0..n. Initialize the first column to 0..m.
3.	Examine each character of s (i from 1 to n).
4.	Examine each character of t (j from 1 to m).
5.	If s[i] equals t[j], the cost is 0. If s[i] doesn't equal t[j], the cost is 1.
6.	Set cell d[i,j] of the matrix equal to the minimum of: a. The cell immediately above plus 1: d[i-1,j] + 1. b. The cell immediately to the left plus 1: d[i,j-1] + 1. c. The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost.
7.	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d[n,m].

Table 1: Steps of Levenshtein edit distance algorithm

A commonly-used bottom-up dynamic programming algorithm for computing the Levenshtein distance involves the use of an $(n + 1) \times (m + 1)$ matrix, where n and m are the lengths of the two strings. This algorithm is based on the Wagner-Fisher algorithm for edit distance.

5.1.2 Algorithm

a) Step 1

The URL of the site entered by the user is taken as input string. This string is edited such that the host name of the site is retained and the path for the resource is eliminated.

b) Step2

The edited URL is compared with all the URL's present in the whit list. The comparison is carried out by using Levenshtein edit distance algorithm.

For example, here is the Levenshtein edit distance between two strings "*icicibank*" and "*iciciibank*". The first string is the host name of the genuine ICICI bank site and the other is of a phishing site.

		i	**c**	**i**	**c**	**i**	**b**	**a**	**n**	**k**
	0	1	2	3	4	5	6	7	8	9
i	1	0	1	2	3	4	5	6	7	8
c	2	1	0	1	2	3	4	5	6	7
i	3	2	2	0	1	2	3	4	5	6
c	4	3	3	1	0	1	2	3	4	5
i	5	4	4	2	1	0	1	2	3	4
i	6	5	5	3	2	1	1	2	3	4
b	7	6	6	4	3	2	1	2	3	4
a	8	7	7	5	4	3	2	1	2	3
n	9	8	8	6	5	4	3	2	1	2
k	10	9	9	7	6	5	4	3	2	**1**

Here, is the *Levenshtein edit distance between* two strings **"icicibank"** and **"iciciibank"**. The first string is the host name of the genuine ICICI Bank site and the other is of a phishing site

Levenshtein edit distance

Figure 14: Example of Levenshtein distance matrix

c) Step 3

The minimum of all the edit distances calculated is taken and compared with the threshold value. The threshold value is given such that the banking sites in the white list are distinguished from the non-banking sites i.e, if the minimum edit distance

value is greater than the threshold value then it is not a banking site. If that value is less than threshold then the URL will go for IP check, the next step.

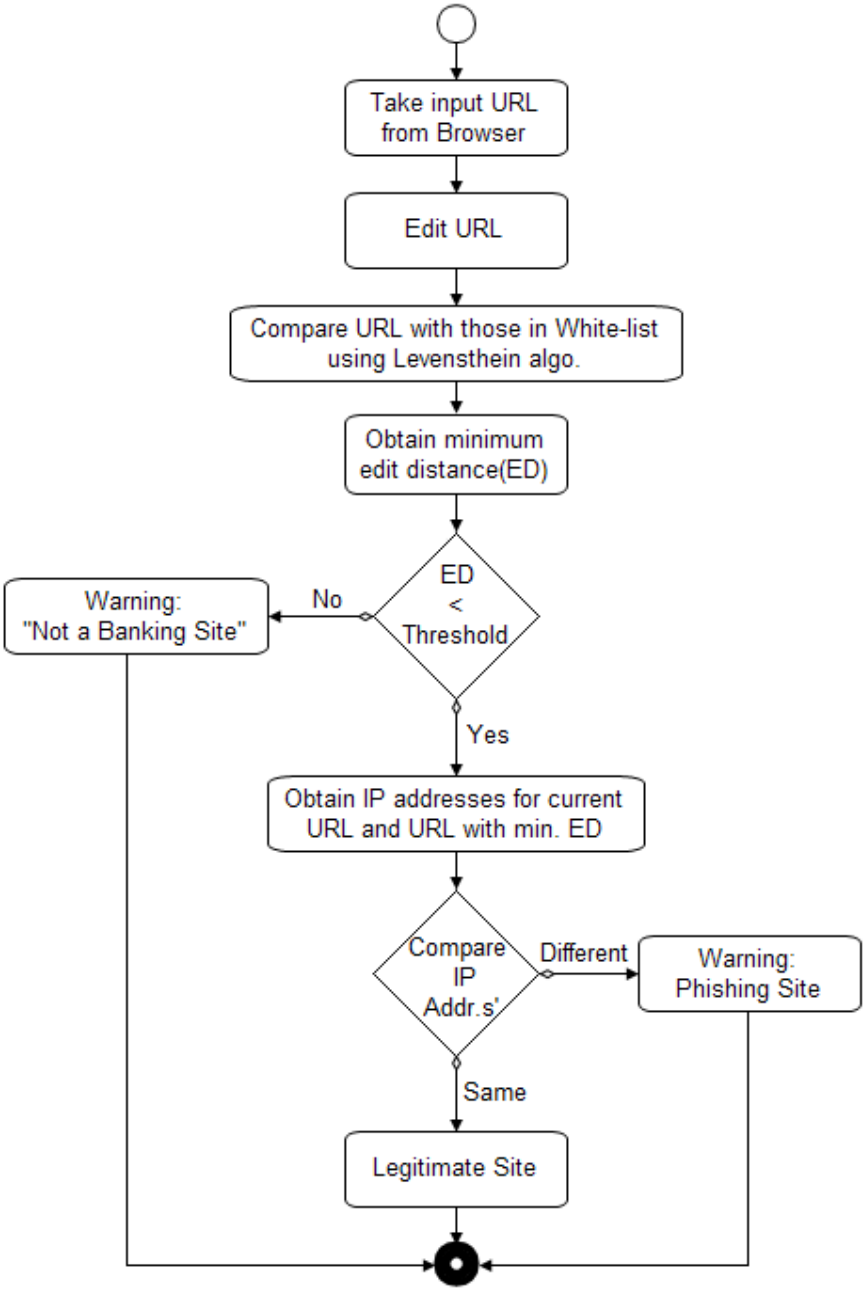

Figure 15: Flowchart for the first approach

Step 4

As the phishing sites uses the host name which is very near to the legitimate site, the edit distance value will be low. So the IP address of the entered site is compared with the IP address of the site in the white list which encountered the minimum edit

distance. If both the addresses are same, then it is a legitimate. If not, it is a phishing site. In this way the user is alerted.

A phishing site can be caught at step 2, if its domain name largely different from the original one or at step 3, if its domain name is close to the legitimate one. The phishing sites with look-alike domain names are caught at step 3.

5.2 The Second Approach

The second approach is a browser extension called Pwdhash++, an enhanced Pwdhash. When the user hits a special prefix (@@) or F2 key at the password field on a web page, this browser extension gets activated. The pop-up window contains identity key, password field, submit and cancel button with a background image.

At the time of installation, Pwdhash++ asks the user to enter an identity key. This identity key is displayed on the password prompt window whenever Pwdhash++ is activated. This proves genuineness of that prompt window.

The background image is user specific. The user assigns an image for a website which is regularly used. A database is maintained at the user side with two fields, URLs' in the white-list and its corresponding image. So, whenever the user enters a website and activates Pwdhash++, then its corresponding image is displayed in the background of the prompt window. If no image is displayed it indicates that no image is specified by the user for that site. So, when the user enters into a fraudulent site he can recognize the site as a phish site as no image was assigned to the fake site.

To address the single password problem and to safeguard user credentials from phishing attacks, Pwdhash++ transparently hashes the password with one-way mechanism and generates a different strong password.

5.2.1 Algorithm

a) Step 1

When Pwdhash++ is activated, it checks the conditions whether the focus is on password field and the field is empty. If these conditions does not satisfy then the user is warned about it.

b) Step 2

If the above conditions satisfy, then a prompt window to enter password pops up. This window contains identity key, which is given by the user at the time of installation,

password field and a background image, which is site specific custom image. If no image is assigned for that site then no background image is displayed on the pop-up windows.

c) Step 3

If the identity key or background image is not same as the one user entered before, user can understand the trick. Preferably, the user stops interacting with the site.

d) Step 4

If the indication is fine, then the user enters his/her master password into the password field. This password is hashed with domain name of the site and entered at the password field on the webpage.

Here, step 3 is completely user driven. The user has to observe the prompt window and recognize the identity key and the background image. If the identity key is not the one he entered at the time of installation, then it is clear that this pop-up window is a trick. If the background image is not the one that the user assigned for that site, then the user can understand it to be a phishing site.

Figure 16: Pwdhash++ pop-up window.

Steps 1 through 4 take place, when Pwdhash++ is activated by hitting "@@" key. For user convenience, F2 key directly activates the Pwdhash++ by shifting the focus to password field and also emptying it automatically. So, step 1 and step 2 in the above algorithm will be skipped when Pwdhash++ is activated using F2 key. whole algorithm is explained in the following.

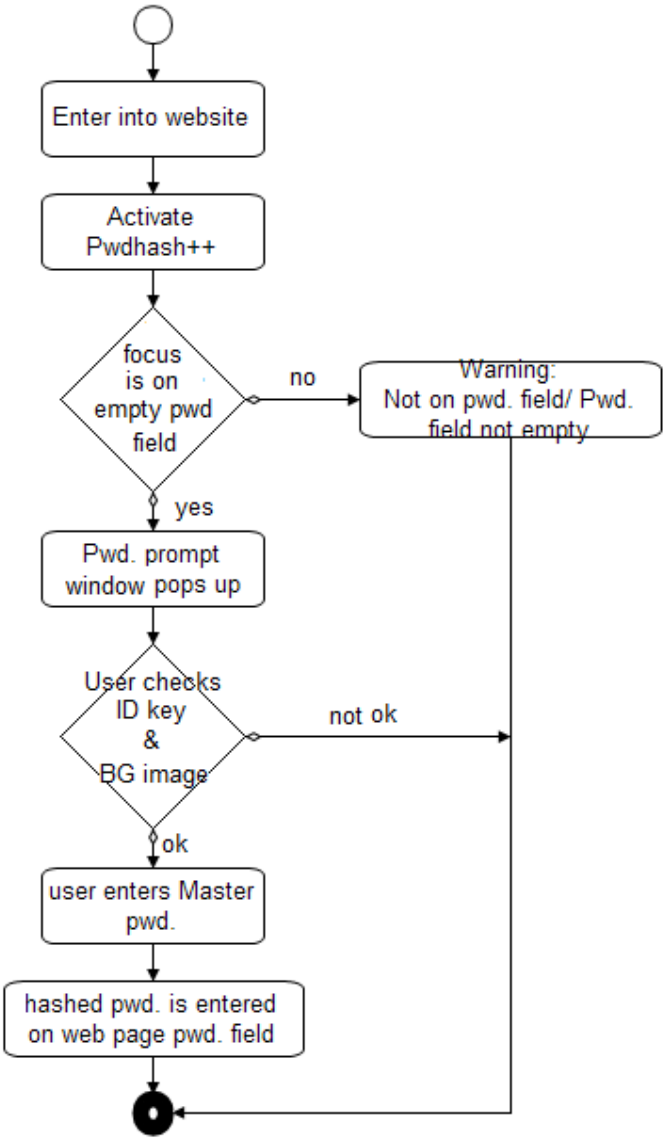

Figure 16: Flowchart for the first approach

5.2.2 Added Advantages

- *Lightweight browser extension* – Very light browser extension containing only JavaScript.
- *Easy to handle* – Completely depends on user whether to use this tool or not
- *Best hashing algorithm* – Hashing algorithm used by Pwdhash++ is *MD5*, which is non-reversible.
- *High password strength* – As password is obtained by MD5 algorithm containing alphabets (lower as well as capital), numbers (0-9), and some special characters generally accepted by every site. Due to this reason, the

password strength is very high**.** It is difficult to remember or attack (dictionary attack, brute-force attack, or any other attack) it.

- *Different password for every site* – For different site different password is generated from same master password. It is possible because Pwdhash++ using *domain name* as the key for encryption of password.
- *Prevent phishing* – Pwdhash++ generates different password for different sites as they have different domain names. Phishing website have different domain name from that of original one. So, phishing website will only know the hashed password, which is useless .
- *Invisible to server* – The whole procedure is done on client's machine only. No interference of server.
- *Invisibility of activation to webpage* – Activation procedure of Pwdhash++ is still same as of Pwdhash, but Pwdhash++ never let the webpage to know about the activation. So any type simple or complex JavaScript cannot steal master password.
- *Visible to users* – As Pwdhash++ asks for password using password prompt popup and then fill password field in site automatically with hashed password. It makes user to be sure about password hashing.
- *Spoofing proof* – Upon activation of Pwdhash++ a prompt window pops, which contains custom site specific image and unique identity key, asking for password. Entered password will be hashed and then sent to webpage.
- *Unavailability of password as plain text* – Password is filled on password field of pop-up prompt window. Password is then converted to hash and then sent to the webpage i.e., password is available only in hashed format.
- *No effect to webpage* – Hashed password will sent to the webpage character-by-character. So webpage differentiate the password entrance in password field, whether it is entered by human or by Pwdhash++.
- *Impedes password theft* – As mentioned above, the theft of master password is made tough.

6 CONCLUSION

We looked at various solutions proposed in literature to mitigate phishing and we presented a new one. We have discussed about the disadvantages of Spoofguard and Pwdhash and enhanced them. The efficiency of Spoofguard is increased, especially regarding false positive rate, with the implementation of white-listing added to it. The efficiency of Pwdhash also increased to withstand Javascript attack. Both the approaches complement each other desirably to protect the user from phishing attack.

7 FUTURE SCOPE

The first approach of the proposed solution can be embedded in Spoofguard which can tackle with most of the flaws with the previous toolbars available. There is a large scope of improvement since not much work has been done on the concept of white-listing till now.

Also we can introduce a keyword method in our algorithm. In this keyword method improvisation every banking site would be given a unique keyword to embed in their domain name. So after proper advertisement of this characteristic into the public, every phishing site would be forced to use this keyword into their domain name, which in turn would be quite easy for SpoofGuard to catch any phishing site.

The second approach is an enhancement for Pwdhash that works in Mozilla firefox browser. This approach need to be extended to other browsers like Internet Explorer, Opera etc. being a light weight browser extension, this approach can further extended to mobile browsers too.

REFERENCES

[1] Identity Theft: What to Do if It Happens to You. http://www.anti-phishing.org/consumer_recs2.htm.

[2] Anti-Phishing Working Group. http://www.antiphishing.org/.

[3] Tan, Koontorm Center. *Phishing and Spamming via IM (SPIM).* http://isc.sans.org/diary.php?storyid=1905.

[4] Microsoft Corporation. *What is social engineering?* http://www.microsoft.com/protect/yourself/phishing/engineering.mspx.

[5] Josang, Audun et al.. Security Usability Principles for Vulnerability Analysis and Risk Assessment. In *Proceedings of the Annual Computer Security Applications Conference,* 2007. http://www.unik.no/people/josang/papers/JAGAM2007-ACSAC.pdf, 2007.

[6] Phishing, n. OED Online, March 2006, Oxford University Press. Oxford English Dictionary Online. http://dictionary.oed.com/cgi/entry/30004304/.

[7] Phishing. *Language Log*, September 22, 2004. http://itre.cis.upenn.edu/~myl/languagelog/archives/001477.html

[8] Aaron Emigh and John Mitchell, *Anti-Phishing Technology*. Report of the US Secret Service San Francisco Electronic Crimes Task Force, January 19, 2005.

[9] Aaron Emigh, *Online Identity Theft: Phishing Technology, Chokepoints and counter measures*. ITTC Report, October 3, 2005

[10] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C.Mitchell. A Browser Plug-In Solution to the Unique Password Problem. http://crypto.stanford.edu/Pwdhash/, 2005.

[11] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C.Mitchell. Stronger Password Authentication Using Browser Extensions. In *14th Usenix Security Symposium*, 2005.

[12] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. Mitchell. Client-side defense against web-based identity theft. In *11th Annual Network and Distributed System Security Symposium (NDSS '04), San Diego*, 2005.

[13] Spoofguard. Client-side defense against web-based identity theft. http://crypto.stanford.edu/Spoofguard/, 2005.

[14] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security, New York, NY*, pages 77–88. ACM Press, 2005.

[15] E. Kirda and C. Kruegel. Protecting Users Against Phishing Attacks with AntiPhish. In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 1*, pages 517–524, Washington, DC, USA, 2005. IEEE Computer Society.

[16] E. Kirda and C. Kruegel. Protecting Users against Phishing Attacks. *The Computer Journal*, 2006.

[17] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz (1998). A Bayesian Approach to Filtering Junk e-mail. In *AAAI'98 Workshop on Learning for Text Categorization*. http://robotics.stanford.edu/users/sahami/papers-dir/spam.pdf.

[18] Ye Cao, Weili Han and Yueran Le. Anti-phishing Based on Automated Individual White-List. In *DIM'08, October 31, 2008, Fairfax, Virginia, USA.*

[19] Chinmay Soman, Hrishikesh Pathak, Vishal Shah, Aniket Padhye, and Amey Inamdar. An Intelligent System for Phish Detection, using Dynamic Analysis and Template Matching. In *World Academy of Science, Engineering and Technology 42 2008*

[20] Levenstein, A., Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710.

[21] Michael Gilleland. Levenshtein Distance. http://www.merriampark.com/ld.htm#REFS.

[22] Alex. Attacking the Pwdhash Firefox Extension. http://kuza55.blogspot.com/2007/02/attacking-pwdhash-firefox-extension.html

[23] Lorrie Cranor, Serge Egelman, Jason Hong, and Yue Zhang. Phinding Phish: An Evaluation of Anti-Phishing Toolbars. In *CyLab Carnegie Mellon University Pittsburgh, PA 15213, November 13, 2006.*

[24] NetCraft, Netcraft Anti-Phishng Toolbar. http://toolbar.netcraft.com/.

[25] Google Safe Browsing for Firefox. http://www.google.com/tools/firefox/safebrowsing.

[26] EarthLink Tool. http://www.earthlink.net/software/free/toolbar/.

[27] GeoTrust, Inc. TrustWatch Tool. http://toolbar.trustwatch.com/tour/v3ie/toolbar-v3ie-touroverview.html.

[28] eBay Toolbar's Account Guard.http://pages.ebay.com/help/confidence/account-guard.html.

[29] McAfee, Inc. McAfee SiteAdvisor. http://www.siteadvisor.com/.

[30] Microsoft's Approach to Anti-Phishing. http://www.microsoft.com/mscorp/safety/technologies/antiphishing/vision.mspx

[31] Netscape Communications Corp. "Security Center". http://browser.netscape.com/ns8/product/security.jsp.

[32] Password Composer. http://www.xs4all.nl/~jlpoutre/BoT/Javascript/PasswordComposer/.

[33] Arantius**.** Password Generator. http://trac.arantius.com/wiki/Extensions/MagicPasswordGenerator.

[34] Password generator. http://www.angel.net/~nic/passwdlet.html.

[35] Hassapass. http://www.hashapass.com/.

[36] Genpass. http://supergenpass.com/genpass/.

[37] Password Hasher. http://wijjo.com/PasswordHasher.