Anti-PhAshing: User Study Proposal

1. INTRODUCTION

People have done much research on anti-phishing. The focus of resolving phishing problems from User Interface aspect has been highly respected in different ways. Following the proposals on various anti-phishing toolbars, Ross et al proposed *PWDHash* [4], Dhamija et al proposed to *Dynamic Security Skin* [2], and Wu et al proposed *Web Wallet* [1] around the user interface design improvement. However behind the discussion on these anti-phishing applications, there is another interesting problem to address: "*How to prove your anti-phishing tools are real?*" In this proposal, we introduce a concept *PhAshing*, which reads [fæʃing] and means faked application. The approach of counter measures against *PhAshing* is *Anti-PhAshing*.

In this proposal, we address several useful methods that can help users improve their security protection. We use *Web Wallet* as an example application to provide referential UI, and use *Genuine Skin*, *Application Trace*, *Spring-Loaded Button*, and *Interactive Image Filter* as candidate methods to carry out the investigation. We also use the merged approach of these methods to study the final effect of integrative method.

In this paper, we identify the problem in Section 2. We propose method we would like to use in Section 3. Finally, we address the experiment design in Section 4.

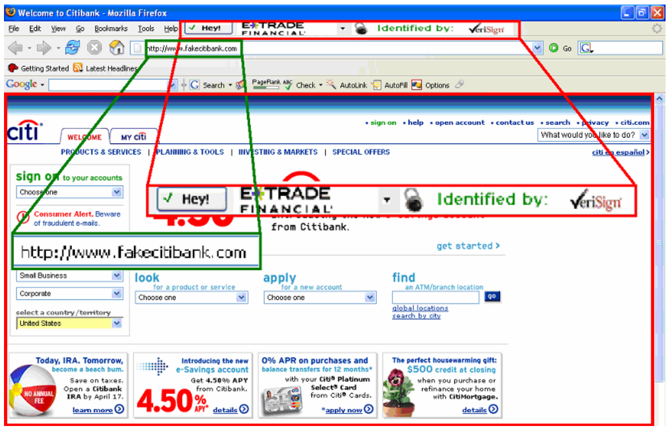
2. PROBLEM IDENTIFICATION

2.1 No Part of the Screen can be Assumed to be Secure

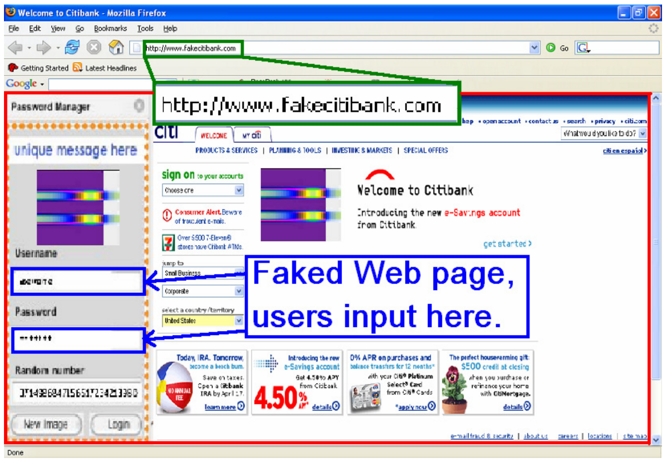
The major ways that we are using to access the Web are web browsers. Phishers are always trying their best to make the faked web pages look and behave as similar as possible to the targeted ones. On the other hand, the web browsers are adding more and more features (Java Script, ActiveX, Java Applet, Macromedia Flash, etc.), plug-ins (Adobe Acrobat, Google Toolbar, MSN Toolbar, OICQ, etc.), and more other will come to be added. These features and applications are making web browsers "too" powerful that information is no just limited to the area of client windows in web browsers any more. For instance, ActiveX components and Java Scripts have been used by phishers constantly to hide real address bars in web browsers. So under no circumstances we can assume a part of the computer screen is secure (not faked) any more.

2.2 The Applications with GUIs are in Danger

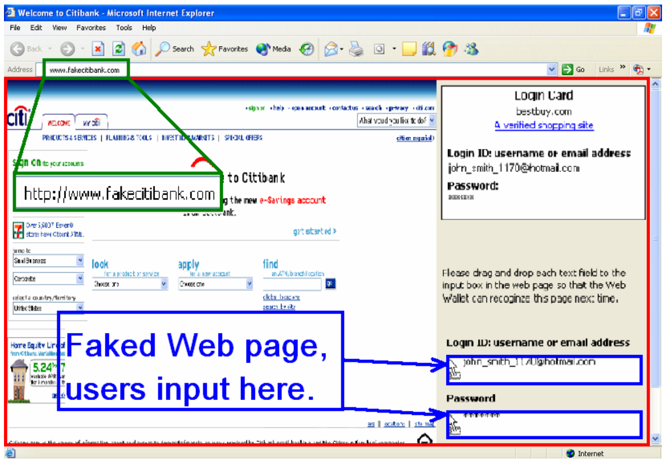
Most software products are using graphical user interfaces nowadays, including most of the security relative applications. Anti-phishing toolbars are using GUIs to do configurations, system maintenance, and phishing alerts' indications. Other new web security technologies other than phishing toolbars, such as *PWDHash* [4], *Dynamic Security Skin* [2], and *Web Wallet* [1] are also using GUIs to finish similar processes. The problem turns out that these applications are also under potential attacks from malicious people. They can just simply fake GUIs or cover the legitimated ones. So we need to design a counter measure against this kind of attacks. Figure 1 shows the examples of faked *TrustBar*, *Dynamic Secure Skin*, and *Web Wallet*.


Welcome to Citibank - Mozilla Firefox
Hey!
E*TRADE FINANCIAL
Identified by: VeriSign
http://www.fakecitibank.com

(a) Faked TrustBar


Welcome to Citibank - Mozilla Firefox
Password Manager
unique message here
Username
Password
Random number
New Image
Login
http://www.fakecitibank.com
Welcome to Citibank
Faked Web page, users input here.

(b) Faked Dynamic Secure Skin


Welcome to Citibank - Microsoft Internet Explorer
www.fakecitibank.com
http://www.fakecitibank.com
Login Card
bestbuy.com
A verified shopping site
Login ID: username or email address
john_smith_1170@hotmail.com
Password:
Please drag and drop each text field to the input box in the web page so that the Web Wallet can recognize this page next time.
Faked Web page, users input here.

(c) Faked Web Wallet

Figure 1 The demonstrations of faked TrustBar, Dynamic Secure Skin, and Web Wallet. The contents in red rectangles are faked areas.

3. ANTI-PHASHING TECHNIQUES TO BE USED

In this investigation, we use Web Wallet as an example to carry out UI design, user study, and implementation. We plan to use the following counter measures in a user study.

3.1 Referential UI for User Study

We provide referential UI, as shown in Figure 2, to make the other UI *anti-phAshing* improvements comparable. The difference from original *Web Wallet* design in [1] is that we use isolated window rather than embed one IE. Although theoretically every place in a computer screen can be faked, the spoofing activities still most probably happen in the client window of web browsers. So we simply try to build *Web Wallet* outside of web browsers.

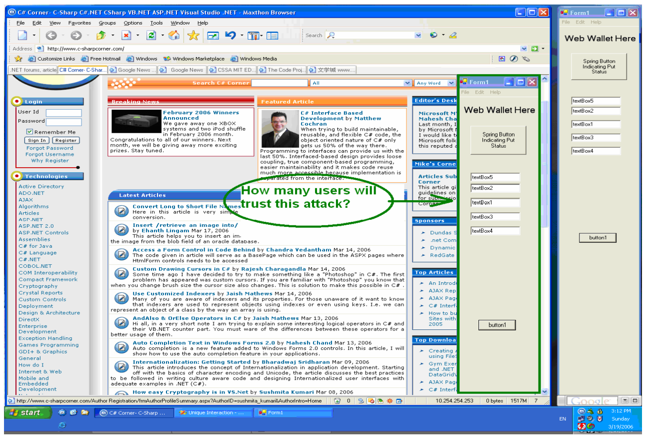
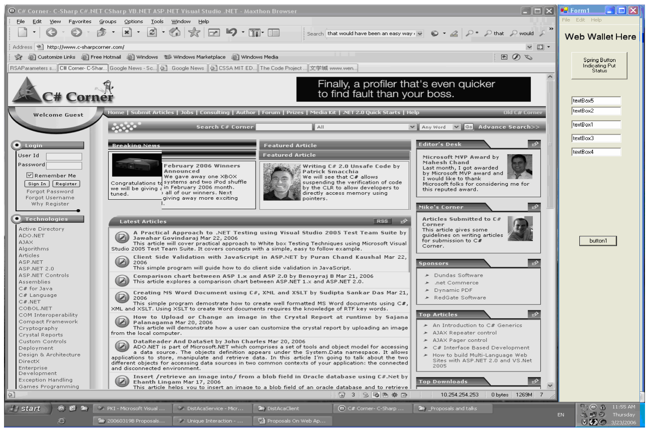

C# Corner - C-Sharp C#.NET CSharp VB.NET ASP.NET Visual Studio .NET - Maxthon Browser
Web Wallet Here
Spring Button Indicating Put Status
How many users will trust this attack?
button1

Figure 2 The Referential UI

3.2 Spring-Loaded Button and Disabling Other Applications

Following [1], it requires all of the security sensitive information be inputted to the web pages through GUI of *Web Wallet*. However in practical it is hard to prevent the users from exposing their security sensitive information to other websites. Blake Ross et al [4] first propose train users to press "*F2*" as the security key to input password in *PWDHash*. *Web Wallet* also uses "*F2*" as the security key to open GUI and require people to always remember to press "*F2*" before inputting security sensitive information. Maybe "*F2*" can work well for handling normal web pages but not for the faked *PWDHash* or *Web Wallet*. Is it possible to design the *PWDHash of PWDHash* or *Web Wallet of Web Wallet*? We can do this, but it will end to deriving new problems. The "*F2*" in *PWDHash* and *Web Wallet* only indicate "*to finish one task*". However what we really need are actions to indicate both "*start to do one task*" and "*finish doing it*". So we propose to use Spring-Loaded button. We use "*F2*" to indicate the "*start to input security sensitive information*" and use "*Shift+F2*" to indicate that "*finish inputting*" (or we can use "*F2 key down*" to indicate the start and "*F2 key up*" to indicate the end of input). In the meanwhile, only real *Web Wallet* can accept keystrokes, and all other areas will be disabled. Disabled areas will be set to gray.


C# Corner - C-Sharp C#.NET CSharp VB.NET ASP.NET Visual Studio .NET - Maxthon Browser
Web Wallet Here
Spring Button Indicating Put Status
Finally, a profiler that's even quicker to find fault than your boss.
button1

3.3 Application Trace

Web browsers use little lock-like-icons to indicate the *Secure Socket Layer* (*SSL*) [9] connections. *Dynamic Security Skin* [2] is trying to indicate the genuineness of web pages using hash visualization. Zishuang Ye et al [5] have proposed the trusted paths for web browsers. All these methods depend on users to detect and find out the difference between real web pages and the real ones through visual indicators. However Min Wu et al in [1][6] argue that users are not dependable simply because the web page validity verifications based on visual effect are not in the critical path of users objectives. We accept this concern. However despite this, we argue the three visual indicator based methods will not work under the assumption in Section 2.1. We want to redesign a visual indicator and call it *Application Trace*. To testify one visual area is real, we have to set somewhere in the screen is unfakible. We tried to put the unfakible indicator as an icon in taskbar, however soon we found it is not secure too, because the taskbar can be faked.

We propose to put the visual indicator at a fixed place of the physical area in computer screen, for instance, at one of the four corners of the screen or at one the four sides of the screen. Figure 3 shows a prototype that the visual indicator is put to the most upper side of the screen. We use animate hash visualization [7] [8] as the content in the visual indicator. We also add decoration to the *Web Wallet* with the same style of visual indication. The Web Wallet decoration in Web Wallet GUI should synchronize with the visual indicator. We need to make the visual indicator always appear at the most top of screen (Operating Systems can do this) such that the indicator itself can not be faked. Users always know the visual indicator appears on the top of the screen. So if the decoration in Web Wallet GUI can synchronize with the visual indicator then users will trust it. I have a personal experience that when I'm using wireless network. I always check the existence of the lock-like-icon in my web browser in case I'm inputting some security sensitive information into web browser. I want to make sure my information will not be eavesdropped by malicious people through the wireless network. Actually, if the lock-like-icon is faked, I may have no idea of whether my personal information is still secure or not. *Application Trace* can help a lot in this case.

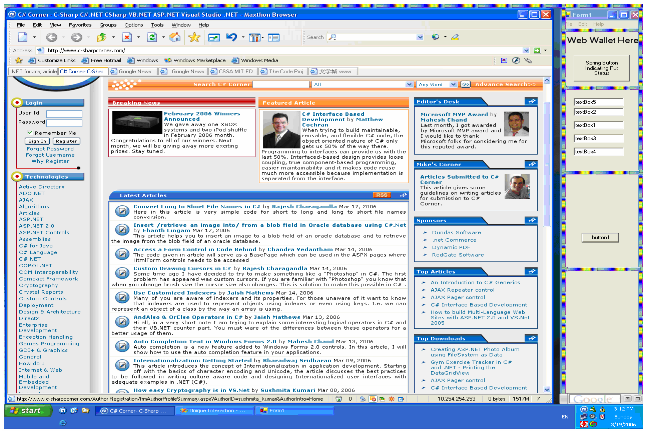


Figure 3 Application Trace Prototype

3.4 Genuine Skin

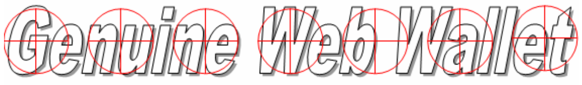
One way to detect faked *Web Wallet* is to use image recognition techniques to detect the visually similar appearances in screens to the real *Web Wallet* GUI. However the detection processes could be some complicated. There are difficult problems need to be solved before we can use this method. E.g. How to find the position of the faked GUI? What if malicious people adjust some items in the GUI to differentiate from the real one? What if malicious people only allow a part of the GUI appear in the screen when people will think it's still similar to the real one?

Genuine Skin is a method we would like to propose to mitigate the difficulty of automatic faked GUI detections while keeping the functionality of *Anti-PhAshing*.

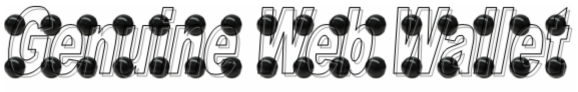
There are several guidelines to design *Genuine Skin*:

- *Genuine Skin* is like trade mark, it is identical for a certain company, and a series of products of this company can use the same *Genuine Skin*. We consider it as illegal to use the images identical or similar to the *Genuine Skin* of one company without authentication.
- There's one *Anti- PhAshing* engine running in the back ground and watch if patterns similar to the *Genuine Skin* appear in the computer screen while it is not in the known places. If similar pattern is found, the suspected area will be circled and alert will given out, as shown in Figure 6. If the faked *Web Wallet* doesn't have *Genuine Skin*, then users will not tend to believe it is real, as shown in Figure 7.
- The design of *Genuine Skin* should be both easily recognizable by both human eyes and computers. We need to train the *Anti-PhAshing* engine to simulate the average human classification curve, such that we minimize the total numbers of both false positive and false negative. Good design of *Genuine Skin* can help reduce the false cases.

Figure 4 shows two examples of *Genuine Skin* pattern designs for *Web Wallet*. Figure 5 shows the prototypes of using the two *Genuine Skin* patterns.

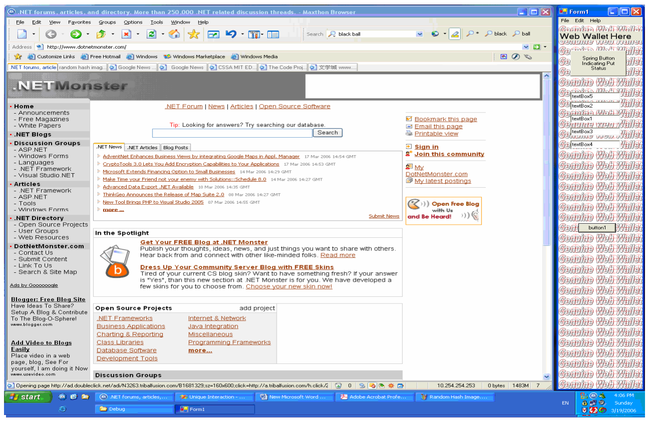


(1) Genuine Skin Pattern Example 1

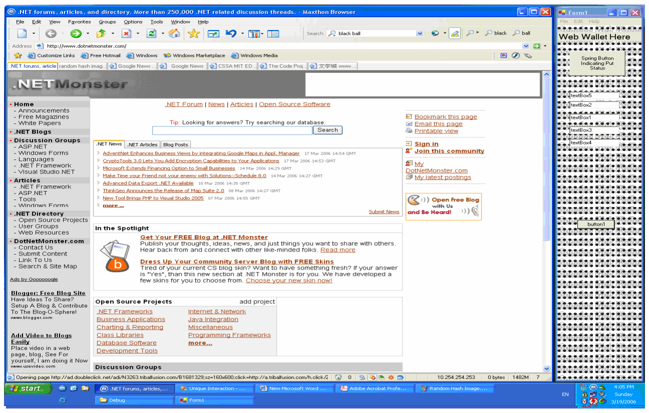


(2) Genuine Skin Pattern Example 2

Figure 4 Genuine Skin Patter Examples


.NET forums, articles, and directory. More than 250,000 .NET related discussion threads. - Maxthon Browser
.NETMonster
Form1
Web Wallet Here
Spring Button Indicating Put Status
button1

(1) Genuine Skin Prototype 1


.NET forums, articles, and directory. More than 250,000 .NET related discussion threads. - Maxthon Browser
.NETMonster
Form1
Web Wallet Here
Spring Button Indicating Put Status
button1

(2) Genuine Skin Prototype 2

Figure 5 Genuine Skin Prototypes (on Web Wallet)

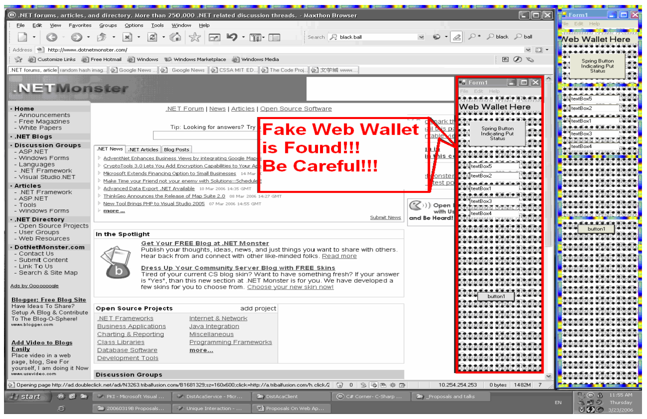

.NETMonster
Form1
Web Wallet Here
Fake Web Wallet is Found!!! Be Careful!!!
Spring Button Indicating Put Status
button1

Figure 6 Anti-PhAshing engine found suspected pattern and gives out alert.

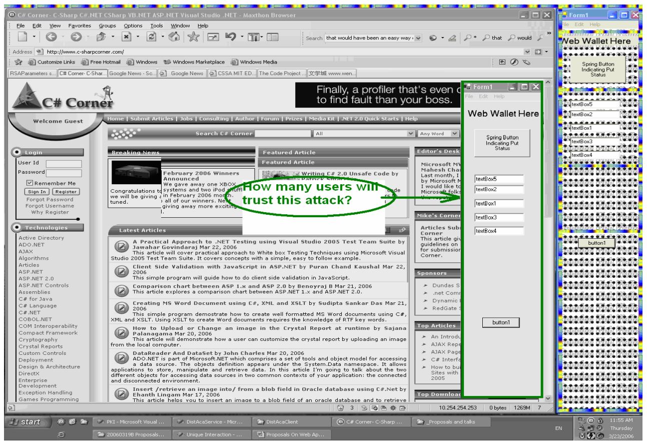

Finally, a profiler that's even quicker to find fault than your boss.
How many users will trust this attack?
Web Wallet Here
Spring Button Indicating Put Status
C# Corner
button1

Figure 7 Users can easily recognize the faked Web Wallet

3.5 Merged

The Anti-Phishing task is difficult and there's no single tool can prevent users from Phishing attacks at 100%. The truth is, if you use more anti-phishing techniques, you computer will be more secure. The same also happens to *anti-phAshing* tasks. The more techniques you apply to your computer, your computer will tend to be more secure from PhAshing attacks. So we would like to integrate all above techniques above together to carry out one user study to learn the effectiveness.

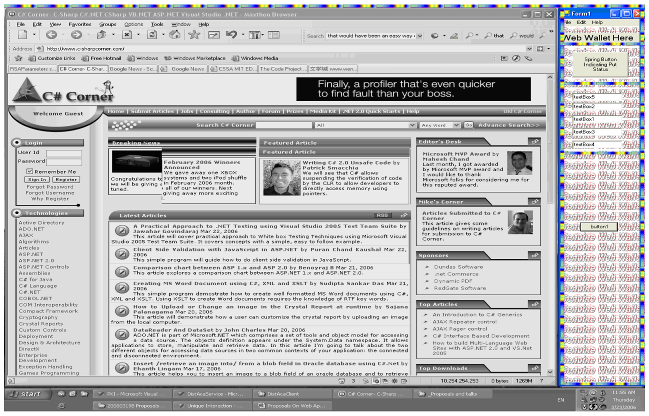

Finally, a profiler that's even quicker to find fault than your boss.
Web Wallet Here
Spring Button Indicating Put Status
C# Corner
button1

Figure 8 Prototype for merged anti-phAshing approach

3.6 Interactive Image Filter (User Challenge)

Another interesting approach we may want to try is the *Interactive Image Filer* (or *User Challenge*). Users can select personalized images (maybe can consider using the pictures that users are familiar with) as background of *Web Wallet*. When one user first uses the *Web Wallet*, he will be requested to select the personalized image and image filter. An image filter is an algorithm can apply to a given image to render or generate into another image with special visual effect. Figure 9 shows the example of a statue as background image and "*whirl like effect*" algorithm as image filter. When a user is typing into the *Web Wallet*, the predefined image filer will be continuously applied to the background image. Hopefully, the user will fill comfortable when they see the real image and image filter behavior, because they are configured by user. A *phAsher* must guess what kind of image and image filter the user is using. There's an assumption in [1] that all security sensitive information should be inputted through *Web Wallet*, such that each time the user inputs into *Web Wallet*, challenges from the user will happen. However, this approach relies heavily on user, because it totally depends on user himself to pick out the *phAshing* attack out.

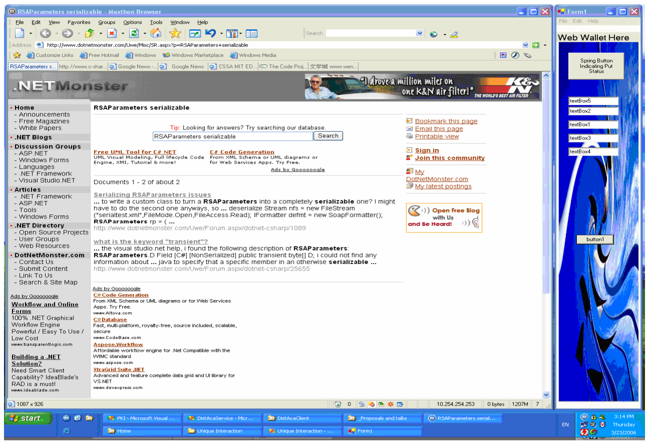

.NETMonster
RSAParameters serializable
Web Wallet Here
Spring Button Indicating Put Status
button1

Figure 9 Prototype for interactive image filter (user challenge).

4. EXPERIMENT/USER STUDY DESIGN

As introduced in Section 3, we have 6 *anti-phAshing* methods to be considered to carry out in this user study. We may recruit 10 users to test each type of *anti-phAshing* method. We may also design 40 "yes/no" questions for each user to answer. Each question will ask the user whether the given *Web Wallet* is trustable or not. In the 40 questions, we may include 10~15 faked *Web Wallet* cases, the rest ones are real *Web Wallet* cases.

References

[1]. Wu, M., Miller, R. C., and Little, G. Web Wallet: Preventing hishing Attacks by Revealing User Intentions, submitted to Symposium on Usable Privacy and Security 2006.

[2]. Dhamija, R., Tygar, J.D. The Battle Against Phishing: ynamic Security Skins. SOUPS 2005.

[3]. Herzberg, A. TrustBar: Re-establishing Trust in the Web. 2006. http://www.cs.biu.ac.il/~herzbea/TrustBar/

[4]. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J. Stronger Password Authentication Using Browser Extensions. Proceedings of the 14th Usenix Security Symposium, 2005.

[5]. Ye, Z., Smith, S.W., Anthony, D. Trusted Paths for Browsers. ACM Transactions on Information System Security. 8 (2): 153-186. May 2005.

[6]. Wu, M., Miller, R., Garfinkel, S. L. Do Security Toolbars Actually Prevent Phishing Attacks, to appear in Conference on Human Factors in Computing Systems 2006.

[7]. Perrig, A. and Song, D. Hash Visualization: A New Technique to Improve Real-World Security, in Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce

[8]. Dhamija R. Hash visualization in user authentication. In Proceedings of the Computer Human Interaction 2000 Conference, April 2000.

[9]. Netscape, http://wp.netscape.com/eng/ssl3.