Trusted e-Commerce User Agent Based on USB Key

Dawei Zhang, Peng Hu

***Abstract*—With the development of e-commerce, the client-side security becomes more and more important. In order to assure that the transactions behave as what users expect, be aware of and approved, a trusted e-commerce user agent based on USB Key and Java Card platform is designed and implemented in this paper. E-commerce providers can develop and load user agent applets on this framework to execute authentication and establish secure channel between server and client, etc. At fist, design principles are given based on analysis of known attacks today. Secondly, the software architecture and security features of this user agent are discussed in detail, which includes how to implement HTTP server, CGI, security agent framework and applet load on USB Key. Thirdly, SSL application scenarios based on user agent and security analysis shows that this scheme can effectively improve the client-side security in e-commerce. At last, advantages of this user agent and what to do further are given in conclusion and future work.**

***Index Terms*—e-commerce, Java Card, Smart Card, User Agent**

I. Introduction

With the development of the Internet, more and more people are surfing the Internet for day-to-day activities, from shopping, banking and paying bills to consuming media and entertainment. But as the value of what people do online has increased, the internet itself has become more complex and dangerous. On line identity theft, fraud, security and privacy concerns are on the rise [1]. Phishing is one of those criminal activities on the Internet. Phishing is online identity theft in which confidential information is obtained from an individual [2].Data suggest that some phishing attacks have convinced up to 5% of their recipients to provide sensitive information to spoofed websites [3]. About two million users gave information to spoofed websites resulting in direct losses of $1.2 billion for U.S. banks and card issuers in 2003 [4]. 25624 unique phishing attacks were submitted to APWG in August 2007 [5].

Several papers have summarized the phishing attacks until now [2][6][7]. We will discuss the followings which will threaten the security of e-commerce seriously:

Maleware-based phishing:

Malware-based phishing refers generally to any type of phishing that involves running malicious software on the user's machine. Malware-based phishing can take many forms [2]:

Dawei Zhang is with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044 China (e-mail: dwzhang@ bjtu.edu.cn).

Peng Hu is with the Department of Products, Beijing Watchdata system Co. Ltd , Chaoyang District, Beijing 100015 China (e-mail: peng.hu@watchdata. com).

- **Keyloggers and Screenloggers**: Keyloggers will monitor data being input and send relevant data to a phishing server. A screenlogger monitors both the user's inputs and the display on screen.
- **Session Hijackers:** Session hijacking refers to an attack in which a user's activities are monitored, typically by a malicious browser component, which will "hijacks" the session to perform malicious actions once the user log in.
- **Data Theft:** Once malicious code is running on a user's computer, it can directly steal confidential information stored on the computer.

Why can malware-based phishing succeed? The answer is that the user's computer is not a *secure and trusted* computing environment.

Furthermore, there are other phishing attacks which often occur on the communication channel:

DNS-Based Phishing (Pharming):

DNS-based phishing refers the phishing that interferes with the integrity of the lookup for a domain name.

Man-in-the-Middle (MITM) Phishing:

A man-in-the-middle attack is a form of phishing in which the phisher positions himself between the user and the legitimate site. Man-in-the-middle attacks can also be used for session hijacking.

At last, a kind of phishing attack, which will occur on the real endpoint of transaction system – the user, will be discussed as follows:

Web spoofing:

In their seminal work on Web spoofing, Felten et al [8] showed how, in 1997, a malicious server could forge some browser cues, such as location bar information, SSL icons, SSL warnings, certificate information, and response time. Subsequent papers also researched this issue, Ye et al [7] gave a working definition of Web spoofing: When malicious action causes the reality of the browsing session to differ significantly from the mental model a reasonably sophisticated user has of that session.

Why can pharming, MITM phishing and Web spoofing succeed? The answer is that users themselves can not really authenticate the server by Web browsers even though there is a PKI certificate for server. Some properties concerned with anti-phishing must be taken into consideration here:

The unmotivated user property: Security is usually a secondary goal [8]. Most users prefer to focus on their transaction rather than the certificate of server.

The limited human skills property: Humans are not general purpose computers and limited by their inherent skills and abilities. Rather than only approaching a problem from a traditional cryptography-based security framework (e.g., "what can we secure?"), a usable design must take into account what humans do well and what they do not do well

[10].

A lot of anti-phishing work has been done recently. We summarized some of those proposals here:

Enhancement of browsers' security usability

Many proposals for stopping phishing attacks, such as SpoofStick [11], Trustbar [12] and eBay's Account Guard [13], rely on a *security toolbar* that displays warnings or security-related information in the web browser's interface.

Wu et al [14] conducted two user studies of security toolbars and other browser security indicators and found them all ineffective at preventing phishing attacks. Furthermore, they are vulnerable to Web spoofing because security toolbar can be forged by Web graphic components.

Ye et al [14] proposed "Synchronized Random Dynamic Boundaries" to secure the path from users to their browser. A weakness of this approach is that it ignores the "limited human skills" property and the security depends on how many border frequency options are available and how many users can differentiate [10].

R. Dhamija and J.D. Tygar [10] proposed "Dynamic Security Skins" scheme. In this scheme, they design the browser extension provides a trusted window with a specific photographic image in the browser and users can authenticate content from the server by images match.

The research [10] and [14] are all based on the open source Mozilla Browser and they all need to make changes on the source code, which is not very convenient for common users. Both two can resist Web spoofing but are all vulnerable to malware-based phishing.

Therefore, in our opinion, enhancement of browsers' security usability is not enough for secure e-commerce nowadays.

PKI and Multi-factor authentication

Since Netscape introduced the Secure Socket Layer (SSL v2) protocol based on PKI (Public Key Infrastructure) in 1995, the protocol and its successors, SSLv3 and TLS, have been touted to consumers as a safe and secure means for conducting web commerce. Users can authenticate servers by their certificates and vice-versa. At the same time, multi-factor authentication [15] combined with PKI become more and more popular especially in e-banking applications. As far as the e-banking in China is concerned, most of sophisticated users will choose the USB Key as a hardware security token, which will store private keys, client certificates and execute crypto operations [16]. Users must insert USB Key on the machine's USB port and verify their password for USB Key at first when they log in e-banking and do transactions. This application scenario is a classic two-factor authentication:

What you know: users' password for e-commerce

What you have: USB Key purchased from banks

It can resist some of malware-based phishing attacks, such as keyloggers and screenloggers, data theft because hijackers can not get the USB Key although they got the password.

Is it really secure for common e-commerce users?

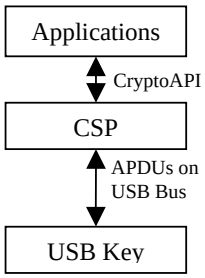

Figure 1. Work flow of USB Key call

Before analyzing the security vulnerabilities of this scheme, we consider the basic work theory of call between applications and USB Key in Windows at first. The applications of Web service (e.g. IE, Outlook) will call the CSP by cryptoAPI and then the CSP will send APDU (Application Protocol Data Unit) commands to USB Key. When USB Key sends the response the CSP will return it to applications. This work flow is shown in Fig. 1.

Based on this work flow, Marchesini et al [17] leverage the "dll proxy" method [18] to hijack calls from applications (such as IE) to CryptoAPI. Therefore they can use the USB Key, for example sign arbitrary data with user's private key, even after the IE has been shut down. Moreover, malware on users' machine can hijack the APDU communication on USB Bus. In other words, the current model of USB Key usage is vulnerable to session hijack of malware-based phishing. The reason is that USB Key in this scenario is a passive device and "behind" the Web browser. USB Key is a secure and trusted computing environment but the calls to it on users' machine is not.

Furthermore, the security of SSL in this application scenario should be carefully considered, too. It is impossible for security designers to expect most of users to check the certificate chain carefully because of *unmotivated user property*. Therefore the man-in-the-middle attacks on SSL, which include technology and social engineering issues, can succeed [19] [20]. An even greater risk is posed by unprotected systems where an attacker can preload his/her own trusted root authority certificates. In public environments such as libraries and computer labs, there is little to prevent such an attack from taking place [19]. It is more interesting and dangerous that Web spoofing is feasible in this scenario because of *limited human skills property*.

In summary, it is very necessary to improve the client-side security further to secure e-commerce. Security system designers must assure that the transactions behave as what users expect, be aware of and approved.

II. Principles of Our Solution

The objective of our solution is to efficiently improve the client-side security and usability in e-commerce so that users can make sure that the transactions behave as what they expect, be aware of and approved.

Our solution is to provide a trusted user agent for e-commerce to improve the client-side security. Therefore we will define the concept of user agent at first. The user agent is the software system that is intended to serve and protect the interests of the user [23]. In traditional way, Web

browser is the user agent for e-commerce but not secure as mentioned above. We will adopt USB Key (hardware and software on it) as a trusted user agent for e-commerce in this paper.

Based on analysis of security threats in section 1 and some important usability studied of security applications [21] [22] [23], the design principles of our solution are given as follows:

- **Multi-factor authentication**: In order to authenticate users' identity and assure the security of e-commerce transaction, we select USB Key, which can store certificate, private key and sign transaction data, as the hard token for users.
- **Trusted path between users and transactions:** The users must have an unspoofable and incorruptible channel to do transactions with trusted remote servers. We will construct a trusted and secure channel for end users by trusted user agent.
- **Path of Least Resistance:** To the greatest extent possible, the natural way to do any task should also be the secure way [23]. Our solution expects to make users do as what they do on paper transactions as much as possible. At the same time, it is very convenient that our solution need not modify user machines' configurations.
- **Customized Solution:** In our opinion, as far as e-commerce, such as e-banking, e-shopping, is concerned, customized solutions are better than general solutions because they can provide more detailed security information and countermeasures.

III. Overview of Our Solution

In this paper, a new idea of trusted user agent based on USB Key is given.

The hardware platform of USB Key consists of a security chip, a flash disk and trusted input and output devices – buttons and screen. The core software, which will be discussed in the next section, will run on the security chip. The flash disk will store programs running in PC OS so that users need not to install any additional software. The screen is used to display sensitive data (such as security status information, transaction amount etc.) and users press the button when they approve some actions. Only the end users and applets on USB Key can access and control those screen and buttons, malware on users' machine can not access screen and simulate the signals of pressing buttons on Key. Therefore a trusted path between users and user agent is established. The appearance of USB Key is shown in fig. 2.

The basis of software platform is a Java Card Platform on USB Key. Based on this platform Web Server and development framework of security agent are implemented in this paper, which will be discussed in the next section. Therefore users can browser the content on the USB Key with the Web browser. E-commerce providers can develop and deploy customized secure user agent applet, which is an applet of Java Card, on this framework. In fact, USB Key with customized applet becomes a secure user agent that is a network proxy with special security functionalities. Users can use it to finish mutual authentication, establishment of secure channel, e-transactions between remote server and client. In other words, USB Key become an active device and "before" the Web browser in this scenario. All of the security – related operations, e.g. checking certificate chain, key exchange, derivation of session key in SSL, encryption and decryption of data etc., will be done on USB Key. This scheme can resist session hijack of malware-based phishing effectively. The application scheme of user agent is shown in fig. 3.

Furthermore, e-commerce providers can design a more natural and secure way of doing e-transactions with customized user agent applets on USB Key. Therefore e-commerce providers are able to provide more secure, flexible and convenient e-transaction manner. When doing e-transaction, users check the transaction information on Key screen just like they check the receipts of paper transactions and they will accept or reject this transaction by pressing buttons on Key. Because of trusted input and output devices, user agent can assure 'What they see is what they sign with their approval'. Therefore a trusted path between users and transactions is established by user agent based on USB Key.

Our design and implementation of this user agent will be discussed in the following sections. The software architecture of this user agent is presented in section 2. Application scenario and security analysis are presented in section 3 and finally the conclusion and future work.

IV. Software Architecture of Trusted User Agent

The main modules in trusted user agent include:

- **Web server on USB Key**: Our solution implemented an embedded Web sever on USB Key, which supports HTTP sever, CGI functions.
- **Security Agent Framework:** Our solution implemented a development framework of security agent on USB Key for secure communication between remote server and client machine by user agent. After e-commerce providers distributed their security applets on this framework, customized authentication and secure transaction schemes can be implemented.
- **Applets Load:** Applets developed by security designers can be remote loaded onto USB Key by the Internet and Browser so that the e-commerce providers can distribute security agent or security patches on line.

Figure 2. Appearance of our USB Key

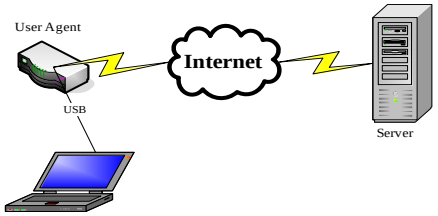

Figure 3. Application Scheme of User Agent

Recent proposals have been made to connect USB key (or smart card) to the Internet [24] [25] [26]. There are some differences between our scheme and those proposals:

1) The goal of our design is to provide a trusted user agent for e-commerce, which will supports third party's security agent development and distribution but those proposals emphasize to provide a Web – enabled device. The fact is that an open security infrastructure is a trend on the Internet [27].
2) The full TCP/IP and HTTP will be implemented in security chips in those proposals but the precondition is a high performance chip platform [3]. The cost is high and the HTTP performance is not very high. In this paper, the security chip is the 8-bit 8051 architecture chip with 6 K RAM and 64 K EEPROM and two layers architecture is deployed on our platform. This scheme will improve the whole performance and lower cost.

There are two layers of software in our solution. At first, a communication agent that gets HTTP data from TCP/IP connection will stay in the FLASH disk on USB Key and run in PC operating system (e.g. windows XP). This agent is responsible for establishing TCP connect, encapsulate HTTP data into APDU and transmit to the USB Key by USB Port. Secondly, a HTTP Server and security agent framework in Java are implemented on the USB Key security chip.

The advantages of this scheme are as follows:

1) All the procedures of HTTP and security agent will be done on USB Key with Java Card platform, which is a secure computing environment [28][29]. Therefore, system security is guaranteed.
2) The communication agent in FLASH disk will run automatically (auto run mechanism provided by operating system) when the USB Key is inserted on USB port. So it is convenient to use and need not to install any additional software on operating system.

The communication agent is simple in this system and unnecessary to be described further in this paper. So we will discuss the design architecture of software on USB Key security chip in the following sections, which includes request switch, HTTP server, CGI, security agent framework and applet load components. All of those components are implemented in Java language on Java Card platform. The architecture is depicted in fig. 4.

A. Request Switch Component

The first component on USB Key that will process the HTTP data encapsulated in APDU is Request Switch. At first this component will get the HTTP request data from APDU, then parse the command and switch it to the appropriate components.

All of those methods are encapsulated in a class – *RequestSwitchObject*, which can be provided for application development.

B. HTTP Server Component

HTTP server component will manage the Web page data, parse HTTP protocol and construct response data. It consists of two sub-components.

The first sub-component is a Web data management component. All of the Web - related data will be stored in

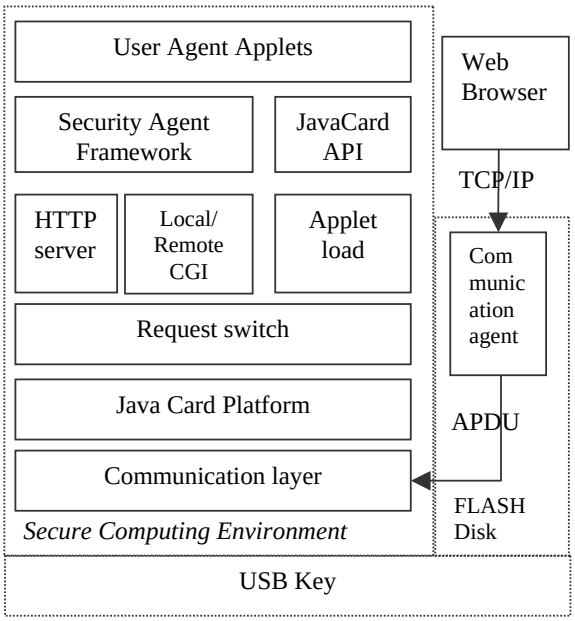

Figure 4. Software Architecture on Security Chip

WebObject object in USB Key persistent memory. The relationship between a Web page and WebObject is:

Web Page = { x | x is a WebObject }

At the same time, there is an index table to indicate the map between PC file and WebObject. The reason to maintain this table is that the source identifier in Web page is based on PC file system.

Another sub-component is a HTTP parser. This component will parse HTTP data from request switch component and finish the corresponding operations according to HTTP method. We implement this component in a class – HTTPparser.

The USB Key can be visited by HTTP through Web browser based on this component. Furthermore, more HTTP–related applets can be built on those classes by developers.

C. CGI Component

CGI (Common Gateway Interface) is supported on USB Key middleware. Two approaches to implement CGI are used in our design.

The first approach is to implement CGI in terms of APIs (methods) in HTTP server component. In other words, CGI programs run in the same execution context [26] as HTTP server. The advantage of this approach is to eliminate context switch between different applets and improve the running speed. The disadvantage is that HTTP server and CGI programs execute in the same context, which is not secure. Any bug in CGI programs coded by third party can negatively impact the entire server, including URL requests have nothing to do with the bug-containing programs. Therefore we only take this approach to implement some simple, USB Key specific CGI functions, such as verify user PIN, generate SHA-1digest of message etc. We call is as local CGI, which is part of HTTP server.

In the second approach, we implement CGI programs as Java Card applets with SIO (Shareable Interface Object) [30].

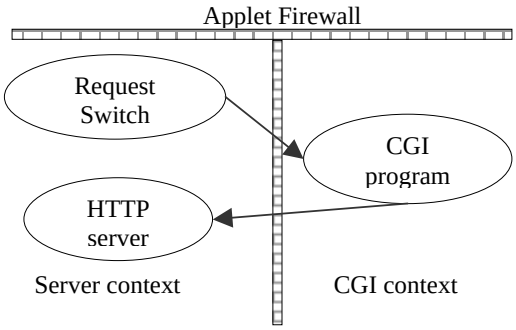

Figure 5. Remote CGI call principles

The request switch component received the CGI request and then will call this CGI program by SIO. At the same time, execution context will switch to CGI programs and switch back to HTTP server (in fact, request switch and HTTP server share the same context in our design) after CGI execution. There is an applet firewall to protect runtime environment between different contexts on Java Card platform [29]. Therefore any bug in CGI programs can not impact entire server because they run in different context. We think this approach is more secure than the first approach. Any CGI programs developed by third party on our framework can be loaded into USB Key on line at any time, which is an open, scalable and secure architecture. We call it as remote CGI. Remote CGI principles are depicted in fig. 5.

D. Security Agent Framework

It is not difficult to implement security agent framework based on components mentioned above.

At first, the communication agent is designed to be able to establish two connections at the same time, one is to local browser and another is to remote server which the agent wants to reach.

Secondly, the security agent framework is implemented as a remote CGI call for security considerations.

The SIO of agent defined as follows:

```
public interface securityAgentInterface extends Shareable
 {
  void securityProcess (byte[] buffer);
 }
the base class of this framework defined as follows:
public class secrityAgentClass extends Applet
implements securityProxyInterface
 {
public Shareable getShareableInterfaceObject(AID clientAID, byte parameter) {  return this;  }
public static void install(byte[] bArray, short bOffset, byte bLength)  {…}
public  void securityProcess (byte[] buffer) {…}
protected final  void waitForUserAccept()  {…}
protected  void sendData()  { Send data to remote server; }
protected  void receiveData() { Receive data from remote server; }
 }
```

Developers can extend *securityAgentClass* and override the *securityProcess()* method according to the specific security scheme(e.g. SSL,TLS) between client and remote server.

Another important method is *protected final boolean WaitForUserAccept ()*. We implemented this method as follows:

1) Display a Web page in Web browser to prompt the user to press accept button on Key.
2) Wait for the user to press button (detect the interrupt signal of pressing button in Key). If not, goto 2.
3) If accept button is pressed, return true else return false.

When developers install a security agent applet with corresponding URL in parameters buffer, the *install()* method implemented by us will bind this URL to the applet in agent register table. While the end user visit this URL, the security agent component will look for in register table and call the corresponding applet's *securityProcess()* to establish security connection.

E. Applet Load Component

Applet load component can receive the program code of Java applets on line and install or update (if there is an old version of this program) it on USB Key.

At first, the Java Card Applet code will be verified by verifier, which is stored in FLASH disk and called by communication agent at first.

Secondly, the Applet code data will be converted to GP(Global Platform) download command sequences by this component and then be installed according to GP specification (with secure channel) [31].

Because of security considerations, this component is not open for user agent application development.

V. Application Scenario Based on User Agent

A. Trusted User Agent with SSL

In this section, we will describe an application scenario for SSL with mutual authentication.

At first, the user certificate, sever certificate and user's private key will be stored on Key. The customized user agent will support SSL protocol and compare the received server certificate with stored server certificate.

Secondly, the e-commerce provider link will be stored in home page of USB Key.

Thirdly, during transaction procedure, the transaction amount, account will be displayed on screen and wait for users' approval.

The *securityProcess()* user agent applet can be described as follows:

```
securityProcess()
{
Send ClientHello to start SSL connection;
Receive SeverHello and get Certificate of Server;
Compare received Certificate with stored Certificate of server,
If  (Certificate is illegal) {
Send detailed alarm information to users' browser;
End this session;
   }
Send Certificate of client;
Establish SSL channel and transfer application data;
```

```
If ( doing transaction is required) {
   Extract transaction information and display it on Key
   screen;
   If ( waitForUserAccept() )
      Do transaction;
   Else
   Abort transaction;
   }
}
```

The work flow of this scenario as follows:

1) The user input password for Key usage. The home page of USB Key showed on browser and the user visits e-commerce Server.
2) *securityProcess()* start. User agent receives the certificate of server and checks it according to the stored certificate in USB Key. If failed, send warning information to user's browser and end this session. If succeeded, continue.
3) User agent finishes the establishment of SSL connection.
4) If the user wants to do transactions, user agent will extract the transaction information and display it on USB Key screen. And then in Web browser prompt the user to check the screen on Key.
5) User checks the transaction information, such as accounts, amount etc. If the user approves the transaction, press the "accept" button. User agent signs the transaction data and sends it to server. Else, press the "Reject" button to abort transaction.
6) When Web pages close user agent ends this session.

B. *Security Analysis of Application Scenario*

As far as known attacks are concerned, the security analysis is given as follows:

1) Web spoofing, Pharming and MIMT attacks can not succeed because the customized authentication of server executes in user agent. The forged certificate of server and security status information can not pass this authentication on USB Key.
2) The malware on users' machine can not attack SSL process, which is done in Key.
3) The malware on users' machine can not hijack the transaction session because the transaction is done on Key with user's approval and the malware can not access and control Key and user.

In summary, the trusted user agent assures that the transactions behave as what users expect, be aware of and approved.

VI. Conclusion and Future Work

A new idea of trusted e-commerce user agent based on USB Key is given in this paper. End users can do secure e-transactions as what they expect, be aware of and approved with this user agent. Some contributions of this technology summarized as follows:

- We made a choice of a design following multi-factor authentication and open infrastructure argument. Developers can build and distribute an open, scalable and secure user agent for e-commerce on USB Key.
- USB Key, which is a passive device and "after" the Web browser in the past, becomes an active device and "before" the Web browser depending on this technology. In this active scheme, we can build a trusted path between users and transactions based on this user agent.
- Customized user agent on USB Key can authenticate the e-commerce servers, establish secure channel with specific cipher algorithms, etc. users can do e-transactions in a more natural and understandable manner. This scheme will improve the client-side security and avoid the problems of unmotivated user and limited human skills property.

The enhancement to this project is still going on. We will optimize communication and storage management so as to improve access speed. Furthermore, we are going to support more application scenarios with different authentication mechanisms on our user agent.

Acknowledgment

Dawei Zhang thanks Dr. Shengguang Li and Dr. Yixin Xu, who participated in the development of HTTP Server. Peng Hu thanks the members of USB Key research group in Beijing Watchdata system Co. Ltd., who developed the application scenario in this project.

References

[1] Microsoft. (2005, May). Microsoft's vision for an identity metasystem [Online]. Available: http://www.identityblog.com/stories/2005/07/05/IdentityMetasystem.htm
[2] Emigh, A. (2005, June).Online identity theft: Phishing technology, chokepoints and countermeasures [Online]. Available: http://www.antiphishing.org/Phishing-dhs-report.pdf
[3] Loftesness, Scott. (2004, August). Responding to "Phishing" attacks [Online]. Available: http://www.glenbrook.com/opinions/phishing.htm
[4] Litan, Avivah. (2004, May). Phishing attack victims likely targets for identity theft [Online]. Available: http://www.gartner.com/resources/120800/120804/phishing_attack.pdf
[5] Anti-Phishing Working Group. (2007, August). Phishing activity trends report for the month of August 2007 [Online]. Available: http://antiphishing.org/
[6] Jason Milletary.(2005, May). Technical trends in Phishing attacks [Online]. Available: http://www.uscert.gov/reading_room/phishing_trends0511.pdf
[7] E. Ye, Y. Yuan, S. Smith.(2002, February). Web spoofing revisited: SSL and beyond [Online]. Available: http://www.cs.dartmouth.edu/~pkilab/demos/spoo_ng/
[8] E. Felten, D. Balfanz, D. Dean, and D. Wallach, "Web spoofing: an Internet con game," 20th National Information Systems Security Conference, 1997.
[9] Alma Whitten, J.D. Tygar, "Why Johnny can't encrypt: a usability evaluation of PGP 5.0," Proceedings of the 8th Usenix Security Symposium, 1999,pp. 169-184.
[10] R. Dhamija, J.D. Tygar, "The battle against Phishing: dynamic security skins", Symposium on Usable Privacy and Security, 2005, pp. 77-88.
[11] eBay. (2005, December). SpoofStick [Online]. Available: http://www.spoofstick.com/
[12] Herzberg, A., Gbara. (2004, September). TrustBar: Protecting (even Naïve) Web users from spoofing and Phishing attacks [Online]. Available: http://www.cs.biu.ac.il/~herzbea/Papers/ecommerce/spoofing.htm
[13] eBay. (2005, January).eBay toolbar and account guard. [Online]. Available: http://pages.ebay.com/help/confidence/account-guard.html

[14] Zishuang Ye, Sean Smith and Denise Anthony, "Trusted paths for browsers", Proceedings of the 11th Usenix Security Symposium, 2002, pp. 121-135.
[15] Technology Administration U.S. Department of Commerce, National Institute Standards and Technology. (2004, June). Electronic Authentication Guideline. [Online]. Available: http://www.cio.gov/eauthentication/documents/SP800-63V6_3_3.pdf
[16] CFCA. (2007, February). 2006 e-banking survey in China [Online]. Available: http://www.cfca.com.cn/
[17] John Marchesini, S.W.Smith, Meiyuan Zhao, "Keyjacking: risks of the current client-side infrastructure," Proceedings of the 2nd Annual PKI Research Workshop, 2003. pp. 80-95.
[18] OS Security, Inc. (2004, February). Round one: "DLL Proxy" attack easily hijacks SSL from Internet Explorer [Online]. Available: http://www.securityfocus.com/archive/1/353203/2004-02-09/2004-02-15/2
[19] Peter Burkholder.(2002, February). SSL Man-in-the-Middle attacks [Online]. Available: http://www.sans.org/reading_room/whitepapers/threats/480.php
[20] Song, Dug. (2000, December). sshmitm, webmitm, http://cert.uni-stuttgart.de/archive/bugtraq/2000/12/msg00285.html,
[21] A. Adams and M. A. Sasse, "Users are not the enemy: why users compromise security mechanisms and how to take remedial measures," Communications of the ACM, December 1999, pp. 40-46.
[22] M. Walker and K.-P. Yee. (1999,January). Interaction design for end-user security [Online]. Available: http://www.cs.berkeley.edu/~pingster/sec/desktop/
[23] Ka-Ping Yee, "User interaction design for secure systems," the 4th International Conference on Information and Communication Security, Singapore, December 2002, pp. 52-66.
[24] Christophe Muller, Eric Deschamps. (2004, March). Smart cards as first-class network citizens [Online]. Available: http://www.gemplus.com/smart/rd/publications/pdf/MD02gdcc.pdf
[25] Henrich C. Pohls, Joachim Posegga, "Smartcard firewalls revisited," CARDIS'06, 2006 August, pp.115-130
[26] Sun Microsystems, Inc. Sun's Brazil research project [Online]. Available: http://research.sun.com/brazil
[27] Microsoft. (2005, May). Thelaws of identity [Online]. Available: http://www.identityblog.com/?page_id=354
[28] Sun Microsystems, Inc. (2006, March). Java Card 2.2.2 Virtual Machine Specification [Online]. Available: http://java.sun.com/products/javacard/specs.html
[29] Sun Microsystems, Inc. (2006, March). Java Card 2.2.2 Runtime Environment Specification [Online]. Available: http://java.sun.com/products/javacard/specs.html
[30] Zhiqun Chen, Java Card technology for smart cards: architecture and programmer's guide, Boston: Addison Wesley, 2000, pp. 70-88.
[31] Global platform organization. (2006, March). GlobalPlatform card specification V2.2 [Online]. Available: http://www.globalplatform.org