

# In-the-wire authentication: Protecting client-side critical data fields in secure network transactions.

Mark William Currie, Ziliant Systems, Durban, South Africa, *Member, IEEE*

**Abstract**—Secure Internet services like online banking require a “trusted terminal” on the client-side. However, even where strong client-side security is employed, the client PC is often used for input and output of sensitive information like PINs/passwords, amounts, account numbers, etc. These transactions are therefore vulnerable to manipulation by malware. A method is presented here allowing web users to share small amounts of secret information including passwords and account numbers with a large number of existing Internet services by creating a cryptographically secure trusted path between the web user and the service. The trusted path is created with the support of a hand-held user terminal device “in-the-wire” between the user’s PC and the service thus preventing malware on the user’s PC from manipulating login and other sensitive data. A key feature is that the trusted terminal device can be retrofitted on the client-side and require no changes to the server-side. This creates a new class of client-centric communications security hardware allowing web users to protect their transactions using strong hardware security without relying on service providers. It offers the industry an alternative to the current service-centric approach which is often hamstrung by a chicken-and-egg problem of critical mass adoption.

**Index Terms**—TLS security, Computer network security, Internet security, Public key cryptography, Man-in-the-middle.

## I. INTRODUCTION

ANY countries have legislation already in place (or pending) that enforces liability on companies holding personal information on their clients. This forces service providers to invest in strong information security infrastructure for the protection of their client information. If the company provides an online service allowing access to their client’s personal information, then this liability carries through to their web servers. However this also allows service providers to “ring fence” their liability. If the service provider can show that the compromise of a client’s personal information was not caused by any security breach within their “fence”, then it becomes difficult for the client to claim any liability from the service provider. Indeed many banks already have liability disclaimers for client losses that may occur through the use of their online banking services. While

some service providers do a fair job of providing their clients with additional security measures, they do not always have the same incentive as their clients. Their concerns are likely to be more around damage to their image rather than direct financial loss. Therefore it is important that clients of these secure online services have access to means of empowering themselves with strong “client-side” security. Other than anti-virus and firewall software, there is currently very little else that web users can do to improve the security of their Internet transactions. On the Internet-side, users can be spoofed by fake websites and on the web user’s side, any user input and output (I/O) done on their PC is vulnerable to manipulation by PC malware.

We may never be able to prevent PC malware infections but we can detect website spoofing and we can prevent malware from accessing those small critical data that we rely on for the protection of our identity and finances. This paper describes a method and means of allowing ordinary web users to protect their highly sensitive information such as PINs, passwords and account numbers from online and local PC malware within the existing infrastructure and without relying on service providers.

## II. INTERNET SECURITY

While the system presented here generally applies to all client-server security networks, most of our attention is focused in the Internet context. Virtually all Internet service providers that handle sensitive data use the ubiquitous TLS [1] cryptographic security protocol. The initial TLS protocol sets up the secret message encryption keys between the user’s browser and the server. This is done within a HTTPS [2] session. Since all secure web servers and all web browsers support TLS, web users can enjoy strong protection of transaction messages over the public network. The system, however, is only secure “in-the-wire” between the user PC and the server. The user’s PC and the server machine remain vulnerable. Service providers that derive income from their service can afford to employ strong security on their servers while the average web user cannot. Most web users also do not understand the importance of client-side security. A report measuring browser version statistics [4] shows by way of their “Web Browser Insecurity Iceberg”, that in June 2008 around half of Internet users are not using the most secure version of

their browser. According to the report, this is only the tip of the iceberg. Lying beneath the surface is a much larger user base with outdated vulnerable browser plug-ins. This of course ignores the number of users with vulnerable operating systems without the latest security patches. It is safe to say that most web-attached PC's are vulnerable to being exploited by online attacks. While most computer viruses are simply annoying and disruptive, newer forms of highly sophisticated malware such as the Silent Banker Trojan and the Pinch Trojan [5] target Internet banking users and allow the attackers to obtain your hard-earned cash. These new forms of malware can immediately infect programs that are already running in memory through API Hijacking techniques [6], i.e. they don't even have to modify any disk files directly. They can use the infected program to write to disk, thus avoiding detection by anti-virus software. Malware is now big business [7]. Key loggers [8] and man-in-the-browser software [9] can be purchased online (including support!) [5]. No amount of spending on PC computer security can prevent exploitation using hardware key loggers. Governments and large companies are often easily exploited in this way [10]. Gone are the days of virus programmers trying to prove how clever they are – money is what it's all about now. The threat to web users is the theft of passwords used to access private services or the manipulation of transactions between the web clients and web services.

#### *A. Internet banking*

With Internet banking, the main vulnerability is the ability to create payment beneficiaries. If fraudsters get access to your online account, they can steal your money by creating a beneficiary and transferring all your money to that account. Some banks require clients to present themselves at the bank to prove their identity before a new online beneficiary is created. However many banks allow the convenience of online beneficiary creation as well as once-off payments. Some of these banks use stronger authentication mechanisms such as one-time passwords. However, virtually none of these measures are effective against a man-in-the-browser attack. A man-in-the-browser attack is just a more localized form of man-in-the-middle (MTM) attack. MTM malware sits somewhere between both ends of a secure link and is able to manipulate the traffic to its advantage. The only effective methods against this form of attack in a PC network require clients to be issued with some form of security hardware device. Even then, the device must be used not only for user authentication but also for complete transaction authentication and that generally requires some inconvenient additional interaction from the user. These forms of protection have not been successful in obtaining wide-scale adoption by banks. They are reluctant to shoulder the cost and do not wish to inconvenience their customers. The cost is much higher than just the purchase of the client devices since the bank effectively now also has to become an IT vendor. A complete infrastructure must be set up covering the full client demographic base. This includes customer awareness

programmes, staff training, device distribution and support. Any negative factors arising from the use of the device – such as “finger trouble”, device unreliability/failure, helpdesk frustrations, etc. – impact directly on the bank's overall service reliability and image. There are also negative factors for clients. From a client perspective, these gadgets are service-specific. A separate device with its own unique idiosyncrasies would have to be used for each secure service. The device can also become a form of client “lock-in” to a particular service provider. Small wonder that these forms of protection have not enjoyed the widespread adoption that one would expect.

#### *B. Service-centric model*

The basic problem is that the current model is service-centric, and unfortunately this is usually the preferred path taken by crypto vendors as most are not consumer product companies. They are often small companies used to selling to service providers, building corporate relationships and securing lucrative support agreements. However they are forced to go this route anyway as their service-centric solutions usually rely on an additional cryptography layer which requires back-end support by the service providers. The system presented in this paper tries to turn this traditional paradigm on its head.

#### *C. Client-centric model*

While there may be many forms of sensitive data on a typical PC – such as financial records, health records and other private information – often the most critical information is actually quite small – such as passwords, identity numbers, credit card numbers etc. These critical data elements also do not tend to change that often and can quite easily be stored, entered and/or displayed on a device outside of the PC environment. However in order for this to work while maintaining a trusted path between the client and service, the external device must implement the communications security function. In the Internet case, this is the browser's TLS function. By controlling the client-side of the TLS cryptographic system, the device can allow clear-text data to be inserted, substituted and viewed without being vulnerable to interception on the client PC. The device can, in effect, provide a direct “in-the-wire” user interface for critical data fields such as passwords and account numbers. By maintaining the common Internet security model in which servers are authenticated using the TLS protocol and clients are authenticated using passwords, a client-centric solution can be realized independent of service providers.

### III. AUTHENTICATION

Thwarting MTM malware requires strong authentication not only of the communicating identities but also of the data communicated between them. Both user directives (input) and user notifications (output) must be authenticated in the most direct way possible. Researchers at IBM published a method of viewing critical data “in-the-wire” [11] based on using a

device which they have called the ZTIC. The ZTIC acts as a TLS proxy during a normal online banking session allowing it to independently validate a website certificate and provide a trusted user display. This solves the user notification (output) authentication problem. However the ZTIC does not solve the user directive (input) authentication problem. The rationale here is that to prevent online banking theft via man-in-the-browser malware it is only necessary to confirm the correct entry of a payment beneficiary on the device display. Without a trusted user-to-server path the ZTIC has to rely on the use of the TLS “client certificate” [1] mode of operation for user authentication. However, this is in fact not user authentication but device authentication, and anyone borrowing/stealing the device can also use it to transfer funds. Even if the server still requires a user password, the password has to be typed in on a web browser where it is vulnerable to interception.

#### A. TLS client certificate

Client certificate mode (or client-side TLS) requires not only services but also clients to obtain X.509 [3] public key certificates. Currently, the vast majority of Internet servers conduct HTTPS sessions based on server-side-only authentication where only the server is required to have a valid certificate. While most servers and browsers supposedly support client certificate operation, this has not been tested on a large scale and there are likely to be many instances of incompatible, incorrect and even insecure implementations. Many schemes have been proposed that require client-side TLS mode but none have ever achieved wide-scale adoption. The reasons for this are the same well documented reasons for the failures of large-scale Public Key Infrastructure (PKI) [12]. When you expand the web’s current server-side PKI model to include client certificates, the problems become exponentially larger. TLS client certificate based solution providers often cite universal compatibility but the fact is that these solutions are still server-centric. A client-centric approach would require the use of Trusted Third Parties (TTP) to issue client certificates. Banks do not like using a Trusted Third Party (TTP) as it would entail complicated liability agreements. Therefore the banks themselves would have to implement broad-based PKI, allowing them to issue certificates to their clients (client “lock-in”). The banks would also have to build a secure facility to initialize the security devices. They would be encumbered with non-core infrastructure required to support the client devices including all the other overheads associated with a server-centric solution discussed earlier.

#### B. Retaining PIN/Password authentication

While PKI has its problems [12], the server-side authentication model currently used on the Internet works reasonably well if we can solve the security problem around certificate storage in the browser. The current PIN/password method of authenticating clients at the application layer will work well if we can prevent interception by malware. It would be far preferable for banks and other online services to retain

the PIN/password method of user authentication. A PIN/password is a direct human-to-service trust mechanism which is not the same as the device-to-service mechanism offered by client-side TLS. Used for remote access control, PINs and passwords are not threatened by password crackers since the service will typically only allow a few bad attempts before permanently locking the user out. PIN’s also fit nicely into a bank’s existing infrastructure. It enables them to encrypt the user’s PIN at the web server using their existing security systems, and from there on treat the transaction in exactly the same way as they would from an ATM or debit card POS

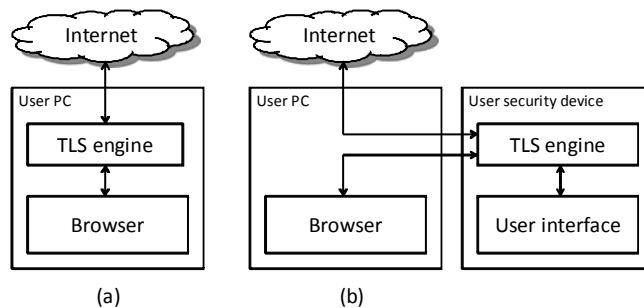


Fig. 1. Secure UI integration concept. (a) Normal configuration. (b) Configured with security device.

terminal.

Strong PIN/password authentication requires a trusted path between the user and the service. This can never be achieved if a general purpose PC is used as the only human interface. This, in fact, applies to any form of general purpose computer including PDA’s and mobile phones. A dedicated User Interface (UI) is required with a cryptographically secure path to the service. While there are various implementation approaches, Fig. 1 shows the basic concept of how the secure UI can be integrated into a normal web-attached PC.

#### IV. TLS SECURITY DEVICE

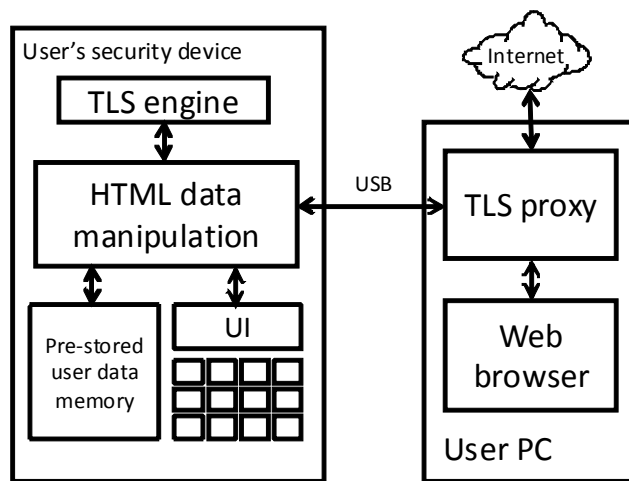


Fig. 2. Secure TLS device.

Fig. 2 describes a trusted terminal device consisting of a display, keypad and a PC connection (e.g. USB). Like the ZTIC it must be capable of handling a full TLS security session. This enables the device to verify server certificates and indicate the validity directly to the user. A TLS proxy on the user's PC could be responsible for routing packets to and from the hardware device. A more PC-independent implementation could be realized where the proxy resides on the device which connects to the PC with an Ethernet interface. A simpler implementation could also be realized where the PC browser's crypto interface (Crypto API) is modified to provide an interface to the security device.

Within the device, HTML data can now be viewed and/or manipulated and substituted with pre-stored or manually typed-in user data. Providing a keypad as well as a display makes it possible for a user to login to a website without disclosing the PIN/password on the user's PC.

There are a few variations to the way that this can be achieved.

#### A. Password substitution (preset)

The first way is that the device parses the web page and recognizes the standard password field in an HTML form – e.g. `<input type="password" name="Password" />`. It then

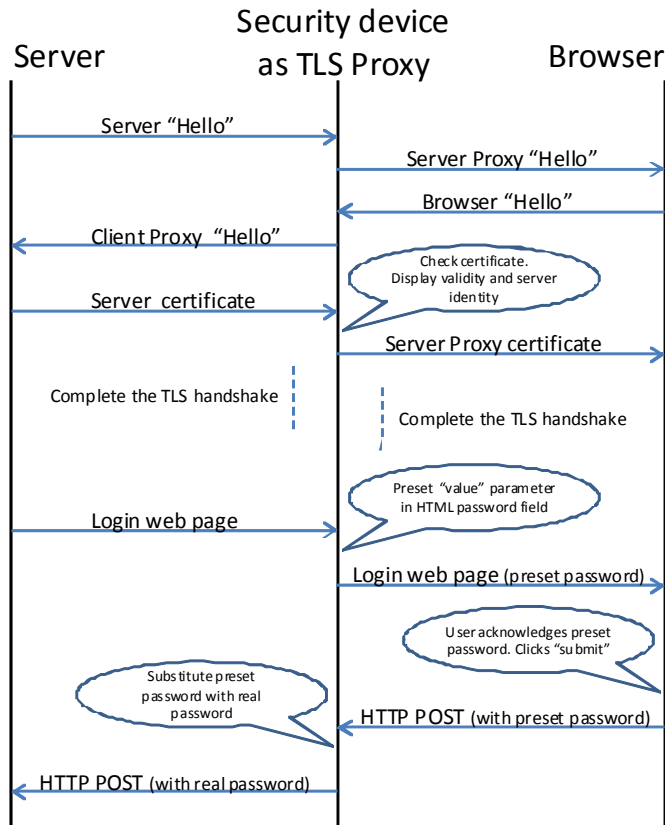


Fig. 3. During a TLS Login session the device validates the server certificate and displays the server identity to the user. After receiving the login web page the device presets the HTML form password field with a random value and then later substitutes it with the real password during the HTTP POST response.

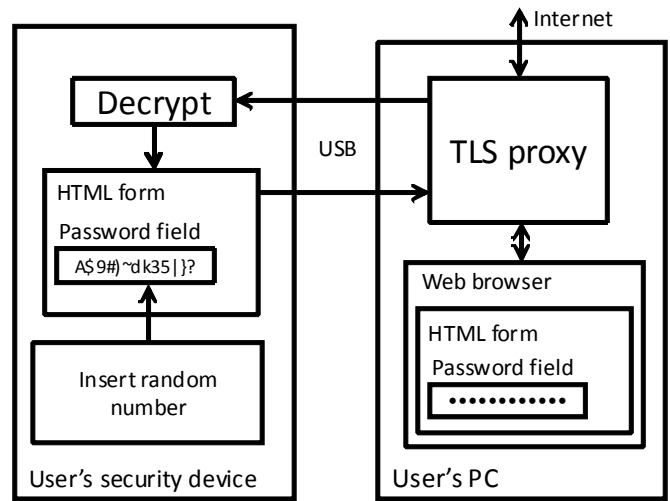


Fig. 4. Password field presetting.

adds a random value into the field – e.g. `<input type="password" name="password" value="A$9#\~d k35|}?" />` before passing the web page through to the user browser (Fig. 3 and Fig. 4).

When the web page is viewed by the user on the PC, the browser is none the wiser and simply shows the password field as being preset, i.e. the password box will show a row of dots (see Fig. 4). The user simply accepts this (e.g. presses Enter) and continues. The browser then sends the web form parameters back to the web server in a HTTP POST message

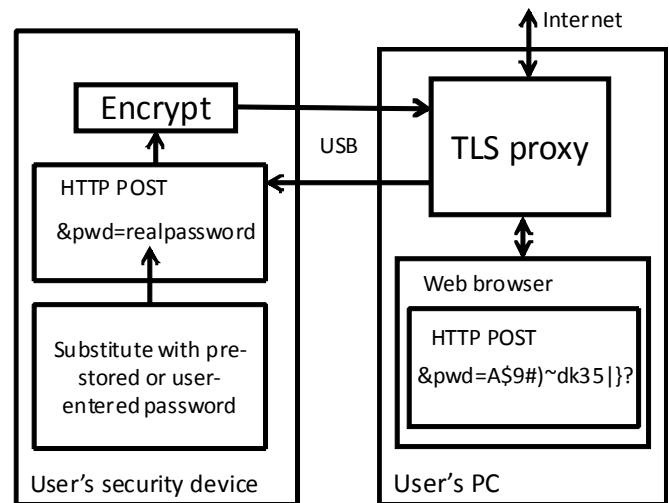


Fig. 5. Password substitution.

and this is intercepted by the user's security device (see Fig. 3 and Fig. 5). The security device then looks for the random number which it originally inserted into the password field. This is now substituted either by a pre-stored user PIN/password or one which the user has been prompted to enter on the device keypad. In the latter case the security device could prompt the user to enter the password when the

HTTP POST message is received.

From the web server's perspective, there is no change, i.e. it is as if the user typed the password directly into the browser web form.

### *B. Password substitution (manual)*

Another variation would be for the device to prompt the user to type a dummy password on the PC into the browser password field. This could be a fixed value known to the device and the user, or it could be a random value displayed to the user by the device. The device would then replace the dummy password with the real password during the substitution phase shown in Figure 4. This method would work for websites that don't use HTML password fields but some other method e.g. a Java applet. As long as the password is not encrypted by the applet, the device will find it in the HTTP POST message during the substitution phase.

### *C. Other data field substitutions*

This login substitution method could also be used for other critical data fields such as an account number, credit card number or a national ID number. For example, a fixed known pattern/phrase could be entered by the user into an account number field in a web form. This pattern/phrase would be found by the device within the resulting HTTP POST message. It can then substitute this value for a user-entered value or a preset stored value before being encrypted and sent back to the server. Critical data fields could also be filtered for viewing only on the device display in the same manner as the ZTIC device. In this case, however, as with the ZTIC, the device would have to use search criteria specific to each website or else the website would have to mark this field in some way within the HTML – e.g. in a comment field.

## V. COMPATIBILITY

The most common method of entering passwords today is still the standard HTML form password field. Therefore the preset method of login substitution described above will work on a large number of secure websites without change. The manual substitution method should have even broader compatibility. The few websites that make use of some complicated obfuscation method for login may have to provide an alternative standard HTML method to allow the device to be used. Presumably, if the device became popular enough, service providers would be only too pleased to offer such an alternative. Indeed web service providers would encourage the use of such a device as it removes any real or perceived liability that they may have for their customer's client-side security. Compatibility is more of an issue in the general purpose use of critical field viewing-in-the-wire. Website-specific search criteria would have to be used to locate the critical data fields within the HTML/HTTP. This is not ideal as it could change when the service updates its website. In these cases, one would have to specify a standard search pattern to be embedded in an HTML comment field within the web page. This is an extremely trivial change and is

unlikely to be met with much resistance from the service providers. The ideal, of course, would be if new HTML form fields could be defined. For instance, it would be useful if a special account number field or a special amount field could be defined that could be recognized by a security device parsing the HTML. The security device could then detect fields requiring trusted confirmation by the user or those requiring local data entry.

## VI. TRUSTED PATH

The number of root Certification Authority (CA) certificates required to validate almost all secure web servers on the net is quite small and could easily be stored on the security device's memory allowing it to validate virtually any server certificate on the net. Unlike the user's PC the user's security device provides a dedicated function, and cannot run any other programs. Therefore it is far more trustworthy. The device checks the validity of the service by verifying its X.509 certificate and then displays the certificate URL. If the TLS proxy resides on the device, the device can compare the actual URL with the one embedded in the certificate and display a simpler "go/no-go" image e.g. green for go, red for no-go. The user can then confirm with a high level of trust that the website is legitimate and not a spoof. Since the encryption keys are derived only between the validated server and the device, the critical user data inserted by the device is cryptographically protected between the device and the server and cannot be extracted by any software on the user's PC. This creates a direct trusted path between the user and the web service. One might argue that this is not strong mutual authentication in the cryptographic sense as there is no client certificate or strong client-side authentication. However, any service that requires strong client authentication would normally issue client certificates and enable client-side TLS anyway. Our technique does not preclude the use of client certificates and this would in fact only serve to strengthen the security, since an attacker would have to steal both client private key and the client PIN/password. In the typical case where the client private key is stored on the client's PC, it can easily be harvested by a Trojan (see [6]) even though it is encrypted. However, our PIN/password and client private key would never be disclosed on the PC. The device could also be designed to only run its program in read-only memory thus protecting it from being hacked due to some error in the program code. Given the limited dedicated functionality, it may be possible to model the behavior of the device under all conditions to pick up any vulnerability. It might, therefore, be possible to make the device provably secure against malware attacks from the user's PC. Note that any crypto system relying on critical UI from a general purpose PC is provably insecure and this applies to most security solutions today.

## VII. OTHER APPLICATIONS

IT administrators often have to access their corporate networks remotely in order to make changes to server

configurations e.g. after hours. This would typically be done through a Remote Access Service (RAS) which may be secured using TLS. This would typically be done using a work laptop or a home PC which may also be used for various other activities including connecting to the Internet. The risk here cannot be understated. If the laptop or PC is exposed to malware infection, the entire corporate network could be compromised. Our security device could be used here not only to protect the corporate network login but also to issue simple configuration commands to a server through the trusted path using the manual substitution method described above. By using similar HTML tunneling techniques described above, additional functionality could be added to the device allowing it to be used in other security applications as well. For instance, a digital signature capability could be included allowing the device to be used in e-Government applications. One could define a limited command set which could be tunneled to the device via HTML comments. This effectively creates a new class of client-centric communications security device, one that can be sold directly to the public in which service providers use to enhance the security of their clients PC web browsing.

### VIII. CONCLUSION

Software-only solutions will never solve our Internet security problems. It's just too easy to infect a PC with malware capable of manipulating user input and output. While we cannot prevent malware infections, we have shown that by using specialized hardware it is possible to prevent online thieves getting at your hard-earned cash without necessarily requiring the cooperation of service providers. Client hardware solutions have been around for many years but even the few that are actually effective rely on service provider buy-in. This has probably been the single biggest obstacle preventing wide-scale deployment of strong Internet security products. A device based on the methods described in this paper can be produced very cost-effectively and can be sold directly to the public. This client-centric approach allows users to have one device that can be used for many security applications. Since the system is seamlessly incorporated into the existing Internet PKI, it saves service providers from investing in additional security infrastructure. On an Internet scale this is a big saving indeed.

The service-centric approach to Internet security has largely failed due to the large investments that must be made by service providers and the chicken-and-egg problem of critical mass adoption. A client-centric solution removes the chicken-and-egg problem allowing the solution to be driven by ordinary people. This is the way that de-facto standards come about and is often the best path to wide-scale adoption. With "cloud" computing becoming more and more prevalent on the Internet, we will see more and more services requiring passwords for the protection of important user data. The client-centric approach empowers users to make use of the services of their choice while providing end-to-end hardware-

strength protection of their passwords and server notifications.

### REFERENCES

- [1] "The Transport Layer (TLS) Security Protocol Version 1.2", IETF, RFC 5246, Aug. 2008.
- [2] "HTTP over TLS", IETF, RFC 2818, May, 2000.
- [3] "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF, RFC 2459, January 1999.
- [4] Stefan Frei, Thomas Duebendorfer, Gunter Ollmann, Martin May, "Understanding the Web browser threat: Examination of vulnerable online Web browser populations and the "insecurity iceberg"", ETH Zurich Tech Report Nr. 288, June 30, 2008.
- [5] ScanSafe US, "Web 2.0 – The New Generation of Web Threats", A ScanSafe White Paper, June 2008.
- [6] John Marchesini, S.W. Smith, Meiyuan Zhao, "Keyjacking: The Surprising Insecurity of Client-side SSL", Dept. of Computer Science Dartmouth College, Technical Report TR2004-489, February 13, 2004.
- [7] Brian Krebs, (2008, December 18) , "Hundreds of Stolen Data Dumps Found", Washington Post [Online]. Available: [http://voices.washingtonpost.com/securityfix/2008/12/hundreds\\_of\\_stolen\\_data\\_dumps.html](http://voices.washingtonpost.com/securityfix/2008/12/hundreds_of_stolen_data_dumps.html).
- [8] Thorsten Holz Markus Engelberth Felix Freiling. (2008, December 18). Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones. University of Mannheim, Laboratory for Dependable System [Online]. Available: <http://honeyblog.org/junkyard/reports/impersonation-attacks-TR.pdf>
- [9] Philipp Gühring, (2006, August 12) , "Concepts against Man-in-the-Browser Attacks" [Online]. Available: <http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf>. Available e-mail: [pg@futureware.at](mailto:pg@futureware.at).
- [10] Adv. Jacqueline Fick, (2009, March), "Cyber Crime in South Africa: Investigating and Prosecuting Cyber Crime and the Benefits of Public-Private Partnerships", PriceWaterhouseCoopers South Africa [Online]. Available: [http://www.coe.int/t/dghl/cooperation/economiccrime/cybercrime/Documents/Reports-Presentations/2079if09pres\\_JaquiFick\\_report.pdf](http://www.coe.int/t/dghl/cooperation/economiccrime/cybercrime/Documents/Reports-Presentations/2079if09pres_JaquiFick_report.pdf)
- [11] Thomas Weigold, Thorsten Kramp, Reto Hermann, Frank Höring, Peter Buhler, Michael Baentsch, "The Zurich Trusted Information Channel – An Efficient Defence against Man-in-the-Middle and Malicious Software Attacks", TRUEST 2008, LNCS 4968, pp.75-91, 2008, Springer-Verlag Berlin Heidelberg 2008.
- [12] Peter Gutman, "PKI: It's Not Dead, Just Resting," Computer, vol. 35, no. 8, pp. 41-49, Aug. 2002, doi:10.1109/MC.2002.1023787