2005-06-01

Phishing Warden: Enhancing Content-Triggered Trust Negotiation to Prevent Phishing Attacks

James Presley Henshaw
Brigham Young University - Provo

PHISHING WARDEN: ENHANCING CONTENT-TRIGGERED TRUST NEGOTIATION TO PREVENT PHISHING ATTACKS

by

James Presley Henshaw

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

May 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

James Presley Henshaw

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date Dr. Kent E. Seamons, Chair

Date Dr. Charles Knutson

Date Dr. Robert Burton

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of James Presley Henshaw in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Dr. Kent E. Seamons Chair, Graduate Committee

Accepted for the Department

Dr. David Embley
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean, College of Physical and
Mathematical Sciences

ABSTRACT

PHISHING WARDEN: ENHANCING CONTENT-TRIGGERED TRUST NEGOTIATION TO PREVENT PHISHING ATTACKS

James Presley Henshaw

Department of Computer Science

Master of Science

Phishing attacks are spam e-mails that attempt to fool recipients into divulging their identifying information by posing as a message from a well known company and using that company's branding and logos. It is estimated that phishing attacks have cost bank and credit card customers $1.2 billion in the U.S. in 2003. Previous work, content-triggered trust negotiation (CTTN), filters Internet traffic for sensitive data, and prevents a user from disclosing sensitive information to an un-trusted server. However, existing CTTN implementations are vulnerable to client-side scripts that obfuscate any data the client's browser sends to the web server in order to bypass CTTN's filter.

To increase the security of CTTN, this thesis introduces Phishing Warden, a browser-plug-in that filters content before client-side scripts can execute, thereby preventing the scripts from obfuscating data in order to bypass the filter. Phishing Warden negotiates the release of sensitive data through web forms via the AutoFill button. After Phishing Warden determines the web server is trustworthy of the requested information, the sensitive data is automatically inserted into the form, indirectly informing the user that Phishing Warden trusts the server with this information

Besides potentially obfuscating data, scripts in Internet browsers can exploit security vulnerabilities which allow malicious scripts to potentially take over the computer, or deceive the user with a fake toolbar [31]. In addition to preventing data obfuscation by client-side scripts, Phishing Warden also allows a user to customize script control with the push of a button, letting the user decide which websites to trust enough to run scripts. Phishing Warden extends CTTN to remember past sites deemed trustworthy by the user.

ACKNOWLEDGEMENTS

I would like to thank all the people that have helped me with this thesis. Namely, Dr. Kent Seamons for his mentoring and wonderful inspirations, the members of the ISRL for their reviews and feedback, and the university for providing me with my education. I would also like to personally thank Tim van der Horst for his help in implementing Phishing Warden. I am grateful for my father and brother for motivating me to get my master’s degree. Last and most important of all, I especially thank my wife, Joy, for her love and support.

This research was supported by funding from Zone Labs, Inc., DARPA through SSC-SD grant number N66001-01-1-8908, the National Science Foundation under grant no. CCR-0325951 and prime cooperative agreement no. IIS-0331707, and The Regents of the University of California.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Phishing attacks are one of the newest, fastest growing, and perhaps costliest forms of attack on the Internet. They are performed through spoofing (impersonating) email addresses to lure unsuspecting individuals into cleverly replicated websites that trick users into revealing their sensitive information. The cost of damages due to phishing attacks targeting patrons of banks and credit card companies in the U.S. is estimated at $1.2 billion for 2003. Between the months of August 2003 and April 2004, the number of phishing attack e-mails increased by a factor of 10, to 3.1 billion per month [21].

The following example illustrates a typical phishing attack:

> *Bob checks his e-mail account from his home computer using his favorite e-mail viewer. He notices an e-mail marked "urgent" from "eBay Accounts" with the subject "Account verification, please follow instructions so your eBay account will not be disabled." Bob nervously opens the e-mail to discover a message stating that eBay is purging their database of unused accounts and telling him to follow this link to eBay to verify his account information so his account will not be disabled. Bob follows the link and sees the familiar eBay logo with a form that requires him to fill in his identifying information including: eBay user name, eBay password, full name, address, social security number, credit card information, and PIN number. Bob fills in the information and submits it, thinking he has successfully verified his account to eBay, when in reality he has become the latest victim of a phishing attack.*

Because an attacker can easily duplicate a web site using copies of company logos and adopting a similar website template, it can be difficult for a user to visually distinguish between legitimate and fraudulent websites. Phishing attacks exploit this vulnerability by luring users to fraudulent sites with the express purpose of coercing the user into divulging sensitive information. Usually, these attacks are instigated through forged e-mails. Another way of luring individuals to cloned websites, called typo pirating, is by slightly changing the domain name of a website from a name like paypal.com to paypa1.com. Several proposals have been made to help internet surfers detect duplicate web pages and prevent the submission of sensitive information to untrustworthy sites [1].

One way to thwart a phishing attack is to prevent the fraudulent e-mail from being received and believed. At the heart of the problem is the lack of authentication in the e-mail transport protocol, SMTP. To overcome this shortcoming, several researchers have proposed to introduce e-mail authentication through digital signatures [25] [26]. The idea is simple enough: Don’t trust what is not signed. E-mail clients can detect spoofed e-mail addresses by verifying the digital signature using the known public key of the sender, and displaying a visual icon to notify the user that the signature is valid. Users are less likely to visit links in an e-mail that does not contain a valid digital signature.

One of the latest phishing solutions prevents a user from visiting known phishing sites by creating a black list. Every time a browser opens a new page, the program checks to see if the page is on the list of known phishing sites. If the site is on the list, the user is alerted and prevented from submitting data to the website, thus reducing the risk of identity theft. EarthLink is one of the first companies to provide this program free for download [11]. While this solution is effective for users fortunate enough to go to phishing sites on the black list, those who visit phishing sites too new for the blacklist may still fall victim to identity theft. The black-list approach is limited due to the short lived nature of phishing sites which are usually only available for a few hours or days [6].

Another approach to thwarting a phishing attack is to prevent the user from leaking sensitive information to a phishing server. Two solutions that adopt this approach are SpoofGuard [6] and content-triggered trust negotiation (CTTN) [16] [18].

SpoofGuard is an Internet Explorer (IE) plug-in developed at Stanford to thwart phishing attacks. SpoofGuard monitors all incoming web pages and computes a score, ranging from 0 to 1, for the URL, images, links, password field, domain name, referring page, and image-domain associations. The results of each computation are fed through a scoring function to create a composite score. A larger composite score denotes a higher risk of a phishing attack. According to the composite score, SpoofGuard's safety indicator changes from green (safe), to yellow (caution), to red (this is a phishing attack). The key to SpoofGuard phishing prevention is the monitoring of all information the browser posts for sensitive content and warning the user if the current web page has a non-green safety indicator [6].

Another solution to phishing attacks is based on trust negotiation [28]. Trust negotiation is an attribute-based authentication paradigm, unlike current identity-based, username/password schemes. Most identity-based systems map users to a role by verifying that they know the password associated with the username. In trust negotiation, the username and password are replaced with credentials (digital certificates signed by an issuing authority), such as a driver's license. With these credentials, parties are able to prove that they possess certain attributes. When a client requests a secure resource on a server, the server checks the policy governing the resource. The policy states the credentials for which the client must prove ownership in order for the resource to be released. The server may initiate trust negotiation with the client by sending a policy to be satisfied. Either party may consider their credentials to be sensitive, and first disclose a policy protecting the credential in order to determine whether they should trust the other party enough to disclose the sensitive credential. The process of negotiating trust continues with the bi-lateral exchange of credentials and policies until the initial policy that began the negotiation is satisfied or the negotiation fails.

Content-triggered trust negotiation extends trust negotiation to allow the client to initiate the negotiation to verify that the server can be trusted with sensitive information. A proxy is placed between the browser and the server to monitor all traffic being sent to the server by scanning the transmitted data for sensitive information. If any sensitive information is recognized, the corresponding policy for each sensitive piece is used in the generation of a conglomerate policy that is sent to the server. The proxy initiates trust negotiation to decide whether the server is trustworthy to receive the sensitive information. If the server proves trustworthy, then the proxy sends the sensitive data to the server and returns the response received from the server. If trust negotiation determines the server is not trustworthy, the user receives a warning message and is given the option to ignore the failed negotiation and send the sensitive content despite the possibility of an attack [16] [18].

Both content-triggered trust negotiation and SpoofGuard help raise the difficulty level of launching a successful phishing attack. However, both proposals rely on reading the post data after it is sent by the browser but before it is received by the server. This reliance makes both methods susceptible to client-side scripting that can obfuscate sensitive information before it is filtered. An adept phishing attacker could encrypt the user input before submitting the form, thus eluding the scans of both SpoofGuard and CTTN. Both previous works acknowledge this weakness and suggest turning off client-side scripting to prevent such an attack. While this would successfully prevent scripting obfuscation, it would also hinder web page functionality, driving users away from such a solution.

In order to maintain full functionality and prevent any scripts from obfuscating the post data, the sensitive data must be analyzed at a higher layer. One method to accomplish this is through a prevalent browser plug-in feature, AutoFill, which scans the web page for forms and fills in the requested input fields. Before filling in the input fields, each field can be checked for sensitivity.

The current implementation of content-triggered trust negotiation permits the user to submit information to be processed for trust negotiation and returns the status (success or failure) of the negotiation. If trust negotiation fails, the user can choose to trust the site and submit the data. The current implementation does not, however, retain the trusted site in memory and prompts the user on subsequent visits to verify whether the current site should be trusted.

Losing sensitive information is not the only worry of browser users. Since malicious sites are able to exploit security holes in browsers, surfers should be wary of the sites they visit. Client-side scripts pose a threat to browser security. CERT recommends that browser users turn scripts off while surfing [4]. Internet browser users seldom heed this warning because needed features of trusted websites would stop functioning. Some scripting attacks can allow an attacker to gain full control of the computer, while others try to deceive the user with a fake toolbar. Although most browsers let the user customize the browser scripting permissions, they do not provide a readily available interface for permitting scripts to run temporarily.

1.1 Thesis Statement

In order to prevent the loss of sensitive data in web form submissions to malicious client-side scripts that obfuscate form data, it is proposed that content-triggered trust negotiation be moved higher in the application layer. To accomplish this, a toolbar must be added to IE. This toolbar negotiates trust on the filling of web forms through the AutoFill feature, preceding any client-side scripting which could obfuscate form data. The toolbar also provides an interface to control the IE security zones, which are used to control the script execution privileges. Content-triggered trust negotiation must also be enhanced to remember human interaction in the trust-building process, allowing users to control the addition of new trusted sites. These additions to content-triggered trust negotiation, collectively integrated into a browser helper object called Phishing Warden, will assist Internet users in their fight against identity theft.

1.2 Motivating Scenario

The following scenario revisits the earlier example of Bob checking his e-mail and illustrates how a phishing attack is thwarted using Phishing Warden.

> *Bob checks his e-mail account from his home computer using his favorite e-mail viewer. He notices an e-mail marked "urgent" from "eBay Accounts" with the subject "Account verification, please follow instructions so your eBay account will not be disabled." Bob nervously opens the e-mail to discover a message stating that eBay is purging their database of unused accounts and telling him to follow this link to eBay to verify his account information so his account will not*

be disabled. Bob follows the link and sees the familiar eBay logo with a form that requires him to fill in his identifying information including: eBay user name, eBay password, full name, address, social security number, credit card information, and PIN number. Bob clicks Phishing Warden's AutoFill button which starts a trust negotiation for the sensitive information. Bob is prompted with a warning message stating that trust negotiation has failed and notices that none of the sensitive values are filled in by Phishing Warden. He thus assumes that the website must be fraudulent because his past dealings with eBay have succeeded using trust negotiation. No information is sent to the phishing website and Bob informs eBay of this latest phishing attack.

Chapter 2

Related Work

2.1 SpoofGuard

SpoofGuard is an IE plug-in designed to detect if a website exhibits the common characteristics of phishing sites. SpoofGuard does this by computing a score based on seven web page characteristics: the URL, images, links, password field, domain name, referring page, and image-domain associations [6]. The score ranges from 0 to 1 where a 0 indicates the safest web site score and a 1 indicates the web site is most likely to be a phishing attack.

The URL and links are checked to see if they contains the '@' character. While this was a common characteristic of phishing URLs when SpoofGuard was created, IE has since been updated to prevent users from following links with the '@' character, rendering this security check obsolete for users who run the cumulative security update for IE [20]. This check is still useful for users who have not updated IE.

Since most phishing sites use images directly from the websites they are spoofing, SpoofGuard checks all images against a database of frequently spoofed website images. The image test can also be fooled by splitting the image into segments that appear as one continuous image to the user, but are in fact separate image files.

SpoofGuard's referring page test detects whether the page the user is coming from is a known e-mail site, such as hotmail.com. Because phishing attacks rely on links embedded in the e-mail, the referring page test is of great help in detecting possible phishing attacks by scoring pages from e-mail sites higher.

The password check is one of the most valuable features of SpoofGuard. It protects users from sending passwords in plain text by checking to see that the user is using HTTPS and that the certificate is valid. SpoofGuard hashes each password and checks it against a database of hashed passwords. SpoofGuard checks all values submitted in the form to see if they are possibly a password. If SpoofGuard detects a password associated with another domain, then SpoofGuard increases the score for the web page to reflect a possible attack.

2.2 Trust Negotiation

Trust negotiation is a new authentication paradigm that leverages digital credentials and access control policies to negotiate trust. Digital credentials contain the attributes that a trusted third party asserts are true about another party. An example of a credential is a student credential, where a university (the trusted third party), asserts that the holder of the student credential is a student of the university. Access control policies can be satisfied by the exchange of these digital credentials. An example access control policy states that a customer must prove he is a student in order to receive a student discount. Trust agents act on behalf of the negotiating parties to automate the process of negotiating trust.

Use of the example student credential and access control policy is illustrated in the following scenario:

> *Johnny is a student of Brigham Young University (BYU) and has been issued a student credential from BYU. Johnny visits a local cinema online and requests a student discount on a movie ticket. The cinema’s website trust agent has an access control policy protecting the student discount which states that the requesting party must be a student to receive the discount. Trust negotiation*

commences as the website discloses the policy governing the student discount to Johnny's trust agent. His trust agent reads the policy and discovers that it can satisfy the policy by releasing Johnny's credential to the website, and does so. The website's trust agent asserts the received credential is valid and that the access control policy has been satisfied. The website grants Johnny the student discount.

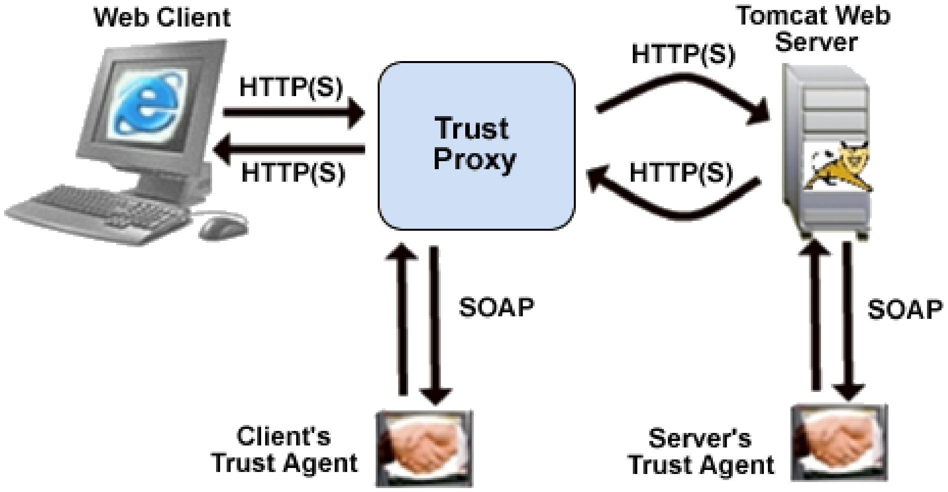

Figure 1: Architecture for Trust Negotiation

TrustBuilder, the current implementation of trust negotiation used by the Internet Security Research Lab (ISRL) at BYU, accepts digital credentials in the form of X.509v3 certificates. These X.509v3 certificates must be digitally signed by a trusted third party in order to be considered valid. Access control policies are implemented as XML policies in the TPL format, used by IBM's Trust Establishment (TE) system [15]. The policies are verified using TE's policy compliance checker. Policies and credentials are exchanged using the protocol on which the service is requested. TrustBuilder has been implemented to negotiate trust over HTTP, SSH, TLS [17], and SMTP [18].

2.3 Content-Triggered Trust Negotiation

Content-triggered trust negotiation allows the client to initiate the negotiation to verify that the server can be trusted with sensitive information. In order to discover sensitive information, content-triggered trust negotiation employs content analysis. Because client content is dynamic, it is impossible to create a resource-to-policy mapping for every possible combination of sensitive resources. To cope, content-triggered trust negotiation dynamically generates a policy each time sensitive information is filtered. Figure 2 illustrates the state diagram for dynamic client content access control.

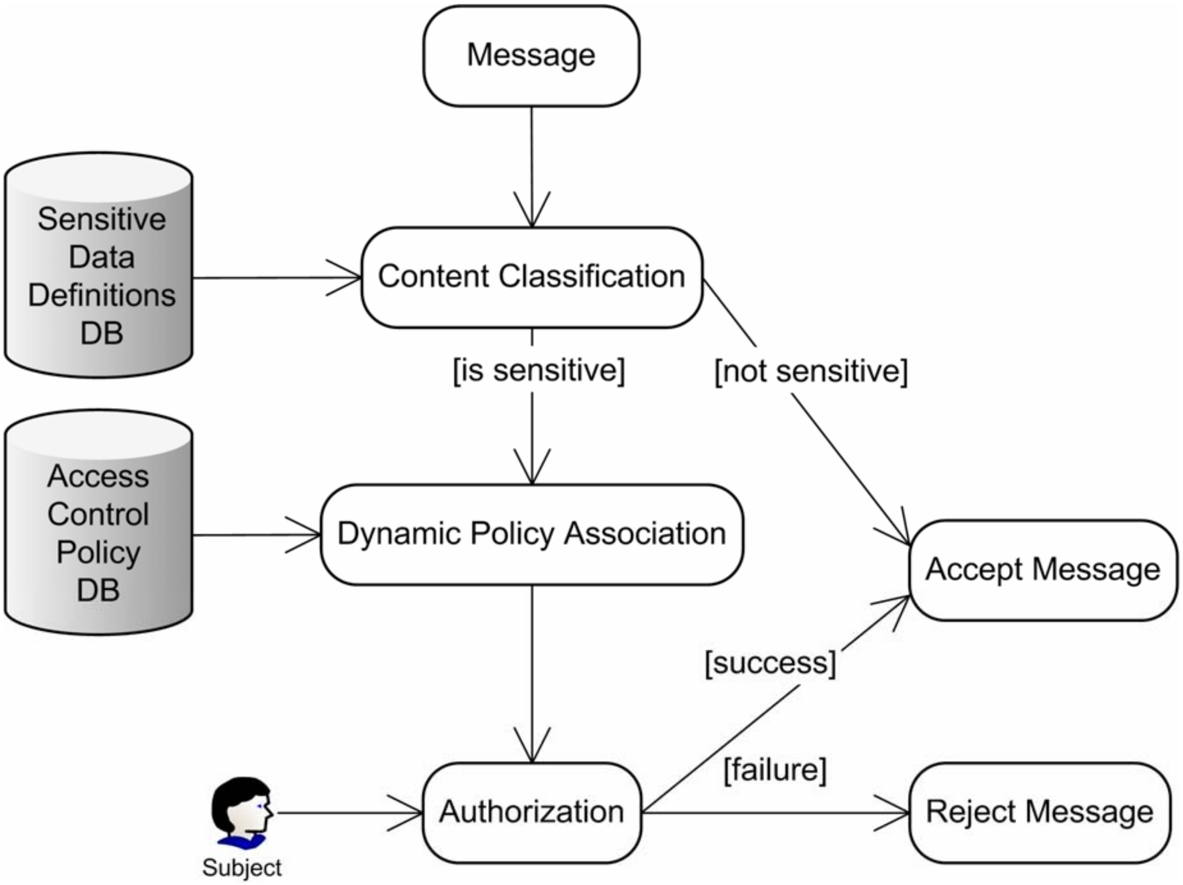

Figure 2: State diagram for dynamic client content access control system in content triggered trust negotiation

The first state that CTTN enters is the “Content Classification” state. Here CTTN uses document classification methods to determine the sensitivity of the content being transmitted. CTTN allows for a variety of content classification domains in order to support a wide range of classification techniques. To generalize the classification model, there are two parts to classification: the set of queries (*Q)* and the content (*m*). The classify formula (see Figure 3) explains how a set of content types (*T*) can be found by taking the union of each content type returned by the *sim* (filter function) for each of the individual queries (*q*) found in *Q*.

$$classify(m,Q) = \bigcup_{\forall q \in Q} sim(m,q) = T$$

Figure 3: Content-triggered trust negotiation’s content classification function

To illustrate this model, assume there is a user with a social security number that is considered sensitive. The user creates a query to detect the disclosure of this number. This query could be a simple query requiring pattern matching, or more complex algebraic, probabilistic, or machine learning algorithms. Each query is associated with a content type, which is then mapped to a policy. This enables CTTN to be easily managed, giving it a feel similar to role-based access control where multiple queries (users) fall under the same content type (group).

If sensitive content is found, CTTN either continues to the dynamic policy creation state, or discloses the non-sensitive content. CTTN contains a dynamic policy module which maintains a database of mappings between content types and access control policies. This module is used in the dynamic policy creation state to create a policy from all the policies governing each sensitive content type found in the submitted content.

The conjunction of each policy's role expression is used to form this "super" policy. The policy is then simplified using Boolean algebra rules; absorption [$a \wedge (a \vee b) = a \vee (a \wedge b) = a$] , distributive [$(a \wedge b) \vee (b \wedge c) = b \wedge (a \vee c)$], and idempotent [$a \wedge a = a$ and $a \vee a = a$]. The client's trust agent uses the final product of this simplification in trust negotiation [16] [18].

The filtering process of content-triggered trust negotiation takes place after the browser has posted that data to the web server and before the web server receives the posted data. A clever attacker may use a client-side scripting language such as JavaScript to encrypt any posted data before it is sent to the proxy in order to avoid CTTN's filter from detecting the sensitive data being posted. This would cause the *sim* function to return an empty set of content types resulting in the content being classified as "not sensitive". The server would then receive the sensitive information in encrypted form without having to negotiate trust. The server could easily decrypt the sensitive information using the proper decryption algorithm in order to access the sensitive data.

Chapter 3

Phishing Warden

This chapter discusses the design and implementation of Phishing Warden.

3.1 Design

The design of Phishing Warden includes three goals:

1. Prevent client-side scripting from obfuscating form data before filtering software can scan the form for sensitive information.
2. Enable users to more easily control the execution of client-side scripts for a given domain.
3. Remember each site that a user deems trustworthy and the information disclosed to each trusted site.

Because phishing attackers send e-mails with misleading URLs that reference a forged website, Phishing Warden is designed to assist users in safely visiting web sites by providing users with greater assurance that a website can be trusted with sensitive information such as a credit card number or a social security number. Phishing Warden leverages trust negotiation to establish confidence in web sites.

Trust negotiation uses access control policies to safeguard sensitive data. For example, a simple access control policy might specify that the role Better Business Bureau (BBB) Member must be satisfied before credit card information is released. The party requiring the credit card information can demonstrate they are a BBB member by sending their BBB credential. For trust negotiation to be effective, the access control policies governing the release of sensitive resources must be designed to eliminate the possibility of leaking sensitive information to an untrusted party. Phishing Warden assumes that users will not configure their own access control policies since it may be a complex and time consuming task. Instead, security experts are relied upon to provide the policies.

In order for Phishing Warden to be effective, it must be able to glean the specific information types the web form is asking for so that it can automatically fill in the web form accordingly. If the web form is created according to the Electronic Commerce Modeling Language (ECML) [12], a form field naming standard (see Appendix A for an example), then Phishing Warden has a systematic way of extracting the type of information required and determining whether trust negotiation is needed for that data. If the web form does not follow the ECML, then the AutoFill feature fails and no data is inserted into the web form.

Previous solutions that address phishing attacks require the disabling of client-side scripts in order to thwart attacks that obfuscate sensitive data. This is due to the power given scripts to manipulate any data being sent. Disabling client-side scripts hampers usability, as almost every website requires JavaScript (the most commonly used client-side scripting language) to navigate through and use the website. Phishing Warden must include the ability to successfully detect phishing attacks with client-side scripting enabled.

To address post data obfuscation while still allowing the execution of client-side scripts, the content analysis of sensitive information must occur before client-side scripting can obfuscate the data. There are many layers to network capable applications (see Figure 4). Phishing Warden must place this functionality at an appropriate position in the application layer that enables Phishing Warden to detect phishing attacks when client-side scripting is enabled by negotiating trust on sensitive content types requested in the form. This approach maintains usability by enabling client-side scripts while also ensuring the integrity of the filtering process.

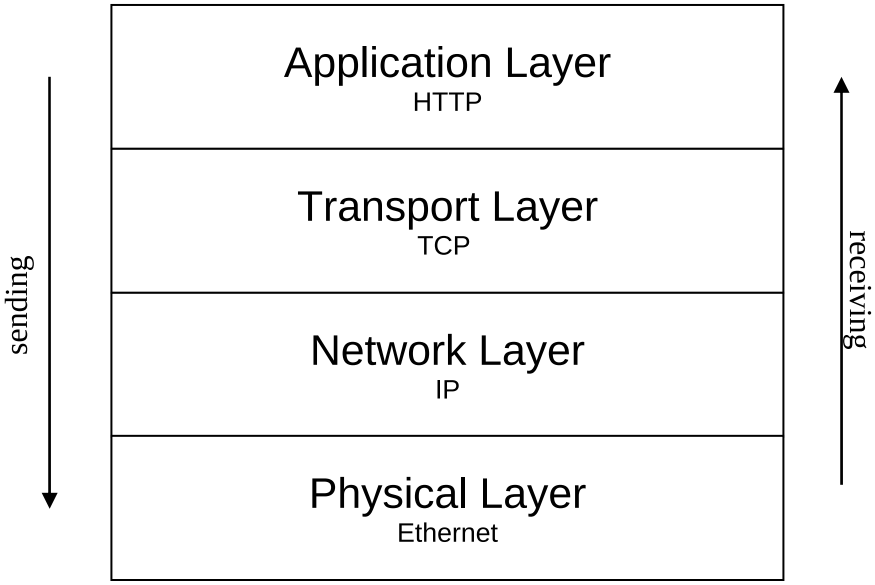

Figure 4: Flow of information in an Internet browser

Phishing Warden creates a buffer of protection between the Internet and the web client by utilizing IE's plug-in interface, the Browser Helper Object (BHO) [13]. The BHO allows Phishing Warden to monitor all HTML forms at the user's request, and verify that the server is trustworthy of each input field before submitting the data (see Figure 5). This act of verifying before submitting eliminates the threat of client-side obfuscation that scripts pose.

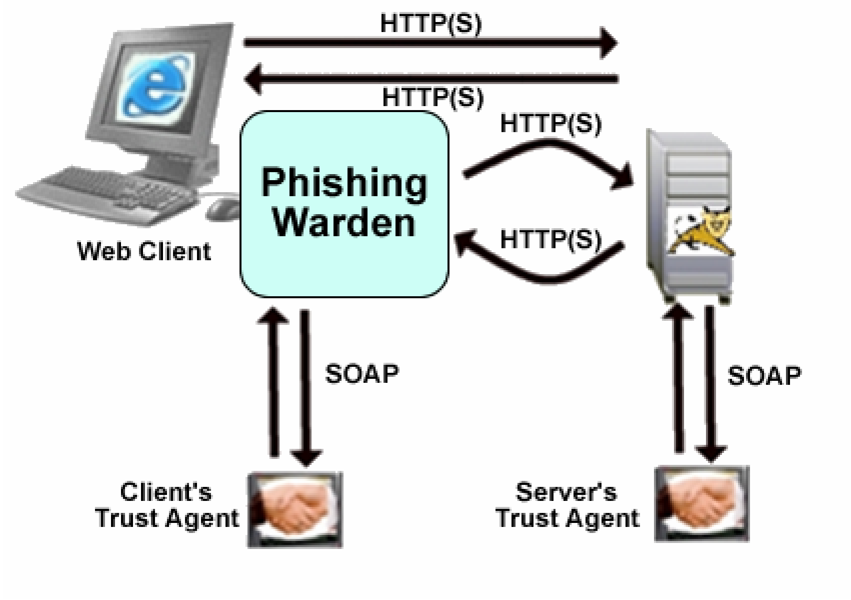

Figure 5: Model of trust negotiation without a proxy by using the browser plug-in Phishing Warden

As previously stated in the introduction, currently there is no means provided for users to easily adjust security settings for individual domains in IE. Controlling the browser's execution of client-side scripts is a tedious task requiring eight mouse clicks by the user via three dialogues to finally effectuate the desired change in security level. Currently, IE allows users to divide domains into separate security zones. Associated with each security zone is a security level which sets the permission for a site to run scripts and download and execute programs with or without prompts. There is no method for a user to change a site's security zone, except to remove it from the current zone and insert it into the new zone. Other browsers, such as Mozilla and Netscape, do not allow for the customization of security zones for individual sites, offering an all or nothing approach. By placing a widget on the tool bar for controlling browser security, Phishing Warden permits users to more easily manage the running of client-side scripts based on the domain name of the web server.

Controlling client-side scripts is only one aspect of IE's security settings. IE security settings also control the download and installation of software from the Internet. Many times a user is presented with pop-up windows displaying ads and dialogue boxes and a yes/no prompt to install new software. By accidentally clicking yes, in order to quickly get rid of a dialogue box, the user could enable the installation of malicious software on the computer. Recent attacks on IE use this vulnerability to install a BHO that attaches itself to IE and reads all the passwords the user submits over an HTTPS connection [2] . A user should not be presented with download prompts on untrusted sites that could attempt to install such malware. If a user wants the full functionality of a web page that includes scripts, she should be able to denote that the site is trustworthy by changing the security level for the domain. The user will then be prompted to install the new software and can respond by clicking yes or no.

The current implementation of content-triggered trust negotiation permits users to override a failed trust negotiation, but it has no option to remember the site a user decides to trust. The present implementation classifies all data being submitted into content types. The server must successfully negotiate trust for each submitted content type before the data is released. If trust negotiation fails for any content type, a pop-up box containing an error message and a list containing all failed content types will be displayed for the user to read. The pop-up error message states that trust negotiation has failed and prompts the user to check if the data should nevertheless be submitted. A user is presented with this pop-up box each time trust negotiation fails, even when the user has previously expressed trust in the site during the same session.

Phishing Warden extends the ability to express trust in web sites that have failed trust negotiation by allowing the user to manage a list of trusted domains. Each trusted domain is associated with a list of content types. A user expresses trust in a domain to receive a given set of content types. Each time a user attempts to submit data to a domain through IE, Phishing Warden verifies that the domain is on the trusted domains list and the domain is trusted with all submitted content types. If so, trust negotiation is unnecessary. Trust negotiation will only take place for those content types for which the domain is not trusted. This provides an easy override method for failed trust negotiations that the user can establish on the first visit and use on subsequent visits, such as a web log-in page.

Phishing Warden maintains a list of previously trusted domains in a file on the hard drive so that trust does not have to be reestablished when a user revisits a website. Each domain name is associated with the sensitive content types that were previously disclosed. There is a risk that an attacker could modify the file to cause Phishing Warden to trust a web site that the user does not trust, or to disclose additional data to a trusted web site that the user does not wish to disclose. To prevent such an attack, Phishing Warden stores a message authentication code (MAC) in the file to verify the integrity of the file. Phishing Warden also encrypts the files containing the list of trusted domains to prevent prying eyes from discovering a possibly sensitive list of trusted sites.

3.2 Implementation

Phishing Warden is a browser helper object (BHO) that is capable of negotiating trust for sensitive content. Phishing Warden contains three components; a toolbar, a Sensitive Data Manager, and a Trust Manager (see Figure 6). Phishing Warden exists as one dynamic link library (DLL).

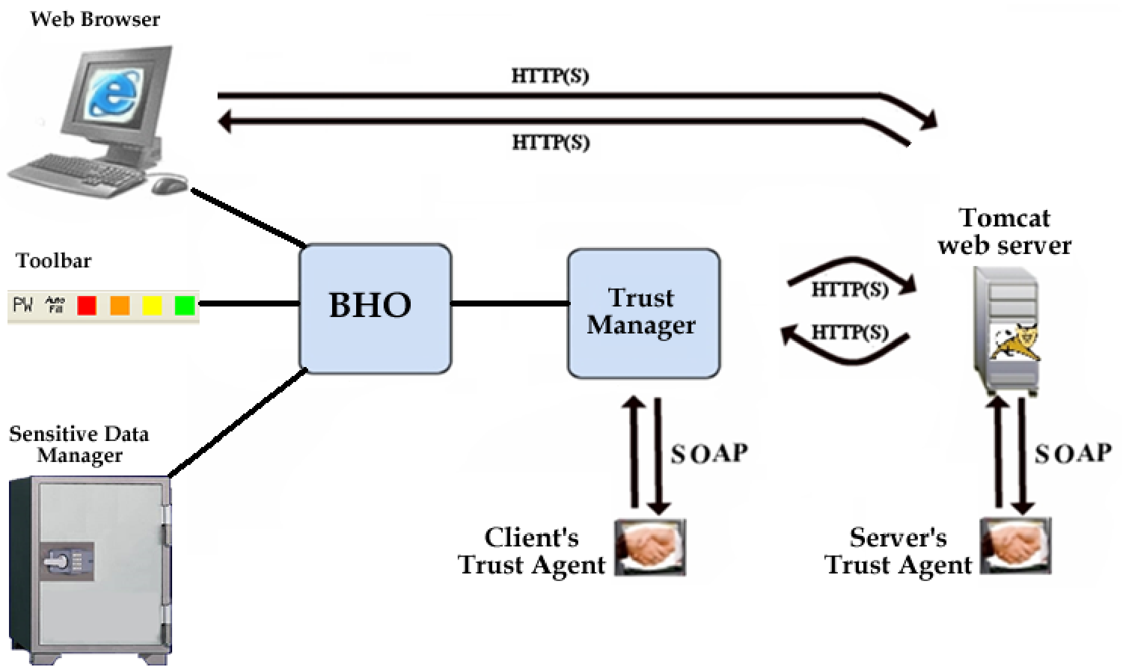

Figure 6: The components of Phishing Warden

3.2.1 Browser Helper Object

A Browser Helper Object (BHO) is a COM object that plug-in applications use to interface with IE. Phishing Warden is a BHO that interfaces with the browser in order to negotiate the disclosure of sensitive data. Phishing Warden has three components that control the release of sensitive data: the toolbar, the Sensitive Data Manager, and the

Trust Manager. Phishing Warden receives commands from the toolbar and uses the Sensitive Data and Trust Managers to effectuate these commands.

In order for Phishing Warden to AutoFill the forms on the current web page in IE, the specific information types must be gleaned from the form. To accomplish this, Phishing Warden scans the current web page for input elements that contain a name field that Phishing Warden uses to discover the information requested by the web form. All sensitive information governed by Phishing Warden is associated with a field name and a content type. The content type is the security classification used by TrustBuilder to determine the policy the web server must satisfy before the resource being requested is released. Not all information handled by Phishing Warden is sensitive. For non-sensitive data, specifying no content type permits Phishing Warden to release the information.

3.2.2 Toolbar

The Phishing Warden toolbar provides users with an interface to trust negotiation, allowing them to request that Phishing Warden fill a web form with information the server is trusted to receive. To fill the web form, Phishing Warden checks the name field of each input tag against a list of name fields. Phishing Warden maintains a mapping between these field names and sensitive content types. The toolbar also contains security buttons that enable the user to set the security zone for the current domain. According to the MSDN [9], IE toolbars are actually tool bands which are COM objects. Programs can call COM objects using the corresponding DLL or executable found in the Windows registry. IE checks the registry for active tool bands and invokes them.

Phishing Warden displays red, orange, yellow, and green boxes on the toolbar corresponding to the following security zones: High, Medium, Medium-low, and Low.

(see Figure 7). IE always displays the current security zone on the status bar in the bottom right corner of the browser. When a user changes the security zone, the toolbar calls the BHO, which changes the registry setting for the current website's domain. The changes do not come into effect until the current web page is refreshed. Phishing Warden automatically refreshes the current page when a security zone change occurs in order to avoid potential security vulnerabilities.

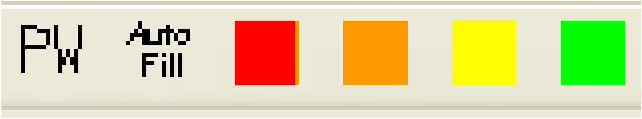

Figure 7: Phishing Warden toolbar

To effectuate the changes in security zone, Phishing Warden updates the Windows registry entry that holds the site's security zone. Since IE refreshes the security zone without having to restart the application, the user's experience is seamless. Each security zone has an associated security level (see Table 1). The toolbar uses IE's default security zones and levels.

The default security levels for IE are as follows:

- High provides the most security, but at a cost of functionality, and is recommended for sites thought to be harmful.
- Medium provides for safe browsing while still maintaining functionality. Users are prompted to grant explicit permission for running potentially harmful programs.
- Medium-low is the same as Medium, except it has fewer prompts by enabling some features automatically.

- Low has minimal safeguards and warning prompts, and allows most content to be downloaded and run without prompting the user [29].

The following table illustrates the default security mappings between security zones and security settings:

Security Zone	**Default Security Level**
Internet	Medium
Local intranet	Medium-low
Trusted sites	Low
Restricted sites	High

Table 1: Default security zone to security level mappings

An analysis of the Windows Registry indicates how IE modifies the registry to adjust the security level of website domains. When a user adds a website domain to a security zone through IE's security options, a new registry entry appears in HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap \ProtocolDefaults\http and https. If the user moves the domain from one security zone to another, IE changes only the registry key's value. Phishing Warden can leverage this to insert a new entry at will and change its value whenever the user desires. This affects the user's browsing experience by allowing Phishing Warden to manage changing between security zones. IE will automatically read the registry for any new entries and adopt any changes in security.

The toolbar also contains a configuration GUI that appears when the user presses the button labeled **PW** (see Figure 8). This GUI interacts with the Sensitive Data Manager to set and retrieve the user’s stored sensitive information. Phishing Warden associates each value entry with a form field name for later use with AutoFill. Besides the form field name, Phishing Warden associates each sensitive value entry with a content type that maps the sensitive value to an access control policy. A blank content type field indicates that the value is not sensitive.

For surfing convenience, Phishing Warden remembers any site that the user accepts as trustworthy of certain sensitive information. Phishing Warden adds a check box that enables this memory feature. This check box appears on the failed negotiation pop-up. If the user checks the box, Phishing Warden will store the current domain name of the page the user is visiting along with the corresponding content types that the user permits Phishing Warden to override. As long as web forms from the domain request no new content types, Phishing Warden will submit the sensitive information to that domain without prompting the user for permission. The user can remove the trusted domain by editing the list of trusted sites stored on the computer via the Phishing Warden Configuration GUI.

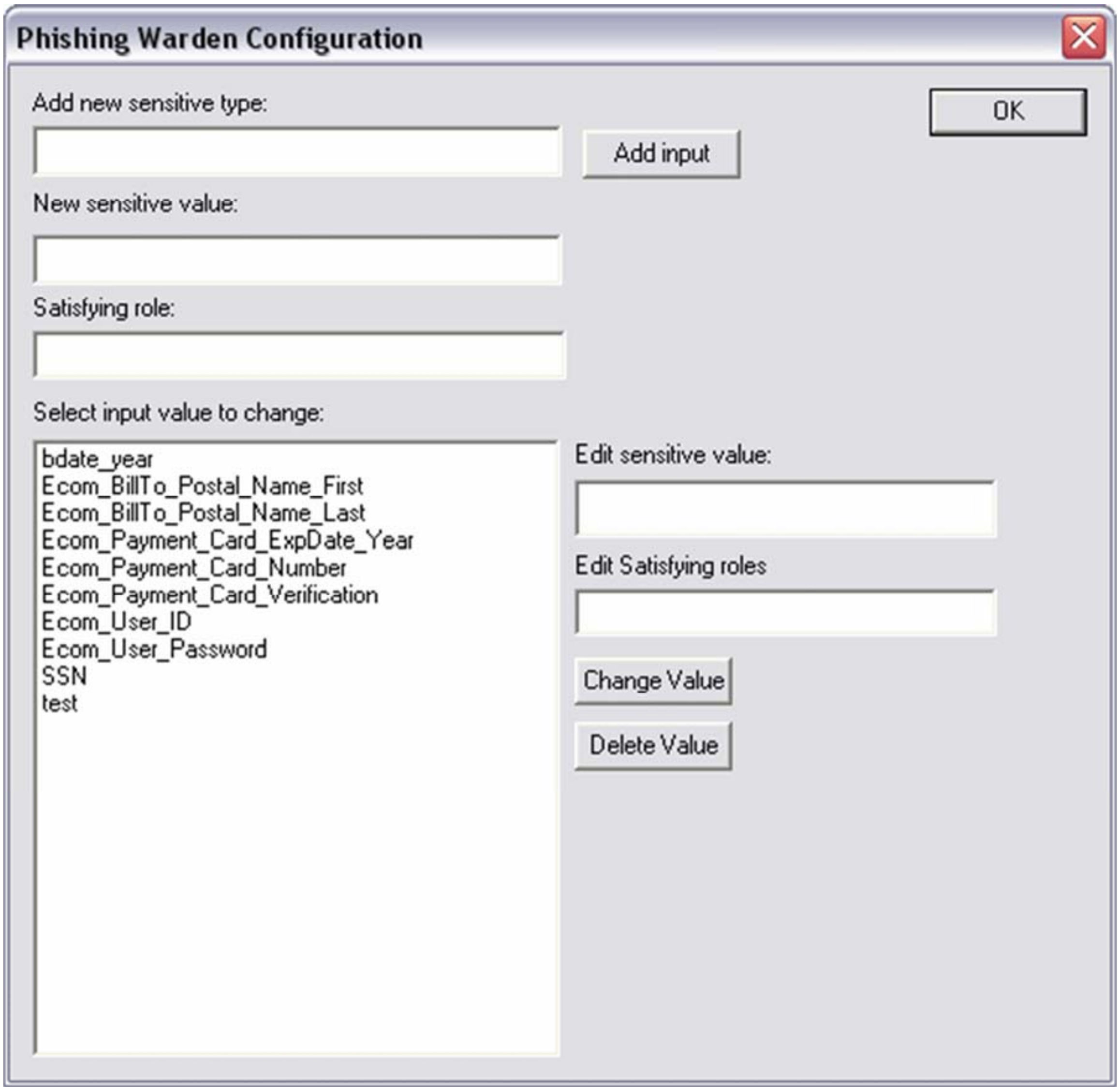

Figure 8: Phishing Warden sensitive type configuration GUI

3.2.3 Sensitive Data Manager

The last part of Phishing Warden concerns the storing of sensitive information using the Sensitive Data Manager. SpoofGuard and CTTN avoid storing a plaintext copy of sensitive information by storing only a one-way hash of the data. In contrast, Phishing Warden must have access to the plaintext data in order to automatically fill in the form fields. To do this, Phishing Warden stores user data in an encrypted file on the local machine. When a user presses the AutoFill button or the PW configuration button, he is prompted for a password to decrypt the file. A user is able to unlock the data only if he presents the correct password.

Although entering a password requires more steps for a user to fill out a form than just pressing a button, it is usually less information to type compared to completing an entire form. To limit the number of prompts a user receives, Phishing Warden requests a password only the first time the sensitive information needs to be accessed during a session. This password is stored in memory for as long as the user's IE session remains open.

Phishing Warden derives a key to encrypt or decrypt the sensitive information from the password. The key is generated using PBKDF2, a password based key derivation function from PKCS #5 [22]. To ensure strong encryption, PBKDF2 generates a 128 bit key from the user's password. Crypto++ Library 5.2.1 contains the PBKDF2 and is freely available to the public [10].

Crypto++ also has the necessary functions to enable Phishing Warden to encrypt and decrypt the sensitive user data. The current implementation of Phishing Warden uses Crypto++'s AES function as the encryption/decryption algorithm. AES accepts keys of length 128, 192, 256, or multiples of 32 beyond 256. AES was chosen for its wide acceptance as a symmetric block cipher.

To protect sensitive data from being accessed in virtual memory, it is necessary for the Sensitive Data Manager to pin the pages of memory containing sensitive data. This prevents the sensitive data from being written to disk during a page fault. If the sensitive data is written to disk, a wily attacker could search the disk for the sensitive data and find it in unencrypted form. The Windows Platform SDK contains memory management functions that allow a programmer to override typical OS behavior. Phishing Warden uses VirtualLock from the Platform SDK to prevent Windows from caching Phishing Warden's sensitive variables to disk [27].

3.2.4 Trust Manager

For trust negotiation to occur, a communication link between the client (browser) and the server (web server) needs to be established. Both client and server require trust agents to process the trust negotiation headers sent between the client and server. The Trust Manager component of Phishing Warden handles the calls for trust negotiation, taking a sensitive content type and a web server as the only input. Phishing Warden initializes the Trust Manager once per AutoFill request. If Phishing Warden finds several sensitive content types in a web form, it invokes the Trust Manager to negotiate the release of each unique content type found in the web form.

The Trust Manager relays all trust negotiation requests to the client's trust agent using SOAP RPC. The Trust Manager talks to the client's trust agent and transmits the trust negotiation information in an HTTPS session with the server, separate from the browser's HTTPS session. The Trust Manager redirects all replies from the server to the client's trust agent and returns the result of the trust negotiation, success or failure, to Phishing Warden.

When a user clicks the AutoFill button, IE calls the Phishing Warden BHO with a mouse-click event. Phishing Warden searches through the current web page for HTML text input tags. Each text input tag contains a name field that Phishing Warden sends to the Sensitive Data Manager to find the associated sensitive data type. To avoid possible mismatches, Phishing Warden assumes that the content of the input tag name fields follows the ECML, otherwise Phishing Warden ignores the text input tag. If Phishing Warden finds a matching sensitive content type, then Phishing Warden negotiates for the release of that data type by calling negotiate on the Trust Manager, giving only the name of the server with which to negotiate trust and the sensitive content type to negotiate. Phishing Warden extracts the name of the server from the domain name contained in the IE address bar. Phishing Warden assumes that the web server at this location can interpret the trust negotiation headers sent in the HTTP request.

If the Trust Manager returns a failed trust negotiation result, Phishing Warden adds the failed content type to a list of failed content types to be displayed to the user after checking all input fields. If trust negotiation succeeds, the Trust Manager adds the domain name to a list of trusted domains and the sensitive data types for which the domain has successfully negotiated trust. The Trust Manager caches the list in memory for the length of the browser session to eliminate redundant negotiations. At the end of all negotiations, Phishing Warden automatically fills in the value field of the text input tags with the non-sensitive and sensitive values for which the server is trusted.

The Trust Agent's SOAP RPC interface requires the implementation of the following steps for the Trust Manager to successfully negotiate trust:

1. The Trust Manager creates the Trust Agent.
2. The Trust Manager initializes the Trust Agent by calling the `init` function, giving the location of the TrustBuilder XML configuration file.
3. The Trust Manager initiates the trust negotiation by calling `generateHelloMessage` on the Trust Agent to enumerate the possible

strategy families and formats and initializes the signature/verify material used to verify certificates.

4. The Trust Manager's Trust Agent receives a completed `HelloMessage` from the server in `receiveClientHelloMessage` and concludes initialization.
5. The Trust Manager calls `negotiate` on the Trust Agent to begin negotiation.
6. The Trust Agent negotiates trust until it reaches a state of success or failure and returns the results to the Trust Manager.

The Trust Manager's C++ soap interface to TrustBuilder was implemented using the gSOAP toolkit. gSOAP is open source Web services toolkit with an easy to use WSDL parser and skeleton function generator that creates C/C++ functions to interact with SOAP services on different platforms. The necessary RPC calls (`init`, `generateHelloMessage`, `receiveServerHelloMessage`, and `negotiate`) were created using gSOAP [14].

Chapter 4

Results and Analysis

The main goals of Phishing Warden are to protect sensitive content from phishing attacks and provide users with an intuitive interface that is not prohibitive to use. Performance is a secondary concern. The following sections discuss Phishing Warden's minimal impact on usability, a threat analysis of Phishing Warden, and performance results.

4.1 Usability

Programs that safeguard information are often ignored if the user cannot understand how to use them. An example of this is the security zones of Internet Explorer that are rarely used by the average user. The goal of Phishing Warden is to enable browser users to securely disclose sensitive information to parties they can trust. If users bypass Phishing Warden because it is difficult to use, then it cannot accomplish this goal.

Phishing Warden addresses the usability problems of Internet Explorer's security zones by reducing the number of steps that a user must take to change security zone settings. Instead of changing security zones by clicking through a series of menus and options, Phishing Warden enables users to change security zones with the click of a button that is easily visible on the toolbar. This functionality change increases browser security usability by allowing IE users to manage the execution of client-side scripts from the toolbar.

Now instead of all web pages being seen as part of the Internet zone (medium security level), Phishing Warden increases security by making all web pages default to the non-trusted (high security level) zone. The user can adjust the security zone (website domain) when they deem necessary by simply pressing the corresponding security button on the toolbar. They can even change the security zone back to non-trusted after they are done visiting the website. To change the default security zone, a simple registry change is required to make the default security zone the non-trusted zone. This feature reduces the attack surface as a result of web sites receiving reduced permissions.

To effectuate the change in default security zones, Phishing Warden must change the registry settings for HTTP and HTTPS. The registry keys can be found at: HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap \ProtocolDefaults\http and https. The default setting for these registry keys is 3 (Internet Zone). To change to a non-trusted zone for both HTTP and HTTPS, the keys should be changed to 4 (Restricted sites).

Phishing Warden decreases usability by requiring web sites to conform to the ECML to allow the AutoFill feature to parse the HTML form for the requested data types. While users can map sensitive values to form field names that differ from the standard, this puts a large responsibility on the user and does not guarantee that Phishing Warden will comprehend the form names on every website. One solution is to create a list of commonly used synonyms for certain types of sensitive information. For example, the list for a field containing a social security number could look something like the following: ssn, social, soc_num, and social_security_number. Users could append new entries to the list each time they found a new variant on a web form. This approach requires manual intervention each time the user encounters a new field name in web forms.

4.2 Threat Analysis

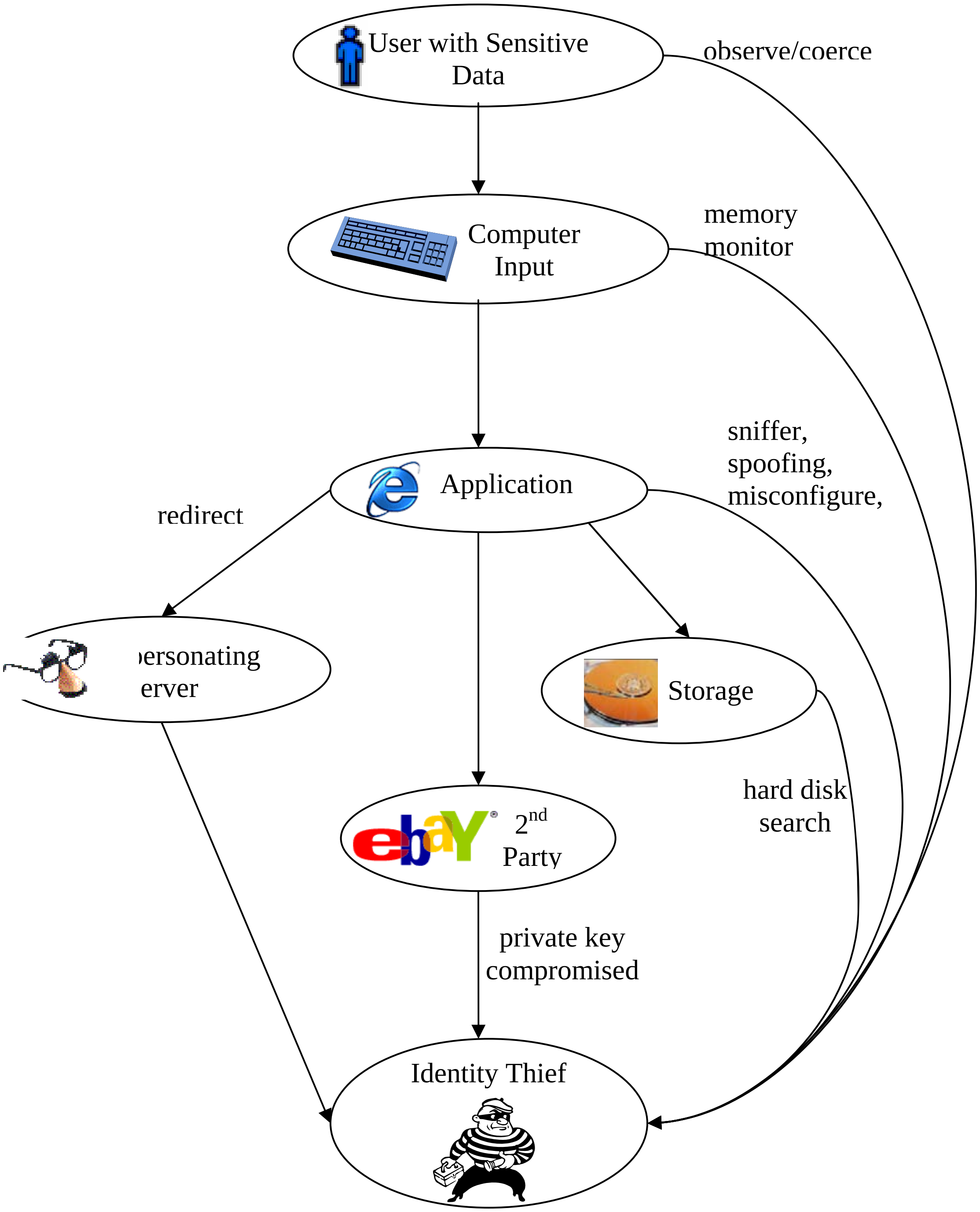

Figure 9: Diagram illustrating the ways an identity thief might steal personal information

4.2.1 Threat Analysis of Information Flow between Browser User and Web Server

Figure 9 illustrates various ways that an identity thief might steal the personal information a user enters into a form on a web page. Identity theft could take place if an attacker is able to observe a user while he inputs or uses the sensitive data. For example, thieves can shoulder surf kiosks at a public library and view an Internet surfer’s credit card number during an online purchase. To protect themselves, users should be wary of onlookers when in public places and store sensitive data in a secure location.

The next potential state where a loss can take place is inside the computer. Internet users entrusting their computer with sensitive information must takes steps to secure their machine. If a hacker gains access to the computer, installs a memory monitoring program such as a keystroke monitor, and possesses the means of extracting the data, then they may steal sensitive information. To better protect the sensitive data contained on the computer, users should take steps to reduce the threat from attacks such as Trojans, Internet worms, and e-mail viruses by installing anti-virus and firewall solutions.

The applications a user trusts with sensitive information can also foster identity theft in three different ways: memory, misuse, and in transit. Applications store information in memory or on disk where attackers can find them. Firewall and anti-virus solutions can help prevent memory scanners. Misuse of applications or naivety can lead users to divulge information through applications with parties that they should not trust (e.g., misdirection of a request from a browser to a masquerading site). If an application transmits sensitive data to a trusted party using an insecure communication channel, it is possible for an eavesdropper to capture the sensitive information during transmission. Programs transmitting sensitive information should use secure protocols such as TLS.

Once a user discloses sensitive information to a server, the user loses control over further dissemination of the information. If the server stores the information and does not adequately protect it, hackers might be able to obtain it. Another threat is the server's administrator might decide to divulge the data to some other party that the user does not trust. These threats illustrate the need for users to carefully choose the servers they trust.

Phishing Warden focuses on preventing identity thieves from misdirecting browser users. Trained users can use Phishing Warden to remove the guess work regarding which web sites to trust with sensitive information. To do this, Phishing Warden must store the sensitive information in a file on the user's computer. This adds a new security concern that an attacker might exploit this file.

Phishing Warden stores the user's sensitive data using PKCS #5, which helps safeguard the information. However, the protection is only as strong as the user's password. The risk of losing sensitive data is only present only when an attacker has access to the encrypted file containing the sensitive data. Users can minimize this risk by using a firewall/anti-virus solution.

4.2.2 Private Key Compromise

Since Phishing Warden relies on trust negotiation, it faces the same risks. The first risk is the compromise of a CA private key. For example, if an access control policy states that a server must be a member of the Better Business Bureau (BBB), and an attacker steals the Better Business Bureau's private key, then the attacker could create a fake business credential signed with the stolen key. Any trust agents that receive the fake business credential would determine that the credential has been properly signed by the BBB and would authenticate the fake business to the role of member of the BBB. To resolve this problem, the BBB must revoke all credentials signed by the compromised private key, generate a new private key, acquire a new BBB certificate, and reissue all legitimate credentials.

Another threat to trust negotiation is the loss of a trusted credential's private key. If the BBB issues a credential to a legitimate company, and an attacker stole the company's associated private key, then the attacker would be able to impersonate the legitimate company. Any user negotiating trust with the attacker would determine that the attacker was the legitimate company and their trust agent would authenticate the attacker to the role of member of the BBB. To resolve this problem, the BBB must revoke the credential and reissue the legitimate company a new credential, while the legitimate company must generate a new key pair and reissue all credentials signed with the compromised key.

4.2.3 Attacking Stored Information

By storing sensitive information in a file on the computer, Phishing Warden introduces a risk that the information may be stolen by someone who has access to the file. The thief can be a hacker who cracks into an account with administrative privileges or a legitimate user of the computer who has read access if there is a mistake in file permissions. To mitigate this risk, Phishing Warden encrypts the sensitive information that it stores on the computer's hard drive.

Encryption is not invulnerable to attack by cryptographically savvy hackers who can try to use a brute force attack or dictionary attack to retrieve sensitive data stored on the user's machine. While these two attacks are not new cryptographic attacks, the introduction of sensitive data stored on the user's machine, even though encrypted, is a security risk. However, if Phishing Warden is used with an effective anti-virus/firewall solution and the user has chosen a well-formed password, the threat of these new risks is minimized.

Brute force attacks are an attempt to exhaustively search the key space for a key that will decrypt the file storing the sensitive information. A brute force attack on 128 bit symmetric keys is infeasible with today's computers. Silverman [23] states that it would take 10^{16} years to brute force a 128 bit symmetric key. In the time it takes to brute force a 128 bit symmetric key, a phishing victim not using Phishing Warden would have likely entered in the information by hand, making them an easy target for a key stroke monitor. Historically, the answer to thwarting a brute force attack is to increase the key size.

The current implementation of the Phishing Warden uses the Advanced Encryption Standard (AES) with a 128 bit key. AES, also known as Rijndael, is a cipher with variable block and key length. The 128 bit key generated by PBKDF2 is large enough to prevent brute force attacks today. However, this attack could become practical in the future. Kaliski estimates that 128 bit keys are safe until 2031 when hardware technology is expected to catch up with the 128 bit key [19]. If unanticipated improvements in processor speeds and cracking algorithms appear, the time to break the encryption may be reduced, but the encryption key size can be increased to compensate for any new threats.

A dictionary attack is a threat when using password-based encryption to safeguard secrets stored on a computer. An attacker who obtains a copy of the encrypted information can conduct the attack off-line. Instead of doing an exhaustive search of the entire space of possible passwords, an attacker uses a dictionary of commonly used words and phrases to quickly guess probable, poorly chosen passwords. An encryption algorithm is only as strong as its weakest link, and for password-based encryption algorithms, that weakest link is a poorly chosen password.

An experiment conducted at Purdue “illustrated how unconstrained user choices for passwords may compromise security, with at least 20% of all chosen passwords being weak [24].” To prevent dictionary attacks, users should be informed that security depends on the strength of their password. Strong passwords can even be generated by programs, or users can follow guidelines on creating strong passwords [3] [5].

Another possibility of attack is if a user with permission to use the machine is able to access the encrypted file and successfully launch a dictionary attack. This bypasses any security measures by using the operating system’s user privileges. To secure the information from other users, the owner of the encrypted file must be the only user to have read permission. Administrator accounts would have to be trusted. While this seems like a new possibility for identity theft by co-workers, it was possible before, though in a different form. Users with access to the machine and administrator privileges could install a key stroke monitor which would catch any information entered into web pages.

Phishing Warden relies heavily on a good anti-virus/firewall solution to provide for information security. The main goal of an anti-virus/firewall solution is to protect the user from outsiders who would hijack their machine. Phishing Warden’s main goal is to protect the user from sending sensitive information to outsiders who would steal their identity. By storing sensitive information on the hard drive, the user is vulnerable to a whole new form of identity theft that doesn’t require them to fall victim to a phishing attack. While the file containing sensitive information is encrypted, there is now a possibility for an attacker to decrypt the file once they gain read access through brute force or password guessing. By coupling Phishing Warden with a machine secured from viruses and worms, the chances of an attacker obtaining sensitive information are decreased significantly.

4.2.4 Spoofing Attacks

Ye et al. demonstrated another form of attack, called web spoofing, in which the downloaded web page creates a toolbar using Java and/or JavaScript to replicate the browser toolbar [31]. The reason this attack is so formidable is because it completely replicates the browser toolbar. If the toolbar is designed well and the user is using the corresponding browser with the default toolbar, users cannot distinguish between a real toolbar and a fake. The location bar is a critical part of any browser that lets the user know which site they are visiting. The status bar on the bottom of a browser lets the user know the current web page was sent over a secure communication. A fake toolbar can lull a user into submitting sensitive data by supplanting the location bar and SSL security lock.

In the case of Phishing Warden, the replicated toolbar would need to include the Phishing Warden plug-in. If AutoFill is clicked in the spoofed toolbar, the attacker could try to confuse the user with an error message that makes it appear that Phishing Warden is not working and causes the user complete the form manually. This attack requires that the attacker replicate the exact look and feel of the browser toolbar and launch it against someone using Phishing Warden.

To prevent such an attack, JavaScript and Java applets should only be enabled on trusted sites, or Internet browsers should not be permitted to open new windows without displaying the standard toolbar. Disabling JavaScript and Java applets on un-trusted sites limits functionality, but increases overall security. Another solution is to present a different look to browser windows that are opened without toolbars enabled so that users can distinguish between the two [30]. Phishing Warden copes with this attack by setting the default security levels high enough to prevent an elaborate deceptive toolbar from being displayed.

While Phishing Warden does AutoFill input form fields, a clever attacker may attempt to use other means of inputting information, such as buttons, applets, and select boxes. Phishing Warden does not detect other forms of user input. To prevent users from inputting sensitive information in an alternate fashion, Phishing Warden requires that all form submission elements be input fields. To enforce such a ruling, users must be instructed not to use any other means of submitting data into web pages. Another solution is to standardize non-text input so that Phishing Warden is able to discern the type of information requested, and how to insert the correct data into the non-text input.

4.2.5 Misconfiguration

The last security threat is due to mis-configured access control policies and security zones. If inexperienced users create their own access control policies, it is possible that non-trusted sites will be able to access the user's sensitive information. It is also possible that user created policies will cause trust negotiation to fail when a site should be trusted. To avoid any possible confusion in policy creation, security experts should create the access control policy and make them available for download by average users.

With incorrect security settings, malicious websites could compromise the integrity of the system. Unsafe security settings could be set by a user misclicking a security button, or by the user's lack of understanding of the true consequences of setting the security level at a low setting. There are three possible solutions to help prevent users from opening their browsers to security vulnerabilities: make all zones safe, remove unsafe zones from the toolbar, or use trust negotiation to manage security zones.

The first solution involves changing the default security zones to safer settings where all zones prompt or disable browser options known to contain vulnerabilities. While this does prevent users from opening themselves to malicious websites, it could hinder usability with trusted websites. Removing unsafe zones from the toolbar is essentially the same as the first solution except that there are now fewer security zones.

The last possibility is to remove control of security zones from the user and instead use trust negotiation to manage which sites will be given different security zone access based on access control policies. This solution shifts the weight of security decisions from the user to security experts that are relied upon to create the access control policies. Using trust negotiation will ensure that safe zones are fully functional while unsafe zones are prevented from mounting attacks that use the privileges of a lighter security zone. Trust negotiation as a browser security manager is left for future work.

4.3 Performance Results

Trust negotiation rapidly exchanges credentials and policies through a HTTP or HTTPS connection. The implementation of Phishing Warden completes a negotiation for one sensitive content type in approximately 1 second. The test was conducted using two Pentium 4 machines running at 3.2 GHz with 1GB of RAM connected through a 100 Mb/s router on the same LAN. Over a wide-area network such as the Internet, network latency would increase the total elapsed time for trust negotiation for each round trip. As for encryption and decryption of the sensitive data, the time to encrypt and decrypt nine sensitive data types using AES averages 90 milliseconds. These times include the time to write to disk when encrypting, and the time to read from disk when decrypting. The only significant impact on usability for the user is the time to enter the password before the first negotiation during a session.

The average user likely has sensitive data types for user account passwords, social security number, credit card, CVV2, checking account number, mother's maiden name, employment information, contact information, and date of birth. In addition, the typical access control policy controlling disclosure of a user's sensitive data usually can be satisfied by one or two roles. Experiments involving policies with two roles indicate that Phishing Warden performs adequately under these assumptions. Phishing Warden takes an average of 1030 milliseconds for trust negotiation to authenticate the server to the first role, and 60 milliseconds to authenticate a server to each additional role.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Overall the security of transferring sensitive information over the Internet has been increased due to Phishing Warden which has successfully raised the bar for phishing attacks. With Phishing Warden, users will be more confident when they have submitted sensitive data and more hesitant to trust a website that cannot complete a successful trust negotiation. Phishing Warden also makes it easy for users to convey trust in websites that do not have trust negotiation, helping improve the transition of trust negotiation from research into mainstream use. Along with trust negotiation aware browsers come new possibilities such as ensuring websites conform to privacy policies. Despite the benefits of Phishing Warden, there are two new significant security risks; namely storing sensitive information on the hard drive and decreasing browser security by misconfiguring security zones.

Phishing Warden increases confidence upon successful completion of trust negotiation by allowing users to know that the server receiving the sensitive information has successfully authenticated to the required roles for each sensitive content type submitted. Users will also be more wary when web sites are not able to complete a trust negotiation. A warning message will appear stating that trust negotiation has not been successful, followed by the sensitive content types requested by the form.

When the failed trust negotiation warning message is displayed, an option is given to ignore the failed trust negotiation and send the information despite the failure. By modifying content-triggered trust negotiation, Phishing Warden also includes a check box to indicate whether this site should be permanently trusted for these sensitive content types. This increases the ease of use of content-triggered trust negotiation by remembering previous sites that the user has deemed trustworthy and not continually annoying the user with another pop-up.

Although Phishing Warden raises the bar for phishing attacks, it also enables new forms of attack on the sensitive information. If attackers are aware that a machine uses Phishing Warden, and they are able to compromise the machine, then the attackers can retrieve the sensitive data in encrypted form and mount a dictionary and/or brute force attack on the encrypted file. To reduce the threat of dictionary attacks, a password checker can be integrated with Phishing Warden to identify easily broken passwords. Brute force attacks are not feasible for the average phishing attacker if the encryption key size follows standard guidelines.

The threat model (Fig. 9) in section 4.2 illustrated the paths an identity thief can use to obtain a victim's sensitive information. Security starts with the individual safeguarding their information from shoulder surfers while entering data into the computer. Once information is entered into the computer, an anti-virus/firewall solution is relied upon to prevent hackers from monitoring the keystrokes and computer memory. Next, applications must be configured correctly and the user must know how to use them securely. Lastly, information shared with the second party must be kept securely from identity thieves.

Phishing Warden helps computer users foil phishing identity thieves who prey on web users by misdirecting their Internet applications. Through the use of a browser plug-in, Phishing Warden can control the submission of sensitive data through web forms. By funneling all sensitive data to be inserted into web forms into a central security manager, Phishing Warden is able to control the flow of information between web users and web server, effectively preventing identity theft.

5.2 Future work

In order to increase overall security, sensitive data should not be stored on a hard drive. The need for storing sensitive data on the hard drive could be eliminated by implementing a pop-up window which asks the user for the sensitive information. The pop-up window would only appear when the AutoFill button is pressed and would contain all the values of the form that are being submitted. The user would be required to type in each of the sensitive values corresponding to the name of the input field found on the form. This increases information security, as no sensitive information is stored on disk for hackers to harvest, but it also decreases usability as users must now type in each requested field. Another possible solution is to store sensitive information on a smart card to eliminate the need for a user to input data and also refrain from storing the data on the hard drive.

Another possibility for future work is to increase security by removing the security buttons and replacing them with an automated system of changing security levels based on trust negotiation. This can be done by creating access control policies for each security level. The web site must satisfy the policy before Phishing Warden will grant permission to run at that security level. A variation of automated security levels includes using the principle of least privilege. Phishing Warden could accomplish this by using content analysis to determine the security features a web page is requesting and negotiate only on those features.

Similar to the way trust negotiation could manage security levels, trust negotiation could also be used to manage cookies. Web servers have the ability to set cookies in the headers of web pages they send. Each time a user requests a web page, the web browser sends the web site's cookie back to the server. This enables the server to cache information unique to each user. IE allows users to manage their privacy through the controlling of cookies. Because IE uses the windows registry to control cookies, it is possible for cookie settings to be managed by Phishing Warden according to the results of trust negotiation. Websites could easily be authenticated using cookie access control policies and the cookie privacy setting could be set accordingly. This addition to Phishing Warden would eliminate the need to configure cookie privacy settings for individual sites. Users of Phishing Warden would only be required to obtain an access control policy, created by security experts, that expresses the desires of the user and Phishing Warden would dynamically change the setting of privacy levels.

Another potential use for Phishing Warden is to provide assurance to users that a web server adheres to the privacy practices published on its web site. Assuming that a trusted organization was commissioned to audit website privacy practices, the privacy auditing body could issue certificates to websites that submitted to an audit. The certificates would state each privacy policy test that the web site has passed. These certificates could expire to ensure that the web site is periodically audited. A user could configure Phishing Warden to require that a server satisfy certain privacy roles before the user's private data is submitted. Used in conjunction with the WC3's Platform for Privacy Preferences Project (P3P) [8], leveraging trust negotiation can provide greater assurance that the privacy practices received from the website are followed.

Bibliography

[1] Anti-Phishing Work Group, http://www.antiphishing.org, 2004.

[2] J. Bambenek, “BHO Scanning Tool and New Scam Targets Bank Customers,” Handler's Diary, SANS.org, June 29, 2004.

[3] R. Barbalace, “How to Choose a Good Password (And Why You Should),” MIT Student Information Processing Board, http://www.mit.edu/afs/sipb/project/doc/passwords/passwords.html, August 11, 1999.

[4] “CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests,” CERT, http://www.cert.org/advisories/CA-2000-02.html, February 3, 2000.

[5] “Choosing a Good Password,” Netscape, http://wp.netscape.com/security/basics/passwords.html, July 26, 2004.

[6] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, “Client-Side Defense Against Web-Based Identity Theft,” *Proceedings of the 2004 Network and Distributed System Security Symposium*, San Diego, CA, February 2004.

[7] “Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design,” Electronic Frontier Foundation, July 1998.

[8] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle, “The Platform for Privacy Preferences 1.0 (P3P1.0) Specification,” W3C Recommendation, April 16 2002.

[9] “Creating Custom Explorer Bars, Tool Bands, and Desk Bands,” Microsoft Developer Network, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/Shell/programmersguide/shell_adv/bands.asp, May 2004.

[10] “Crypto++ Library 5.2.1,” http://www.cryptopp.com, July 23, 2004.

[11] “EarthLink Toolbar Featuring ScamBlocker,” EarthLink, http://www.earthlink.net/earthlinktoolbar/download/, July 22, 2004.

[12] D. Eastlake and T. Goldstein, “ECML v1.1: Field Specifications for E-Commerce,” Request for Comments: 3106, April 2001.

[13] D. Esposito, “Browser Helper Objects: The Browser the Way You Want It,” Microsoft Corporation, January 1999.

[14] “gSOAP: SOAP C++ Web Services,” http://www.cs.fsu.edu/~engelen/soap.html, July 23, 2004.

[15] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor and Y. Ravid, ”Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers,” *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.

[16] A. Hess, J. Holt, J. Jacobson, and K. E. Seamons, “Content-Triggered Trust Negotiation,” *ACM Transactions on Information System Security*, Vol. 7, No. 3, August 2004.

[17] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. E. Seamons, and B. Smith, “Advanced Client/Server Authentication in TLS,” *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, February 2002.

[18] A. Hess and K. Seamons, “An Access Control Model For Dynamic Client-side Content,” *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, Como, Italy, June 2003.

[19] B. Kaliski, “TWIRL and RSA Key Size,” RSA Laboratories Technical Report, May 6, 2003.

[20] "Microsoft Security Bulletin MS04-004," Microsoft, http://www.microsoft.com/ technet/security/bulletin/MS04-004.mspx, April 12, 2004.

[21] "'Phishing' Scams Shooting Up," http://www.cnn.com/2004/TECH/internet/05/06/ internet.phishing.reut/, May 6, 2004.

[22] "PKCS #5 v2.0: Password-Based Cryptography Standard," RSA Laboratories, March 25, 1999.

[23] R. Silverman, "A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths," RSA Laboratories Bulletin #13, April 2000.

[24] E. Spafford, "Observing Reusable Password Choices," Purdue Technical Report CSD–TR 92–049, July 31, 1992.

[25] G. Tally, R. Thomas, and T. Van Vleck, "Anti-Phishing: Best Practices For Institutions and Consumers," McAfee Research White Paper, www.mcafeesecurity.com, March 2004.

[26] Tumbleweed, "Using Digital Signatures to Secure Email and Stop Phishing Attacks," Tumbleweed White Paper, www.tumbleweed.com, 2004.

[27] "VirtualLock," Microsoft Developer Network, http://msdn.microsoft.com/ library/default.asp?url=/library/en-us/memory/base/virtuallock.asp, July 23, 2004.

[28] W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation," *DARPA Information Survivability Conference and Exposition*, Hilton Head, SC, January 2000.

[29] "Working with Internet Explorer 6 Security Settings," Microsoft, http://www.microsoft.com/windows/ie/using/howto/security/settings.asp, May 17, 2004.

[30] E. Ye, S.W. Smith, ``Trusted Paths for Browsers," *Proceedings of the 11th Usenix Security Symposium*. San Francisco, CA, August 2002.

[31] E. Ye, Y. Yuan, and S. Smith, “Web Spoofing Revisited: SSL and Beyond,” Technical Report TR2001-417 at Dartmouth College, February 1, 2002.

Appendix A – ECML Example

The following is an example of an ECML compliant HTTP form taken from RFC 3106. The input tags each have a name field which follows the ECML. The first input tag, named Ecom_Payment_Card_Name, is preceded by descriptive text in a paragraph tag that enables the user to read the form while the ECML field name hidden in the HTML code allows a computer programs to parse the form in a systematic manner. A properly configured Phishing Warden would associate the next input tag, Ecom_Payment_Card-_Number, with a sensitive data type to protect the user’s credit card number. Once Phishing Warden was called upon to AutoFill the page, this request for sensitive data would trigger a round of trust negotiations to determine whether the credit card number should be released.

```
<HTML>
<HEAD>
<title> eCom Fields Example </title>
</HEAD>
<BODY>
<FORM action="http://ecom.example.com" method="POST">
        Please enter card information:
        <p>Your name on the card
        <INPUT type="text" name="Ecom_Payment_Card_Name" SIZE=40>
        <br>The card number
        <INPUT type="text" name="Ecom_Payment_Card_Number" SIZE=19>
        <br>Expiration date (MM YY)
        <INPUT type="text" name="Ecom_Payment_Card_ExpDate_Month" SIZE=2>
        <INPUT type="text" name="Ecom_Payment_Card_ExpDate_Year" SIZE=4>
        <INPUT type="hidden" name="Ecom_Payment_Card_Protocol">
        <INPUT type="hidden" name="Ecom_SchemaVersion"
        value="http://www.ecml.org/version/1.1">
        <br>
        <INPUT type="submit" value="submit"> <INPUT type="reset">
</FORM>
</BODY>
</HTML>
```