Reliability, usability and security in anti-phishing software and its design

Linfeng Li

University of Tampere
Department of Computer Sciences
Computer Science
M.Sc. thesis
Supervisors: Eleni Berki, Marko Helenius
April 2008

University of Tampere
Department of Computer Sciences
Computer Science / Interactive Technology
Linfeng Li: Quality features in anti-phishing software design
M.Sc. thesis, 41 pages, 13 index and appendix pages
April 2008

Phishing, a typical online identity theft, has become one of the most critical threats of on-line business. Most victims are deceived by forging authorized websites. Being cheated by fraudulent websites does not only mean the loss of money, but also the damage of online trust relationship. From the whole economic and social system's point of view, the damage is not stopped at the point of destroying online trust relationship. In fact, the side effects of missing the business basis, trust relationship, will finally cause an economic slowdown. Undoubtedly, the final consequence will turn to be a tragedy.

In order to protect users from this simple cheating attack, we collected and classified three types of phishing attacks on client side, server side and transmission media. In addition, we also selected and carefully experimented, from usability perspective, four representative anti-phishing toolbars, Google Safe Browsing, Netcraft antiphishing toolbar, SpoofGuard, and my own software Anti-phishing IEPlug. Additionally, we employed misuse-oriented method to illustrate how to design phishing-resistant information system at design or requirement stage. According to the results of these studies, we suggest that end users should trust and use anti-phishing software to protect themselves. Moreover, for ordinary users, it is also highly recommended to observe and report any suspicious websites and attemps. Meanwhile, it is always a good habit to carefully check URLs and certificate authorities of online banking websites. From the results of my antiphishing studies, it also shows that the war between phishers and anti-phishers never ends. Phishing techniques are constantly evolving, as well as existing phishing preventive client side applications' own defects are not overcome yet. However, the study results are showing that phishing may be killed out at system design stage, e.g. by using misuse case method.

Key words and terms: software quality, phishing, phishing prevention, software design, malware

Contents

1. Introduction 1
2. Phishing & software quality 3
3. Phishing classification & existing phishing preventions 6
   3.1. Phishing classification 6
      3.1.1. Online banking user side attacks 6
      3.1.2. Network transmission media attacks 7
      3.1.3. Phishing attacks on the financial service side 8
   3.2. Prevention against phishing attacks 8
      3.2.1. Client side applications 9
      3.2.2. Server side applications 11
      3.2.3. Analysis of the existing anti-phishing applications 11
4. My own suggestions on phishing preventions 13
   4.1. Anti-phishing IEPlug 13
      4.1.1. Introduction 13
      4.1.2. Detailed design 14
   4.2. Heuristic usability evaluation of anti-phishing toolbars 18
      4.2.1. Detailed design and implementation of heuristic evaluation 19
      4.2.2. Conducting heuristic evaluation 22
      4.2.3. Results from the evaluation 23
      4.2.4. Statistics 29
   4.3. Misuse-oriented phishing prevention 31
      4.3.1. Misuse case methodology 31
      4.3.2. Misuse cases to design an online music purchase website 32
5. Conclusions and future work 36
   5.1. Known vulnerabilities & countermeasures of Anti-phishing IEPlug 36
   5.2. Suggestions on toolbar usability improvement 37
   5.3. Future work on usability evaluation of antiphishing preventions 39
   5.4. Conclusions on misuse case method against phishing and its future work ... 40

References 41

Appendices

1. Introduction

Phishing is one type of online identity theft. It usually uses a deceptive link or email containing fake web address resembling the authentic service. For example, the authorized website address could be *www.bank.com*. And phishers could imitate the address and register a website resembling the original one. The style and site structures may be the same as in the authorized one (for example, *www.securebank.com*). Careless users are likely to ignore the slight differences and provide their confidential information, like PIN code, credit card number or passwords, to this fake website. As a result, the personal confidential information is stolen and may be misused, and victims may suffer unnecessary financial loss or inconveniency.

Typically, it is not so difficult to detect deceptive phishing. Comparing the actual address with the intended one and checking certificate authenticity are the most efficient and easiest ways to prevent the attack [Emigh, 2005]. However, some e-commerce participators, especially the customers, are not sophisticated enough to recognize phishing or aware of the relevant basic protection. The unawareness of the relevant protection causes severs consequences in various aspects. First of all, phishing attack ruins the trust relationship in society. Phishing victims may misunderstand or misjudge that banks are not reliable to provide financial services or their online services. The direct consequence of abandoning online services is the enhancement of banking business cost. That might result in unpredictable impact of whole national economy. For example, without using online services, the same business transactions may take more man date and efforts than online business transactions. That means a customer pays more money but gets less or worse services. On the other side, phishing and other electronic crimes may also provoke each other, e.g. the new phishing techniques of compromising clients (i.e. personal computers) may be taken advantage of by worm creators to launch more massive attacks on the Internet, or vice versa. After emphasizing the sever consequence of phishing, it is imperative to illustrate how to prevent phishing attack.

In order to enhance users' capacity to identify phishing, software on the client machine would be very useful to teach new users some elementary security knowledge during daily commerce transactions. In the thesis, I try to find application design that will help users learn secure computing in electronic commerce. In order to make my solution convictive, one of research questions to be answered is how to design my own anti-phishing Internet Explorer plug-in software, Anti-phishing IEPlug. Different from the existing similar applications, this one is able to make users protect their own online banking web sites, and even improve the existing Secure Socket Layer (SSL) by actively showing the Certificate Authority (CA).

Even though some other organizations have implemented anti-phishing applications as well, especially Internet browser toolbars, for identifying deceptive activities, phishing is not stopped efficiently. One reason is that phishing keeps evolving continously. Another reason may be from anti-phishing toolbars themselves. Detection rate of these applications, one of key metrics of evaluating anti-phishing toolbars, is not satisfactory [Cranor *et al.*, 2006]. Moreover, the level of usability and user interfaces varies. Some of the toolbars have obvious usability problems can ultimately affect the performance of these toolbars. For the sake of the future improvement, usability evaluation is indispensable. In this thesis, we answer the research question how to design a dig out and what are the usability problems of typical anti-phishing toolbars: Google toolbar, Netcraft Anti-phishing toolbar, SpoofGuard. In addition, we included the Internet Explorer plug-in, Anti-phishing IEPlug.

My hypothesis was that usability of anti-phishing toolbars-and in consequence also security of the toolbars-could be improved. Indeed, according to the heuristic usability evaluation, a number of usability issues were found. In the article, I will describe the anti-phishing toolbars, I will discuss anti-phishing toolbar evaluation approach and I will present our findings. Finally, I will propose advice for improving usability of anti-phishing toolbars, including inducing three key components of anti-phishing client side application (main user interface, warnings and help system). One result of my research is also a classification of anti-phishing toolbar applications.

In addition to studying existing anti-phishing toolbars, I am also asking questions: are there any other better ways to solve the phishing threat? Is it possible to stop phishing at design or requirement stage? To answer the questions, I employ and illustrate misuse case method to design phishing-resistant information systems.

The following chapters of the thesis are organized as follows: the second chapter presents the essential quality features, concepts, and definitions in phihsing and anti-phishing software design. In addition, I also briefly introduce what aspects of the anti-phishing research I am focusing on and the reason why I am doing these studies. The next chapter after that gives the classification of phishing attacks. After classifying phishing attacks, I also reviews and classifies the typical prevention applications against phishing attacks. My own suggestions and the answers to the research questions are presented in the forth chapter. Those readers who are interested in the detailed design, development and implementation of my own anti-phishing software, Anti-Phishing IEPlug application, may look through the first section of the 4th chapter, where logical structures and algorithms of each module in this application are discussed. In the second section of the forth chapter, I describe the heuristic evaluation methodology and present the evaluation results I found. As my another suggestion, I introduce the misuse-oriented methods and give a group of sample misuse cases for preventing from phishing attacks. In so doing, I aim at the demonstration of the method's applicability

and design strengths. Based on my research results, I conclude the study and summarize the future work in the last chapter. In this chapter, the analysis of my Anti-phisihing IEPlug design's vulnerabilities and countermeasures is given. Besides I also give advice for improving the toolbar's usability design. In conclusion, the usability evaluation is summarized, and the impact of poor or weak usability performance of the toolbars is discussed. Moreover, for future research, I underline misuse case method's potential to serve as a proof-phishing thinking and technical tool. Its utilisation could advance conceptual understanding and cognitive thinking for metamodelling and could enrich software quality assurance techniques in the early analysis and design phases.

All in all, I believe that my work in this thesis can serve as a reference guide to software developers and any e-people/end-users who want to find a set of simple, understandable and practical new knowledge on phishing, anti-phishing techniques and concepts. This concise, recent collection of updated, subsequently simplified and classified information could also serve as a safety guide for novices and inexperienced users while interacting in virtual communities.

2. Phishing & software quality

The spread of phishing intensifies the need for well-defined security requirements in the design of an information system. Phishing goes on increasing, especially in the e-services domain, even though a variety of prevention methods have been developed and used against it. Phishing attacks compromise the software quality features of a system.

Berki *et al.* [2004, 2006] and ISO [1991] described that software quality is not a set of essentially wanted and desirable features that can be added to a system after its realisation; Berki and Georgiadou [1996, 2003] also stated that software quality features, and especially those that deal with functionality, are planned and designed from the very initial phases of the software development lifecycle. Dealing with system functionality, ISO 9126 defines functionality as a set of attributes that bear on the existence of functions and their specified properties in the articles of Berki and her co-authors [1996, 2003, 2004, 2006]. The functions are those that satisfy stated or implied needs; therefore, they must be and prove to be suitable, accurate, and secure and with certain interoperability features [see e.g. ISO, 1991]. Evidently, many existing information systems do not bear these characteristics no matter what and how software development methods, tools and quality models are used, for example, in Berki's papers [2004, 2006]. Another software quality feature, usability, deserves to be considered very carefully in anti-phishing research. According to the definition of ISO 9241-11, usability is depicted as follows:

"The effectiveness, efficiency, and satisfaction with which specified users achieve specified goals in particular environments."

Effectiveness refers to as the accuracy and completeness with which specified users achieve specified goals. Efficiency means the resources expended in relation to the accuracy and completeness of goals achieved. Satisfaction is the comfort and acceptability of the system to the users and other people affected by its use. Those three key aspects require sofware to have learnability, flexibility and robustness. Learnability makes users start effective interaction and conduct maximal performance. Flexibility mentions that system offers more than one way to interact with users. Robustness highlights the recover capability when malfunctioning or errors happen.

Design quality issues regarding system's functionality (security in particular) and usability (operation in particular) are still not holistically considered by information systems designers [Berki *et al.*, 2003]. As a result, an inaccurate and not precisely designed system will not bear the security required. In web-based information systems, for instance, virtual identities are required for interaction [Jäkälä and Berki, 2004]. Virtual *identity theft* is a frequent phenomenon, which is not new; it has, though, become a problem haunting people's daily lives. Identity theft is a very general term, which can further be categorised into many sub-classes based on the media it takes advantage of when stealing the identity used for communication. Identity thieves can profoundly exploit a number of insecure transmission tools including telephone, mail, email and various websites. Apparently, the most convenient and significant technique to steal someone's identity is using the Internet, that is email and website; this technique is called phishing [APWG, 2006].

In phishing attacks the aim is to steal the users' confidential information (e.g. credit card number, password, PIN code etc.) by *social engineering* and *technical subterfuge* [Wikipedia, 2006]. According to the definition from Anti-Phishing Work Group (APWG), these concepts of phishing are described as follows:

"Phishing attacks use both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. Social-engineering schemes use 'spoofed' e-mails to lead consumers to counterfeit websites designed to trick recipients into divulging financial data such as credit card numbers, account usernames, passwords and social security numbers. Hijacking brand names of banks, e-retailers and credit card companies, phishers often convince recipients to respond. Technical subterfuge schemes plant crimeware onto PCs to steal credentials directly, often using Trojan keylogger spyware."

Those emails and websites, used - or rather abused - by social engineering and technical subterfuge, impersonate the authentic ones, normally including the same website layouts and logos, and even similar domain names. All these basic design characteristics and virtual features are imitated so well that the majority of the end-users can hardly distinguish between them. In addition, due to the lack of effective

fraud detection techniques, a great number of inexperienced email and website end-users' identities are compromised under the threat of theft and fraud.

According to information obtained from the records of APWG, the number of phishing attack reports reached 18480 in March 2006 [APWG, 2006]. Although there is no economy loss specified in the report, one could (not!) imagine how much that could be. Based on a report by Javelin Strategy and Research on April 2006, the economy loss reached 20 billion [Javelin, 2006]. Most likely, the veracity of the figure might be argued. Nevertheless, phishing seriously challenges and collapses the trust to electronic commerce and e-services security. Less and less users feel secure and, as a result of this insecurity in e-services, they might stop using otherwise convenient online services, since they are not sure whether their credentials are in danger. Therefore, the questions on (i) how to identify the fraud, and (ii) how to design and build a reliable and secure environment for business transactions have become the most imperative requests of this research field.

So far, there has been some significant, albeit not adequate, progress in this field that has taken into account both the clients' considerations and the servers' design quality features. On the client side, there have been more than eighty (80) types of user-centred applications developed; these are automated techniques such as browser toolbars and plug-ins. Meanwhile, more and more researchers on the topic of security realise the need for improving server security, in order to holistically protect against phishing by considering both the client and the server [Stop-phishing, 2006]. Regarding the latter, there are two typical outcomes: One is the *phishing email filter* on the email server developed by Carnegie Mellon University [Kumaraguru, 2006]. This filter was designed with *learnability* as the main software quality feature in mind; this means that the filter contains *self-learning capacity* and intelligence to detect and identify the phishing emails. The other one is a *light weight trust architecture* designed by Massachusetts Institution of Technology [Chau, 2005]. The key software quality feature of this new platform is the light weighted self-certificate for *verification*; this design quality characteristic can verify each other's identities in private communication situations.

Undoubtedly, all these research efforts and outcomes are quite helpful for fighting against phishing attacks. However, the performance (or efficiency) and usability of these tools are not satisfactory [Cranor *et al.*, 2006] and, as far as I know, no research focuses on how to build a *phishing-proof* system from the system's design point of view. Therefore, in my thesis, the study mainly focuses on how to design whitelist based anti-phishing software, how to evaluate usability of phishing preventions on client side. Besides, I look at conceptual design and technical design issues for such a phishing-proof system and explain how, after the requirements analysis stage, an information

system can be designed, validated and documented against phishing, with the *misuse case method*.

3. Phishing classification & existing phishing preventions

3.1. Phishing classification

In order to assume proper, or at least reasonable, misuse cases when defining the security requirements at design stage it is necessary to have access to updated information and possess suitable design knowledge. Otherwise, there is a need to collect, analyse, understand and finally classify the existing phishing attacks. There are various classifications available. Jakobsson [2006], for instance, grouped phishing attacks in terms of various entities involved in. While, Helenius [2006] analysed and classified the phishing attacks based on the different techniques used for phishing. These include the following: (i) user request, (ii) redirection, (iii) user-end hijacking, (iv) banking side hijacking, while the "co-operation" of some of these techniques is also possible.

I, hereby, define my own classification criteria considering different entities that apparently are the current popular target of phishing attacks. These are: a. online banking users, b. transport media, and c. financial institutions' servers. Before presenting our full classification on the latter in later section of this chapter, I proceed to a brief description of these entities and domains below. The following sections 3.1.1 – 3.1.3 provide, among other information, an analysis on the threats and vulnerability of the entities in these domains and on how security is compromised.

3.1.1. Online banking user side attacks

Online banking user side phishing attacks are very common, compared with the other two target entities. The reason why phishers prefer to aim at end-users is that they can be easily cheated or spoofed without using professional techniques. These attacks are very successful regardless the level of experience or inexperience that the end-users possess.

The most common and easiest phishing on the client side is the *man-in-the-middle* attack. Phishers mislead and spoof users with emails or phishing pages. When victims reply to them with confidential information enclosed, the information is collected by the phishers. After that, phishers are able to log into the victims' accounts. According to the sources of emails, phishing emails could be classified into two different types.

1. The first one is from the phony banking institutions. Usually, these emails come with an announcement like the next: there is bait, which says there is something wrong with the account, or a bonus is distributed, or a competition takes place and so on. Credentials are asked for confirmation. This may be

elusive even for experienced users, as one would not like to miss something financially important.

2. The second type of the spoofing emails seems to be coming from the friends or relatives of victims. This is called *social phishing* [Jagatic *et al.*, 2005]. This kind of phishing email takes advantage of the trust relationships between people and their acquaintances. Because of the *trust relationship*, victims may be convinced. Moreover, this kind of phishing emails spread much faster than in the first type. However, deceptive emails are not the end. To harvest confidential information, there are two basic ways. One is via email itself, and the other is by offering an URL of a fraudulent website, which looks the same as its authentic website (with the similar logo, website roadmap, page layouts, and domain name). Apparently, all these are difficult to test and identify; testing for security and authenticity is another research milestone to be reached in software development.

A more technical attack resorts to ActiveX or other explorer plug-in techniques. The most notorious plug-in is a *keylogger*, which can record the key pressing and send the recorded key strokes to the attacker stealthily. Obviously, these types of attacks are more dangerous, since the potential victims are not aware of the attack. Moreover, due to the fact that end-users are not necessarily equipped with that specialised computer science or IT knowledge, it is extremely difficult for novice and occasional end-users to detect keyloggers. Similarly, another technique named *screen logger*, also collects the movements of a mouse cursor. For the customers using on-screen virtual keyboard, screen logger exposes them to phishing.

Undoubtedly, these malicious plug-ins are not the only form of technical phishing attacks on the client side. The traditional malware can also be employed here. When a computer virus, worm or software Trojan horse controls an operating system, any confidential information can be divulged and the whole appearance of the system can be changed.

3.1.2. Network transmission media attacks

Hacking *network transmission media*, literally, means that attacks are launched towards network transport media, such as routers, switchers, and especially DNS (Domain Name Server) servers. These core network infrastructures are valuable for Internet criminals. It can be a nightmare, if phishers control some of these transport media. In practice, phishers aim to redirect user's TCP/IP (Transmission Control Protocol /Internet Protocol) requests to a preserved forged website by hijacked DNS servers[Igor, 2005]. These compromised DNS servers are still difficult to detect manually.

3.1.3. Phishing attacks on the financial service side

Fortunately, there is a limited number of reports about the fact that the servers of financial institutions have been compromised so far. However, hacking online banking servers is still a possible threat. Most of the popular server side attacks can also be used by phishers, including SQL injection attack and PHP injection attack. These attacks are still a risk for wrongly configured servers.

3.2. Prevention against phishing attacks

We found that, there exist currently four basic types of toolbars, classified by their architecture and functionalities.

1. Toolbars based on client-server architecture and combined anti-phishing prevention with other functionalities. These types of toolbars need to communicate with their servers, in order to protect users from being spoofed. However, these kinds of toolbars are not tailored just for phishing prevention.
2. Toolbars based on client-server architecture and only designed for phishing prevention. These are also based on client-server structure, but the functionality is only phishing prevention. Therefore, users can only find phishing related functionalities from their interfaces.
3. Toolbars deployed on local computer and detecting spoofing websites by browsing records. Because of the lack of data service on the server side, these kinds of toolbars have to use the browsing records or history of hosted browsers.
4. Toolbars deployed on the local computer and detecting spoofing websites. Different from the previous type, those toolbars must use some other methods to identify spoofing websites, like a whitelist[1] or general detection.

In addition to these existing toolbar types, we observed that the classification can be developed further. Therefore present techniques could be combined when developing toolbars further. The classification can be based on differences in architecture, detection method and identification mechanism. Some classification variables can be:

- Is the toolbar client-server based?
- What types of lists the toolbar uses for detection (blacklist, whitelist and/or graylist)?
- Does the toolbar use local history/cache information for phishing site detection?

In this section, I chose three toolbars, in addition to our own. I am aware that there are also other toolbars. However, because of limited time and resources, I picked one

[1] Whitelist (in phishing prevention) is a list of addresses or domain names that are known to belong to authentic services.

typical toolbar of each existing type. I selected these toolbars according to two criteria. First, the selected toolbars must be common ones and be downloadable from the Internet. Second, their performance should be satisfactory. Referring to the outcome of Cranor's article, we selected Google toolbar, Netcraft toolbar and SpoofGuard, which received highest scores in performance evaluation [2006]. The following sections provide my detailed classification with a commentary on the basic quality features that can be found in each of them.

3.2.1. Client side applications

Nowadays, most prevention methods concentrate on the most vulnerable client side. For example a prevention application is installed on a personal computer to assist the user while identifying fraud attempts. In general, the most widely used application to fight against phishing is a browser toolbar, which is embedded into an Internet browser. According to the toolbar implementation architecture, toolbars can be divided into two types: One is based on the client-server structure, while the other works only on a personal computer without any anti-phishing server.

1. *Client-server structured applications.* These kinds of toolbars are normally deployed both on a client machine and on a server. The toolbar requests the server for regular updates and maintenance frequently. In this regard, this type of toolbars is normally developed and released by commercial companies, such as Google, Netcraft and Microsoft.

   For example, Google Safe Browsing (Figure 1), is a part of the Google toolbar extension for Firefox. It is able to alert users, when the web page visited is judged as a fraudulent one. Google Safe Browsing blocks the visit of web pages by using a blacklist. According to the introduction on the toolbar's download page, the blacklist is generated and maintained by a server hosted by Google [Google, 2007]. In order to determine the authenticity of a web page, the toolbar sends the visited URL to the server, when a user tries to browse any web page. If the URL is considered as a deceptive one, the toolbar will stop the user's activity and give appropriate advice (e.g. stop visiting the fraudulent web site).

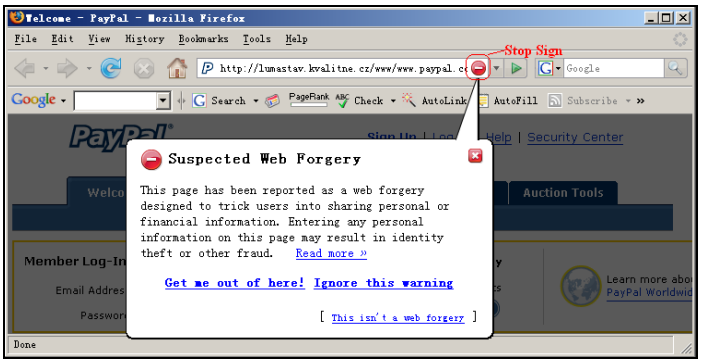

Figure 1. Google Safe Browsing toolbar, when a phishing page is detected.

The mechanism of the Netcraft Anti-Phishing toolbar (Figure 2) is similar to the Google's toolbar. It communicates with the Netcraft site's report database [Netcraft, 2006] and obtains the blacklist and whitelist information. Moreover, the toolbar offers extra information concerning the page a user visits. For example, when a user visits the Netcraft site report page, he or she can be aware of the site ranking, following the link on the toolbar. However, this ranking is based on the level of popularity, rather than security criteria. Moreover, on the toolbar, users can clearly know where the current website is hosted (in the Figure 2, it is hosted in U.S.). This design is considered to be helpful for users, because it is common sense that American Express cannot be hosted, for example, in Middle East countries.

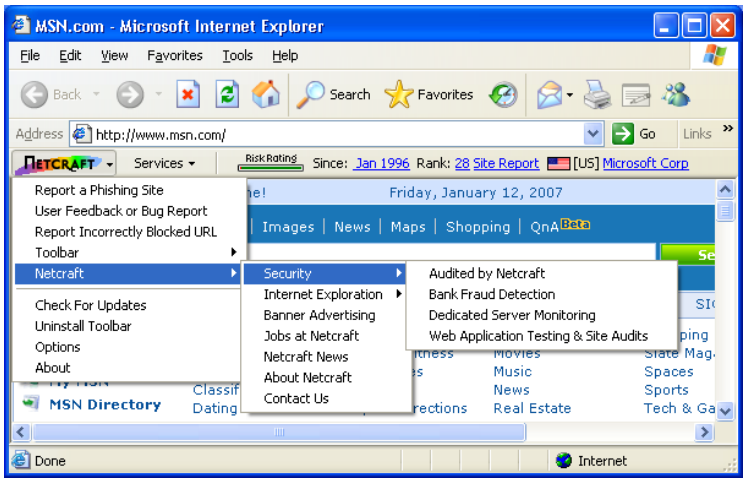

Figure 2. Netcraft anti-phishing toolbar

2. *Independent toolbars.* Compared to client-server toolbars, independent toolbars identify a deceptive website based only on the data stored on a personal computer. The general mechanism of this type of toolbar is like this: After a web page is downloaded to a personal computer the toolbar will compare the characteristics of this web page to with the previously saved data, e.g. domain name, image harsh and so on. If any differences are found, the toolbar will warn the user and give some suggestions.

SpoofGuard, shown in Figure 3, is the outcome of a study that took place at Stanford University. The toolbar is compatible only with Microsoft Internet Explorer. Compared with previous two toolbars, this one uses a kind of *whitelist*: the browsing history of the Internet Explorer. There are three buttons on the toolbar, which are next briefly described: one acting as 'traffic light' for indicating the security status of the visited page, another as 'hammer' for configuring the settings, and a third acts as 'crossing' for removing all data

collected by SpoofGuard, that is image hashes and password hashes. SpoofGuard is able to identify fraudulent web pages by checking the browsing history as well as other information, including domain name, URL, email, password field, image and links on the visiting page [SpoofGuard, 2006]. Moreover, SpoofGuard can alert a user, when he/she submits the same ID and password to different web pages. When a warning shows up, two choices are given: to 'continue' or 'stop visiting'.

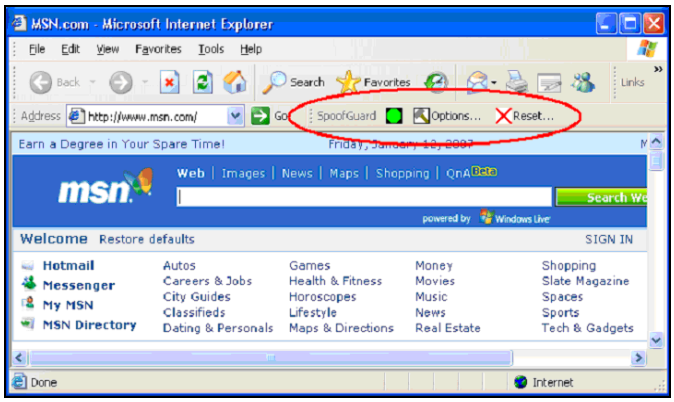

Figure 3. SpoofGuard is circled in red

3.2.2. Server side applications

The term 'server side' here does not only mean the servers in financial institutions. Instead, it also includes servers running on the whole network.

One of the available applications focused on the server side is, as already mentioned in the Introduction part, the phishing email filter, which is developed in Carnegie Mellon University [Kumaraguru, 2006]. This application is deployed on the email servers. The basic detecting mechanism is similar to the traditional spam filters. However, the most significant software quality feature of the filter is that it is equipped with the capacity of learning (learnability). This means that the application can improve its intelligence-based detection capacity during and after many emails detection. This is undoubtedly a beneficial characteristic utilised for a phishing attack. Because phishing emails do not follow any specifications or rules, to effectively detect them is a big challenge for the non-intelligent applications.

3.2.3. Analysis of the existing anti-phishing applications

These anti-phishing applications are helpful in fighting against phishing. However, the performance of these applications is not so satisfactory. According to the results of an evaluation test from Carnegie Mellon University [Cranor *et al.*, 2006], some toolbars are not that ideal to realize the goal of phishing prevention. In the test, two (2) out of five (5) toolbars with best evaluation score can only successfully identify eighty per

cent (80%) of the pages that phishing takes place. The other toolbars have even worse performance, since they are just able to detect only forty per cent (40%) of these pages.

In fact, as discussed in the previous sections, phishing is not the same as traditional viruses or other attacks. Attempting a cognitive association, we may state that phishing actually takes advantage of the security breaches in end-users' thinking model. Examining this further, we construct the following simple scenario:

1. A user wants to confirm the password of his or her bank account.
2. The user logs into the system, and then gives what the phishing web page asks for.
3. After that the user clicks the submit button on the page.

These three simple steps can easily be conducted by any people. Usually, people are familiar with a similar scenario, which is when they want to change their bank account password in the bank's webpage. Moreover, the user interface of the bank's webpage and the phishing webpage may be the same.

In the real world, the bank users need to contact bank clerks, who can be identified from their appearances and the office in the bank place. In this case, the bank users just need to verify who is managing their accounts. However, in the virtual world, this trust relationship between the service and the customer is not the same any more. The e-Customers make their e-transactions with the help of web pages, the Internet and computers. The appearances of these objects can easily be forged. Therefore, if the bank or any other e-users keep following the same thinking model to check the e-service's outlook (appearance), the breach is obvious: the users are exposed at the unawareness of the identification in online banking and other online services.

Even though the existing toolbars can assist users' safe browsing, they hardly change the mental model in users' minds. Thus, the designers of phishing prevention applications should find solutions to enhance functionality and reliability of the IT services from the system design point of view. At design level, the software and system design explicitly stated requirements normally include the description the demands and conditions for the final system's secure transaction procedure. Moreover, the design style and design method could cater for time behaviour and resource behaviour and enforce users to behave in a more secure and stability-oriented mental model, for example by using the one-time password. Of course the question of how to design a robust system to prevent from phishing requires catering for more than for the one-time password. Information systems should be planned and analysed well at the requirements planning and analysis stage. At this stage, designers could elicit essential security requirements against phishing. In the next chapter, I introduce my suggestions on phishing preventions.

4. My own suggestions on phishing preventions

In this chapter, my own suggestions on phishing preventions are carefully illustrated. The first section introduces the design of my own antiphishing toolbar, Anti-phishing IEPlug. The next section presents how to design and what are the results of a usability evaluation on four introduced toolbars. The last section is focus on the misuse case method and its application in anti-phishing field.

4.1. Anti-phishing IEPlug

4.1.1. Introduction

According to the usability evaluation result from Downs *et al*. [2006],

"only 40% of those who were aware of the lock realized that the lock had to be within the chrome of the browser."

This shows that even though a lock icon acts to ensure the secure connection, users still possibly confuse the technical issues of the lock icon: where and how to check this lock icon. Inspired by assisting users to correctly use the lock icon, we designed and developed Anti-phishing IEPlug.

Anti-phishing IEPlug (Figure 4) is a whitelist based on Microsoft Internet Explorer plug-in against phishing web pages, implemented by the authors of this paper at the Department of Computer Sciences at the University of Tampere [IEPlug, 2006]. This plugin is able to show the *certificate and authority* of web pages containing a password field. The certificate is shown after a user has added the domain name to the whitelist. The toolbar offers an interface for maintaining the domain names including adding, editing and removing with accuracy. Only a limited number of users can add domain names to the list, while administrators can maintain the list. The toolbar also has a limited capability to warn about fraudulent web pages. The detection is based on a keyword search from the address(es) visited.

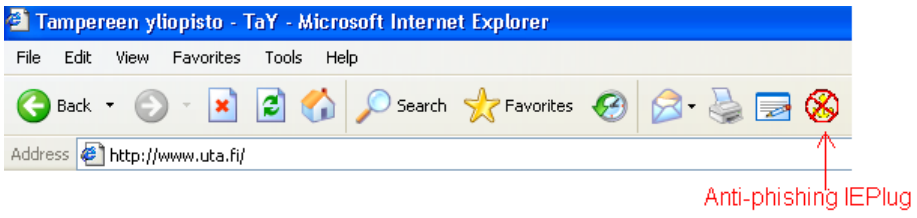

Figure 4. IEPlug button on the toolbar of IE browser

The development platform of this IE plug-in is Visual C++ studio 6.0, including its GUID generator (guidgen.exe) to generate global unique identifier for this application, and Windows XP. The techniques used in the development contain Browser Help Objects (BHO), IE programming [Kirants, 2005], permutation encryption method and block check characters (BC or BCC). In Kirants' article [2005], IE programming associated with the BHO technique was introduced. BHO is a special plug-in used for

specified functionalities, like blocking preserved websites. In IE programming technique, IE browser was considered as a large container with several indispensable interfaces for communication with plug-ins, like BHO (e.g. SetSite() is used for connecting the BHO with the IE browser, Invoke() is used for retrieving parameters from the IE and for dispatching commands to the IE). In the encoding part, traditional permutation is applied, which permutates each character's ASCII code with specified arguments into a new ASCII code. BCC, which contains the special pre-set characters, will be encrypted and saved together with the encoded content to the Anti-Phishing.txt. Integrity of the file can thus be verified by BCC later on. The more detailed development is discussed in the next section.

The program's general structure and data flow are described below (Figure 5):

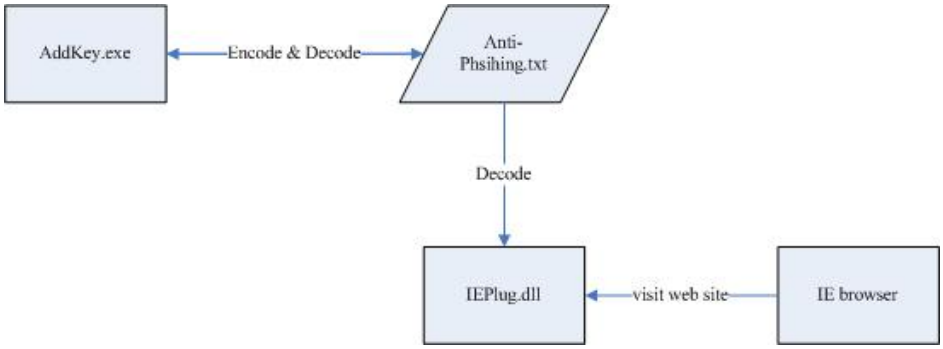

Figure 5. Data flow diagram of Anti-phishing IEPlug

Obviously, according to this kind of design, there are at least two modules. One is AddKey, which is used for the maintenance of the Anti-Phishing.txt file. The other one is a dynamic linked library (IEPlug.dll), which is used for monitoring users' visiting activities.

Herein, for the sake of assuring the integrity of the Anti-Phishing.txt, the critical data saved in the file should be encrypted. In the following section, the detailed design and development idea and its original intention are elaborated.

After successful installation, the only available user interface is the dialog used for maintaining domain keywords. The program AddKey.exe manages the domain keywords. IEPlug's main task is just to monitor the visiting web address. If the visited webpage contains a password text field and the address contains the keyword saved in the Anti-Phishing.txt, IEPlug will alert the user and give advice. Based on this intention, the two modules were developed (AddKey.exe & IEPlug.dll).

4.1.2. Detailed design

As mentioned in the first part of the article, there are seven different operations allowed for administrators and limited users. Administrators have the privilege of every operation, but limited users can not change the domain names, which are saved before. There are two reasons for this design. On the one hand, it would be inconvenient to rely

only on administrators for adding keywords. On the other hand, even though limited users are able to add keywords, this will not increase the risk of being compromised. This is because the Certification Authority of every website, whose domain name is saved as a keyword, will be verified in order to check the authenticity.

When the AddKey.exe is launched, initialization will be called automatically. In this initial procedure (Figure 6), current user privilege will be determined by trying to create a register key and trying the program's operations available only for the administrator. Afterwards, the program will check the integrity of the Anti-Phishing file. The integrity of both the access properties and the content of the file will be verified. The access properties verified include whether every user has rights to read and write data associated with the content of the file. The content integrity is checked by comparing the preserved BCC (here is AKBCH) with decoded BCC which is elicited from the Anti-Phishing.txt file. The decryption algorithm employed is traditional permutation, and its arguments for permutation are the encrypted text content in the Anti-Phishing file and its length. The function CAddKeyDlg .CheckSeInfo() will be responsible for testing properties of the file. If the preceding steps in initialization function are accomplished successfully, the domain keywords in the content of the Anti-Phishing.txt will be decrypted and added to the control list one by one. The algorithm of decryption is also primitive permutation like BCC used, but the arguments used are one preserved password and the length of the encrypted text.

After initialization, users are able to operate the domain key words with available operations.

The add operation is not just in charge of adding the website domain but also to verify the link present. This verification checks whether the given link contains any illegal characters. The legal characters in this application are only letter characters, because the website domain of bank should not contain any other characters, like numbers or symbols. After that, the verified website domain name will be added to the control list.

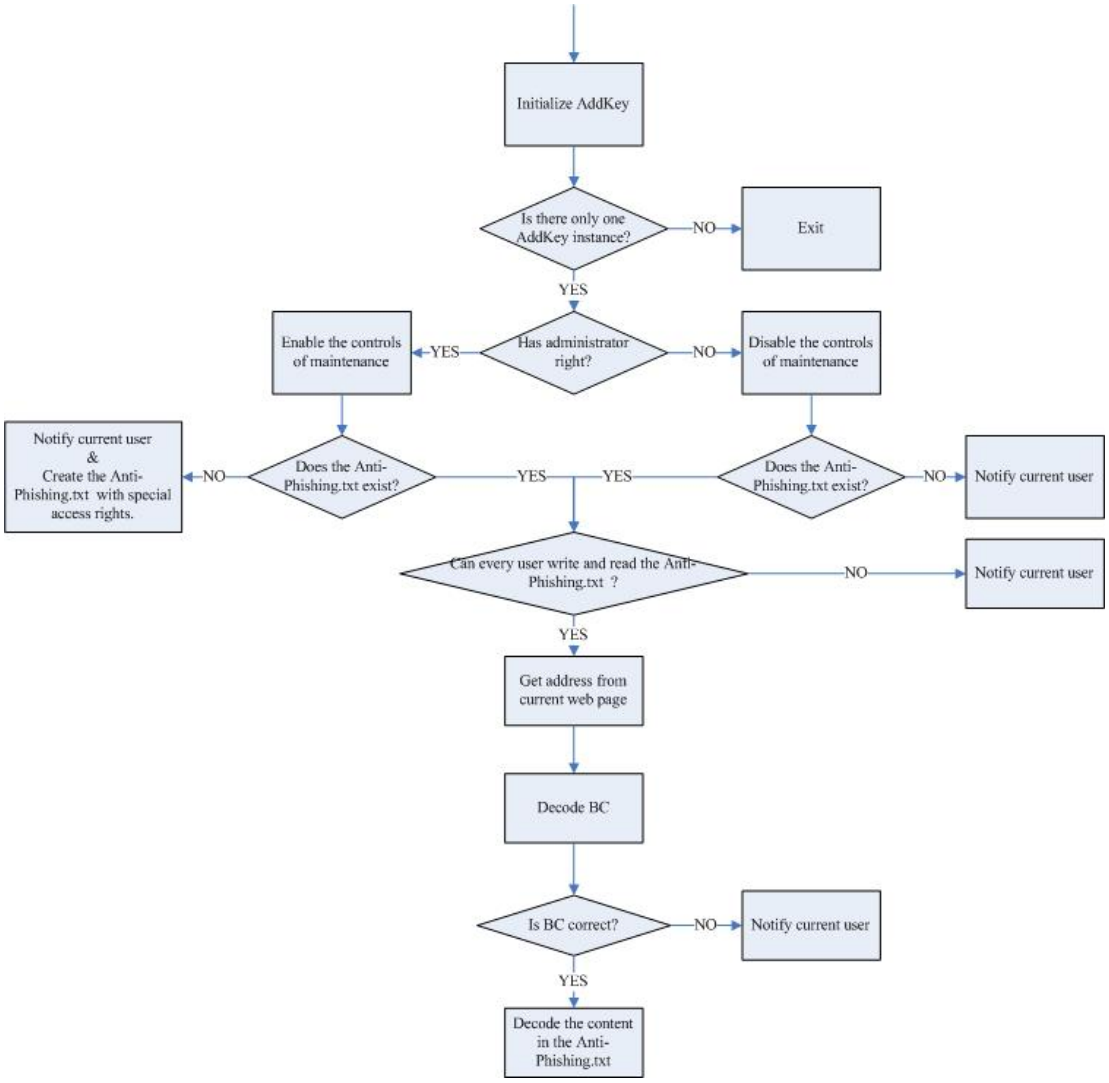

Figure 6. Initialization

Administrators are able to edit, remove or update the specified items from the control list. When any user closes the AddKey dialog, every item text in the control list associated with its encrypted BCC will be saved to the Anti-Phishing.txt (Figure 7). If users click the Undo button, then the control list will be updated based on domain names in the Anti-Phishing.txt. In other words, the key words added or maintained before saving to the file will be removed from the control list.

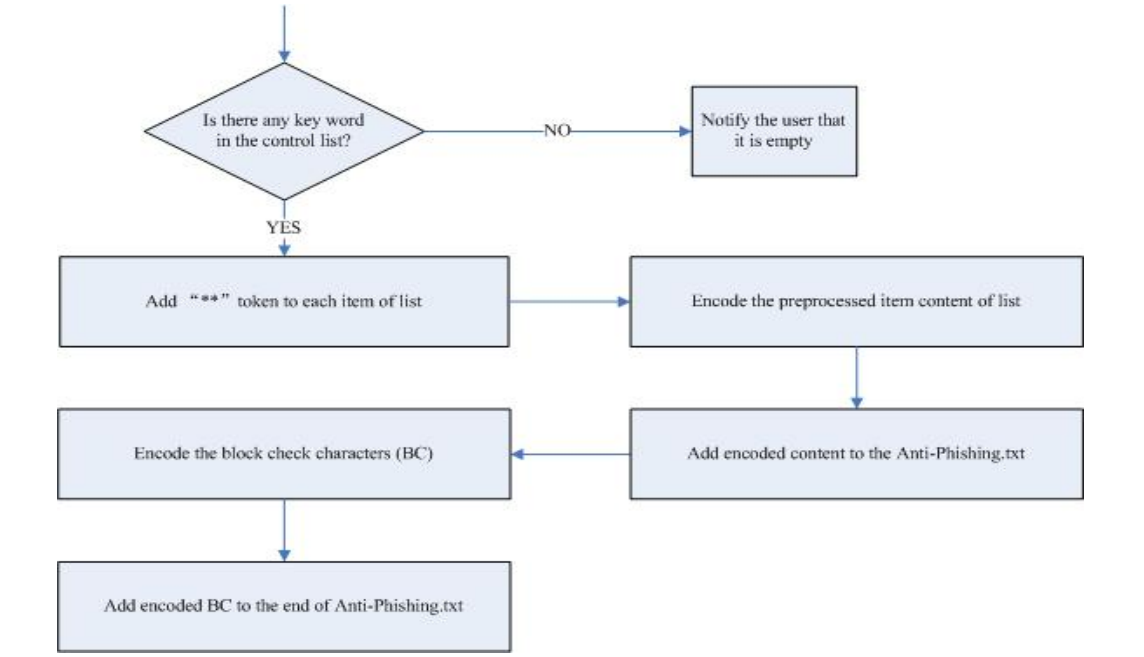

Figure 7. The save operation

Actually, there are two steps before saving to the file: adding a token, elaborated later and encryption. A double asterisk is used as a token and the token is used for identifying each website domain. In other words, there will be a double asterisks added to the end of each item in the control list. After this, the processed content in the list control will be encrypted with the same permutation algorithm as used for decryption. These encoded website domain keywords then will be used as one argument to encrypt the plain BCC. The encryption algorithm for the BCC is the same as the encryption of the website domain, except that the password argument is substituted by encoded website domains. After website domain and BCC have been successfully encrypted, they are saved to the Anti-Phishing.txt file.

The implementation of the other module IEPlug.dll is comparatively simple. This DLL just provides some alerts and advice to users (Figure 8). In two cases, the IEPlug DLL will alert. One is when there is a mistake or a problem with the Anti-Phishing.txt file, e.g. modification by outsiders. The other situation is when a user tries to visit any confidential sensitive website whose domain name has been preserved before. In order to alert properly when visiting a website, the keyword matching function is invoked.

The keyword matching function could be regarded as one of the most core part due to the need of precise detection. Moreover, because the matched keywords will be provided as recommended, careless treatment of keywords may become a critical vulnerability. For example, the authentic web address saved as a keyword could be www.ebank.com. Another fake web domain www.bank.com could also be saved to the keywords list. So when users visit www.ebank.com, it is possible to give deceptive website www.bank.com as the recommended web address, which will mislead or

confuse users. Thus, in order to avoid this potential vulnerability, an algorithm is used for finding the most matched characters. Therefore www.ebank.com will be given, instead of the fake one.

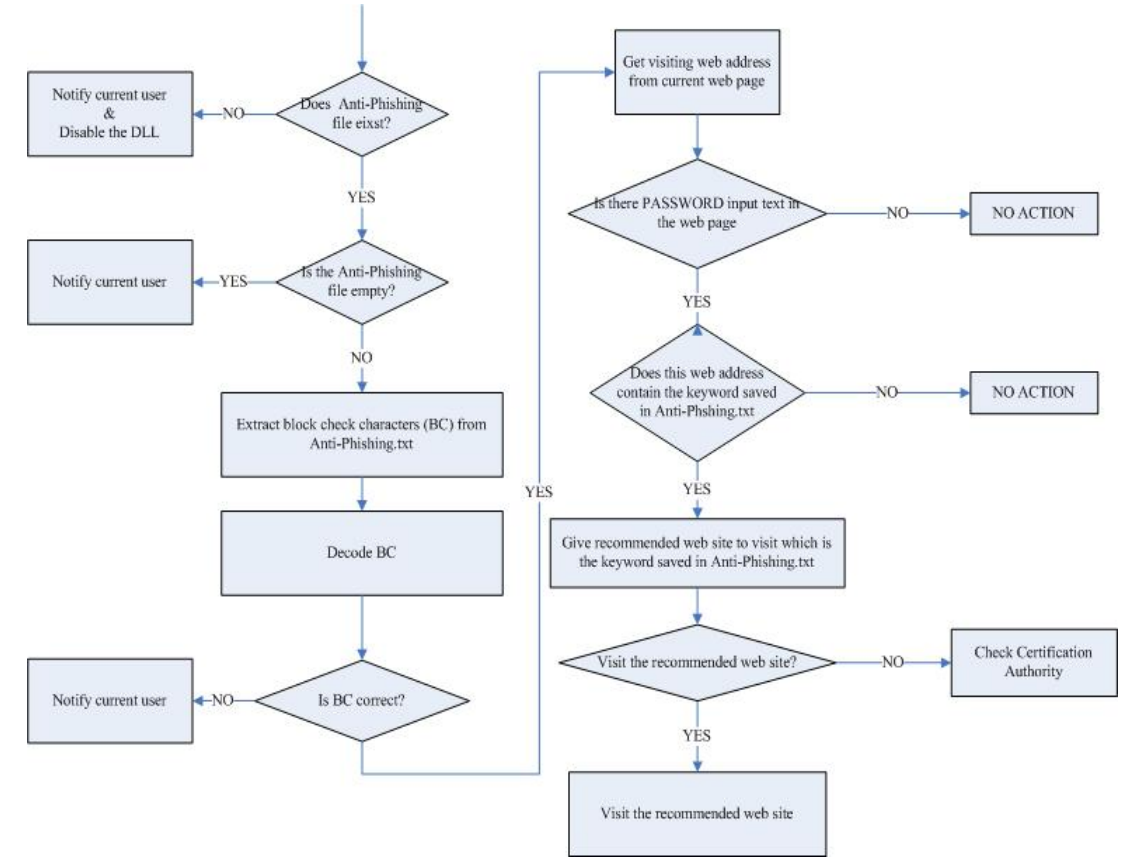

Figure 8. IEPlug.dll

4.2. Heuristic usability evaluation of anti-phishing toolbars

Several usability evaluations were done from users' aspect (e.g. [Dhamija et al., 2006], [Downs *et al.*, 2006].). Despite those studies collected users' responses after using toolbars, those results are not comparable. This is due to the fact that users' preferences would be affected by previously used toolbars. Meanwhile, users would learn how to identify phishing attempts during the evaluation. Therefore, it is very necessary to use a generic or universal set of rules to evaluate different toolbars [Li and Helenius, 2007].

For this usability evaluation, we applied Jakob Nielsen's heuristic evaluation method [Nielsen, 2006], which is the most common way to inspect software's usability. There are two reasons why I chose this method. First is that heuristics evaluation is flexible and efficient to find out potential usability issues. In addition, this method is helpful and necessary for forthcoming usability tests, because they can be based on the outcome of heuristics evaluation. Different from other inspections, heuristic evaluation specifically involves evaluators examining the interface and judging its compliance

with recognized usability principles (the "heuristics"). In order to get more reliable results, usability test specialists are required.

To guarantee the quality of evaluation results, the minimum number of evaluators is six [Nielsen, 2006]. In our evaluation, therefore, we invited four outside evaluators, who had sufficient working experience in either usability test or software design, and both of authors of this paper were involved in the evaluation as well. We two were involved, because both of us are familiar with the technical part and design of anti-phishing applications, and this may provide another helpful perspective to conduct the evaluation.

The heuristics used are listed in Appendix 1 [Pierotti, 2006]. Following the same items on the list, two evaluation results can be combined together to induce the final testing outcome. In the following parts of this section, we present the methodology how we designed the heuristic evaluation and what were the results. Afterwards, we summarize the evaluation and what we found during the inspection.

4.2.1. Detailed design and implementation of heuristic evaluation

Software usability mainly focuses on some specific characteristics of software, including easy to learn (learnability), efficient to use (efficiency), easy to remember, few errors, subjectively pleasing. All of these aspects should be dedicatedly evaluated. Moreover, testing anti-phishing toolbars is not the same as testing other applications. Evaluators have to pay much attention to the own features of anti-phishing toolbars during the usability inspection. According to this principle, we designed the heuristic evaluation items carefully.

1. *Evaluation environment.* We used one personal computer that was dedicated to testing usability of the anti-phishing applications. In this computer, Firefox 2.0 and Internet Explorer browsers 6.0 were installed. The operating system was Windows XP with all security updates installed. Sometimes, phishing websites contain malicious programs which may compromise the system or interfere with the evaluation. Thus, it was necessary to protect the computer. For this evaluation, F-Secure anti-virus client security was installed and the system was backed up to an image file. In case the system would have been compromised it was easy to recover. Furthermore, administrators of the department were aware of the testing, network traffic was monitored, the computer was physically isolated from internal network connections and a hardware firewall was present. There were two monitors installed on the computer, one was the normal screen for the evaluators' and the other was for the observers of the evaluation. The monitors showed the same screen. The phishing websites were collected from the phish tank web site [2006]. Because our focus was on usability of toolbars, instead of performance, we picked up the fake sites randomly. This enabled us to see the warnings of

each toolbar application in a real environment. For more detailed parameters of the evaluation environment, please refer to Appendixes 3 and 4.

2. *Design of the heuristic evaluation*. We collected heuristics following Jakob Nielsen's rules. In terms of these heuristics, a detailed questionnaire (see Appendix 1) was implemented.

In the following part, the detailed heuristics are listed and elaborated.

1) *Visibility of system status.* This heuristic inspects visual capability of the toolbars. Visual capability should be checked in three stages, which are visibility before checking the authenticity of the website, during checking the authenticity of the website, and visibility of checking the result. At each stage, anti-phishing toolbars should always keep users aware of what is going on and what is the result of identifying the web page. Moreover, response times and types should be reasonable and appropriate.
2) *Match between system and the real world.* Most of vulnerable users do not have enough knowledge of computers and the Internet. From this follows that each operation of the toolbar should be understandable and predictable for non-sophisticated users. This means that people who do not have any professional knowledge about computer and e-commerce should be able to protect themselves based on instructions or warnings from toolbars.
3) *User control and freedom.* As mentioned in the second heuristic rule, we should not expect online commerce customers having learned a lot about computers previously. Toolbar designers should not assume that every user can operate each functionality of the toolbar correctly, or as expected. Furthermore, it is necessary to provide additional functionality to undo and redo what users have done when they recognize that there is something wrong with their operations. In addition, it should be possible for users to leave the unwanted state before the whole transaction completes.
4) *Consistency and standards.* This requirement comes from the system requirements. For example, it is difficult to force a Microsoft user to get used to other systems, unless other systems' user interface resembles Microsoft Windows. So is the case with toolbars. The language of the toolbar should follow the platform and browser conventions as well. Moreover, advice should be consistent, when the same risk level of suspicious web pages or emails are detected.
5) *Help users recognize, diagnose, and recover from errors.* When users successfully pass the validation to conduct their problematic or incorrect operation, toolbars should also alert or give further advice to correct and

recover from errors. This correction should be offered before, during or after users' decisions.

6) *Error prevention.* Similar to the third heuristic rule, error prevention can also avoid crash or potential problems from users' operations. Toolbars are expected to provide necessary check or confirmation before any action is committed. Different from the third heuristic rule, error prevention focuses on the validation of users' each operation and input, instead of undo functionality.
7) *Recognition rather than recall.* It is required that any user, no matter who is sophisticated or not, can make a correct decision that prevents phishing without complicated operation sequences. Every warning or advice that a toolbar gives should be understandable enough. In this way users do not need to worry about being compromised due to forgetting correct instructions, even though users are misusing the toolbar.
8) *Flexibility and efficiency of use.* In order to prevent phishing, typically users have to perform some action, when a fraud attempt is detected. However, sometimes expert users are familiar with how to prevent specific phishing attempts when a warning comes up and they do not want to read repeated explanations. In this case, it is necessary that flexibility and efficiency of the toolbar can facilitate experienced users' operations, and enable skipping repeated instructions, or conduct the pre-saved default operation. Obviously, flexibility may also cause mis-operation or new vulnerabilities. Therefore, it is also required that users are able to return to the default settings.
9) *Aesthetic and minimalist design.* This heuristic mainly concentrates on the concision of toolbars' user interface. The task of anti-phishing toolbars is to assist users to identify and stop the fraud, not for example commercial promotions. It is meaningful and important to make sure there is only phishing prevention related information in the toolbars. A concise and well-designed toolbar will not confuse users what should be taken into account and what should be done next, when a warning is displayed.
10) *Help and documentation.* Users are not omnipotent, and they need to learn how to use different anti-phishing toolbars by themselves. In this case, user manuals, tutorials and instant help should be available with the toolbars.
11) *Skills*. Phishers usually take advantage of users' shortage of network or operating system knowledge [4]. Therefore, we expect that toolbars can support, extend, supplement, or enhance users' skills and background knowledge of phishing prevention. Herein, the enhancement should only be resorted to from the client side, because it is not valuable to evaluate the toolbars' capability against all kinds of phishing techniques.

12) *Pleasurable and Respectful Interaction with the User*. In this heuristic evaluation rule, we try to find out how convenient users' experience usage of the phishing prevention toolbar is. Both function and aesthetically pleasing value should be considered.
13) *Privacy*. Toolbars are used for protecting users' confidential information from being abused or stolen. However, some toolbars also need to know personal information about users, like contact methods. This kind of information should also be carefully protected when toolbars' providers obtain the information.

Besides these heuristic rules, severity of each usability problem should also be defined. Herein, we use the following rating rules provided by TAUCHI (Tampere Unit for Computer-Human Interaction) [2005].

"1. Major usability problem
– Prevents the users from using the product in a feasible manner and therefore needs to be repaired before the product is launched.
2. Severe usability problem
– Complicates the use significantly and should be repaired immediately.
3. Minor usability problem
– Complicates the use of the product and should be repaired.
4. Cosmetic usability problem
– Should be repaired for the use of the product to be as pleasant as possible.

T. Technical problem
–Problems marked with a 'T' are most likely due to technical problems with the product (for example, a feature that has not been implemented yet). Although they are not marked as usability problems, they will be such if left as they currently are.
C. Comment
–Comments (or questions) that are used for suggesting operations or point out successful implementations."

4.2.2. Conducting heuristic evaluation

In order to guarantee the quality of evaluation results, the whole procedure of heuristic evaluation was carefully designed. Each evaluator inspected the four toolbars selected. Each evaluator followed the heuristic evaluation criteria presented as designed and the following steps:

1) Individual preparation. The selected toolbars were inspected once in average about 2,5 hours by each evaluator. The aim was to get a general

feeling of each toolbar. Before conducting the evaluation evaluators received the heuristics checklists (Appendix 1).

2) Conducting the evaluation. The evaluation was conducted in the office of Marko Helenius. We observed each evaluation sequence and our main tasks were to record the findings of each evaluator. At first, we gave a basic introduction and demonstration of each toolbar in about fifteen minutes. Then, evaluators tested each toolbar based on the preparation and checklist. At this time, they had to explain their findings of usability problems, and their findings were recorded by the two authors of this paper.

   The time for each evaluation is given below:

   - First evaluator, Lauri Immonen, 2 hours (14.12.2006, 13.00-15.00)
   - Second evaluator, Markku Myllylahti, 2 hours (14.12.2006, 15.00-17:00)
   - Third evaluator, Wenfeng Liu, 3 hours (15.12.2006, 13.00-16:00)
   - Fourth evaluator, Lily Wen Lin-Marsalo, 3,5 hours (15.12.2006, 14:30-18:00)

   We also played the role of evaluators, but the evaluation method was different. When collecting data and writing, we added our ideas and findings to the final results.

3) Gathering evaluation results. The findings are combined into a single list and the severity of each distinct problem is rated and conclusions are discussed:

   a) What were the most severe problem types?
   b) What was the overall feeling about the usability of the toolbars?

4) Reviewing the problem list. The information from the discussion is gathered and summarized for the final evaluation outcome.

4.2.3. Results from the evaluation

After the evaluation, we gathered the findings from each evaluator. Based on the evaluation and guidance of the checklist (see Appendix), we gained a number of useful usability issues. Please note, that in heuristic usability evaluation the checklist is meant for guidance, but during the evaluation the evaluators do not need to strictly follow the checklist, but rather to bring forth each usability aspect they will find. We will next discuss the key findings from each toolbar.

Google Safe Browsing

Good usability design was observed especially when there was a phishing website detected (Figure 9). “The dimmed area of the browser feels like something, which cannot be accessed. And the balloon can draw user’s attention. ”, an evaluator said. There are no professional terms used, such as phishing or pharming. The optional

advice, “Get me out of here!” and “Ignore this warning, are understandable”. “Any user can easily get the points of them.”, said one evaluator. The “Read more” link introduces web forgery and phishing at technical level. This gives users freedom and is informative. The design principle of the Google Safe Browsing seems to follow the philosophy of a good security product design. The toolbar shows a clear warning only when a phishing is detected and otherwise the product remains silent.

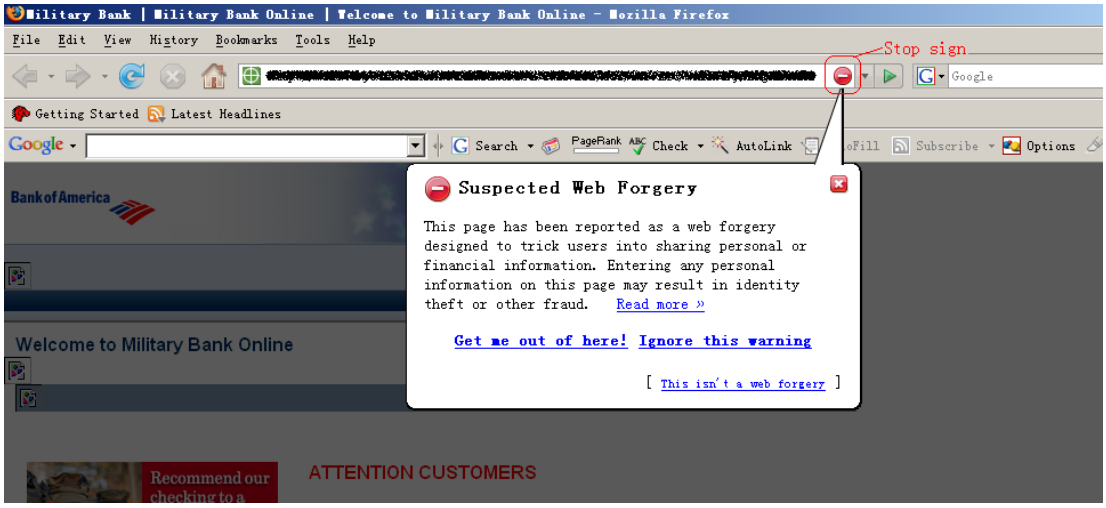

Figure 9. A phishing site detected by the Google Safe Browsing

However, also some usability problems were found during the evaluation. First of all, there are too many functionalities on the toolbar, but no access to the Safe Browsing functionality. It is not obvious that this toolbar can prevent phishing websites. Moreover, some experienced users may misunderstand that PageRank is part of the Safe Browsing functionality (Figure 10). In addition, loading the phishing site regardless of the warning (dimmed area in Figure 9) may cause some malware stealthily being installed from the visiting phishing website. Moreover, when a user clicks the option, “Ignore the warning”, there is no further warning about the danger any more. This is a problem, for example, when a user clicks the choice mistakenly.

Figure 10. PageRank functionality circled in red

Furthermore, some evaluators believed that the phishing indicator is not consistent enough. The indicator shows up only when a phishing website is detected. We also found that when Google’s web site cannot be loaded, (e.g. because of network traffic load) the option “Get me out of here!” will leave the user in the phishing web page, instead of being redirected to the Google site. As a consequence a user may erroneously

believe to be in a safe site. Finally, one evaluator was concerned that the warning icon (circled in Figure 10) may be confused with the lock icon .

Netcraft anti-phishing toolbar

Compared to the Google toolbar, Netcraft anti-phishing toolbar is designed only for phishing prevention. Therefore the toolbar's user interface is straightforward. The information about the visited website is displayed on the toolbar directly (Figure 2). The online tutorials are well designed.

However, the information present on the toolbar is not easily understandable. The most obvious usability problem is the implementation of the two drop-down menus "Netcraft" and "Services". Some useful options are placed unexpectedly and inconsistently. For example, "Report a phishing site" and "Report Incorrectly Blocked URL" should be services, but they are found from the "Netcraft" menu. Furthermore, the structure of the menus is too complex to be easily understandable.

There are also some other minor usability issues, which may cause ambiguity. For instance, the toolbar item, "Since", is right after the "RiskRating", which may confuse users; the criteria of "Rank" is imprecise; "Site Report" does not tell whether a site is fraudulent or not. Rather the site report shows technical server information. Not every user understands the information or needs it, especially normal end-users.

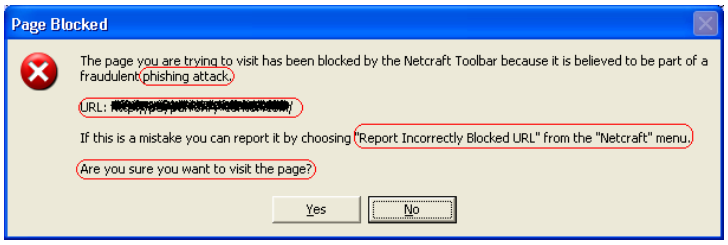

Figure 11. Netcraft anti-phishing toolbar's warning of a phishing site

When a phishing website is detected, inexperienced users may not understand information in the warning dialog (Figure 7). First of all, the popup dialog is similar to a website dialog or operating system dialog (e.g. illegal memory reference). Secondly, toolbar designers cannot assume every user knows these professional terms, such as phishing and URL. Furthermore, it is not necessary to show the URL, because not every user understands the web expression. In addition, "Report Incorrectly Blocked URL" highlighted in the warning dialog should be clickable to make users submit their reports, instead of forcing users to remember what they should do and where they can find the functionality. Finally, the expression "*Are you sure you want to visit the page?*" is not clear enough. When a user quickly reads this she or he may click a wrong button. Similarly, when there is a suspicious website detected, there is no advice, except changing the color of the "RiskRating" indicator.

Netcraft toolbar has a powerful website to support its services. Some of the important functionalities, such as reporting, have to resort to the website. Therefore the

related web pages should be evaluated as well. As the program relies on web pages, problems will appear when there is network load or the web pages are not available. This should be taken care in the product. For example, there could be internal help system, in addition to the web page help system.

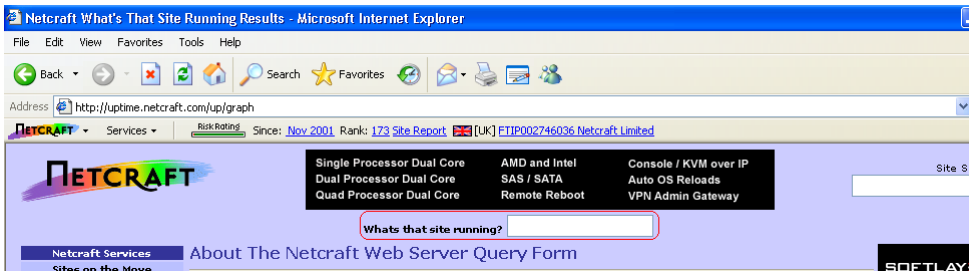

Figure 12. Not well-designed input field in Necraft website

One typical usability problem is that the input field of "What's that site running" is not distinguished enough (Figure 12). First of all, the input field is buried under other more distinguished text and advertisement. The input field can hardly draw users' attention. In addition, there is no "submit" or "go" button.

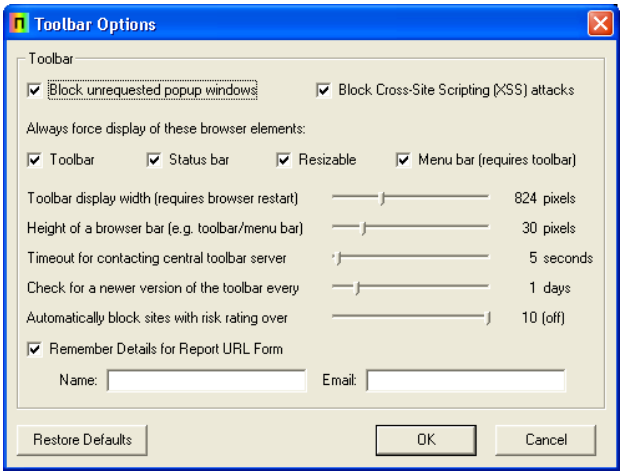

Figure 13. Options of the Netcraft toolbar

The toolbar configuration settings (Figure 13) are not designed well from the usability perspective. At first, the options are not grouped appropriately. For example, the appearance, and the functional settings (e.g. automatic blocking level) should be separated; the Remember Details for Report URL Form option should be grouped together with Name and Email fields. In addition, the controls used do not follow the common sense, such as checking for a newer program version.

SpoofGuard

SpoofGuard is the outcome of a research project at Stanford University. Some advantages in the user interface were found. The traffic light is consistent enough to help users identify the current visiting web page. Furthermore, the toolbar keeps user informed and the design is clear and concise enough.

However, from the usability point of view, there are some places to be improved. Firstly, the user interface is not always visible at the initial and locked location of toolbar, especially when a user visits a site whose domain name is too long (Figure 14). Herein, the indication showing identity of a web page is the color of the traffic light. However, this may be an obstacle for color-blind users. In addition, there are cultural differences as, for example, in India the color codes are different.

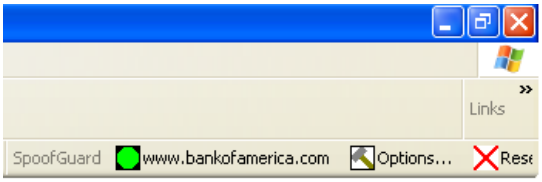

Figure 14. Too long domain name in SpoofGuard

Furthermore, the function of the reset button is unclear. Nothing seems to happen, when a user clicks the button. Users may also confuse “Reset” button with resetting the options to the default values, but actually clicking the button will remove the image hashes and password hashes. Moreover, while the red cross refers to deleting something, it is unobvious that clicking the button resets the configuration data.

Similarly, the suggestion for users is not explicit enough when a suspicious web page is detected. The suggestion is likely to confuse inexperienced users, when they want to know whether they should keep visiting the web page.

Finally, there are some comments about the warning when a spoof web page is detected (Figure 15). Similar to those comments about Netcraft toolbar, some terms are too professional to be understood by normal users. Furthermore, when SpoofGuard lists suspicious places of a web page, users should be able to learn more about them, for example, why images cannot be identical to those on another web site. Finally, users can hardly expect what will happen when they click the “Yes” or “No” button. The question should be clearer.

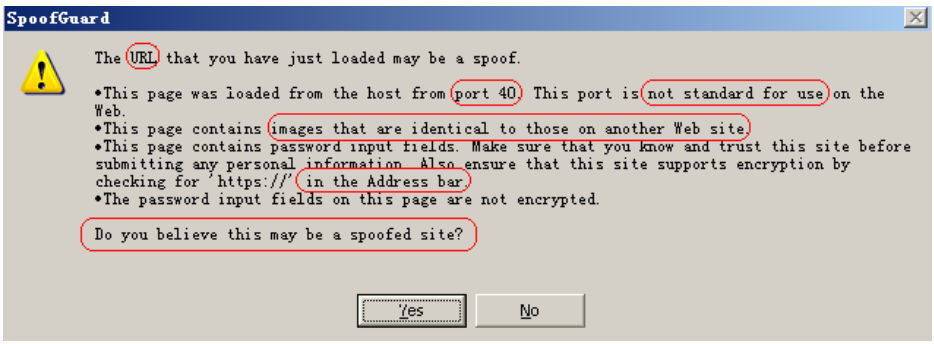

Figure 15. Warning of a spoof web site

Anti-phishing IEPlug

Similar to SpoofGuard, Anti-phishing IEPlug is a result of a University research project. Compared to the previous three toolbars, the user interface of this application on Internet Explorer toolbar is very simple, only one button. Furthermore, the idea to maintain only a whitelist was considered convenient, because it gives power to users and does not require constant updating.

Figure 16. Popup message in Anti-phishing IEPlug

However, some parts of the program can be improved. The popup messages are used too frequently. Sometimes, they are annoying. For example, when Anti-phishing IEPlug is installed successfully, the message is shown each time Internet Explorer is opened (Figure 16). The reason for showing the dialog was to show a user that the program is active and protecting the user. A better solution would be to show the program status, for example, as an icon on the toolbar. Another example is that when Anti-phishing IEPlug adds a domain name, there is no need to remind users with a popup message. Furthermore, the message text is too technical. End-users are likely to be confused.

The interface of the domain name configuration (whitelist) dialog is not satisfactory either (Figure 17). First of all, the title of the dialog does not make sense. The title should represent the purpose of the dialog. Likewise with other toolbars, some terms are too difficult to understand. The edit dialog for domain names is not long enough, and adding domain names is not consistent. The address is shown in the edit box, but only the domain name is added to the web site list. Furthermore, the dialog should be stretchable so that users can view as many domain names as possible at one glance. The controls on the dialog are not easy enough to use. The buttons are not well designed: there are two “Close” buttons; there is no feedback when users click “Refresh” or “Undo”.

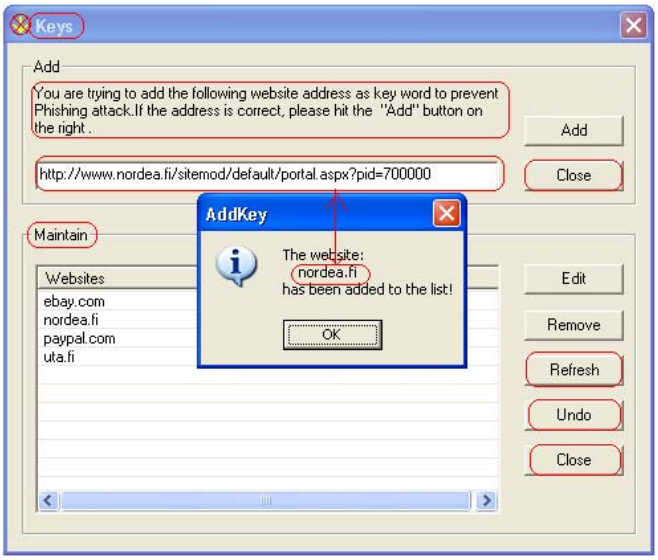

Figure 17. Popup message when adding a domain name

The warnings from Anti-phishing IEPlug are also problematic from usability aspect. Similarly, there are too many technical terms and there is no further information available to users, and it is imperative to develop a constant and well-designed help system to give user's more information. In addition, it will be convenient that the popular authentic website address could be pre-saved to the whitelist of Anti-phishing IEPlug. This feature may be implemented with client-server architecture, which collects the websites from every user.

4.2.4. Statistics

After the evaluation, we firstly reviewed the comments and then completed the final heuristic checklist. Based on the criteria in Section 4.2.1, we assigned the severity level to each usability problem. After that, we collected the usability problems of each anti-phishing application and constructed the following statistics tables. Obviously, limitation on these statistics does exist. First of all, these statistics are very rough evaluation results, which cannot reflect every usability problem precisely. Even though the heuristics checklist was designed beforehand, the entire evaluation is based on the evaluators' personal opinion. Therefore, the result of evaluation may be not comprehensive enough. For further research, it is necessary to conduct a usability testing to collect users' experiences and feedbacks directly. The statistics for their evaluation results are listed below:

Table 1. Statistics for the Google Safe Browsing

	Major	**Severe**	**Minor**	**Cosmetic**
Visibility	2	1	0	3

Matching the Real World	1	1	1	0
User Control & Freedom	1	3	1	0
Consistency & Standards	0	1	1	0
Help User Recognize	1	0	0	1
Error Prevention	0	1	0	0
Recognition	2	0	0	0
Flexibility	1	3	1	0
Aesthetic Design	0	0	0	0
Help & Documentation	1	1	2	1
Skills	1	2	1	0
Pleasurable Interaction	0	1	0	0
Privacy	0	0	0	0

Table 2. Statistics for the Netcraft anti-phishing toolbar

	Major	**Severe**	**Minor**	**Cosmetic**
Visibility	3	5	0	0
Matching the Real World	5	5	0	0
User Control & Freedom	1	2	1	0
Consistency & Standards	3	4	0	1
Help User Recognize	3	0	0	0
Error Prevention	2	1	0	0
Recognition	3	0	0	0
Flexibility	0	5	1	0
Aesthetic Design	1	2	0	0
Help & Documentation	0	1	1	0
Skills	2	1	1	0
Pleasurable Interaction	0	3	0	0
Privacy	0	1	0	0

Table 3. Statistics for the SpoofGuard

	Major	**Severe**	**Minor**	**Cosmetic**
Visibility	1	1	0	1
Matching the Real World	4	2	0	0
User Control & Freedom	1	2	0	0
Consistency & Standards	2	3	0	0
Help User Recognize	1	1	0	0
Error Prevention	2	0	0	0
Recognition	0	2	0	0

Flexibility	0	5	1	0
Aesthetic Design	1	0	0	0
Help & Documentation	5	5	0	0
Skills	2	1	2	0
Pleasurable Interaction	1	2	0	0
Privacy	2	0	0	0

Table 4. Statistics for the Anti-phishing IEPlug

	Major	**Severe**	**Minor**	**Cosmetic**
Visibility	1	5	3	0
Matching the Real World	1	4	0	0
User Control & Freedom	1	2	0	0
Consistency & Standards	3	2	0	0
Help User Recognize	1	0	0	0
Error Prevention	1	1	0	0
Recognition	2	3	0	0
Flexibility	0	4	2	0
Aesthetic Design	1	2	0	0
Help & Documentation	3	6	1	0
Skills	3	2	0	0
Pleasurable Interaction	1	3	0	0
Privacy	0	0	0	0

4.3. Misuse-oriented phishing prevention

In this chapter, I introduce the misuse case method to prevent phishing from the system design perspective [Li, et al,. 2007]. First, the misuse case methodology is explained. After that, there is a system design example given to illustrate how to find the requirements of fighting against phishing. Finally, a brief analysis of the misuse case method is given.

4.3.1. Misuse case methodology

The first time that a misuse case was used for security requirements elicitation was achieved by the co-operation between the Norwegian University of Science and Technology and the University of Bergen [Sindre and Opdahl, 2001]. According to the article, the misuse case was based on the analysis of use cases at the requirements gathering stage. In brief, a designer is asked to impersonate a misuser to abuse every use case (misuse case). When a threat is defined by a misuse case, the corresponding countermeasures can be identified and employed to prevent from the threat in the future. Of course, it is possible that there remain potential vulnerabilities in the countermeasure.

Therefore, a next round misuse case design against the countermeasure(s) is required, until there is no possible vulnerability found.

In summary all the steps of the methodology of the misuse cases [Sindre and Opdahl, 2001] can be induced in the following six (6) steps:

a. Design the use cases of the system;
b. personate a misuser, who intends to compromise the system;
c. design the misuse for a specific use case;
d. find a countermeasure for a misuse case;
e. judge whether the countermeasure is vulnerable; if yes, go to step c, otherwise go to the next step;
f. find whether there is possible vulnerability or misuse; if yes, go to step c, otherwise security requirements elicitation ends.

In the following section we provide an example scenario to demonstrate and clarify the use of the above.

4.3.2. Misuse cases to design an online music purchase website

In order to demonstrate the value of the misuse case method, let us consider an e-shop MusicBox for music purchasing with an available website. For the purpose of extracting reasonable misuse cases for the design rationale, the description of the normally available e-services on the website is given below.

The MusicBox is an online music products provider. Its e-services are designed with functionality and availability in mind to include, among other online services, free listening of a few favourites, online latest music clips listening, and vending music products and so on. This website comprises several functional and usable components: (i) a web browser, (ii) client application, (iii) front-end server, (iv) content database and (v) credit card server (see Figure 18).

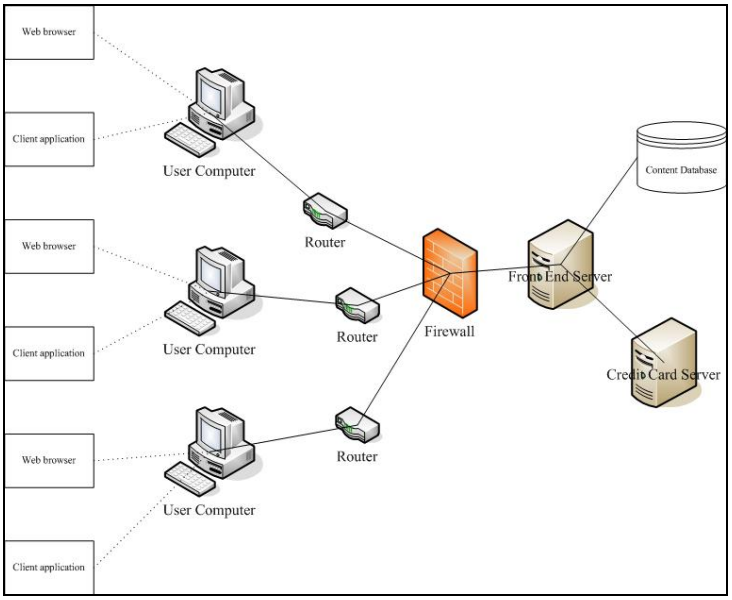

Figure 18. Architecture overview of a music purchase website

A web browser, such as Internet Explorer or Firefox, runs on the webpage users' personal computers and users can also access the website of the MusicBox via the browser. The functions offered on the website contain listening to the music clip promotions, opening a customer account and logging into an account when purchasing music products. Only after a user creates an account in the e-shop of the MusicBox, he/she is able to purchase the music products. In order to buy music products, valid credit card details are required. When valid credit card information is given, credit card server will handle the credit card transaction. Afterwards, the content database makes the music products available, which can also be downloaded to the user's personal computer. The client application, a decode player, can decode and play the music file.

Herein, the use cases are utilised to design the misuse case(s). The use cases, thus, are not listed within any industrial template but within our framework of thinking. In the following lines UC is an abbreviation of Use Case, and the number is a simple sequential indicator for the use cases listing. In summary, the use cases for the MusicBox described in the previous paragraph, will be:

UC-1. Any user can listen to the music clip promotions.

UC-2. The user who wants to create an account in MusicBox should offer a valid email address, password and a user name.

UC-3. The email for confirming the account is sent to the user's registered email address.

UC-4. After confirmation, the user's account is activated. Then, the user is able to purchase the music products.

UC-5. A user can purchase a music product, after logging in.

UC-6. A user should provide valid credit card number for purchasing.

UC-7. Credit card server should delete users' credit card information after successful purchase.

UC-8. A confirmation email, 1) to notify user about successful purchase, should be sent to user's email address left, when he or she has created the account. A valid link to download the music product is added into the account profile; 2) to notify the user who does not use the account for a certain long time.

UC-9. User can download the music from the link in his or her account profile.

UC-10. After downloading completes, a user can display the music via client application which is downloadable from the MusicBox free of charge.

UC-11. The downloaded music from the MusicBox cannot be displayed by other music players.

UC-12. The purchased music product can be downloaded as many times as needed by the account owner.

After listing the Use Cases one might ask who could be a future system misuser. Possible system misusers in the future could be code crackers and MusicBox users who want to re-sell or share the music with others. After the use cases are elicited, misuse cases can then be designed. Herein, the Sindre's template [2001] is applied. One sample of misuse case is given in Table 5.

Table 5: Misuse cases sample A

Name:	Sending the email to confirm the use of one account
Case ID:	MC-1
Summary:	A cracker can obtain the account name and password by sniffing plain messages.
Author:	******
Date:	2007-3-10
Basic path:	A cracker can obtain the account name and password from sniffing the communication of the server.
Alternative paths:	1. A cracker can harvest account name and password from a forged website. 2. Man-in-the-middle attack can be used for obtaining this information 3. Compromising the server
Capture points:	1. Password is changed 2. Forged site has been detected by the toolbar
Extension points:	Keep sniffing communication
Trigger:	This can happen at any time.
Assumptions:	1. The server and forged website are ready. 2. The MusicBox server security capacity is not sufficient. 3. A number of customers created their own accounts
Preconditions:	The communication is not encrypted.
Worst case threat:	1. Launch another social phishing attack 2. The music can be downloaded freely.
Prevention guarantee:	1. SSL or TSL is used to protect secure communication 2. Login history offered to check the account history 3. Frequently change password
Detection guarantee:	Check login history
Related business rules:	1. The purchased music product is listed in the account profile. 2. The purchased music product can be downloaded for several times. 3. Users can access the account with the account name and

	password.
Potential misuser Profile:	Sufficient computer skills
Stakeholders and Threats:	MusicBox: users complain that the system is not secure to be used. Customers: they lose their money.
Scope:	Entire business
Abstraction level:	Misuser goal

Still, the system is not completely phishing resistant. For example, when a user has not used his or her account, the system should notify the user to confirm its account. In this case, it is also possible to be spoofed. Its misuse case can be given as follows.

Table 6: Misuse case sample B

Name:	Steal password and account name by sniffing plain messages
Case ID:	MC-2
Summary:	A phisher impersonates the MusicBox staff and sends a email to make a user confirm the use of his or her account.
Author:	******
Date:	2006-11-20
Basic path:	1. A phisher can send the email by any email client, such as Outlook.
Alternative paths:	A phisher may impersonate a friend of the target, and pretend to remind the victim by sending an email with "official" link to confirm the use of the account A phisher may ask for the account confirmation so as to update the security status of it.
Capture points:	1. Using a specific customized certificate to identify the authenticity of the email source. 2. Pausing the processing of suspicious business transactions. Using confirmation code for each transaction.
Extension points:	1. Guess the customized certificate
Trigger:	This can happen at any time.
Assumptions:	1. The server and forged website are ready, and successfully avoid to be detected. 2. The targets' emails are collected. 3. The forged emails are sent to valid the targets' email boxes.
Preconditions:	There should be a regulation to remind users to keep their

	account activated.
Worst case threat:	1. Users do not believe even the authentic emails from MusicBox. More users quit to use the MusicBox.
Prevention guarantee:	1. The customized certificate to identify the authenticity of the email source has been decided when a user register. 2. The requested confirmation code for each transaction varies all the time.
Detection guarantee:	1. Checking the transaction history
Related business rules:	1. The purchased music product is to be confirmed by the user itself. 2. Every user account should be activated.
Potential misuser Profile:	Awareness of building the website, sending fake emails with the URL of the forged sites.
Stakeholders and Threats:	MusicBox: users complain that the system is not secure to be used, and there are less and less users using MusicBox. Customers: some of their accounts become the tool of social phishing.
Scope:	Entire business
Abstraction level:	Misuser goal

The misuse case sample B is also not phishing resistant. Phishers may launch the second round of man-in-the-middle attack to ask the confirmation code, after they get the victims' account name and password. In this case, requirement engineers and system designers need to create another misuse case, and then recursively repeat the previous steps until there is no more possible breach to be found.

5. Conclusions and future work

5.1. Known vulnerabilities & countermeasures of Anti-phishing IEPlug

Anti-phishing IEPlug is designed to prevent phishing attack with a deceptive web address. However, when the fake web address does not imitate the domain of protection-required address, the DLL cannot alarm properly, even though the content of this fake web address are the same as the authentic one. One of the countermeasures of this could be found from banks and clients themselves who could detect and report the suspicious websites.

Secondly, Anti-phishing IEPlug cannot detect the password input text elements from web pages written by JavaScript. Considering the performance of this application,

checking password input text elements is placed before finding keywords from visiting page address. This means that when the fake web page only using JavaScript (rather than HTML), it is successful to evade checking keywords function. In order to prevent it, the web designers could avoid to use plain JavaScript design where SSL is used, and clients should reject to use plain JavaScript page to input password or other confidential information.

In third, The Anti-Phishing file is not safe enough. The content of the entire file including BCC could be replaced by malware (e.g. fake keywords and its new fake BCC following the same permutation algorithms). The application cannot work as expected until the administrator finds them, and normally, it is hard to detect the modification even manually. The countermeasure could resort to backup the file and its maintenance by administrator users.

In addition, a careless or even malicious user could add a fake domain name as a keyword, which would thus be recommended by the IEPlug. The countermeasure is to check certificate of the visited web page. In this application, checking certificate is compulsory protection when the visited link contains a keyword stored in Anti-Phishing.txt. If a recommended domain name is not valid, the administrator user should remove the domain name from the keyword list.

Also possibly, IEPlug.dll could be unloaded by anyone. When there is no alarm the unwitting users including administrators may falsely believe that there is no threat of a phishing attack. Therefore there is the notification of successful loading message when starting IE. Of course, malware targeting the application could also invoke the message.

Moreover, it is possible to modify the IEPlug DLL deliberately or unintentionally. This could be prevented by means of administrators' configuration to the access right of the DLL. Finally, the encryption is not secure enough, and could be compromised, which could be improved and strengthened in the further research.

5.2. Suggestions on toolbar usability improvement

According to the comments and statistics constructed from the evaluation, we made a number of findings for anti-phishing client side application usability design. Generally, we found that there are three basic components that should be well designed: the main user interface of the toolbar, warnings, and help system. We will next discuss our key findings for these components.

1) *Main user interface of the toolbar.* According to our perceptions, the main user interface of the toolbar is very important. First of all, the status of the toolbar should be shown appropriately. This means that whenever browsing a web page, the user should easily observe what the toolbar is doing and whether the current visiting web page is authentic or not. Secondly, the anti-phishing client side application interface should be simple enough so that it is easy to understand and it does not take too much space from the

browser's interface. Of course, frequently used and important functionalities, such as configuration settings and viewing the website identity analysis result, reporting a suspicious or misjudged web page, should be convenient enough to be found. In this regard, some parts of the SpoofGuard of the interface design could be a very good example, such as the traffic light indicator and Options button, which can keep informing the user of the available and the frequently used functionalities.

2) *Warnings*. Considering the lack of reliable strategy to detect fraud, the warnings of the application need to be carefully designed. It is important that a user is able to react correctly when a fraud or suspicious web page is found. According to the evaluation by Cranor *et al.* [2006] and our observation, a false and undetermined detection is not minor. It would be problematic if a user relies only on these toolbars with fixed detection algorithms. Therefore, there should be at least four levels of security indication: the warning for detected web forgery, the warning for a determined suspicious page, the warning for the page unable to be determined, and the indication for an innocent or authentic page.

   The warning from the Google Safe Browsing is a good example for showing web forgery. Google's warning can stop users' faulty visits properly. The warning for a suspicious page can be the same as the one for the forgery. The differences between them could be on the given advice and their indications. For example, there may be only one piece of advice (stop visiting) available for the forgery, and the indicator for the warning could be a stop sign. However, there could be two further pieces of advice (stop visiting, or check authenticity manually), when a suspicious page is found. Likewise, the indicator should not be so strong as that for the forgery (e.g. an exclamatory mark). The undetermined page requires to be notified to users as well. When this kind of page is found, instant help documentation or manual is needed to help the user identify the page manually. Additionally, in order to be consistent, innocent pages should be indicated respectively. For instance, the indicator could be shown at the same location of other levels of phishing warning indicators. Finally, a double warning should be used in case an erroneous choice is made. If a user accidentally selects a choice that leads to visiting a phishing website, the second warning should be available to correct the mistake.

3) *Help system*. Compared to other software, the client side anti-phishing application should be able to help users at any occasion. These occasions may include when users may select a dangerous choice, when they are confused by some terms, and when they want to learn how to identify the

correct service manually. Regarding the efficiency and convenience of help, the ways of showing help for different occasions may not be the same. For example, when a user tries to find further advice, an instant help system is needed. Another example is when a user should find out the consequences of different choices when a warning is present. However, the text which may help the user to understand some terms or consequences of choices cannot be put together with the warning. It must be remembered that too much information will confuse users. For the other two occasions, online help documentation would be better, because there can be much more information. The help system of the Netcraft toolbar could be a valuable example.

Finally we have one general observation. Anti-phishing client side applications should not rely merely on the Internet, because sometimes the online traffic is not good enough. For example, when a user chooses an option to leave the phishing website, the toolbar could direct the user to a safe page. However, if the connection fails at that time (we met this occasion during the evaluation), the user will stay on the spoofing website. This may place the user at risk. Therefore, anti-phishing applications had better redirect to the locally saved page or eliminate the online redirection. Furthermore, it should be taken note into of that online help systems and reporting systems that rely on the Internet connection may not work all the time.

5.3. Future work on usability evaluation of antiphishing preventions

In the conducted heuristic usability evaluation, we presented a design of heuristic evaluation of four typical anti-phishing client side applications, and discussed our findings of the evaluation. As far as we know, this evaluation is novel usability research in the phishing prevention domain. We found some important usability issues which could be helpful for further improvement of anti-phishing toolbars. Furthermore, the heuristics checklist could be reusable for future testing as well.

However, there are also some limitations in the evaluation. Due to the natural drawbacks of heuristic evaluation, we cannot get precise and direct users' feelings on using these toolbars. Moreover, because of the limited number of evaluators, not every usability problem was found. Conducting the future usability tests, those drawbacks could be overcome with larger resources. In addition, we failed to indicate whether in our application the Certificate Authority is usable or not when showing actively. Finally, the number of evaluated application types is limited. When we were conducting the evaluation, we realized that there could be more than four types of anti-phishing client side applications.

Despite the limitations of this evaluation, there are some contributions to the anti-phishing research domain. We succeeded to construct a heuristic checklist, and found out some key usability issues based on the evaluation, including what an appropriate

warning should look like, how to warn a user in an understandable way and polite manner, and what are the essential components of client-side anti-phishing applications. All of these may facilitate future usability design and be a basic guidance in the anti-phishing usability domain.

5.4. Conclusions on misuse case method against phishing and its future work

From the example presented in Table 5, we can easily find that the misuse case method is a system-oriented, and rather design-centred, method. This means that different systems can have different misuse cases. Due to this feature, the misuse cases have to be deductively designed for every information system. The systems designed with misuse cases could, thus, be phishing-resistant. Since misuse cases are part of the design documentation we propose that existing systems could be re-designed and re-engineered for improving their security without great financial cost. Misuse cases identification in legacy systems, in the way we demonstrated it earlier, could be a cost effective quality solution to provide security. Moreover, the misuse case method could be helpful for validating security requirements when designing new information systems without considering the budget expenses for the future system updates.

On the other hand, the quality and the quantity of misuse cases prompt to adequate knowledge and experience on system design and testing phases by the designers and testers. In order to find out the possible phishing attempts, or at least as many as possible, the designers and testers should be familiar with most of or all system vulnerability issues that exist. Particular attention should be paid in the essence and philosophy of each type of attack. If system designers and testers know little about phishing, software quality assurance techniques alone can not offer much. Regarding system security, for instance, a worthy misuse case to simulate a phisher cannot be provided by an inexperienced designer, even though the misuse case’s recursive procedure may help.

Software-based systems for e-transactions are currently facing many unresolved software quality aspects regarding their operability, security and efficiency. Security requirements in particular are not well-stated in the requirements gathering and analysis stages. This might be the result of the inexperienced designers decisions, and not adequate knowledge on test and use cases. On the other hand, impacts on software quality might also be factors such as lack of reliable, mature and understandably suitable (for the particular system's domain) misuse cases and test cases. Consequently, the system's fault-tolerance, recoverability and functionality are compromised when the information system becomes a target for phishing.

In contributing to an improved understanding of the phishing and anti-phishing methods that will be useful to researchers and especially practitioners when they deal with system design quality features, we proceeded as follows: We provided a critical review and comparative analysis of the features of the existing phishing techniques and

their prevention methods. In so doing, we provided a broad classification with the distinct features of the phishing and anti-phishing activities under examination. Phishing itself, a semantically serious attack on organisational, social, personal and interpersonal information systems, is very difficult to prevent. The purpose of each phishing technique, (e.g. in email context) may vary randomly, while the existing prevention methods are fixed. This natural defect has already restricted the overall performance and detection effectiveness of the prevention methods significantly, regardless the continuous evolution of phishing techniques.

Considering this special case, we may conclude that the more is known about limitations of existing applications against phishing, the more applicable a method on misuse case can be. Therefore the design quality features can offer reliability to adequately prevent system users from phishing. The drawbacks, however, of the misuse cases in design are also obvious. In particular, this method requires designers with ample experience in system design and system security. Our future research efforts will concentrate on (i) defining further software quality criteria for a design architecture that contributes to anti-phishing prevention; and (ii) making misuse cases more valuable (and perhaps more reusable) and more formal in order to be used as a quality assurance technique in the validation and verification of a system examining its design.

At system's design phase, testing of software design architectures could provide early feedback on the robustness and vulnerability of the domain-specific system architecture including information on both client and server identities, pitfalls and possible threats. In software development early feedback on software design quality features is a software process improvement issue, in practice. Our ongoing research deals with this and other software process improvement issues searching to provide an answer in the future research study. Other research efforts concentrate on advancing theoretical knowledge on the way system design and software architectures provide safety to e-people interacting in virtual communities worldwide.

References

[APWG, 2006] Anti-Phishing Working Group (APWG), 2006, Phishing attack trends report - March 2006. Also available as http://www.antiphishing.org/reports/apwg_report_mar_06.pdf. (checked on November 9th, 2006)

[Berki and Georgiadou, 1996] Eleni Berki and Elli Georgiadou, Towards resolving data flow diagramming deficiencies by using finite state machines. In: *Proc. of the 5th International Software Quality Conference*.

[Berki *et al.*, 2003] Eleni Berki, Hannakaisa Isomäki, and Mikko Jäkälä, Holistic Communication Modelling: Enhancing Human-Centred Design through Empowerment. *HCI International* **3**, 2003, 22-27.

[Berki *et al.*, 2004] Eleni Berki, Elli Georgiadou, Mike Holcombe, Requirements engineering and process modelling in software quality management – towards a generic process metamodel. *The Software Quality Journal* **12**, 2004, 265-283.

[Berki, 2006] Eleni Berki. Examining the Quality of Evaluation Frameworks and Metamodeling Paradigms of Information Systems Development Methodologies. *Measuring Information Systems Delivery Quality.* Hershey, 2006.

[Chau, 2005] David Chau,. Prototyping a Lightweight Trust Architecture To Fight Phishing. *MIT undergraduate projects*. Also available as http://theory.lcs.mit.edu/~cis/theses/chau-uap.pdf (checked in December 2006)

[Cranor *et al.*, 2006] Lorrie Cranor, Serge Egelman, Jason Hong, and Yue Zhang, Phinding Phish: An Evaluation of Anti-Phishing Toolbars. Carnegie Mellon University, Cylab, Technique Report CMU-CyLab-06-018, November 2006. Also available as http://www.cylab.cmu.edu/files/cmucylab06018.pdf.

[Dhamija *et al.*, 2006] Rachna Dhamija, J. D. Tygar and Marti Hearst, Why phishing works. In: *Proc. of the Conference on Human Factors in Computing Systems (CHI2006)*.

[Downs *et al.*, 2006] Julie S. Downs, Mandy B. Holbrook, Lorrie Faith Cranor, Decision strategies and Susceptibility to Phishing. In: *Proc. of the 2006 Symposium On Usable Privacy and Security*, 79-90.

[Emigh， 2005] Aaron Emigh, Online identity theft: phishing technology, chokepoints and countermeasures, Radix Labs, 2005. Also available as http://www.antiphishing.org/Phishing-dhs-report.pdf.

[Georgiadou *et al.*] Elli Georgiadou, Kerstin V. Siakas and Eleni Berki, Quality improvement through the identification of controllable and uncontrollable factors in software development. In: *Proc. of EuroSPI 2003: European Software Process Improvement*, IX 31-45.

[Google, 2007] Google Toolbar. http://www.google.com/support/firefox/bin/static.py?page=features.html&v=2.0f (checked in January 2007)

[Helenius, 2006] Marko Helenius, Fighting against phishing for online banking recommendations and solutions. In: *papers and presentations of the15th Annual EICAR Conference "Security in the Mobile and Networked World"*, 252-267.

[IEPlug, 2006] IEPlug-in. http://www.cs.uta.fi/~ll79452/ap.html (checked in December 2006)

[Igor, 2005] Muttik Igor, Manipulating the Internet. In: *Proc. of the 15th Virus Bulletin International Conference*, 1825.

[ISO, 1991] International Organisation for Standardisation (1991). Information Technology-Software product evaluation: Quality Characteristics and Guidelines for their use. ISO/IEC IS 9126. Geneva: ISO.

[Jagatic *et al.*, 2005] Tom N. Jagatic, Nathaniel A. Johnson, and Markus Jakobsson, Social phishing. *Communications of the ACM* **50**(10), 2007, 94-100.

[Jakobsson, 2006] Markus Jakobsson, Modeling and preventing phishing attacks, *Phishing Panel of Financial. Cryptography.* Also available as www.informatics.indiana.edu/markus/papers/phishing_jakobsson.pdf. (checked in November 2006)

[Javelin, 2006] Javelin Strategy and Research. http://www.javelinstrategy.com/ (checked in November 2006)

[Jäkälä and Berki, 2004] Mikko Jäkälä, and Eleni Berki, Exploring the principles of individual and group identity in virtual communities. In: *Proc. of the 1st IADIS Conference on Web-based Communities*, 19-26.

[Kirants, 2005] Kirants, Customize an IE context menu to add CodeGuru favorites , Codeguru, 2005. Also available as www.codeguru.com/cpp/misc/misc/internetexplorer/article.php/c11007__1.

[Koskinen, 2005] Minna Koskinen , Eleni Berki, Katja Liimatainen, Mikko Jakala, The human context of information systems. In: *Proc. of the 38th Hawaii International Conference on Systems Sciences, (HICSS 2005),* 219a-234.

[Kumaraguru, 2006] Ponnurangam Kumaraguru, Yong Woo Rhee, Alessandro Acquisti, Lorrie Cranor, Jason Hong, Elizabeth Nunge, Protecting People from Phishing: The Design and Evaluation of an Embedded Training Email System. Carnegie Mellon University, Cylab, Technique Report CMU-CyLab-06-018, November 2006. Also available as http://www.cylab.cmu.edu/files/cmucylab06017.pdf.

[Li and Helenius, 2006] Linfeng Li and Helenius Marko, Anti-phishing IEPlug. http://www.cs.uta.fi/~ll79452/ap.html (checked on September 1st, 2006)

[Li and Helenius, 2007] Linfeng Li and Marko Helenius, Usability evaluation of anti-phishing toolbars, *Journal in Computer Virology* **3**(2), 2007, 163-184

[Li, et al,. 2007] Linfeng Li, Marko Helenius and Eleni Berki, Phishing-resistant information systems: security handling with misuse cases design. In: *Proc. of British Computer Society Quality Specialist Group's Annual International Software Quality Management conference.*

[Netcraft, 2006] Netcraft anti-phishing toolbar. http://toolbar.netcraft.com/ (checked in November 2006)

[Nielsen, 2006] Jakob Nielsen, Heuristic Evaluation online writings. http://www.useit.com/papers/heuristic/. (checked on October 18th, 2006)

[Phish tank, 2006] Phish tank, Phish tank website, www.phishtank.com. (checked on November 5th, 2006)

[Pierotti, 2006] Deniese Pierotti, Usability techniques: heuristic evaluation - a system checklist, *Xerox Corporation*. Also available as www.stcsig.org/usability/topics/articles/he-checklist.html. (checked on October 18th, 2006)

[Sindre and Opdahl, 2001] Guttorm Sindre, Andreas L. Opdahl, Templates for misuse case description. In: *Proc. 7th Int'l Workshop Requirements Eng.: Foundation for Software Quality (REFSQ 2001).* Also available as http://swt.cs.tu-berlin.de/lehre/saswt/ws0506/unterlagen/TemplatesforMisuseCaseDescription.pdf.

[SpoofGuard, 2006] SpoofGuard. http://crypto.stanford.edu/SpoofGuard/ (checked in November 2006)

[Stop-phishing, 2006] Phishing research groups, http://www.indiana.edu/~phishing/?people=external, (checked on October 20th, 2006)

[TAUCHI, 2005] TAUCHI group, Usability evaluation method course slides in University of Tampere. (checked on November 20th, 2005)

[Wikipedia, 2006] Wikipedia. www.wikipedia.com (checked in November 2006)

Appendix 1

Toolbars Heuristic Evaluation – Checklist

1. Visibility of Toolbar Status

The toolbar should always keep users informed about what is going on, through appropriate feedback within reasonable time.

#	Review Checklist	Yes	No	N/A	Comments
1.1	Does every display begin with a title or header that identify itself?	☐	☐	☐	
1.2	Is the toolbar status shown before visiting any new web page?	☐	☐	☐	
1.3	Is it easy to find which operations are available before visiting any web page?	☐	☐	☐	
1.4	Is the toolbar status shown during verifying legitimacy of web pages?	☐	☐	☐	
1.5	Is there any suggestion on what user should do during waiting for verification result?	☐	☐	☐	
1.6	If there are observable delays (greater than fifteen seconds) in the toolbar's verification response time, is the user kept informed of the toolbar's progress?	☐	☐	☐	
1.7	Is the web page legitimacy analysis result shown properly after showing the content of web page?	☐	☐	☐	
1.8	If error occurs because of users' mis-operation, is user able to see the field in error?	☐	☐	☐	
1.9	Is there some distinguished form of toolbar feedback for warning, when the fraud is detected?	☐	☐	☐	

1.10	Is it clear to know which items in the dialog or toolbar are selectable?	☐ ☐ ☐	
1.11	Are every two items separated properly on the toolbar?	☐ ☐ ☐	
1.12	Are the buttons on the toolbar separated obviously?	☐ ☐ ☐	
1.13	Is there any clear explanation from toolbar before significant operation (e.g. decide to keep visiting the warned suspicious web page)?	☐ ☐ ☐	
1.14	Are the fraud detection response time less than 1 second, regardless connection delay?	☐ ☐ ☐	
1.15	Is the used terminology consistent with anti-phishing domain?	☐ ☐ ☐	
1.16	Can image on button express the function of it correctly?	☐ ☐ ☐	
1.17	Can user distinguish different GUI controls from each other (e.g. Drop-down list does not look like a button) ?	☐ ☐ ☐	
1.18	Can users know what operations are available, when the fraud is found?	☐ ☐ ☐	
1.19	Can users know whether the visiting web page is deceptive one or not, after toolbar's verification?	☐ ☐ ☐	

2. Match Between Toolbar and the Real World

The toolbar should speak the user's language, with words, phrases and concepts familiar to the user, rather than software oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

#	Review Checklist	Yes No N/A	Comments

2.1	Are icons meaningful and concrete?	☐ ☐ ☐	
2.2	Are the menu items and buttons on the toolbar ordered in the logic way, giving users what will be done after selection?	☐ ☐ ☐	
2.3	If there is a visual cue (e.g. images on buttons), does it follow real-world conventions?	☐ ☐ ☐	
2.4	Are there any obvious differences between selected and unselected?	☐ ☐ ☐	
2.5	On user input field, are tasks described in terminology familiar to users?	☐ ☐ ☐	
2.6	Are the questions understandable, which are given by popups of toolbar?	☐ ☐ ☐	
2.7	Do menu choices or words on buttons have readily understood meanings?	☐ ☐ ☐	
2.8	Are menu items parallel grammatically?	☐ ☐ ☐	
2.9	Do commands follow the language in daily life, instead of computer science domain?	☐ ☐ ☐	
2.10	If in need of input, is it available to give uncommon letters?	☐ ☐ ☐	
2.11	Is key function of toolbar labeled clearly and distinctively?	☐ ☐ ☐	
2.12	Are fields on the window separated appropriately?	☐ ☐ ☐	
2.13	Are fields on the window grouped appropriately?	☐ ☐ ☐	
2.14	Are buttons or menu items grouped appropriately?	☐ ☐ ☐	

3. User Control and Freedom

Users should be free to select and sequence tasks (when appropriate), rather than having the toolbar do this for them. Users often choose toolbar functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Users should make their own decisions (with clear information) regarding the costs of exiting current work. The toolbar should support undo and redo, when user choose advice given by toolbars.

#	Review Checklist	Yes No N/A	Comments
3.1	Can users check legitimacy of web page at any time when they want?	☐ ☐ ☐	
3.2	Can users conduct several same functional operations together, instead of repeating them one by one?	☐ ☐ ☐	
3.3	Is there “Undo” function provided, or can user cancel the former operation which can cause serious consequence?	☐ ☐ ☐	
3.4	Are menus on the toolbar broad (many items on a menu) rather than deep (many menu levels)?	☐ ☐ ☐	
3.5	When manipulating sensitive data (e.g. delete or add the fraud web page name from black list), can any user have the access?	☐ ☐ ☐	
3.6	Can users set their own preferred layout of toolbar?	☐ ☐ ☐	
3.7	Can users set their own preferred warning methods?	☐ ☐ ☐	

4. Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

#	Review Checklist	Yes No N/A	Comments
4.1	Are the locations of buttons on toolbar in the browser fixed and consistent?	☐ ☐ ☐	

4.2	Is the terminology used consistently?	☐ ☐ ☐	
4.3	Are icons labeled clearly?	☐ ☐ ☐	
4.4	Are the titles of popups consistent?	☐ ☐ ☐	
4.5	Is color tone used on toolbar consistent with browser?	☐ ☐ ☐	
4.6	Does the menu structure and buttons layout match the task structure?	☐ ☐ ☐	
4.7	Do online instructions appear in the consistent place?	☐ ☐ ☐	
4.8	Do toolbar error messages follow hosted browsers' message standards?	☐ ☐ ☐	
4.9	Do toolbar warnings follow hosted browsers' warning standards?	☐ ☐ ☐	
4.10	Are attention-getting techniques used with care?	☐ ☐ ☐	
4.11	Is the attention-getting technique used only for warning of the fraud detection or also for exceptional conditions?	☐ ☐ ☐	
4.12	Is the most important information placed at the beginning of the prompt?	☐ ☐ ☐	
4.13	Are advice for users named consistently across all prompts in the toolbar, no matter what the risk level of warning is?	☐ ☐ ☐	
4.14	Does the structure of menu items' names match their corresponding contents?	☐ ☐ ☐	
4.15	Do abbreviations follow a simple primary rule?	☐ ☐ ☐	

5. Help Users Recognize, Diagnose, and Recover From Errors

Error messages should be expressed in plain language.

#	Review Checklist	Yes	No	N/A	Comments
5.1	If toolbar can not verify the legitimacy of web pages, is user kept informed what user could do to check it manually?	☐	☐	☐	
5.2	Is sound used to signal an error?	☐	☐	☐	
5.3	Are popups brief and unambiguous?	☐	☐	☐	
5.4	Are error messages worded so that the toolbar, not the user, takes the blame?	☐	☐	☐	
5.5	Do prompts imply that the user is in control?	☐	☐	☐	
5.6	Are error messages and warnings correct in grammar?	☐	☐	☐	
5.7	Is wording in error messages and warnings in good manner or politely?	☐	☐	☐	
5.8	Are warnings able to show the origin of error?	☐	☐	☐	
5.9	Do warnings inform the user of the error's severity?	☐	☐	☐	
5.10	Do warnings inform the user how to correct the error?	☐	☐	☐	

6. Error Prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

#	Review Checklist	Yes	No	N/A	Comments
6.1	Are menu items logical, distinctive, and mutually exclusive?	☐	☐	☐	
6.2	Are data inputs case-blind whenever possible?	☐	☐	☐	
6.3	Are data inputs type-sensitive?	☐	☐	☐	

6.4	Does the toolbar alert users if they are about to make a potentially serious error?	☐ ☐ ☐	
6.5	Do fields in data entry screens and dialog boxes contain default values when appropriate?	☐ ☐ ☐	
6.6	Is there help or instruction for data inputs?	☐ ☐ ☐	

7. Recognition Rather Than Recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of toolbar should be visible or easily retrievable whenever appropriate.

#	Review Checklist	Yes No N/A	Comments
7.1	For question and answer interfaces, are visual cues and white space used to distinguish questions, prompts, instructions, and user input?	☐ ☐ ☐	
7.2	Are prompts, cues, and messages placed where the eye is likely to be looking on the screen?	☐ ☐ ☐	
7.3	Have warning or error messages been formatted using white space, justification, and visual cues for easy scanning?	☐ ☐ ☐	
7.4	Is it easy to find necessary operation for checking the legitimacy?	☐ ☐ ☐	
7.5	Does the toolbar gray out or delete labels of currently inactive functions?	☐ ☐ ☐	
7.6	Have items on a dialog or popup been grouped into logical	☐ ☐ ☐	

	zones, and have headings been used to distinguish between zones?		
7.7	Are significant button or menu item groups identified and highlighted?	☐ ☐ ☐	
7.8	Does the toolbar provide mapping: that is, are the relationships between controls and actions apparent to the user?	☐ ☐ ☐	
7.9	Are inactive menu items grayed out or omitted?	☐ ☐ ☐	
7.10	Are the optional and non-optional settings of toolbar distinguished?	☐ ☐ ☐	
7.10	Is it easy to find place for changing toolbar related settings?	☐ ☐ ☐	

8. Flexibility and Minimalist Design

Accelerators-unseen by the novice user-may often speed up the interaction for the expert user such that the toolbar can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. Provide alternative means of access and operation for users who differ from the "average" user (e.g., physical or cognitive ability, culture, language, etc.)

#	Review Checklist	Yes No N/A	Comments
8.1	If toolbar supports both new and experienced users, are multiple levels of warning detail available?	☐ ☐ ☐	
8.2	Can user customize the filter settings of toolbar?	☐ ☐ ☐	
8.3	Can user customize the layout of toolbar?	☐ ☐ ☐	
8.4	If menu lists are short (seven items or fewer), or there are limited buttons (no more than 10) on the toolbar , can users select an item or button by moving the cursor?	☐ ☐ ☐	

8.5	Do users have the option of either clicking directly on a field (e.g. menu item, input field, and dialog box)or using a keyboard shortcut?	☐ ☐ ☐	
8.6	Can users set their own default operation for the detected fraud?	☐ ☐ ☐	
8.7	Can users nullify their default operation for the detected fraud?	☐ ☐ ☐	
8.8	If users set their own default alert level too low, is there any warning or reminding for this?	☐ ☐ ☐	

9. Aesthetic and Minimalist Design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

#	Review Checklist	Yes No N/A	Comments
9.1	Is only (and all) information essential to decision making displayed on the screen?	☐ ☐ ☐	
9.2	Are meaningful groups separated properly?	☐ ☐ ☐	
9.3	Does each group have meaningful title?	☐ ☐ ☐	
9.4	Are menu items' titles brief, yet long enough to communicate?	☐ ☐ ☐	
9.5	Do warnings concisely and correctly show the most important information about phishing detection?	☐ ☐ ☐	
9.6	Can users be misled to other websites not related to phishing and its prevention?	☐ ☐ ☐	

10. Help and Documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

#	Review Checklist	Yes	No	N/A	Comments
10.1	Is there any user manual, tutorial, or help documentation available?	☐	☐	☐	
10.2	Is there online help available?	☐	☐	☐	
10.3	Is there instant help available?	☐	☐	☐	
10.4	Is there necessary report to toolbar's provider available, when current version of toolbar can not successfully judge the web page?	☐	☐	☐	
10.5	Do instructions or manuals follow the sequence of user actions?	☐	☐	☐	
10.6	Are instructions and other help documentations understandable?	☐	☐	☐	
10.7	Is it easy to search what user wants to know from help documentation?	☐	☐	☐	
10.8	Is the help function visible and easy to find?	☐	☐	☐	
10.9	Does the terminology of help documentations follow the toolbar general design conventions and standards.	☐	☐	☐	
10.10	Is there context-sensitive help?	☐	☐	☐	
10.11	Is it easy to access and return from the help system?	☐	☐	☐	
10.12	Can users resume work where they left off after accessing help?	☐	☐	☐	
10.13	Is the help detailed enough?	☐	☐	☐	

10.14	Is there report to toolbar's provider available, when user can't find answer from existing help documentations?	☐ ☐ ☐	

11. Skills

The toolbar should support, extend, supplement, or enhance the user's skills, background knowledge of anti-phishing ----not replace them.

#	Review Checklist	Yes No N/A	Comments
11.1	Can users learn from toolbar's functions or documentations what is phishing?	☐ ☐ ☐	
11.2	Can users learn from toolbar's functions or documentations what are common phishing techniques?	☐ ☐ ☐	
11.3	Can users learn from toolbar's functions or documentations how to identify the fraud WITH toolbar?	☐ ☐ ☐	
11.4	Can users learn from toolbar's functions or documentations how to prevent the fraud WITHOUT toolbar?	☐ ☐ ☐	
11.5	Can users learn from toolbar's functions or documentations how to protect their personal, or confidential information?	☐ ☐ ☐	
11.6	Are there daily, or weekly phishing reports or news?	☐ ☐ ☐	
11.7	Can users be informed about serious consequences, if users fail to follow the expected security advice?	☐ ☐ ☐	

12. Pleasurable and Respectful Interaction with the User

The user's interactions with the toolbar should enhance the quality of her or his browsing experience. The user should be treated with

respect. The design should be aesthetically pleasing- with artistic as well as functional value.

#	Review Checklist	Yes	No	N/A	Comments
12.1	Is each warning labeled properly in terms of its severity?	☐	☐	☐	
12.2	Can warning draw users' attention?	☐	☐	☐	
12.3	Is warning too overwhelming to disturb users' pleasant browsing?	☐	☐	☐	
12.4	Are there many false and undetermined detection warnings?	☐	☐	☐	
12.5	Are there any settings to simplify users' phishing detection sequences?	☐	☐	☐	
12.6	Are the frequently used function or button put in the most accessible position?	☐	☐	☐	

13. Privacy

The toolbar should help the user protect any personal, private or sensitive information- belonging to the user.

#	Review Checklist	Yes	No	N/A	Comments
13.1	Are protected areas completely inaccessible?	☐	☐	☐	
13.2	Can protected or confidential areas be accessed with certain passwords?	☐	☐	☐	
13.3	Is this feature effective and successful, or is the toolbar provider reliable enough to protect all of users' personal information submitted? (e.g. email, telephone number, etc.)	☐	☐	☐	

Appendix 2

Toolbar details

Toolbar	***Version number***	***Download date***
Google Safe Browsing	N/A	Dec. 13th 2006
Netcraft toolbar	1.7.0 (20061016)	Dec. 13th 2006
SpoofGuard	N/A	Dec. 13th 2006
Anti-phishing IEPlug	N/A	Dec. 10th 2006

Evaluation environment

Hardware	***CPU***	Pentium III, 800 MHz
	Memory	512 MB
	Monitors	2 monitors, resolution 1024x768, 32 bit color
	Keyboard	US-International English keyboard
Software	***Operating System***	Windows XP, SP2
	Anti-malware product	F-secure Anti-Virus Client Security 5.58
	Internet Explorer	IE 6.0, SP2
	Firefox	Firefox 2.0